# Prefetchingverfahren für verteilte hypermediale Lernanwendungen

Dissertation
zur
Erlangung des akademischen Grades
doctor rerum politicarum (Dr. rer. pol.)
der Wirtschafts- und Sozialwissenschaftlichen Fakultät
der Universität Rostock

vorgelegt von
Jan Tamm, geboren am 24. September 1972 in Berlin
aus Rostock
Rostock, 1.November 2007

urn:nbn:de:gbv:28-diss2008-0037-8

Gutachter:
Prof. Dr.-Ing. Hans Röck, Lehrstuhl für Wirtschaftsinformatik, Universität Rostock
Prof. Dr.-Ing. Peter Forbrig, Lehrstuhl für Softwaretechnik, Universität Rostock

Datum der Verteidigung: 28.5.2008

# Inhaltsverzeichnis

# Abbildungsverzeichnis

# Tabellenverzeichnis

# Abkürzungsverzeichnis

| | |
|---|---|
| 3*DM* | 3-Dimensionales Matching |
| 2*MLB* | Minimierung der maximalen Latenzzeit aller Knoten des Baums |
| 2*MLS* | Minimierung der maximalen Latenzzeit aller Knoten einer beliebigen Sequenz |
| CGI | Common Gateway Interface |
| CPU | Central Processing Unit |
| DNS | Domain Name Service |
| DSL | Digital Subscriber Line |
| *ERL* | Einfaches „redundante“ Ladeproblem |
| GP | Ganzzahliges Programm |
| HTML | Hypertext Markup Language |
| HTTP | Hypertext Transfer Protokoll |
| ISDN | Integrated Services Digital Network |
| ISP | Internet Service Provider |
| LAN | Local Area Network |
| LP | Lineares Programm |
| *M2SLS* | Minimierung der maximal auftretenden Summe aller Latenzzeiten der Knoten einer beliebigen Sequenz |
| *MDS* | Minimierung der Dauer einer Session |
| *MERL* | Modifiziertes einfaches „redundantes“ Ladeproblem |
| 2*MDS* | Minimierung der maximalen Dauer eines beliebigen Knotens über alle Knoten einer Session |
| *MSLB* | Minimierung der Summe aller Latenzzeiten der Knoten des Baums |
| *MSLS* | Minimierung der Summe aller Latenzzeiten aller Knoten über alle möglichen Sequenzen |
| MTU | Maximum Transmission Unit |
| NP | Nicht-deterministisch polynomial (Problemklasse) |
| NPC | Nicht-deterministisch polynomial complete (Problemklasse) |
| OSI | Open Systems Interconnection |
| P | Polynomial (Problemklasse) |
| RFC | Requests for Comments |
| SMIL | Synchronized Multimedia Integration Language |
| TCP/IP | Transmission Control Protocol/Internet Protocol |
| W3C | World Wide Web Consortium |
| WWW | World Wide Web |

# Symbolverzeichnis[1]

Mengen

| | |
|---|---|
| $A$ | Menge der Aufenthaltsknoten des Baums $B$ |
| $B$ | Navigationsbaum, bestehend aus Aufenthaltsknoten $A$ und deren Verknüpfungen |
| $G_k$ | $k$-te Teilmenge aus $A$,<br>enthält zueinander redundante Aufenthaltsknoten aus $A$ bzw. $B$ |
| $G_{ki}$ | Menge aller zueinander redundanten Aufenthaltsknoten $G_k$,<br>welche in Teilbaum $T_i$ enthalten sind |
| $S$ | Menge von möglichen Inputsequenzen *seq* eines Verfahrens |
| $s, s_i^j$ | Session bzw. Sessionausschnitt des $i$-ten bis $j$-ten Knoten der Session |
| $\sigma, \sigma_i^j$ | Sequenz des Navigationsbaums $B$ bzw. Sequenzausschnitt<br>des $i$-ten bis $j$-ten Knotens der Sequenz |
| $T_i$ | Teilbaum aus Navigationsbaum $B$ mit Wurzelknoten $a_i$ |
| $X$ | Entscheidungsmatrix, bestehend aus bis zu $n \times n$ Entscheidungsvariablen $x_{ij}$. |
| $X_i$ | Entscheidungsmatrix $X$ für die $i$-te Anfrage bzw. $i$-ten Knoten eines Session |
| $Y$ | Entscheidungsmatrix, bestehend aus bis zu $n \times n$ Entscheidungsvariablen $y_{ik}$ |

Variablen, Hilfsvariablen, Konstanten

| | |
|---|---|
| $\alpha$ | additiv wirkende Konstante |
| $a_i$ | $i$-ter Aufenthaltsknoten, bestehend aus Nutzzeit $u_i$, Ladezeit $\ell_i$ und Objekt $o$ |
| $\beta$ | Verhältnis aus Summe der Nutz- und Summe der Ladezeiten<br>aller Knoten einer Sequenz $\sigma$ |
| $c$ | konstanter Faktor |
| $C_{hik}$ | Konstante, die für alle Knoten der Gruppe $G_k$ des Teilbaums $T_i$ der $h$-ten Sequenz<br>des Navigationsbaums $B$ den Wert $\{0, 1\}$ annimmt |
| $D_i$ | Anzahl der Blätter des Teilbaums $T_i$ aus $B$ mit Wurzel $a_i$ |
| $d_i$ | Aufenthaltsdauer des Aufenthaltknotens $a_i$ |
| $f$ | Fehlerwert |

[1] Aufgeführt sind nur alle abschnittsübergreifend verwendeten Symbole.

| | |
|---|---|
| $k_i$ | Anzahl der direkten Kindknoten des Knotens $a_i$ im Baum $B$ |
| $k_{max}$ | Maximalwert über alle $k_i$ des Baums $B$ |
| $\ell_i$ | Ladezeit des Aufenthaltknotens $a_i$ |
| $n$ | Anzahl der Knoten eines Baums bzw. Teilbaums |
| $m$ | Anzahl der Knoten einer Sequenz $\sigma$ bzw. Session $s$ |
| $o, o_p, o_q$ | Objekt eines Aufenthaltsknotens $a$, auch indiziert mit $p$, $q$, um zwischen voneinander verschiedenen Objekten zu unterscheiden |
| $r_i$ | Restladezeit des Aufenthaltknotens $a_i$ |
| $S_u$ | Wert einer unteren Schranke |
| $S_o$ | Wert einer oberen Schranke |
| $u_i$ | Nutzzeit des Aufenthaltknotens $a_i$ |
| $x_{ij}$ | Entscheidungsvariable, beschreibt die Entscheidung über die Verwendung der Nutzzeit $u_i$ des Knotens $a_i$ zum Laden eines Objekts des Knotens $a_j$ |
| $y_{ik}, y_{jk}$ | Entscheidungsvariable, beschreibt die Entscheidung über die Verwendung der Nutzzeit $u_i$ bzw. $u_j$ des Knotens $a_i$ bzw. $a_j$ zum Laden des Objekts aller zueinander redundanten Knoten in $G_{ki}$ bzw. $G_{kj}$, welche auch im Teilbaum $T_i$ bzw. $T_j$ mit Wurzelknoten $a_i$ bzw. $a_j$ enthalten sind |
| $z_{max}$ | Zielfunktionswert |

Indizes

| | |
|---|---|
| $h, i, j$ | Indizieren $a, d, \ell, r, s, \sigma, T, u, x, X, y$ |
| $k$ | Indiziert $C, G, T, y$ |
| $p, q$ | Indiziert $o$ |
| $r$ | Indexwert für den Wurzelknoten des Navigationsbaums $B$, indiziert $a, d, \ell, u, x, y$ |

Funktionen

| | |
|---|---|
| $D(s_i^j)$ | Liefert die Dauer eines Ausschnitts einer Session $s_i^j$ |
| $D_{ON}(s_i^j)$ | Liefert die Aufenthaltsdauer, resultierend aus einem Online-Verfahren |
| $D_{OPT}(s_i^j)$ | Liefert die Aufenthaltsdauer, resultierend aus einem optimalen Offline-Verfahren |
| $H(B)$ | Liefert die Höhe des Navigationsbaums $B$ |
| $\hat{I}(s_1^m, B)$ | Informationsfunktion, welche eingeschränkt Informationen über $s_1^m$ und $B$ liefert |
| $I(s_1^m, B)$ | Informationsfunktion, welche $s_1^m$ und $B$ liefert |
| $\lambda(s_i^j)$ | Liefert Summe der Ladezeiten aller Knoten des Sequenzausschnitts $s_i^j$ |
| $L^{\mathbb{R}}(B)$ | Liefert den Lösungswert eines linearen Programms für Navigationsbaum $B$ mit reellwertigen Entscheidungsvariablen |

| | |
|---|---|
| $L^{\mathbb{Z}}(B)$ | Liefert den Lösungswert eines ganzzahligen linearen Programms für Navigationsbaum $B$ |
| $\nu(s_i^j)$ | Summe der Nutzzeiten aller Knoten des Sequenzausschnitts $s_i^j$ |
| $R(B)$ | Liefert eine Entscheidungsmatrix, entsprechend einem R-optimalen Zielfunktionswert einer Zielfunktion des Offline-Modells für Navigationsbaum $B$ |
| $R^*(B)$ | Liefert eine Entscheidungsmatrix, entsprechend einem optimalen Zielfunktionswert einer Zielfunktion des Offline-Modells für Navigationsbaum $B$ |
| $R(x,B)$ | Liefert das selbe Ergebnis wie $R(B)$, jedoch unter der Annahme, dass die Nutzzeit des Wurzelknotens des Baums $x$ beträgt |
| $R^*(x,B)$ | Liefert das selbe Ergebnis wie $R^*(B)$, jedoch unter der Annahme, dass die Nutzzeit des Wurzelknotens des Baums $x$ beträgt |
| $t(s_i^i)$ | Liefert den Zeitpunkt, in welchem das Objekt des $i$-ten Knotens der Session $s$ angefordert wird |
| $Wert(X)$ | Liefert den Zielfunktionswert entsprechend der Werte einer Entscheidungsmatrix mit der der Struktur von $X$ |

Sonstige Bezeichner

| | |
|---|---|
| $ALG$ | Ein beliebiges algorithmisch beschreibbares Verfahren |
| $OFF$ | Ein Offline-Verfahren $ALG$ |
| $ON$ | Ein Online-Verfahren $ALG$ |
| $OPT$ | Ein Verfahren $OFF$, welches die bestmögliche, das heißt optimale Lösung, unter vollständiger Information liefert |
| $seq$ | Inputsequenz eines Verfahrens $ALG$, $OFF$, $ON$ bzw. $OPT$ |

# Algorithmenverzeichnis

# Kapitel 1

# Einleitung

*„Man verliert die meiste Zeit damit,*
*dass man Zeit gewinnen will.“*
*John Steinbeck*

In der praktischen Entwicklung verteilter Lernanwendungen zeigt sich, dass das Antwortzeitverhalten in besonderem Maße im Entwicklungsprozess sowie Betrieb zu berücksichtigen ist. Der Einsatz von Prefetching stellt eine äußerst vielversprechende Möglichkeit dar, durch vorzeitiges sowie vorausschauendes Anfordern von Daten, Verzögerungen während der Benutzerinteraktion drastisch zu senken. Mit dem rasanten Wachstum des World Wide Web (WWW) sind in den letzten acht Jahren vielzählige Verfahren zur Realisierung von Prefetching für webbasierte Anwendungen publiziert worden.[1] Die entwickelten Konzepte sind auf allgemein gehaltene, zumeist anwendungsunabhängige, von den Technologien des WWW geprägten Erfordernisse zugeschnitten. Sie eignen sich zwar überwiegend auch für verteilte Lernanwendungen, werden jedoch den besonderen Erfordernissen nur bedingt gerecht.

Diese Arbeit untersucht erstmalig gezielt den Einsatz des Prefetchings für verteilte Lernanwendungen. Der Fokus der Arbeit liegt dabei auf Lernanwendungen, deren Lerninhalte mittels Hypermedien dem Lernenden präsentiert werden. In Publikationen Ende der 90er Jahre von *Joachims*, *Freitag* und *Mitchell* in [JFM97] sowie *Kroeger*, *Long* und *Mogul* in [KL97] werden neben der elektronischen Geschäftstätigkeit im WWW auch verteilte Lernanwendungen als ein interessantes Anwendungsfeld für Prefetching herausgestellt. Spezielle Verfahren, zugeschnitten auf die Erfordernisse verteilter Lernanwendungen, werden aber nicht weiter verfolgt. Besonderen Anforderungen an Prefetchingverfahren, sich aus einem bestimmten Anwendungsbereich ergebend, wird mit den publizierten Verfahren meist nicht entsprochen. Einige Entwicklungen dieser Art zeichnen sich im Bereich der elektronischen Geschäftstätigkeit im Zusammenhang mit adaptiven Benutzerschnittstellen der Anwendungssysteme ab. Sie verknüpfen Methoden des Dataminings mit Prefetchingansätzen (vgl. [SOBB03], [YHN03] und [NKM03]).

Das Problemfeld „Einsatz von Prefetching in verteilten und hypermedialen Lernanwendungen“ stellte sich für mich in der praktischen Entwicklung rechnergestützter Lernumgebungen im universitären Bereich dar (vgl. [TKG02] und [TKG03]). Die sich abzeichnende Problematik kann durch eine Vielzahl von Publikationen der letzten fünf Jahre untermauert werden. Einerseits

[1] Sehr ausführlich stellt *Davidson* die Entwicklung bis 2002 in seiner Dissertationsschrift [Dav02a] dar.

steht die Forderung nach qualitativ hochwertiger, visuell und didaktisch ansprechender Software, welche über die Möglichkeiten klassischer Lehrmedien wie Folie, Buch und Film weit hinausgeht.[2] Andererseits werden Lernszenarien spezifiziert, die nicht nur einen Zugriff auf die Lerninhalte über zumeist hochschuleigene Breitbandnetze beschreiben, sondern insbesondere die Möglichkeit des Selbststudiums in privater Lernathmosphäre zu möglichst kostengünstigen Konditionen betonen. Vorherrschend ist die Meinung, dass eigenständiges Lernen, so genannte Selbstlernszenarien, besonders effektiv mit Web-Technologien unterstützt werden können,[3] wobei die Beschränkung auf Web-Technologien nur aus der Popularität des WWWs resultiert. Stellvertretend dafür stehen aktuelle und zukünftige Technologien zur Realisierung von Diensten auf weit verbreiteten, verteilten und heterogenen Rechnernetzen, mit welchen räumliche und zeitliche Einschränkungen klassischer Lehrformen überwunden werden können. Zusätzlich werden die widersprüchlichen Anforderungen noch verschärft, indem sich Entwickler für ein exploratives, hochgradig interaktives Design zur besseren Unterstützung der Lernprozesse entscheiden.[4] Zur Realisierung solch hochwertiger, spezialisierter Informationensangebote für rechnergestützte Lehre sind trotz Einsatz moderner Technologien im Regelfall verhältnismäßig große Datenmengen zwischen den verteilten Komponenten der Lernanwendung zu übertragen (vgl. [LP01] und [CHR00]). Dieses kann zu langen, auf den Lernprozess außerordentlich störend wirkenden Wartezeiten führen und spiegelt sich in einem schlechten Antwortzeitverhalten der Anwendung wider (vgl. [Röl03], S.323ff.).

Allein für den deutschsprachigen Hochschulbereich wurden in den letzten beiden Jahren knapp dreihundert Lernprodukte fertig gestellt. Überwiegend sind Produkte mit hypermedial strukturierten Lerninhalten, aufsetzend auf Web-Technologien, anzutreffen.[5] Ähnliche, jedoch wesentlich schneller fortschreitende Entwicklungen lassen sich auf dem nordamerikanischen sowie britischen Bildungsmarkt erkennen. Schlagworte wie Distance-Learning, Teleteaching oder e-Learning, welche vor wenigen Jahren noch unterschiedliche Technologien sowie Lehrszenarien beschrieben haben, werden jetzt synonym für den weltweit rapide wachsenden Bereich der webbasierten Lernanwendungen verwendet.[6]

## 1.1 Problemstellung und Forschungshypothesen

Aus der praktischen Entwicklung verteilter Lernanwendungen heraus ergab sich die Notwendigkeit, den Einsatz von Prefetchingverfahren zur gezielten Steuerung des Antwortzeitverhaltens verteilter und hypermedialer Lernanwendungen zu prüfen.

Dabei war auffällig, dass in der umfangreich gesichteten Literatur der letzten 12 Jahre kei-

[2] Vgl. [HG97], [BB98], [Har99], [Ale01], [For01], [Gla01] und [UCS03]

[3] Vgl. [Isk02], [Pet00], [Wil04] und S.25ff. in [Bre04]

[4] In der Literatur wird die besondere Eignung explorativ angelegter, in hohem Maße interaktiv gestalteter Lernanwendungen für das rechnerunterstützte, selbstgesteuerte Lernen hervorgehoben. Häufig wird die Einbettung der hochgradig interaktiven Elemente in ein Hypermedium, welches auch als interaktives Medium angesehen wird, favorisiert (vgl. [UAD+02], S.71ff. in [Röl03], S.149 in [NHHM+04] und S.16ff. in [PMC99] ).

[5] Eine genaue Auflistung der in Deutschland staatlich geförderten Entwicklungen mit detaillierten Informationen sind im Produktkatalog des Bundesministeriums für Bildung und Forschung zu finden (vgl. [Han04] und [BMB05]).

[6] Eine Ausführliche Beschreibung der Entwicklung in den letzten Jahren sowie die zunehmende Dominanz von webbasierten Lösungen und den zugrunde liegenden Ursachen kann in [UCS03] nachgelesen werden.

ne wissenschaftlichen Arbeiten sich dem Thema des Einsatzes von Prefetching für verteilte Lernanwendungen angenommen haben. Im Gegensatz dazu steht ab Mitte der 90er Jahre eine nahezu unüberschaubare Zahl publizierter Prefetchingansätze für webbasierte Anwendungen sowie vielzählige weitere empirische Studien, die sich mit den Besonderheiten der Mensch-Maschine-Interaktion in hypermedialen Lernanwendungen beschäftigen.[7]

Es wird hier die Idee verfolgt, dass der Prozess des rechnergestützten Lernens besondere Anforderungen an das Antwortzeitverhalten während der Interaktion des Lernenden mit der Softwareanwendung stellt. Prefetchingverfahren könnten dann gezielt zur Reduktion von Verzögerungen in der Mensch-Maschine-Interaktion eingesetzt werden, wenn die Besonderheiten in den Verfahren berücksichtigt würden.

Daraus ergeben sich direkt zwei Forschungsfragen:

- Welche speziellen Eigenschaften verteilter hypermedialer Lernanwendungen haben Einfluss auf die Wahl eines Prefetchingverfahrens zur gezielten Einflussnahme auf das Antwortzeitverhalten?

  Empirische Studien zu beobachteten Lernprozessen in hypermedialen Lernanwendungen verweisen darauf, dass Lernverhalten und Navigationsverhalten eng miteinander verknüpft sind. Das Navigationsverhalten hat wiederum drastisch Auswirkungen auf die Leistung eines Prefetchingverfahrens. Die Forschungsergebnisse sind unbedingt miteinander zu verknüpfen. Aufgrund der vielzählig fundierten wissenschaftlichen Arbeiten zur Erklärung von Lernverhalten und Lernergebnissen sowie den Besonderheiten in der Entwicklung von Lernanwendungen existieren detaillierte Vorstellungen über das Navigationsverhalten der Lernenden. Die Kenntnisse über das Navigationsverhalten sind mit einer professionellen Entwicklung hypermedialer Lernanwendungen zu berücksichtigen. Dieser Umstand stellt für den Einsatz von Prefetchingverfahren eine besondere Chance dar. Es gilt, diese Erkenntnisse gezielt im Lebenszyklus verteilter Lernanwendungen auszunutzen.

- Ist es möglich, Prefetchingverfahren zu konstruieren, welche den Besonderheiten verteilter hypermedialer Lernanwendungen entsprechen?

  Es sind Verfahren gesucht, die den Anforderungen aus Entwicklung wie Betrieb verteilter hypermedialer Lernanwendungen im Besonderen entsprechen, um die sich abzeichnende Chance nutzen zu können. Dazu ist eine Betrachtung der existierenden Ansätze sinnvoll. Sie sind entsprechend der Eignung eingesetzter Methoden zu klassifizieren, so dass kritische Aussagen über ihren Einsatz getroffen werden können. Wie bereits angesprochen, ist zu erwarten, dass existierende Verfahren nur teilweise den Anforderungen gerecht werden.

  Kernproblem dieser Arbeit ist es, Optimierungsprobleme zu formulieren, dafür Lösungsverfahren in Form von Prefetchingverfahren zu konstruieren sowie deren Optimierungsergebnisse in den praktischen Anwendungszusammenhang zu integrieren, mit dem Versuch, den Besonderheiten verteilter hypermedialer Lernanwendungen zu entsprechen und kritische Nachteile existierender Verfahren aufzugreifen.

[7] Dazu ausführlich mehr im folgenden Kapitel

Letztendlich wird die Vision verfolgt, existierende, bereits für webbasierte Anwendungen als erfolgsversprechend bewertete Verfahren, mit den hier vorgeschlagenen zu kombinieren und so leistungsfähigere Prefetchingverfahren zur gezielten Beeinflussung des Antwortzeitverhaltens verteilter hypermedialer Lernanwendungen zu schaffen.

Es ist damit zu rechnen, dass Antworten auf die Fragen, insbesondere der zweiten Kernfrage, nicht allein für verteilte hypermediale Lernanwendungen von Interesse sein werden.

Es ergeben sich aus den Fragen folgende zu prüfende Forschungshypothesen:

- Während Entwurf und Betrieb verteilter hypermedialer Lernanwendungen sind eine Vielzahl von Besonderheiten zu beachten, die mit dem Einsatz von Prefetching unbedingt zu berücksichtigen sind.
- Die nach aktuellem Wissensstand bekannten Prefetchingansätze entsprechen nur teilweise den besonderen Anforderungen von Entwicklung und Betrieb verteilter hypermedialer Lernanwendungen.
- Es können für verteilte hypermediale Lernanwendungen entscheidungsunterstützende, rechnergestützte Prefetchingansätze entwickelt werden, welche in besonderem Maße auf bestimmte Besonderheiten zugeschnitten sind. Sie lassen sich entwicklungsunterstützend wie auch im Betrieb einsetzen.

Die ersten beiden Hypothesen sollen anhand des derzeitigen Wissenstands und den eigenen Erfahrungen geprüft werden. Sie werden aber nicht als Kern der Arbeit angesehen. Deren Bestätigung ist jedoch notwendig, um die dritte Hypothese sinnvoll bearbeiten zu können. Zur Bestätigung der dritten Hypothese sind neuartige Verfahren zu konstruieren und deren Eigenschaften zu diskutieren.

## 1.2 Untersuchungsziel, Arbeitsaufgaben und eingesetzte Methoden

Mit dieser Arbeit wird das Ziel verfolgt, Prefetchingansätze zu entwickeln,

- die den speziellen Anforderungen verteilter hypermedialer Lernanwendungen genügen und
- die sowohl für den Betrieb als auch für die Entwicklung verteilter hypermedialer Lernanwendungen sinnvoll nutzbar sind.

Dazu wird folgender Weg beschritten:

- Es sind jene Besonderheiten verteilter hypermedialer Lernanwendungen herauszuarbeiten, die sich auf die Leistung von Prefetchingverfahren auswirken.

  Die bisher nur voneinander getrennt betrachteten Themengebiete „Verteilte hypermediale Lernanwendungen“ und „Prefetching“ sind miteinander zu verknüpfen, so dass bereits existierende Verfahren in Hinblick auf ihre Eignung leicht bewertet werden können. Es

wird vermutet, dass eine Klassifikation in Hinblick auf die Eignung der existierenden Prefetchingansätze möglich ist. Die wissenschaftliche Literatur der letzten Jahre ist für beide Themengebiete zu begutachten und praktische Erfahrung aus Entwicklung und Betrieb von Lernanwendungen sind bei der Bewertung zu berücksichtigen.

- Es ist ein Modell zu erstellen, auf dessen Grundlage eine exakte Problemstellung formuliert, das Anwendungspotential einer möglichen Lösung bewertet und Prefetchingverfahren konstruiert werden können. Das Modell muss hinreichend von den noch zu beschreibenden Anwendungssituationen und den eingesetzten Technologien abstrahieren und die herausgearbeiteten Besonderheiten berücksichtigen.

  Kern dieses Modells sollen kombinatorische Optimierungsprobleme sein. Für diese werden effiziente rechnergestützte Lösungsverfahren gesucht. Auf Basis der Problemlösungen können optimale Entscheidungen für die Prefetchingprobleme gefällt werden. Um den Wert eines solchen Modells und den Wert der die Probleme lösenden Verfahren abschätzen zu können, ist das Anwendungspotential, das soll hier heißen die zu erwartende Leistung bzw. Lösungsgüte potentiell möglicher Verfahren, zu bestimmen. Aufgrund des Online-Charakters und der Unsicherheit in Entscheidungssituationen wird dafür die kompetitive Analyse eingesetzt.[8] Weiterhin ist die Komplexität der formulierten Probleme zur Konstruktion von Lösungsverfahren von Interesse. Zu deren Bestimmung werden Methoden aus der Komplexitätstheorie eingesetzt.

- Es sind Verfahren zu entwickeln, die die im Modell formulierten kombinatorischen Optimierungsprobleme effizient lösen bzw. effizient approximieren.

  Dazu sind effizient lösbare Probleminstanzmengen zu identifizieren und effiziente Lösungsverfahren mit Methoden der Modellierung und Algorithmik zu entwerfen. Es ist zu klären, ob verbreitete „Standardverfahren“ zur Lösung der Optimierungsprobleme bzw. dessen Approximation Anwendung finden können.

  Dieses Teilziel ist von immanenter Bedeutung für das Kernziel der Arbeit. Hier zeigen sich die Möglichkeiten des Einsatzes von Prefetching für verteilte hypermediale Lernanwendungen. Die vorab zu formulierenden Optimierungsziele und Modellbedingungen wirken drastisch auf die Erreichbarkeit dieses Ziels. Bekannte Methoden der Algorithmik und des Entwurfs sowie Beweistechniken zum Nachweis der Korrektheit und der Effizienz sind zwar hilfreich und müssen gezielt eingesetzt werden, letztendlich ist jedoch die grundlegende algorithmische Idee, meist resultierend aus einem kreativen Denkprozess, entscheidend für den Erfolg.

- Abschließend ist zu klären, wie die Ergebnisse der Optimierungsverfahren in die Entwicklung und den Betrieb einfließen können.

  Die Optimierungsverfahren allein sind nutzlos, wenn deren Ergebnisse nicht praktisch nutzbar sind. Hier sind für den praktischen Einsatz der zu entwickelnden Verfahren zwei Bereiche von Interesse. Einerseits ist das angestrebte Antwortzeitverhalten schon mit dem Entwurf der Lernanwendung mittels des so genannten Software Performance Engineering gezielt zu beeinflussen (vgl. [SW02]). Andererseits ist während des Betriebs mittels klassischem Performance Tuning das spezifizierte Antwortzeitverhalten zu sichern. Mit der

[8] Vgl. zur Kompetitivitätsanalyse [PW99]

Arbeit soll gezeigt werden, welche Informationen die konstruierten Optimierungsverfahren für diese beiden Bereiche liefern können, worin dessen Vorzüge und Nachteile zu anderen Verfahren bestehen und wie sich diese mit anderen vorteilhaften Ansätzen verbinden lassen.

## 1.3 Aufbau der Arbeit

Der zum Beschreiten dieses Weges verfügbare Wissensstand wird ausführlich im folgenden Kapitel dargestellt und diskutiert. Diesen Zielen folgend unterteilt sich die Arbeit neben dieser Einführung und der abschließenden Zusammenfassung in fünf weitere Kapitel:

Das zweite Kapitel beschäftigt sich mit der grundlegenden Verknüpfung der Themengebiete „Prefetching“ und „verteilte hypermediale Lernanwendungen“. Es werden die notwendigen Grundlagen dargestellt und abgegrenzt. Dazu werden die elementaren Begriffe „Lernanwendung“, „Hypermedium“, „Verteilung“ und „Prefetching“ geklärt und für den betrachteten Zusammenhang konkretisiert. Es werden weiterführende Überlegungen zu den Besonderheiten der Lernanwendungen bezüglich des Navigationsverhaltens und der Konstruktion von Hypermedien dargestellt und im Zusammenhang mit dem Konzept des Prefetchings diskutiert. Dazu wird eine Klassifikation für Prefetchingverfahren entwickelt und die einzelnen Klassen in Hinblick auf ihre Eignung bewertet, die durch Prefetchingverfahren abzudeckenden Funktionalitätsbereiche anhand eines Komponentenmodells herausgearbeitet sowie Risiko und Chancen des Einsatzes der Verfahren dargestellt. Zum Schluss des Kapitels werden eigene Erfahrungen mit der Entwicklung verteilter Lernanwendungen dargestellt, besonders vielversprechende Lehrszenarien für den Einsatz von Prefetching herausgearbeitet und damit die Ziele der folgenden Modellbildung nochmals präzisiert.

Im dritten Kapitel werden zwei miteinander eng verknüpfte Modelle bezüglich des Hypermediums und des Zugriffs auf Lerninhalte entwickelt. Es werden ausführlich die Vorstellungen zu drei Entscheidungsfeldern und den sie limitierenden Randbedingungen für den Einsatz von Prefetching entwickelt. Darauf aufsetzend werden zwei formal dargestellte Modelle präsentiert und die damit getroffenen Einschränkungen sowie gesetzten Schwerpunkte diskutiert. Kern der Modelle bilden vierzehn, sich teilweise nur marginal in der Zielstellung unterscheidende, kombinatorische Optimierungsprobleme. Ausgewählte, besonders interessante Eigenschaften werden nach der Diskussion der Annahmen analysiert.

Das vierte Kapitel beschäftigt sich mit Optimierungsproblemen des so genannten Offline-Modells, mit welchem insbesondere der Entwurf des Hypermediums der Lernanwendung unterstützt werden kann. Das Kapitel unterteilt sich in zwei große Abschnitte. Im ersten Teil werden für einige der Optimierungsprobleme effiziente Lösungsverfahren dargestellt. Diese Lösungsverfahren vereinen ein gemeinsames Prinzip der Konstruktion einer Lösung, welches vorab ausführlich beschrieben wird. Dann wird die Komplexität der aufgeworfenen Optimierungsprobleme diskutiert und aus der Formulierung des Modells als lineares Programm ein effizientes Approximationsverfahren konstruiert.

Im fünften Kapitel werden Verfahren für vier Optimierungsziele des so genannten Online-

Modells entworfen, welche insbesondere im Betrieb der Lernanwendung eingesetzt werden können. Es wird ein allgemeines Algorithmengerüst für Online-Verfahren des Modells entworfen und darauf aufsetzend ein sehr einfach gehaltenes heuristisches Entscheidungsverfahren betrachtet, welches nicht gezielt eine Optimierung verfolgt. An diesem wird dann die Güte anderer Online-Verfahren diskutiert. Dabei handelt es sich um Online-Verfahren, die sich der für das Offline-Modell konstruierten Verfahren bedienen. Zum Schluss des Kapitels wird auf bemerkenswerte Eigenschaften dieser Online-Verfahren eingegangen.

Das sechste Kapitel beschäftigt sich mit dem praktisch möglichen Einsatz der konstruierten Optimierungsverfahren. Es beschreibt im ersten Teil den Einsatz der Verfahren des Offline-Modells in der Entwicklung von hypermedialen Lernanwendungen. Schwerpunkt ist der Entwurf von Hypermedien in Abhängigkeit von Optimierungsergebnissen, mit welchen auf das zu erwartende Antwortzeitverhalten der Lernanwendung unter Einsatz eines Prefetchingverfahrens geschlossen werden kann. Im zweiten Teil wird der Einsatz der Verfahren des Online-Modells im Betrieb dargestellt. Es werden Möglichkeiten des Performance Tunings und kritische, auf die Leistung der konstruierten Verfahren wirkende Annahmen des formulierten Online-Modells aufgezeigt. Zum Abschluss des Kapitels wird beispielhaft eine von vielen möglichen Verknüpfungsmöglichkeiten der hier vorgeschlagenen Prefetchingverfahren mit anderen publizierten, als in der vorliegenden Literatur sehr erfolgsversprechend herausgestellten Ansätzen, diskutiert.

Das siebente Kapitel liefert eine Zusammenfassung und Bewertung der Forschungsergebnisse sowie einen Ausblick auf weitere offene Fragestellungen.

# Kapitel 2

# Grundlagen und weiterführende Überlegungen

*„Und das Ende all unseres Kundschaftens*
*wird sein, am Ausgangspunkt anzukommen.*
*Und den Ort zum erstenmal zu erkennen.“*
*T.S. Eliot, Little Gidding*

Zur Bearbeitung des Themas sind hier einige grundlegende begriffliche und sachliche Überlegungen notwendig. Diese werden detaillierter als in der Einführung möglich auf die bereits formulierte Zielstellung der Arbeit hinführen.

Es wird zuerst geklärt, was hier unter verteilten hypermedialen Lernanwendungen sowie Prefetching verstanden wird. Dann werden einige Besonderheiten der Entwicklung und des Betriebs verteilter hypermedialer Lernanwendungen herausgestellt, welche wesentlichen Einfluss auf den Einsatz von Prefetching in Kombination mit Lernanwendungen haben. Schwerpunkt liegt dabei auf dem Navigationsverhalten in diesen Anwendungen. Das Navigationsverhalten hat unterschiedlich starken Einfluss auf unterschiedliche Kategorien von Prefetchingansätzen. Eine entsprechende Unterteilung wird in diesem Kapitel angeboten. Darauf aufsetzend werden für die Realisierung eines Prefetchingverfahrens die grundsätzlich benötigten Komponenten dargestellt sowie ausgewählte Gütekriterien zur Bewertung von Prefetchingverfahren und die sich daraus ergebenden Anforderungen an Prefetchingkonzepte für Entwicklung und Betrieb der Lernanwendungen diskutiert. Zum Schluss wird nochmals die Aktualität der Problemstellung sowie sich aus dem Anwendungszusammenhang ergebende Details an einem realen Anwendungsszenario erläutert.

## 2.1 Grundbegriffe

Die hier verwendeten Begrifflichkeiten werden in der vorliegenden Literatur teilweise nicht eindeutig definiert oder in variierender Bedeutung verwendet. Hier soll kein neues Verständnis für die verwendeten Begriffe gebildet werden, jedoch ist eine klare Abgrenzung zu ähnlich oder synonym verwendeten Begrifflichkeiten notwendig.

### 2.1.1 Lernanwendung

**Definition:** Unter dem Begriff *Lernanwendung* soll hier Anwendungssoftware verstanden werden, welche speziell für Lern- und Lehrzwecke entwickelt wurde. Das heißt, Lernanwendungen sind Softwareprodukte, welche die Anwendungsaufgabe des Lehrens bzw. Lernens unterstützen. Der Anwendungsbereich für Lernanwendungen wird rechnergestütztes Lernen bzw. rechnergestützte Lehre genannt. Statt rechnergestützt wird in der sozialwissenschaftlichen Literatur auch häufig der Begriff computerge- bzw. computerunterstützt verwendet. □

Dieses Verständnis über Lernanwendungen lehnt sich an die von *Issing* und *Klimsa* (vgl. [IK02], S.558) allgemein anerkannte Auffassung zum Begriff Lernsoftware an:

„Software, die speziell für Lehr- und Lernzwecke konzipiert und programmiert wurde. Die didaktische Komponente liegt vor allem im Produkt, d.h. in der Software selbst, und zeigt sich im Programmdesign, in der Gestaltung und Gliederung der Benutzeroberfläche, den vorgesehenen Feedback-Mechanismen und Interaktionsmöglichkeiten der BenutzerInnen."

Mit der oben gewählten Definition wird der Anwendungscharakter unterstrichen. Bei Lernanwendungen handelt es sich um Software, welche der oberen Schicht des TCP/IP Referenzmodells zuzuordnen ist.[1] In der Entwicklung stehen gewöhnlich die Modellierung der Benutzerschnittstelle und die daraus resultierenden Interaktionsmöglichkeiten, einem didaktischen Konzept folgend, im Vordergrund.

Die Begriffe Lernsoftware und Lernanwendung werden in der Literatur zum Thema rechnergestützte Lehre und rechnergstütztes Lernen häufig intuitiv verwendet (vgl. zum Beispiel [IK02], S.231ff.).[2] Lernsoftware bzw. Lernanwendungen werden eingesetzt, um rechnergestützte Lernumgebungen zu schaffen. Sie können als ein Teil bzw. Softwaremodul dieser angesehen werden oder eine rechnergestützte Lernumgebung vollständig abbilden. Unter Lernumgebungen werden jedoch primär die äußeren Umstände verstanden, welche zu einer Situation des rechnergestützten Lehrens und Lernens führen.

„Der Begriff der Lernumgebung zielt in erster Linie auf die äußeren Bedingungen ab. Im besonderen geht es um Lernmaterialien und Lernaufgaben sowie um deren Gestaltung, wodurch erwünschte Lernprozesse ausgelöst werden sollen." (vgl. [DS01], S.23)

*Kerres* verwendet statt rechnergestützter Lernumgebung auch den allgemeiner gefassten Begriff der multimedialen und telemedialen Lernumgebung (vgl. [Ker99] und S.25 in [Ker01c]), wobei er mit telemedial auf die räumliche Verteilung einzelner Komponenten der Lernumgebung abzielt, multimedial die Verwendung unterschiedlicher Medien betont, jedoch nicht explizit eine Unterstützung durch Rechentechnik fordert. Im Fokus dieser Arbeit steht nicht die Unterstützung und Gestaltung spezieller äußerer Umstände, dem so genannten Lernarrangement oder der Lernsituation[3], sondern die Unterstützung des eigentlichen Lernprozesses, basierend auf einem starken Maß von Eigenaktivitäten der Lernenden.

Anwendungen, welche solche Lernarrangements rechnerunterstützen, in der Literatur auch virtuelle Lernarrangements genannt, heißen Lernplattformen (vgl. [MS02]). *Schulmeister* schreibt Lernplattformen folgende Funktionalitäten in [Sch03], S.10 zu:

[1] Der Verarbeitungsschicht, siehe [Tan03], S. 41ff.

[2] Tele-, Online- oder eLearning werden immer wieder synonym verwandt und beziehen sich auf Lernangebote, in welchen Telemedien, insbesondere das Internet, zur Distribution von Lernmaterialien zum Einsatz kommen (vgl. [KJ01]).

[3] Vgl. zu Lernarrangement und Lernsituation [Ker00]

- Benutzerverwaltung
- Kursverwaltung
- Rollen- und Rechtevergabe mit differenzierten Rechten
- Kommunikationsmethoden (Chat, Foren) und Werkzeuge für das Lernen (Whiteboard, Notizbuch, Annotationen, Kalender etc.)
- Darstellung der Kursinhalte

Die in dieser Arbeit betrachteten Methoden und Konzepte unterstützen die zuletzt genannte Funktionalität der Darstellung von und des Zugriffs auf Lernmaterialien und Lernaufgaben. Bei *Schulmeister* werden diese in Form so genannter Kurse themen- und/oder benutzerbezogen strukturiert. Das heißt, die hier betrachteten Lernanwendungen erfüllen alle die wesentliche Funktionalität der Darstellung von Lerninhalten, Lernmaterialien und Lernaufgaben und realisieren damit auch den Zugriff auf diese.[4]

Neuere Entwicklungen versuchen, die Darstellung von und den Zugriff auf Lerninhalte personen- oder auch gruppenbezogen zu gestalten. In Abhängigkeit vom Nutzerprofil sollen die Inhalte dargestellt werden. In einem Nutzerprofil werden persönlichen Eigenschaften, Präferenzen des Nutzers, die Rolle des Nutzers innerhalb der Anwendung und das beobachtbare Nutzerverhalten zusammengefasst. Entsprechenden Lernanwendungen, die dieses berücksichtigen, wird die Eigenschaft der Adaptivität zugeschrieben. Sie werden adaptive Lernanwendungen genannt. Grundvoraussetzung, um Adaptivität zu erreichen, ist die Aufzeichnung und Analyse von Nutzereigenschaften und -verhalten (vgl. [TVB$^+$04] und [KM04]). Beides ist für die hier untersuchten Ansätze von Bedeutung und wird in diesem Kapitel diskutiert.

Nach der Abgrenzung und Definition des Begriffs Lernanwendung wird in den folgenden beiden Abschnitten die Verteilung der Komponenten einer Lernanwendung und die Strukturierung der Lerninhalte in Form von Hypermedien dargestellt.

### Verteilte Lernanwendungen

**Definition:** Verteilte Lernanwendungen sind Lernanwendungen, welche aus zueinander räumlich getrennten Komponenten zusammengesetzt sind. □

*Kerres* charakterisiert solche Anwendungen wie oben schon erwähnt auch mit dem Begriff telemedial bzw. Tele-eLearning. Telemedien, insbesondere das Internet, sind für ihn Distributionswege zur Verteilung von Lernmaterialien (vgl. [KJ01], S. 267). Rechnerunterstützung fordert er zwar nicht explizit, alle weiteren Ausführungen beziehen sich aber auf rechnergestützte Lernszenarien.

Der Verteilungsbegriff soll hier jedoch im Sinne von verteilten Systemen aufgefasst werden. *Tanenbaum* wie auch *Coulouris*, *Dollimore* und *Kindberg* heben die Transparenzeigenschaft

[4] Es wären auch Lernanwendungen denkbar, welche nur die Organisation und Verwaltung der Lehre sowie Kommunikation zwischen den Nutzern unterstützen. Diese unterstützen dann jedoch im Regelfall nur sekundär den eigentlichen Lernprozess des Lernenden. Der systematische Wechsel zwischen rechnergestütztem Lernen und Präsenzlehre wird auch hybrides Lernarrangement genannt (vgl. [KJ99]).

verteilter Systeme gegenüber Rechnernetzen hervor (vgl. [Tan03], S. 2 und [CDK02], S. 42ff.), welche hier auch betont werden soll. Die Möglichkeit der räumlichen Verteilung einzelner Komponenten ist zwar wesentlich, um wichtige Funktionalitäten der Lernanwendung zu erfüllen, soll aber für den Nutzer transparent bleiben, das heißt nicht sichtbar. Informationen über das zugrundeliegende Rechnernetz sind für Lernende und Lehrende normalerweise verborgen. *Coulouris*, *Dollimore* und *Kindberg* wie auch *Tanenbaum* unterscheiden zwischen sieben unterschiedlichen Arten der Transparenz (vgl. [CDK02], S. 42 und [Tan02], S. 6ff.). Hier ist die Zugriffs- und Positionstransparenz, von *Tanenbaum* als „location transparency“ bezeichnet, von besonderem Interesse. Mit der Transparenzeigenschaft der Lernanwendung ist zu sichern, dass der Zugriff auf entfernte Ressourcen wie z.B. Lernmaterialien unter Verwendung identischer Operationen ohne Kenntnis über deren Position und Ort erfolgt. Auch ist die Dauer zwischen Anforderung räumlich entfernt gespeicherter Daten und der Auslieferung dieser Daten an den Nutzer in angemessener Zeitdauer zu gestalten. Im Idealfall unterscheidet sich diese Zeit nicht von einem Zugriff auf lokal gespeicherte Daten. Es ist wesentlich für die Benutzbarkeit einer verteilten Lernanwendung, dass der Nutzer nicht eigenständig dafür Sorge tragen muss, woher die für den Lernprozess notwendigen Informationen kommen und dass diese rechtzeitig in angemessener Zeit für ihn verfügbar sind.[5]. Diese beiden für verteilte Lernanwendungen besonders wichtigen Eigenschaften werden Orts- und Zeittransparenz genannt.[6]

Lernplattformen sind im Regelfall verteilte Lernanwendungen. Sie dienen primär der Distribution von Lernmaterialien. *Graf* betont in [Gra04], S. 72:

„Es macht nur Sinn, mit einer Lernplattform zu kommunizieren, wenn die Mitglieder der Gruppe geografisch verteilt sind. Sitzen die Lernenden ohnehin am selben Ort, tauschen sie sich besser, effizienter und lustvoller direkt und persönlich aus und stellen anschließend zuhanden der Lehrperson oder anderer Gruppen die Resultate ihrer Arbeiten in digitaler Form auf die Plattform.“

Lernplattformen dienen dazu, räumliche Distanz durch synchrone und asynchrone Kommunikation der Nutzer zu überbrücken. Die in dieser Arbeit betrachteten Methoden und Konzepte unterstützen die asynchrone Kommunikation von Nutzern. Anwendungen zur Darstellung von Lerninhalten wie Videokonferenzen oder Chats zur Realisierung synchroner Kommunikation zwischen den Nutzern soll hier nicht betrachtet werden. In diesen Anwendungsfeldern haben Prefetchingverfahren keine Bedeutung.[7]

In der Praxis wie auch in der Literatur sind vorherrschend Architekturen für verteilter Lernanwendungen basierend auf dem WWW, das heißt basierend auf Technologien spezifiziert durch das World Wide Web Consortium (W3C). Sie werden webbasierte Lernanwendungen genannt und der Klasse so genannter Web-Anwendungen zugeschrieben.[8] Umgesetzt werden diese über-

[5] Über das, was angemessen heißt, ist noch zu diskutieren.

[6] Den zeitlichen Aspekt erwähnen *Tanenbaum* wie auch *Coulouris*, *Dollimore* und *Kindberg* in ihren Ausführungen zu unterschiedlichen Arten von Transparenz nicht explizit. Um diesem gerecht zu werden, wäre ein zusätzlicher Typ von Transparenz, die Zeittransparenz, in ihren Ausführungen sinnvoll.

[7] Durch den bestimmten, angemessen kurzen Zeitabstand zwischen Senden und Empfang einer Nachricht, spielen Prefetchingverfahren zur Verkürzung der Dauer der Nachrichtenübermittlung, welche spekulativ Daten im Voraus zum Laden bestimmen, dass heißt vor Empfang der eigentlichen Nachricht vom Empfänger, keine Rolle zur Verbesserung des Antwortverhaltens. Grundvoraussetzung ist, dass die Daten vor Anforderung dieser durch den Empfänger auch im System vorhanden sind.

[8] Eine Darstellung unterschiedlicher Klassen und Anwendungsfelder von Web-Anwendungen hat *Murugesan* in [Mur00] vorgenommen.

wiegend auf Client-Server-Architekturen in Form von Web-Browser- und Web-Server-Technologien. Die Kommunikation erfolgt überwiegend per Hyper-Text-Transfer-Protokoll (HTTP). Dieses ist auf die weite Verbreitung webbasierter Technologien, Client-Server-Architekturen und dem HTTP insbesondere zurückzuführen. Die in dieser Arbeit vorgeschlagenen Methoden und Konzepte zur Verbesserung des Antwortzeitverhaltens verteilter Lernanwendungen lassen sich für solche Anwendungen besonders gut verwenden, sind aber auf die Anwendungsklasse Web-Anwendungen und Client-Server-Architekturen nicht beschränkt. Andere Architekturvarianten wie zum Beispiel die Nutzung von Peer-to-Peer-Architekturen sind denkbar. Wesentlich jedoch ist, dass diese nicht nur verteilt sind, sondern dass die Lerninhalte hypermedial strukturiert vorliegen.

### Hypermediale Lernanwendungen

In Anlehnung an *Casteleyn* und *De Troyer*, welche Eigenschaften von Hypermedien in [CT02] beschreiben, wird hier unter einem Hypermedium folgendes verstanden:

**Definition:** Ein Hypermedium stellt eine Sammlung untereinander verknüpfter Informationselemente dar. □

Die Verknüpfung zweier oder mehrerer Informationselemente wird als Link bezeichnet.

Nach dem sehr allgemein gehaltenen Verständnis von *Casteleyn* und *De Troyer* sind Links Verknüpfungen zweier oder mehrerer Informationselemente. Ein Link ist entweder uni- oder bi-direktional. Ein uni-direktionaler Link verweist von einem Element oder mehreren Elementen, der Quelle, auf ein oder mehrere andere Elemente, das Ziel bzw. die Ziele. Bi-direktionale Links verknüpfen ein oder mehrere Elemente zu ein oder mehreren Elementen in beide Richtungen. Die Rolle von Quelle und Ziel ist austauschbar.

Ein uni-direktionaler Link beschreibt dem nach eine Vorgänger-Nachfolger-Beziehung zwischen Elementen. Bi-direktionale Links sind hingegen ungerichtet und beschreiben nur die direkte Nachbarschaft zwischen den Informationselementen.

Hier sind Eigenschaften von Hypermedien aus dem Blickwinkel der Navigation von besonderem Interesse. *Casteleyn* und *De Troyer* unterscheiden je nach Funktion des Links in einem Hypermedium zwischen vier unterschiedlichen Linktypen.[9] Einer dieser Typen beschreibt navigationsbezogene Links. Dieser Typ wird zur Darstellung der möglichen Navigation durch ein Hypermedium verwendet. Auf die einzelnen Informationselemente kann in Abhängigkeit von navigationsbezogenen Links zugegriffen werden. Diese beschreiben die Möglichkeit, ausgehend von einem oder mehreren Knoten, auf die in der Regel unterschiedlichen Wege, das heißt Abfolgen von Informationselementen bzw. Informationselementmengen, welche durch Links miteinander verbunden sind, auf andere Informationselemente zuzugreifen. Eine solche Abfolge von Elementen bzw. Elementmengen wird Navigationspfad (vgl. [FW03], S.591), Anfragemuster[10] (vgl. [Dav04]), Zugriffsmuster[11] (vgl. [PHMaZ00]) oder auch Nutzmuster[12] (vgl. [CSM97], [BHS02] und [BS00]) genannt. Hier soll der Begriff Nutzmuster bzw. in Bezug auf Lernanwendungen Lernpfad oder Lernweg verwendet werden.

[9] Sie unterscheiden zwischen navigationsbezogenen, strukturellen, semantischen und prozessbezogenen Links (engl. navigational, structural, semantic, process logic links).

[10] englisch request pattern

[11] englisch access pattern

[12] englisch usage pattern

Die sich aus den Verknüpfungen und Elementen ergebende, im Regelfall nicht-lineare Struktur eines Hypermediums kann einfach als Graph dargestellt werden, indem die Informationselemente des Mediums als Knoten des Graphen und die Verknüpfungen zwischen diesen als gerichtete (unidirektionale Links) und ungerichtete (bi-direktionale Links) Kanten abgebildet werden. Werden nur die navigationsbezogenen Links des Mediums betrachtet, dann heißt der daraus resultierende Graph Navigationsgraph (vgl. [Spi99],[Her02]).

Im Gegensatz zum Hypertext als ein spezieller Typ eines Hypermediums bestehend aus Informationselementen in Textform, wird mit Verwendung des Begriffs Hypermedium betont, dass die Informationselemente aus unterschiedlichen Medien bestehen können (vgl. [Jon96], S. 703 sowie [Ker01c]). Es wird zwischen vier Informationstypen unterschieden:

- Text
- Audio
- Einzelbild
- Bewegtbild

Das hier im Folgenden verwendete Modell eines Hypermediums orientiert sich allein an der Beschreibung möglicher Navigation innerhalb des Mediums. Es modelliert die navigationsbezogenen Links. In der Literatur lassen sich eine Vielzahl Hypermedia-Modelle finden, welche sich hauptsächlich in der Verwendung unterschiedlicher Linktypen, insbesondere in Bezug auf spezielle Anwendungsfelder, unterscheiden. Auch spielen die unterstützten Informationstypen in diesen Modellen häufig eine prägende Rolle. Alle haben jedoch die nicht-lineare Verknüpfung von Informationselementen mit Hilfe von Links gemeinsam. Zudem bieten alle mindestens einen navigationsbezogenen Linktyp zur Beschreibung von Navigation innerhalb des Mediums an.[13] Neben dem Hypermedium selbst werden auch Handlungen zum Zugriff und zur Manipulation der darin abgelegten Daten unter Berücksichtigung der vorgegebenen Struktur benötigt.

**Definition:** Hypermediale Lernanwendungen sind Lernanwendungen, dessen Lerninhalte überwiegend in Form von Hypermedien so strukturiert sind, dass die Navigation der Nutzer, insbesondere Lernender, mit Hilfe navigationsbezogener Links erfolgt. Sie stellen Funktionalitäten bereit, mit deren Hilfe Nutzer auf die Informationselemente des Hypermediums in Abhängigkeit der Links zwischen diesen zugreifen können. □

Nach neueren Entwicklungen in den letzten vier Jahren werden die Informationselemente, aus denen die hypermedial strukturierten Lerninhalte der Lernanwendung zusammengesetzt sind, mit Hilfe so genannter Lernobjekte beschrieben (vgl. [Fri01], [Bru04] und [SLH04]).[14] Hier soll angenommen werden, dass die Informationselemente in Form von persistenten Objekten

[13] Neuere Modellentwicklungen sind zum Beispiel WebML, beschrieben in [CFB+03] für ein Konferenzbegutachtungssystem, Vorschläge von *Koch* und *Kraus* in [KK02], basierend auf der Unified Modelling Language, oder in [GCP00] ein Vorschlag für ein objektorientiertes Hypermedia-Modell. *Schwinger* und *Koch* untersuchen in [SK04] unterschiedliche Hypermedia-Modelle sowie Vorgehensweisen zur Konstruktion von Hypermedien aus Sicht des Softwareengineerings.

[14] Mit diesen Objektbeschreibungen ist jedoch eher eine Klassenbeschreibung im Sinne der Objektorientierung als eine Objektbeschreibung gemeint. Es handelt sich eigentlich um relativ gut erweiter- und wiederverwendbare „Lernklassen“. Der Begriff Klasse ist jedoch schon mit einer anderen Bedeutung belegt.

bzw. Klassenbeschreibungen, aus welchen Objekte instanziiert werden, in der Lernanwendung als Lernobjekte gespeichert sind. Diese beschreiben im Sinne der objektorienierten Softwareentwicklung eine Einheit aus Daten und Handlungen, die auf diese Daten zugreifen.

Weiterhin sind für die in der Einleitung umrissene Problemstellung und für die später noch zu formulierenden Optimierungsprobleme nicht alle Navigationslinks einer hypermedialen Lernanwendung von Bedeutung. Es werden die Lernobjekte in so genannten Aufenthaltsknoten zusammengefasst. Sie bestehen jeweils aus genau einem Objekt, in welchem ein oder mehrere Lernobjekte eines Hypermediums zusammengefasst sind. Die Lernobjekte in einem solchen Objekt eines Aufenthaltsknotens haben gemeinsam, dass nach der Anforderung dieser bzw. Navigation zu einem dieser, mittels eines navigationsbezogenen Links, ausgehend von einem anderen Aufenthaltsknoten, alle ab einem bestimmten Zeitpunkt vollständig durch die Lernanwendung dem Nutzer zur Verwendung zur Verfügung gestellt werden. Vollständig zum Verwenden für den Nutzer verfügbar heißt, dass ab diesem Zeitpunkt auf alle Daten und Schnittstellen der Handlungen der Lernobjekte ohne zeitliche Verzögerungen zugegriffen werden kann.[15]

Die Auswahl der Lernobjekte, welche einem Aufenthaltsknoten zugeordnet werden, findet mit dem Entwurf des Hypermediums statt. Die daraus resultierende Größe der Knoten, auch Granularität oder Korngröße genannt (vgl. [Blu98] in Abschnitt 2.1.3.1), hat entscheidenden Einfluss auf die Benutzbarkeit des Mediums zur Unterstützung von Lernprozessen.[16] Mit der Fokussierung auf Aufenthaltsknoten werden die Navigationsmöglichkeiten zwischen den Lernobjekten innerhalb eines Aufenthaltsknotens vernachlässigt. Navigationslinks zwischen Aufenthaltsknoten betreffen immer genau ein Paar solcher Knoten und sind uni-direktional. Eine bi-direktionale Beziehung zwischen zwei Knoten wird durch zwei entgegengesetzt gerichtete uni-direktionale Links abgebildet. Reflexive Links, das heißt Navigationslinks mit demselben Aufenthaltsknoten als Quelle und Ziel, werden nicht betrachtet. Mit der Navigation von einem Aufenthaltsknoten zu einem anderen Aufenthaltsknoten können durch die Dauer des Bereitstellens zur Verwendung der Lernobjekte für den Lernenden unerwünschte Verzögerungen, so genannte Latenzzeiten, auftreten. Diese können den Lernprozess beeinflussen. Die Dauer des Bereitstellens, das heißt die Zeitdauer zwischen Anforderung und vollständiger Verfügbarkeit des Objekts eines Aufenthaltsknotens, wird von einer Vielzahl von Faktoren beeinflusst, insbesondere aber von der möglich entfernten Speicherung der Lernobjekte in einer verteilten Lernanwendung.[17]

Aufenthaltsknoten und deren Verknüpfungen lassen sich einfach in einem Graph oder Netz darstellen. Er bzw. es wird in der Literatur Navigationsgraph oder Navigationsnetz genannt und beschreibt mit seinen Knoten und Kanten die Sicht möglicher, das heißt vom Nutzer wählbarer, Navigationpfade bzw. Nutzmuster durch das Hypermedium. Die Aufenthaltsknoten sind Knoten

[15] Im folgenden Kapitel wird der Aufenthaltsknoten formal mit der Modellbeschreibung des Hypermediums definiert und noch um zusätzliche zeitbezogene Daten angereichert. Hier ist erst einmal nur das Zusammenfassen von Lernobjekten von Interesse.

[16] *Feldmann* und *Wagner* gehen in [FW03] ausführlich auf den Entwurf von Hypermedien in Hinblick auf Benutzbarkeit ein. Sie entwickeln eine Vielzahl quantitativer Maße zur Bewertung der Benutzbarkeit. Auf den Entwurf von Hypermedien wird noch genauer eingegangen.

[17] Zum Beispiel können auch komplexe graphische Darstellungen von Bildschirmpräsentationen zu zeitlichen Verzögerungen führen, spielen aber mit immer ausgefeilteren Methoden der graphischen Darstellung und zunehmender Rechenleistung eine untergeordnete Rolle für Lernanwendungen. In [RCM+96] wird dieser Aspekt ausführlich diskutiert. In [RS02] und [RS03] erfolgt das auch in Bezug auf Hypermedien, soll aber als ein sehr spezielles Randthema in Bezug auf die Zielstellung der Arbeit nicht weiter untersucht werden.

des Graphen. Die Verknüpfungen zwischen diesen werden als gerichtete Kanten im Graphen abgebildet (vgl. [FW03], S.592ff.).

In der so genannten „eLearning-Literatur“ werden Hypermedien als so genannte interaktive Medien angesehen. *Blumstengel* schreibt zum Beispiel in [Blu98], Absatz 2.2.3.6: „Ein hinreichender Grad an Interaktivität ist ein konstituierendes Merkmal von Hypermedia.“, wobei unklar bleibt, was genau sie unter hinreichendem Grad versteht. Die Interaktivität nach *Röll* besteht darin, dass der Lernende einen Weg durch das Hypermedium, den so genannten Navigationsweg, frei wählen kann, der Lernende erst mit dem Akt des Lernens durch die Wahl eines Navigationsweges ein eigenes Bild aus den einzelnen Informationselementen zusammensetzt (vgl. [Röl03], S.74). Ein solcher von *Röll* beschriebener Navigationsweg stellt hier ein Nutzmuster des Hypermediums, beginnend mit einem Startknoten des Navigationsgraphen, dar. Wesentlich ist, dass dieser Weg der Lernenden durch Mensch-Maschine-Interaktion über die durch die Lernanwendung bereitgestellte Benutzerschnittstelle gestaltet werden kann (vgl. [Isk02], S.29 und [Sch02]). Im Regelfall steht dieser nicht mit Beginn des Lernens, das heißt der Anforderung des ersten Aufenthaltsknotens eines Nutzmusters fest, sondern entwickelt sich erst im Verlauf der Nutzung und spiegelt in gewisser Weise den Lernprozess des Lernenden wider.

Die Navigation durch das Hypermedium einer Lernanwendung sowie der Entwurf von Hypermedien soll mit Prefetchingverfahren unterstützt werden. Das Konzept Prefetching sowie die Abgrenzung zu ähnlichen Konzepten wird im folgenden Abschnitt vorgenommen.

### 2.1.2 Prefetching

Das Konzept Prefetching ist eine von vielen Möglichkeiten, das Antwortzeitverhalten in verteilten Systemen zu verbessern. Folgend den Ausführungen von *Davidson* in [Dav02a], S.2 wird hier unter Prefetching folgendes verstanden:

**Definition:** *Prefetching* ist das spekulative Anfragen einer Ressource. Die Antworten auf die Anfragen werden antizipativ mit einem Cachedienst zwischengespeichert, um an diesen Dienst zukünftig gestellte Anfragen beantworten zu können. □

**Definition:** Dementsprechend sind *Prefetchingverfahren* algorithmisch beschreibare Verfahren, welche das Konzept Prefetching umsetzen. □

Mit dem Konzept Prefetching wird das Ziel verfolgt, die Latenzzeit[18], auch Wartezeit genannt, zu reduzieren. Unter Wartezeit wird die Zeitdauer verstanden, die zwischen den Zeitpunkten Beginn des Sendens einer Anfrage und vollständigem Empfang der dazugehörigen Antwort entsteht (vgl. [CDK02], S.71).[19] Grundsätzlich kann die Reduktion von zeitlichen Verzögerungen erfolgen, indem die Antworten mit Hilfe von Cachediensten zwischengespeichert werden.[20] Das Konzept Prefetching unterscheidet sich aber grundsätzlich von dem des Cachings. Cachedienste werden dazu eingesetzt, durch eine Komponente wiederholt gestellte Anfragen mit verkürzter Wartezeit zu beantworten. Prefetching hingegen eignet sich dazu, die Wartezeit auch dann zu reduzieren, wenn die Anfrage durch eine Komponente erstmals gestellt wird,

[18] englisch latency time

[19] Auch als „end-to-end response time“ bezeichnet (vgl. [MA02], S.129).

[20] Verkürzen der „round-trip-time“ oder Erweiterung der zur Verfügung stehenden Bandbreite werden als klassische Alternativen zum Prefetching in [MA02], S.133ff. diskutiert und sollen nicht Gegenstand dieser Arbeit sein.

indem eine weitere Komponente im Voraus, dass heißt spekulativ, diese Anfrage stellt, das Ergebnis zwischenspeichert und im nachhinein zeitnah liefern kann. Neben der Reduktion von Wartezeiten kann mit dem Einsatz von Prefetching ein weiteres Ziel verfolgt werden: die effiziente Ausnutzung bereitstehender Ressourcen in Form von Rechnernetzen bzw. die Reduktion der Kosten für benötigte Ressourcen, um bestimmte Dienste in einer festzulegenden Qualität anbieten zu können.

Verfahren, die die Konzepte Prefetching wie auch Caching umsetzen, können dazu eingesetzt werden, die für verteilte Lernanwendungen als besonders wichtig herausgestellte Eigenschaft der Zeittransparenz zu verbessern bzw. überhaupt zu gewährleisten. Ziel ist natürlich nicht, Wartezeiten nach beliebigen Anfragen im verteilten System zu reduzieren. Vielmehr geht es immer darum, aus Sicht der Konzeption einer Lernanwendung die vom Nutzer wahrgenommenen Unterbrechungszeiten[21] in einem festgelegten, nach Benutzbarkeitsüberlegungen erträglichen Rahmen zu gestalten: im Idealfall genauso, dass der Nutzer nicht unterscheiden kann, ob er mit seiner Interaktion über die Benutzerschnittstelle der Lernanwendung eine Anfrage an eine entfernt oder lokal verfügbare Ressource ausgelöst hat.

In Bezug auf verteilte Anwendungen, im Speziellen webbasierte Anwendungen, wird Prefetching als eine Option zur Verbesserung des Antwortzeitverhaltens seit Mitte der 90er Jahre betrachtet. In Publikationen von *Padmanabhan* in [Pad95] sowie *Dingl* und *Partl* in [DP96] und weiteren folgenden Veröffentlichungen (vgl. [Bes96], [PM96] und [RCM+96]) wird die Problematik verzögerter Antwortzeiten und deren Auflösung mit Hilfe von „verstecktem Laden“[22] der Daten diskutiert.[23]

In der wissenschaftlichen Literatur wird der Begriff Prefetching und dessen Abgrenzung zu anderen Konzepten nicht einheitlich verwendet. Von *Tanenbaum* wird statt Prefetching der Begriff „Proactive Caching“ verwendet (vgl. [Tan03], S.658), bezieht jedoch das Konzept nur auf die Verwendung innerhalb hierarchisch organisierter Caches mit Hilfe so genannter Proxydienste in Rechnernetzen.[24] Das Prefetching verallgemeinernde, auch an das Zwischenspeichern mittels Cache gebundene Konzept, wird Preloading genannt. *Davidson* zum Beispiel definiert: „Preloading is the speculative installation of data in a cache in the anticipitation that it will be needed in the future.“ (vgl. [Dav02a], S.2). Preloading beinhaltet auch das Konzept des Prepushing, welches häufig zur Konstruktion von Disk-, CPU- oder Server-Caches eingesetzt wird. Prepushing und Prefetching unterscheiden sich nach *Davidson* insbesondere dadurch, dass bei ersterem keine Abfragen über ein bei einem Cachedienst realisierten Entscheidungsverfahren initiiert werden. Die Daten werden auch im Voraus in den Speicher des Caches abgelegt, jedoch von anderen initiiert, von zum lokalen Cache räumlich entfernten Diensten des Rechnernetzes.

Diese Unterscheidung wird in der Literatur jedoch keineswegs einheitlich durchgehalten. So unterscheidet *Oren* in einem Überblicksartikel zu Prefetchingtechniken zur verbesserten Prozessorauslastung nicht zwischen den Konzepten Preloading und Prefetching, sondern nutzt nur den Begriff Prefetching für das übergeordnete Konzept (vgl. [Ore00]). *Bestavros* sowie *Jacobson*

[21] englisch user-perceived latency, vgl. zum Begriff [BKK03]

[22] Siehe zu „verstecktem Laden“ [Pad95]

[23] Eine der ersten bzw. wahrscheinlich die erste Veröffentlichung eines Prefetchingverfahrens für webbasierte Anwendungen ist [Lie95]. *Lieberman* beschreibt ein Verfahren, welches die vom Web-Browser präsentierten Dokumente auf darin enthaltene Hyperlinks parst und Dokumente, auf die die Links verweisen, zusätzlich anzeigt.

[24] Die Funktionsweise von Web-Proxys erklärt Tanenbaum in [Tan03], S. 657ff.. Dieses sind architekturabhängige Details, die für das Thema der Arbeit keine Rolle spielen und auf die nicht weiter eingegangen wird.

und *Cao* beschreiben in [Bes96] und [JC98] serverseitig initiiertes Prepushing für Client-Server-Architekturen, nennen dieses aber Prefetching. *Wang* verwendet im Überblicksartikel [Wan99] zu Prefetching den Begriff Preloading synonym zu Prefetching. Auch in aktuelleren Artikeln ist eine ungenaue Abgrenzung zwischen Preloading und Prefetching zu finden. Zum Beispiel *Bouras*, *Koundinaris* und *Kostoulas* schreiben in dem bemerkenswerten Artikel [BKK03] zu empirischen Messergebissen über die Effizienz von ihnen entwickelter Prefetchingverfahren, meinen jedoch ganz allgemein Preloading-Verfahren.

Weiterhin werden in der Literatur teilweise die Konzepte Prefetching bzw. Preloading und die Vorhersage der im Voraus in den Cache zu ladenden Daten nicht klar voneinander getrennt. Vorhersageverfahren über zukünftige Anfragen an den Cache werden im Zusammenhang mit Preloading aufgrund des spekulativen Charakters des Konzepts häufig verwendet. Es werden Daten im Speicher des Caches in der Hoffnung abgelegt, dass diese zukünftig auch angefordert werden. Vorhersagealgorithmen, Predictoren genannt und für Prefetching beispielsweise in [Dav02a] und [NKM03] beschrieben, werden dazu verwendet, möglichst zutreffende Vorhersagen über zukünftige Abfragen zu generieren. Dadurch kann die Entscheidungsfindung eines Prefetchingverfahrens unterstützt werden. Aber nicht nur die Vorhersage, sondern auch andere noch zu diskutierende Faktoren sollten in die Entscheidungsfindung eines Prefetchingverfahrens eingehen. Aus diesem Grund soll hier klar zwischen dem Konzept des Prefetching und dem der Vorhersage zukünftiger Anfragen getrennt werden. Steht die Vorhersage als Entscheidungsgrundlage eines Prefetchingverfahrens im Vordergrund, dann werden solche Verfahren auch unter dem Konzept „Predictive Prefetching“ gefasst (vgl. [NKM03]).

Die Aufgaben eines Prefetchingverfahrens sind grundsätzlich dadurch charakterisiert, dass zu entscheiden ist,

- welche Daten, Datenobjekte oder Datenobjektbeschreibungen,
- in welchem Umfang,
- zu welchen Zeitpunkten bzw. unter welchen Bedingungen

in den Speicher des Caches abzulegen sind. Je nach Problemlage kann auch noch zu entscheiden sein,

- welche Daten aufgrund begrenzter Kapazitäten aus dem Speicher des Caches zu entfernen sind und
- von welcher bzw. welchen Ressourcen Daten, Datenobjekte oder Datenobjektbeschreibungen anzufordern sind.

Die Auswahl aus dem Speicher des Caches zu entfernender Daten aufgrund begrenzter Kapazitäten wird Prefetchingverfahren im Regelfall nicht zugeschrieben, kann jedoch stark mit den drei oben genannten Punkten korrespondieren. In dieser Arbeit werden von diesen fünf Punkten nur die ersten drei detailliert betrachtet. Im Rahmen verteilter hypermedialer Lernanwendungen, wie diese in der Praxis häufig anzutreffen sind, spielen die beiden zuletzt genannten eine untergeordnete Rolle. Der vierte Punkt wird äußerst selten berücksichtigt. Dieses hängt wahrscheinlich auch mit der Komplexität der zugrunde liegenden Problemstellung zusammen. Auch in dieser Arbeit wird der vierte Punkt kaum eine Rolle spielen. Es sei jedoch auf [CK01b]

verwiesen. Hingegen für den fünften Punkt lassen sich in der gesichteten wissenschaftlichen Literatur keine Ausführungen in Bezug auf Prefetching finden.

Aufgrund einiger Ungenauigkeiten in der Literatur soll hier nochmals betont werden, dass

- Prefetching immer an einen Cache gebunden ist und damit auch durch die Eigenschaften von Cachediensten limitiert ist,
- Prefetching sich nicht auf ein bestimmtes Architekturkonzept wie zum Beispiel Client-Server-Architekturen bezieht,
- Prefetchingverfahren in Client-Server-Webarchitekturen beim Clienten, Server als auch beim Proxy ausgeführt werden können,
- die Auswahl eingesetzter Predictoren zwar im Regelfall wesentlichen Einfluss auch auf Prefetching hat und damit auch bei der Bewertung von Prefetchingverfahren zu berücksichtigen ist, aber die Vorhersage als ein eigenständiges, vom Prefetching abzugrenzendes Konzept bzw. Problemfeld angesehen wird.

**Web-Prefetching**

Ein Großteil der in der Literatur diskutierten Konzepte und Verfahren zum Prefetching beziehen sich auf den durch das WWW, kurz Web, gegebenen Rahmen und werden in der Regel mit dem Begriff Web-Prefetching bezeichnet (vgl. [BRS03], S. 235). Nicht alle als Web-Prefetching dargestellten Konzepte sind auch an das WWW, insbesondere an standardisierte Kommunikationsprotokolle gebunden. Jedoch haben alle gemeinsam, die vom Nutzer eines Clienten (Web-Browser) erfahrene Wartezeit[25] nach Anfrage entfernt gespeicherter Daten, i.d.R. vom Web-Server verwaltete Web-Seiten, zu reduzieren (vgl. [BKK04], [Dav02b], [FS00] und [JK00]).

Web-Prefetchingverfahren bedienen sich jedoch der durch das World Wide Web Consortium als Standard festgelegten Technologien und überwiegend der im Web vorherrschenden Client-Server-Architektur. Eine Vielzahl der Web-Prefetchingverfahren sind auf Besonderheiten des HTTPs sowie der Hypertext-Markup-Language (HTML) zugeschnitten. Diese nutzen die standardisierte Struktur von HTML-Dokumenten sowie Besonderheiten des HTTPs, basierend auf den ebenfalls standardisierten Protokollen der TCP/IP-Familie[26]. Ausgenutzt werden Besonderheiten beim Verbindungsauf- sowie Verbindungsabbau und der inhaltlichen Analyse der übertragenen Daten (vgl. [SDMML03a], [Dav02b], [CK00], [CKR99]) und [Duc99]).

Bemühungen des W3Cs, Prefetching durch Standards zu unterstützen, spiegeln sich in den Spezifikationen RFC 2068 für HTTP Version 1.1 sowie in der für HTML 4.01 wieder (vgl. [FGM+97], Abschnitt 19.6.2.4. und [RHJ99]). Ab RFC 2068 können Attribute zur genaueren Beschreibung von Links zwischen zwei Ressourcen verwendet werden. Auch mit HTML 4.01 ist es möglich, Links in HTML strukturierten Dokumenten als im Voraus ladbar zu markieren.

[25] englisch user perceived latency
[26] TCP/IP - transmission control protocol/internet protocol

So ist der Weg seit 1997 offen, basierend auf Standards des W3Cs für Web-Dienste Prefetchingverfahren zu entwickeln.[27]

### DNS-, Connection- und Content-Prefetching

Werden Prefetchingverfahren für verteilte Anwendungen betrachtet, dann lassen sich die verwendeten Prefetchingkonzepte in die drei Kategorien Connection-, DNS- und Content-Prefetching unterteilen.[28] Wartezeiten in Rechnernetzen ergeben sich nicht nur aus der Zeitdauer zur „reinen" Übertragung der angeforderten Daten. Zusätzlich wird Zeit benötigt, um Komponenten, welche die Daten liefern können, zu lokalisieren sowie alle benötigten Verbindungen zur Datenübertragung auf- und abzubauen.[29] In [NGBS+97] ist diese Problematik am Beispiel des HTTPs Version 1.1 ausführlich beschrieben worden.

DNS-Prefetching wird eingesetzt, um im Voraus Ressourcennamen in Adress-Namen zu übersetzen (vgl. [CK01a]). Connection-Prefetching zielt auf den Verbindungsaufbau im Voraus gegebenenfalls mit Hilfe persistenter Verbindungen ab.[30] In [CK00] werden beide Prefetchingansätze für das HTTP, basierend auf dem TCP/IP, erläutert. DNS- und Connection-Prefetching nutzen spezifische Aspekte der eingesetzten Protokolle der Sitzungs- und Transportschicht aus. Beide Konzepte wären grundsätzlich dafür geeignet, als anwendungsunabhängige Dienste in einem Rechnernetz realisiert zu werden, nach dem OSI-Referenzmodell in der Transport- und Sitzungsschicht, nach dem TCP/IP-Referenzmodell in der Transport und Anwendungsschicht. Von *Cohen* und *Kaplan* werden in [CK00] beide Ansätze neben weiteren für das Hypertext-Transfer-Protokoll dargestellt sowie Vor- und Nachteile auf Basis empirischer Untersuchungen diskutiert. Die Reduktion der Wartezeiten mittels Connection- bzw. DNS-Prefetching ist gegenüber Content-Prefetching eher gering. Sie ist unabhängig vom Umfang der zu übertragenden Daten und hängt von der Anzahl benötigter Verbindungen sowie angefragter Ressourcen ab. Empirische Untersuchungen haben ergeben, dass sich Wartezeiten für Web-Anfragen durchschnittlich aufgebauter Web-Dokumente um Größenordnungen von Milli-, Zehntelsekunden bis zu wenigen Sekunden je übertragenem Dokument reduzieren lassen (vgl. [CK00], [VYK+02]). Das Anwendungspotential dieser Konzepte für das WWW ergibt aus den überdurchschnittlich häufig ähnlich aufgebauten Dokumenten. *Cohen* und *Kaplan* und weitere Studien zur Webstruktur[31] charakterisieren Web-Dokumente in Bezug auf die zu übertragende Datenmenge im

[27] Einfache Prefetchingverfahren werden zum Beispiel seit 2003 von Browsern der Mozilla-Familie wie Mozilla 1.2 oder Netscape 7.01 realisiert (vgl. [FS03]). Link-Tags in HTML strukturierten Dokumenten sind mit rel="prefetch" zu markieren, werden vom Browser als solche erkannt und können je nach im Browser realisiertem Prefetchingverfahren behandelt werden.

[28] Abkürzung DNS: Domain Name Service. Der Name ergibt sich aus der Anwendung für das WWW, in welchem die Auflösung von Namen in Adressebereiche mit Hilfe von DNS erfolgt. Das zugrunde liegende Konzept könnte auch Adress-Prefetching genannt werden. In der Literatur lassen sich auch die englischen Schlagworte pre-connecting und pre-resolving für die beiden Prefetchingansätze finden (zum Beispiel siehe [CK00]).

[29] Natürlich werden auch zur Realisierung dieser Dienste Daten übertragen. Genau jene sind aber mit obiger Formulierung „reiner Übertragung der angeforderten Daten" nicht gemeint.

[30] Persistente Verbindungen werden nicht nur beim Connection-Prefetching sondern auch beim Caching eingesetzt. Untersuchungen und umfangreiche Erläuterungen dazu und zum Konzept allgemein lassen sich in [CKZ03], [CDF+98], und [FCD+99] finden.

[31] Vgl. [DFKM97], [DMF97], [AFJ99], [LWP+01], [BYE03] und [NCO04]. Die Ergebnisse der Studien lassen keine drastischen Änderungen der hier interessierenden Parameter im Zeitablauf erwarten. Zwar ändert sich die

Durchschnitt als verhältnismäßig klein, bestehend aus vielen kleinen, jeweils zu übertragenden Text- und Bildelementen mit einer durchschnittlich hohen Anzahl von Links. Damit spielen diese Konzepte für verteilte Lernanwendungen als ein spezieller Anwendungstyp nur insofern eine Rolle, als das sie auch für jede andere verteilte Anwendung zum Nachrichtenaustausch eingesetzt werden könnten.

Content-Prefetching zielt hingegen auf das vorzeitige Anfordern der Daten ab, welche in der Anwendungsschicht manipuliert bzw. dargestellt werden, wie dieses zum Beispiel die darzustellenden Lerninhalte einer verteilten Lernanwendung sind. Es werden anwendungsspezifische Aspekte ausgenutzt. Eine Vielzahl von Verfahren basiert auf der Analyse bereits übertragener Daten (vgl. [Dav02b]). Die Wartezeiten lassen sich nicht nur in Abhängigkeit vom Umfang der zu übertragenden Daten verkürzen. Ein wesentlicher Aspekt ist auch die zur Verfügung stehende Zeitdauer, in welcher Ressourcen zum Übertragen von Daten ohne Einsatz von Prefetching ungenutzt bleiben (vgl. [JK98], [JC98] und [JWS02]). Diese wiederum ergibt sich aus dem jeweiligen Anwendungskontext - hier das Unterstützen von Lernprozessen.[32]

Aufgrund der dargestellten begrifflichen Abgrenzungen soll hier im Gegensatz zu *Badach*, *Rieger* und *Schmauch* zwischen Prefetching basierend auf Technologien des WWW (Web-Prefetching) und Content-Prefetching unterschieden werden (vgl. [BRS03], S.235ff.). Ist ein Content-Prefetching-Konzept bzw. -Verfahren auf Technologien des WWW zugeschnitten, dann wird dieses als Web-Content-Prefetching-Konzept bzw. -verfahren bezeichnet (vgl. [KPRR04], S.315).

Die Abgrenzung der das Thema der Arbeit betreffenden Begriffswelt und Konzepte ist abgeschlossen. In den folgenden Abschnitten werden die Themenbereiche „Verteilte, hypermediale Lernanwendung“ und „Prefetching“ in Hinblick auf die Zielstellung der Arbeit miteinander verknüpft. Aus den genannten Gründen ist es sinnvoll, nur Content-Prefetching aus dem Blickwinkel verteilter hypermedialer Lernanwendungen genauer zu untersuchen. Bevor dieses geschieht, sind die das Konzept Prefetching betreffenden Besonderheiten verteilter hypermedialer Lernanwendungen herauszuarbeiten und dann folgend die Eignung unterschiedlicher Prefetchingansätze für verteilte hypermediale Lernanwendungen zu hinterfragen.

## 2.2 Besonderheiten verteilter hypermedialer Lernanwendungen

Entwicklung und Betrieb verteilter hypermedialer Lernanwendungen unterscheiden sich nicht grundsätzlich von anderen rechnergestützten Anwendungen. Jedoch sind einige Aspekte aufgrund der Verteilung von Komponenten, der Strukturierung der Inhalte als Hypermedium sowie des Umstandes, dass durch diese ein multimediales Lernarrangement beschrieben wird, in Bezug auf das Untersuchungsziel dieser Arbeit zu betonen. Es ergeben sich spezielle Anforderungen, denen teilweise mit dem Einsatz von Prefetchingkonzepten Rechnung getragen werden kann.

*Pauen* und *Six* weisen in [PS99], S. 146 zu Entwicklung und Betrieb multimedialer Lernanwendungen in einer umfangreichen Studie (vgl. [NBW+99]) über „Softwaretechnische Anforderun-

Größe des WWW, jedoch kaum die Eigenschaften und Struktur der Dokumente.

[32] Dazu ausführlich im folgenden Abschnitt mehr

gen an multimediale Lehr- und Lernsysteme“ ausdrücklich darauf hin, dass sich deren Entwicklung von der „konventioneller“ Softwareanwendungen grundsätzlich nicht unterscheidet, jedoch eine Vielzahl spezieller Faktoren zu berücksichtigen sind. Sie betonen, dass das Fokussieren auf Präsentations- und Navigationsfunktionalitäten dazu führt, dass neuartige Ansätze in Spezifikation und Entwurf für die Entwicklung notwendig sind. In derselben Studie unterstreicht *Weidauer* in [Wei99], S. 145, dass im Gegensatz zur Entwicklung konventioneller Software Dynamik und Zeitbehaftung multimedialer Lernanwendungen eine besondere Berücksichtigung von zeitabhängigen Reaktionsunterschieden zur Laufzeit der verwendeten Medien bedürfen. Es sind demnach Methoden des Performance-Engineering und -Tuning anzuwenden, worauf noch eingegangen wird. Er schlägt ein phasenorientiertes Vorgehensmodell vor (vgl. [Wei99], S.114ff.), welches sich von anderen Vorgehensmodellen insbesondere durch die Betonung des Entwurfs der Benutzerschnittstelle, das heißt der Mensch-Maschine-Interaktion, unterscheidet.

Auch *Bolchini*, *Paoline* und *Randazzo* weisen in [BPR03] darauf hin, dass die Entwicklung von Multimediaanwendungen, insbesondere hypermedial strukturierter Lernanwendungen, Besonderheiten unterliegt. Sie charakterisieren den Entwicklungsprozess als nutzerzentriert, ausgerichtet an Entwicklungsaktivitäten, bezogen auf die Navigation und Inhaltserstellung. Dieses schlägt sich schon in der Definition der Anforderungen von solchen Anwendungen nieder (vgl. [BP02]). Jedoch nicht nur das Requirements Engineering, sondern der gesamte Entwicklungsprozess hypermedial strukturierter Anwendungen ist von Aktivitäten gekennzeichnet, welche die

- Strukturierung der Inhalte,
- Präsentation der Inhalte,
- Zugriffsmuster auf die Inhalte und die damit eng verknüpfte
- Navigation durch die Anwendung

betreffen (siehe auch [SK04]).

Auch für hypermedial strukturierte Inhalte von Web-Anwendungen ganz allgemein werden diese vier Dimensionen betont und lassen sich in Vorgehensmodellen zur Entwicklung von Hypermedien, eingebettet in so genanntes Web-Engineering, wieder finden in [KPRR04], [Lan02] und [GSV00]. Jedoch unterscheidet sich die Entwicklung von hypermedialen Lernanwendungen in einem Punkt ganz wesentlich von Web-Anwendungen allgemein. Durch die Ausrichtung einer Lernanwendung auf den zu unterstützenden Lernprozess und auf konkrete Zielgruppen sind meistens bzw. sollten immer detaillierte Vorstellungen über die zu unterstützenden kognitiven Prozesse beim Lernenden innerhalb jeder Phase der Entwicklung, insbesondere Spezifikation und Entwurf sowie dann auch Betrieb und Wartung vorhanden sein. Je nach kognitivem Stil des Lernenden ist ein bestimmter Lernstil zu erwarten, von welchem wiederum das Navigationsverhalten und damit die Art und Weise des Zugriffs auf Lerninhalte einer Lernanwendung geprägt wird. Es kann angenommen werden, dass Strukturierung und Präsentation der Lerninhalte, Navigationsmöglichkeiten durch die Lernanwendung, Lernstil und Präferenzen für bestimmten Medientypen das Navigationsverhalten des Lernenden maßgeblich beeinflussen. Eine Vielzahl von Ergebnissen empirischer Untersuchungen weisen auf diesen Sachverhalt hin.[33]

[33] Siehe folgender Abschnitt „Navigationsverhalten“

Im noch folgenden Abschnitt „Prefetchingverfahren und Konzepte im Überblick“, S.29 wird darauf eingegangen, dass die Vorhersagbarkeit von Navigationsverhalten das Leistungspotential von Prefetchingverfahren beeinflusst. Hier sei vorweggenommen, dass für hypermedial strukturierte Web-Anwendungen, unabhängig vom konkreten Einsatzbereich, eine Vorhersagbarkeit des Navigationsverhaltens in einem gewissen Rahmen zu beobachten ist. Die im folgenden Abschnitt 2.2.1 aufgeführten Indizien zur Vorhersagbarkeit von Navigationsverhalten der Lernenden in hypermedialen Lernanwendungen weisen darauf hin, dass Navigationsverhalten in besonderem Maße gut vorhersagbar sein sollte. Die Eignung von Vorhersagekonzepten, speziell auf den Lernprozess unterstützt durch hypermediale Lernanwendungen abgestimmt, soll jedoch in dieser Arbeit nicht weiter diskutiert werden, weil sie für die in den Kapiteln 4 und 5 vorgestellten Verfahren eine untergeordnete Rolle spielen.[34] Aufgrund der für Prefetchingverfahren zentralen Bedeutung des Navigationsverhaltens der Lernenden als Spiegelbild der durch Lernanwendungen zu unterstützenden Lernprozesse beschäftigen sich die folgenden zwei Abschnitte mit dem Navigationsverhalten im Speziellen.

### 2.2.1 Navigationsverhalten

Unter dem Navigationsverhalten eines Nutzers oder einer Nutzergruppe soll hier die Art und Weise verstanden werden, in welcher auf die Lerninhalte einer Lernanwendung zugegriffen wird. Das Navigationsverhalten äußert sich in der Reihenfolge und dem zeitlichen Ablauf des Zugriffs auf die Inhalte sowie in den zum Zugriff genutzten Funktionalitäten. In einer Vielzahl von empirischen Studien zur Abschätzung der Effizienz und des Leistungspotentials von hypermedialen Lernanwendungen wurde überwiegend in den 90er Jahren untersucht, von welchen Faktoren das Navigationsverhalten beeinflusst wird.

#### Vorhersage

Aufgrund der folgenden Indizien wird angenommen, dass das Navigationsverhalten in hypermedialen Lernanwendungen gut vorhergesagt werden kann, zumindest genauso gut wie eine anwendungsunspezifische Vorhersage des Navigationsverhaltens von Nutzern hypermedialer Anwendungen des WWW.

*Alomyan* stellt in dem Überblick gebenden Artikel [Alo04] heraus, dass der kognitive Stil eines Lernenden, das heißt die Art und Weise in welcher Personen Informationen bearbeiten und auswerten, stark von den individuellen Präferenzen und Gewohnheiten Wissen zu organisieren geprägt wird. Der kognitive Stil eines Lernenden wiederum prägt nachhaltig den Lernstil bzw. die Lernstrategie, das heißt die Art und Weise des Lernens, und damit auch den Lernweg bzw. das Navigationsverhalten (vgl. [Alo04], S.189). *Alomyan* verweist in diesem Zusammenhang nur auf eine einzige empirische Studie von *Ford* und *Chen* (vgl. [FC02]), welche aufzeigt, dass Navigationsverhalten von drei miteinander eng korrespondieren Faktoren abhängig ist:

- dem kognitiven Stil,
- den Präferenzen und

[34] Dieses Thema wurde in der gesichteten aktuellen Literatur bis jetzt nicht aufgegriffen. Aus dieser Arbeit heraus ergeben sich aber einige Ansatzpunkte und die Motivation, ein solches Forschungsthema zu bearbeiten.

- den Gewohnheiten des Lernenden.

Nach diesen Faktoren werden die Lernenden in einer Vielzahl von Studien zur Bewertung der Lerneffizienz von Hypermedien in feldabhängige und feldunabhängige Lernende unterteilt. Die beiden Gruppen sind dadurch charakterisiert, dass feldabhängige Lerner präferieren in Gruppen zu arbeiten, auf die Umwelt gerichtetes Verhalten zeigen, leicht von außen beeinflussbar sind, präsentierte Ideen schnell als gegeben hinnehmen. Feldunabhängige hingegen zeigen ein individuelles, selbstgesteuertes Verhalten, akzeptieren Ideen durch analytisches Verständnis(vgl. [Alo04] und [FC02]).[35]

Auch *Brenstein* unterscheidet in [Bre96] zwischen grundsätzlich zwei beobachtbaren Verhaltensmustern: dem tiefen- und dem oberflächen-strategischen Vorgehen beim Lernen. Oberflächenstrategisches Vorgehen entspricht einem Lernverhalten, welches von den Anreizstrukturen der Lernanwendung bzw. von zufälligen Entscheidungen geprägt ist. Tiefenstrategisches Vorgehen ist hingegen durch die eigenständige Organisation des Lernprozesses durch den Lernenden gekennzeichnet. Je nach Verhaltensmuster sind lineare oder nichtlineare Zugriffsmuster zu beobachten. Auch unterscheidet sich die Dauer der Lernprozesse, die Nutzdauer der Lerninhalte sowie die verwendeten Inhalte von Lernenden, die diesbezüglich unterschiedliches Verhalten charakterisieren. Mehrere Studien weisen auf einen signifikant beobachtbaren Unterschied zwischen diesen zwei Verhaltensmustern hin.[36] Offen bleibt jedoch die Frage, anhand welcher quantitativen Maße Lernende der einen bzw. der anderen Gruppe in Abhängigkeit vom beobachteten Verhalten zugeordnet werden können. Für eine Klassifikation der Lernenden mit Hilfe algorithmischer Verfahren wären solche Betrachtungen notwendig. Ansatzpunkt könnten Maße sein, welche die abweichende Auswahl von Lernwegen des Lernenden von standardisierten bzw. vom Lehrenden geplanten Lernwege je Lernstil beschreiben. [37]

Andere Studien verweisen auf mehr als zwei beobachtbare Gruppen hin, die ein unterschiedliches Lern- und Navigationsverhalten in hypermedialen Lernanwendungen aufweisen. So wird von *MacGregor* der „sequential", „video viewer" und „concept connector" beobachtet (vgl. [Mac99]). *Melara* sowie *Ougthon* und *Reed* beschreiben den „Diverger", „Accomodator", „Converger" und „Assimilator" und weisen darauf hin, dass die Gruppenzugehörigkeit auch mit dem Studiengang des Lernenden korreliert.[38] In [HBD00] sowie [LK98] wird die Aussage getroffen, dass je nach beobachtbarer Gruppenzugehörigkeit unterschiedliche Lerndauern beobachtet werden können. Die Gruppen unterscheiden sich dadurch, wie lange sie sich mit bestimmten Lerninhalten beschäftigen sowie welche Inhalte, wie häufig und in welcher Reihenfolge angefordert werden.

Die Studien [SK99] und [RAL96] zeigen, dass nicht nur die Charakteristik des Lernenden den Lernprozess nachhaltig beeinflusst, sondern auch das Layout von Lernanwendungen wesentlichen Einfluss auf die Dauer von Lernprozessen hat. Die Lerndauer, gemessen als die Zeit, die

[35] Eine exakte Unterscheidung zwischen beiden Lernertypen wird in [FC02] dargestellt. Erstmals 1977 wird in [WMGC77] diese Unterscheidung zur Charakterisierung von kognitiven Stilen verwendet (vgl. [Alo04]).

[36] Vgl. [LH99] , [CF00], [HBD00], [BB01] und [Alo04]

[37] Vgl. dazu Vorschläge von *Berendt* und *Brenstein*, welche solche Maße zur Visualisierung von Lernstilen betrachten. Zur algorithmischen, automatisierten Auswertung erscheint der Einsatz von Methoden zur Beschreibung von Graphen- und Baumstrukturen sinnvoll, wie dies zum Beispiel in [FW03] für webbasierte Anwendungen allgemein dargestellt wird.

[38] Vgl. dazu die Studien [Mel96] und [OR99] sowie die von *Röll*, welcher diese Lerntypen detailliert beschreibt und auf ein Instrumentarium zur Lernstildiagnose verweist (vgl. [Röl03], S. 135ff.).

Lernende Inhalte der Anwendung nutzen, kann je nach Layout bei gleich bleibenden Inhalten stark variieren. In beiden Studien herrscht die Meinung vor, dass die Art und Weise der Strukturierung und Aufbereitung der Lerninhalte das Lernverhalten und insbesondere die Zeit der Nutzung der Lerninhalte wesentlich stärker beeinflusst als die drei oben genannten Faktoren zur Beschreibung der Charakteristik des Lernenden. In [PRW04] hingegen wird eine bestimmte Affinität der Lernenden zu bestimmten Medientypen beobachtet, wodurch je nach Aufbereitung der Lerninhalte für unterschiedliche Medientypen im Hypermedium, auch ein unterschiedliches Verhalten beobachtet werden kann.

Es ist unstrittig, dass der zeitliche Ablauf von Lernprozessen sich im Navigationsverhalten der Lernenden innerhalb hypermedialer Lernanwendungen widerspiegelt (vgl. [Ker01b], S.112). Jedoch in welchem Maße Lernverhalten und Navigationsverhalten im Zusammenhang stehen, ist nicht bis in das letzte Detail geklärt und wird weiter Gegenstand der Erforschung menschlicher Verhaltensweisen sein.

**Erfassung**

Auch der Umkehrschluss ist zulässig. Anhand des Navigationsverhaltens kann auf die zu unterstützenden Lernprozesse und damit auf das Lernverhalten teilweise geschlossen werden (vgl. [Sch99] und [HBD00]). Dazu ist das Navigationsverhalten zu erfassen bzw. zu protokollieren. Die Verhaltensprotokollierung ist von zentraler Bedeutung in der Entwicklung von Lernanwendungen. Sie ist die Basis für eine professionelle Evaluation der Lernergebnisse, dem Feedback für Lehrende als auch Lernende und kann auch für Prefetching genutzt werden.

*Scholler* sieht in [Sch99] zwei durch die Verhaltensprotokollierung zu unterstützende zentrale Fragestellungen:

- Wie ist die Lernleistung der Lernenden zu beurteilen?
- In welchem Lernzustand befinden sich die Lernenden?

Rechnergestützte Lernumgebungen ermöglichen eine „automatische“ Erfassung des Navigationsverhaltens. *Berendt* und *Brenstein* diskutieren in [BB01] ausführlich unterschiedliche Konzepte der Erfassung von Lernverhalten, um diese beiden Fragen beantworten zu können. Zur rechnergestützten Erfassung des Navigationsverhaltens wird das so genannte „Learner-Tracking“ angewendet die Speicherung des Zugriffs auf Lerninhalte und der Verweilzeiten des Lernenden (vgl. [Ast01], S.4 und [NHHM$^+$04], S.303). Je nach Untersuchungsziel und Möglichkeiten wird gespeichert, wer, wann, worauf mit Hilfe welcher Funktionalität der Anwendung zugegriffen hat und welche Benutzereingaben mit dem Zugriff im Zusammenhang stehen. Die Möglichkeiten der Erfassung ergeben sich aus dem technisch sinnvoll machbaren und dem organisatorischen Rahmen, welcher nicht zuletzt auch von Aspekten der Datensicherheit und des Datenschutzes vorgegeben wird.[39]

- Auf Basis der erhobenen Daten und deren Analyse kann die Evaluation der Lernanwendung unterstützt werden. *Kerres* und weitere Autoren betonen, dass eine solche für eine professionelle Entwicklung neben den softwaretechnisch ausgerichteten Tests zwingend notwendig ist (vgl. [Ker01a] und [RKFH02], S.130ff. sowie [Pet99], S. 243ff. und [NHHM$^+$04], S. 291ff.).

[39] Dazu sei auf [Zai01], [DHNS04] und [SHF03] verwiesen.

- Weiterhin ist neben der entwicklungsbegleitenden eine fortlaufende Evaluation während des Betriebs der Lernanwendung zu ermöglichen. Lehrende sind adäquat zur konventionellen Präsenzlehre mit Informationen zur Lernerfolgskontrolle zu versorgen (vgl. [Kel99], S.19).

- Einige Kernfunktionalitäten von rechnergestützten Lernanwendungen lassen sich mit einer „automatisierten" Erfassung und Analyse von Lernverhalten teilweise unterstützen. So ist die Erkennung von Anomalien im Lernverhalten sowie das Feedback an Lernende über ihr Lernverhalten in der rechnergestützten Lehre besonders wichtig (vgl. [Kel99], [Ker00] und [NHHM+04], S.227ff.).

In der Literatur wird überwiegend eine Analyse des Navigationsverhaltens in hypermedial strukturierten Lernanwendungen anhand so genannter Lernwege, auch Lernpfade genannt, propagiert (vgl. [BB01], [CM02] sowie [Ker01b] auf den Seiten 225 und 232ff.). Das sind Wege, die eine zeitliche Abfolge des Zugriffs auf die Lerninhalte der Anwendung beschreiben. Dieses ist sinnvoll, da Hypermedien sich von anderen Medien insbesondere durch die Freiheiten im Zugriff auf die Lerninhalte unterscheiden. Das Nutzen dieser Freiheit in der Navigation lässt Rückschlüsse in Hinblick auf die beiden oben genannten Fragestellungen zu.

Mechanismen zur Protokollierung des Zugriffs auf entfernte wie lokale Ressourcen in verteilten Rechnernetzen mit dem Ziel der Überwachung des Nachrichtenverkehrs sind weit verbreitet. Die daraus resultierenden Datenmengen, gespeichert in Log-Dateien, eignen sich aber nur bedingt zur Erstellung detaillierter Lernwege. *Prolli* und *Pitkow* stellen in [PP99] überblicksartig Techniken zur Erstellung solcher Wege dar. In diesem Zusammenhang sei auch auf [Dav99] und [JK00] verwiesen, in welchen besonders auf die Qualität von Log-Dateien als Informationsquelle unterschiedlichster Analyseziele eingegangen wird.

Als ein Ergebnis der Analyse der Log-Dateien mittels Methoden der Zugriffsanalyse, auch „Web-Usage-Mining" für webbasierte Anwendungen genannt[40], entstehen die bereits erwähnten Navigations- bzw. Nutzmuster.[41] *Herder* liefert in [Her02] einen Überblick, welche Metriken zur Auswertung des Navigationsverhaltens und der zugrunde liegenden Struktur des Hypermediums sinnvoll erscheinen. Er und weitere Autoren präferieren Nutzmuster sowohl zur qualitativen als auch zur quantitativen Analyse von Navigationsverhalten in Hypermedien in Form von Baumstrukturen darzustellen und zu speichern (vgl. [FW03], [FR04] und [Her02]). Die in dieser Arbeit entwickelten Prefetchingansätze verwenden solche Nutzmuster in Form vom Bäumen gespeichert, hier Navigationsbäume genannt. Sie setzen damit auf ein weit verbreitetes Vorgehen zur Analyse von Navigationsverhalten in hypermedialen Lernanwendungen und von Hypermedien auf. Die für die Methoden benötigten Navigationsbäume entstehen sowohl in der professionellen Entwicklung als auch im Betrieb von Lernanwendungen.[42] Zusätzlich rücken moderne Techniken zur Erfassung und Auswertung von Navigationsverhalten mittels Navigationsmustern aufgrund der wieder lauter werdenden Forderung nach adaptiven Lernanwendungen in den Mittelpunkt (vgl. [DHNS04] und [KA03]).

[40] Eine Einführung in Web-Usage-Mining für Lernanwendungen liefern *Zaiane* und *Luo* in [ZL02] und [Zai01]

[41] Vgl. zu Navigations- und Nutzmustern auch [EW04], S. 201ff.

[42] Auf die in diesen Bäumen im Einzelnen enthalten Informationen wird mit der formalen Beschreibung der Problemstellung im folgenden Kapitel 3 noch genauer eingegangen.

### 2.2.2 Berücksichtigung der Lernzeit

Das Abschätzen der Lernzeit bzw. Lerndauer, das heißt der Zeit, in welcher sich der Lernende mit bestimmten Lerninhalten beschäftigt, spielt für die Entwicklung von Lernanwendungen eine herausragende Rolle. *Kerres* verweist in seiner Diskussion zur Planung von Lernszenarien explizit darauf (vgl. [Ker01b], S. 142):

„Für die Planung mediengestützter Lernangebote besonders wichtig ist die zu erwartende Lerndauer, d.h. wie viel Zeit die Lerner mit dem Lernmedium arbeiten werden bzw. wollen, auch wenn diese Variable bisher kaum Gegenstand mediendidaktischer Erörterungen gewesen ist."

Anhand des Zeitbedarfs zur Wissensaufnahme des Lernenden wird die Effizienz bzw. das Leistungspotential von Lernarrangements bewertet (vgl. [BB96], S.140ff. und [Ker01b], S.112 sowie [QB99]). Auch ist der Zeitbedarf für die Segmentierung[43] und Sequenzierung[44] der Lerninhalte ein wichtiger Anhaltspunkt (vgl. [NHHM$^+$04], S. 100ff. und [Ast01], S.9ff. und [RKFH02], S.65 sowie [BB96], S.141).

Weiterhin können Wartezeiten beim Zugriff auf entfernte Ressourcen während des Lernprozesses vom Lernenden als Störung empfunden werden (vgl. [Röl03], S. 323 und [Epp99], S. 140). *Scholz* weist in [Sch01], S.25 darauf hin, dass Systemantwortzeiten von mehr als zwei Sekunden Benutzer besonders anfällig für exogene Einflüsse wie z.B. Störungen durch Lärm werden lassen. Benutzer werden während des Wartens abgelenkt und der Lernprozess wird negativ beeinflusst. Jedoch nicht nur die anfallende Wartezeit allein erhöht die Lerndauer. Die Störungen führen dazu, dass Denkprozesse an die durch das Warten vorgegebene „Rhythmik des Lernens" anzupassen sind. Die Möglichkeit des Benutzers, die Geschwindigkeit der Informationsaufnahme, die „Rhytmik des Lernens", selbst zu bestimmen, wird häufig als ein entscheidender Qualitätsfaktor multimedialer Lernanwendungen in der Diskussion um die Vorteile hypermedial strukturierter Lernanwendungen angeführt (vgl. [Mer99], S.139). Dazu gehört insbesondere, dass Lernende das Warten oder besser Pausieren selbst bestimmen können.

*Scholz* beschreibt aber auch, dass Antwortzeiten weit unter einer Sekunde die natürliche Antwortzeit eines menschlichen Interaktionspartners unterschreiten und vermieden werden sollten. In Tabelle 2.1 sind die in Anlehnung an *Scholz* beschriebenen Auswirkungen durch mittlere Antwortzeiten auf Nutzereingaben dargestellt (vgl. [Sch01], S.24ff.). Zusätzlich betont *Scholz*, dass nicht nur die Antwortzeit selbst, sondern unbegründete Abweichungen von den vom Benutzer erwarteten Antwortzeiten oftmals als Fehlfunktion des Systems interpretiert werden. Es erscheint selbstverständlich, dass ein gutes Antwortzeitverhalten auf interaktive Eingaben essentiell für Nutzerzufriedenheit und Produktivität sind (vgl. [DT82], [Bra86] und [Roa98]). Dies betrifft auch die navigationsbezogene Interaktion in Hypermedien (vgl. [RBP98], [BBK00] und [KR01], S.130). Nach einer diesbezüglichen Studie zu webbasierten Anwendungen, zitiert von *Davidson* in [Dav02a], S. 9, ist davon auszugehen, dass aufgrund der schlechten Erfahrungen mit der Benutzung des WWW längere Wartezeiten von bis zu 8 Sekunden toleriert werden.[45] In Bezug auf verteilte Lernanwendungen mag dieses jedoch bezweifelt werden, da nicht nur die

[43] Prozess der Zerlegung der Lerninhalte in Curicula, Kurseinheiten bis hin zur Bildschirmseite und einzelnen Lernobjekten

[44] Prozess des Erstellens einer zeitlichen Abfolge der segmententierten Lerninhalte in Form von Lernwegen

[45] In dieser Studie [Zon99] wird die 8-Sekunden-Regel postuliert. Erst nach 8 Sekunden empfinden WWW-Nutzer das Warten als störend.

| Mittlere Antwortzeit $x$ in Sekunden | Beschreibung | Auswirkungen |
|---|---|---|
| $x << 1$ | unmittelbare Reaktion auf Nutzereingaben | Stresssituationen möglich |
| $1 \leq x \leq 2$ | angemessene Antwortzeit | notwendig für alle Aufgaben, die mehrere zeitnah aufeinander folgende bzw. in ihrer Bedeutung eng verknüpfte Interaktionen erfordern |
| $2 \leq x \leq 4$ | Antwortzeit als Warten wahrgenommen | kein konzentriertes Arbeiten möglich |
| $4 \leq x \leq 15$ | als sehr unangenehmes Warten wahrgenommen | nur nach abgeschlossener Hauptinteraktion tolerierbar, Kurzzeitgedächtnis überfordert |
| $x > 15$ | als zu langes Warten wahrgenommen | nicht tolerierbar |

Tabelle 2.1: Toleranzgrenzen für mittlere Antwortzeiten auf Benutzereingaben

Wahrnehmung und daraus resultierende Akzeptanz für verteilte Lernanwendungen von Bedeutung ist, sondern auch der Lernprozess negativ beeinflusst wird.

Damit ergeben sich aus der Forderung nach zeittransparenten verteilten Lernanwendungen verschärfte Anforderungen an das Antwortzeitverhalten für verteilte Lernanwendungen, unabhängig davon, ob sie auf Web-Technologien aufsetzen und als so genannte webbasierte Anwendungen vom Lerner wahrgenommen werden oder nicht. Eine möglicherweise gute Vorhersage des Navigationsverhaltens bietet eine Chance, dieser Forderung mit Prefetchingkonzepten gerecht zu werden. Diese Konzepte müssen jedoch dem als wesentlich herausgestellten Aspekt der Lernzeit gerecht werden. Mit dem Entwurf einer verteilten Lernanwendung ist das Antwortzeitverhalten zu berücksichtigen. Entsprechende Methoden lassen sich im Bereich des Software-Performance-Engineering wiederfinden.

### 2.2.3 Performance-Engineering und -Tuning

*Smith* und *Williams* beschreiben in ihrem als Lehrbuch angelegten Veröffentlichung [SW02] eine Vielzahl von Methoden, mit welchen die Entwicklung und der Betrieb zeitkritischer Anwendungen begleitet werden kann. Das Software-Performance-Engineering bezieht sich dabei

auf Methoden, welche darauf abzielen, Software so zu konstruieren, dass diese den gestellten leistungsabhängigen Anforderungen genügt (vgl. [SW02], S.16ff.). Dabei können Methoden der Simulation wie auch analytische Methoden zur Anwendung kommen. Performance-Tuning hingegen umfasst reaktive Maßnahmen, i.d.R. Aktivitäten der Phase Wartung und Betrieb im Softwarelebenszyklus (vgl. [Sch99], S.3).

Verteilte hypermediale Lernanwendungen sind nach den oben diskutierten Aspekten unbedingt als zeitkritisch anzusehen, was zwar offensichtlich ist, aber nicht als selbstverständlich angesehen wird. Das Management von Ressourcen für die Realisierung zeitkritischer Zugriffe auf entfernte Lerninhalte ist von entscheidender Bedeutung für die Qualität der Lernanwendung. Performance-Engineering sowie -Tuning spielen bis heute in Softwareentwicklungsprojekten verteilter Lernanwendungen eine eher untergeordnete Rolle. Unterschiedlichste Ursachen führen dazu, dass Probleme der Projektorganisation, -durchführung und Qualitätssicherung der Lerninhalte im Vordergrund stehen (vgl. [PS99], [FS02] und [Ale01]). Es ist jedoch anzunehmen, dass durch die zunehmende Vernetzung der Lehrangebote mittels Internet (vgl. [UCS03] und [JL04]) der Verteilungsaspekt eine immer wichtigere Rolle spielen wird. Auch kann angenommen werden, dass die Erwartungen bezüglich der technischen Qualität der Lehrinhalte und des Grades an Interaktivität zukünftig weiter ansteigen werden. Dies führt dazu, dass sich das Antwortzeitverhalten verteilter Lernanwendungen drastisch verschlechtert, wenn dem nicht gezielt im Entwicklungsprozess entgegengesteuert wird.

Die in den folgenden Kapiteln dieser Arbeit dargestellten Ansätze lassen sich im Rahmen von Performance-Engineering als auch -Tuning verteilter hypermedialer Lernanwendungen verwenden. Mit ihnen kann der Entwurf von solchen Lernanwendungen unter Berücksichtigung der möglichen Verwendung vom Prefetching unterstützt werden. Jedoch die dem Performance-Engineering und -Tuning zugrunde liegende Vorgehensweise, die Integration in Entwicklungs- und Wartungsaktivitäten, sollen in dieser Arbeit nicht betrachtet werden. Dazu sei auf die systematischen Betrachtungen von *Scholz* in [Sch99] sowie *Smith* und *Williams* in [SW02] verwiesen, welche jedoch auf die Besonderheiten der Entwicklung und des Betriebs von hypermedialen Lernanwendungen keinen Bezug nehmen. Entsprechende methodisch angelegte, wissenschaftliche Arbeiten, bezogen auf die performance-orientierte Entwicklung von Hypermedien und Lernanwendungen, sind nach heutigem Stand nicht verfügbar.[46] Aufgrund dieses Mankos wird im vorletzten Kapitel der Arbeit (S.167ff.) ein knapp gehaltener Ausblick gegeben, wie die hier entwickelten Ansätze zur Leistungsbeurteilung und -steigerung während der Entwicklung und des Betriebs hypermedialer Lernanwendungen eingesetzt werden können.

## 2.3 Klassifikation von Prefetchingverfahren und -konzepten

Die in der Literatur zu findenden Prefetchingkonzepte eignen sich unterschiedlich gut, Entwicklung und Betrieb verteilter hypermedialer Lernanwendungen zu unterstützen. Deshalb sind grundsätzlich verschiedene Konzepte zu identifizieren und diesbezüglich zu bewerten.

[46] *Bultermann* verweist in [Bul04] auf diesen Mangel und stellt ein Werkzeug dar, welches zur Erkennung von Performanceproblemen während der Entwicklung von Präsentationssequenzen im Rahmen des Web Engineerings von Hypermedien Verwendung finden kann. Er beschränkt sich jedoch hauptsächlich auf die Visualisierungsaspekte zur Unterstützung des Entwicklungsprozesses.

Dazu wird in diesem Abschnitt eine Möglichkeit zur Klassifikation dargestellt, welche sich von Klassifikationen anderer Autoren in [Wan99], [Dav02a] und [VL96] insbesondere dadurch unterscheiden, dass nicht die verwendeten Technologien oder die zugrunde liegende Rechnernetzarchitektur in den Mittelpunkt gestellt werden.[47] Eine solche Klassifikation erscheint hier ungeeignet. So mag zwar letztendlich mit der Realisierung eines konkreten Prefetchingverfahrens auch die zugrunde liegende Rechnernetzarchitektur wichtig sein, zur Konstruktion bzw. Auswahl eines Verfahrens für architekturunabhängige Anwendungsklassen wie Lernanwendungen spielt diese jedoch eine untergeordnete Rolle. Weiterhin wird in diesem Abschnitt auf das Zusammenspiel der für ein Prefetchingverfahren notwendigen Komponenten eingegangen und dargestellt, wie die Güte eines Prefetchingverfahrens in Hinblick auf Lernanwendungen zu bewerten ist. So lassen sich unter Berücksichtigung der im vorherigen Abschnitt beschriebenen Besonderheiten verteilter hypermedialer Lernanwendungen Schlussfolgerungen zur Konzeption von Prefetchingverfahren für diese ziehen.

### 2.3.1 Kategorien

Hier werden die Prefetchingverfahren in fünf unterschiedliche Kategorien in Abhängigkeit von den angewandten Konzepten zur Bestimmung der im Voraus anzufordernden Daten unterteilt. Es wird unterschieden zwischen

- idealen,
- vorhersagebasierten,
- schwellenbasierten,
- nutzergesteuerten und
- blinden Verfahren.

Die fünf Kategorien wurden gebildet, um Prefetchingverfahren in Hinblick auf ihre Eignung zur Realisierung von Zeittransparenz für verteilte Anwendungen unabhängig von konkreten Rechnernetzarchitekturen bewerten zu können. Die erste Kategorie „Ideale Prefetchingverfahren“ beinhaltet Verfahren rein theorethischer Natur, welche aus Überlegungen entstanden sind, was mit Hilfe von Prefetching machbar wäre.

Die Kategorien vorhersage- und schwellenbasierte Verfahren beinhalten zueinander konträre Sichtweisen. Verfahren beider Kategorien sind so aufgebaut, dass mit Hilfe von Algorithmen eine festgelegte Menge im Voraus stellbarer Anfragen bewertet wird. Vorhersagebasierte Verfahren schließen mit Hilfe von Bewertungsverfahren auf zukünftig wahrscheinliche Anfragen und stellen diese im Voraus. Schwellenbasierte Verfahren hingegen schließen mit Hilfe der Bewertung Anfragen aus, welche zukünftig zwar gestellt werden könnten, jedoch im Voraus nicht gestellt werden sollten, weil sie sich aufgrund der Bewertung nicht dafür eignen, und stellen

[47] In der Literatur ist eine Unterteilung der Verfahren in Abhängigkeit von der Rechnernetzarchitektur vorherrschend (vgl. [Dav02a], S. 61ff.). So wird zum Beispiel zwischen server-, proxy- und browserbasierten Prefetchingverfahren unterschieden. Eine solche Unterteilung ist sinnvoll, wenn die verwendeten Technologien im Mittelpunkt stehen.

nicht diese sondern andere Anfragen im Voraus. Nutzergesteuerte Verfahren überlassen es dem Nutzer einer verteilten Anwendung, interaktiv während der Nutzung über im Voraus anzufordernde Daten zu entscheiden. Diese Verfahren haben rein entscheidungsunterstützenden Charakter. Die Entscheidung ist Gegenstand der Anwendung selbst. Blinde Verfahren bewerten die im Voraus stellbaren Anfragen nicht. Verfahren aller fünf Klassen setzen voraus, dass eine bestimmte Menge im Voraus stellbarer Anfragen bekannt ist. Diese kann mit dem Laden von Daten in den Cache im Voraus bzw. vom Nutzer initiiert variieren.

Hier sollen die grundlegenden Ideen der Konzepte, auf welchen die Verfahren dieser Kategorien beruhen, dargestellt und in Hinblick auf den Einsatz für verteilte hypermediale Lernanwendungen bewertet werden. Es werden Kriterien zur Auswahl und Bewertung entsprechender Verfahren aufgeführt.

### Ideales Prefetching

*Kroeger* und *Long* untersuchen in [KLM97] mit Hilfe empirisch erhobener Daten und Simulationsexperimenten das Anwendungspotential von Prefetchingverfahren unter der Prämisse, dass alle zukünftigen an einen Cache gestellten Anfragen bekannt sind. Prefetchingverfahren erhalten in ihren Untersuchungen idealerweise vollständiges Wissen über alle zukünftigen Anfragen an den Cache. Sie beschreiben diesen Umstand als den am besten anzunehmenden Fall.

Unter dieser Annahme entsteht für beliebige Zielstellungen eine theoretische, untere Schranke. Erhält ein Prefetchingverfahren vollständiges Wissen über zukünftige Anfragen und nutzt es dieses optimal, dann stellt das Ergebnis der vom Verfahren gelieferten Entscheidungen eine untere Schranke für beliebige Prefetchingverfahren dar. Ein solches Verfahren wird als ideales Prefetchingverfahren bezeichnet.

**Bemerkung:** *Werden spezielle Cachestrategien bezüglich der Verwaltung begrenzter Ressourcen, insbesondere Speicherplatz, vernachlässigt, dann entscheidet ein ideales Prefetchingverfahren mit dem Ziel, die Summe der Wartezeiten auf alle Anfragen bzw. die Wartezeit auf jede einzelne Anfrage zu minimieren, der Art, dass die Antworten auf die im Voraus gestellten Anfragen vom Cache genau in der Reihenfolge gespeichert werden, in welcher diese zukünftig bei diesem Cache angefragt werden.*

Dieses bedeutet jedoch nicht, dass die Wartezeit zwangsläufig mit einer korrekten Vorhersage auf Null sinkt. Vom Cache bzw. von einem Prefetchingverfahren nicht beeinflussbare, aber sich auch auf die Wartezeit der an den Cache gestellten Anfragen auswirkend, sind

1. die Reihenfolge der Anfragen,
2. die Zeitdauer zwischen den gestellten Anfragen,
3. die kürzeste Zeitdauer, in welcher eine Antwort geliefert werden kann, wenn diese nicht vom Cache repliziert wurde,
4. die kürzeste Zeitdauer, in welcher eine Antwort geliefert werden kann, wenn diese beim Cache repliziert wurde.

Punkt 4 kann für eine Vielzahl von Anwendungen vernachlässigt werden. Die Zeitdauer, die für das Ausliefern durch den lokalen Cache benötigt wird, ist häufig für Anwendungen vernachlässigbar klein und spielt im Rahmen verteilter Rechnernetze keine Rolle. Die Problematik

bezieht sich für verteilte Lernanwendungen insbesondere auf die Bereitstellung der zu präsentierenden Lerninhalte, das Rendern aufwendig darstellbarer Inhalte.[48]

Die Punkte 1 und 2 werden von den Möglichkeiten nicht deterministischer Programmabläufe innerhalb einer Anwendung beeinflusst. Insbesondere menschliches Verhalten, das heißt Mensch-Maschine-Interaktion über Programmschnittstellen, können zu verschiedenen Anfragereihenfolgen sowie Zeitabständen zwischen den Anfragen an den Cache führen. Hier kommt dem Navigationsverhalten der Nutzer der verteilten Anwendung eine besondere Rolle zu. Die Zeitdauer zwischen gestellten Anfragen ist einerseits dazu zu verwenden, die an den Cache gestellten Anfragen durch diesen zu beantworten sowie gegebenenfalls die Antworten von entfernten Ressourcen zu erfragen. Andererseits kann die Zeitdauer dazu verwendet werden, Anfragen an entfernte Ressourcen im Voraus zu stellen - Zeit, in welcher ohne Prefetching Netzwerkressourcen ungenutzt blieben.

Zu Punkt 3 ist zu bemerken, dass die Zeitdauer zwischen Anfrage an und Antwort von einer entfernten Ressource von den zur Verfügung stehenden Ressourcen des Rechnernetzes sowie deren Auslastung beeinflusst wird. Die Zeitdauer kann nur dann von der Entscheidung eines Prefetchingverfahrens beeinflusst werden, wenn die Wahl einer entfernten Ressource und/oder eines Nachrichtenübertragungskanals zu dieser zur Entscheidung stehen würde. Solche Szenarien spielen für verteilte Lernanwendungen eine untergeordnete Rolle. Diese Betrachtungsweise ist auch in der Literatur zu Prefetchingverfahren für verteilte Systeme wie webbasierte Anwendungen nicht zu finden.

Für verteilte Lernanwendungen sind die ersten beiden Punkte von Bedeutung. Diese gestalten sich in Abhängigkeit vom Navigationsverhalten der Lernenden, so dass diesem eine besondere Rolle zukommt.

Die vier aufgeführten Punkte führen dazu, dass Wartezeiten auftreten können, trotz des Einsatzes eines imaginären, idealen Prefetchingverfahrens. Hinzu kommt der Umstand, auf welchen *Davidson* in [Dav02a], S.259 und *Duchamp* in [Duc99] hinweisen, dass nicht alle Anfragen sich dazu eignen, im Voraus gestellt zu werden und binden die Möglichkeit, dass Anfragen im Voraus stellbar sind, von ihnen als „prefetchability“ bezeichnet, an die Eigenschaft, dass die Antworten beim Cache zwischengespeichert werden könnten, „cacheability“ genannt. Sie unterscheiden zwischen „cache-“ und „prefetchability“. Jedoch genaue Anmerkungen zu dem, was sie unter „cacheability“ verstehen, sind nicht in den Ausführungen zu finden. Eine Studie, 1998 von *Feldmann* et al. erstellt (vgl. [FCD+99]), belegt, dass nur etwas mehr als 50 % der im WWW anforderbaren Dokumente sich zum Zwischenspeichern im Cache eignen. Als „non-cacheable“ werden von Ihnen Anfragen eingestuft, welche auf im WWW verwendete Technologien wie Cookies und cgi-Skripte basieren.

Hier soll nochmals betont werden, dass im Voraus stellbare und damit für Prefetching geeignete Anfragen

- im Voraus lieferbar sein müssen,

  das heißt lieferbar von der angefragten Ressource genau ab dem Zeitpunkt, ab welchem die entsprechende im Voraus gestellte Anfrage von der Ressource zu beantworten ist,

[48] Detaillierte Betrachtungen dazu sind in [RCM+96] und [RS03] zu finden.

- bis zum Eintreffen der korrespondierenden Anfrage beim Cache auch noch dieselbe Antwort gültig zu sein hat. Hier ist die Änderungshäufigkeit[49] von Bedeutung, welche *Davidson* und *Duchamp* sowie *Feldmann* et al. implizit auch als ein Maß für „cacheability" bzw. „non-cacheability" von Antworten fassen.

Beide Punkte unterstreichen nochmals den spekulativen Charakter von Prefetching. Damit ergibt sich für ideale Prefetchingverfahren die zusätzliche Forderung, dass diese zu berücksichtigen haben, ab welchem Zeitpunkt eine Anfrage mittels Cache im Voraus frühestens gestellt werden kann, so dass die korrespondierende Antwort auch jene ist, die durch den Cache, folgend auf eventuell an diesen zukünftig gestellte Anfragen, zu liefern ist. Dieses erfolgt in den oben genannten Betrachtungen zu idealen Prefetchingverfahren nicht. Ein weiterer Punkt sind die zu berücksichtigenden Cachestrategien. So wird in [Duc99] idealerweise ein beliebig großer Speicher des Caches für ideales Prefetching angenommen, so dass Ersetzungsstrategien der im Speicher verwalteten Daten keine Rolle für die Ergebnisse des Verfahrens spielen. Andere Autoren problematisieren Cachestrategien in ihren Studien nicht (vgl. zum Beispiel [KLM97]).

Deshalb sollen Verfahren, welche die folgenden beiden Punkte zusätzlich berücksichtigen, unter „streng idealen Prefetchingverfahren" gefasst werden:

- Der Zeitpunkt wird berücksichtigt, ab wann Daten im Voraus angefragt werden können, so dass zur korrespondierenden Anfrage an den Cache eine konsistente Antwort geliefert werden kann, und
- der Speicher des Caches wird als beliebig groß angenommen.[50]

*Kröger* und *Long* vergleichen die Szenarien „kein Einsatz von Cacheverfahren", „Einsatz von Cacheverfahren ohne Prefetching„ und „Einsatz idealer Prefetchingverfahren". Sie kommen auf Basis von Simulationsexperimenten in Anlehnung an empirisch erhobenes Datenmaterial bezüglich beobachtbarer Datenstrukturen und Nutzerverhalten im WWW zu dem Ergebnis, dass mit Hilfe von klassischen Cacheverfahren bzw. idealem Prefetching gegenüber dem Einsatz keiner dieser Verfahren von 20 bis zu über 50% der durchschnittlichen Wartezeiten reduziert werden können. Ideales Prefetching reduziert die Wartezeiten durchschnittlich um mehr als das Doppelte als die verwendeten klassischen Cacheverfahren. Auch andere empirische Untersuchungen dieser Art kommen zu ähnlichen Ergebnissen.[51] Dieses unterstreicht das Anwendungspotential von Prefetchingverfahren, wenn eine „gute" Vorhersage möglich ist.

### Vorhersagebasierte Verfahren

Erstaunlich ist, dass dieses Anwendungspotential auch für nicht ideale Prefetchingverfahren, welche kein Wissen über zukünftige Anfragen besitzen, nachgewiesen wurde. *Loon* und *Bagharghaven* nehmen Untersuchungen zur Unterstützung von Clienten durch so genanntes „Browsen" durch das WWW im Vergleich zu Szenarien „kein Einsatz von Cacheverfahren" und „Einsatz von Cacheverfahren ohne Prefetching" vor. Sie beschreiben eine Reduzierung der Wartezeit

[49] englisch update frequency

[50] In der Literatur werden diese beiden Punkte häufig vernachlässigt und unterschlagen. Es ist deshalb notwendig, darauf hinzuweisen und diesen Typ von Prefetchingverfahren explizit zu benennen.

[51] Vgl. [Dav04], [EJM00], [Duc99] und [KL97]

durch Prefetching um das 3 bis 7-fache, mehr als dieses mit klassischen Cacheverfahren im Vergleich zu keinem Einsatz von Cacheverfahren möglich ist. Sie verwenden ein Vorhersageverfahren, welches sich nur auf Daten der Vergangenheit stützt. Bemerkenswert an ihren Untersuchungsergebnissen in [LB97] sind zwei Aspekte. Der Einsatz von Cacheverfahren ohne Prefetching führt kaum zu reduzierten Wartezeiten. Weiterhin beziehen sich ihre Untersuchungen auf keine speziellen Anwendungen sondern ganz allgemein auf „Browsen“ durch das WWW, so dass für das Vorhersageverfahren auch keine speziellen Annahmen über anwendungsabhängiges Nutzerverhalten getroffen werden konnten. *Davidson* stellt ähnliche Untersuchungen in [Dav04] an, kommt aber nicht zu so drastischen Ergebnissen. Die von ihm verwendeten Vorhersageverfahren basieren auch auf Vergangenheitsdaten und verwenden ähnliche Konzepte wie *Loon* und *Baharghaven*.

Die Ursache für die unterschiedlichen Ergebnisse ist aus den Veröffentlichungen nicht ersichtlich. Es kann vermutet werden, dass das den Simulationsexperimenten zugrunde liegende Datenmaterial, auf welches hin ein bestimmtes Anfrageverhalten im WWW simuliert worden ist, zu unterschiedlichen Ergebnissen mit aber derselben Tendenz führte.[52] Die zwei oben aufgeführten Untersuchungen kommen jedoch auch für nicht ideale Prefetchingverfahren zu dem Ergebnis, dass eine erhebliche Reduktion der Wartezeiten gegenüber Cacheverfahren ohne Prefetching möglich ist. Diese Aussage wird auch von einer Vielzahl anderer Veröffentlichungen gestützt.[53]

In der Literatur lassen sich vielzählige, im Zusammenhang mit Prefetching eingesetzte Vorhersageverfahren finden. Ganz allgemein werden Techniken des maschinellen Lernens eingesetzt, um Nutzerverhalten abzubilden und vorherzusagen. *Mitchel* liefert in [Mit97] zu diesen Techniken einen umfassenden Überblick. Im Zusammenhang mit Prefetching erweisen sich Verfahren basierend auf der Analyse von Vergangenheitsdaten als besonders effektiv zur Vorhersage von Anfragen im WWW, welche überwiegend durch Nutzer initiiert werden und damit eng mit deren Verhalten verknüpft sind (vgl. [Dav04]). Diese auf Historien basierenden Verfahren[54] häufig werden Historien bzw. Nutzerprofile auf Grundlage beobachtbaren Verhaltens anhand von Logdaten erstellt spekulieren mit Hilfe in der Regel einfacher statistischer Verfahren über ein oder mehrere zukünftige durch einen Nutzer initiierte Anfragen.

Ein häufig verwendetet Maß zur Einschätzung der Güte von Vorhersageverfahren ist die Vorhersagegenauigkeit[55] (vgl. [ZA01], [SOBB03], [DJX02], [EJM00], [CKR99] und [CW99]). Darunter soll hier in Anlehnung an *Zukerman* und *Albrecht* (vgl. [ZA01]) der Anteil aller korrekt vorhergesagten Anfragen, das heißt Anfragen, welche vorhergesagt wurden und dann auch wie vorhergesagt gestellt wurden, zu allen gestellten Anfragen über einen festgelegten Zeitraum verstanden werden.[56]

Die für Prefetching eingesetzten Vorhersageverfahren unterscheiden sich durch

[52] Das stellt die Vergleichbarkeit dieser und anderer Untersuchungen in Frage.

[53] Nahezu jede Veröffentlichung eines Verfahrens führte auch zu einer empirischen Untersuchung der Leistungsfähigkeit des Verfahrens. Untersuchungen zur Vergleichbarkeit der Ergebnisse sind leider nicht publiziert worden. Simulationsstudien bis 1999 wertet *Wang* in [Wan99] aus. Spätere Studien liefern ähnliche Ergebnisse (vgl. [YK00], [EJM00], [Dav02b], [BKK03] und [LL02]).

[54] englisch history-based prediction

[55] „accuracy“ oder auch „precision“ genannt

[56] In [ZA01] wird ein Überblick zur Evaluation von Vorhersageverfahren mittels unterschiedlicher Metriken gegeben.

- verwendete statistische Verfahren

  *Zuckerman* und *Albrecht* unterteilen die verwendeten statistischen Verfahren in Abhängigkeit von den verwendeten Methoden in sieben unterschiedliche Kategorien (vgl. [ZA01]).

  In der Literatur beschriebene Ansätze bedienen sich überwiegend empirischer Häufigkeitsverteilungen vergangener Anfragen bzw. Anfragefolgen. Diese werden dazu verwendet, die zu erwartenden Wahrscheinlichkeiten möglicher zukünftiger Anfragen bzw. Anfragereihenfolgen zu ermitteln.[57]

- berücksichtigte Anfragen bzw. Anfragereihenfolgen

  Weiterhin unterscheiden sich die Verfahren in der Zahl der berücksichtigten Anfragen aus der Vergangenheit. Auch spielt die berücksichtigte Anzahl der auf eine Anfrage folgenden Anfragen und deren Reihenfolge eine wichtige Rolle.

  Eine Vielzahl von Verfahren berücksichtigt jeweils nur Anfragereihenfolgen der Länge zwei. Das heißt, sie berücksichtigen mit ihrer Vorhersage jeweils nur die auf in der Vergangenheit gestellte Anfragen jeweils folgende Anfrage. [58] *Pirolli* und *Pitkow*, *Sen* und *Hansen* sowie *Zukerman*, *Albrecht* und *Nicholson* haben mit ihren Untersuchungen zu längeren Anfragereihenfolgen die nahe liegende Vermutung bestätigt, dass die Berücksichtigung längerer Anfragereihenfolgen von drei und vier zu genaueren Vorhersagen führt (vgl. [PP99], [SH03] und [ZAN99]). Auch ist zu vermuten, dass die Berücksichtigung immer längerer Anfragereihenfolgen nicht immer auch zu genaueren Vorhersageergebnissen führt. Je nach Anwendungszusammenhang wird die Zahl der durch Nutzer initiiert aufeinander folgenden Anfragen, welche sich aufgrund der Anwendung inhaltlich sowie zeitlich sinnvoll zusammenfassen lassen, unterschiedlich groß sein. So kann zum Beispiel vermutet werden, dass zwischen zwei zeitlich voneinander weit auseinander liegenden aber direkt aufeinander folgenden Anfragen, initiiert durch ein und denselben Nutzer, in Abhängigkeit von der Anwendungssituation kein Zusammenhang besteht.

  Je nach Zahl der zu berücksichtigenden Anfragen und Länge der berücksichtigten Anfragereihenfolgen werden unterschiedliche Datenstrukturen zu deren Speicherung eingesetzt. Bei großen Datenmengen ist die Speicherung, gegebenenfalls die Aggregierung der Daten sowie der effiziente Zugriff darauf, ein nicht zu vernachlässigendes Problem. Zum Vermerken von Anfragereihenfolgen sowie der Häufigkeit des Auftretens in der Vergangenheit werden baumbasierte Datenstrukturen eingesetzt, welche in der Tiefe des Baums je nach Länge der zu berücksichtigenden Anfragereihenfolgen beschränkt sind.

- Vorhersagefenster

  Ein wesentliches Problem besteht darin, das so genannte Vorhersagefenster festzulegen, das heißt, die Zahl vorherzusagender Anfragen bzw. der Zeitraum, für welches zukünfige

[57] Verfahren dieser Art, absteigend sortiert nach deren Veröffentlichungsdatum ab Mitte der 90er Jahre, lassen sich finden in: [BKK04], [CZ03], [BKK03], [SOBB03], [DJX02], [FMK02], [NKM03], [GZ01],[Sar00], [Duc99], [FCJ99], [HRBA99], [ZAN99], [PM99a], [PM99b], [JC98], [JFM97], [Vit96] und [PM96]. Auffällig ist die häufige Verwendung von Markow-Modellen zur Bestimmung zu erwartender Wahrscheinlichkeiten der nächsten Anfragen bzw. Anfragereihenfolgen. Die Verfahren weisen teilweise nur marginale Unterschiede auf.

[58] Dazu gehören Verfahren aus [LL02], [FS00], [SR00], [LBO99], [Duc99], [CKR99], [NZA98], [JK98], [Bes96], [PM96] und [Pad95].

Anfragen vorherzusagen sind. Die Verfahren lassen sich in der Regel entsprechend konfigurieren. Sie enthalten zu konfigurierende Parameter, mit welchen die Anzahl der unmittelbar als nächstes folgenden Anfragen festgelegt werden kann. Mit der Bewertung von Vorhersageverfahren wird jedoch häufig nur die Vorhersage der unmittelbar als nächstes folgenden Anfrage berücksichtigt (vgl. zum Beispiel [DJX02] und [Sar00]).

- Vorhersage von Anfragen einzelner Nutzer oder Nutzergruppen

  *Davidson* unterscheidet zwischen der Mikro- und der Makrosicht bei der Bewertung von Verfahren (vgl. [Dav04], S. 4). So existieren mikroskopisch angelegte Verfahren, welche darauf zugeschnitten sind, Anfragen aus vergangenen Anfragen einzelner Nutzer zu generieren. Hier sind in der Regel kleine Datenmengen zu berücksichtigen. Anfragen aufgrund der Eigenschaften eines einzelnen Nutzers, welche sich in den von ihm in der Vergangenheit initiierten Anfragen widerspiegeln, sind vorherzusagen. Unter der makroskopischen Sicht hingegen sind jene Verfahren zu fassen, welche in der Regel eine große Zahl vergangener Anfragen, initiiert durch unterschiedliche Nutzer, berücksichtigt. Beim Erzeugen von Vorhersagen wird nicht explizit berücksichtigt, welcher Nutzer, welche vergangene Anfrage initiiert hat, so dass kein spezifisches Verhalten des Einzelnen Eingang in der Vorhersage findet. Je nach Problemlage sind die unterschiedlichen Sichten vorteilhaft. Praktisch wurden überwiegend Verfahren der makroskopischen Sicht eingesetzt. Sie eigenen sich insbesondere zur Unterstützung von Prefetching für Web-Proxys und Web-Server. Der Fokus auf makroskopische Verfahren beruht wahrscheinlich auf dem wissenschaftlichen Arbeitsumfeld der Forschergruppen sowie der Förderung bestimmter Zielsetzungen durch die Wirtschaft.

All diese auf Historien basierenden Verfahren besitzen einen gemeinsamen Nachteil: Sie können nicht Anfragen vorhersagen, welche niemals vorher beobachtet wurden.

So lassen sich einige wenige andere Ansätze finden, welche auf der Analyse der Inhalte der angeforderten Daten beruhen. So beschreiben zum Beispiel *Chinen* und *Yamaguchi* in [CY97] einen sehr einfach gehaltenen Ansatz, in welchem die Reihenfolge zukünftiger Anfragen aus der Reihenfolge der zuletzt angeforderten HTML-Dokumente generiert wird. In [Dav02b] und [ESGS98] werden Ansätze beschrieben, in welchen die inhaltliche Ähnlichkeit bereits angeforderter Web-Dokumente mit noch nicht angeforderten verglichen wird. Anhand eines Ähnlichkeitsmaßes wird über die als nächstes im Voraus anzufordernden Web-Dokumente entschieden.

#### Schwellenbasierte Verfahren

Ein weiterer grundsätzlich von den vorhersagebasierten Verfahren zu unterscheidender Ansatz ist die Bewertung im Voraus anforderbarer Daten mittels so genannter Schwellen, hier schwellenbasierte Verfahren genannt. Diesen Verfahren liegt die Idee zugrunde, im Voraus anforderbare Daten mit Hilfe von unabhängig vom Navigationsverhalten gestalteten Heuristiken unabhängig vom Navigationsverhalten zu bewerten sowie nur solche Daten im Voraus anzufordern, welche einer oder mehreren Bedingungen bezüglich einer Bewertung genügen. Im Voraus stellbaren Anfragen werden über ein Bewertungsverfahren zahlenmäßige Werte zugeordnet. Die Bewertung erfolgt aufgrund von Eigenschaften zukünftiger Anfragen, welche nicht mit dem Navigationsverhalten sondern der Auslastung von Ressourcen im Zusammenhang stehen,

zum Beispiel dem Umfang der auf eine Anfrage hin zu übertragenden Daten. Unter- bzw. überschreitet[59] die Bewertung einer Anfrage eine Schwelle, dann wird diese als nicht sinnvoll im Voraus zu stellen eingestuft und nicht im Voraus gestellt. Der Wert der Schwelle ist entweder fix, geht damit als festzulegender Parameter in das schwellenbasierte Verfahren ein, oder er gestaltet sich variabel in Abhängigkeit von einer Funktion, der Schwellenwertfunktion. Diese wird dazu verwendet, dynamisch, das heißt abhängig von sich im Zeitablauf verändernden Parametern, einen Schwellenwert zu generieren. Sie drückt Abhängigkeiten von sich verändernden Rahmenbedingungen während des Betriebs aus, wie z.B. Auslastung der Netzwerkressourcen oder anfallende Kosten für den Nachrichtentransport.[60]

*Jiang*,*Wu* und *Shu* beschreiben ein schwellenbasiertes Verfahren, welches im Voraus anforderbare Datenobjekte bezüglich ihrer Größe bewertet (vgl. [JWS02]). Daraus lässt sich auf benötigte Kapazitäten für deren Übertragung sowie auf möglicherweise anfallende Wartezeiten in einer verteilten Anwendung schließen. Sie führen den Begriff der Kosten für negative Effekte durch Prefetching, hier Seiteneffekte genannt, und den Begriff des Benefits zur Beschreibung des zu erwartenden Nutzens durch Prefetching ein. Sie bewerten Daten in Bezug auf das Verhältnis zwischen Kosten und Benefit. Es sind nur solche Daten im Voraus anzufordern, deren Verhältnis einen bestimmten Wert, den Schwellenwert, nicht übersteigt.[61] Von *Angermann* wird ein ähnlicher Ansatz untersucht in [Ang03]. In diesem werden die anfallenden Kosten durch das Anfordern und Übertragen von Daten explizit modelliert. Die in [JWS02] und [Ang03] beschriebene Reduktion der Wartezeiten entspricht ungefähr den im vorherigen Abschnitt beschriebenen Ergebnissen für Prefetching, unterstützt mittels Vorhersageverfahren.

Vorhersagebasierte und schwellenbasierte Verfahren sind zwei sich ergänzende Ansätze, welche zwei unterschiedliche Herangehensweisen darstellen. Mit vorhersagebasierten Verfahren wird aus der Menge im Voraus stellbarer Anfragen die Teilmenge von Anfragen gesucht, welche mit hoher Wahrscheinlichkeit zukünftig gestellt wird. Die Wahrscheinlichkeit wird hypothetisch in Abhängigkeit von empirischen Wahrscheinlichkeitsverteilungen der Vergangenheit bestimmt. Schwellenbasierte Verfahren schließen hingegen aus der Menge im Voraus stellbarer Anfragen jene Anfragen aus, welche in Bezug auf eine Heuristik nicht im Voraus geladen werden sollten, da der zu erwartende Benefit in Bezug auf die entstehenden Kosten zu gering ist. Sie bringen im Gegensatz zu den vorhersagebasierten Verfahren die im Voraus zu stellenden Anfragen in keine Rangfolge. Eine Kombination aus Schwellen- und Vorhersageverfahren erscheint aufgrund des sich ergänzenden Charakters beider Ansätze sinnvoll. In [AZN99] und [VYK+02] werden Verfahren beschrieben, welche Vorhersagen mit Hilfe schwellenbasierter Verfahren bewerten. Das führt dazu, dass nicht jede als wahrscheinlich zukünftig eintreffend vorhergesagte Anfrage auch im Voraus gestellt wird. Die zu erwartende Wahrscheinlichkeit, mit der diese auch zukünftig gestellt wird, wird in Relation zu dem zu erwartenden Nutzen durch das vorzeitige Anfordern der Daten betrachtet.

Die in dieser Arbeit entwickelten Ansätze enstanden in Anlehnung an die Idee der schwellenbasierten Verfahren. Es wird jedoch keine Schwelle zum Ausschluss von potentiell im Voraus stellbaren Anfragen verwendet, sondern stattdessen ein Optimalitätskriterium eingeführt. Die Entscheidung über im Voraus anzufordernde Daten wird als Optimierungsproblem aufgefasst.

[59] Je nach Verfahren

[60] Verfahren dieser Art werden in [JWS02] und [VYK+02] dargestellt. Der allgemein beschriebene Ansatz schwellenbasierter Verfahren beruht auf den in diesen Publikationen dargestellten Ideen.

[61] In [FS00] ist ein ähnlicher Ansatz zur Verwaltung der im Cache gespeicherten Daten zu finden.

Es kann in Abhängigkeit von der Probleminstanz des noch zu formulierenden Optimierungsproblems, die auch aus allen potentiell möglichen Anfragereihenfolgen und bisher gestellten Anfragen besteht, zwischen Anfragen unterschieden werden, welche zum Erreichen des Optimums zwingend im Voraus gestellt werden müssten und solchen, für welche dieses nicht gilt. Dieser Ansatz kann mit Vorhersageverfahren kombiniert werden. Dazu in den jeweiligen Kapiteln mehr.

### Nutzergesteuerte Verfahren

Neben der Bewertung im Voraus anforderbarer Daten durch Algorithmen wird in [EJM00] und [JFM97] ein alternativer Ansatz verfolgt. Die Entscheidung über die im Voraus anzufordernden Daten zu bestimmten Zeitpunkten werden dem Nutzer während der Nutzung der Anwendung überlassen. Die Entscheidung wird durch Aufbereiten von Informationen über die im Voraus anforderbaren Daten sowie deren anwendungsbezogene Präsentation unterstützt.

*Eden*, *Joh* und *Mudge* präferieren einen simplen Mechanismus für Hypertext strukturierte Webdokumente. Der Nutzer kann mit diesem Mechanismus während der Verwendung eines Dokuments Hyperlinks in diesem als im Voraus anzufordern markieren. Ein im Hintergrund ablaufender Prozess nimmt diese Markierungen entgegen. Er fordert die sich hinter den markierten Links verbergenden Daten entsprechend an, soweit sie nicht schon im Cache gespeichert sind. Die Autoren betonen, dass durch solche Mechanismen eine Vorhersagegenauigkeit von 100% erreicht würde, was zu bezweifeln ist[62]. Die zu erwartende hohe Genauigkeit der Verfahren, wenn auch nicht bei 100%, stellt einen wesentlichen Vorteil gegenüber Verfahren der Kategorien der vorhersage- und schwellenbasierten Verfahren dar. Jedoch mit der Zielstellung, zeittransparente verteilte Lernanwendungen zu realisieren, ist ein solcher Ansatz nur eingeschränkt verwendbar. Wie stark Lernprozesse in auf diese Weise unterstützten Anwendungen durch Erzwingen vorzeitiger Navigationsentscheidungen beeinflusst werden, wäre zu untersuchen. Spielt die Zeittransparenz eine untergeordnete Rolle oder wird gar der entfernte Zugriff auf Ressourcen mit der Navigationsschnittstelle absichtlich für den Nutzer sichtbar dargestellt, wie zum Beispiel in [ADW01] für Dienste in drahtlosen Rechnernetzen, kann dieser Ansatz viel versprechend eingesetzt werden.

### Blinde Verfahren

Der Vollständigkeit halber ist eine Kategorie notwendig, welche eine Vielzahl von Verfahren beinhaltet, die ohne die im Voraus anforderbaren Daten zu bewerten, relativ wahllos und willkürlich auf diese zugreifen.

So werden in [Lie97], [CY97], [Kle99] und [TX00] Verfahren beschrieben, welche die zuletzt durch den Nutzer initiiert angeforderten Daten inhaltlich analysieren und die in ihnen enthaltenen Links auf entfernte Ressourcen als einzige Informationsquelle zum Generieren der folgenden im Voraus zu stellenden Anfragen nutzen. Die Menge der Links sowie der Umfang der Daten, welche diese referenzieren, kann je nach Anwendungszusammenhang sehr groß sein. Es ist leicht einzusehen und dementsprechend herrscht in der wissenschaftlichen Literatur nahezu

[62] Dies würde bedeuten, dass der Nutzer im Voraus seinen vollständigen „Weg“ durch ein Hypermedium planen könnte. Solch ein zielgerichtetes Navigationsverhalten kann typischerweise nicht beobachtet werden.

übereinstimmend die Meinung vor, dass solch aggressive Strategien zu erheblichen Seiteneffekten führen. Verfahren dieser Art sind nur bedingt einsetzbar und belasten in der Regel entfernte, vielfach gleichzeitig genutzte Ressourcen erheblich, insbesondere dann, wenn sie systematisch in großem Umfang eingesetzt werden.

Aus den Betrachtungen der vielzähligen, in den letzten 8 Jahren publizierten Prefetchingverfahren ergibt sich folgendes Bild. Es ist zwischen informierten und uninformierten Verfahren zu unterscheiden. Zu den informierten Verfahren sind die schwellenbasierten, vorhersagebasierten, nutzergesteuerten und idealen Prefetchingverfahren zu zählen, wobei die idealen Prefetchingverfahren praktisch keine Rolle spielen. Die Entscheidungen dieser Verfahren über im Voraus anzufordernde Daten basieren nicht nur auf grundsätzlich notwendigen Informationen, welche Daten im Voraus angefordert werden könnten, sondern auch auf weitergehende Bewertung bekannter Anfragemöglichkeiten. Uninformierte Verfahren hingegen, auch als blinde Verfahren bezeichnet, treffen Entscheidungen nur auf Basis der bekannten Menge im Voraus anforderbarer Daten. Zur sinnvollen Unterstützung verteilter Lernanwendungen kommen nur schwellen- sowie vorhersagebasierte Verfahren in die engere Wahl.

### 2.3.2 Komponenten

Aus den obigen Betrachtungen von Prefetchingverfahren ergibt sich der in Abbildung 2.1 dargestellte Vorschlag für das grobe Zusammenspiel einzelner Komponenten zur Umsetzung von Prefetching für verteilte Anwendungen bzw. Anwendungssysteme auf der Anwendungsschicht nach dem TCP/IP-Referenzmodell. Die Komponenten haben drei Funktionalitätsbereiche zur Umsetzung eines Prefetchingkonzepts abzudecken:

- Beobachtung und Analyse des Datenstroms zwischen lokalen Anwendungskomponenten und Transportschicht,
- Bewertung der bekannten im Voraus stellbaren Anfragen (optional) ,
- Selektion und Stellen der Anfragen.

In Abbildung 2.1 sind die drei Komponenten Beobachter, Prefetcher und Bewerter zur Realisierung von Prefetching grau hinterlegt.

- Beobachter

  Der Beobachter hat drei Funktionalitäten abzudecken.

  – Es sind die Eigenschaften der durch die lokalen Anwendungskomponenten initiierten Anfragen an den lokalen Cache sowie die vom lokalen Cache gelieferten Antworten zu vermerken. Die zu registrierenden Eigenschaften sind in Abhängigkeit von den benötigten Daten des Bewerters und Prefetchers zu wählen, wie zum Beispiel die Reihefolge oder genauer die Zeitpunkte, in welcher bzw. zu welchen die Anfragen gestellt wurden.

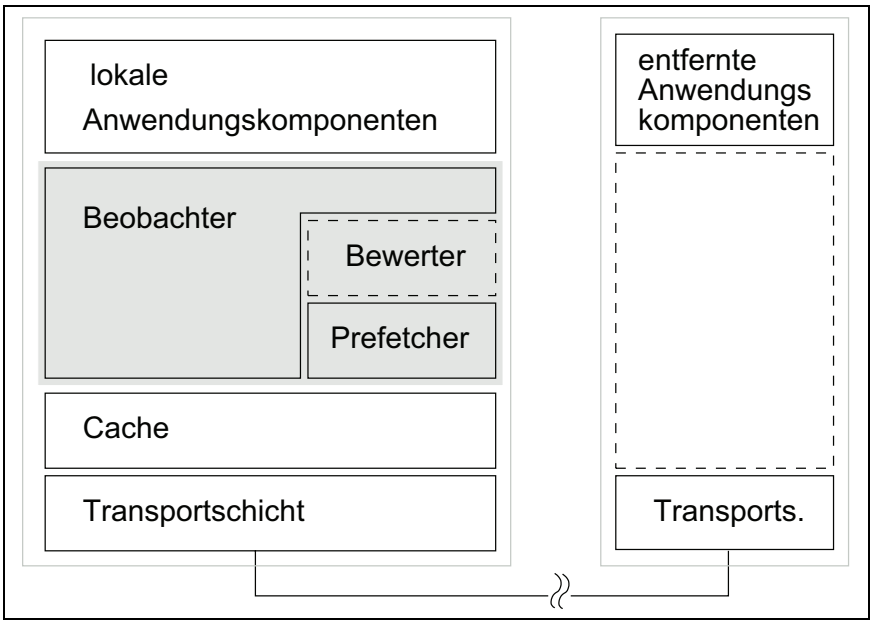



Abbildung 2.1: Komponenten zur Realisierung von Prefetching für verteilte Anwendungen im Schichtenmodell, links die Sicht auf lokale Komponenten, rechts die Sicht auf entfernte Komponenten angedeutet

  - Es sind alle durch die lokalen Anwendungskomponenten stellbaren Anfragen zu verwalten und gegebenenfalls zu protokollieren. Prefetcher wie Bewerter erhalten über den Beobachter Zugriff auf eine bestimmte Menge durch lokale Anwendungskomponenten stellbare Anfragen. Der Beobachter kann diese Informationen durch Analyse der Antworten des lokalen Caches ermitteln und/oder durch direkten Zugriff auf Daten, welche diese Informationen enthalten. So ist denkbar, dass vor Betrieb der Anwendung alle im Voraus stellbaren Anfragen bekannt sind und diese an den lokalen Beobachter übermittelt werden. Dieses kann zum Beispiel sinnvoll mit dem Beginn der Ausführung der lokalen Anwendungskomponenten erfolgen.
  - Zur Umsetzung der ersten beiden Funktionalitäten ist der Bitstrom zwischen jeweiliger Anwendungskomponente und Cache zu interpretieren. In ihm enthaltene Anfragen der Anwendungskomponente sowie Antworten des Caches sind zu identifizieren. Damit gestaltet sich diese Funktionalität, im Gegensatz zu den anderen beiden, abhängig von der jeweiligen Anwendungskomponente bzw. vom verwendeten Anwendungsprotokoll.

- Bewerter

  Der Bewerter liefert dem Prefetcher auf Basis der Daten des Beobachters Bewertungen der durch die lokalen Anwendungskomponenten an den Cache stellbaren Anfragen. Er ist einzusetzen, wenn das umgesetzte Prefetchingverfahren eine Bewertung vornimmt.

  Die in der Abbildung dargestellte Schichtung erlaubt keine Schnittstelle zwischen Anwendungskomponenten und Bewerter. Für verteilte Anwendungen ist dies nicht notwendig. Allein für nutzergesteuerte Verfahren ist eine Schnittstelle zwischen Anwendungskomponente und Bewerter sinnvoll. Mit ihnen werden Teile der Bewertungsfunktiona-

litäten an die Anwendungskomponente ausgelagert. Jedoch werden nutzergesteuerte Bewertungsverfahren hier für verteilte Lernanwendungen im Rahmen einer guten Zeittransparenz abgelehnt.[63] Nur für solche wäre eine Schnittstelle zwischen Anwendungskomponenten und Bewerter sinnvoll.

- Prefetcher

  Dem Prefetcher kommt das eigentliche Initiieren der Anfragen im Voraus an den lokalen Cache zu. Er beinhaltet den eigentlichen Entscheidungsmechanismus, die Realisierung eines Zielkriteriums, nach welchem zu entscheiden ist, welche Daten, in welchem Umfang, zu welchem Zeitpunkt im Voraus anzufordern sind. Das erfolgt auf Basis der vom Beobachter ermittelten Daten zu den durch lokale Komponenten stellbaren Anfragen, deren Bewertungen durch den Bewerter und den Informationen des Caches darüber, welche Daten bereits lokal verfügbar sind und welche Antworten zukünftig in ihm abgelegt werden.

Die in Abbildung 2.1 dargestellte Schichtung erzwingt, dass ein Zugriff auf entfernte Anwendungskomponenten nur mittelbar über den Beobachter zu erfolgen hat. Das ist sinnvoll, um den Zugriff auf lokale wie entfernte Daten transparent zu gestalten und weil nur so eine genaue Beobachtung von Anfrageverhalten lokaler Anwendungskomponenten möglich ist. Alleinige Beobachtung des Zugriffs auf entfernte Komponenten mittels dieser selbst führt je nach beobachteten Eigenschaften zu drastischen Verfälschungen (vgl. [Dav99]). Sind Beobachtungen lokaler als auch entfernter Anwendungskomponenten für Bewerter und/oder Prefetcher von Bedeutung, dann ist ein Datenaustausch zwischen einzelnen Beobachtern zu realisieren.

Kritisch zu beurteilen ist die Trennung zwischen Prefetcher und Cache. Auswirkungen von Entscheidungen des Prefetchers als auch des Caches sind in der Regel voneinander abhängig. Deshalb ist dem Cache mit einer Anfrage zu übermitteln, ob diese durch lokale Anwendungskomponenten oder durch den Prefetcher initiiert wurde. So sind Cachestrategien, welche dieses berücksichtigen, zur Verwaltung der knappen Ressourcen des Caches, insbesondere Speicherplatz, möglich. Aus diesem Grund ist auch der lokale Speicher des Caches für im Voraus angeforderte Daten des Prefetchers zu nutzen. Eine Trennung der Daten in im Voraus angeforderte und von der Anwendung initiiert angeforderte Daten erscheint nicht, wie in [Zha01] und [JC98] dargestellt, sinnvoll; im Gegenteil, in [CK01b] werden diebezüglich spezielle Strategien zum Cache-Speichermanagement unter Berücksichtigung vom Prefetching diskutiert.

Das vorgeschlagene Modell erzwingt im Gegensatz zu anderen Vorschlägen keine konkrete Anwendungsarchitektur sowie Verwendung bestimmter Prefetchingverfahren.[64] Die lokalen Anwendungskomponenten können beliebige Komponenten einer verteilten Anwendung wie Web-Server oder Browser in einer Client-Server-Architektur sein. Beobachter, Bewerter und Prefetcher können informierte als auch uninformierte Prefetchingverfahren realisieren.

### 2.3.3 Gütemaße

Zum Abschluss dieses Überblicks zu Prefetching sollen kurz Metriken zur Beurteilung der Güte erwähnt werden. Die Güte von Prefetchingverfahren soll hier primär an deren Leistung gemes-

[63] Siehe Diskussion der Zeittransparenz für verteilte Lernanwendungen, Abschnitt 2.1.1, S.11

[64] Andere Architekturvorschläge sind zu finden in [Wan99], [Duc99], [ESGS98] und [LB97].

sen werden. Als Indikator für die Leistung soll die durch Einsatz des zu bewertenden Prefetchingverfahrens reduzierte Wartezeit dienen. Neben der Leistung werden weitere Kriterien zur Bewertung der durch ein Prefetchingverfahren erzeugten und keinesfalls zu vernachlässigenden, negativen Seiteneffekte herangezogen (vgl. [ESGS98] und [JC98]). Neben der Beurteilung der Leistung und Seiteneffekte sind weitere Eigenschaften, insbesondere mit der Realisierung im Zuge einer konkreten zugrunde liegenden Architektur, wichtig. Diese sollen hier nicht Gegenstand sein, werden aber von *Wang* in [Wan99] ganz allgemein in Bezug auf Cacheverfahren und -Architekturen diskutiert.[65]

**Leistung**

Das Antwortzeitverhalten einer Anwendung kann als Indikator dafür herangezogen werden, was eine Anwendung oder deren Komponenten entsprechend der Spezifikation leisten (vgl. [SW02], S.3ff.). Genauso heben *Coulouris*, *Dollimore* und *Kindberg* die Latenz als eine wesentliche Leistungseigenschaft für die Kommunikation in verteilten Rechnernetzen hervor (vgl. [CDK02], S.71). Mit dem Einsatz von Prefetching wird primär das Ziel verfolgt, die Wartezeit bzw. Antwortzeit auf an den Cache gestellte Anfragen zu reduzieren. So liegt es nahe, die Leistung von Prefetchingverfahren anhand der Wartezeit zu beurteilen. Eine Vielzahl von Simulationsexperimenten folgen diesem Ansatz und beurteilen Prefetchingverfahren mit Hilfe von Maßen, welche sich auf die Wartezeiten einer festgelegten Menge von Anfragen an den Cache beziehen.

Sinnvoll erscheinen je nach zu unterstützender Anwendung zum Beispiel

- die durchschnittliche Wartezeit,[66]
- die maximal auftretende Wartezeit und
- die Summe der Wartezeiten.

Werden in der Literatur auf die Wartezeit bezogene Maße verwendet, dann beziehen sich die Autoren nahezu ausschließlich auf die durchschnittliche Wartezeit (vgl. [Bes96], [PM96], [Duc99], [LB97] und [ADW01]).

Zur eigentlichen Bewertung spielen die absoluten Größen der Maßzahlen eine untergeordnete Rolle. Wichtig ist der Vergleich zu anderen Prefetchingverfahren bzw. zu keinem Einsatz eines solchen. Um relative Maße zum besseren Vergleich zwischen den Verfahren zu erhalten, werden die Ergebnisse zweier Szenarien, der Einsatz des zu bewertenden Verfahrens mit einem Vergleichsverfahren, in Relation zueinander gesetzt. In der Literatur werden als Vergleichsverfahren nahezu ausschließlich klassische Cacheverfahren ohne Prefetching verwendet[67].

Bezogen auf die Wartezeit wird z.B. von *Jacobson* und *Cao* die relative Wartezeitreduktion[68] verwendet. Sie definieren diese wie folgt:

[65] Dazu gehören typische auch in [Tan03] und [CDK02] diskutierte Eigenschaften verteilter Anwendungen und Rechnernetze: Stabilität, Robustheit, Skalierbarkeit, Transparenz, Effizienz etc.

[66] z.B. als arithmetisches Mittel

[67] Siehe zum Beispiel [JC98] und [FCD+99]

[68] Von ihnen „latency reduction“ genannt

$$\textit{relative Wartezeitreduktion} = \left(1 - \frac{\textit{Wartezeit mit Prefetching}}{\textit{Wartezeit ohne Prefetching}}\right) \tag{2.1}$$

Sie beschreibt, um welchen Anteil die aus dem Vergleichsverfahren resultierende Wartezeit reduziert werden kann, wenn stattdessen das zu bewertende Pretching-Verfahren eingesetzt wird. *Jacobson* und *Cao* verwenden in obiger Formel für die Wartezeit die durschnittlichen Wartezeiten, genauer das arithmetische Mittel über die Wartezeiten aller Anfragen. Da die verwendeten Vergleichsverfahren sich jedoch voneinander unterscheiden bzw. unzureichend dargestellt werden, ist ein Vergleich der in den Studien angegebenen relativen Maßzahlen häufig genauso fragwürdig wie der Vergleich der Absolutwerte. *Davidson* beschreibt zwar in [Dav02a] ein umfassendes Framework zur Evaluation von Cacheverfahren, greift dieses Problem aber nicht auf.

Die Wartezeit als alleiniger Indikator zur Bewertung der Leistung eines Prefetchingverfahrens ist unzureichend. In der Wartezeit, relativ oder absolut, stecken keine Informationen über die zur Verfügung stehenden Ressourcen, welche verwendet wurden oder verwendet werden konnten, um das Ziel zu erreichen. Die Nutzzeit, die Zeit, in welcher von verteilten Anwendungskomponenten angeforderte Daten bearbeitet bzw. genutzt und keine Anfragen an den Cache generiert werden, darf nicht vernachlässigt werden. Während der Nutzung können Ressourcen, welche außerhalb der Nutzung zum Beantworten der von Anwendungskomponenten initiierten Anfragen bereitgestellt werden, ausschließlich für im Voraus gestellte Anfragen verwendet werden. *Loon* und *Bharghavan* sowie *Khan* und *Tao* berücksichtigen deshalb die Nutzzeit[69] in ihren Untersuchungen [LB97] und [KT01]. Ein Beurteilungskriterium aus Nutz- und demgegenüber stehender Wartezeit ist für Lernanwendungen zwingend notwendig.

Aus diesem Grund erscheint als weiteres Kriterium zur Leistungsbeurteilung die Gesamtzeit einer Anfrage als Summe aus der zu einer Anfrage korrespondieren Wartezeit und Nutzzeit der durch diese angeforderten Daten sinnvoll.

Jeweils bezogen auf eine bestimmte Menge von Anfragen kann auch hier

- die durchschnittliche Gesamtzeit,
- die maximal auftretende Gesamtzeit bzw.
- die Summe der Gesamtzeiten

zur Beurteilung herangezogen werden.

Hier sei nochmals bemerkt, dass insbesondere für den Entwurf und Betrieb von Lernanwendungen die Gesamtzeit, die ein Nutzer der Anwendung aufbringen muss, um eine bestimmte Menge von Daten, das heißt Lerninhalte, zu bearbeiten, zu berücksichtigen ist.

Neben direkt auf die Wartezeit bezogene Maße werden überwiegend indirekte Maße verwendet. Dies sind Cache-Treffer- und Cache-Fehler-Rate[70], die auch zur Beurteilung von Cacheverfahren ohne Prefetching in [DMF97] und [KV98] Anwendung finden.[71]

[69] englisch usage time

[70] englisch cache hit rate bzw. cache miss rate

[71] Andere verwendete Bezeichnungen für dieselben Maße sind: Treffgenauigkeit (accuracy, vgl. [EJM00] und [LD97]), sichere Anfragen (request savings, vgl. [JC98]) und Prefetching Effektivität (vgl. [BKK03]).

Die Raten ergeben sich aus folgenden relativen Häufigkeiten nach [ESGS98]:

$$Trefferrate = \frac{Zahl\ der\ Treffer}{Zahl\ aller\ Anfragen} \tag{2.2}$$

$$Fehlerrate = \frac{Zahl\ der\ Fehler}{Zahl\ aller\ Anfragen} \tag{2.3}$$

Eine Anfrage an den Cache ist ein Treffer, falls die korrespondierende Antwort ohne Zugriff auf entfernte Ressourcen geliefert werden kann, und ein Fehler, wenn mindestens ein entfernter Zugriff notwendig ist. Alle anderen Anfragen sind Fehler ($Trefferrate + Fehlerrate = 1$).[72] Mit Hilfe dieser Maße kann die Leistung bezüglich der oben genannten primären Zielstellung nur bedingt bewertet werden. Zusätzliche Informationen, um auf die Wartezeit schließen zu können, sind zwingend notwendig. Zum Beispiel Informationen über den Umfang der angeforderten Daten und die genutzte Bandbreite verwendeter Übertragungskanäle lassen Rückschlüsse auf die Wartezeit zu.

**Seiteneffekte**

Durch den spekulativen Charakter von Prefetching werden zusätzliche Ressourcen genutzt, welche ohne Prefetching ungenutzt blieben. Die daraus resultierenden Effekte werden negative Seiteneffekte oder auch nur Seiteneffekte genannt. Die Leistung eines Prefetchingverfahrens darf nicht losgelöst von den Seiteneffekten durch den Einsatz von Prefetching betrachtet werden (vgl. [CB98]). Wie schon angesprochen, ist die Leistung in Bezug auf die dafür zusätzlich benötigten Ressourcen zu beurteilen.

Seiteneffekte lassen sich anhand der Kriterien

- Anstieg des Nachrichtenverkehrs (vgl. [JC98], [BKK04], [JWS02], [JLC00] und [FCD+99])
- Anstieg des Speicherbedarfs des Caches (vgl. [CZ03])
- Anstieg des Rechenzeitaufwands (vgl. [Duc99])

beurteilen. Nicht vergessen werden darf, dass nicht nur Mehraufwand durch Übertragen zusätzlicher Daten sowie Speicherplatz- und Rechenzeitaufwand beim Cache- wie auch Prefetchingverfahren selbst entstehen, sondern auch Mehraufwand bei entfernten Ressourcen, die zusätzliche Anfragen beantworten. Mit dieser Problematik beschäftigen sich *Kroeger* und *Long* und *Mogul* sehen Prefetching in [KL97] auch als eine Möglichkeit, auftretende Lastspitzen entfernter Ressourcen zu senken bzw. zu glätten.

Herauszuheben ist die relativ frühe Veröffentlichung von *Bestavros* mit [Bes96], die nicht die Leistung eines Prefetchingverfahrens, wie oben vorgeschlagen, zum Optimierungsgegenstand macht. Stattdessen wird versucht, für ein vorgegebenes Leistungsniveau, festgelegt als durchschnittliche Wartezeit pro Anfrage, die Seiteneffekte möglichst klein zu gestalten. Eine Vielzahl

[72] Ein ideales Prefetchingverfahren liefert nicht zwangsläufig eine Trefferrate von Eins für beliebige Probleminstanzen.

von Problemstellungen sind denkbar, welche dieser anderen Sichtweise auf Prefetching entsprechen. Betrachtungen dieser Art sind jedoch sehr selten anzutreffen. Bemerkenswert ist unter diesem Gesichtspunkt die Arbeit von *Venkataramani* et al., welche analytische Gleichgewichtsbetrachtungen in Hinblick auf den Wert der zu erwartenden Leistung und die Kosten erzeugter Seiteneffekte eines Verfahrens, angewendet über einen sehr langen Zeitraum auf große Nutzerzahlen, anstellen. Sie ermitteln einen Gleichgewichtszustand, in welchem der Wert zusätzlicher Leistung durch Prefetching die Kosten für zusätzlich benötigte Bandbreite innerhalb eines verteilten Rechnernetzes aufwiegen (vgl. [VYK+02]).

Letztendlich sind Prefetchingverfahren nur dann sinnvoll einzusetzen, wenn der Nutzen, sich aus der vorzeitigen lokalen Verfügbarkeit von Daten ergebend, den Mehraufwand, resultierend aus den spekulativen Anfragen, übersteigt.

Es liegt die Vermutung nahe, dass dieses für Prefetchingverfahren insbesondere dann der Fall ist, wenn

- vom Benutzer wahrgenommene Wartezeiten als besonders Nutzen mindernd bzw. die vom Benutzer aufgebrachte Zeit zur Bedienung der Anwendung als besonders wertvoll anzusehen ist,
- zur Spekulation aufgewendete Übertragungs-, Rechenzeit- sowie Speicherplatzkapazitäten als besonders günstig anzusehen sind, was insbesondere dann der Fall ist, wenn zusätzlich zu erbringende Leistung der Rechnernetzressourcen (verursacht durch spekulative Anfragen) keine Kapazitätserweiterung dieser nach sich zieht, wie dieses zum Beispiel mit der Ausnutzung von zwangsläufig anfallenden Leerlaufzeiten der Fall wäre.

## 2.4 Erfahrungen aus dem Projekt „Methodenlehre Baukasten“

Nach der Darstellung und Annährung der Themengebiete Prefetching sowie Betrieb und Entwicklung verteilter hypermedialer Lernanwendungen werden eigene praktische Erfahrungen des Autors in Bezug auf die Problemstellung der Arbeit beschrieben. Im Rahmen eines staatlich geförderten Softwareentwicklungsprojekts „Methodenlehre Baukasten“ (vgl. [Han04], S.106) ist die rechnergestützte Lernanwendung „Anwendung statistischer Methoden in der klinisch experimentellen Forschung“ entstanden. Mit deren Entwicklung wurde das in der Einleitung der vorliegenden Arbeit beschriebene Untersuchungsziel aufgeworfen. Anhand der Entwicklung dieser Lernanwendung sollen einige ausgewählte Aspekte die Problematik des Einsatzes von Prefetching für verteilte hypermediale Lernanwendungen illustrieren.

### 2.4.1 Zielgruppe und unterstützte Lernszenarien

Spezielle Themengebiete der Statistikgrundlagenausbildung von Medizinstudenten des 6. Semesters an der Universität Rostock sind in der Präsenzlehre im klassischen Vorlesungs- und

Seminarstil vernachlässigt worden.[73] So entstand die Zielstellung, für zwei Themenbereiche, „Randomisierungsverfahren“ und „Validierung diagnostischer Verfahren“, ein rechnergestütztes Lernangebot zum Selbststudium der Studierenden zu entwickeln.[74]

Mit der Spezifikation wurde festgelegt, dass ein rechnergestütztes explorativ angelegtes Lernarrangement auf Basis eines in sich relativ geschlossenen Hypermediums zu schaffen ist.[75] Die Wahl für exploratives Lernen mittels Hypermedium fiel aufgrund didaktischer Überlegungen zu Zielgruppe und Lernstoff (vgl. [TKG02]). Zu unterstützen waren einerseits das Lernszenario Lernen innerhalb der Bildungseinrichtung in den dafür vorgesehenen Rechnerlaboren und andererseits der entfernte Zugriff auf die Lerninhalte von außerhalb der Universität, wie zum Beispiel der Wohnung des Studierenden. Mit der Spezifikation wurde eine Umsetzung mittels webbasierter Technologien festgelegt. Das Abrufen der Inhalte sollte per Web-Browser, das Anbieten der Inhalte über einen üblichen Web-Server erfolgen. Mit der Spezifikation zeichnete sich schon die Problematik des entfernten Zugriffs auf die Inhalte bei schlechter Netzanbindung zum Web-Server über einen Internetserviceprovider (ISP) ab.

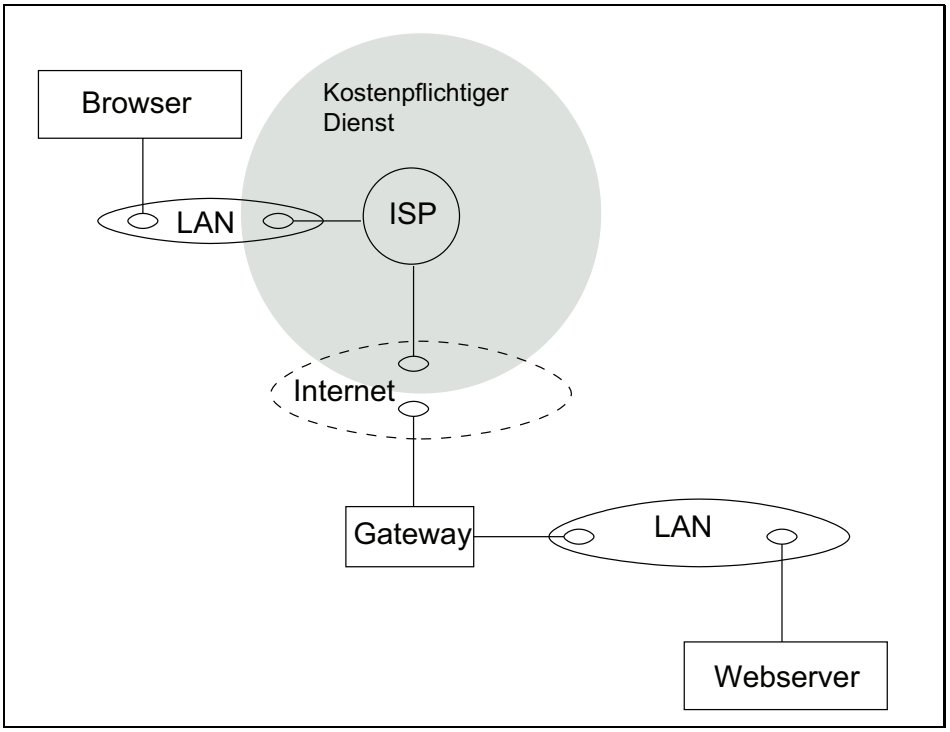



Abbildung 2.2: Stark vereinfachte Darstellung des Zusammenspiels von Web-Browser und Web-Server über einen je nach genutztem Dienst für Lernende kostenpflichtigen Internet Service Provider (ISP)

Der mögliche Zugriff über das universitätsinterne, breitbandige, lokale Rechnernetz, also die

[73] Ursache hierfür war eine so weit gestraffte Lehre in der Grundlagenausbildung, dass spezielle, aber für Mediziner äußerst wichtige Themen, ausgespart blieben – Erklärungen dazu in [KT04] und [TKG03].

[74] Eine genau Beschreibung der Ziele und der aktuelle Stand des Entwicklungsprojekts sind in [KTS05] zu finden.

[75] Geschlossen heißt ohne Einbeziehen extern bereitgestellter Lerninhalte, welche außerhalb der zu schaffenden Lernanwendung verfügbar wären.

Transportmöglichkeit großer Datenmengen in kürzester Zeit, stand in starkem Gegensatz zum entfernten Zugriff über besonders schmale Übertragungskanäle zum lokalen Rechnernetz. Im schlechtesten Fall war mit einer Netzanbindung der Studierenden mit 56KBit-Modem zu kalkulieren. In Abbildung 2.3 ist diese Rechnernetzstruktur und Anbindung der Dienste stark vereinfacht dargestellt. Der Anbieter des Internetzugangs fungiert als Gateway zu entfernten Ressourcen des Internets. Für die von ihm bereitgestellten Dienste fallen Kosten je nach übertragenem Datenvolumen und/oder verfügbarer Bandbreite an.

Erste prototypische Tests ergaben, dass der Umfang zu übertragender, bereits komprimierter und inhaltlich auf das notwendige beschränkter Datenobjekte, welche die Inhalte und interaktiven Funktionalitäten einer Bildschirmseite umfassen, damit auch in ihrer Bedeutung eng zusammenhängende Lerninhalte realisieren, in den Größenordnungen von 30-120 KByte lagen - im arithmetischen Mittel ca. 45 KByte. Daraus ergaben sich bei schlechter Netzanbindung per 56KBit/s Bandbreite je nach Auslastung der Verbindung vom Clienten zum ISP und zu übertragenden zusätzlichen Protokollinformationen erhebliche Wartezeiten beim Zugriff auf die Datenobjekte von mindestens 5 bis über 20 Sekunden. Auch bei besserer Netzanbindung über einen ISP per ISDN-Verbindung oder DSL-Anschluss war noch mit drastischen Einschränkungen zu rechnen.[76] Demgegenüber standen prognostizierte Nutzzeiten der Datenobjekte von 120 bis 360 Sekunden je Datenobjekt, welche im Betrieb der Anwendung auch weitestgehend bestätigt wurden. Im Vergleich dazu sind hypermediale Webseiten im WWW eher klein (10-15 KByte) und die Nutzzeit der Seiten ist durchschnittlich unter 3 Sekunden aufgrund der starken Tendenz der Nutzer zum „Überfliegen“ der Seiten, welches auch Scannen genannt wird.[77]

Die parallele Entwicklung zweier Lösungen zur Präsentation je nach verfügbarer Bandbreite kam aufgrund beschränkter Entwicklungsressourcen nicht in Frage. Auch war davon auszugehen, dass die Präsentationsqualität der Inhalte sich drastisch voneinander unterscheiden würde schlimmstenfalls einige Lerninhalte für Lernende mit schlechter Netzanbindung technisch nicht sinnvoll realisierbar wären. In diesem Zusammenhang wird in der wissenschaftlichen Literatur häufig die erreichbare Qualität der Präsentation von Einzelbildern oder Bewegtbildern diskutiert. Bis zu einem gewissen Maße sind Qualitätseinbußen hinnehmbar (vgl. [Epp99], S.130ff.). In diesem Entwicklungsprojekt stand jedoch die Entwicklung interaktiver Lerninhalte im Vordergrund. Komplexe Mensch-Maschine-Interaktion erfordert entsprechend umfangreichen Programmcode, welcher zwar komprimiert als vorcompilierter Objektcode übertragen werden und clientenseitig dekomprimiert und interpretiert werden kann, qualitative Abstriche mit der Übertragung sind jedoch unmöglich. Sie hätten die Kernfunktionalitäten der Lernanwendung getroffen. Einschränkungen hätten massive Einbußen in der Interaktivität nach sich gezogen oder den Entwicklungsaufwand erheblich gesteigert.[78]

Diese abzusehenden technischen Probleme führten dazu, dass das Lernszenario „Lernen bequem von Zuhause“:

- von Zuhause aus, in vertrauter Umgebung lernen können,
- bei relativ schmalbandiger aber kostengünstiger Netzanbindung

[76] ISDN (Integrated Services Digital Network) mit 64Kbit/s, DSL(Digital Subscriber Line) mit variabel wählbaren Netzverbindungen (Up-/Downstream) 1024-3072 KBit/s.

[77] Vgl. dazu Studien zur Größe der Datenobjekte und Webseiten als auch zum Navigationsverhalten im WWW (vgl. [CH03], [AJ00], [Pit99], [AFJ99] sowie S.16ff. von [Abd98]).

[78] In [TKG04] wird diese Problematik in Bezug auf das Entwicklungsprojekt nochmals ausführlicher diskutiert.

als verteilte Lernanwendung vorerst nicht realisierbar war. Durch die sehr frühzeitige Entwicklung von Prototypen in der Spezifikationsphase war dies rechtzeitig erkennbar.

Es wurde mit allen daraus resultierenden Nachteilen entschieden, eine „Offline-Lösung“ als klassisch nicht verteiltes Lernprogramm verfügbar auf einen Datenträger zu entwickeln und parallel dazu eine „Online-Lösung“ für das Selbststudium im Rechner-Labor. Der Mehraufwand in der Entwicklung war vertretbar gering.[79] Diese Entscheidung wurde während des Entwurfs nochmals angepasst.

Der Verlauf dieses Projekts bis zu diesem Zeitpunkt erscheint typisch für Softwareentwicklungsprojekte verteilter Lernanwendungen. *Astleitner* verweist bereits 2001 auf die folgenden zwei Punkte zur Entwicklung von so genannten Online-Studienangeboten (vgl. [Ast01], S. 7):

„Die Online-Nutzung von Lernmaterialien ist stark von Internet-Netzwerküberlastungen gestört. Sehr viel mehr Anklang findet deshalb die Offline-Nutzung von download-baren Lehrmaterialien oder Lehrmaterialien auf CDROM.“

und bewertet in einem folgenden Abschnitt den Entwicklungsprozess:

„In der Regel wird nicht bei spezifischen Problemen der Betroffenen (Studierenden, Hochschullehrer) angesetzt, sondern bei dem, was technisch machbar ist und nur unter bestimmten optimalen Bedingungen (z.B. hoher Datenübertragungsgeschwindigkeit) auch tatsächlich funktioniert.“

Es ist zu bezweifeln, dass nach fünf Jahren grundsätzlich Änderungen im Denken der Entwickler stattgefunden haben. Das Problem wird zwar immer wieder benannt, doch das Angebot von Methoden zur Auflösung der Problematik, insbesondere in der Entwicklung von Hypermedien, ist eher dürftig (vgl. [BL01] und [Lan04]).

### 2.4.2 Entwurf und technische Umsetzung der Lektionen

Mit dem Entwurf wurden die per Spezifikation inhaltlich ausgearbeitet gelieferten Lerninhalte in Abhängigkeit von den zwei Themengebieten und der zu erwartenden Lernzeit klassisch hierarchisch strukturiert. Es entstanden zwei Lektionen, herunter gebrochen bis hin zu ca. 10-minütigen Lerneinheiten. Für jede Lektion wurde eine Lernzeit von ca. 2 Stunden veranschlagt. Das Zusammenfassen von Lerninhalten zu einer Lerneinheit erfolgte anhand der Lernziele, der veranschlagten Lernzeit zum Erreichen der Ziele und der Abgrenzung zu anderen Lerneinheiten. Darauf aufsetzend wurden einzelne Lerneinheiten in bis zu drei Bildschirmseiten in Form von Drehbüchern[80] zerlegt und alle Bildschirmseiten einer Lektion über navigationsbezogene Links anhand von Empfehlungen des Lehrenden zu sinnvollen Lernwegen miteinander verknüpft.

Das resultierende Navigationsmodell ergab sich letztendlich aus

- der Vorgabe ein Hypermedium zu nutzen sowie den Lernzielen und Lerninhalten,

[79] Die „Offline-Lösung“ enthielt letztendlich optional verwendbare Softwarekomponenten, die auf entfernte Ressourcen zugreifen.

[80] Für multimediale Anwendungen ist in der Regel eine Entwicklung eines Feinkonzepts in enger Anlehnung an ein zu erstellendes Drehbuch und an dem sich daraus ergebenden Bildschirmlayout zu orientieren (vgl. [Thi02], S. 122ff. und [RKFH02], S. 94ff.).

- der gegebenen hierarchischen Unterteilung der Lerninhalte,
- deren Zerlegung in 10-minütige Lerneinheiten,
- der Zerlegung von Lerninhalten in Bildschirmseiten in Form von Drehbüchern,
- den Verknüpfungen zwischen den Bildschirmseiten in Abhängigkeit von sinnvollen Lernwegen sowie semantischen Abhängigkeiten.

Im Entwurfsprozess hat sich gezeigt, dass die drei zuletzt genannten Punkte sehr eng miteinander korrespondieren. Entscheidungen bezüglich des Bildschirmlayouts können bis hin zur vorab festgelegten Unterteilung in Lerneinheiten durchschlagen und eine Änderung notwendig machen.

Aus der technischen Umsetzung der Drehbücher in Medieninstanzen, platziert auf Bildschirmseiten, ergibt sich mit deren Zugriffsmöglichkeiten durch die entworfenen Navigationslinks von einer Bildschirmseite auf eine andere direkt das an die Anwendungskomponenten, hier den Web-Browser, zu übertragende Datenvolumen. Je nach Platzierung der Datenobjekte, lokal bzw. entfernt, kann daraus das resultierend aus entfernten Zugriffen zu übertragende Datenvolumen, je nach gewähltem Navigationsmuster ermittelt werden. Datenobjekte einer Bildschirmseite wurden im Entwurf gezielt zu einem Datenobjekt zusammengefasst.[81]

Vom Lehrenden wurden Lernwege durch die so segmentierten Lerninhalte vorgeschlagen. Diese zeigten, dass mehrfaches Benutzen von Bildschirmseiten und damit klassisches Cachen zur Reduktion der Wartezeiten kaum eine Rolle spielen würde. Aufgrund des explorativen Lernarrangements ist aber trotzdem, insbesondere bei Lernenden mit tiefenstrategischem Navigationsverhalten, eine starke Abweichung von den Lernwegen zu erwarten und so wiederholtes Nutzen von Bildschirmseiten ein Thema. Weiterhin zeigte sich, dass in der Menge empfohlener Lernwege die Zahl der Alternativwege gering ist. Auf einer Bildschirmseite wurde aufgrund didaktischer Überlegungen festgelegt, nie mehr als fünf navigationsbezogene Links zu platzieren, welche sich auf die Entscheidungsfreiheit des Lernenden für einen bestimmten Lernweg beziehen.[82] Dies äußerte sich in der Struktur des sich aus der Verlinkung ergebenden Navigationsbaums mit allen möglichen Lernwegen.

Hätte der Einsatz von Prefetchingverfahren zu diesem Zeitpunkt zur Wahl gestanden, um das Lernzenario „Lernen bequem von Zuhause“ mittels verteilter Anwendung zu stützen, wären solche Überlegungungen gezielt im Rahmen des Performance-Engineering zur Beeinflussung des Entwurfs anzustellen gewesen.

Nach Abschluss der Entwicklung eines Großteils der Bildschirmseiten wurde die mögliche Realisierung des oben angegeführten Lernszenarios „Lernen bequem Zuhause“ sowie im Rechner-Labor in Form einer verteilten Anwendung nochmals geprüft. Der aus dem Blickwinkel eines methodisch fundierten Performance-Engineerings eher zufällig entstandene Entwurf ließ

[81] Dadurch ergaben sich Vorteile bezüglich der „cache-“ und „prefetchability“. Handlungen der Datenobjekte wurden, soweit wie möglich, clientenseitig beim Browser ausgeführt. So konnte die Zahl der Nachrichten zwischen Web-Browser und Web-Server, initiiert aufgrund interaktiver Eingaben des Lernenden, möglichst klein gehalten werden. Letztendlich wurden fast ausschließlich nur noch Zugriffe auf entfernte Ressourcen durch das Verwenden navigationsbezogener Links des Lernenden notwendig.

[82] Dazu gehören nicht Links, welche generell verfügbare Funktionalitäten der Lernanwendung referenzieren wie zum Beispiel Such-, Glossar-, Hilfe-, Druckfunktionalitäten oder der Aufruf von Menüstrukturen.

erahnen, dass Prefetchingverfahren eine technische Lösung darstellen könnten. Aufgrund der den Wartezeiten gegenüberstehenden hohen Nutzzeiten der Bildschirmseiten und dem geringen Verzweigungsgrad des Navigationsbaums wurde der Einsatz von Prefetchingmethoden nun geprüft. Als Prototyp wurde ein sehr einfach zu implementierendes blindes Pretching-Verfahren realisiert. Dieses erstellt aus einem das entworfene Hypermedium beschreibenden Navigationsgraphen einen kürzesten Wegebaum.[83] Dessen Knoten sind die Bildschirmseiten und dessen Kanten sind die navigationsbezogenen Links, welche mit Eins gewichtet sind. Der Wurzelknoten des Baums ist die aktuell genutzte Bildschirmseite des Lernenden. Mittels diesem wird über die während der Nutzzeit der aktuellen Bildschirmseite im Voraus anzufordernden Datenobjekte mit Beginn der Nutzung einer Seite entschieden.

Nacheinander werden all jene Datenobjekte der Bildschirmseiten mit zunehmendem Niveau im Baum angefordert, welche nicht im Cache hinterlegt sind (Abbildung 2.3, S.50). Nacheinander heißt, dass eine Anfrage erst dann gestellt wird, wenn die Antwort des vorher angeforderten Datenobjekts eingetroffen ist. Die Auswahl von Datenobjekten eines Niveaus erfolgt zufällig. Die Dauer der Nutzung der Bildschirmseite sowie die Dauer des Auslieferns der einzelnen Datenobjekte an den lokalen Cache beschränkt die Anzahl im Voraus stellbarer Anfragen.

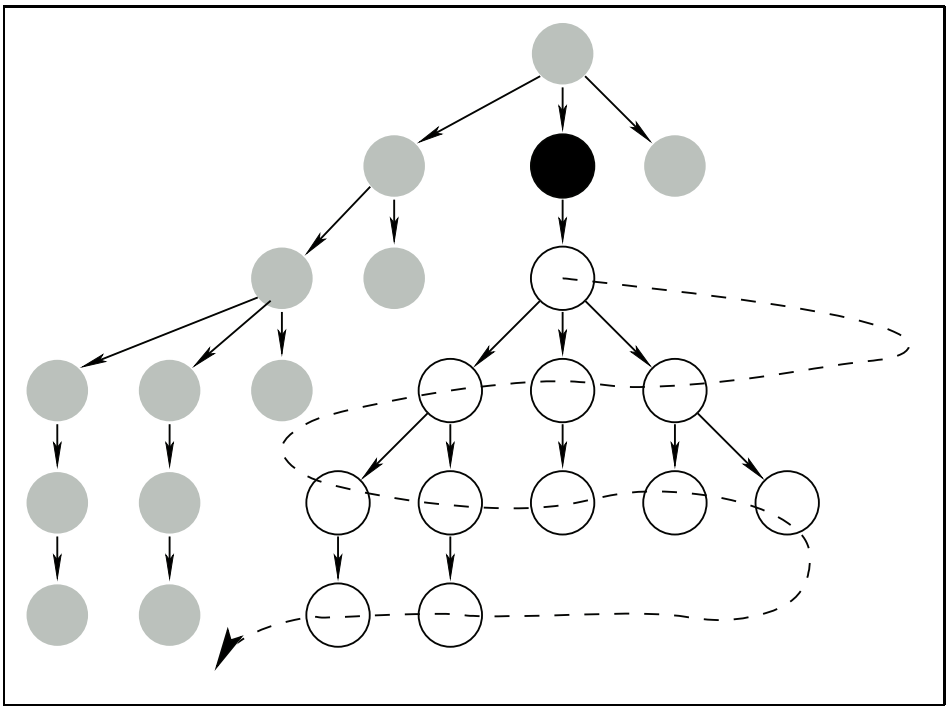


Abbildung 2.3: Sich aus einem Navigationsbaum ergebende Anfragereihenfolge, ausgehend vom aktuell genutzten Datenobjekt bzw. Bildschirmseite, repräsentiert durch den schwarzen Knoten. Graue Knoten sind nicht erreichbar und werden in die Entscheidung nicht einbezogen. Alle weißen Knoten sind, ausgehend vom schwarzen, erreichbar und werden in der durch die gestrichelte Linie symbolisierten Reihenfolge im Voraus geladen, wenn sie nicht lokal beim Cache verfügbar sind.

[83] Einen entsprechenden kürzesten Wegebaum liefert Dijkstras „shortest path“ Algorithmus effizient (vgl. [CLR98], S.527ff.).

Das sehr einfach gehaltene Entscheidungsverfahren ist ein typisches Online-Verfahren[84], welches Entscheidungen der Gegenwart trifft, ohne vollständige Informationen über die Zukunft zu besitzen. Das heißt Informationen, die zukünftige Navigationsentscheidungen des Nutzers entsprechend der Wahl seines Lernweges durch das Hypermedium betreffen.

Wünschenswert wäre eine Prognose, wie sich die Wartezeiten aufgrund entfernter Zugriffe durch Einsatz eines solchen Verfahrens ändern würden, das heißt mit welchen maximalen Wartezeiten auf unterschiedlichen Lernwegen zu rechnen wäre, um eventuelle Ausstiegspunkte des Lernenden aus dem Lernprozess ursächlich längerer Wartezeiten identifizieren zu können. Des Weiteren wäre eine Abschätzung der Lernzeit dringend notwendig gewesen, da sich die zu erwartende Lernzeit eines Lernenden in Abhängigkeit vom gewählten Navigationsweg aus den Nutzzeiten der Lerninhalte und den Wartezeiten auf deren Verfügbarkeit zusammensetzt. Drastische Überschreitungen der spezifizierten zwei Stunden hätten wiederum Entwurfsentscheidungen in Frage gestellt und wären zu diskutieren gewesen.

### 2.4.3 Evaluation und Betrieb

Die Lernanwendung wurde neben klassischen Softwaretests im Rahmen einer systematischen Evaluation zum Lernerfolg bewertet. Dazu wurde der Lernerfolg von über 100 Lernenden beurteilt. Ergebnisse dieser Evaluation sowie eine Beschreibung des Evaluationsprozesses sind in [KT04] beschrieben und sind für die Arbeit nicht weiter von Bedeutung. Im Rahmen dieser Evaluation wurden mittels einer in die Anwendung integrierten Lerner-Tracking-Funktionalität alle Nutzereingaben der Lernenden protokolliert. Anhand dieser ließen sich Navigationsmuster bestimmen. Diese wiederum ließen Rückschlüsse auf die Leistung des eingesetzten Prefetchingverfahrens zu.[85]

Dabei ergab sich folgende qualitative Bewertung des Verfahrens:[86]

- Das konstruierte Verfahren erzeugt in erheblichem Maße negative Seiteneffekte.

  Schlimmstenfalls werden alle von einem Teilbaum repräsentierten Datenobjekte, dessen Wurzel die aktuell genutzte Seite darstellt, im Voraus geladen. Dadurch ist je nach Struktur des Baums, das heißt der durch die Knoten verkörperten Datenobjekte und der Reihenfolge auf den Pfaden des Baums, davon auszugehen, dass eine Vielzahl von Daten im Voraus geladen werden, ohne dass diese dann genutzt werden. Dieser Fall ist insbesondere dann kritisch, wenn die Nutzzeiten in Bezug auf die aus entfernten Zugriffen resultierende

[84] Zu Verfahren dieser Art in den Kapiteln 3 und 5.

[85] Die Evaluation wurde in einem Rechner-Labor der Universität durchgeführt, so dass das lokale Rechnernetz auf die Lerninhalte zugriff. Jedoch konnte anhand der Zugriffe auf ein imaginäres Antwortzeitverhalten bei schlechter Netzanbindung geschlossen werden. Als kritisch zu bewerten ist dieses Vorgehen der Prognose, weil mit den so angenommenen, teilweise sehr hohen Wartezeiten auch Rückkopplungen auf das Lernverhalten zu erwarten sind. Das wurde nicht berücksichtigt, weil nie zur Disposition stand, die Lernenden bewusst mit einer „schlechteren“ Lernanwendung als verfügbar zu exponieren.

[86] Auf die Darstellung der detaillierten statistischen Auswertung auf Basis quantitativer Kennziffern wird hier verzichtet, weil einerseits die Vergleichbarkeit zu anderen Studien dieser Art nicht gewährleistet werden kann (siehe zur Güte Vergleichbarkeit der Leistung von Prefetchingverfahren Abschnitt 2.3.3, S.41) und vorab schon erkennbar war, dass dieses Verfahren nicht von herausragender Güte ist und damit die Ausführungen kaum Aussagekraft besitzen würden.

Wartezeit verhältnismäßig groß sind. Hier wäre ein Mechanismus wünschenswert, welcher prüft, welche Datenobjekte während der Nutzung zwingend geladen werden müssen, um zukünftige Wartezeiten in einem festzulegenden Maße reduzieren zu können.

Des Weiteren ist damit zu rechnen, dass mit dem Beginn der Nutzung der Lernanwendung durch das Verfahren erhebliche Last auf lokalen Anwendungskomponenten, genutzten Übertragungskanälen und Web-Servern erzeugt wird. Die erzeugte Last auf dem Web-Server eines einzelnen Lernenden mit Netzanbindung geringer Bandbreite ist zwar vernachlässigbar klein, aber der Fall, dass viele Lernende mit einem zeitlich nahe liegenden Nutzungsbeginn, beispielsweise kurz nach Ende einer gemeinsamen Lehrveranstaltung, auf Inhalte zugreifen, darf nicht unterschätzt werden.[87]

- Schwankende Wartezeiten

  Durch das Verfahren können zwar die Wartezeiten reduziert werden, jedoch liefert der Mechanismus keine Möglichkeiten bestimmte zu erwartende Wartezeiten gezielt zu reduzieren. Es wäre zum Beispiel wünschenswert, zu erwartende, wenige, sehr lange Wartezeiten gezielt zu reduzieren. Viele kleine Wartezeiten von ein bis zwei Sekunden wären demgegenüber hinnehmbar.

- Gezielte Reduktion der Lerndauer

  Das Verfahren kann nicht dazu verwendet werden, die sich aufgrund von Wartezeiten verlängernde Lerndauer gezielt zu verkürzen, weil sich die einzelnen Lernwege des Navigationsbaums in der sie umfassenden Lernzeit stark voneinander unterscheiden. Mittels Spezifikation wurde eine maximale Lerndauer von 2 Stunden je Lektion veranschlagt. Wünschenswert wäre ein Mechansimus, welcher gezielt Lernwege mit langer Lernzeit berücksichtigt und die Wartezeiten auf diesen Wegen im besonderen reduziert.

Prefetchingverfahren sollten neben dem ersten Punkt gezielt die zweiten beiden Kritikpunkte berücksichtigen. Dieses ergibt sich aus den Diskussionen der vorherigen Abschnitte 2.2.1 und 2.2.2 (S.23 bzw. S.27) zum Navigationsverhalten sowie zur Lerndauer. Weiterhin ist zu beachten, dass das personenabhängige Navigationsverhalten nachhaltig auf Änderungen in der Lernumgebung reagiert. Zum Beispiel kann eine ansteigende Intensität des Beübens von Lerninhalten in korrespondierenden Präsenzveranstaltungen zur Folge haben, dass die Dauer der Nutzung einzelner Lerninhalte sich erheblich verändert. Daraus ergibt sich der Wunsch, Navigationsbäume und Nutzzeiten der Lerninhalte dynamisch an sich verändernde Bedingungen während des Betriebs anpassen zu können. Hierfür könnten algorithmische Verfahren zur Auswertung von Nutzerverhalten Verwendung finden. Eine Erfassung des Verhaltens ist in professionellen Lernanwendungen als Kernfunktionalität anzusehen und stellt damit die Grundlage dar.[88]

[87] In Extremfällen würde das Verfahren kontraproduktiv wirken. Es wäre eine Steigerung der Wartezeiten statt einer Reduktion mit stark ansteigender Last durch den zahlenmäßigen Anstieg der zeitlich eng zusammen liegenden Lesezugriffe beim Web-Server zu erwarten.

[88] Siehe Abschnitt 2.2.1, S.25 zur rechnergestützten Erfassung von Lernverhalten.

### 2.4.4 Fazit

Aus den Erfahrungen zu Entwicklung und Betrieb verteilter hypermedialer Lernanwendungen ergeben sich mehrere offene Fragestellungen:

- Wie sind die Entwicklung und der Betrieb von verteilten Lernanwendungen unter der Forderung von angemessener Zeittransparenz zu gestalten?
- Wann ist der Einsatz von Prefetchingverfahren für verteilte Lernanwendungen sinnvoll?
- Wie können Prefetchingkonzepte aussehen, welche sowohl die Nutzzeit der Lerninhalte als auch Wartezeiten bei entferntem Zugriff, gezielt abgestimmt auf den Lernprozess, berücksichtigen?
- Wie sehen dazugehörige rechnergestützte Entscheidungsverfahren aus?
- Wie lassen sich diese in Entwicklung und Betrieb integrieren?

Auf diese Fragen sind in der umfangreichen, gesichteten Literatur, dargestellt in den vorherigen Abschnitten, keine befriedigenden Antworten zu finden. Nahe liegend ist ein Vergleich zu solch weit verbreiteten webbasierten Anwendungen, deren Kernfunktionalität auch in der Präsentation hypermedial aufbereiteter Inhalte liegt. Es ergeben sich jedoch tendenziell gravierende Unterschiede zwischen verteilten hypermedialen Lernanwendungen und dem Querschnitt webbasierter hypermedialer Anwendungen, für welche aufgrund deren stark anwachsender Verbreitung in den letzten 10 Jahren nahezu alle Prefetchingverfahren entworfen wurden.

Verteilte hypermediale Lernanwendungen unterscheiden sich von webbasierten hypermedialen Anwendungen im allgemeinen dadurch, dass:

- die Aufzeichnung von Nutzerverhalten zur Realisierung von Kernfunktionalitäten benötigt wird,
- die einer Bildschirmseite zuordnenbaren Datenobjekte tendenziell größer sind,
- die Nutzdauer der einer Bildschirmseite zuordnenbaren Datenobjekte tendenziell größer ist,
- den vom Anwender aufgebrachten Nutz- und Wartezeiten eine zentrale Bedeutung mit der Bewertung des durch die Anwendung gestifteten Nutzens beigemessen wird,
- stark variierende Wartezeiten besonders stark das Nutzerverhalten beeinflussen,
- in sich relativ abgeschlossene Hypermedien häufiger Verwendung finden und damit deren Datenobjekte und Verknüpfungen vor der Laufzeit bekannt sind.

Natürlich lässt sich eine Vielzahl von Beispielen konstruieren, die dieser Tendenz scheinbar widerspricht. Auch lassen sich Anwendungsfelder verteilter hypermedialer Anwendungen finden, wie zum Beispiel im Bereich des eCommerce elektronische Verkaufsplattformen, in welchen

mehrere dieser Punkte eine genauso wichtige Rolle spielen. Jedoch beruhen diese Aspekte in ihrer Gesamtheit auf dem Kern einer Lernanwendung: der Rechnerunterstützung von Lernprozessen durch nutzeradäquate Präsentation von Lerninhalten. Aus den umfangreichen Diskussionen dieses Kapitels ist deutlich geworden, dass mit einem Fokus auf die Unterstützung von Lernprozessen durch rechnergestützte, interaktionsreiche Präsentation von Lerninhalten gegenüber dem durchschnittlich im WWW Beobachtbarem, tendenziell mit den oben beschrieben Zusammenhängen zu rechnen ist. Es erscheint sinnvoll, Prefetchingansätze in Hinblick auf verteilte Lernanwendungen und auf Anwendungen mit ähnlichen hier betonten Eigenschaften zu überdenken und weiterzuentwickeln.

Zusätzlich sei noch bemerkt, dass ein effizienter Umgang mit Ressourcen in der Lehre erwartet werden kann insbesondere dann, wenn Lernende mit gutem Willen zusätzlich Ressourcen zur Verfügung stellen bzw. finanzieren, wie dieses zunehmend von ihnen erwartet wird.

## 2.5 Zusammenfassende Würdigung der Grundlagen

Die dargestellte Unterteilung von Prefetchingansätzen nach dem Bewertungsansatz der im Voraus stellbaren Anfragen ermöglicht es, die vielzähligen Verfahren so zu unterteilen, dass nur zwei der fünf Klassen für die Unterstützung von Präsentationsfunktionalitäten verteilter hypermedialer Lernanwendungen von Bedeutung sind. Die Präsentation von Lerninhalten stellt eine Kernfunktionalität von Lernanwendungen bei der unmittelbaren Unterstützung von Lernprozessen dar. Die Strukturierung der Lerninhalte in Hypermedien ermöglicht eine interaktionsreiche Nutzernavigation mit hohen Freiheitsgraden, welche sich dazu eignet, unterschiedliche Lernstile in nahezu gleicher Weise effizient zu unterstützen. Detaillierte Studien belegen, dass Navigationsverhalten der Lernenden nicht willkürlich, rein zufällig stattfindet, sondern eine Vorhersage zukünftigen Verhaltens gut möglich sein müsste. Dieses ist für den erfolgreichen Einsatz eines Prefetchingverfahrens zur Verbesserung des Antwortzeitverhaltens bei Zugriffen auf entfernte Ressourcen von immenser Bedeutung. Studien mit dem originären Untersuchungsziel der Vorhersage von Navigationsverhalten in hypermedialen Lernanwendungen existieren aber nicht.

Mit dem Einsatz von Prefetchingverfahren wird primär die Verkürzung der Wartezeiten beim Zugriff auf entfernte Ressourcen unter zu begrenzenden negativen Seiteneffekten verfolgt. Variierende, häufig zu lange Wartezeiten sind aber nur ein Teilproblem bei der Betrachtung des Antwortzeitverhaltens verteilter Lernanwendungen im Zeitablauf. Hinzu kommt das Abschätzen der Lerndauer, welche eng mit den anfallenden Wartezeiten und den Nutzzeiten segmentierter Lerninhalte verknüpft ist. Je nach angestrebtem Lernszenario und sich damit ergebendem technischen Rahmen konkurrieren zu spezifizierende Nutzzeiten mit zu erwartenden Wartezeiten innerhalb einer maximalen Lerndauer.

Es erscheint offensichtlich, verteilte Lernanwendungen als zeitkritische Anwendungen unter dem Gesichtspunkt der unbedingt zu gewährleistenden Zeittransparenz einzustufen. Der kritische Bereich für zeitliche Verzögerungen wird von der Mensch-Maschine-Interaktion vorgegeben und liegt im Bereich weniger Sekunden. Die Zeitdauer zwischen Anfordern entfernt gespeicherter Daten bis zu deren Präsentation kann je nach Rechnernetz, vorgegeben durch das Lernszenario, um ein Vielfaches das Zumutbare übersteigen. Genaue nutzerspezifische Vorstellungen über Nutzzeiten und Verknüpfung der präsentierten Inhalte ermöglichen es, sehr präzise Entscheidungen darüber zu fällen, wie ungenutzte und zumeist kostenfrei bzw. verhältnismäßig

günstig verfügbare Ressourcen dazu genutzt werden könnten, dieses Missverhältnis aufzulösen. Dieses wäre zu prüfen. Hier erscheint der in der Literatur vorherrschende Schwerpunkt auf vorhersagebasierte Prefetchingverfahrens zwar auch wichtig, aber nicht entscheidend. *Khan* und *Tao* weisen in [KT01] schon darauf hin, dass die Segmentierung eines navigationsabhängigen Datenstroms im Zeitablauf unter Berücksichtigung der Nutzzeiten der Segmente nachhaltigen Einfluss auf die Entscheidung über im Voraus anzufragenden, entfernten Ressourcen haben muss, völlig unabhängig von einer vorausplanenden Bewertung. In der Literatur zu Prefetchingverfahren existieren unzählige, sich teilweise nur marginal unterscheidende Bewertungskonzepte. Die eigentliche Entscheidung auf Basis struktureller und zeitlicher Verknüpfung der entfernt gespeicherten Daten nach einem konkreten Optimierungsziel wird von ihnen jedoch nicht als zentrales Problem angesehen.

Die dargestellte Zerlegung eines Prefetchingverfahrens in Komponenten, welche zwingend benötigte Funktionalitäten zur Realisierung von Prefetching umsetzen, macht es möglich, Beobachtungs- und Bewertungsfunktionalitäten vom eigentlichen Entscheidungsverfahren loszulösen. In dieser Arbeit werden auf Basis einer formalen Problemformulierung Algorithmen diskutiert, die losgelöst von Bewertung und Beobachtung den Entscheidungsprozess unterstützen sollen.

Von besonderem Interesse sind dabei Lernszenarien, in welchen die Zeit als besonders kritisch anzusehen ist und die mittels Prefetching nutzbaren Ressourcen kostenfrei während der Nutzung der Lerninhalte zur Verfügung stehen. Dieses zielt insbesondere auf Lernszenarien ab, welche eine Netzanbindung relativ niedriger Bandbreite des Lernenden an das verteilte Rechnernetz implizieren und Kosten für die Verbindungen zwischen verteilten Anwendungskomponenten abhängig von der Lerndauer unabhängig vom zwischen den verteilten Komponenten übertragendem Datenvolumen anfallen. Darunter fällt insbesondere das im vorherigen Abschnitt erwähnte, klassische Lernszenario „Lernen bequem von Zuhause“. Der Zeitaufwand der Replikation entfernt gespeicherter Daten steht in einem besonderen Missverhältnis zur Kapazität des Nachrichtenübertragungskanals.

Zum Abschluss sei noch bemerkt, dass der Bereich des Datenschutzes und der Datensicherheit für Prefetchingverfahren, insbesondere für Bewertung- und Beobachtungsfunktionalitäten, von herausragender Bedeutung ist. Daraus resultieren Restriktionen, die das Anwendungsfeld von Prefetching erheblich einschränken können und deshalb keinesfalls mit Entwicklung und Betrieb verteilter hypermedialer Lernanwendungen bei eventuellem Einsatz von Prefetchingverfahren vernachlässigt werden dürfen. Da der Bereich aber für die im weiteren diskutierten Ansätze keine unmittelbaren Auswirkungen vermuten lässt, ist dieser in der Arbeit vollständig ausgespart worden.

Thema der folgenden Kapitel dieser Arbeit sind im Rahmen des Konzepts Prefetching spezielle Optimierungsverfahren zur Entscheidungsunterstützung unter Ausnutzung ungenutzter Übertragungskapazitäten verteilter Rechnernetze mit Berücksichtigung des Zeitaspekts und einer besonders schlechten Netzanbindung.

# Kapitel 3

# Modellformulierung und grundlegende Modellanalyse

*„Alles, was die Menschen in Bewegung setzt,*
*muss durch ihren Kopf hindurch;*
*aber welche Gestalt es in diesem Kopf annimmt,*
*hängt sehr von den Umständen ab.“*
*Friedrich Engels*

Aus der umfangreichen Darstellung des Problemfelds im vorigen Kapitel ist eine genaue Problembeschreibung zu extrahieren. Ziel ist es, das Problemfeld soweit einzugrenzen, dass Optimierungsprobleme, deren Lösungen sich entscheidungsunterstützend verwenden lassen, formal in einem Modell beschrieben werden können. Dabei ist ein Gleichgewicht zwischen dem Realitätsbezug des formalen Modells und der voraussichtlichen algorithmischen Lösbarkeit der aus dem Problemfeld zu extrahierenden Problemstellungen zu wahren. Letztendlich erscheint ein Modell bzw. die Formulierung des dazugehörigen Entscheidungs- bzw. Optimierungsproblems völlig wertlos, wenn zwar die praktisch anzutreffende Problemstellung äußerst detailliert dargestellt wird, aber aufgrund der Komplexität die Lösbarkeit in solchem Maße leidet, dass ein in das praktische Anwendungsgebiet sinnvoll integrierbares Lösungsverfahren unmöglich konstruiert werden kann.

In diesem Kapitel wird auf das formale Modell hingeführt. Es wird festgelegt, welche Fragestellungen zu entscheiden sind und welche Informationen zu deren Entscheidung zur Verfügung stehen. Daraus schlussfolgernd wird dann ein Modell formuliert und die Modellannahmen und grundlegende Eigenschaften des Modells ausführlich diskutiert. Es stehen der Realitätsbezug sowie das Anwendungspotential der auf dem Modell basierenden algorithmischen Lösungsverfahren zur Diskussion. Die Lösung der durch das Modell formulierten Optimierungsprobleme ist dann Thema der folgenden beiden Kapitel.

## 3.1 Entscheidungsfelder

In Bezug auf die Verwendung und Gestaltung von Prefetchingverfahren für verteilte hypermediale Lernanwendungen sind hier die folgenden drei voneinander zu trennenden Entscheidungsfelder von Interesse:

**Verwendungsentscheidung**

Mit dem Entwurf der Lernanwendung und der Wahl einzusetzender Technologien ist über den Einsatz eines Prefetchingverfahrens zur Reduktion der Wartezeiten bzw. zur Begrenzung der Lerndauer zu entscheiden.

Es stellen sich zwei Entscheidungsfragen: Sollte ein Prefetchingverfahren zur Verbesserung des Antwortzeitverhaltens eingesetzt werden? Welche Güte im Hinblick auf die spezifizierten Rahmenbedingungen ist von einem Prefetchingverfahren zu erwarten?

**Entwurfsentscheidungen**

Ist die Entscheidung für den Einsatz eines Prefetchingverfahrens gefallen, ist das Antwortzeitverhalten, das heißt die zu erwartenden Wartezeiten sowie Lerndauer bei Einsatz des Verfahrens, zu prüfen. Gegebenenfalls sind Entscheidungen betreffend der Segmentierung und Sequenzierung, aus welchen das Hypermedium resultiert, zu überarbeiten. Es ergeben sich für jeden Navigationslink des Hypermediums, welcher eine im Voraus stellbare Anfrage an eine entfernte Ressource repräsentiert, die Entscheidungsfragen:

Sind die Eigenschaften des Hypermediums so beschaffen, dass die durch die zum Link korrespondierende Anfrage verursachte Wartezeit nicht akzeptabel ist, obwohl ein Prefetchingverfahren eingesetzt wird?

Sind die Eigenschaften des Hypermediums so beschaffen, dass die Lerndauer der durch das Hypermedium repräsentierten Lernwege nicht akzeptabel ist, obwohl ein Prefetchingverfahren eingesetzt wird?

Aus beiden Fragen ergibt sich direkt das Problemfeld, wie der Entwurf des Hypermediums zu verändern wäre, falls denn ein „ja“ auf eine der Fragen zu geben ist.

**Prefetchingentscheidung**

Im weiteren Verlauf dieser Arbeit wird sichtbar, dass die beiden bisher aufgeführten Entscheidungsfelder teilweise auch ohne die Kenntnis eines konkreten Prefetchingverfahrens beantwortet werden können. Auf Grundlage der im Folgenden formulierten Ausgangsinformationen ist jedoch die originäre Entscheidung eines konkreten Prefetchingverfahrens, bezogen auf das Optimierungsziel, von besonderem Interesse. Sind Entscheidungsverfahren und Optimierungsziel festgelegt, dann stellt sich die Frage:

Welche der im Voraus stellbaren Anfragen sind wann im Voraus zu stellen?

Es ist darüber zu entscheiden, wie der unter den gegebenen Rahmenbedingungen auszugestaltende Entscheidungsspielraum eines Prefetchingverfahrens zu nutzen ist.

Alle drei Entscheidungsfelder umfassen Entscheidungen unter Unsicherheit. Wesentlicher Unsicherheitsfaktor ist die Vorhersage des Navigationsverhaltens und damit die Güte des jeweils eingesetzten Prefetchingverfahrens. Mit Hilfe der Vorhersage zukünftiger Anfragen ist es vorhersagebasierten Verfahren möglich, die Güte innerhalb natürlicher Grenzen zu beeinflussen, aufgespannt durch die Szenarien „kein Einsatz eines Prefetchingverfahrens“ und „Einsatz eines idealen Prefetchingverfahrens“. Es ist mit dem Einsatz der verwendeten statistischen Methoden vorhersagebasierter Verfahren jedoch unmöglich, eine bestimmte Güte in enger gesteckten Grenzen als den natürlich vorgegebenen zuzusichern. Deshalb ist es unter dem Gesichtspunkt

möglichst kurzer Lerndauern und Wartezeiten sinnvoll zu prüfen, wie das Hypermedium in Hinblick auf zu erwartende Lerndauern und Wartezeiten unterschiedlicher, auf alternativem Navigationsverhalten basierender Lernwege, zu gestalten ist, so dass herausragend lange Wartezeiten und Lerndauern möglichst kurz sind.

Das dritte Entscheidungssfeld „Prefetchingentscheidung“ unterscheidet sich von den beiden erstgenannten dadurch, dass die Entscheidungen im Zeitablauf des Betriebs der Lernanwendung zu sehen sind. Es handelt sich, im Gegensatz zu den anderen beiden Offline-Problemen, um ein Online-Problem.[1] Aufgrund des dritten Entscheidungsfeldes beeinflusst die zu erwartende Güte des Prefetchingverfahrens die anderen beiden nachhaltig.

## 3.2 Annahmen

Die im Folgenden getroffenen Annahmen sowie deren Unterteilung in unterschiedliche Gliederungspunkte folgt der Zielstellung der Arbeit, den dargestellten Grundlagen sowie den vielzähligen sich unterscheidenden Prefetchingansätzen. Zur Entscheidungsfindung werden folgende auf dem Entwurf des Hypermediums basierende Informationen als bekannt vorausgesetzt:

- Geschlossenes Hypermedium inklusive Navigationslinks

  Die Menge aller von einer lokalen Anwendungskomponente an den lokalen Cache stellbaren Anfragen sowie alle potenziell möglichen Abfolgen, in welcher diese, initiiert von der lokalen Anwendungskomponente, an den ihr zugeordneten lokalen Cache gestellt werden können, ist bekannt.

- Nutzzeiten

  Die Nutzzeit, die nach einer Antwort des lokalen Caches verstreicht, ohne dass innerhalb dieser Zeit eine weitere von der lokalen Anwendungskomponente initiierte Anfrage gestellt wird, ist bekannt.

- Ladezeiten

  Die feststehende zeitliche Verzögerung, genannt Ladezeit, die zwischen dem Stellen einer Anfrage an eine entfernten Ressource und der vollständiger Auslieferung der korrespondierenden Antwort an den lokalen Cache auftritt, ist bekannt.

- Ein Nutzer

  Die Benutzerschnittstelle der lokalen Anwendungskomponente wird von genau einem Nutzer bedient. Damit resultieren die von der lokalen Anwendungskomponente initiierten Anfragen aus dem Navigationsverhalten genau eines Nutzers und nicht einer Nutzergruppe.

[1] Auf die Unterscheidung von On- und Offlineproblemen wird noch detailliert eingegangen.

Während des Betriebs sind die folgenden zusätzlichen Informationen in Abhängigkeit vom jeweils betrachteten Zeitpunkts bekannt:

- Nutzerspezifisches Navigationsverhalten

  Die Abfolge der bis zum betrachteten Zeitpunkt von der lokalen Anwendung initiierten, an den lokalen Cache gestellten Anfragen ist bekannt.

- Zustand des Caches

  Die an den lokalen Cache gelieferten Antworten, die bis zu diesem Zeitpunkt aufgrund der vom Prefetchingverfahren und der lokalen Anwendung initiierten Anfragen abgelegt wurden, sind bekannt.

Im Gegensatz zu anderen Prefetchingverfahren werden folgende Informationen nicht Gegenstand der Entscheidungsfindung sein bzw. nur als Konstanten einfließen:

- Präsentationsverzögerung

  Es wird angenommen, dass der Prozess des Auslieferns und Darstellens lokal verfügbarer Daten keine zeitliche Verzögerungen verursacht.

- Prefetchability

  Es wird davon ausgegangen, dass unabhängig vom Zeitpunkt, zu dem eine Anfrage an eine entfernte Ressource zu stellen ist, immer dieselbe Antwort geliefert wird.

- Ressourcenauswahl

  Es wird angenommen, dass eine Anfrage immer von höchstens zwei Ressourcen des verteilten Rechnernetzes beantwortet werden kann: vom lokalen Cache, falls eine korrespondierende Antwort bei diesem repliziert wurde, und von genau einer zum lokalen Cache entfernten Ressource.

- Variierende Übertragungskapazitäten

  Es wird angenommen, dass dieselbe Anfrage an die entfernte Ressource, unabhängig vom Zeitpunkt des Stellens, immer mit derselben zeitlichen Verzögerung geliefert wird.

- Parallele Anfragen

  Es wird angenommen, dass eine Anfrage nur dann gestellt werden kann, wenn vorher keine Anfrage gestellt wurde oder die Antwort der vorher gestellten Anfrage vollständig beim Cache gespeichert wurde.

- Cachekapazitäten

  Der Cachespeicher wird adäquat zu streng idealen Prefetchingverfahren als beliebig groß angenommen.

- Nutzkosten von Leerlaufkapzitäten

  Es wird angenommen, dass für das Stellen von Anfragen und Liefern der korrespondierenden Antworten innerhalb der Nutzzeit keine Kosten bzw. Aufwendungen anfallen.

Die hier getroffenen Annahmen werden in Abschnitt 3.4, S.71ff. im Zusammenhang mit dem folgenden, formalen Modell diskutiert. Dieses wird unter Maßgabe der zu modellierenden Annahmen und Fragestellungen der Entscheidungsfelder als ein Modell des Hypermediums, Navigationsverhaltens, Datenzugriffs und Nachrichtentransports in Form eines Navigationsbaums und der entscheidungsunterstützend wirkenden Optimierungsziele dargestellt.

## 3.3 Modell

Aus diesen Annahmen wird das folgende Modell entwickelt. Dabei muss zwischen zwei Problemtypen differenziert werden  den Offline- und Onlineproblemen. Da die Unterscheidung eine zentrale Rolle in der Modellformulierung sowie in der weiteren Diskussion und Konstruktion von Lösungsverfahren spielt, wird zuerst kurz auf diese beiden eingegangen.

In Anlehnung an *Atallah* in [Ata99], S.10-18 wird zwischen Off- sowie Online-Problemen und den diese lösenden Verfahren, den On- ($ON$) und Offline-Algorithmen ($OFF$), unterschieden. Die in die Entscheidungsfindung eingehenden Informationen werden mittels einer Inputsequenz *seq* modelliert.

Unter einem Offline-Problem werden solche Probleme verstanden, bei welchen zu einem festgelegten Zeitpunkt einmalig Entscheidungen auf Basis der zu diesem Zeitpunkt verfügbaren Informationen, der Inputsequenz, zu treffen sind. Der dazugehörige Algorithmus $OFF$ heißt Offline-Algorithmus. Das Entscheidungsergebnis wird mit $OFF(\hat{I}(seq))$ beschrieben. Optimale Offline-Algorithmen $OPT$ sind spezielle Offline-Algorithmen, welchen zur Entscheidungsfindung alle Informationen über *seq* zur Verfügung gestellt werden, ihre Entscheidungen auf deren Basis treffen und ein optimales Entscheidungsergebnis $OPT(I(seq))$ liefern. Das unterschiedlich verfügbare Wissen über die Inputsequenz *seq* wird durch die Funktionen $I(seq)$, $\hat{I}(seq)$ ausgedrückt. $\hat{I}(seq)$ liefert im Gegensatz zu $I(seq)$ nur einen Ausschnitt von *seq*.

Dem gegenüber stehen Online-Probleme. Der das Online-Problem lösende Algorithmus $ON$ mit dem Ergebnis $ON(\hat{I}(seq))$ empfängt wie der Offline-Algorithmus eine Inputsequenz *seq*, jedoch erfolgt dies nur schrittweise Input für Input über den Zeitablauf. Mit jedem Schritt sind zum jeweiligen Zeitpunkt Entscheidungen ohne Kenntnis zukünftiger Inputs zu treffen.

**Bemerkung:** *Wird dieselbe aus Optimierungsziel und Inputsequenz bestehende Problemstellung betrachtet, kann ein entsprechender Online-Algorithmus nie ein besseres Optimierungsergebnis erreichen als der dazugehörige optimale Offline-Algorithmus. Wird der zahlenmäßige Wert eines Minimierungs- bzw. Maximierungsproblems als Optimierungsergebnis angenommen, dann gilt:*

$$ON(\hat{I}(seq)) \geq OPT(I(seq)) \quad \text{bzw.} \quad ON(\hat{I}(seq)) \leq OPT(seq) \tag{3.1}$$

**Bemerkung:** *Werden die einzelnen Entscheidungsschritte eines Online-Algorithmus betrachtet, indem die Inputsequenz, bestehend aus n Entscheidungsschritten, in ihre Elemente* $seq = (seq_1, \ldots, seq_n)$ *zerlegt wird, dann kann jeder Entscheidungsschritt wiederum als Offline-Problem aufgefasst werden, wobei jedoch die vorangegangenen Entscheidungen zu berücksichtigen sind. Werden als Optimierungsergebnis die jeweiligen Entscheidungen des verwendeten Offline-Verfahrens angenommen, dann fließen in die in n Schritten zu treffenden Entscheidungen*

*nicht nur die jeweiligen Sequenzelemente sondern auch die vorangegangenen Entscheidungen mit ein. Die Entscheidungsergebnisse der einzelnen Schritte ergeben sich wie folgt:*

$$\begin{array}{ll} 1. & OFF(\ I(seq_1)\ ), \\ 2. & OFF(\ I(\ seq_1, seq_2,\ OFF(I(seq_1))\ )\ ) \\ \vdots & \vdots \\ n. & OFF(\ I(\ seq_1, seq_2, \ldots, seq_n,\ OFF(I(seq_1)), \ldots, OFF(I(seq_{n-1}))\ )\ ) \end{array} \tag{3.2}$$

Das Entscheidungsergebnis des Online-Algorithmus ergibt sich aus der Abfolge aller einzelnen Entscheidungsschritte:

$$OFF(\ I(\ seq_1, seq_2, \ldots, seq_n,\ OFF(I(seq_1)), \ldots, OFF(I(seq_{n-1}))\ )\ ) = ON(\hat{I}(seq)) \tag{3.3}$$

Auch wenn die jeweiligen Offline-Probleme optimal gelöst würden, muss nicht gelten $ON(\hat{I}(seq)) = OPT(I(seq))$. Jeder Entscheidungsschritt, auch der letzte, setzt auf den vorher getroffenen Entscheidungen auf und unterliegt damit den Beschränkungen der Online-Situation.

Für das hier zu lösende Prefetchingproblem bestehen die Elemente der Inputsequenz aus einem Navigationsgraphen bzw. aus diesem resultierenden Navigationsbaum sowie zusätzlich, falls eine Online-Situation abzubilden ist, einem Pfad des Graphen bzw. Baums, welcher einen Navigationsweg eines Nutzers durch das Hypermedium abbildet. In diesem Kapitel wird die Entscheidungsfindung für einzelne Entscheidungsschritte, das heißt Elemente der Inputsequenz, als Offline-Probleme formuliert.

### 3.3.1 Das Offline-Modell

Als Entscheidungsgrundlage dient ein Navigationsbaum $B$, bestehend aus Aufenthaltsknoten (den Knoten) und den navigationsbezogenen Links (den gerichteten Kanten), welche die Inhalte des Hypermediums referenzieren, die durch die Knoten beschrieben werden. Das aus der jeweiligen Zielfunktion resultierende Optimierungsergebnis wird mit $OFF(I(B))$ bzw. entsprechend der noch folgenden Definitionen der Zielfunktionen beschrieben.[2] Im folgenden werden die Eigenschaften des Navigationsbaums beschrieben. Dessen zusammenhängende Kantenfolgen stellen mögliche Navigationswege des Nutzers durch das Hypermedium dar. Ein Weg von der Wurzel zu einem Blatt eines Baumes entspricht einer Abfolge von so genannten Aufenthaltsknoten, die ein Nutzer sequentiell während der Navigation durch das Hypermedium durchläuft.

#### Aufenthaltsknoten

**Definition:** Ein *Aufenthaltsknoten* $a_i \in A, i = 1 \ldots n$ ist ein Tripel $(o_p, u_i, \ell_i)$. Die Parameter stehen für die engl. Ausdrücke (**o**bject, **u**sage time, **l**oading time). Das Objekt $o_p, p = 1 \ldots \leq n$ von $a_i$ wird genau $u_i$ Zeiteinheiten genutzt. Vor Beginn der Nutzung muss es jedoch $\ell_i$ Zeiteinheiten geladen werden, so dass es an die lokale Anwendungskomponente ausgeliefert werden kann. □

[2] *Zielname*$(I(B))$

Ein Objekt muss nicht zwingend nur einem Knoten $a_i$ zugeordnet sein, sondern kann in mehreren Knoten enthalten sein.[3]

Jedem Knoten kann eine Dauer $d_i$ zugeordnet werden, die sich aus der Differenz zwischen Endzeitpunkt der Nutzung von Objekt $o_p \in a_i$ und Ende der Nutzung des vor $o_p$ genutzten Objekts $o_q$ in einer Sequenz von Aufenthaltsknoten ergibt. Es wird angenommen, dass gleichzeitig mit der Beendigung der Nutzung von $o_q$ das nächste Objekt $o_p$, initiiert von der lokalen Anwendungskomponente, angefragt wird. Dieser Zeitpunkt soll Anforderungszeitpunkt des Aufenthaltsknotens heißen. Existiert kein vor $o_p$ genutztes Objekt, dann fällt der Anforderungszeitpunkt von $a_i$ mit dem Beginn des Ladens von $o_p$ zusammen. Weiterhin wird angenommen, dass nur ein Objekt gleichzeitig geladen werden kann. Das erfolgt entweder während der Nutzung eines anderen Objekts oder direkt nach deren Anfrage durch die lokale Anwendungskomponente.

Deshalb gilt:

$$d_i \leq \ell_i + u_i \tag{3.4}$$

Der Zeitraum zwischen den Anforderungszeitpunkten einer Folge von Knoten wird auch jeweils als Aufenthalt bezeichnet. Der Nutzer hält sich $d_i$ Zeiteinheiten in bzw. mit Knoten $a_i$ auf. Die Aufenthaltsdauer bzw. der Aufenthalt in einem Knoten kann dadurch reduziert werden, dass während der Nutzung anderer Objekte vor dem Anforderungszeitpunkts von $a_i$ das Objekt $o_p$ geladen wird. Ist zum Beispiel $o_p$ zum Anforderungszeitpunkt schon vollständig in den lokalen Cache geladen, fällt der Anforderungszeitpunkt von $a_i$ mit dem Beginn der Nutzung von $o_p$ zusammen. Die Aufenthaltsdauer $d_i$ entspricht idealer Weise der Nutzzeit $u_i$.

Objekte können während der Nutzung anderer Objekte in beliebig kleinen, aber ganzzahligen Zeitabschnitten geladen werden.

Jeder Navigationsbaum besteht aus einer Menge Aufenthaltsknoten $A$.

**Redundante Aufenthaltsknoten**

Es wird zwischen zueinander redundanten und nicht redundanten Aufenthaltsknoten unterschieden.

**Definition:** Zwei Aufenthaltsknoten sind genau dann *redundant* zueinander, wenn diese dasselbe Objekt enthalten. Die Ladezeiten solcher Knoten stimmen im Gegensatz zu den Nutzzeiten immer miteinander überein.

Ein Knotenpaar $(a_i, a_j)$ ist redundant zueinander, wenn folgende Funktion $red(a_i, a_j)$ wahr liefert:

$$red(a_i, a_j) = \begin{cases} wahr, & falls\ o_p = o_q \\ falsch, & sonst \end{cases}$$

$$a_i = (o_p, u_i, \ell_i) \tag{3.5}$$

$$a_j = (o_q, u_j, \ell_j)$$

□

[3] Dazu unter der gleich folgenden Definition redundanter Knotenmengen mehr

So lässt sich die Menge aller Aufenthaltsknoten $A$ in knotendisjunkte Knotengruppen $A = G_1 \cup \ldots \cup G_m, m \leq n$ mit jeweils allen zueinander redundanten Knoten zerlegen:

$$\begin{gathered} \forall (a_i, a_j) \in G_k : red(a_i, a_j) = wahr \\ \forall (G_h, G_k) \in A | \forall a_i \in G_h | \forall a_j \in G_k | G_h \neq G_k : red(a_i, a_j) = falsch \end{gathered} \tag{3.6}$$

**Navigationsbaum**

Der Navigationsbaum setzt sich aus $n$ Knoten (der Menge aller Aufenthaltsknoten $A$) und den navigationsbezogenen Links (den gerichteten Kanten zwischen diesen) zusammen.

Ein Navigationsbaum $B$ repräsentiert alle für den Entwurfsprozess bzw. für die Verwendung eines Hypermediums relevanten Aufenthaltssequenzen. Er beschreibt mit seinen Kanten, in welcher Art und Weise ein Nutzer innerhalb des Hypermediums navigieren kann.[4]

**Definition:** Ein *Navigationsbaum* ist entweder leer und enthält keine Aufenthaltsknoten ($B = \emptyset$) oder er besteht aus dem Wurzelknoten $a_r$ und seinen Teilbäumen $T_i, i = 1 \ldots k$, welche wiederum Navigationsbäume sind ($B = \{a_r, T_1, \ldots, T_k\}, k < n$). □

Besteht ein Baum aus dem Wurzelknoten und leeren Teilbäumen, dann wird dieser auch Blatt genannt ($B = \{a_r\} = \Diamond$). Es wird zwischen redundanten und nicht redundanten Navigationsbäumen unterschieden. Die Aufenthaltsknoten nicht redundanter Bäume lassen sich in $n$ einelementige Gruppen nach Bedingung 3.6 aufteilen. Im Gegensatz dazu enthalten redundante Navigationsbäume mindestens eine Gruppe mit zwei Aufenthaltsknoten. Für jeden beliebigen Navigationsbaum gilt, dass auf einem beliebigen Weg von der Wurzel hin zu einem Blatt des Baumes keine zueinander redundanten Knotenpaare existieren.[5]

Jeder der $n$ Teilbäume $T_i = \{a_i, T_1, \ldots, T_k\}$ besteht wiederum aus einer Knotenmenge, welche sich in Knotengruppen je nach Gruppenzugehörigkeit zerlegen lässt. So beschreibt die Menge der Aufenthaltsknoten $G_{ki}$ alle zueinander redundanten Aufenthaltsknoten der Gruppe $k$ im jeweiligen Teilbaum mit Wurzel $a_i$ exklusive des Wurzelknotens $a_i$. Wird ein Teil eines Objekts eines Knotens der Teilgruppe $G_{ki}$ in einer Zeiteinheit während der Nutzung von $a_i$ geladen, dann werden zwangsläufig auch alle anderen Knoten der Gruppe im selben Umfang im lokalen Cache gespeichert.

**Sequenz**

**Definition:** Eine *Sequenz* $\sigma_1^m = \{a_0, a_1, ..., a_{m-1}\}$ beschreibt eine Abfolge von $m$ Aufenthaltsknoten eines gültigen Weges des Navigationsbaums von der Wurzel zu einem Blatt. □

Ein Navigationsbaum symbolisiert genau so viele Sequenzen wie dieser Blätter besitzt. Jeder durch das Hypermedium mögliche Navigationsweg wird durch genau eine Sequenz ausgedrückt.

Innerhalb einer Sequenz des Navigationsbaumes dürfen keine zueinander redundanten Aufenthaltsknoten enthalten sein. Für jede Sequenz gilt:

[4] Navigationsbäume sind zyklenfrei.

[5] Siehe auch 3.7

$$\forall(a_i, a_j) \in \sigma, i \neq j : red(a_i, a_j) = falsch \tag{3.7}$$

**Entscheidungsvariablen**

**Definition:** Eine *Entscheidungsvariable* $x_{ij} \in \mathbb{Z}_0^+$ beschreibt, wie viele Zeiteinheiten zum Laden eines Teils oder des gesamten Objekts des Aufenthaltsknotens $a_j$ während der Nutzzeit $u_i$ des Aufenthaltknotens $a_i$ verwendet werden. □

Die Entscheidungsmatrix $X$ mit den Elementen $x_{ij}$ $(i, j = 1 \ldots n)$ enthält $n \times n$ solche Entscheidungsvariablen. Damit repräsentiert jeweils ein Wert einer Entscheidungsvariablen der Matrix $X$ mit $x_{ij} > 0$ genau eine im Voraus zu stellende Anfrage an eine entfernte Ressource, deren Antwort vollständig in der Zeitdauer $x_{ij}$ Zeiteinheiten im Speicher des lokalen Cache zwischen zu speichern ist.

Es gelten für alle Entscheidungsvariablen $x_{ij} \in X$ die folgenden Bedingungen 3.8 bis 3.12:

$$x_{ii} = 0, \forall i = 1..n \tag{3.8}$$

Während der Nutzzeit des Knotens $a_i$ können keine Zeiteinheiten zum Laden des Objekts desselben Knotens aufgewendet werden.[6]

Weiterhin dürfen während der Nutzung des Objekts $o_p$ aus Aufenthaltsknoten $a_i$ nicht länger andere Objekte geladen werden, als Nutzzeit $u_i$ zur Verfügung steht.

Für die Entscheidungsvariablen $x_{ij}$ muss entsprechend der Knotengruppenzugehörigkeit der Knoten $a_j$ einer Gruppe $G_{ki}$ eines jeden Teilbaums $T_i \in B$ mit jeweiligem Wurzelknoten $a_i$ gelten, wobei eine Entscheidungsvariable $y_{ik}$ eingeführt wird, welche entsprechend der $x_{ij}$ eines Knotens $a_i$ beschreibt, Knoten welcher Gruppen $G_{ki}$ des Teilbaums $T_i$ mit Wurzel $a_i$ in welcher Zeitdauer im Voraus geladen werden:[7]

$$\sum_{\forall G_{ki} \in T_i} y_{ik} \leq u_i$$

$$y_{ik} = \begin{cases} x_{ij}, & falls\ \exists(a_j \in G_{ki}\ und\ a_j \in T_i\ und\ i \neq j) \\ 0, & sonst \end{cases} \tag{3.9}$$

Der Wert einer Entscheidungsvariablen $y_{ik}$ beschreibt die Anzahl der Zeiteinheiten, die darauf verwendet werden, alle Objekte einer Gruppe $k$ zu laden, die im Teilbaum $T_i$ enthalten sind. Für $a_i$ und alle Knoten, die zwar zur Gruppe gehören, jedoch nicht im Teilbaum enthalten sind, ist dieser Wert Null.

[6] Ein Objekt kann erst dann genutzt werden, wenn dieses auch vollständig in den Hauptspeicher geladen ist, so dass die Nutzzeit eines Knotens nicht zum Laden seines Objekts verwendet werden darf.

[7] Die Entscheidungsvariablen $y$ sind notwendig, um auch Entscheidungen für redundante Navigationsbäume darstellen zu können. Sie sind eine Verallgemeinerung der Entscheidungsmatrix $X$. Da jedoch deren Verwendung häufig verwirrend erscheint, werden diese nur dann genutzt, wenn explizit Eigenschaften redundanter Knotenmengen zu modellieren sind. In [Tam05] ist das hier vorgeschlagene Modell weiterentwickelt worden. Es wird nicht mehr explizit zwischen Entscheidungsvariablen für redundante und nicht redundante Navigationsbäume unterschieden. Da die in den folgenden Kapiteln vorgeschlagenen Algorithmen und Nachweise zur Korrektheit und Effizienz auf dieser Unterscheidung basieren, wurde kurzfristig keine Anpassung hin zu einer Vereinfachung des Modells vorgenommen.

Die Ladeentscheidungen $x_{ij}$ für zueinander redundante Aufenthaltsknoten $a_j$, welche einer Teilgruppe $G_{ki}$ angehören, sind von selbem Wert. Zum Laden all dieser Objekte während der Nutzzeit von $a_i$ werden nur einmal $x_{ij}$ Zeiteinheiten benötigt. Existieren keine Knoten der Gruppe $G_k$ im Teilbaum $T_i$, dann ist $y_{ik}$ Null. Es ist sinnvoll, nur solche Objekte zu laden, welche sich auch im jeweiligen Teilbaum befinden, das heißt im weiteren Verlauf einer Sequenz enthalten sein könnten. Für die Knotengruppe $G_{ki}$, welche den Wurzelnknoten des Teilbaums $a_i$ enthält, muss $y_{ik}$ immer Null sein, denn es kann kein weiterer zum Wurzelknoten redundanter Knoten (Bedingung 3.7) im Teilbaum enthalten sein und während der Nutzung von $a_i$ kann das Objekt aus $a_i$ nicht im Voraus geladen werden (Bedingung 3.8).

Für nicht redundante Teilbäume ergibt sich aus Bedingung 3.9 folgender einfacher Zusammenhang, wobei alle $x_{ij}$ Null sind, wenn $a_j \notin T_i$:

$$\sum_{j=1}^{n} x_{ij} \leq u_i \tag{3.10}$$

Genauso wie die verwendbare Nutzzeit eines Aufenthaltsknotens nicht überschritten werden darf, kann nicht länger geladen werden, als die Ladezeit des jeweiligen Knotens beträgt.

$$\forall j = 1 \ldots n : \sum_{i=1}^{n} x_{ij} \leq \ell_j \tag{3.11}$$

Während der Nutzung anderer Objekte vor $o_q \in a_j$ dürfen nicht mehr Zeiteinheiten als $\ell_j$ aufgewendet werden, um Objekt $o_q$ mit einer Ladezeit $\ell_j$ in den Hauptspeicher zu laden.

Wie oben bereits erwähnt, soll die Nutzzeit von $a_i$ zum Laden in den Hauptspeicher nur für Objekte solcher Knoten $a_j$ verwendet werden, welche sich in einem der Teilbäume des Baumes finden, dessen Wurzel $a_i$ selbst ist. Es gilt:

$$x_{ij} : \begin{cases} x_{ij} > 0, & falls \;\; \exists (T_i = \{a_i, T_1..T_k\} \in B \; und \; a_j \in \{T_1..T_k\}) \\ x_{ij} = 0, & sonst \end{cases} \tag{3.12}$$

**Aufenthaltsdauer und Restladezeit**

In Abhängigkeit von den Entscheidungen $x_{ij} \in X$ aller Objekte entstehen so genannte Restladezeiten $r_j$, auch Warte- oder Latenzzeit eines Aufenthaltsknotens $a_j$ genannt.

$$0 \geq r_j \geq \ell_j \tag{3.13}$$

Für jede Restladezeit eines Knotens $a_j$ gilt:

$$r_j = \ell_j - \sum_{i=1}^{n} x_{ij}, \forall j = 1..n \tag{3.14}$$

**Definition:** Die *Restladenzeit* $r_j$ eines Aufenthaltsknotens $a_j$ gibt an, wie viele Zeiteinheiten zum Laden des Objekts des Knotens nicht während der Nutzzeit anderer Aufenthaltsknoten verwendet werden. □

Gilt $r_j = 0, \forall j = 1 \ldots n$ , dann tritt in keinem Aufenthaltsknoten des Baums Wartezeit durch Laden von Objekten in den Hauptspeicher auf. Ein latenzfreies Abarbeiten aller durch den Navigationsbaum beschriebener Sequenzen wäre möglich.

**Definition:** Die *Aufenthaltsdauer* $d_i$ eines Knotens $a_i$ ergibt sich daraus wie folgt:

$$d_i = r_i + u_i \tag{3.15}$$

□

Die folgenden Ziele beziehen sich auf die Optimierung der Restladezeiten der Aufenthaltsknoten des Baums.

**Baumbasierte Optimierungsziele**

Es sollen zwei Optimierungsziele betrachtet werden, die sich auf alle Aufenthaltsknoten des Navigationsbaums beziehen:

2*MLB* **M**inimierung der **m**aximalen **L**atenzzeit aller Knoten des **B**aums

Gesucht ist eine bestmögliche Belegung der Entscheidungsvariablen $x_{ij}$ für einen nicht redundanten Navigationsbaum unter folgender Zielstellung:

$$min \Rightarrow max_{i=1..n}(r_i) \tag{3.16}$$

Das heißt, die maximal auftretende Restladezeit aller Knoten des Baums ist zu minimieren.

*MSLB* (**M**inimierung der **S**umme der **L**atenzzeiten aller Knoten des **B**aums)

Gesucht ist eine bestmögliche Belegung der Entscheidungsvariablen $x_{ij}$ für einen nicht redundanten Baum unter folgender Zielstellung:

$$min \Rightarrow \sum_{i=1}^{n} r_i \tag{3.17}$$

Das heißt, die Summe der Restladezeiten aller Aufenthaltsknoten ist zu minimieren.

**Sequenzbasierte Optimierungsziele**

Es werden drei Optimierungsziele bezüglich der durch einen Baum repräsentierten Sequenzen untersucht.

2*MLS* **M**inimierung der **m**aximalen **L**atenzzeit aller Knoten einer beliebigen **S**equenz

Gesucht ist eine bestmögliche Belegung der Entscheidungsvariablen $x_{ij}$ für einen nicht redundanten Navigationsbaum unter folgender Zielstellung:

$$min \Rightarrow max_{\forall \sigma \in B, \forall a_i \in \sigma}(r_i) \tag{3.18}$$

Das heißt, die maximal auftretende Restladezeit der Aufenthaltsknoten einer beliebigen Sequenz ist zu minimieren.[8]

*2MSLS* **M**inimierung der **m**aximal auftretenden **S**umme aller **L**atenzzeiten der Aufenthaltsknoten einer beliebigen **S**equenz

Gesucht ist eine bestmögliche Belegung der Entscheidungsvariablen $x_{ij}$ für einen nicht redundanten Navigationsbaum unter folgender Zielstellung:

$$min \Rightarrow max_{\forall \sigma \in B} \left( \sum_{\forall a_i \in \sigma} r_i \right) \tag{3.19}$$

Das heißt, die maximale Summe aller Restladezeiten einer beliebigen Sequenz des Navigationsbaumes ist zu minimieren.

*MSLS* **M**inimierung der **S**umme aller **L**atenzzeiten aller Aufenthaltsknoten über alle möglichen **S**equenzen

Gesucht ist eine bestmögliche Belegung der Entscheidungsvariablen $x_{ij}$ für einen nicht redundanten Navigationsbaum unter folgender Zielstellung:[9]

$$min \Rightarrow \sum_{\forall \sigma \in B} \sum_{\forall a_i \in \sigma} r_i \tag{3.20}$$

**Optimierungsziele für redundante Navigationsbäume**

$2MLB^*$, $MSLB^*$, $2MLS^*$, $2MSLS^*$, $MSLS^*$

Die $*$-Zielstellungen entsprechen den jeweilig vorher formulierten Zielen. Jedoch ist jeweils eine optimale Belegung der Entscheidungsmatrix $X$ statt eines nicht redundanten für einen redundanten Navigationsbaum gesucht.

Wie später sichtbar wird, ändert sich die Schwierigkeit der algorithmischen Lösung der mit $^*$ gekennzeichneten Problemstellungen drastisch gegenüber dem jeweiligen Ausgangsproblem. Deshalb werden die Optimierungsprobleme nicht redundanter Bäume losgelöst von denen redundanter Bäume betrachtet.

Zur Illustration der Ziele werden im Anhang in Abschnitt A.1, S.197ff. für Beispielinstanzen eines nicht redundanten Navigationsbaums optimale Ausprägungen der Entscheidungsmatrix in Abhängigkeit von der Zielstellung dargestellt.

[8] Dem Leser fällt vielleicht jetzt schon auf, dass *2MLS* und *2MLB* sowie die noch im Folgenden beschriebenen dazugehörigen $*$-Zielstellungen für redundante Bäume trotz des unterschiedlichen Bezugs auf Sequenz bzw. vollständigen Baum identisch sind. Darauf wird noch genauer in Abschnitt „Offline-Verfahren für nicht redundante Bäume“ 4.1, S.101ff. eingegangen.

[9] Auch hier fällt dem Leser vielleicht auf, dass die Zielstellungen *MSLS* und *MSLB* sowie die dazugehörigen im Folgenden dargestellten $*$-Zielstellungen einander stark ähneln. Im Abschnitt „Offline-Verfahren für nicht redundante Bäume“ 4.1, S.101ff. wird darauf genauer eingegangen. Es zeigt sich, dass *MSLB* bzw. $MSLB^*$ eine Spezialisierung von *MSLS* bzw. $MSLS^*$ ist. Optimale Lösungen von *MSLS* bzw. $MSLS^*$ sind auch optimal für *MSLB* bzw. $MSLB^*$. Jedoch kann *MSLB* mit wesentlich weniger Aufwand als *MSLS* gelöst werden und die approximative Lösung für $MSLS^*$ ist von erheblich schlechterer Güte als $MSLB^*$, so dass eine Unterscheidung sinnvoll erscheint.

### 3.3.2 Das Online-Modell

Das in Abschnitt 3.3.1 dargestellte Offline-Modell wird zur Modellierung des Online-Entscheidungsproblems erweitert. Zu den schon bekannten Modellelementen des Offline-Modells Navigationsbaum, Aufenthaltsknoten, Sequenz und Entscheidungsvariablen wird zusätzlich die Session $s$ eingeführt. Das Optimierungsergebnis des Online-Verfahrens entsprechend der jeweiligen Zielfunktion wird durch $ON(\hat{I}(B,s))$ symbolisiert bzw. entsprechend der noch folgenden Definitionen der Zielfunktionen angegeben. Das Online-Verfahren erhält nur Informationen der Gegenwart und Vergangenheit über die Session und dem daraus resultierenden Navigationsbaum.

#### Session

**Definition:** Eine *Session* repräsentiert eine Folge von $m$ Knoten des Navigationsbaums. Die Knoten enthalten vom Nutzer über einen Zeitraum in einer bestimmten Reihenfolge angeforderte Menge von Objekten. Die den Objekten durch den Navigationsbaum zugeordneten Aufenthaltsknoten werden Session $s_1^m = \{a_1, a_2, \ldots, a_m\}$, kurz $s$, genannt. Eine Session ist identisch zu genau einer der mittels Navigationsbaum $B$ definierten möglichen Sequenzen $\sigma \in B$. Es gilt für jede Session:

$$\exists \sigma_1^m \in B : s_1^m = \sigma_1^m \tag{3.21}$$

□

Ein Bereichsausschnitt der Session wird durch $s_i^j = \{a_{i-1}, \ldots, a_{j-1}\}$ dargestellt. Für einen einzelnen Aufenthaltsknoten an der $i$-ten Position der Session steht $s_i^i$.

#### Sessioneigenschaften

Die Dauer einer Session $D(s)$ ergibt sich aus einem Zeitraum, welcher begrenzt ist durch den Zeitpunkt des Beginns der Session, das heißt der Zeitpunkt des Beginns des Ladens des Objekts im ersten Aufenthaltsknoten, und durch dem Zeitpunkt des Endes der Session, das heißt dem Zeitpunkt der Beendigung der Nutzung des Objekts im letzten Aufenthaltsknoten der Session. Damit ergibt sich die Dauer aus der Summe der einzelnen Nutz- und Wartezeiten aller Knoten der Session, das heißt den Aufenthaltsdauern eines jeden Knotens. Die Wartezeit $W(a_i,t)$ eines Knotens $a_i$ ab einem bestimmten Zeitpunkt $t$ entspricht genau der Restladezeit $r_i$ des Knotens zu diesem Zeitpunkt. Erst mit Wartezeit bzw. Restladezeit von Null kann das Objekt des Knotens vom Nutzer auch benutzt werden.

Während einer Session, das heißt online, sind zu bestimmten Zeitpunkten Entscheidungen darüber zu treffen, welche Objekte welcher Aufenthaltsknoten in welchen Zeiträumen zu laden sind, so dass deren Restladezeiten entsprechend der Zielstellung minimal sind.

Die Funktion $D(s_i^j)$ liefert als Zeitdauer die Differenz zwischen Anforderungszeitpunkt des Objekts im $j$-ten Aufenthaltsknoten und dem Anforderungszeitpunkt des Objekts im $i$-ten Knoten der Session $s$. Der $i$-te Knoten wird genau zum Zeitpunkt $t(s_i^i)$ angefordert. Der Endzeitpunkt einer Session wird mit $t(s_{m+1}^{m+1})$ definiert und ist genau der Zeitpunkt, in welchem die Nutzung des letzten Objekts $s_m^m$ der Sequenz beendet wird.

Es gilt für jeden Ausschnitt des $i$ bis $j$-ten Knotens einer jeden Session:

$$
\begin{aligned}
D(s_i^j) &= t(s_{j+1}^{j+1}) - t(s_i^i) \\
&= W(s_i^i, t(s_i^i)) + u_i + \ldots + W(s_j^j, t(s_j^j)) + u_j \\
&= d_i + \ldots + d_j \\
D(s) &= t(s_{m+1}^{m+1}) - t(s_1^1)
\end{aligned}
\tag{3.22}
$$

**Baumsession**

**Definition:** Unter einer *Baumsession* wird eine spezielle Session verstanden, welche eine Sequenz des Navigationsbaums ist, in welcher aus jeder Knotengruppe des Baums ein Knoten enthalten ist.

**Bemerkung:** *Existiert in einem nicht redundanten Navigationsbaum eine solche Sequenz, dann besteht dieser Baum ausschließlich aus dieser einen Sequenz von Knoten.*

Er beschreibt genau eine Abfolge von zueinander nicht redundanten Aufenthaltsknoten.

**Entscheidungsvariablen**

Die Entscheidungsmatrix $X$ des Offline-Modells wird um die Dimension der Entscheidungszeitpunkte erweitert. Zu jeder Session $s_1^m$ mit $m$ Entscheidungszeitpunkten $\{t(s_1^1), \ldots, t(s_m^m)\}$, den Anforderungszeitpunkten der Aufenthaltsknoten, existiert genau eine Entscheidungsmatrix $X$, wie diese mit dem Offline-Modell beschrieben wurde.

**Definition:** Für jeden Entscheidungszeitpunkt $t(s_i^i), 1 \leq i \leq m$ ist über die Werte einer Entscheidungsmatrix $X_i$ mit den *Entscheidungsvariablen* entsprechend der Matrix $X$ des Offline-Modells zu entscheiden, wobei der Index $i$ einer fortlaufenden Nummerierung der Zeitpunkte $t(s_i^i)$ entspricht.

**Sessionbasierte Optimierungsziele**

Folgende zwei Zielstellungen sollen untersucht werden. Es gilt für jedes Ziel, dass erst zum Zeitpunkt der Anfrage eines Objekts durch die lokale Anwendungskomponente bekannt ist, welches Objekt welches Aufenthaltknotens angefragt wird. Gegenüber dem Offline-Modell ist hier unter eingeschränkter Information zu entscheiden.

MDS **M**inimierung der **D**auer einer **S**ession

Gesucht ist die bestmögliche Belegung der Entscheidungsvariablen $X_i,\ i = t(s_1^1), \ldots, t(s_m^m)$ einer beliebigen Session $s = \sigma \in B$ eines nicht redundanten Navigationsbaums unter folgender Zielstellung.

$$
min \Rightarrow D(s) \quad bzw. \quad min \Rightarrow \sum_{\forall a_i \in s} d_i \tag{3.23}
$$

2MDS **M**inimierung der **m**aximalen **D**auer eines beliebigen Knotens aller Knoten einer **S**ession

Gesucht ist die bestmögliche Belegung der Entscheidungsvariablen $X_i, i = t(s_1^1), \ldots, t(s_m^m)$ einer beliebigen Session $s = \sigma \in B$ eines nicht redundanten Navigationsbaums unter folgender Zielstellung.

$$min \Rightarrow max_{\forall s_i^i \in s} d_i \tag{3.24}$$

Das heißt, es ist die Dauer einer Session zu minimieren, welche die Objekte aller Aufenthaltsknoten des Navigationsbaums enthält.

**Optimierungsziele für redundante Navigationsbäume**

$MDS^*$ und $2MDS^*$ entsprechen den obigen Zielstellungen, jedoch sind diese auf redundante Navigationsbäume bezogen.

## 3.4 Diskussion der Modellannahmen

Mit dem vorgeschlagenen Modell werden die bereits getroffenen Annahmen in Form von Optimierungsproblemen modelliert und teilweise weiter verfeinert. In diesem Abschnitt wird geklärt, in welchem Maße die formal dargestellten Optimierungsprobleme dem im vorherigen Kapitel geschilderten Anwendungsbezug entsprechen.

Das beschriebene Modell umfasst Annahmen bezüglich Hypermedium, Navigation, Zugriff auf sowie Speicherung von Daten.

### 3.4.1 Hypermedium

Die mit dem Modell beschriebenen Objekte $o$ der Aufenthaltsknoten und die Verknüpfung der Knoten beschreiben ein Hypermedium in Form einer Baumstruktur. Die Objekte fassen eine Menge von Daten bzw. wiederum Objekte oder Objektbeschreibungen, zum Beispiel Lernobjekte, zu einer Einheit zusammen. Dabei handelt es sich je Objekt eines Aufenthaltsknotens um eine Menge von Daten, welche folgend auf eine von der lokalen Anwendungskomponente initiierten Anfrage zusammenhängend zu präsentieren ist. Solch eine Präsentationseinheit könnte zum Beispiel alle Daten zur Darstellung einer Bildschirmseite enthalten. Das Zusammenfassen der Daten in solchen Präsentationseinheiten erfolgt mit dem Entwurf, genauer mit der Entwicklung des Layouts der einzelnen Seiten, die dem Nutzer über die lokale Anwendungskomponente präsentiert werden. Die Verknüpfung der Aufenthaltsknoten entspricht den Navigationsmöglichkeiten zwischen den so zusammengefassten, im Voraus anforderbaren Daten. Damit modellieren die Kanten des Baums nur genau jene navigationsbezogenen Links des Hypermediums, welche die Navigation zwischen solchen Präsentationseinheiten symbolisieren. Navigationselemente innerhalb einer Präsentationseinheit werden nicht modelliert.

Navigationsmöglichkeiten in Hypermedien werden im Regelfall, falls keine streng hierarchische Struktur vorausgesetzt wird, in gerichteten Graphen oder auch Multi-Trees abgebildet (vgl. [FW03]). Die hier vorgenommene Strukturierung der Informationen in Form eines Baums schränkt jedoch nicht die Ausdruckskraft des Modells ein. Graphen wie auch Multi-Trees lassen sich mit einem redundanten Navigationsbaum beschreiben. Jeder Weg des Baums von der Wurzel zu einem Blatt beschreibt genau einen Navigationsweg des Hypermediums. Voraussetzung dafür ist, dass jeder dieser Wege endlich ist. Eine Begrenzung der zu betrachtenden Wege wäre zum Beispiel über die kumulierten Nutzzeiten der Aufenthaltsknoten eines Weges möglich. Es erscheint sinnvoll, nur Wege zu betrachten, die zwingend nicht länger sind als die

spezifizierte Lerndauer bzw. die kumulierten Nutzzeiten der Objekte eines solchen Weges. In Graphen treten im Gegensatz zu Bäumen Zyklen auf. Solche zyklusbildenden Kantenzüge des Graphen lassen sich auch im Baum durch die Begrenzung der Weglänge abbilden. Zyklusbildende Kantenzüge sind entsprechend der Anzahl der jeweils modellierten Zyklen mehrfach in den Navigationsbaum aufzunehmen.[10] In Abbildung 3.1 ist ein solches Beispiel dargestellt. Knoten mit derselben Bezeichnung sind im Sinne des Modells zueinander redundant. Knoten A' und B' symbolisieren eine mögliche wiederholte Nutzung der Objekte aus Knoten A und B.

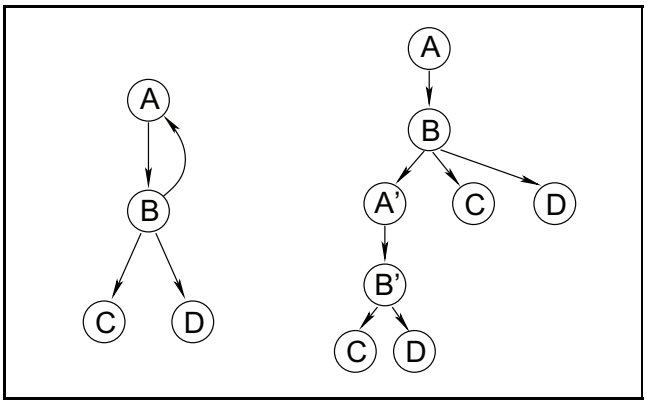



Abbildung 3.1: Ein höchstens zweimal zu durchlaufender Zyklus des Navigationsgraphen (links) wird im Navigationsbaum (rechts) modelliert.

Weiterhin erscheint die Modellierung des Hypermediums mit einem Baum künstlich, weil alle Wege des Baums den selben Anfangsknoten gemeinsam haben, Hypermedien jedoch mehrere so genannte Einstiegspunkte für den Nutzer bereitstellen können. Dieser Fall kann modelliert werden, indem ein zusätzlicher Knoten mit Wert Null für Nutz- und Ladezeit als Wurzel angenommen wird und weiterhin Kanten von diesem Knoten zu allen Anfangsknoten der möglichen Navigationswege durch das Mediums hinzugefügt werden. Abbildung 3.2 zeigt einen solchen Fall mit drei Einstiegspunkten, modelliert durch die Aufenhaltsknoten A,B und C. Eingeführt wird ein imaginärer Einstiegsknoten, mit $*$ markiert, mit einer Nutz- und Ladezeit von Null sowie drei weiteren Kanten jeweils von $*$ nach A, B und C.

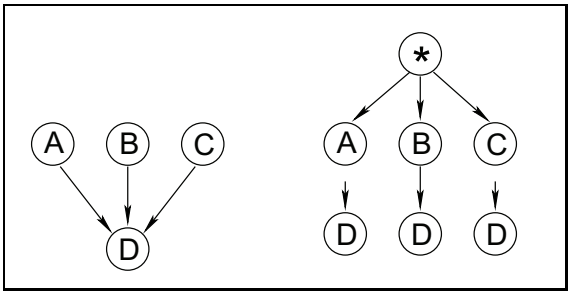



Abbildung 3.2: Drei Einstiegpunkte in das Hypermedium werden von einem zusätzlichen Knoten referenziert.

[10] Im Abschnitt „Speicher und Zugriffsmodell“ wird sichtbar, dass dieses nicht dazu führt, dass ein Objekt auch mehrfach auf einem solchen zyklusdarstellenden Weg enthalten ist.

Eine „echte“ Einschränkung ist die Begrenzung des Problems auf eine fixe Menge von Objekten. Mit dem Zeitpunkt der Anfrage des ersten Objekts einer Sequenz bzw. Session werden alle folgenden Objekte als im Voraus ladbar bekannt angenommen. Viele Prefetchingverfahren nehmen diese Menge der im Voraus ladbaren Daten als im Zeitablauf variabel an. Die hier betrachtete Problemsituation bezieht sich jedoch auf einen eng begrenzten, durch die Anwendung selbst vorgegebenen Datenraum. Lernanwendungen bedienen sich überwiegend eines in sich relativ geschlossenen Hypermediums. Das „Verlassen“ des Hypermediums wird zwar häufig durch Links auf anwendungsfremde Datenräume unterstützt und ist in explorativen Lernumgebungen auch erwünscht, die Navigation in anwendungsfremden Datenräumen unterliegt aber nicht mehr Entwicklungsentscheidungen bezüglich der Anwendung selbst.

### 3.4.2 Navigationsmodell

Ausschließlich mit Hilfe der an die Aufenthaltsknoten gebunden festgelegten Nutzzeiten und der Verknüpfungen von Aufenthaltsknoten wird das mögliche Navigationsverhalten eines Nutzers stark vereinfacht abgebildet. Ein konkreter Navigationsweg eines Nutzers wird über einen im Baum enthaltenen Weg von der Wurzel bis zu einem Blatt, inklusive der Nutzzeiten aller Aufenthaltsknoten auf diesem Weg, repräsentiert.

Trotzdem genaue Vorstellungen über das Navigationsverhalten innerhalb der Lernanwendung existieren sollten, erscheint es nahezu unmöglich, präzise Aussagen über die Nutzzeiten der Lerninhalte im Voraus treffen zu können. Die Beschreibung mit Hilfe von statistischen Verteilungen erscheint glaubwürdiger. Die präzise Angabe von fixen Nutzzeiten erscheint im Rahmen der folgenden Betrachtungen jedoch sinnvoll, weil für die algorithmengestützte Abschätzung von Lerndauer und Wartezeiten das Eintreten besonders schlechter Fälle wichtig ist. Das Hypermedium ist so zu entwerfen, dass auch noch bei denkbar ungünstigen Gegebenheiten[11] die Spezifikation bezüglich des Antwortzeitverhaltens zu erreichen ist. Letztendlich vereinfachen die vorab festgelegten Nutzzeiten drastisch das Entscheidungsproblem.

Auch ist zu kritisieren, dass jeder mögliche Navigationsweg in gleichem Maße, das heißt ohne Modellierung von Präferenzen des Nutzers oder Entwicklers, in das Modell einfließt. Es ist davon auszugehen, dass bestimmte Wege durch eine Nutzergruppe besonders häufig und andere sehr selten beschritten werden. Das Modell lässt nur zu, bestimmte, scheinbar unbedeutende Inhalte bzw. das Warten auf Verfügbarkeit derer, gezielt nicht in das Modell aufzunehmen und so nicht Gegenstand der Entscheidung werden zu lassen. Auch hier sei darauf verwiesen, dass angenommen wird, dass die Spezifikation bezüglich aller durch den Baum dargestellten Wege einzuhalten ist, damit die Häufigkeit deren Verwendung als unerheblich angesehen wird.

Beide Auffassungen sind strittig. Sie werden im Kapitel 6 nochmals aufgegriffen. Hier werden diese beiden kritischen Punkte ausgeglichen, indem eine Methode zur Verknüpfung der auf den hier dargestellten Optimierungsproblemen basierenden Verfahren mit herkömmlichen schwellen- und/bzw. vorhersagebasierten Prefetchingverfahren, die diese Punkte berücksichtigen, noch vorgeschlagen wird. Dadurch können diese beiden wesentlichen Kritikpunkte abgemildert werden.

[11] Baumstruktur, Lade-, Nutzzeiten sowie vom Nutzer gewählte Sequenz

### 3.4.3 Datenspeicherung, -zugriff und Nachrichtentransport

Mit dem vorgelegten Modell wird zwischen zwei Varianten der Anfrage je nach Initiator unterschieden. Das sind jeweils von der lokalen Anwendungskomponente und vom Prefetchingverfahren initiierte Anfragen. Anfragen der lokalen Anwendungskomponente werden im Modell nicht explizit modelliert. Sie werden aus dem Speicher des lokalen Caches und/bzw. von einer entfernten Ressource bedient und ergeben sich aus dem Zustand des Caches zum jeweiligen Anfragezeitpunkt. Nur die Anfragen an entfernte Ressourcen, initiiert durch das Prefetchingverfahren, basieren auf den Werten der Entscheidungsmatrix $X$ bzw. Matrizen $X_i$, wobei ein Wert dieser $x_{ij}$ genau eine Anfrage an eine entfernte Ressource während der Nutzzeiten des Objekts im Aufenthaltsknoten $i$ symbolisiert. Die entfernte Ressource liefert einen Teil des Objekts im Knoten $a_j$ aus. Angefragt wird eine Nachricht bestimmter Länge. Die Länge ergibt sich aus der Zeit, die zur Verfügung steht, um diese vollständig im Speicher des lokalen Caches abzulegen. Entschieden wird darüber, welche Zeitdauer zur Übertragung der Nachricht bei festgelegter Bandbreite zwischen entfernter Ressource und lokalem Cache zu nutzen ist, um einen Teil des Objekts im Cache zu replizieren. Aus dieser Entscheidung ergibt sich die Länge der Nachricht.

Nur während der Nutzzeit eines Objekts können mehrere unterschiedliche Objekte bzw. Teile davon bei unterschiedlichen, entfernten Ressourcen angefragt werden. Die Reihenfolge der Anfragen bzw. ob die Anfragen auch parallel zu stellen wären, ist unerheblich. Es wird davon ausgegangen, dass jede Antwort auf eine solche Anfrage sich immer mit derselben Verzögerung in Abhängigkeit von der jeweiligen Länge der Nachricht gestaltet.

Die Einschränkung, dass Objekte nur dann außerhalb des Zeitraums der Nutzung eines anderen Objekts geladen werden dürfen, wenn sie unmittelbar von der lokalen Anwendungskomponente angefragt wurden, stellt eine massive Einschränkung des Optimierungspotentials der so genannten Mini-Max-Zielstellungen[12] dar. Diese Einschränkung ist jedoch aufgrund des spekulativen Charakters von Prefetchingverfahren sinnvoll. Durch die Einschränkung ist es unmöglich, dass ein Optimierungsverfahren zu einem Navigationsbaum eine Entscheidungsmatrix liefert, welche für eine ausgewählte Sequenz zu einer höheren Summe der Aufenthaltsdauern der Knoten bzw. zu einer höheren Aufenthaltsdauer eines Knotens dieser Sequenz führt als dies für eine Entscheidungsmatrix belegt nur mit Nullwerten, das heißt ohne Prefetching, der Fall wäre. Dadurch wird abgesichert, dass ein Prefetchingverfahren, basierend auf diesem Modell, nie eine schlechtere Lösung liefern kann, als dies der Fall wäre, wenn kein Prefetching zur Anwendung käme.

In Abbildung 3.3 ist eine aus vier Aufenthaltsknoten bestehende Session dargestellt. Sie setzt sich aus den von der lokalen Anwendungskomponente initiierten wechselnden Nutz- und Ladeaktivitäten zusammen. Die einzelnen Entscheidungszeitpunkte sind mit grauen Kreisflächen markiert. Zu diesen Zeitpunkten ist jeweils zusätzliches Wissen im Gegensatz zum vorherigen Zeitraum verfügbar, weil zu genau diesen Zeitpunkten eine durch die lokale Anwendung initiierte Anfrage eines Objekts bekannt wird. Die grau hinterlegten Flächen stellen den existierenden Entscheidungsspielraum dar. In diesem können Objekte bzw. Teile davon im Voraus geladen werden. Der Entscheidungsspielraum wird durch die feststehende Bandbreite des Nachrichtentransportkanals und der Nutzzeit des jeweils genutzten Objekts begrenzt. Die mit weißen Kreisen markierte Zeitpunkte symbolisieren auch Zustandsübergänge, insbesondere der

[12] $2MLB$ und $2MLS$ sowie die equivalenten $*$-Zielstellungen für redundante Navigationsbäume.

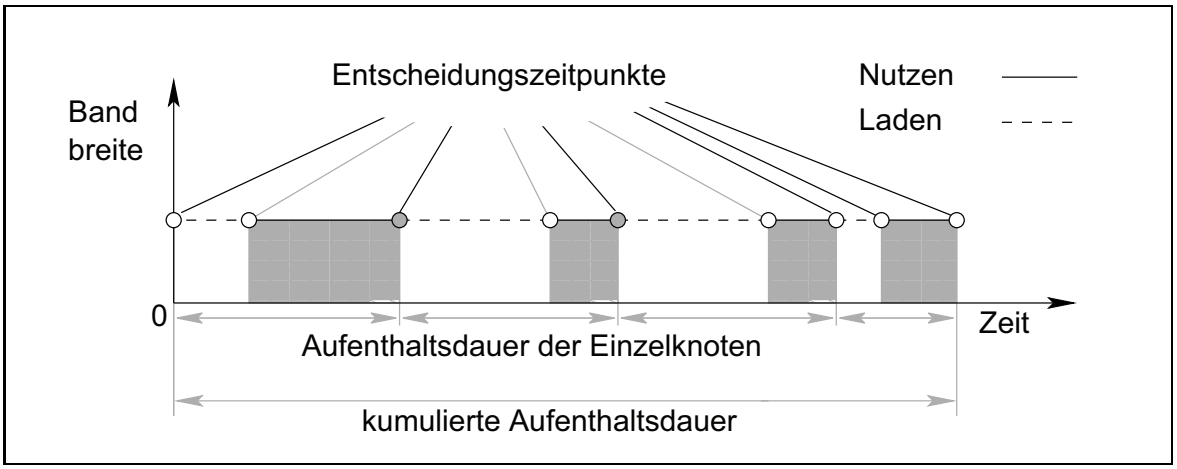



Abbildung 3.3: Entscheidungszeitpunkte und -spielraum in einer Session bestehend aus vier verschiedenen Aufenthalten

im Zeitablauf vorletzte Kreis auch einen Wissenszuwachs, jedoch ziehen diese Zeitpunkte im Rahmen der verfolgten Optimierungsziele keine Entscheidungen über im Voraus zu stellende Anfragen nach sich.[13]

Mit diesem Modell ist nur die Verwendung einer fixen Bandbreite des logischen Nachrichtentransportkanals zwischen lokaler Anwendung und entfernten Ressourcen entscheidungsrelevant. Das ist typisch für Content-Prefetchingverfahren. Verzögerungen, unabhängig von angefragter Größe eines Objektteils wie zum Beispiel in DNS-Prefetching- oder Connection-Prefetchingverfahren berücksichtigt, werden vernachlässigt. Der Einfluss solcher Entscheidungen liegt wie bereits erwähnt in Größenordnungen von maximal 100-500 Millisekunden[14] und spielt hier eine unbedeutende Rolle. Einflüsse dieser Art lassen sich zudem reduzieren, indem schon im Entwurf sowie spätestens mit der Implementation des Hypermediums darauf geachtet wird, dass Daten eines Knotens nicht in Form vieler Objekte kleiner Datenmengen auf unterschiedlichen entfernten Ressourcen abgelegt sind. Für die Nutzung von Ressourcen während der Nutzung eines Objekts werden keine Kosten bzw. Aufwände modelliert. Auch wird die Größe des Cachespeichers als beliebig groß angenommen.

Diese Annahmen folgen insbesondere dem Szenario einer „schlechten“ Netzanbindung, das heißt Verbindungen niedriger Bandbreite (56KBit/s  128KBit/s) zwischen den verteilten Anwendungskomponenten.[15] Der Verbindungsaufbau zwischen diesen erfolgt mit Beginn des Lernprozesses durch Anfordern der Daten des ersten Aufenthaltsknotens der Session. Der Verbindungsabbau kommt spätestens mit dem Ende der Nutzung der Daten des letzten Aufenthaltsknotens der Session zustande. Der Verbindung wird ein maximales Datenvolumen zugeschrieben, welches in der Zeitdauer, begrenzt durch Auf- und Abbau, übertragen werden kann. Kosten sollen nur zeitabhängig und nicht in Abhängigkeit von der übertragenen Datenmenge anfallen. Die Cachegröße spielt im Regelfall keine Rolle, da die im sinnvollen Zeitrahmen übertragba-

[13] Dieser Sachverhalt wird in der weiteren Diskussion der modellimmanenten Eigenschaften in Abschnitt 3.5.2, S.86ff. noch sichtbar.

[14] Vgl. dazu [PD03], S.43 und [CDK02], S.97

[15] Vgl. Kapitel „Grundlagen und weiterführende Überlegungen“, Abschnitt 2.4.1, S.45ff.

ren Datenmengen bei weitem nicht sinnvollen Größen des lokalen Caches entsprechen. Um ein MByte per Bandbreite von 56KBit/Sekunde zu übertragen, ist ein Zeitraum von knapp $2\frac{1}{2}$ Minuten zu veranschlagen. Die Anschaffungskosten von Festplattenkapazität liegen bei ca. 1Euro per GByte.[16] Damit ergibt sich ein Wert von unter 0,1 Cent per zu speicherndem MByte bei einer benötigten Speicherkapazität entsprechend eines Datenstroms, unterbrechungsfrei über einen Zeitraum von 80 Tagen empfangen. Ungefähr das Zehnfache wäre für eine 56 K/Bit-Modem-Verbindung zur Übertragung eines MBytes zu zahlen, wenn dadurch die Zeitdauer zwischen Verbindungsauf- und -abbau verlängert wird.

Weiterhin ist die gewählte Granularität der ganzzahligen Entscheidungsvariablen von 1 und deren Verhältnis zu den Nutz- und Ladezeiten zu diskutieren. Nicht mehr entscheidungsrelavant sind Verzögerungen unter 1 bis 2 Sekunden. Damit ergibt sich eine sinnvolle Fragmentierung der Objekte aus der festgelegten Bandbreite (ca. 4 - 30 KByte/Sekunde) je nach Verbindung zwischen lokaler Anwendungskomponente und entfernten Ressourcen von 4 bis 30 KByte. Antworten dieser Nachrichtenlänge entsprechen ungefähr Größenordnungen der maximal möglichen Nachrichtenlänge eines Pakets (MTU)[17] von 1.5KByte bis 4.5KByte (vgl. [PD03], S.243), die je nach Protokoll der Vermittlungsschicht mit einem Paket übertragen werden kann. Das Internet-Protokoll unterstützt eine MTU-Größe von bis zu 64KByte. Praktisch wird jedoch das Internet-Protokoll mit einer maximalen Paketlänge von 8KByte konfiguriert (vgl. [CDK02], S.105). Anfragen unter einer Nachrichtenlänge von 4KByte erscheinen im Rahmen von akzeptablen Verzögerungen unter einer Sekunde sinnlos. Eine Fragmentierung von Nachrichtenlängen über MTU-Größe erfolgt über die Vermittlungsschicht. Der damit verbundene zusätzliche Zeitaufwand ist weit unter einer Sekunde und kann vernachlässigt werden. Extrem kleine Nachrichtenlängen von wenigen Bits spielen hier keine Rolle, so dass die mit jedem Paket zu übertragenden Verwaltungsdaten vernachlässigt werden können.[18]

Dies gilt nicht unter der Annahme reellwertiger Entscheidungsvariablen. Die Aufhebung der Ganzzahligkeit der Entscheidungsvariablen führt dazu, dass mit beliebig kleinen angefragten Nachrichtenlängen mehr Nutzzeit zum Transfer der Daten in Anspruch genommen werden muss, als durch das Entscheidungsverfahren zugebilligt wird, weil von den zu übertragenden Verwaltungsinformationen im Modell abstrahiert wird. Die Menge der zu übertragenden Verwaltungsdaten je Anfrage wären zusätzlich zu berücksichtigen, ganz abgesehen davon, dass Nachrichtenlängen unter einem Bit auszuschließen sind.

Die Überführung des Navigationsgraphen in einen zyklenfreien Graphen in Form des Navigationsbaums und die Forderung nach Wegen zwischen Wurzel und Blatt, auf welchen redundante Knotenpaare existieren dürfen, ist keine Vereinfachung des Sachverhalts. Die Annahmen eines beliebig großen lokalen Cachespeichers und einer verzögerungsfreien Präsentation der im lokalen Cache verfügbaren Objekte führt dazu, dass für in einer Session wiederholt angefragte Objekte die Ladezeit Null angenommen werden kann. Deren Nutzzeit kann insbesondere aufgrund der wiederholten Nutzung gegenüber dem identischen, bereits genutzten Objekt stark divergieren. Aus diesem Grund werden spezielle Aufenthaltsknoten, die die Nutzung von Objekten repräsentieren, die wiederholt in einer Sequenz genutzt wurden, eingeführt. Ein solcher ist zu keinem anderen Knoten des Baums redundant und besitzt eine Ladezeit von Null. Die

[16] Stand Sommer 2005, Tendenz stark sinkend

[17] Englisch: maximum transmission unit (MTU)

[18] Die minimal mit jedem Paket zu übertragende Datenmenge liegt in Größenordnungen von wenigen Byte.

Nutzzeit ergibt sich aus der Dauer der wiederholten Nutzung des Objekts.

Eine Vereinfachung hingegen stellt die angenommene fixe Bandbreite der Verbindung zwischen lokalen Anwendungskomponenten und entfernten Ressourcen mittels der als fest vorgegeben angenommenen Ladezeiten der Objekte dar. Von geringfügigen Schwankungen während der Laufzeit der Anwendung wird abstrahiert. Gegebenenfalls sind die Ladezeiten so zu wählen, dass diese einer in Bezug auf mögliche Schwankungen eher niedrigeren zur Verfügung stehenden Bandbreite entsprechen.[19]

### 3.4.4 Optimierungsziele

Die Optimierungsziele des Offline- und des Online-Modells sind allein auf die Zeit bezogen. Zusätzlicher Aufwand durch Seiteneffekte wird nicht berücksichtigt.

Als primäre Zielstellung für Lernanwendungen wird die optimale Gestaltung des Lernprozesses angesehen. Wie bereits herausgestellt, spielt die Zeit hier eine zentrale Rolle. Zielstellungen bezüglich der im vorherigen Kapitel aufgeführten Seiteneffekte konkurrieren direkt mit den hier auf das Antwortzeitverhalten bezogenen Optimierungszielen. Auch wenn sie nicht Gegenstand der Optimierung sind, können sie auf Grundlage der Optimierungsergebnisse abgeschätzt werden. Modellformulierungen wären in Erwägung zu ziehen, die Seiteneffekte in Form von zusätzlichen Bedingungen im Modell berücksichtigen zum Beispiel der Art, dass eine optimale Lösung innerhalb eines festgelegten Rahmens des durch Prefetching verursachten Mehraufwands gesucht wird. Solche Betrachtungen werden mit dieser Arbeit nicht durchgeführt. Seiteneffekte werden hier aber keinesfalls als zu unterschätzendes Randthema angesehen. Sie werden immer wieder mit der Diskussion über die Güte der im Folgenden dargestellten Verfahren aufgegriffen.

Durch den Fokus auf zeitbasierte Ziele werden auch kostenbezogene Zielstellungen, sofern nicht die Zeit selbst als Kostenmaß interpretiert wird, vernachlässigt. Die Verwendung der Nutzzeit wird als alleiniger Entscheidungsgegenstand angesehen. Allein die Werte der so genannten Ladeentscheidungen sind Entscheidungsgegenstand. Lade- sowie Nutzzeiten selbst werden als Konstante angesehen, trotzdem das Entscheidungsfeld „Entwurfsentscheidungen“ gerade jene zum Entscheidungsgegenstand macht. Hier wird die Meinung vertreten, dass Entscheidungen, die Segmentierung und Sequenzierung der Lerninhalte betreffen, primär inhaltlich didaktischen Richtlinien zu folgen haben. Das sind Richtlinien, welche sich formal nur sehr begrenzt sinnvoll beschreiben lassen. Lösungen der hier dargestellten Optimierungsprobleme sind als reine Entscheidungshilfen innerhalb dieses von menschlichen Entscheidungen geprägten Entwicklungsprozesses zu sehen. Ziel muss es sein, Entscheidungen des Entwurfs eines Hypermediums zeitnah auch in Hinblick auf das zu erwartende Antwortzeitverhalten prüfen zu können insbesondere dann, wenn schon der Einsatz eines Prefetchingverfahrens in Erwägung gezogen wird. Die Auswahl eines Optimierungsziel[20] hängt letztendlich von den Präferenzen des Entscheidenden und dem Sachzusammenhang ab. Die einzelnen Optimierungsziele sollen hier in Hinblick auf Lernanwendungen nicht detailliert diskutiert werden. Eine grundsätzliche Motivation für die mit den Modellen getroffene Auswahl wurde im vorherigen Kapitel gegeben.

[19] Schwankungen entstehen beispielsweise durch eine variierende Belastung der Komponenten des Rechnernetztes und/oder bzw. gerade deshalb variierenden Nachrichtentransportwegen.

[20] Neben dem hier gezielt ausgesuchten Spektrum sind viele weitere denkbar.

Weiterhin fällt auf, dass innerhalb des Offline-Modells allein die Wartezeiten in Form von Restladezeiten der Aufenthaltsknoten Optimierungsgegenstand sind, Optimierungsziele des Online-Modells sich aber jeweils auf die Zeitdauer der Aufenthalte bezogen. Der Unterschied ist nicht in der Zielstellung selbst begründet. Nutzzeiten gehen als Konstanten in das Modell ein, so dass aus Restladezeiten auf Aufenthaltsdauern geschlossen werden kann und umgekehrt. Alleiniger Grund für diese oberflächliche Divergenz ist die unterschiedliche Schwerpunktsetzung der in diesem und in den beiden weiteren Kapiteln folgenden Analysen. Die Analyse der Eigenschaften und der zur Lösung der Online-Problemstellungen vorgeschlagenen Algorithmen beziehen sich auf die Aufenthaltsdauer. Schwerpunkt ist die Analyse des sich direkt aus den Nutzzeiten ergebenden Anwendungspotentials. Die Betrachtung optimaler Lösungen steht aufgrund des Online-Charakters erst an zweiter Stelle. Schwerpunkt der folgenden Analyse und Algorithmen des Offline-Modells ist hingegen die optimale Gestaltung der Restladezeiten.

### 3.4.5 Zusammenfassung der Annahmen

Die Modellierung mittels Navigationsbaum statt eines Graphen stellt keine Einschränkung dar. Dies beruht auch darauf, dass die Bedingungen für Zugriff, Speicherung und Transport von Daten so gestaltet sind, dass primär Szenarien „schlechter“ Netzanbindung unterstützt werden. Gerade solche Szenarien unterliegen schwerwiegenden kapazitiven Restriktionen, die zu einem schlechten Antwortzeitverhalten der Lernanwendung führen können. Sie sind deshalb von besonderem Interesse.

Ganz bewusst wird nicht die Segmentierung und Sequenzierung der Objekte zum Optimierungsgegenstand des Modells gemacht. Diese Entscheidungen sind Voraussetzungen des Modells und mittels der Optimierungsergebnisse zu prüfen und gegebenenfalls anzupassen.

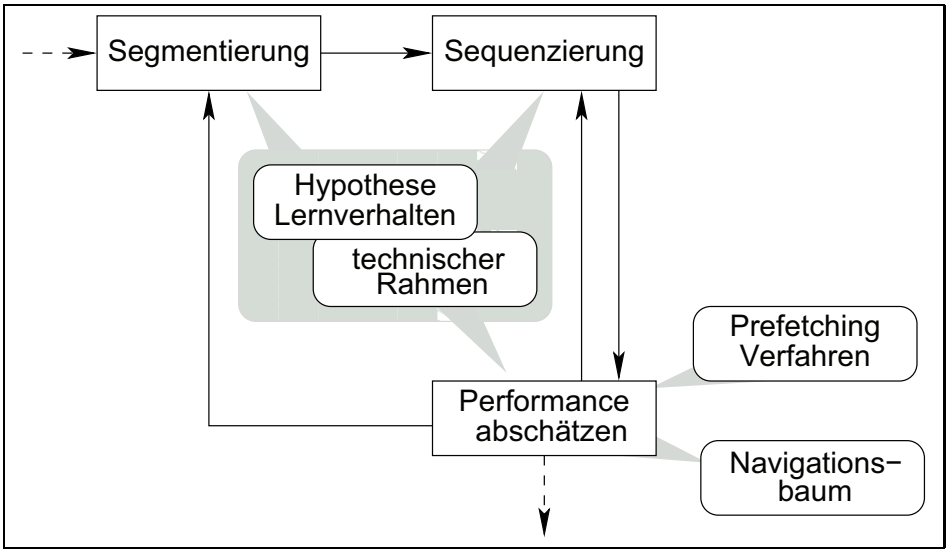



Abbildung 3.4: Iterativer Prozess der Segmentierung und Sequenzierung darzustellender Inhalte unter Berücksichtigung der Performance

Wie mit dem in Abbildung 3.4 angedeuteten Zusammenspiel der Entwurfsaktivitäten Segmentierung, Sequenzierung und Analyse der Performance im Rahmen des Performance Enginee-

rings als Ausschnitt des Entwurfsprozesses eines Hypermediums, werden Entscheidungen der Segmentierung und Sequenzierung von Inhalten des Hypermediums von Betrachtungen zur Performance des Systems indirekt über Rückkopplungen beeinflusst. Grundlage für die Entscheidungen sind die spezifizierten Rahmenbedingungen, insbesondere das Navigationsverhalten und der technische Rahmen[21], sowie zur Performanceabschätzung ein auszuwählendes Prefetchingverfahren, falls benötigt. Ergebnis ist das Hypermedium, hier inklusive der im Voraus getroffenen Ladeentscheidungen des Prefetchingverfahrens in Form eines Navigationsbaums.

Nutz- wie Ladezeiten des Hypermediums stellen im Gegensatz zur Modellannahme in der Praxis variierende Größen dar. Nutzerabhängige Schwankungen lassen sich mit jeweils nutzerbezogenen Navigationsbäumen darstellen. Jedoch unterliegen die Größen auch für einen einzelnen Nutzer Schwankungen. Hier wird sich bewusst von in der Literatur überwiegend verwendeten stochastischen Modellen gelöst, die explizit die Varianz der Größen mit Hilfe von parametrisierten Verteilungskurven darstellen. Die Vereinfachung reduziert die Komplexität der Problemstellungen. Damit werden die Modelle aus algorithmischer Sicht wesentlich einfacher fassbar und sind im Rahmen des Untersuchungsziels sinnvoll. Nutz- und Ladezeiten können so gewählt werden, dass sie Extreme des zu erwartenden Antwortzeitverhaltens darstellen.

Die Diskussion der gewählten Bedingungen und Zielstellungen ist damit abgeschlossen.

## 3.5 Diskussion grundlegender Modelleigenschaften

Aus den getroffenen Modellannahmen ergeben sich für Offline- und Online-Modell grundlegende Modelleigenschaften. Als grundlegend werden hier solche Modelleigenschaften angesehen, welche wesentlich auf die Konstruktion möglicher Lösungsverfahren für die formulierten Optimierungsprobleme wirken.

Für Optimierungsprobleme des Offline-Modells redundanter Navigationsbäume wird die Komplexität untersucht, um die Rechenbarkeit potenziell möglicher Optimierungsverfahren zu bewerten.[22] Aus den Ergebnissen folgend werden zwei Möglichkeiten zur Relaxation aufgezeigt. Aus der jeweiligen Relaxation sich ergebende Lösungsverfahren sind dann Gegenstand des nächsten Kapitels 4.

Die Untersuchungsergebnisse zur Rechenbarkeit und Relaxation wirken auch auf die Konstruktion von Lösungsverfahren für Optimierungsprobleme des Online-Modells. Jedoch von besonderem Interesse sind für Lösungsverfahren des Online-Modells die Auswirkungen der angenommenen unsicheren Informationen. Deshalb wird für das Spektrum sinnvoll möglicher Lösungsverfahren des Online-Modells eine komperative Analyse in Form einer Kompetitivitätsanalyse vorgenommen. Es wird die Situation der Entscheidung bei Unsicherheit über zukünftige Informationen mit der Situation der Entscheidung bei vollständiger Information über die Zukunft verglichen. Dadurch lassen sich Aussagen über die Güte etwaiger Lösungsverfahren im Vergleich zum Bestmöglichen treffen.

[21] Rechnernetzstrukur und -kapazitäten

[22] Da redundante Navigationsbäume eine Verallgemeinerung nicht redundanter Bäume darstellen, ist in einem ersten Schritt die Untersuchung des Modells allein für redundante Bäume ausreichend.

### 3.5.1 Komplexität der Optimierungsprobleme des Offline-Modells für redundante Navigationsbäume

In diesem Abschnitt wird gezeigt, dass die Optimierungsprobleme des Offline-Modells für redundante Navigationsbäume mindestens genauso schwer lösbar sind wie Probleme der Klasse *NPC* und damit für beliebig große Probleminstanzen als schwer rechenbar einzustufen sind.[23]

Es sei angenommen, dass das Optimierungsproblem *oP* Probleminstanzen der fünf Optimierungsprobleme des Offline-Modells für redundante Navigationsbäume beschreibt:

$$oP \in \{2MLB^*, MSLB^*, 2MLS^*, 2MSLS^*, MSLS^*\}, \tag{3.25}$$

wobei *oP* dann folgende Struktur besitzt:

> Optimierungsproblem *oP*
>
> Probleminstanz: Gegeben ist eine Probleminstanz bestehend aus redundantem Navigationsbaum, Nebenbedingungen des Offline-Modells und Zielfunktion eines der fünf Optimierungsprobleme $2MLB^*, MSLB^*, 2MLS^*, 2MSLS^*$ und $MSLS^*$.
>
> Gesucht ist entsprechend der Zielfunktion der Zielfunktionswert für die gegebene Probleminstanz.

Es wird gezeigt, dass folgender Satz gilt:

**Satz 3.1** $oP \in NP-hart,$

wobei ein $NP-hartes$ Optimierungsproblem mindestens so schwer lösbar ist, wie ein *NP*-vollständiges Entscheidungsproblem (vgl. [Weg93], S. 45). Der Satz bestätigt die zu Beginn dieses Abschnitts aufgestellte Behauptung.

**Beweis von Satz 3.1:** Um den Satz bestätigen zu können, wird ein zum Optimierungsproblem *oP* korrespondierendes Entscheidungsproblem *eP* formuliert, für welches gezeigt werden soll:

$$eP \leq_p oP \text{ und } eP \in NPC \tag{3.26}$$

Das Optimierungsproblem *oP* ist *NP*-hart, wenn ein *NP*-vollständiges Problem, hier *eP*, polynomiell auf *oP* reduzierbar ist. Das heißt, es existiert eine polynomielle Transformation für alle Probleminstanzen des *eP* nach Instanzen des *oP*.[24]

Das Entscheidungsproblem *eP* sieht wie folgt aus:

[23] *NPC* umschreibt die Menge der **n**ichtdeterministisch **p**olynomial vollständig (engl. **c**omplete) lösbaren Entscheidungsprobleme (vgl. [Weg93], S. 45).

[24] Vgl. zur polynomiellen Reduktion [Weg93], S.43.

Entscheidungsproblem *eP*

Probleminstanz: Gegeben ist eine Probleminstanz des Optimierungsproblems *oP* und ein Wert $z$.

Gesucht ist eine Antwort auf die Frage: Existiert für die gegebene Probleminstanz ein Lösungswert kleiner als $z$?

um zu zeigen, dass 3.26 gilt, ist zu zeigen, dass folgende Lemma gelten:

**Lemma 3.2** $eP \leq_p oP$

sowie zum Nachweis von $eP \in NPC$

**Lemma 3.3** $eP \in NP$

und

**Lemma 3.4** $3DM \leq^p eP,$

wobei $3DM$ das *NP*-vollständige Entscheidungsproblem 3-Dimensionales Matching ist und noch beschrieben wird.[25]

**Beweis von Lemma 3.2:** Der Nachweis ist trivial, denn für eine gegebene Probleminstanz des Entscheidungsproblems des *eP* kann ein Optimalwert mit einem Lösungsverfahren für *oP* bestimmt werden. Ist dieser Wert größer als $z$ aus *eP*, dann ist *eP* mit „nein“ und sonst mit „ja“ zu beantworten.

Lemma 3.2 kann bestätigt werden. □

**Beweis von Lemma 3.3:** Das Entscheidungsproblem *eP* gehört genau dann der Klasse *NP* an, wenn ein Zeuge, der die Antwort „ja“ auf die obige Frage bestätigen kann, als ein solcher auch mit polynomial begrenztem Rechenzeit- und Speicherplatzaufwand, soll hier heißen effizient, verifiziert werden kann (vgl. [Weg93], S.39). Hier ist der Zeuge für eine gegebene Probleminstanz die Wertebelegung einer $n \times n$ großen Entscheidungsmatrix $X$, wie sie mit dem Offline-Modell definiert wurde.[26]

Es ist leicht zu erkennen und soll deshalb nicht weiter diskutiert werden, dass in Abhängigkeit von der Anzahl der Knoten des Baums für einen gegebenen redundanten Navigationsbaum geprüft werden kann, ob die $n \times n$ Entscheidungsvariablen $x_{ij}$ der Entscheidungsmatrix $X$ den Nebenbedingungen 3.8 bis 3.12, S.65ff. des Offline-Modells genügen.[27] Aus einer Entscheidungsmatrix und den Ladezeiten der Knoten des Baums lassen sich mit einem Aufwand von

[25] Lemma 3.3 und 3.4 beschreiben die beiden Eigenschaften *NP*-vollständiger Probleme. Zum Beispiel sind in [ALR99] oder [GJ79] theoretische Grundlagen eines Nachweises der *NP*-Vollständigkeit eines Problems ausführlich dargestellt.

[26] Siehe S.3.3.1

[27] Jede dieser Bedingungen kann für beliebige Probleminstanzen mit einem Rechenzeit- wie Speicherplatzaufwand je Entscheidungsvariable von $O(n^2)$ geprüft werden.

$O(n^2)$ die Restladezeiten der Knoten und damit auch der jeweils optimale Zielfunktionswert effizient ermitteln.[28]

Damit ist Lemma 3.3 zu bestätigen. □

**Beweis von Lemma 3.4:** Zum Nachweis wird zuerst das Problem 3-Dimensionales-Matching (3DM) in Anlehnung an die Beschreibung in [ALR99] dargestellt. Dann werden spezielle Probleminstanzen aus *eP* durch Formulieren eines weiteren Entscheidungsproblems *ERL* zusammengefasst. Deren optimale Lösung kann dazu verwendet werden, beliebige Probleminstanzen von 3*DM* mit einem polynomiellen Transformationsverfahren optimal zu lösen. Zum endgültigen Nachweis der *NP*-Vollständigkeit von *eP* ist zu zeigen, dass beliebe Probleminstanzen von 3*DM* polynomiell in Probleminstanzen von *ERL* transformiert werden können sowie deren optimale Lösungen polynomiell in optimale Lösungen des 3*DM* transformierbar sind.

Entscheidungsproblem 3-Dimensionales Matching

Probleminstanz: Gegeben sind die Mengen $W, X, Y$ mit jeweils $q = |W| = |X| = |Y|$ Elementen und die Menge $S \subseteq W \times X \times Y$.

Gesucht ist eine Antwort auf die Frage: Enthält $S$ ein Matching, d.h. eine Untermenge $S' \subseteq S$, so dass $|S'| = q$ und keine zwei Tripel in $S'$ in irgendeiner Koordinate übereinstimmen?

Das Problem 3*DM* ist *NP*-vollständig (vgl. [ALR99], S.28-7).

Die folgende Problemstellung beschreibt spezielle Probleminstanzen von *eP*.

[28] Zum Ermitteln des Werts können die im folgenden Kapitel angegebenen, jeweils effizienten Funktionen `Funktionswert` für die Zielstellungen für nicht redundante Navigationsbäume verwendet werden.

Entscheidungsproblem Einfaches „redundantes“ Ladeproblem ($ERL$)

Probleminstanz: Gegeben ist eine Zielfunktion $Z$ aus $eP$ und ein redundanter Navigationsbaum $B$ mit den folgenden speziellen Eigenschaften sowie den Gruppen $G$ der Knoten entsprechend der Gruppenzugehörigkeit $G_k$ eines jeden Knotens.

- Die Ladezeit aller Knoten, exklusive der Wurzel des Baums, ist 1.
- Das Objekt des Wurzelknotens $a_r$ wird $3q+q$ Zeiteinheiten genutzt. Die Ladezeit des Wurzelknotens beträgt Null.
- Der Baum besteht aus $3q = |W| + |X| + |Y|$ Teilbäumen $T_i \neq B$, dessen Wurzelknoten direkte Kinder von $a_r$ sind, wobei die Mengen $W, X, Y$ aus den Elementen $w_j, x_j, y_j, j = 1 \ldots q$ bestehen.
- Zu jedem Element der Mengen $W, X, Y$ existiert genau ein Teilbaum $T_i$ mit jeweiligem Wurzelknoten $a_i = w_j$, $a_i = x_j$ oder $a_i = y_j$ mit jeweils speziell gewählten Nutzzeiten $\hat{u}_i$ der Art $a_i = \{o, \hat{u}_i, \ell_i = 1\}$. Jeder dieser Wurzelknoten $a_i$ solch eines Teilbaums $T_i$ ist ein nicht redundanter Knoten:

$$\begin{aligned} &\forall w_j \in B \; \exists G_k \in B : G_k = \{w_j\} \\ &\forall x_j \in B \; \exists G_k \in B : G_k = \{x_j\} \\ &\forall y_j \in B \; \exists G_k \in B : G_k = \{y_j\} \end{aligned} \tag{3.27}$$

- Jeder dieser Wurzelknoten $w_j, x_j, y_j$ eines Teilbaums hat genau $\hat{u}_i$ Kindknoten, welche Blätter sind.
- Alle Knoten der Teilbäume mit den Wurzelknoten genau einer Menge $W$, $X$ oder $Y$ enthalten nur zueinander nicht redundante Knotenpaare. Damit gilt für jede Gruppe zueinander redundanter Knoten:

$$\forall G_k \in B : |G_k| \leq 3 \tag{3.28}$$

Gesucht ist eine Antwort auf die Frage: Existiert eine Lösung mit dem Zielfunktionswert Null auch dann, wenn die Nutzzeiten der Objekte in den Knoten $w_j, x_j, y_j, j = 1 \ldots q$ nicht $\hat{u}_i$ sondern $\hat{u}_i^* = \hat{u}_i - 1$ beträgt?

Es entsteht ein Baum aus dem ersten bis dritten Knotenniveau: Wurzelknoten, alle direkten Kinder der Wurzel und alle Blattknoten. Der Baum ist in Abbildung 3.5 dargestellt. Für jedes der fünf Ziele muss ein optimaler Lösungswert von Null für beliebige Probleminstanzen von $ERL$ existieren, weil der Baum entsprechend der Knotenniveaus $1, 2$ und 3 folgende Knotenanzahl $|A|$ enthält

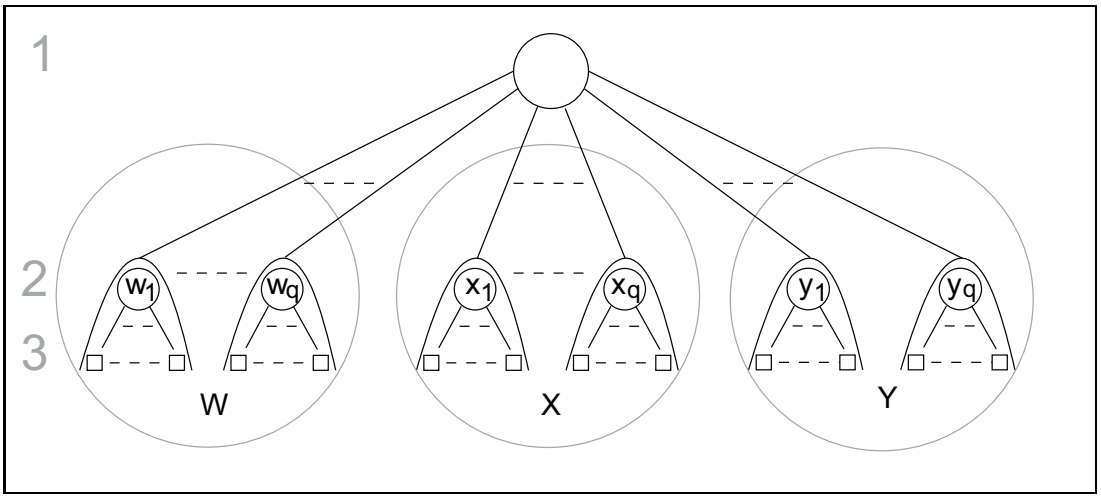



Abbildung 3.5: Baumstruktur beliebiger Probleminstanzen aus $ERL$

$$
\begin{aligned}
|A| &= N_1 + N_2 + N_3 \\
N_1 &= 1 \\
N_2 &= 3q \\
N_3 &= \left( \sum_{\forall a_i \in W} \hat{u}_i \right) + \left( \sum_{\forall a_i \in X} \hat{u}_i \right) + \left( \sum_{\forall a_i \in Y} \hat{u}_i \right)
\end{aligned}
\tag{3.29}
$$

und die Ladezeit aller Knoten, außer der Wurzel, 1 beträgt sowie die Nutzzeiten ermöglichen, dass die Knoten des Niveaus 2 und 3 vollständig geladen werden können. Die Objekte der jeweiligen Elternknoten des darüber liegenden Niveaus werden mindestens so lange genutzt, wie das Laden aller Objekte der jeweiligen direkten Kindknoten des darunter liegenden Niveaus in Anspruch nimmt.

Der Zielfunktionswert muss jedoch nicht Null sein, wenn die Nutzzeiten der Objekte in den Knoten des zweiten Niveaus um 1 auf $\hat{u}_i^*$ gesenkt werden. Genau dann können $2q$ Knoten nicht geladen werden, wenn der Baum nur nicht redundante Knoten enthalten würde.

Als nächstes wird eine für den Nachweis des Lemmas interessante Eigenschaft von optimalen Lösungen des $ERL$ mit der folgenden Bemerkung betrachtet. Dazu sind sogenannte redundante Knotentripel von Interesse.

**Definition:** *Redundante Knotentripel* sind zueinander redundante Knoten, welche genau einer Gruppe $G_k \in B$ angehören. Die Knoten eines solchen Tripels einer Probleminstanz von $ERL$ mit Ladezeit 1 enthalten alle ein Objekt, welches laut Modelldefinition für zueinander redundante Knoten in einer Einheit Nutzzeit eines Aufenthaltsknotens $a^*$ geladen werden kann, wenn alle Knoten dieser Objekte sich im selben Teilbaum mit Wurzel $a^*$ befinden.[29] □

[29] Siehe Modellbedingung 3.9, S.65

**Bemerkung:** *Nur wenn in der Menge aller Blattknoten $A^* \subset A$ genau q redundante Knotentripel $S^*$ existieren, wobei jeder Knoten der Tripel auch nur genau einmal in genau einem Tripel enthalten ist, dann ist der optimale Zielfunktionswert auch dann Null, wenn die Nutzzeiten der Knoten des zweiten Niveaus nicht jeweils $\hat{u}_i$ sondern $\hat{u}_i^* = (\hat{u}_i - 1)$ betragen.*

Dazu die folgenden Erklärungen:

Um alle Objekte der nicht redundanten Knoten des zweiten Niveaus zu laden, muss $3q$ Nutzzeit der Wurzel verwendet werden. Das heißt, $3q$ Zeiteinheiten der Nutzzeit des Wurzelknotens müssen zum Laden aller Objekte aus Knoten des zweiten Niveaus verwendet werden, wenn der Zielfunktionswert Null beträgt. Es verbleiben $q$ Zeiteinheiten Nutzzeit des Wurzelknotens zum Laden von Objekten von Blattknoten.

Die jeweils $\hat{u}_i^*$ Einheiten Nutzzeit der $3q$ Wurzelknoten des zweiten Niveaus reichen jeweils genau dazu aus, um alle Objekte bis auf eines der Blattknoten eines Teilbaums vollständig zu laden, weil alle Knoten jedes Teilbaums mit Wurzel im zweiten Niveau zueinander nicht redundant sind.

Es verbleiben damit genau $3q$ Knoten, jeweils ein Kindknoten zu jedem Knoten des zweiten Niveaus, deren Objekte nicht geladen werden können. Diese $3q$ Knoten können genau dann in den restlichen $q$ Zeiteinheiten Nutzzeit der Wurzel geladen werden, wenn diese $q$ zueinander redundante Knotentripel bilden.

Existieren nur $q-1$ zueinander redundante Knotentripel, dann müssen mindestens drei Knoten existieren, welche mindestens zwei Gruppen angehören und damit nur in zwei Zeiteinheiten vollständig geladen werden können. Somit sind zum Laden der verbleibenden $3q$ Blattknoten nicht $q$ sondern $q+1$ Zeiteinheiten Nutzzeit der Wurzel notwendig. Da jedoch nur $q$ Zeiteinheiten zum Laden der Objekte dieser Menge Blattknoten zur Verfügung stehen, muss mindestens ein Knoten im Baum existieren, dessen Restladezeit größer Null ist. Dann ist der Zielfunktionswert für alle Funktionen aus $Z$ größer Null.

Die obige Bemerkung ist nachvollzogen worden.

Um nun die Gültigkeit von Lemma 3.4 nachzuweisen, soll mit Hilfe oben erläuterter Bemerkung gezeigt werden, dass jede beliebe Probleminstanz aus $3DM$ mit polynomiell in Abhängigkeit von $|S|$ und $q$ in eine Probleminstanz von $ERL$ transformiert werden kann. Weiterhin wird gezeigt, wie aus der optimalen Lösung einer Probleminstanz von $ERL$ auf eine optimale Lösung für $3DM$ mit polynomiell beschränktem Aufwand geschlossen werden kann.

Dazu wird jedes Tripel $(w_h \in W, x_i \in X, y_j \in Y)$ aus $S$ in $3DM$ nach $ERL$ überführt, indem zu jedem Element des Tripels genau ein Blattknoten existiert, welcher dem Teilbaum mit Wurzel $w_h$, $x_i$ oder $y_j$ angehört. Diese Blattknoten bilden jeweils ein redundantes Knotentripel. Sie gehören genau einer Gruppe an, welche auch nicht mehr Knoten enthält. So entsteht zu jedem Tripel in $S$ genau eine Gruppe, das heißt insgesamt $|S|$ Gruppen.

Der Baum enthält damit $3 \cdot |S|$ Blattknoten, $3q$ Knoten des zweiten Niveaus und den Wurzelknoten: insgesamt $3|S| + 3q + 1$ Knoten und um eins weniger Kanten sowie $|S| + 3q + 1$ Knotengruppen. Die Objekte der Knoten $a_i$ des zweiten Niveaus werden $\hat{u}_i^*$ Zeiteinheiten genutzt.

Existiert eine optimale Lösung gleich Null für jede so generierte Probleminstanz, dann müssen in ihr laut obiger Bemerkung $q$ redundante, aus Blattknoten des Baums bestehende Knotentripel mit $S^*$ enthalten sein. Jedes dieser Tripel gehört zu genau einer Knotengruppe, welche ein bestimmtes Tripel aus $S$ repräsentiert. Jede Knotengruppe ist genau so konstruiert, dass zu ihr

jeweils genau ein Blattknoten existiert, der einem Element aus $W$, einem aus $X$ und einem aus $Y$ entspricht. Auch können keine zwei Blattknoten in $S^*$ enthalten sein, welche dasselbe Element aus $W$, $X$ oder $Y$ symbolisieren, weil in $S^*$ jeweils nur ein Blattknoten jedes Teilbaums mit Wurzel des zweiten Niveaus enthalten sein kann, wenn der Zielfunktionswert Null ist.

Die $q$ Gruppen der $3q$ Blattknoten aus $S^*$ repräsentieren das gesuchte Matching $S'$ für $3DM$ bei einem Optimalwert Null. Die gesuchte Antwort für die Probleminstanz aus $3DM$ ist dann „ja“. Ist der optimale Wert größer Null, dann kann eine solche Menge von $q$ Tripeln, die den Eigenschaften von $S'$ aus $3DM$ entspricht, nach der obigen Bemerkung für $S$ nicht existieren. $3DM$ ist mit „nein“ zu beantworten.

Demnach ist Lemma 3.4 zu bestätigen, weil die Menge aller mit $ERL$ definierten Probleminstanzen eine Teilmenge der Probleminstanzen von $eP$ ist. □

Lemma 3.2, 3.3 und 3.4 sind bestätigt worden. Somit ist auch Satz 3.1 korrekt.[30] □

Die $NP$-Härte der Optimierungsprobleme des Offline-Modells basierend auf redundanten Navigationsbäumen hat zur Folge, dass nicht erwartet werden kann, dass ein für deren Lösung effizientes, das heißt mit polynomiell begrenztem Aufwand rechenbares, Verfahren existiert, welches auch für große Problemumfänge praktikabel einsetzbar optimale Lösungen berechnet. Deshalb ist es sinnvoll, Relaxationen der Optimierungsprobleme zu betrachten, die zu effizient rechenbaren Optimierungsverfahren führen. In dieser Arbeit werden zwei Relaxationen betrachtet. Es wird die Nebenbedingung der Ganzzahligkeit für die Entscheidungsvariablen relaxiert[31], indem reellwertige Entscheidungsvariablen angenommen werden, und es werden Optimierungsziele allein für nicht redundante Navigationsbäume, wie mit den Zielstellungen ohne „ * “ bereits formuliert[32], betrachtet. Lösungsverfahren unter beiden Relaxationen werden im nächsten Kapitel 4 betrachtet. Vorausblickend sei kurz erwähnt, dass unter der Bedingung nicht redundanter Navigationsbäume alle Zielstellungen, dass heißt ohne „ * “ markiert, effizient optimal gelöst werden können und dass unter der Bedingung reellwertiger Entscheidungsvariablen ein effizientes Approximationsverfahren für alle Ziele des Offline-Modells angegeben werden kann, aber dessen Güte polynomiell abhängig von der Höhe des Navigationsbaums ist.

Damit ist die grundlegende Eigenschaft des Offline-Modells herausgearbeitet worden. Im Folgenden wird das Online-Modell betrachtet.

### 3.5.2 Komparative Analyse des Online-Modells

Für das Online-Modell ist zu klären, wie der spekulative Charakter eines Prefetchingverfahrens, modelliert durch eingeschränkt verfügbare Informationen, die zu erwartende Güte von Lösungsverfahren der beiden Optimierungsprobleme beeinflusst. Da eine optimale Lösung nicht erwartet werden kann, spielen die Abschätzung von Rechenzeit- sowie Speicherplatzaufwand zur Konstruktion einer solchen eine untergeordnete Rolle. Als viel bedeutender ist einzuschätzen, welcher Wert dem Einsatz eines Prefetchingverfahrens sowie den von diesem benötigten Informationen beizumessen ist.

[30] Die $NP$-Härte von $oP$, für Probleminstanzen, in welchen Knotengruppen mit zwei zueinander redundanten Knoten enthalten sind, ist bis jetzt nicht gelungen. Jedoch vermutet der Autor, dass auch diese schon $NP$-hart sind. Es wurde versucht, entsprechende Problemstellungen in Form von effizient lösbaren Flussproblemen zu formulieren. Dies ist jedoch nicht gelungen.

[31] Vgl. Definition „Entscheidungsvariablen“, S.65

[32] Vgl. Optimierungsziele 3.16-3.20, S.67ff.

Dazu wird die Kompetitivitätsanalyse als eine Methode der vergleichenden Analyse des Outputs eines Online-Algorithmus im Verhältnis zum Output eines Referenz-Algorithmus, i.d.R. eines korrespondierenden optimalen Offline-Algorithmus, gewählt.[33]

Hier soll diese Analysetechnik dazu verwendet werden, das Anwendungspotential von möglichen Algorithmen zur Lösung der Problemstellungen *MDS* bzw. $MDS^*$ und $2MDS$ bzw. $2MDS^*$ abzuschätzen. Dazu werden jeweils zwei Algorithmen *ALG* und *OPT* verwendet, welche die gesamte Bandbreite sinnvoller Lösungsverfahren zur Lösung des Optimierungsproblems aufspannen. Die beiden Algorithmen stellen eine untere (*OPT*) und eine obere Schranke (*ALG*) bezüglich der Optimierungsergebnisse sinnvoller Lösungsverfahren dar. *OPT* stellt den bereits erwähnten optimalen Offline-Algorithmus mit vollständigem Wissen über die Inputsequenz bzw. über zukünftiges Wissen dar. *ALG* ist hier ein Verfahren, welches in Bezug auf die obere Schranke gerade noch als sinnvoll anzunehmen ist. Es werden Optimierungsentscheidungen ohne Kenntnis über die Zukunft betreffende Informationen gefällt. *ALG* erhält damit nur eingeschränkt Kenntnis über die Inputsequenz. *ALG* muss nicht zwangsläufig innerhalb einer solchen Analyse ein Online-Verfahren sein, deshalb die Benennung mit „*ALG*“, stellt aber hier als Repräsentant für Lösungsverfahren von *MDS*, $MDS^*$ und $2MDS$, $2MDS^*$ jeweils ein solches dar. Die vier Problemstellungen sind Online-Probleme, so dass *ALG* in dieser Arbeit ein Online-Probleme lösendes Verfahren ist und deshalb im weiteren Verlauf mit „*ON*“ bezeichnet wird. Die Kompetitivitätsanalyse wird hier eingesetzt, um zu klären, welcher Wert der Kenntnis von Informationen über die Zukunft im Vergleich zu deren Unkenntnis beizumessen ist, die für die angestrebte optimale Lösung des jeweiligen Online-Optimierungsproblems notwendig wären.[34] Im Gegensatz zur klassischen Analyse durchschnittlicher Fälle von Prefetchingverfahren, modelliert mittels stochastischer Modelle, welche Kenntnisse bzw. Hypothesen zur Definition durchschnittlicher Fälle voraussetzt[35], wird mit der Kompetitivitätsanalyse eines Prefetchingverfahrens im Vergleich zum Referenzverfahren geprüft, um welche Größenordnung sich die Optimierungsergebnisse beider Verfahren maximal voneinander unterscheiden können. Es handelt sich dabei um die Analyse der schlechtesten Fälle des Online-Verfahrens in Relation zum Referenzverfahren. Als Referenzverfahren wird hier ein streng ideales Prefetchingverfahren angenommen.

Zur kompetitiven Analyse sind vorab einige Definitionen notwendig.

**Definition:** Der Online-Algorithmus *ON* ist für alle möglichen endlichen Inputsequenzen $S$ *c-kompetitiv*, wenn zwei Konstanten $c$ und $\alpha$ für jede dieser Inputsequenzen $seq \in S$ existieren, für die gilt:

$$ON(\hat{I}(seq)) \leq c \cdot OPT(I(seq)) + \alpha \tag{3.30}$$

□

Algorithmus *OPT* ist ein Offline-Algorithmus, welcher für ein gegebenes Minimierungsproblem eine optimale Lösung mit dem Wert $OPT(I(seq))$ bei vollständiger Information $I$ über die

[33] Diese Analysetechnik zur Abschätzung der Lösungsgüte von Online-Algorithmen im Vergleich zu Offline-Algorithmen wurde von *Slator* und *Tarjan* erstmals in [ST85] systematisch zur Diskussion des „List Accessing Problem“ genutzt. Eine ausführliche Einführung in diese Analysetechnik von Allan Borodin und Ran El-Yaniv ist in [BE98] zu finden.

[34] Mit dem Wissen über den Wert der Informationen lässt sich gegebenenfalls auch der Wert eines optimalen Offline-Verfahrens relativieren, wenn dieses praktisch nicht einsetzbar wäre.

[35] Vgl. beispielsweise die Ausführungen von *Khan* und *Tao* in [KT01], welche Verzögerungsdauern während einer Session hochrechnen, indem sie Eintrittswahrscheinlichkeiten für das Stellen bestimmter Anfragen und Wahrscheinlichkeitsverteilungen für die Dauer der Nutzung der angefragten Daten annehmen.

Inputsequenz *seq* ermittelt.[36] Dem gegenüber löst *ON* dasselbe Minimierungsproblem mit dem Wert $ON(\hat{I}(seq))$, jedoch bei eingeschränkter Information $\hat{I}$ über *seq*.

**Definition:** Existieren ein $c$ und ein $\alpha = 0$, welche Ungleichung 3.30 erfüllen, wird Algorithmus *ON* auch *strikt c-kompetitiv* genannt (vgl. [BE98], S.3). □

Ein Online-Algorithmus wie *ON* besitzt keine Informationen über die Zukunft. Seine zu bestimmten Zeitpunkten getroffenen Entscheidungen basieren gegenüber einem Offline-Algorithmus jeweils nur auf Informationen der Vergangenheit und Gegenwart.

**Satz 3.5** *Für die Optimierungsprobleme der Ziele MDS und MDS* gilt, dass jedes sinnvolle Online-Lösungsverfahren strikt* 2*-kompetitiv ist.*

Innerhalb des folgenden Beweises wird genauer darauf eingegangen, was unter einem noch sinnvollen bzw. einem nicht sinnvollen Lösungsverfahren verstanden wird. Zuvor werden zur Kompetitivitätsanalyse die einzelnen Komponenten für die vorgegebene Problemstellung zusammengefasst:

$seq \in S$**:** Die Inputsequenz *seq* der beiden Algorithmen setzt sich aus der Session $s_1^m$ und dem Navigationsbaum $B$ zusammen.

$\hat{I}(s_1^m, B)$ **und** $I(s_1^m, B)$**:** Die beiden Funktionen modellieren die jeweils für *ON* und *OPT* zur Verfügung stehenden Informationen über $s$ und $B$.

Die Informationsfunktion $I(s_1^m, B)$ liefert *OPT* alle Informationen über $s_1^m$ und $B$, um eine optimale Lösung finden zu können. Das sind Informationen über:

- die Menge aller anforderbaren Objekte mit deren Lade- und Nutzzeiten, dass heißt alle in $B$ enthaltenen Aufenthaltsknoten $(A)$ und deren Verknüpfung sowie
- die Reihenfolge, in der die $m$ Objekte zur Nutzung angefordert werden $(s_1^m)$.

Demgegenüber liefert $\hat{I}(s_1^m, B)$ dem Verfahren *ON* nur eingeschränkte Informationen über $s$. Das sind Informationen über:

- die Menge aller potentiell in $s$ enthaltenen Knoten $(A)$ mit deren Ladezeiten, deren mögliche Reihenfolgen innerhalb einer Session sowie
- alle bis zum Zeitpunkt $t(s_i^i)$ an den lokalen Cache gelieferten Antworten.

  Damit besitzt *ON* keine Informationen über die von der lokalen Anwendungskomponente initiierten und auf den gegenwärtigen Zeitpunkt $t(s_i^i)$ folgenden Anfragen der Session, das heißt über die Menge aller in $s_{i+1}^m$ enthaltenen Aufenthaltsknoten.

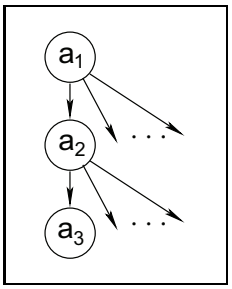
  Der Zeitpunkt $t(s_i^i)$, das heißt wann das $i$-te Objekt der Session angefragt wird, ergibt sich aus der Sessiondauer bis zu diesem Zeitpunkt $D(s_1^{i-1})$, welche sich wiederum aus den vom Lösungsverfahren getroffenen Entscheidungen, den daraus folgenden Restladezeiten und der Reihenfolge der Knoten in $s_1^{i-1}$ ergibt. Der Anforderungszeitpunkt des ersten Objekts der Session steht fest.

[36] Ebenso können Maximierungsprobleme betrachtet werden. Das $c$-fache von $OPT(I(seq))$ und einer additiven Konstante $\alpha$ darf nicht unterschritten werden ($ON(\hat{I}(seq)) \geq c \cdot OPT(I(seq)) - \alpha$).

$OPT(I(s_1^m, B))$ **und** $ON(\hat{I}(s_1^m, B))$**:** Der Offline- sowie der Online-Algorithmus erzeugen für eine Inputsequenz eine im Regelfall voneinander verschiedene Folge von Ladeentscheidungen $X_i, i = t(s_i^i)$ für die Entscheidungszeitpunkte $t(s_1^1) \ldots t(s_i^i) \ldots t(s_m^m)$.

Die gewählten Zeitpunkte werden deshalb als sinnvoll erachtet, weil *ON* zwischen zwei Zeitpunkten $t(s_i^i)$ und $t(s_{i+1}^{i+1})$ keine zusätzlichen Informationen erhält, so dass für diesen Zeitraum zum Zeitpunkt $t(s_i^i)$ alle Entscheidungen getroffen werden können. Das Verfahren *ON* wird so für jeden Zeitpunkt eine Entscheidungsmatrix in diesem Zeitpunkt generieren. Auch *OPT* generiert für jeden Zeitpunkt eine solche Matrix, jedoch kann dieses einmalig schon zum ersten Entscheidungszeitpunkt passieren. Benötigte Rechenzeit zur Entscheidungsfindung wird hier vernachlässigt.

In Abbildung 3.6 ist eine Beispielsession mit drei zueinander nicht redundanten Knoten in der Reihenfolge $a_1$, $a_2$, $a_3$ sowie deren Lade- und Nutzzeiten dargestellt. Die Abbildungen 3.7 und 3.8 sowie die dazugehörigen Entscheidungsmatrizen in Tabelle 3.1 stellen zwei mögliche Abfolgen von Laden und Nutzen der Objekte dieser Knoten dar.




| | Knoten | | |
|---|---|---|---|
| | $a_1$ | $a_2$ | $a_3$ |
| $\ell$ | 2 | 2 | 2 |
| $u$ | 1 | 2 | 1 |

Abbildung 3.6: Beispielsession bestehend aus drei Knoten (links) und Lade- sowie Nutzzeiten (rechts)

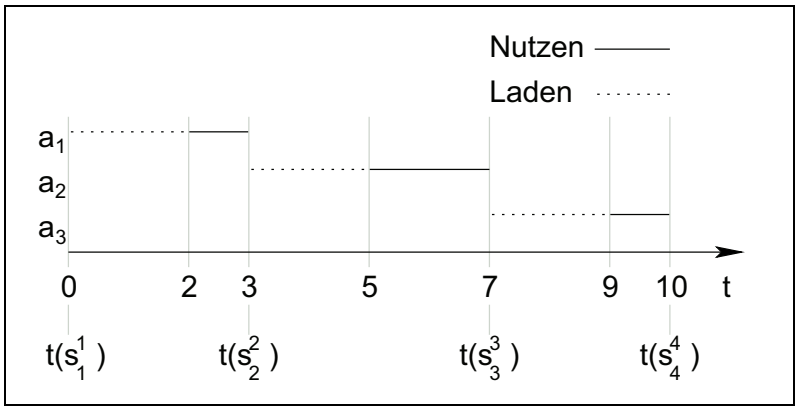



Abbildung 3.7: Zeitliche Abfolge ohne Laden während des Nutzens von Objekten

Abbildung 3.7 illustriert die Lösung eines Verfahrens, welches während des Nutzens eines Objektes keines der zukünftig zu nutzenden Objekte anfragt. Demgegenüber ist in Abbildung 3.8 eine mögliche Abfolge von Laden und Nutzen dargestellt, mit dem Ziel, die Sessiondauer zu minimieren. Das Objekt aus Knoten $a_3$ kann frühestens nach 6 Zeiteinheiten genutzt werden, weil

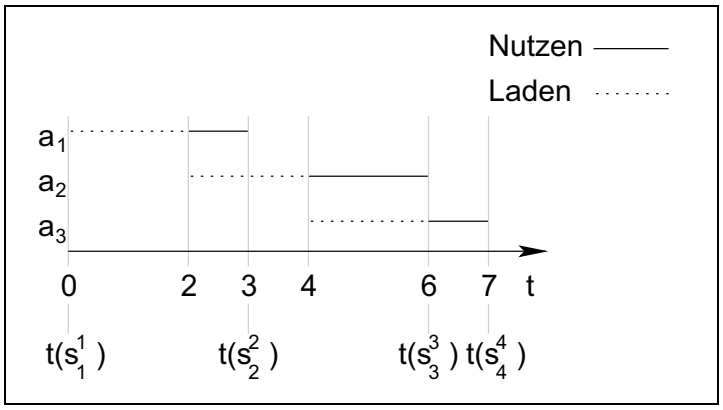



Abbildung 3.8: Zeitliche Abfolge bei höchst möglicher Parallelisierung von Laden und Nutzen der Objekte nach Probleminstanz aus Abb. 3.6

erst dann das Laden aller Objekte der Vorgängerknoten $(a_1, a_2)$ abgeschlossen ist. Alle Objekte der Knoten aus $s_1^3$ werden ununterbrochen geladen, so dass keine Abfolge von Entscheidungen bei kürzerer Sessiondauer existieren kann. Damit ist die kürzest mögliche Sessiondauer dieser Session 7 Zeiteinheiten.

Für die angegebenen Lösungen der Instanz ergeben sich die Entscheidungsmatrizen aus Tabelle 3.1.

**Bemerkung:** *Ist zu einem beliebigen Zeitpunkt einer Session die Restladezeit eines Aufenthaltsknotens größer Null und dessen Objekt wurde bereits bzw. wird zu diesem Zeitpunkt von der lokalen Anwendungskomponente angefragt, dann ist dieses Objekt ab dem Zeitpunkt von der Dauer des Werts der Restladezeit in den lokalen Cache zu laden.*

Das ergibt sich direkt aus der Forderung, dass nicht von lokalen Anwendungskomponenten angefragte Objekte ausschließlich während der Nutzzeit anderer Objekte geladen werden dürfen.[37]

Weiterhin sind die Matrizen $X_i$ so strukturiert, dass alle Entscheidungen der Vorgängermatrix jeweils erhalten bleiben. Eine Matrix $X_i$ bildet so jeweils die Summe aller Entscheidungen aus Vergangenheit und Gegenwart ab. Die Matrizen der einzelnen Zeitpunkte, von *OPT* erzeugt, unterscheiden sich nicht, da zu jedem Zeitpunkt dieselben Informationen zur Verfügung stehen. Jedoch differieren die Restladezeiten der Entscheidungszeitpunkte, weil bis zu jedem Zeitpunkt unterschiedlich viel Nutzzeit zur Verfügung stand.

**Beweis von Satz 3.5:** In den folgenden Ausführungen wird die Summe der Zeiteinheiten aller Nutz- und Ladezeiten passend zu einer Session $s$ mit $\nu(s)$ und $\lambda(s)$ zusammengefasst:

$$\nu(s) = \sum_{\forall a_i \in s} u_i \tag{3.31}$$

$$\lambda(s) = \sum_{\forall a_i \in s} \ell_i \tag{3.32}$$

Auch Bereichsausschnitte von $s$ können so angegeben werden. So beschreibt zum Beispiel $\lambda(s_1^i)$ die Summe aller Ladezeiten der ersten $i$ Aufenthaltsknoten einer Session $s_1^m$. Es sind mindestens

[37] Vgl. Definition der Aufenthaltsknoten, S.62

| *ON* | Entscheidungsmatrizen $X_i, i = 1,2,3$ | | | | | | | | | Restladezeiten in $t(s_i^i)$ | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $x_{11}$ | $x_{12}$ | $x_{13}$ | $x_{21}$ | $x_{22}$ | $x_{23}$ | $x_{31}$ | $x_{32}$ | $x_{33}$ | $r_1$ | $r_2$ | $r_3$ |
| $X_{t(s_1^1)}$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 2 | 2 |
| $X_{t(s_2^2)}$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 2 |
| $X_{t(s_3^3)}$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 |

| *OPT* | Entscheidungsmatrizen $X_i, i = 1,2,3$ | | | | | | | | | Restladezeiten in $t(s_i^i)$ | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $x_{11}$ | $x_{12}$ | $x_{13}$ | $x_{21}$ | $x_{22}$ | $x_{23}$ | $x_{31}$ | $x_{32}$ | $x_{33}$ | $r_1$ | $r_2$ | $r_3$ |
| $X_{t(s_1^1)}$ | 0 | 1 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 2 | 2 | 2 |
| $X_{t(s_2^2)}$ | 0 | 1 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 1 | 2 |
| $X_{t(s_3^3)}$ | 0 | 1 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 |

Tabelle 3.1: Entscheidungsmatrizen des Beispiels aus Abb. 3.7, S.90 für *ON* und *OPT*

$\lambda(s_1^i)$ Zeiteinheiten notwendig, um alle Objekte dieser Knoten zu laden, welche zur Nutzung in den Zeitpunkten $t(s_1^1),\ldots,t(s_i^i)$ angefordert werden. Ein Prefetchingverfahren kann innerhalb dieses Zeitraums, im Rahmen der sich aus der Summe aller Nutzzeiten dieser Knoten ($\nu(s_1^i)$) ergebenden Zeitdauer, Objekte jener Knoten anfordern, welche zukünftig ab Zeitpunkt $t(s_{i+1}^{i+1})$ durch die lokale Anwendungskomponente angefragt werden.

**Bemerkung:** *Die Summe aller Nutz- und Ladezeiten für eine Session $s_1^m$ mit m Knoten ist für einen denkbar schlechten jedoch noch sinnvollen Algorithmus somit eine natürliche obere Schranke ($S_o$).*

$$ON(\hat{I}(s_1^m,B)) = \sum_{\forall a_i \in s_1^m} d_i \leq \lambda(s_1^m) + \nu(s_1^m) = S_o \tag{3.33}$$

Es handelt sich um ein Entscheidungsverfahren, welches Objekte der Aufenthaltsknoten nicht im Voraus anfordert bzw. nur solche Objekte „blind“ lädt, welche zukünftig ungenutzt bleiben.

Algorithmen mit Ergebnissen höher als diese obere Schranke werden als nicht sinnvoll angesehen und nicht weiter betrachtet. Eine ununterbrochene Sequenz von abwechselnden Lade- und Nutzvorgängen einzelner Objekte, in der durch $s$ vorgegebenen Reihenfolge, führt immer zu einer Sessiondauer in Höhe der oberen Schranke, wie im Beispiel in Abb.3.7, S.89 dargestellt.

Ebenso existiert eine untere Schranke $S_u$ für die Sessiondauer. Jeder beliebige Algorithmus

kann auch mit vollständiger Information über zukünftige Anforderungen aller zu nutzender Aufenthaltsknoten nicht besser sein als die Summe aller Lade- bzw. Nutzzeiten einer beliebigen Session:

$$\begin{gathered} S_u = max(\lambda(s_1^m), \nu(s_1^m)) \leq OPT(I(s_1^m, B)) \leq ON(\hat{I}(s_1^m, B)) \\ \forall s_i^i \in s_1^m : \lambda(s_i^i) \geq 0 \; und \; \nu(s_i^i) > 0, \end{gathered} \tag{3.34}$$

wobei $\lambda(s_i^i)$ und $\nu(s_i^i)$ die benötigte Zeit für das vollständige Laden und Nutzen des Objekts des $i$-ten Aufenthaltsknotens in der Session $s_1^m$ beschreiben.

Die untere Schranke lässt sich präzisieren, falls die Ladezeit des Objekts im ersten Aufenthaltsknoten und die Nutzzeit des letzten Aufenthaltsknotens einer Session größer Null sind:

$$S_u = max(\lambda(s_1^m) + \nu(s_m^m), \lambda(s_1^1) + \nu(s_1^m)) \leq OPT(I(s_1^m, B)) \leq ON(\hat{I}(s_1^m, B)) \leq S_o \tag{3.35}$$

Jeder Algorithmus muss vor Beginn der Nutzzeit des Objekts im ersten Knotens dieses geladen haben. Die Dauer des Ladevorgangs ist $\lambda(s_1^1)$. Auch muss jeder Algorithmus mit dem Nutzvorgang des Objekts im letzten Knoten enden. Dessen Dauer beträgt $\nu(s_m^m)$.

Aus den Ungleichungen 3.33, 3.34 und 3.35 sowie 3.30 kann geschlossen werden:

$$ON(\hat{I}(s_1^m, B)) \leq S_o < 2 \cdot OPT(I(s_1^m, B)) \leq 2 \cdot S_u \tag{3.36}$$

Die untere sowie obere Schranke 3.34 und 3.35 schließen alle Lösungen von *OPT* und *ON* ein. Deshalb kann das Verhältnis aus beiden zum Nachweis der Kompetitivität von *ON* gegenüber *OPT* genutzt werden. Es gilt:

$$\frac{S_o}{S_u} = \frac{\lambda(s_1^m) + \nu(s_1^m)}{max(\lambda(s_1^m), \nu(s_1^m))} \leq 2 \tag{3.37}$$

Mit Ungleichung 3.30, S.87 wurde die Eigenschaft kompetitiver und folgend strikt kompetitiver Verfahren definiert. Es ergibt sich für diese Ungleichung nach 3.37 ein $c = 2$ und $\alpha = 0$. Beliebige, aber noch sinnvolle Prefetchingverfahren sind damit strikt 2-kompetitiv. Satz 3.5 kann bestätigt werden. □

Somit kann ein Prefetchingverfahren auch bei Wahl einer denkbar günstigen Reihenfolge der Ladevorgänge die Dauer einer Session gegenüber einem denkbar schlechten Algorithmus nicht mehr als halbieren. Es handelt sich um eine obere Schranke, die sich nicht aus der Wahl eines guten Algorithmus, sondern sich aus der Problemstellung selbst ergibt.

Jedoch wird der Faktor $c = 2$ für das Verhältnis aus oberer und unterer Schranke nur dann erreicht, wenn die Lade- $\lambda(s_1^1)$ und Nutzzeit $\nu(s_m^m)$ Null betragen. Eine solche Session hat folgende Eigenschaften:

$$\begin{gathered} \lambda(s_1^1) = \nu(s_m^m) = 0 \\ \forall i = 1 \ldots m : \sum_{j=1}^{i} \nu(s_j^j) = \sum_{j=1}^{i+1} \lambda(s_j^j) \end{gathered} \tag{3.38}$$

Zu diesen Annahmen sei bemerkt, dass eine Ladezeit von Null in der praktischen Anwendung noch motiviert werden kann, jedoch eine Nutzzeit von Null eines Objekts dem Sachverhalt entspricht, dass dieses nicht genutzt wird und somit auch nicht zu laden ist. Zur Vereinfachung weiterer Ausführungen soll trotzdem angenommen werden, dass die Nutzzeit des letzten Objekts der Session Null betragen kann, so dass das Zweifache des Bestmöglichen durch das Online-Verfahren erreichbar ist. Diese Vereinfachung ist im Rahmen der hier vorgenommenen theoretischen Überlegungen zum Nachweis der Kompetitivität unproblematisch, da dadurch *ON* gegenüber *OPT* schlechter gestellt wird, als praktisch sinnvoll ist.

Es soll damit gelten:

$$\frac{S_o}{S_u} \leq \frac{ON(\hat{I}(s_1^m, B))}{OPT(I(s_1^m, B))} \leq c = 2 \tag{3.39}$$

Die obere Schranke $S_o$ soll weiter präzisiert werden, indem das Verhältnis $\beta$ aus Lade- und Nutzzeiten aller Knoten einer Session betrachtet wird. Es wird gezeigt, dass dieses Verhältnis den Kompetitivitätsfaktor $c$ beeinflusst:

$$\beta = \frac{\lambda(s_1^m)}{\nu(s_1^m)}, \nu(s_1^m) > 0 \tag{3.40}$$

**Satz 3.6** *Die c-Kompetitivität eines denkbar schlechten Prefetchingverfahrens ON ist abhängig vom Faktor* $\beta$. *Für c gilt:*

$$c \leq \begin{cases} \beta + 1 & falls\ 0 \leq \beta \leq 1 \\ 1 + \frac{1}{\beta} & sonst\ \beta \geq 1 \end{cases} \tag{3.41}$$

**Beweis von Satz 3.6:** Es werden die beiden Fälle $0 \leq \beta \leq 1$ und $\beta \geq 1$ betrachtet.

Fall 1: $0 \leq \beta \leq 1$

Aus dieser Bedingung, Ungleichung 3.33 sowie 3.34 ergibt sich zwingend, dass $\lambda(s_1^m) \leq \nu(s_1^m)$ ist und damit die obere Schranke für *ON* wie folgt formuliert werden kann:

$$ON(s_1^m) \leq \beta\nu(s_1^m) + \nu(s_1^m) \tag{3.42}$$

Die Höhe der unteren Schranke gestaltet sich abhängig von $\nu(s_1^m)$ und $\beta$ folgend Ungleichung 3.35 und Gleichung 3.38.

Deshalb gilt für $0 \leq \beta \leq 1$:

$$\frac{S_o}{S_u} = \frac{\beta\ \nu(s_1^m) + \nu(s_1^m)}{\nu(s_1^m)} \leq \beta + 1 \tag{3.43}$$

Damit ist für Fall 1 $c = \beta + 1$.

Fall 2: $\beta \geq 1$

Ist das Verhältnis $\beta$ mindestens 1, dann ist $\nu(s_1^m) \leq \lambda(s_1^m)$ und die Höhe der oberen Schranke für *ON* lässt sich adäquat zu Fall 1 aus 3.34 formulieren.

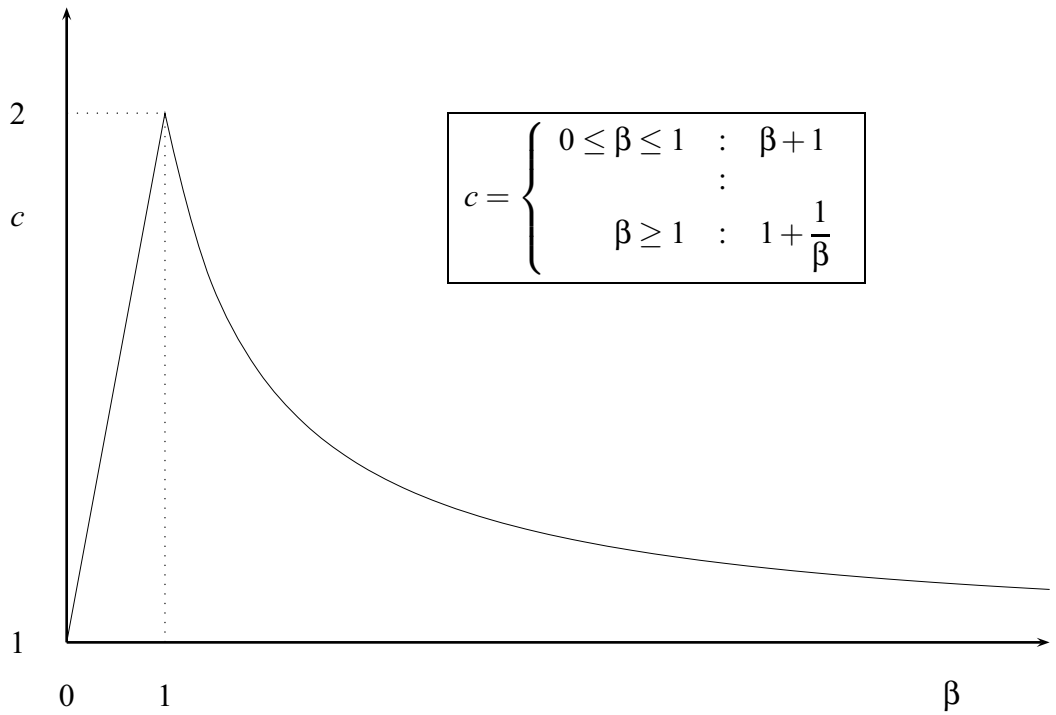



Abbildung 3.9: Obere Schranke des Kompetitivitätsfaktors in Abhängigkeit von β

$$ON(s_1^m) \le \lambda(s_1^m) + \frac{1}{\beta}\lambda(s_1^m) \tag{3.44}$$

Es ergibt sich für den Kompetitivitätsfaktor $c$ folgende Ungleichung:

$$\frac{S_o}{S_u} = \frac{\lambda(s_1^m) + \frac{1}{\beta}\lambda(s_1^m)}{\lambda(s_1^m)} \le 1 + \frac{1}{\beta} \tag{3.45}$$

Damit ist für Fall 2 $c = 1 + \frac{1}{\beta}$.

Die in Fall 1 und 2 bestimmten Ausprägungen für $c$ beschreiben die in Satz 3.6 formulierte Abhängigkeit des Kompetitivitätsfaktors $c$ von β. Der Satz ist zu bestätigen. □

Ein $\beta = 1$ führt zu einem größtmöglichen Kompetitivitätsfaktor zwischen *ON* und *OPT*. Das heißt, für Inputsequenzen mit $\beta = 1$ kann das Einsparungspotential eines guten Prefetchingalgorithmus am größten sein.

Jedoch hängt das Einsparungspotential, das heißt die Differenz aus oberer und unterer Schranke für eine gegebene Sequenz und einen Baum nicht nur von β ab. Die Reihenfolge der Nutzanforderungen ist genauso zu berücksichtigen. Zur Illustration dieses Sachverhalts sind im Anhang, S.199 dazu drei Beispielinstanzen mit $\beta = 1$ und jeweils mögliche Lösungen dafür angegeben.

Je nach Reihenfolge spannen obere und untere Schranke die angegebene maximale Differenz der Lösungswerte für *ON* und *OPT* auf bzw. fallen obere und untere Schranke zusammen.

Probleminstanzen mit der folgenden Struktur können dazu führen, dass das Ergebnis des Online-Verfahrens das 2-fache des optimalen Offline-Verfahrens erreicht:

$$\begin{gathered}\forall i, 1 \leq i \leq m-1 : \nu(s_1^i)) \geq \lambda(s_2^{i+1}) \ \textit{und} \\ \lambda(s_1^1) = \nu(s_m^m) = 0 \ \textit{und} \\ \nu(s_1^m) = \lambda(s_1^m)\end{gathered} \tag{3.46}$$

Mit der Session wird eine Reihenfolge vorgegeben, die es ermöglicht, ein Objekt während der Aufenthaltsdauer vorheriger Knoten vollständig zu laden, bevor sie genutzt werden.

Im Anhang, S.199ff. werden dazu drei Beispiele dargestellt. Die Beispiele 1 und 2 entsprechen der Struktur nach Bedingung 3.46, wobei aber mit Beispiel 1 folgende spezielle Struktur dargestellt wird:

$$\begin{gathered}\forall i, 1 \leq i \leq n-1 : \nu(s_i^i) = \lambda(s_{i+1}^{i+1}) \ \textit{und} \\ \lambda(s_1^1) = \nu(s_m^m) = 0 \ \textit{und} \\ \nu(s_1^m) = \lambda(s_1^m))\end{gathered} \tag{3.47}$$

Die Nutzzeit des Objekts im $i$-ten Knoten stimmt mit der Ladezeit des $i+1$-ten Knotens überein. *ON* erreicht das 2-fache von *OPT*, wenn angenommen wird, dass Nutzzeit des Objekts im letzten Knoten und Ladezeit des ersten Knotens aus $\sigma$ Null sind.

Dagegen fallen obere und untere Schranke eines Online-Verfahrens genau dann zusammen bzw. *ON* wie *OPT* liefern dasselbe Optimierungsergebnis, wenn die Nutzzeit keines der Objekte einer Session dazu verwendet werden kann, mindestens ein Teil eines anderen Objekts der Session zu laden. Dieses ist genau dann der Fall, wenn gilt:

$$\forall s_i^i \in s_1^m : \begin{cases} \nu(s_1^i) = 0 \\ \quad \textit{oder} \\ \lambda(s_{i+1}^m) = 0 \end{cases} \tag{3.48}$$

So strukturierte Session sind praktisch sinnlos. Im Anhang, S.199ff. ist mit Beispiel 3 eine solche Session angegeben.

Das Ergebnis dieser Analyse erscheint nahezu trivial, denn $\beta = 1$ symbolisiert genau solche Inputsequenzen, in welchen die Summe der Nutzzeiten zum Laden der Objekte aller Aufenthaltsknoten der Sequenz verwendet werden kann, wenn die Reihenfolge der Aufenthaltsknoten dies erlaubt.

Ein Spezialfall ist, dass die Nutz- und Ladezeiten nahezu gleich sind. Diese Eigenschaft einer Inputsequenz lässt eine größtmögliche Reduktion der Sessiondauer durch Anwendung eines Prefetchingverfahrens unabhängig von der Reihenfolge der Knoten zu.

Navigationsbäume, deren Nutz- und Ladezeiten der Knoten weitestgehend miteinander übereinstimmen, bergen das größte Potential, um die Sessiondauer mit einem Prefetchingverfahren zu senken. Dieses Potential kann jedoch nur mit einer guten Vorhersage entsprechend dem optimalen Offline-Algorithmus erreicht werden.

**Bemerkung:** *Kann für die Knoten eines Navigationsbaums oder seiner einzelnen Sequenzen ein besonders großer bzw. kleiner Faktor* $\beta$ *ausgemacht werden, dann spielen in der Regel Prefetchingverfahren in Bezug auf die Minimierung der Sessiondauer eine untergeordnete Rolle.*

Mit einem großen $\beta$ übersteigt die Summe der Nutzzeiten die Summe der Ladezeiten weit. Durch Prefetchingverfahren sollte es möglich sein, viele Objekte während der zur Verfügung stehenden Nutzzeit auch zu laden. Jedoch muss damit gerechnet werden, dass in Einzelfällen durch eine ungünstige Reihenfolge der zur Nutzung angeforderten Objekte auch für den Nutzer inakzeptable Wartezeiten entstehen können. Ein kleines $\beta$ beschreibt, dass die Ladezeiten die Nutzzeiten immens übersteigen. Auch mit idealen Prefetchingverfahren ist eine Reduktion der Sessiondauer kaum möglich.

Handelt es sich beim betrachteten Navigationsbaum um einen speziellen Baum, welcher nur Sequenzen enthält, die die Eigenschaft einer Baumsession erfüllen, dann kann die obere Schranke für die Zielstellung *MDS* weiter präzisiert werden.

**Satz 3.7** *Sind alle Sequenzen des Navigationsbaums der Gestalt, dass sie jeweils genau einen Knoten jeder Knotengruppe des Baums enthalten, dann ist jede Lösung eines sinnvollen Online-Algorithmus für MDS 1-kompetitiv und für MDS*$^*$ $\frac{3}{2}$*-kompetitiv.*

Es wird ein Spezialfall von *MDS* bzw. *MDS*$^*$ betrachtet. Die Summe der Ladezeiten aller Knoten einer jeden Sequenz des Baums entspricht genau der Zeitdauer, die notwendig ist, um die Objekte aller Knoten des Baums zu laden. Damit ist *ON* zusätzlich bekannt, welche Objekte der Aufenthaltsknoten zukünftig zu laden sind, jedoch nicht, in welcher Reihenfolge dies im Optimalfall zu geschehen hat. Dieses Wissen führt zu einer Reduktion des Kompetitivitätsfaktors.[38]

In den folgenden Betrachtungen wird angenommen, dass die Session $s_1^m$ eine Baumsession ist. Das heißt, die Knoten des Baums lassen sich in genau $m$ zueinander disjunkte, nicht redundante Knotengruppen zerlegen und in jeder durch den Baum beschriebenen Sequenz ist genau ein Knoten jeder Gruppe enthalten, wie zum Beispiel in Abb. 3.10.

**Definition:** Eine *Baumsession* $s_1^m$ eines redundanten wie auch nicht redundanten Navigationsbaums hat folgende Eigenschaft:

$$\begin{gathered} s_1^m = \{a_1, \ldots, a_m\} \\ \textit{und} \\ a_1 \in G_1, \ldots, a_m \in G_m \\ \textit{und} \\ |A| = G_1 \cup \ldots \cup G_m \end{gathered} \tag{3.49}$$

□

[38] Diese Anwendungen können so aufgebaut sein, dass ein Teil der Menge aller Objekte vollständig durch Benutzer bearbeitet werden muss, um eine bestimmte Grundfunktionalität zu gewährleisten. Wird ein Objekt eines Knotens angefordert, kann so davon ausgegangen werden, dass auch eine bestimmte Menge von Aufenthaltsknoten durchlaufen wird. Wird nur die Teilsequenz bzw. Teilmenge betrachtet, dann kann der Faktor $c$ um $\frac{1}{2}$ gesenkt werden.

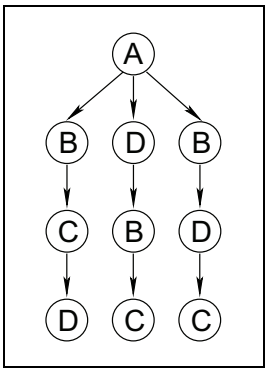



Abbildung 3.10: Ein redundanter Navigationsbaum mit drei Baumsession

Ist das Objekt genau eines beliebigen Knotens einer jeden Knotengruppe geladen worden, z.B. alle Objekte der Knoten aus $s_1^m$, dann beträgt die Restladezeit für alle Knoten des Baums Null.

**Beweis von Satz 3.7:** Der Nachweis der 1-Kompetitivität eines beliebigen sinnvollen Online-Algorithmus für *MDS* unter der getroffenen Annahme ist trivial. Enthält der Navigationsbaum keine redundanten Knotenpaare und existiert eine Sequenz, welche eine Baumsession ist, wird vom Navigationsbaum genau eine ganz bestimmte Abfolge von Aufenthaltsknoten beschrieben, welche dann auch zu jedem Entscheidungszeitpunkt *ON* bekannt ist.

Sinnvoll kann nur ein Verfahren sein, welches jeweils zum Zeitpunkt $t(s_i^i)$ entscheidet, Objekte bzw. Objektteile genau der Knoten in $s_{i+1}^m$ während der Nutzung von $s_i^i$ zu laden, deren Restladezeit größer Null ist und die in der Folge am nächsten zu $s_i^i$ sind.

Um den Nachweis für $MDS^*$ zu führen, wird die Knotenabfolge einer Baumsession in zwei Teile durch den Zeitpunkt $t(s_1^1) \leq t^* \leq t(s_m^m)$ zerlegt. Im Zeitpunkt $t^*$ gilt, dass die bis zu diesem Zeitpunkt angefallene Nutzzeit $N^*$ genau der Summe der Ladezeiten aller ab diesem Zeitpunkt noch zu nutzenden Objekte bzw. Objektteile $L^*$ entspricht. Weiterhin wird davon ausgegangen, dass ein zur Lösung von *MDS* sinnvolles Prefetchingverfahren während der gesamten Nutzzeit $N^*$ ununterbrochen Objekte der Aufenthaltsknoten aus $s_1^m$ lädt.

Im ungünstigsten Fall, unter der Zielstellung Minimierung der Sessiondauer, sind dies genau die Objekte bzw. Objektteile, welche ab dem Zeitpunkt $t^*$ genutzt werden, so dass die Summe der anfallenden Restladezeiten vor dem Zeitpunkt $t^*$ maximal ist und nach $t^*$ Null beträgt, so dass ab $t^*$ keine Nutzzeit mehr für das Laden verwendet wird.

Eine solche Zerlegung soll deshalb für genau diesen Fall betrachtet werden.

Im Zeitraum $(t^*, t(s_{m+1}^{m+1}))$ sind die Objekte aller Aufenthaltsknoten aus $s_1^m$ beim lokalen Cache verfügbar und können ab $t^*$ unterbrechungsfrei genutzt werden.[39] Demgegenüber werden im Zeitraum $(t(s_1^1), t^*)$ ununterbrochen alle Objekte aus $s_1^m$ geladen und einige auch genutzt.

Die Funktionen $\nu$ und $\lambda$ sollen die entsprechende Summe aus anfallenden Nutz- und Ladezeiten für die Zeitabschnitte liefern.

[39] Zur Erinnerung: Laut Definition der Sessioneigenschaften (S.69ff.) ist $t(s_{m+1}^{m+1})$ genau der Zeitpunkt, in welchem die Nutzung des Objekts des letzten Aufenthaltsknotens in $s$ beendet wird.

Es gelten folgende Eigenschaften der zu betrachtenden Baumsession:

$$\begin{aligned} \lambda(t^*, t(s_{m+1}^{m+1})) &= 0 \\ \lambda(s_1^m) &= \lambda(t(s_1^1, t^*)) \end{aligned} \tag{3.50}$$

Spätestens mit der Anfrage des $i+1$-ten Objekts durch die lokale Anwendungskomponente sind alle Objekte der Knoten aus $s_1^i$ und $s_{i+1}^m$ im Speicher des lokalen Caches verfügbar. Dieser Zeitpunkt $t(s_{i+1}^m)$ ist für die folgenden Betrachtungen von besonderem Interesse.

Damit ergeben sich für $ON(\hat{I}(s,B))$ folgende identische obere Schranken:

$$\begin{aligned} ON(\hat{I}(s_1^m, B)) &\leq \lambda(t(s_1^1), t^*) + \nu(t(s_1^1), t^*) + \nu(t^*, t(s_{m+1}^{m+1})) \\ &\leq \lambda(s_1^m) + \nu(t^*, t(s_{m+1}^{m+1})) \end{aligned} \tag{3.51}$$

Betrachtet werden nun im Weiteren nur solche Session $s$, für welche ein größtmöglicher Kompetitivitätsfaktor zwischen $ON$ und $OPT$ zu erwarten ist. Das heißt, dass $\beta = 1$ ist (siehe Satz 3.6). Die benötigte Summe der Ladezeiten aller Aufenthaltsknoten ist identisch mit der Summe aller Nutzzeiten ihrer Objekte.

$$\lambda(s_1^m) = \nu(s_1^m) \tag{3.52}$$

Zusätzlich gelten die beiden folgenden Gleichungen, wobei $\Delta_\nu$ die Summe der Nutzzeiten ist, welche nach $t^*$ anfällt und $\Delta_\lambda$ die Summe von Ladezeiten genau der Objekte, welche bis zu $t^*$ auch genutzt werden:

$$\begin{aligned} \Delta_\nu &= \nu(s_1^m) - \nu(t(s_1^1), t^*) \\ \nu(t(s_1^1), t^*) &= \lambda(s_1^m) - \Delta_\lambda \end{aligned} \tag{3.53}$$

Aus 3.52 und 3.53 folgt:

$$\nu(t(s_1^1), t^*) = \Delta_\lambda = \Delta_\nu = \nu(t^*, t(s_{m+1}^{m+1})) \tag{3.54}$$

Damit ergibt sich für $ON$ aus 3.51 folgende obere Schranke,

$$ON(\hat{I}(s,B)) \leq 3 \cdot \nu(t(s_1^1), t^*) = 3 \cdot \Delta_\lambda = 3 \cdot \Delta_\nu = 3 \cdot \nu(t^*, t(s_{m+1}^{m+1})) \tag{3.55}$$

aus welcher der Kompetitivitätsfaktor abgeleitet werden kann, weil für $OPT$ weiterhin dieselben unteren Schranken $\lambda(s_1^m))$ und $\nu(s_1^m)$ bestehen:

$$c \leq = \frac{\nu(t(s_1^1), t^*) + 2 \cdot \nu(t^*, t(s_{m+1}^{m+1})}{\nu(s_1^m)} = \frac{\nu(t^*, t(s_{m+1}^{m+1}) + 2 \cdot \nu(t(s_1^1), t^*)}{\nu(s_1^m)} = \frac{3}{2} \tag{3.56}$$

Es kann die strikte $\frac{3}{2}$-Kompetitivität und damit Satz 3.7 bestätigt werden:

$$\frac{ON(\hat{I}(s_1^m,B))}{OPT(I(s_1^m,B))} \leq c = \frac{3}{2} \tag{3.57}$$

□

Ein entsprechend konstruiertes Beispiel, in welchem obere sowie untere Schranke durch Entscheidungen eines Online- und optimalen Offline-Verfahrens erreicht werden, ist im Anhang in Abschnitt A.2.2, S.201 dargestellt.

Der Zusammenhang zeigt: Ist die Menge der Objekte aller zukünftigen Aufenthaltsknoten bekannt, dann steigt, unabhängig von einer guten Vorhersage zukünftiger Anfragen, das Anwendungspotential, gemessen als Kompetitivität eines Prefetchingalgorithmus, drastisch gegenüber einem optimalen Offline-Algorithmus an. Der Prefetchingalgorithmus verwendet den zeitlichen Umfang aller Nutzvorgänge soweit wie möglich zum sinnvollen Laden im Voraus, so dass mit diesem Wissen die obere Schranke für die Sessiondauer vom 2-fachen auf das $\frac{3}{2}$-fache des Offline-Algorithmus fällt.

**Bemerkung :** *Im Gegensatz zu MDS bzw. MDS$^*$ ist nicht jeder beliebige, aber noch sinnvolle Online-Algorithmen zur Lösung der Optimierungsprobleme mit den Zielen* $2MDS$ *und* $2MDS^*$ *c-kompetitiv.*

Die Nutz- und Restladezeiten einzelner Knoten einer Session können sich im Gegensatz zu *MDS* und *MDS*$^*$ völlig unabhängig voneinander entwickeln. Das ist an einem einfachen Beispiel erkennbar. Es ist eine Sequenz $s_1^m$ gegeben. Die Nutzzeiten und Ladezeiten aller Knoten außer der des letzten Knotens sind 1. Wird nun angenommen, dass die Ladezeit $\ell_m$ des letzten Knotens $m-1$ beträgt, dann ist eine optimale Lösung vom Wert 1 möglich. Hingegen kann ein Online-Algorithmus, welcher ununterbrochen Objekte der Knoten in der Sequenz während der Aufenthalte im Voraus lädt, eine Lösung vom Wert $m-1$ liefern.

**Bemerkung:** *Für Zielstellungen, welche nur die Restladezeit eines Knotens oder einer Knotenmenge minimieren, kann kein kompetitives Online-Verfahren ohne Wissen über zukünftige Anfragen konstruiert werden.*

Es kann einfach eine Session konstruiert werden, deren Summe der Restladezeiten aller Knoten nur dann Null betragen kann, wenn Wissen über zukünftige Anforderungen existiert. Für eine solchen Session $s_1^m = a_1,\ldots,a_m$ gilt:

$$\begin{gathered} l_1 = 0 \\ \textit{und} \\ \forall(a_i \in s_2^m, i = 2\ldots m) : u_{i-1} = \ell_i \end{gathered} \tag{3.58}$$

Nur mit vollständigem Wissen über den jeweiligen nächsten Aufenthaltsknoten, der zum Zeitpunkt $t(s_i^i)$ angefordert wird, kann ein Verfahren konstruiert werden, welches einen optimalen Wert bezüglich der Restladezeiten von Null liefert. Wird zu irgendeinem Zeitpunkt zwischen $t(s_{i-1}^{i-1})$ und $t(s_i^i)$ nicht das Objekt $o_p \in a_i$ für beliebige $i, 1 < i \leq m$ geladen, dann muss der Zielfunktionswert der Lösung größer Null sein. □

Dieser Umstand resultiert daraus, dass nicht die Nutzzeit in die Zielfunktion eingeht. Die Berücksichtigung der Nutzzeit mit $MDS$ und $2MDS$ führt dazu, dass für den optimalen Offline-Algorithmus immer eine natürliche untere Schranke größer Null existent ist.

Basierend auf diesen Ergebnissen muss die Aussage des vorherigen Kapitels, dass die Güte der Vorhersage zukünftiger Anfragen, initiiert von der lokalen Anwendung, entscheidend die Güte des Prefetchingverfahrens beeinflusst, relativiert werden. Eine gute Vorhersage allein ist nicht ausreichend. Das Leistungspotential wird auch maßgeblich von den Eigenschaften des Navigationsbaums beeinflusst. Damit ist der Wert des Wissens über zukünftige Anfragen und der Einsatz von Verfahren, solches Wissen zu generieren, abhängig von der Reihenfolge und den Eigenschaften der zukünftig gestellten Anfragen. Sind die zukünftig möglichen Anfragen sowie deren mögliche Reihenfolgen bekannt, kann das Leistungspotential eines Prefetchingverfahrens auf Basis dieser Informationen abgeschätzt werden.

## 3.6 Zusammenfassung der Diskussionen von Modellannahmen und der grundlegenden Modelleigenschaften

Mit dem vorgelegten Modell lässt sich besonders gut das Szenario einer Netzanbindung niedriger Bandbreite abbilden. Damit wird das Lernszenario „Lernen von Zuhause aus“ unter den beschriebenen Bedingungen modelliert. Das Modell setzt auf dem sich aus Entwurfsentscheidungen der Sequenzierung und Segmentierung von Lerninhalten ergebenden Navigationsbaum als Modell des Navigationsverhaltens auf. Vorraussetzung für das Treffen von Entscheidungen sind Kenntnisse über Nutz- und Ladezeiten der segmentierten Lerninhalte. Auf Basis dieser Informationen lassen sich die drei beschriebenen Entscheidungsfelder von Lösungsverfahren für die mit den Modellen beschriebenen Optimierungsprobleme unterstützen.

Mittels des Modells kann Prefetching losgelöst von Vorhersage- bzw. Schwellenwertverfahren betrachtet werden. Das Anwendungspotential möglicher Optimierungsverfahren lässt sich in Abhängigkeit von wichtigen Eigenschaften des Navigationsbaums abschätzen.

Die mit dem Offline-Modell beschriebenen Optimierungsprobleme für redundante Navigationsbäume sind $NP$-hart. Damit ist eine praktikable Lösung nur für kleine Probleminstanzen zu erwarten. Für Lösungsverfahren der Optimierungsprobleme des Online-Modells konnte eine problemimmanente obere wie untere Schranke angegeben werden. Aus diesen Schranken kann auf den Wert von Informationen über die Zukunft geschlossen werden und Auswirkung der Unsicherheit auf die Lösungsgüte bewertet werden.

Beide grundlegenden Eigenschaften sind bei der Konstruktion von Lösungsverfahren für das Off- und Online-Modell im folgenden Kapitel zu berücksichtigen. Für das Offline-Modell steht im Vordergrund, effiziente Lösungsverfahren bzw., falls nicht möglich, effiziente Approximierungsverfahren zu den Optimierungsproblemen zu finden. Für das Online-Modell ist zwar auch eine optimale Lösung wünschenswert, erscheint jedoch praktisch unmöglich bzw. nur rein zufällig möglich, so dass hier das Ziel sein muss, solche Verfahren zu konstruieren, die das Anwendungspotential möglichst weit ausschöpfen und die negativen Seiteneffekte so weit wie möglich unterdrücken.

# Kapitel 4

# Offline-Verfahren

*„Events that start in different places and different times*
*all bear down on that one tiny point in space-time,*
*which is the perfect moment.“*
*Terry Pratchet, Thief of Time*

Dieses Kapitel ist in zwei Hauptteile unterteilt: in die Betrachtung optimierender Verfahren für nicht redundante Navigationsbäume und redundante Navigationsbäume. Die Unterteilung resultiert aus der sich jeweils unterscheidenden Komplexität der Optimierungsprobleme.

Zuerst werden für alle Zielstellungen des nicht redundanten Navigationsbaums effiziente Optimierungsverfahren konstruiert. Dann wird die schwere Rechenbarkeit der Optimierungsprobleme für Zielstellungen redundanter Navigationsbäume nachgewiesen und ein Approximationsverfahren, basierend auf den bereits getroffenen Überlegungen zur Modellformulierung, als lineares Programm vorgeschlagen.

## 4.1 Offline-Verfahren nicht redundanter Navigationsbäume

In diesem Abschnitt werden effiziente Lösungsverfahren für die Optimierungsziele 2*MLB*, *MSLB*, 2*MLS*, *MSLB* und 2*MSLS* diskutiert. Alle fünf in den folgenden Abschnitten vorgestellten Optimierungsverfahren basieren auf demselben Lösungsansatz. Zuerst wird ein Basisalgorithmus zur Umsetzung dieses Lösungsansatzes konstruiert und ausführlich Eigenschaften des für die fünf Ziele grundlegenden Lösungsverfahrens betrachtet.

### 4.1.1 Basisverfahren

Alle fünf Optimierungsziele für nicht redundante Navigationsbäume werden nach demselben Prinzip der Zerlegung des Gesamtproblems in genau $n$ Teilprobleme gelöst. Der Navigationsbaum $B$ mit $n$ Knoten wird in genau $n$ unterschiedliche Teilbäume zerlegt. Die Wurzelknoten dieser Teilbäume sind disjunkt zueinander und in einem Teilbaum sind alle die Knoten, welche auf dem Weg von der jeweiligen Wurzel zu allen erreichbaren Blättern des Gesamtbaums enthalten sind. Eine solche Zerlegung ist in Abbildung 4.1 dargestellt.

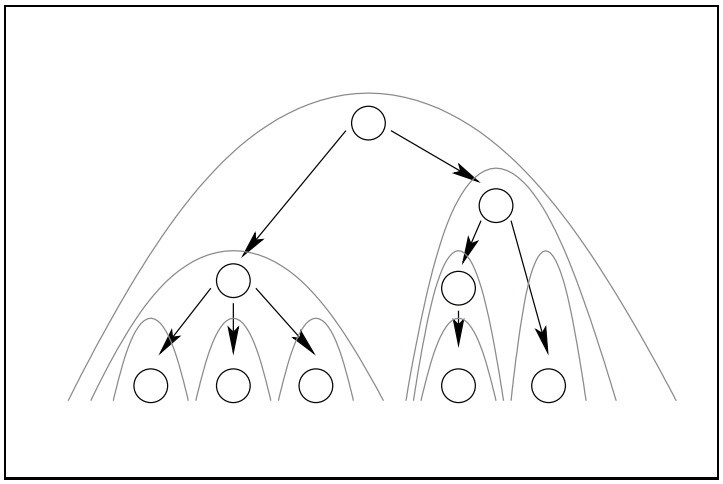


Abbildung 4.1: Ein Navigationsbaum wird in seine Teilbäume zerlegt. Jeder dieser Teilbäume stellt ein zu lösendes Teilproblem dar.

Die einzelnen Teilprobleme sind in einer festgelegten Reihenfolge zu lösen und werden wieder miteinander verschmolzen, so dass eine Lösung des Gesamtproblems entsteht. Es wird das klassische Prinzip des Divide & Conquer[1] umgesetzt. Die unterschiedlichen Optimierungsverfahren sind dadurch gekennzeichnet, wie eine Lösung für die einzelnen Teilprobleme erzeugt wird und wie letztendlich der Optimalwert aus der Gesamtlösung zu bestimmen ist.

Dieses Prinzip wird mit folgendem iterativ, imperativ beschriebenen Basisalgorithmus und den dazugehörigen Datenstrukturen umgesetzt.[2] Die Datenstrukturen sind in Anlehnung an das obige Modell inklusive der Parameterbenennung entworfen worden.

[1] Dieses Prinzip wird für eine Vielzahl von algorithmischen Lösungsverfahren erfolgreich angewendet, z.B. für worst-case-effiziente Sortierverfahren wie merge-sort (vgl. [OW02], S.9ff.).

[2] Die Beschreibung erfolgt in Anlehnung an die Entwurfssprache Anton, verwendet zur studentischen Ausbildung, in [Röc05] beschrieben.

Verwendete Datenstrukturen:

```
AKnoten=Verbund(
    u,l,r:Zeit
    x:Reihe[1..n]:Zeit
    i:Index
    G:Liste(Index) )

Baum=Verbund(
    a:AKnoten
    TeilBäume:Liste(Baum))

PSchlange=PrioSchlange( SElement )

SElement=Verbund(
    i:Index
    anz:Zähler)
```

Die Datenstruktur AKnoten wird zum Speichern der Ladezeit (l), der Restladezeit (r) und der Nutzzeit (u) des Objekts des Aufenthaltsknotens verwendet. Alle Entscheidungen bezüglich der Verwendung der Nutzzeit des Objekts im jeweiligen Knoten zum Laden anderer Objekte werden im Vektor x in Form der Zeitdauer vermerkt, in welcher ein Objekt des jeweiligen Aufenthaltknotens im Voraus zu laden ist. Somit stellt x eine der $n$ Komponenten der Entscheidungsmatrix $X$ dar. Jeder Knoten erhält einen eindeutigen Index i sowie eine Liste (G) der Indizes aller Knoten, welche zur selben Gruppe redundanter Knoten gehören. Diese Liste ist für die hier betrachteten nicht redundanten Navigationsbäume leer.

Die Datenstruktur Baum ermöglicht es, alle Informationen eines Navigationsbaumes entsprechend dem Offline-Modell zu speichern.

Zusätzlich werden Verwaltungsstrukturen zur Problembearbeitung benötigt. Mit der Datenstruktur PSchlange werden Informationen über die Reihenfolge der Abarbeitung einzelner Optimierungsschritte des Verfahrens verwaltet. Jedes Element der Prioritätsschlange besteht aus dem Index eines Knotens und der Anzahl der Knoten auf dem Weg zu diesem, ausgehend vom Wurzelknoten des Baums. Die Prioritätsschlange ist absteigend nach der Komponente anz sortiert. Mit der Handlung nimmAb kann das Element mit dem höchsten Wert von anz der Schlange entnommen werden.

Der Algorithmus OFF stellt die Basis für alle folgenden Lösungsverfahren dar. Er ermittelt je nach Zielstellung eine optimale Entscheidungsmatrix. Die Aktion R(...) ist für jede einzelne Zielstellung zu entwerfen. Mit ihr werden die Restladezeiten der einzelnen Knoten der Teilbäume bestimmt.

```
1  Aktion OFF(<-> B:Baum, <-  Opt:Zeit)
2  S:PSchlange, eS:SElement, T:Baum
3  Beginn
4     InitOFF(B,S)
5     solange nicht leer?(S)
6     Beginn
7        eS = nimmAb(S)
8        T = Teilbaum(B, eS.i)
9        R(T)
10       aktualisiereBaum(B,T)
11    Ende
12    Opt = OptimalWert(B)
13 Ende
```

**Alg. 4.1** OFF(...)

```
1  Aktion InitOFF(<-> B:Baum, <-  S:PSchlange)
2  eS:SElement, T:Baum
3  Beginn
4     S = leer
5     wiederhole für alle Teilbäume T in B Beginn
6        eS.i = T.a.i
7        eS.u = KnotenAufWeg(B, B.a.i, T.a.i)
8        hängeAn(S,eS)
9        T.a.r=T.a.l
10       alle T.a.x[1..n]=0
11       aktualisiereBaum(B,T)
12    Ende
13 Ende
```

**Alg. 4.2** InitOFF(...)

Verwendete Elementarhandlungen der beiden Algorithmen sind in der Tabelle B.1, S.202 im Anhang inklusive des jeweils angenommen Rechenzeit- und Speicherplatzaufwands zusammengefasst.

Das Basisverfahren OFF besteht aus drei Schritten:

**1** Der **Initialisierung** (Zeile 4) der Entscheidungs- (x) und der Hilfsvariablen (r), den Restladezeiten, welche als Komponenten der Knoten des Baums gespeichert werden, sowie die Initialisierung der Prioritätsschlange (siehe Algorithmus 4.2),

**2** der **Berechnung der Entscheidungs- und Hilfsvariablen aller** $n$ **Teilbäume** des Navigationsbaums (Zeilen 5 bis 11), wobei die Handlung R über die Verwendung der Nutzzeit des jeweiligen Objekts eines Knotens zum Laden anderer Objekte des Teilbaums mit Wurzelknoten T.a entscheidet, und

**3** dem **Ermitteln des Optimalwerts** aus der Menge aller getroffenen Entscheidungen, welcher sich aus allen in Schritt 2 berechneten Restladezeiten r direkt ergibt (Zeile 12).

Schritt 2 wird weiter verfeinert durch folgende Schritte:

**2.1** Ermitteln des nächsten zu lösenden Teilproblems (Zeilen 7 und 8),

**2.2** Berechnung der Entscheidungs- und Hilfsvariablen zur Lösung des Teilproblems (Zeile 9)

**2.3** Zusammenführen der Entscheidungen bereits gelöster Teilprobleme, vermerkt in B, mit den in dieser Iteration ermittelten Entscheidungen, vermerkt in T, in das Datenobjekt B.

Mit Hilfe der Prioritätsschlange S wird die Wahl des in einer Iteration als nächstes zu lösenden Teilproblems gesteuert. Es ergibt sich eine Reihenfolge der Abarbeitung der Teilprobleme $P(T_i), i = 1 \ldots n$ zur Lösung des Gesamtproblems $P(B)$, wobei $T_i$ ein Teilbaum des Gesamtbaums $B$ mit Wurzelknoten $a_r$ und $a_i$ der Wurzelknoten des Teilbaums ist.

$P(T_i)$ wird vor $P(T_j)$ genau dann gelöst, wenn gilt:

$$KnotenAufWeg(a_r, a_i, B) > KnotenAufWeg(a_r, a_j, B),$$

wobei die Funktion $KnotenAufWeg(a_x, a_y, B)$ die Zahl der Knoten auf dem Weg im Gesamtbaum $B$ von $a_x$ nach $a_y$ liefert.

Diese Reihenfolge hat wesentlichen Einfluss auf die Korrektheit der Optimierungsverfahren, weil Lösungen eines Teilproblems Lösungen eines anderen Teilproblems beeinflussen können und zwar in unterschiedlicher Weise, je nachdem ob die Knoten der den Teilproblemen zugrundeliegenden Teilbäume zueinander disjunkt oder nicht disjunkt sind.

Mit der Initialisierung der Datenobjekte (siehe Algorithmus 4.2) wird nicht nur die Schlange mit Indizes der Knoten und den Werten des Parameters anz, nach welchem diese sortiert ist, gefüllt. Außerdem werden auch alle Restladezeiten der Knoten auf den Wert der Ladezeit des jeweiligen Knotens und alle Entscheidungsvariablen auf den Wert Null gesetzt. Für den Gesamtbaum mit seiner Entscheidungsmatrix entsteht so eine korrekte, im Regelfall nicht optimale Ausgangslösung, welche genau den Fall darstellt, dass die Nutzzeit keines Objekts eines Knotens dazu verwendet wird, andere Objekte im Voraus zu laden.

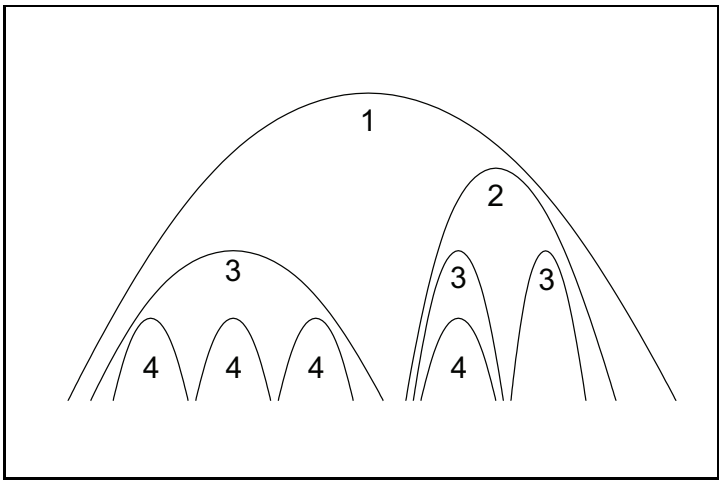



Abbildung 4.2: Die Lösung des Gesamtproblems aus Abb. 4.1, S.102 wird wie abgebildet abgearbeitet. Teilprobleme mit höherer Markierung werden vor Teilproblemen mit niedrigerer Markierung gelöst.

Im Folgenden wird der Rechenzeit- und Speicherplatzaufwand für OFF vorerst unabhängig vom Optimierungsziel, dass heißt ausschließlich für das zielunabhängige Zerlegen in Teilprobleme und Zusammenführen von Teillösungen abgeschätzt. Dazu wird für das Unterprogramm R zunächst Rechenzeit- und Speicherplatzaufwand unabhängig von der Problemgröße angenommen. Danach wird der zielabhängige Aufwand sukzessive je Zielstellung diskutiert und der jeweilige Gesamtaufwand ermittelt.

**Satz 4.1** *Rechenzeit- sowie Speicherplatzaufwand für* OFF *wachsen asymptotisch mit* $O(n^2)$*, wenn diese für* R *mit* $O(1)$ *angenommen werden.*

**Beweis von 4.1:** Der Aufwand für InitOFF beträgt $O(n^2)$ und soll nicht weiter untersucht werden. Die solange-Schleife von OFF wird genau $n$-mal durchlaufen, weil in der Schlange S genau die $n$ Indizes aller Knoten des Baumes B enthalten sind. Kritisch ist die Betrachtung der Handlungen Teilbaum und aktualisiereBaum, weil diese auf den ersten Blick einen Aufwand von $O(n^2)$ verlangen, welches zu einem Gesamtaufwand von $O(n^3)$ führen würde.

Statt T kann R auch B mit Index des Wurzelknotens des zu betrachtenden Teilbaums übergeben werden. Aufgrund der im Folgenden diskutierten Eigenschaften von R beeinflusst dieses nicht die Funktionalität von R, so dass T auch als Zeiger auf einen konkreten Knoten in B implementiert werden könnte. Damit ergäbe sich dann für Teilbaum ein Rechenzeitaufwand von $O(n)$ für die Suche des Knotens mit Index eS.i in B sowie ein Speicherplatzaufwand der gesamten Handlung OFF von $O(n^2)$.

Damit wäre auch eine Aktualisierung des Baums B durch aktualisiereBaum überflüssig, weil T als Alias auf einen Teil von B an R übergeben würde.[3] Daraus ergibt sich für den $n$-mal zu durchlaufenden Schleifenrumpf der solange-Schleife von OFF ein Rechenzeitaufwand von $O(n)$. Satz 4.1 ist zu bestätigen. □

[3]Es wäre eine Implementierung zu wählen, mit welcher nicht eine Kopie sondern ein Zeiger auf den entsprechenden Abschnitt im Hauptspeicher übergeben würde.

Um die Korrektheit der folgenden Entwürfe von R nachzuweisen, ist zu zeigen, dass:

- die durch OFF vorgenommene Zerlegung sinnvoll ist,
- die jeweilige Aktion R eine Teillösung erzeugt, welche mit Hilfe anderer Teillösungen zu einer Gesamtlösung zusammengesetzt werden kann,
- die Reihenfolge der Zusammenführung der Teilprobleme zur optimalen Gesamtlösung führt,
- der Optimalwert aus der Gesamtlösung einfach bestimmt werden kann.

Weiterhin macht das beschriebene Vorgehen nur Sinn, wenn alle Teilschritte effizient rechenbar sind.

### Optimal und R-optimal gelöste Teilprobleme

Zum Nachweis der Korrektheit und zur Prüfung der Effizienz von OFF bzw. des Unterprogramms R (Aufruf Zeile 9 in OFF) in Abhängigkeit vom jeweils verfolgten Optimierungsziel werden zwei Funktionen $R$ und $R^*$ eingeführt. Sie liefern eine Belegung der Entscheidungsmatrix $X$, wobei die Funktion $R$ dasselbe Ergebnis liefern soll, welches auch mit dem Aufruf aus Zeile 9 in OFF berechnet wird.[4] $R$ und $R^*$ werden definiert und grundlegende Möglichkeiten der Konstruktion einer optimalen Lösung diskutiert. Dadurch ist es möglich, die Untersuchungen der für R je Zielstellung noch folgenden Optimierungsverfahren drastisch zu vereinfachen. Zum Nachweis der Korrektheit ist nur noch ein Bezug zur Funktion $R$ und zur Zielstellung herzustellen. Grundsätzlich ist zu klären, unter welchen Eigenschaften von R ein Verfahren OFF optimale Lösungen für die fünf Zielstellungen liefern kann.

**Definition:** $X = R^*(B)$ liefert eine Entscheidungsmatrix $X$ entsprechend dem Offline-Modell für eine der fünf Zielstellungen. Die Funktion $Wert(X)$ liefert den zu einer Entscheidungsmatrix gehörigen Zielfunktionswert, wobei dieser für $Wert(R^*(B))$ der optimale Wert ist.[5] □

**Definition:** Die Funktion $X = R(B)$ liefert eine Entscheidungsmatrix. Es handelt sich dabei um jene, welche durch das jeweilige Verfahren R aus OFF für die jeweilige Probleminstanz $B$ erzeugt wird. □

Die Funktion $R$ ist entsprechend der iterativen Beschreibung von OFF rekursiv definiert. Es gilt folgender Zusammenhang, bei überlagerter Schnittstelle von $R$:

$$\begin{aligned} X = R(B) &= R(u_r, R(T_1), \ldots, R(T_k)) \\ B &= \{a_r, T_1, \ldots, T_k\} \end{aligned} \tag{4.1}$$

Die vom Verfahren R erzeugte Entscheidungsmatrix ergibt sich aus den von R erzeugten Entscheidungsmatrizen für die Teilprobleme, basierend auf allen $k$ Teilbäumen des Gesamtbaums

[4] Das „R“ steht für „Rest“ und „R-optimal“ für „Rest-optimal“. Die Wahl der Begrifflichkeiten wird in den folgenden Absätzen deutlich.

[5] Auf die exakte Parametrisierung der Art $Wert(X, B, Ziel)$ wurde zugunsten einer besseren Lesbarkeit verzichtet. Der gelieferte Zielfunktionswert $Wert(X)$ bzw. $Wert(R^*(X))$ bezieht sich immer auf den Baum und die jeweils betrachtete Zielstellung, aus welchen heraus $X$ bestimmt wurde.

sowie aus der Nutzzeit des Objekts des Wurzelknotens $a_r$ des Baums $B$. Jeder Teilbaum kann wiederum als Gesamtbaum aufgefasst werden, für den mittels R eine Lösung generiert werden kann.

Falls ein Teilbaum $T_i$ ein Blatt ($\Diamond$) ist, dann ist die zugehörige Matrix $X$, welche durch $R$ erzeugt wird, eine Nullmatrix:

$$\forall T_i \in B : \begin{cases} R(T_i) = \begin{matrix} 0 & \dots & 0 \\ \vdots & & \vdots \\ 0 & \dots & 0 \end{matrix}, & falls\ T_i = \Diamond \end{cases} \tag{4.2}$$

Nach dem durch OFF beschriebenen schrittweisen Vorgehen zur Problemlösung werden Teilprobleme $P(T_i)$ erst dann gelöst, wenn entweder $T_i$ selbst ein Blatt ist oder die Teilprobleme des Teilproblems $P(T_i)$ gelöst sind. Die Nutzzeit von Blattknoten kann nicht sinnvoll zum Laden anderer Objekte verwendet werden. Die Entscheidungsmatrix für $R(\Diamond)$ muss eine Nullmatrix sein (siehe Abb. 4.2, S.105).

R in OFF wäre genau dann korrekt, wenn nachgewiesen werden kann, dass gilt:

$$Wert(R(B)) = Wert(R^*(B)) \tag{4.3}$$

Es muss aber nicht gelten:

$$R(B) = R^*(B) \tag{4.4}$$

Für die folgenden Abschnitte soll deshalb zwischen optimalen und R-optimal gelösten Teilproblemen unterschieden und diese miteinander verglichen werden.

**Definition:** Es wird angenommen, dass die Entscheidungsmatrizen $R(T_1),\dots,R(T_k)$ der Teilprobleme $P(T_1),\dots,P(T_k)$, auf Basis welcher die Matrix $R(B)$ mit Verfahren R erzeugt wird, folgende Eigenschaft haben:

$$\begin{matrix} Wert(R(T_1)) & = & Wert(R^*(T_1)) \\ & \vdots & \\ Wert(R(T_k)) & = & Wert(R^*(T_k)) \end{matrix} \tag{4.5}$$

□

Solche Entscheidungsmatrizen sind optimale Entscheidungsmatrizen bzw. eine optimale Lösung für den jeweiligen Teilbaum.

**Definition:** Zur Beschreibung der Konstruktion solch optimaler Entscheidungsmatrizen wird noch eine weitere Funktion $R^*(x,B)$ definiert, welche eine optimale Entscheidungsmatrix unter der Prämisse liefert, dass die Nutzzeit des Objekts $o_p$ der Wurzel $a_r$ von $B$ nicht $u_r$ sondern $x$ ist. □

### Eigenschaften R-optimal gelöster Teilprobleme

Insbesondere aufgrund der Ganzzahligkeit der Entscheidungsvariablen muss nicht zwangsläufig gelten $R(B) = R^*(B)$, obwohl beide Matrizen für den jeweiligen Baum bzw. Teilbaum optimal

sind. Deshalb wird die strengere Eigenschaft der R-Optimalität definiert. Diese ermöglicht es, R-optimale Teillösungen einfach miteinander zu einer optimalen Gesamtlösung zu verschmelzen.

**Definition:** Eine optimale Entscheidungsmatrix $R(B)$ ist genau dann R-optimal, wenn zur verfügbaren Nutzzeit $u_r$ der Wurzel des Baums $B$ entweder nur ein minimaler Wert $w^*$ hinzuaddiert werden muss, so dass der Zielfunktionswert um 1 sinkt, oder $Wert(R^*(B)) = 0$ ist.

Der Wert $w^*$ für eine Entscheidungsmatrix $X$ ist minimal, wenn für ein gegebenes Paar $(w^*, X)$ Bedingung 4.6 erfüllt ist, wobei zwei $n \times n$ große Entscheidungsmatrizen $Y$ und $Y^*$ mit Entscheidungsvariablen $y \in Y$ und $y^* \in Y^*$ und $(X+Y)$ sowie $(X+Y^*)$ Lösungen in Form von Entscheidungsmatrizen für denselben Baum $B$ beschreiben.

Für ein gegebenes Paar $(w^*, X)$ gilt:

$$\nexists(w,Y): \begin{cases} w < w^* \quad und \quad Wert(X) > 0 \quad und \\ Wert(X) - 1 \quad = \quad Wert(X+Y^*) \quad und \\ Wert(X+Y) \quad = \quad Wert(X+Y^*) \quad und \\ \left(\sum_{\forall y \in Y} y\right) = w \quad und \quad \left(\sum_{\forall y^* \in Y^*} y^*\right) = w^* \quad und \\ w \in \mathbb{Z}_0^+ \quad und \quad w^* \in \mathbb{Z}_0^+ \end{cases} \tag{4.6}$$

Das heißt, $w^*$ ist für eine gegebene Matrix $X$ minimal, wenn gilt:

Wird zu Matrix $X$ die Matrix $Y^*$ hinzuaddiert, deren Werte aufsummiert $w^*$ ergeben, so dass der Zielfunktionwert $Wert(X+Y^*)$ um 1 sinkt, und keine andere Matrix $Y$ existieren kann, deren Werte aufsummiert $w$ ergeben, welches kleiner als $w^*$ wäre, sowie $Wert(X+Y)$ auch einen um 1 gesenkten Zielfunktionwert liefern würde, dann ist $w^*$ minimal.

Eine Matrix $X$ ist R-optimal, falls entweder

$$Wert(X) = 0 \tag{4.7}$$

ist oder für das Paar $(w_1^*, X)$ kein weiteres Paar $(w_2^*, X_2)$ des Baums existiert, welches auch 4.6 erfüllt:

$$\nexists(w_2^*, X_2) : w_2^* < w_1^* \; und \; Wert(X) = Wert(X_2) \; und \; X \neq X_2 \tag{4.8}$$

□

Bedingung 4.6 kann umgeformt werden, wenn angenommen wird, dass $X$ von $R(B)$ entschieden wurde ($X = R(B),\ B = \{a_r, T_1 \ldots T_k\}$) und der Wert der Nutzzeit des Objekts der Wurzel $u_r$ ist:

$$\nexists(w,Y): \begin{cases} w < w^* \quad und \quad Wert(X) > 0 \ \ und \\ Wert(R^*(B)) - 1 \quad = \quad Wert(R(u_r + w^*, T_1, \ldots, T_k)) \ \ und \\ Wert(R^*(u_r + w, B)) \quad = \quad Wert(R(u_r + w^*, T_1, \ldots, T_k)) \ \ und \\ \left(\sum_{\forall y \in Y} y\right) = w \quad und \quad \left(\sum_{\forall y^* \in Y^*} y^*\right) = w^* \ \ und \\ w \in \mathbb{Z}_0^+ \quad und \quad w^* \in \mathbb{Z}_0^+ \end{cases} \tag{4.9}$$

Somit ergeben sich für R-optimal gelöste Problemstellungen einige für den weiteren Verlauf der Arbeit interessante Eigenschaften.

**Bemerkung:** *Zu einem Zielfunktionswert kann es mehrere R-optimale Entscheidungsmatrizen geben.*

Aus der obigen Definition R-optimaler Entscheidungsmatrizen folgt nicht zwangsläufig, dass zu jedem optimalen Zielfunktionswert auch genau eine R-optimale Matrix existiert. Unter bestimmten zusätzlichen Umständen ist dies jedoch so.

**Bemerkung:** *Ist $w^* = 1$ aus Bedingung 4.9 für eine R-optimale Entscheidungsmatrix $X$, dann sinkt ein Zielfunktionswert $Wert(X)$ größer Null mit jedem Anstieg der Summe der Entscheidungswerte in der Entscheidungsmatrix $X$.*

Ist die Problemstellung so angelegt, dass sich mit jedem zusätzlichen Entscheidungswert größer Null in $X$ bzw. mit jeder zusätzlich verwendeten Einheit Nutzzeit des Objekts der Wurzel auch der optimale Zielfunktionswert ändert, dann ist jede optimale Entscheidungsmatrix auch zwangsläufig R-optimal. Das ergibt sich direkt aus 4.9.

**Bemerkung:** *Eine R-optimale Entscheidungsmatrix für einen Baum B hat die Eigenschaft, dass für jeden beliebigen Teilbaum $T_i$ des Baums B mit jeweiligem Wurzelknoten $a_i$ mindestens eine der beiden folgenden Bedingungen gilt:*

1) *Die gesamte Nutzzeit $u_i$ der Wurzel des jeweiligen Teilbaums $T_i = \{a_i, T'_1, \ldots, T'_k\}$ wird zum Laden von Objekten aus Knoten $a_j$ der Teilbäume $T'_1 \ldots T'_k$ verwendet.*

$$u_i = \sum_{j=1}^{n} y_{ij}, \quad y_{ij} = \begin{cases} x_{ij}, & falls\ a_j \in \{T'_1, \ldots, T'_k\} \\ 0, & sonst \end{cases} \tag{4.10}$$

2) *Alle Restladezeiten, außer der der Wurzel des Teilbaums, sind Null.*

$$\forall a_j \in \{T'_1, \ldots, T'_k\} : r_j = 0 \tag{4.11}$$

Aus der R-Optimalität einer Entscheidungsmatrix folgt zwingend, dass entweder die Nutzzeit der Wurzel des betreffenden Baumes vollständig zum Laden von Objekten anderer Knoten verwendet wird oder die Restladezeiten aller Knoten des Teilbaums exklusive Wurzel Null betragen müssen. Ist dem nicht so, dann müsste eine Matrix $R^*(B)$ verschieden von der R-optimalen Matrix existieren, welche Bedingung 4.8 widerlegen würde.

Das Verfahren R erzeugt solche R-optimalen Entscheidungsmatrizen entsprechend der jeweiligen Zielfunktion für alle Teilprobleme und vermerkt diese im Datenobjekt B in der durch OFF vorgegebenen Reihenfolge.

Kann die R-Optimalität für jede dieser Lösungen nachgewiesen werden, dann muss auch die Gesamtlösung R-optimal und damit optimal sein.

### Verschmelzen R-optimal gelöster Teilprobleme

Zum Schluss sei darauf hingewiesen, dass die Entscheidungsmatrizen R-optimal gelöster Teilprobleme einfach durch Addition der Matrizen miteinander verschmolzen werden können, weil die Objekte in den Knotenmengen der Teilprobleme in nicht redundanten Bäumen zueinander disjunkt sind.

**Satz 4.2** *Es gilt für R-optimal gelöste Teilprobleme eines Baums $B = \{a_r, T_1 \ldots, T_k\}$, dass durch Addition der Entscheidungsmatrizen $\{R(T_1) \ldots R(T_k)\}$ eine R-optimale Entscheidungsmatrix $R(0, R(T_1), \ldots, R(T_k))$ bei einer Nutzzeit $u_r$ der Wurzel von Null entsteht.*

*Und existiert ein Verfahren R, welches über die Verwendung der Nutzzeit der Wurzel in Höhe von x R-optimal entscheidet, das heißt, die Entscheidungen $R(x,B) - R(0, R(T_1), \ldots, R(T_k))$ sind wiederum R-optimal, dann ist die Lösung $R(x,B)$ R-optimal sowie dann auch optimal.*

*Es gilt:*

$$
\begin{aligned}
Wert(R^*(x,B)) &= Wert(R(x,B)) \\
&= Wert\left( \left( \begin{pmatrix} 0 & \ldots & 0 \\ \vdots & \vdots & \vdots \\ 0 & \ldots & 0 \end{pmatrix} + R(T_1) + \ldots + R^*(T_k) \right) + \right. \\
&\qquad \left. (R(x,B) - R(0, R(T_1), \ldots, R(T_k)) \right)
\end{aligned}
\tag{4.12}
$$

**Beweis von 4.2:** Die Objekte der Knotenmengen in den Teilbäumen $T_1 \ldots T_k$ sind zueinander disjunkt, weil $B$ ein nicht redundanter Navigationsbaum ist.

Damit können Ladeentscheidungen $x_{ij} > 0$ der in den Matrizen enthaltenen Entscheidungsvariablen $R(T_1) \ldots R(T_k)$ voneinander unabhängig getroffen werden.

Es muss für die Entscheidungsvariablen zweier beliebiger, voneinander verschiedener Teilbäume $T_p$ und $T_q$ eines Baums $B$

$$
\begin{gathered}
B = \{a_r, T_1, \ldots, T_k\}, \\
T_p \in \{T_1, \ldots, T_k\}, T_q \in \{T_1, \ldots, T_k\}, \\
p \neq q, p = 1 \ldots k, q = 1 \ldots k
\end{gathered}
\tag{4.13}
$$

gelten:

$$\forall x_{ij} \in \{R(T_1) \ldots R(T_k)\} : x_{ij} = 0 \; und \; x_{ij} \in R(T_q), \;\; falls \; \exists x_{ij} > 0 \; und \; x_{ij} \in R(T_p) \qquad (4.14)$$

unter der Annahme:

$$\forall T_h \in \{T_1 \ldots T_k\} : (\forall j = 1 \ldots n : x_{ij} = 0, \;\; falls \; a_i \notin T_h), \qquad (4.15)$$

welche für jedes gelöste Teilproblem mit der Initialisierung aller Entscheidungsvariablen durch InitOFF mit Null (S.103) sowie der Zuweisung der Nullmatrix den Entscheidungsvariablen eines Baums, welcher ein Blatt ist (Bedingung 4.2, S.107), zugesichert wird.

So kann die Entscheidungsmatrix zweier gelöster, zueinander disjunkter Teilprobleme miteinander addiert werden und es entsteht eine R-optimale Entscheidungsmatrix mit der Lösung beider Teilprobleme.

Da dann $R(0,B) = R(T_1) + \ldots + R(T_k)$ R-optimal ist und laut Annahme ein Verfahren existiert, welches $R(0,B)$ nach $R(x,B)$ so überführt, dass auch $R(x,B)$ R-optimal ist, muss auch die Gesamtlösung $R(x,B)$ R-optimal und damit optimal sein.

Entspricht der Wert von $x$ der Nutzzeit der Wurzel von $B$, dann gilt auch $R(x,B) = R(B)$, womit eine R-optimale Lösung für den Baum $B$ gefunden wäre. □

Aus Satz 4.2 folgt, dass die folgenden Verfahren R für die einzelnen Problemstellungen nur noch auf R-Optimalität bezüglich der Entscheidungen $R(B) - R(0,B)$ zu prüfen sind. Dadurch wird die Beweisführung der Korrektheit der Verfahren maßgeblich vereinfacht.

Die betrachteten Eigenschaften werden nun zur Konstruktion einer R-optimalen Handlung R für OFF je Zielstellungen genutzt.

### 4.1.2 $2MLB$ - Minimierung der maximalen Latenzzeit aller Knoten des Baums

Zur Lösung von $2MLB$ wird für R ein Verfahren vorgeschlagen, welches sukzessive die Restladezeiten der im Teilbaum T enthaltenen Knoten, außer die der Wurzel selbst, absenkt. Das Ergebnis ist allein abhängig vom Wert der Nutzzeit des Objekts der Wurzel sowie den Werten der Restladezeiten aller anderen Knoten des Teilbaums, welche nicht mit den Ladezeiten der Objekte übereinstimmen müssen, sondern sich aus bereits gelösten R-optimalen Teilproblemen ergeben.

Die für R für $2MLB$ zusätzlich benötigten elementaren Handlungen sind in Tabelle B.2 im Anhang, S.203 zusammengefasst.

Das in Algorithmus 4.3, S.112 dargestellte Verfahren arbeitet wie folgt. In jeder Iteration der wiederhole-Schleife $S_1$ (Zeilen 6 bis 19) wird das Maximum der Restladezeit aller Knoten des Baums, exklusive der Wurzel, um genau den Wert Diff subtrahiert. Es werden in einer Iteration die Restladezeiten von Anzahl unterschiedlichen Knoten gesenkt, welche alle zusammen den Maximalwert für die Restladezeit des Teilbaums in dieser Iteration repräsentieren (siehe innere Schleife, Zeilen 13 bis 18). $S_1$ wird genau dann abgebrochen, wenn die Nutzzeit des Objekts der

Wurzel vollständig zum Laden von Objekten der Knoten der Teilbäume verwendet worden ist (Restzeit==0) bzw. die verbleibende ungenutzte Nutzzeit nicht ausreicht, um den Maximalwert weiter zu reduzieren, weil die Restladezeit von mehr Knoten maximal ist, als der Wert der restlichen Nutzzeit beträgt (Diff==0) oder die Restladezeiten aller Knoten, außer die der Wurzel, Null betragen (Diff==0).

Mit Schleife $S_2$ (Zeilen 21 bis 25), aufgerufen wenn die Restladezeit mindestens eines Knotens exklusive Wurzelknoten größer Null ist (Zeile 20), wird die Restladezeit von genau solchen Knoten um 1 reduziert, deren Restladezeit des Teilbaums maximal ist.

Mit jeder Reduktion der Restladezeit wird eine Entscheidung über die Verwendung der Nutzzeit des Objekts der Wurzel getroffen. Diese Entscheidung wird in der Komponente T.o.x[...] vermerkt (siehe Zeilen 15, 23), um die so genannten Ladeentscheidungen als Summe x[i] dieser Entscheidungen zum Laden eines Objekts des Aufenthaltsknotens $a_i$, die zur optimalen Gesamtlösung führen, nachvollziehen zu können.

```
1  Aktion R{für 2MLB}(<-> T:Baum)
2  Restzeit,Diff,SDiff:Zeit,Anzahl:Zähler,j:Index
3  Beginn
4     Restzeit=T.a.u
5     falls Restzeit > 0 und Anzahl(T.Teilbäume) >0
6     wiederhole
7        Anzahl=AnzahlMaximaRestladezeit(T.Teilbäume)
8        SDiff=(MaxRestladezeit(T.Teilbäume) -
9                 Max2Restladezeit(T.Teilbäume)) * Anzahl
10       falls SDiff > 0 dann Beginn
11          falls SDiff > Restzeit dann SDiff=Restzeit
12          Diff = SDiff div Anzahl
13          wiederhole Anzahl mal Beginn
14             j=ReduziereMaxRestladezeit(T, Diff)
15             T.a.x[j]=T.a.x[j]+Diff
16             Restzeit=Restzeit-Diff
17          Ende
18       Ende
19    bis (Diff==0) oder (Restzeit==0)
20    falls MaxRestladezeit(T.Teilbäume)>0
21    wiederhole Restzeit mal Beginn
22       j=ReduziereMaxRestladezeit(T, 1)
23       T.a.x[j]=T.a.x[j]+1
24       Restzeit=Restzeit-1
25    Ende
26 Ende
```

**Alg. 4.3** R(...) *zur Lösung von* 2*MLB*

```
1  Funktion Funktionswert{für 2MLB}
2                ( -> B:Baum):Zeit
3  MaxRestzeit:Zeit, a:AKnoten
4  Beginn
5     MaxRestzeit=B.a.l
6     wiederhole für alle Knoten a in B
7     Beginn
8        falls ( MaxRestzeit < a.r)
9           Maxrestzeit=a.r
10    Ende
11    liefere(MaxRestzeit)
12 Ende
```

**Alg. 4.4** Funktionswert(...)
*zur Lösung von* 2*MLB*

Die obige Funktion liefert den Optimalwert, nachdem alle Teillösungen durch R erzeugt und wieder zusammengeführt wurden. Dieser ergibt sich aus den endgültigen Restladezeiten aller Knoten, wobei die Restladezeit der Wurzel des Gesamtbaums immer deren Ladezeit entsprechen muss. Die Restladezeiten ergeben sich aus den Ladezeiten der Knoten sowie den Entscheidungsvariablen. Diese werden von R für 2*MLB* mit jeder getroffenen Ladeentscheidung aktualisiert (siehe Zeile 14 und 22, links).

Ein Beispiel für die von R erzeugt Lösung ist in Abb. 4.3, S.113 illustriert.

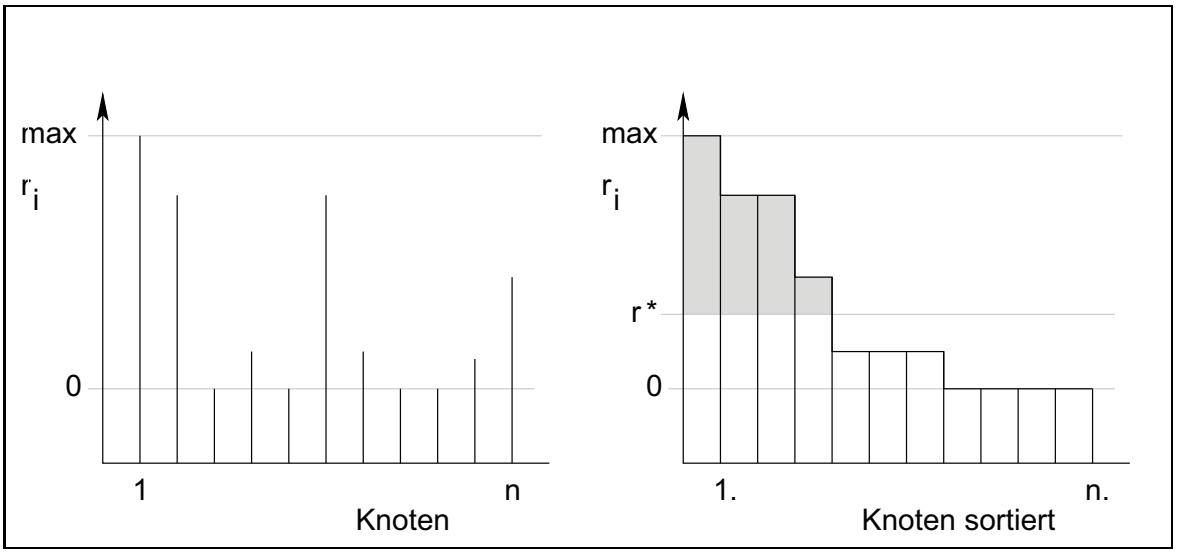



Abbildung 4.3: Funktion der Restladezeit für 2*MLB*, links sind die Restladezeiten der Knoten eines Teilproblems dargestellt, rechts die dazugehörige Treppenfunktion zur Reduktion der maximalen Restladezeit. Um den Zielfunktionswert $r^*$ zu erreichen, muss die Nutzzeit der Wurzel im Umfang der grau schattierten Fläche zum Laden verwendet werden.

**Satz 4.3** *Das mit* R{*für 2MLB*}(...) *und* OFF(...) *beschriebene Verfahren ermittelt für das Problem* 2*MLB eine optimale Lösung sowie die dazugehörige Belegung der Entscheidungsmatrix* $X$.

**Beweis von 4.3:** Die Entscheidungen von R für den Baum $B = \{a_r, T_1, \ldots, T_k\}$ basieren auf Lösungen der Teilprobleme nach den Teilbäumen $T_1 \ldots T_k$. Es wird angenommen, dass diese Teilprobleme R-optimal gelöst sind. Die R-optimale Lösung eines Teilproblems, welche auf einem Blattknoten basiert, muss, wie schon mit Bedingung 4.2, S.107 angemerkt, die Nullmatrix sein (siehe Alg. 4.3, Zeile 5).

Damit muss Gleichung 4.12, S.110 für Verfahren R gelten.

Weiterhin gilt, dass $S_1$ eine R-optimale Lösung

$$R(u_r - w, R(T_1), \ldots, R(T_k)) \tag{4.16}$$

erzeugt, wobei $w$ den Wert des Datenobjekts Restzeit nach Beendigen von Schleife $S_1$ enthält und $u_r$ der Nutzzeit des Objekts des Wurzelknotens im Baum des jeweils zu lösenden Teilproblems $P(T_i)$ entspricht.

Wenn $S_1$ mindestens einmal durchlaufen wurde, gilt:

$$Wert(R^*(u_r - w - 1)) < Wert(R^*(u_r - w)) \tag{4.17}$$

$S_2$ reduziert die Restladezeit genau der Knoten, deren Restladezeit nach Beendigen von $S_1$ auf Grundlage der Matrix $R(u_r - w, R(T_1, \ldots, T_k))$ maximal ist. Das müssen mehr als $w$ Knoten sein, weil sonst nicht 4.17 gelten würde. Die Restladezeit von genau $w$ Knoten mit maximaler Restladezeit wird mit $S_2$ um den Wert 1 gesenkt. Mit dem Absenken maximaler Restladezeiten

einzelner Knoten in $S_2$ um den Wert 1 wird der durch die Matrix repräsentierte Zielfunktionswert nicht weiter reduziert, denn dieser ist bereits vorher optimal, jedoch wird dadurch die R-Optimalität der Matrix gewährleistet.

Wird $S_1$ nicht durchlaufen, dann gilt $w = u_r$. Aufgrund von Bedingung 4.12, S.110 und der Forderung der R-Optimalität der Teillösungen muss die Matrix aus 4.16 R-optimal sein und damit auch $R(B)$.

Wird nicht die innere Schleife von $S_1$ und nicht $S_2$ durchlaufen, dann ist die Nutzzeit der Wurzel Null bzw. die aus $R(T_1) + \ldots + R(T_k)$ resultierenden Restladezeiten müssen Null sein. Mit der Addition der Teillösungen entsteht wiederum eine R-optimale Lösung $R(B)$.[6]

Damit kann bestätigt werden, dass das Verfahren R für $2MLB$ R-optimale Teillösungen und damit auf deren Basis auch eine optimale Lösung für den Gesamtbaum erzeugt. Aus der R-optimalen Entscheidungsmatrix kann einfach der optimale Wert ermittelt werden (siehe Alg. 4.4). □

**Satz 4.4** *Das Verfahren* OFF(...) *unter Verwendung von* R *für* $2MLB$ *ist effizient in* $O(n^3)$ *rechenbar.*

**Beweis von 4.4** Zum Nachweis soll zuerst der Aufwand von R abgeschätzt werden und dann der daraus resultierende Aufwand für OFF.

**Lemma 4.5** *Das Verfahren* R(...) *für* $2MLB$ *ist effizient in* $O(n^2)$ *rechenbar.*

**Beweis von 4.5:** R setzt sich aus zwei Schleifen $S_1$ (Zeilen 6-19) und $S_2$ (Zeilen 21-25) zusammen. $S_1$ wird höchstens $n$-mal durchlaufen, weil mit jedem Schleifendurchlauf die Anzahl der maximalen Restladezeiten, welche sich aus dem aktuellen Zustand der Entscheidungsmatrix nach jeder Iteration von $S_1$ ergibt, um mindestens 1 steigt. Deshalb muss nach dem $n$-ten Durchlauf das Maximum den Wert 0 besitzen und die Schleife wird abgebrochen.[7]

Nach Durchlauf der Schleife $S_1$ kann der Wert von Restzeit höchstens $n-1$ betragen, so dass sich für $S_2$ höchstens $O(n)$ Schleifendurchläufe wie für $S_1$ ergeben.

Zur Bestimmung der Anzahl der Maxima sowie zur Reduktion eines maximalen Wertes um einen konstanten Betrag aus einer Menge von $n$ Werten (reduziereMaximaUm, MaxRestladezeit) wird ein Aufwand von $O(n)$ angenommen, jedoch ist eine effizientere Implementierung mit erhöhtem Verwaltungsaufwand möglich. Daraus würde sich dann eine effizientere Variante für das Gesamtverfahren ergeben.[8]

Der Rechenzeit- wie Speichplatzaufwand für $S_1$ und $S_2$ entwickelt sich asymptotisch mit $O(n^2)$ Damit kann Lemma 4.5 bestätigt werden. □

Weil für R in Satz 4.1, S.105 zur Aufwandabschätzung von OFF ein Aufwand von $O(1)$ anstatt von $O(n^2)$ angenommen wurde sowie die solange-Schleife von OFF $n$-mal durchlaufen wird

[6] Siehe Diskussion spezielle Eigenschaften R-optimaler Entscheidungsmatrizen, S.109ff.

[7] Diff ist dann Null, weil die Funktion AnzahlMaximaRestladezeit den Wert Null liefert und damit Bedingung Zeile 19 erfüllt ist.

[8] Von dieser Variante wird hier Abstand genommen, um die Verständlichkeit des Gesamtverfahrens nicht unnötig zu erschweren. Im Vordergrund steht hier der Nachweis, dass überhaupt eine effiziente und korrekte Lösung des Problems möglich ist.

und InitOFF nur mit einem Aufwand von $O(n^2)$ arbeitet, ergibt sich für das Gesamtverfahren, bestehend aus OFF und R für 2*MLB*, ein asymptotischer Aufwand von $O(n^3)$. Satz 4.4 kann bestätigt werden. □

### 4.1.3 *MSLB* - Minimierung der Summe aller Latenzzeiten des Baums

Für das Problem *MSLB* kann ein effizientes Lösungsverfahren angegeben werden. Es wird gezeigt, dass die Probleme 2*MLB* und *MSLB* einander sehr ähnlich sind. So ähnlich, dass das für 2*MLB* angegebene Verfahren zur Lösung von *MSLB* verwendet werden kann. Vorab jedoch wird ein einfacheres Verfahren angegeben, welches auf den im Folgenden beschriebenen Problemeigenschaften basiert.

**Satz 4.6** *Jede optimale Lösung für MSLB ist auch R-optimal.*

Für jede optimale Lösung sowie auch optimale Teillösung gilt die in der Bemerkung mit den Bedingungen 4.10 und 4.11 auf S.109 beschriebene Eigenschaft für R-optimale Lösungen.

Ist dem nicht so, kann die Nutzzeit mindestens eines Objekts eines Aufenthaltsknotens dazu verwendet werden, einen Teil eines anderen Objekts zu laden, so dass der Zielfunktionswert für die entsprechende Entscheidungsmatrix sinkt, weil für jede beliebige optimale Matrix $R(B)$ mit $w \in \mathbb{Z}_0^+$, $\left( \sum_{\forall x_{ij} \in R(B)} R(B) \right) - w \geq 0$ und der Nutzzeit $u_r$ des Objekts der Wurzel $a_r$ des Baums $B$ gilt:

$$Wert(R^*(B)) + w = Wert(R(u_r - w, R(T_1), \ldots, R(T_k))) \tag{4.18}$$

□

Die Bestimmung einer optimalen Entscheidungsmatrix soll analog zu den Überlegungen für 2*MLB* des vorherigen Abschnitts erfolgen.

```
   KListe = Liste (AKnoten)
1  Aktion R{für MSLB}(<-> T:Baum)
2  a:AKnoten, L:KListe, Restzeit,Diff:Zeit
3  Beginn
4     falls Anzahl(T.Teilbäume)>0 Beginn
5        fülleListe(L,T)
6        Restzeit=T.a.u
7        falls (Restzeit > 0) und (nicht leer?(L))
8        wiederhole
9           a = entnimm(L)
10          Diff = a.r
11          falls Diff > Restzeit dann Diff = Restzeit
12          ReduziereRestladezeit(T, a.i, Diff)
13          Restzeit=Restzeit-Diff
14       bis (leer?(L)) oder (Restzeit==0)
15    Ende
16 Ende
```

**Alg. 4.5** R(...) *zur Lösung von MSLB*

Es wird ein Verfahren R angegeben (links), welches über die Verwendung der Nutzzeit, entsprechend dem obigen Satz und der in 4.18 angegebenen Eigenschaft, entscheidet. Das Verfahren bedient sich einer zusätzlichen Verwaltungsstruktur OListe, welche in den Teilbäumen enthaltene Knoten verwaltet, deren Restladezeiten größer Null sind. Der Algorithmus R berechnet für einen gegebenen Baum auf Basis der optimalen Entscheidungsmatrizen der Teilprobleme eine optimale Entscheidungsmatrix für den Gesamtbaum, so dass die Bedingungen 4.10 und 4.11, Basis zum Nachweis des obigen Satzes 4.6, erfüllt sind. Die verwendeten elementaren Handlungen sind im Anhang in Tabelle B.3, S.204 beschrieben.

Mit jedem Aufruf von Zeile 12 wird der Zielfunktionswert um genau Diff verringert, solange bis das Abbruchkriterium in Zeile 14 der wiederhole-Schleife erfüllt ist. Ist dieses erfüllt, kann der Zielfunktionswert der Matrix nicht weiter gesenkt werden, denn es wird solange die Restladezeit eines Knotens aus T außer die der Wurzel reduziert, bis entweder alle Restladezeiten Null sind (leer?(L)) oder die Nutzzeit des Objekts der Wurzel vollständig zum Laden anderer Objekte im Voraus verwendet wird (Restzeit==0).

```
1  Funktion Funktionswert{fürMSLB}
2                ( -> B:Baum):Zeit
3  S:Zeit, a:AKnoten
4  Beginn
5     S=0
6     wiederhole für alle Knoten a in B
7           S=S+a.r
8     liefere(S)
9  Ende
```

**Alg. 4.6** Funktionswert(...) *für MSLB*

Die Funktion Funktionswert für *MSLB* (links) liefert den zu einer Entscheidungsmatrix gehörigen Zielfunktionswert, indem die aus der Entscheidungsmatrix resultierenden Restladezeiten aller Knoten aufsummiert werden. Asymptotischer Rechenzeit- und Speicherplatzaufwand der Funktion entwickelt sich mit $O(n)$ und $O(n^2)$.

Aufgrund der Gültigkeit von 4.18 muss das angegebene Verfahren R für *MSLB* eine R-optimale Lösung und damit eine optimale Lösung liefern.

**Satz 4.7** *Das mit Alg. 4.5 angegebene Verfahren für* R *unter Verwendung von* OFF *liefert eine optimale Lösung für MSLB.*

Der Nachweis des Satzes stützt sich auf die vorherigen Überlegungen und soll nicht weiter diskutiert werden.

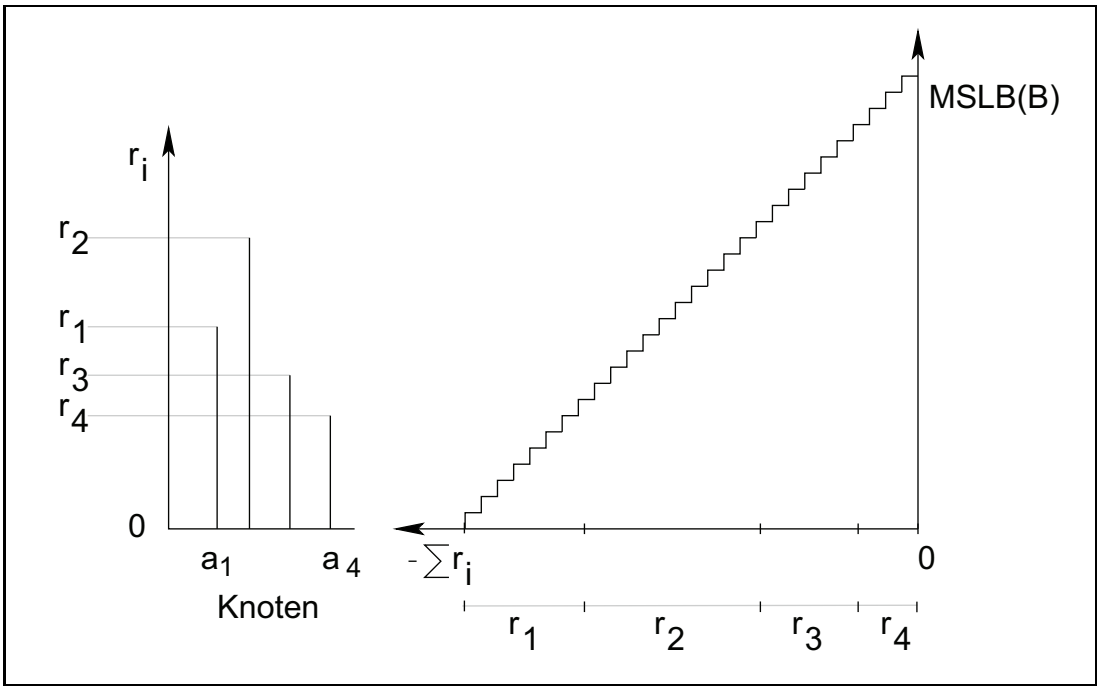



Abbildung 4.4: Links die Verteilung der Restladezeiten von vier Knoten, Rechts die gleichförmige Entwicklung des Zielfunktionswert von *MSLB* je nach verwendeter Nutzzeit der Wurzel, jedem Wert verwendeter Nutzzeit kann genau ein Zielfunktionswert zugeordnet werden

Weiterhin gilt folgender interessanter Zusammenhang zwischen 2*MLB* und *MSLB*.

**Satz 4.8** *Wird zur Lösung von MSLB für* R *der Algorithmus 4.3, S.112* R *für* 2*MLB verwendet, dann liefert* OFF *genauso eine optimale Lösung bezüglich des Problems MSLB wie bei Verwendung von Algorithmus 4.5.*

**Beweis von 4.8:** Kann für jede der durch die Schleifen $S_1$ und $S_2$ des Algorithmus 4.3 generierten Lösung einer jeden Iteration nachgewiesen werden, dass die jeweils verwendete Nutzzeit $w =$ T.a.u - Restzeit des Objekts der Wurzel R-optimal bezüglich der Zielstellung von *MSLB* verwendet wird, dann liefert R für *MSLB* auch eine R-optimale Teillösung.

Da der Wert $w$ mit jeder Iteration von $S_1$ bzw. von $S_2$ um mindestens 1 sinkt und nach dem Verlassen von $S_2$ entweder w den Wert 0 oder die Restladezeiten alle Objekte der Teilbäume Null sind, muss die von R für 2*MLB* erzeugte Lösungsmatrix $R(B)$ für *MSLB* R-optimal sein, wenn auch die Teillösungen $R(T_1) \ldots R(T_k)$ R-optimal sind. □

Jedoch arbeitet Algorithmus 4.5 effizienter als 4.3 für 2*MLB*.[9]

**Satz 4.9** *Das Verfahren* OFF *unter Verwendung von* R *für MSLB (Alg. 4.5) findet mit einem asymptotischen Aufwand von $O(n^2)$ eine optimale Lösung.*

[9]Effizienter nur entsprechend dem hier konkret entworfenen Verfahren. Letztendlich kann, wie bereits angemerkt, 2*MLB* auch mit einem Aufwand von $O(n^2)$ gelöst werden (siehe nächster Satz).

**Beweis von 4.9:** Es ist leicht zu erkennen, dass R für *MSLB* mit einem Speicherplatz- sowie Rechenzeitaufwand von $O(n)$ arbeitet. Die wiederhole-Schleife wird höchstens $n$-mal durchlaufen, denn mit jedem Durchlauf wird genau einer der maximal $n$ durch fülleListe in die Liste L eingefügten Knoten entnommen. Die Schleife wird spätestens dann abgebrochen, wenn die Liste leer ist.

Damit ergibt sich für die wiederhole-Schleife von OFF ein asymptotischer Aufwand von $O(n^2)$, welcher dem Aufwand von InitOFF entspricht. □

### 4.1.4 2*MLS* - Minimierung der maximalen Latenzzeit einer beliebigen Sequenz

Für das Problem 2*MLS* - Minimierung der maximalen Latenzzeit in einer beliebigen Sequenz des Navigationsbaums - kann 2MLB als ein Lösungsverfahren angegeben werden. Es zeigt sich, dass für die Probleme 2*MLB* und 2*MLS* die Mengen optimaler Lösungen für jede beliebige Probleminstanz $B$ identisch sind.

**Satz 4.10** *Das Problem* 2*MLS kann mit Hilfe des oben beschriebenen Verfahrens* 2MLB *effizient gelöst werden.*

**Beweis von 4.10** Jeder Weg von der Wurzel hin zu einem Blattknoten stellt genau eine mögliche Sequenz $\sigma \in B$ dar. Da die Sequenz, welcher der Nutzer während des Navigationsvorgangs erzeugt, unbekannt ist, ist über alle möglichen Sequenzen das Optimum zu bestimmen. Die Menge aller möglichen Sequenzen kann mittels eines Navigationsbaums $B$ dargestellt werden. Für diesen Baum wird mit Hilfe von 2MLB eine optimale Lösung erzeugt. Aus dieser Lösung heraus kann der Wert des Optimums bestimmt werden. Auch können all jene Sequenzen bestimmt werden, die Knoten enthalten, dessen minimale Restladezeit maximal ist und somit das Optimum bilden. Der Nachweis über effiziente Verwendung von Rechenzeit und Speicherplatz von 2MLB wurde bereits geführt. □

### 4.1.5 2*MSLS* - Minimierung der maximalen Summe der Latenzzeiten aller Knoten einer Sequenz über alle Sequenzen

Das Problem 2*MSLS* hat zum Ziel, die maximale Summe aller Restladezeiten der Knoten einer Sequenz über alle möglichen Sequenzen des Baums zu minimieren.

Die Aktion R wird nach der Zielstellung in Anlehnung an R für 2*MLB* wie folgt modifiziert. Die genutzten Elementarhandlungen sind in der Tabelle B.4 im Anhang, S.205 beschrieben.

Da die Benennung der Unterprogrammaufrufe nicht immer selbstsprechend gewählt werden konnte, werden diese im Gegensatz zu den anderen Verfahren nicht nur im Anhang ausführlich mit Schnittstellen, Rechenzeit- und Speicherplatzaufwand dargestellt, sondern zur besseren Verständlichkeit nachfolgend kurz erläutert.

```
1  Aktion R{für 2MSLS}(<-> T:Baum)
2  Restzeit,Diff,SDiff,Max,Max2:Zeit
3  a:AKnoten, aL:Liste(AKnoten),
4  sL:Liste(Liste(AKnoten))
5  Beginn
6     Restzeit=T.a.u
7     wiederhole
8        Max=MaxSumRestladezeit(T)
9        falls Max>0 Beginn
10          sL=SeqMaxSumRestladezeit(T)
11          aL=KnotenJeSeqMinRest(sL)
12          entferneDuplikate(aL)
13          Diff=MinRestladezeit(aL)
14          Max2=MaxSumRestladezeitOhne(aL,T)
15          falls Max-Diff<Max2  Diff=Max-Max2
16          SDiff=Anzahl(aL) * Diff
17          falls SDiff>Restzeit SDiff=Restzeit
18          Diff= SDiff div Anzahl(aL)
19          falls (Diff == 0) und (Restzeit>0) Diff=1
20          solange (Restzeit>0) und (nicht leer?(aL))
21          Beginn
22             a=entnimm(aL)
23             reduziereRestladezeit(T,a.i,Diff)
24             Restzeit=Restzeit-Diff
25          Ende
26       Ende
27    bis (Restzeit==0) oder (Max==0)
28 Ende
```

**Alg. 4.7** R(...) *zur Lösung von* 2*MSLS*

MaxSumRestladezeit(T) liefert den Wert der maximalen Summe aller Restladezeiten je Sequenz über alle Sequenzen des Baums T, das heißt den aktuellen Zielfunktionswert in Abhängigkeit von der Entscheidungsmatrix.

SeqMaxSumRestladezeit(T) liefert in einer Liste alle Sequenzen, die den Maximalwert von MaxSumRestladezeit(T) erreichen. Die Knoten jeder dieser Sequenzen, außer der Wurzelknoten, werden entsprechend ihrer Reihenfolge in T auf dem Weg von der Wurzel zum Zielknoten in einer Liste vermerkt.

KnotenJeSeqMinRest(sL) liefert je Sequenz in sL höchstens einen Knoten. Dieser Knoten ist der Knoten einer Sequenz, welcher von den Knoten, deren Restladzeit größer Null ist, sich auf dem niedrigsten Niveau des Baums T befindet, das heißt dessen Anzahl Vorgängerknoten in der Sequenz am kleinsten ist.

entferneDuplikate(aL) liefert eine Knotenliste, in welcher jeder Knoten aus aL nur genau einmal vorkommt.

MinRestladezeit(aL) liefert den Wert der kleinsten Restladezeit eines Knotens aus aL.

MaxSumRestladezeitOhne(aL,T) liefert den Maximalwert wie MaxSumRestladezeit(T), jedoch nur der Sequenzen in T, welche nicht Knoten aus aL enthalten.

entnimm(aL) liefert ein beliebiges Element aus aL und entfernt dieses aus der Liste.

reduziereRestladezeit(T,a.i,Diff) subtrahiert den Wert Diff von der Restladezeit des Knotens mit Index a.i in T und aktualisiert die Entscheidungsvariable des Wurzelknotens entsprechend (T.a.x[a.i]=T.a.x[a.i]+Diff).

Die Funktionsweise von R wird kurz beschrieben.

Die Aktion R setzt wie schon die bekannten Verfahren auf die Summe der R-optimal gewählten Entscheidungsvariablen aus den gelösten Problemen der Teilbäume auf. Unter Berücksichtigung dieser Restladezeiten soll eine R-optimale Verwendung der Nutzzeit des Objekts der Wurzel bestimmt werden, so dass die maximale Summe der Restladezeiten aller Knoten einer beliebigen Sequenz des Baums minimal ist.

Je Iteration wird der Zielfunktionswert aus den Restladezeiten der Sequenzen ermittelt. Ist dieser oder die Restzeit größer Null, ist die Lösung für die gegebene Nutzzeit des Wurzelknotens nicht R-optimal und eine Iteration der äußeren Schleife ist vollständig zu durchlaufen. Es werden mit jeder Iteration alle maximumbildenden Sequenzen ermittelt, das heißt jene, deren Summe über die Restladezeiten ihrer Knoten maximal ist. Für jede dieser Sequenzen wird unter allen Knoten, deren Restladezeit größer Null ist, genau der Knoten ermittelt, welcher sich am dichtesten zur Wurzel all dieser Knoten der Sequenz befindet. Würde die Restladezeit all dieser so ermittelten Knoten je Sequenz jeweils um 1 gesenkt werden, dann würde auch das Maximum um 1 sinken. Es sind dafür nie mehr Zeiteinheiten Nutzzeit des Wurzelknotens aufzuwenden als die Anzahl dieser Knoten. Warum genau die Knoten am dichtesten zur Wurzel für die Reduktion der Restladezeit zu wählen sind, ist noch zu klären. Nachdem diese Knotenmenge ermittelt wurde, folgen Schritte, welche den Betrag berechnen, um welchen die Restladezeiten dieser Knoten zu reduzieren sind, denn eine Reduktion jeweils nur um den Wert 1 würde zu keinem effizienten Verfahren führen.

Die Restladezeiten der Knoten in der Knotenmenge aL werden um einen der Wert, in den folgenden vier Fällen beschrieben, reduziert. Jede Iteration der äußeren Schleife kann genau einem Fall zugeordnet werden, wobei entweder Fall 1 oder 2 eintreten und in den letzten Iterationen entweder Fall 3 und 4 oder Fall 3 oder Fall 4 oder keiner der beiden.

Der sich aus den im Baum T vermerkten Ladeentscheidungen ergebende Zielfunktionswert wird von Iteration zu Iteration um folgenden Wert gemindert:

1. die Differenz aus dem Maximalwert und dem maximalen Zielfunktionswert der Sequenzen, deren Summe Restladezeiten in dieser Iteration nicht gemindert wird. Existieren solche nicht, dann um den Maximalwert.

2. genau den kleinsten Wert der Restladezeit der Knoten aus aL.

   Die Restladezeiten aller Knoten aus aL werden um diesen Wert reduziert.

3. den Wert, der mittels verfügbarer Restzeit (Nutzzeit des Wurzelknotens) möglich ist, um die Restladezeit all dieser Knoten um denselben Wert zu senken.

4. den Wert 1 bei genau Restzeit vielen Knoten aus aL, falls der Wert von Restzeit kleiner als die Anzahl der Knoten der Liste ist.

Mit der inneren solange-Schleife werden die Restladezeiten der Knoten entsprechend dem Wert einer der vier Fälle gesenkt und die Entscheidungsvariablen aktualisiert.

```
1  Funktion Funktionswert{für2MSLS}
2                ( -> B:Baum):Zeit
3  MaxS,S:Zeit, a:AKnoten, Seq:Liste(AKnoten)
4  Beginn
5    MaxS=0
6    wiederhole für alle Sequenzen Seq in B
7    Beginn
8      S=0
9      wiederhole für alle Knoten a in Seq
10       S=S+a.r
11     falls S>MaxS dann MaxS=S
12   Ende
13   liefere(MaxS)
14 Ende
```

**Alg. 4.8** Funktionswert(...) *für* 2*MSLS*

Der Algorithmus links ermittelt den Zielfunktionswert, resultierend aus den Entscheidungen in B vermerkt. Dazu werden alle in B enthaltenen Sequenzen von der Wurzel bis zu den jeweiligen Blättern betrachtet. Über die Menge aller Knoten einer jeden Sequenz werden die aus der optimalen Entscheidungsmatrix resultierenden Restladezeiten der Knoten summiert. Das Maximum dieser Summen wird als Zielfunktionswert zurückgeliefert. Der asymptotische Rechenzeitaufwand $O(n^2)$ von Funktionswert{*für* 2*MSLS*} ist geringer als die benötigte Rechenzeit zum Erzeugen einer optimalen Entscheidungsmatrix (siehe Satz 4.13, S.124).

**Satz 4.11** *Das angegebene Verfahren zur Lösung von* 2*MSLS findet eine optimale Lösung.*

**Beweis von 4.11:** Zum Nachweis wird wieder angenommen, dass die Teillösungen von $R(B)$: $R(T_1)\ldots R(T_k)$ R-optimal bestimmt wurden und zur Lösung von $R(B)$ verwendet werden.

Es ist zu zeigen, dass die durch Verfahren R generierten Entscheidungen $Y = R(B) - (R(T_1) + \ldots + R(T_k))$ wiederum zu einer $R$-optimalen Matrix $R(B) = Y + (R(T_1) + \ldots + R(T_k))$ führen.

Das konstruierte Lösungsverfahren für 2*MSLS* funktioniert grundsätzlich wie 2*MLS*. Es wird in jeder Iteration der Maximalwert identifiziert und, falls mit der verfügbaren Restzeit möglich, auf den jeweils zweithöchsten Wert oder einen höheren Wert, aber nicht höher als der Zielfunktionswert zu Beginn der Iteration, reduziert.

$\Delta_j$ soll genau der Wert sein, um welchen Restzeit in der $j$-ten Iteration der äußeren Schleife verringert wurde. Das heißt, $\Delta_j$ ist die Differenz aus den beiden Werten von Restzeit vor und nach der $j$-ten Iteration bzw. nach der $j-1$-ten und $j$-ten Iteration. Kann nachgewiesen werden, dass die Matrix nach der $j$-ten Iteration

$$R\left(\left(\sum_{i=0}^{j}\Delta_i\right), R(T_1), \ldots, R(T_k)\right) \tag{4.19}$$

$$\Delta_0 = 0$$

R-optimal ist, wenn die Matrix der $j-1$-ten Iteration R-optimal war, dann muss der Algorithmus R nach beliebig vielen Iterationen für den Baum $B = \{a_r, T_1, \ldots, T_k\}$ R-optimal sein. Dem ist genau dann so, wenn $u_r = \sum_{\forall \Delta_j} \Delta_j$ erreicht ist bzw. die Summe der Restladezeiten entsprechend der erzeugten Entscheidungsmatrix Null ist, weil R eine Lösung unter der Voraussetzung erzeugt, dass $R(0, R(T_1), \ldots, R(T_k))$ als R-optimal gegeben war.

Die Wahl der Knoten in der Liste aL sowie der Wert, um welchen deren Restladezeiten mit jeder Iteration reduziert werden, soll jetzt für jeden der bereits aufgeführten 4 Fälle in Hinblick auf R-Optimalität geprüft werden.

Dazu jedoch vorab folgende Bemerkung, die auch für das nächste Lösungsverfahren von Bedeutung ist.

**Bemerkung:** *Wird die Restladezeit eines Knotens a reduziert, dann wird ebenfalls die Summe der Restladezeiten genau so vieler Sequenzen abgesenkt, wie der Teilbaum mit Wurzel a Blätter besitzt.*

Daraus ergibt sich: Desto kleiner das Niveau von *a* im Gesamtbaum, desto höher die Anzahl der Sequenzen, deren Summe Restladezeiten durch die Reduktion auch gemindert wird.

Daraus ergibt sich wiederum folgender Zusammenhang, welcher in R{*für 2MSLS*} ausgenutzt wird.

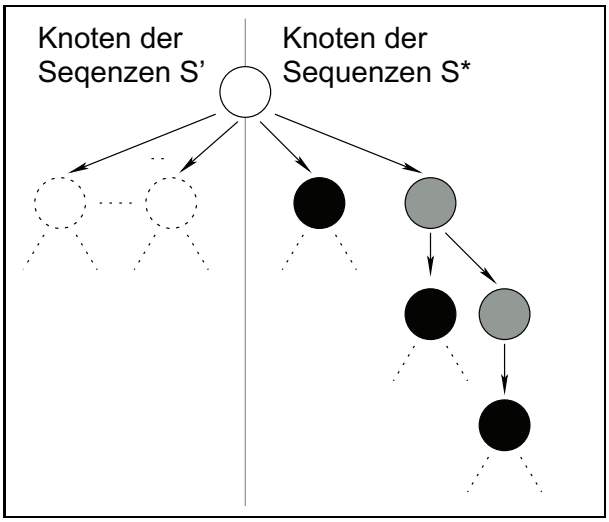
**Lemma 4.12** *Für einen gegebenen Baum B und R-optimal gewählte Entscheidungsmatrix X sei $S^*$ die Menge der Sequenzen, deren Summe Restladezeiten jeweils maximal ist, S die Menge aller anderen Sequenzen des Baums und x die zur Verfügung stehende Nutzzeit, mittels welcher der Maximalwert größer 1 um 1 reduziert werden soll. Es sind dann keinesfalls mehr als $|S^*|$ Zeiteinheiten Nutzzeit notwendig, um die Summe der Restladezeiten in $S^*$ um 1 zu reduzieren. Es sei zusätzlich angenommen, $A^*$ ist eine knotendisjunkte Menge, bestehend aus Knoten, mit jeweils höchstens einem Knoten jeder Sequenz aus $S^*$. Dieser Knoten soll jeweils genau der Knoten sein, welcher exklusive des Wurzelknotens des Baums unter allen Knoten einer Sequenz mit einer Restladezeit größer Null das niedrigste Niveau in B besitzt. (Ist der Zielfunktionswert des Baums exklusive Wurzel größer Null, dann existiert in jeder Sequenz aus $S^*$ genau ein solcher Knoten, der auch in $A^*$ enhalten ist.) Unter diesen Annahmen sind dann genau $|A^*|$ Zeiteinheiten Nutzzeit notwendig, um den Maximalwert des Baums B (ohne Wurzel) bei aus X resultierenden Restladezeiten um 1 zu reduzieren.*

**Beweis von 4.12:** Wird die Restladezeit jedes einzelnen Knotens in $A^*$ um jeweils 1 gesenkt, dann wird auch die Summe der Restladezeiten aller Sequenzen aus $S^*$ um 1 auf einen Zielfunktionswert *z* gesenkt. Entsprechend vorangegangener Bemerkung sind die Knoten in $A^*$ so gewählt, dass die Restzeit möglichst weniger Knoten um 1 gemindert werden muss, um den Zielfunktionswert *z* für alle Sequenzen aus $S^*$ zu erreichen, denn es existiert für jeden Knoten $a^* \in A^*$ kein anderer Knoten dichter zum Wurzelknoten, mittels welchem der Zielfunktionswert der jeweilig $a^*$ enthaltenden Sequenzen um 1 reduziert werden kann.

Weiterhin kann die Summe der Restladezeiten aller Sequenzen $S'$, welche keinen Knoten aus $A^*$ enthalten, höchstens *z* sein. Damit ist der Zielfunktionswert der resultierenden Gesamtmatrix von *B* nach der Reduktion der Restladezeiten *z*. □

In Abb. 4.5, S.123 ist ein Beispiel für eine Zerlegung der Sequenzen in die Mengen $S'$ und $S^*$ dargestellt.

Ein iteratives Vorgehen dieser Art zur Berechnung des Zielfunktionswertes bei Reduktion der Restladezeiten um jeweils 1 unter durch Restzeit gegebener Nutzzeit führt zu einer R-optimalen Matrix, falls auch dann die Restladezeiten genau Restzeit vieler beliebiger Knoten aus $A^*$ um 1 reduziert werden, wenn Restzeit$< |A^*|$.




Im Beispiel links sind die Knoten eines Navigationsbaums in die Sequenzmengen $S'$ und $S^*$ zerlegt. Die Restladezeit aller grauen Knoten ist Null. Die schwarzen Knoten, deren Restladezeit größer Null ist, entsprechen der Knotenmenge $A^*$. Es existiert für jeden Knoten aus $A^*$ mindestens eine Sequenz in $S^*$, die diesen enthält und deren Summe Restladezeiten maximal ist. $S'$ besteht hingegen aus der Menge aller Sequenzen, die keinen der schwarz markierten Knoten enthalten. $S'$ und $S^*$ sind zueinander disjunkt.

Abbildung 4.5: Knotensplitt nach Eigenschaften der Sequenzmengen $S^*$ und $S'$

Genau das wird mit Fall 4, insbesondere mittels Zeile 19, gesichert. Es gilt für Fall 4, wobei $\Delta_i$ dem Wert von Restzeit zu Beginn der i-ten Iteration entspricht:

$$Wert\left(R\left(\left(\sum_{i=0}^{j-1}\Delta_i\right),R(T_1),\ldots,R(T_k)\right)\right) = Wert\left(R\left(\left(\sum_{i=0}^{j}\Delta_i\right),R(T_1),\ldots,R(T_k)\right)\right) \tag{4.20}$$

Für alle anderen drei Fälle gilt:

$$\begin{aligned} &Wert\left(R\left(\left(\sum_{i=0}^{j-1}\Delta_i\right),R(T_1),\ldots,R(T_k)\right)\right) \\ &\qquad = Wert\left(R\left(\left(\sum_{i=0}^{j}\Delta_i\right),R(T_1),\ldots,R(T_k)\right)\right) - (\Delta_j \; div \; x), \end{aligned} \tag{4.21}$$

wobei $x = |A^*|$ der jeweiligen Iteration bzw. der Anzahl Knoten in Datenobjekt aL nach Entfernen der Duplikate in Zeile 12 entspricht.

Da $\Delta_j$ aus Effizienzgründen nicht nur 1 betragen kann, sind die Fälle 1 bis 3 zu untersuchen. Es ist jeweils zu prüfen, ob für jeden Fall, unter Annahme einer Reduktion der Restladezeiten der Knoten im Baum um $\Delta_j$ Zeiteinheiten in der $j$-ten Iteration, nach der $j$-ten Iteration eine R-optimale Matrix entsprechend 4.19 geliefert wird.

**Fall 1:** tritt ein, wenn mit dem Wert von Restzeit mindestens so viele Zeiteinheiten zur Verfügung stehen, dass die Summe der Restladezeiten aller Sequenzen aus $S^*$ auf den Wert der maximalen Summe der Restladezeiten aller Sequenzen $S'$ gesenkt werden kann und das Minimum der Restladezeiten der Knoten aus $A^*$ den Differenzwert der jeweiligen maximalen Summen von Restladezeiten der Sequenzen aus $S^*$ und $S'$ nicht unterschreitet, wobei $S'$ die Menge all jener Sequenzen ist, die nicht Knoten aus $A^*$ enthalten.[10]

[10]Gilt die Bedingung, aber der Differenzwert der maximalen Summen ist kleiner als die minimale Restladezeit aller Knoten aus $A^*$, tritt Fall 2 ein (Zeilen 13–15). Die Bedingung in Zeile 15 ist für Fall 1 erfüllt.

Die Summe der Restladezeiten jeder Sequenz aus $S^*$ wird auf das Maximum der Summe aller Restladezeiten über alle Sequenzen aus $S'$ reduziert, indem die Restladezeiten aller Knoten $A^*$ genau um diesen Differenzwert gesenkt werden. $\Delta_j$ ergibt sich aus dem Differenzwert multipliziert mit $|A^*|$.

Die daraus resultierende Entscheidungsmatrix muss R-optimal sein, weil es laut Lemma 4.12 nicht möglich ist, die Summe der Restladezeiten der Knoten aller Sequenzen $S^*$ um einen kleineren Wert als $\Delta_j$ zu reduzieren, und dabei denselben Maximalwert für den Gesamtbaum gleich dem Maximalwert der Sequenzen aus $S'$ zu erreichen.

**Fall 2:** entspricht nahezu Fall 1, jedoch unterschreitet die Restladezeit mindestens eines Knotens $a_i$ aus $A^*$ die Differenz der Maxima aus $S^*$ und $S'$, so dass das Maximum der Sequenzen aus $S^*$ durch Reduktion der Restladezeiten der Knoten aus $A^*$ nicht auf das Maximum aus $S'$ reduziert werden kann. Es wird entsprechend die Summe der Restladezeiten jeder Sequenz nur um die Restladezeit $r_i$ abgesenkt, indem die Restladezeiten aller Knoten aus $A^*$ um den Wert $r_i$ reduziert werden. Mindestens die Restladezeit von $a_i$ sinkt dabei auf Null. Die daraus resultierende Entscheidungsmatrix muss wiederum aufgrund obigem Lemma 4.12 R-optimal sein, weil der Zielfunktionswert über alle Sequenzen $S^*$ und $S'$ größer ist, als der Wert allein für Sequenzen aus $S'$ betrachtet.

**Fall 3:** tritt genau dann ein, wenn in der $j$-ten Iteration nicht ausreichend Restzeit Zeiteinheiten zur Verfügung stehen, um das Maximum der Sequenzen $S^*$ entsprechend Fall 1 oder 2 zu reduzieren.

Dementsprechend kann die Summe der Restladezeiten je Sequenz aus $S^*$ nur jeweils um Restzeit *div* $|A^*|$ reduziert werden, indem die Restladezeiten der Knoten aus $A^*$ genau um diesen Wert reduziert werden. $\Delta_j = |A^*| \cdot (\text{Restzeit } div\ |A^*|)$. Aufgrund Lemma 4.12 muss die daraus resultierende Entscheidungsmatrix wiederum R-optimal sein.

Nach einer Iteration des Falls 3 ist der optimale Wert für den Baum bestimmt. In der letzten, folgenden Iteration tritt Fall 4 ein, wenn $\Delta_j$ kleiner als der Wert von Restzeit zu Beginn der $j$-ten Iteration war und die Summe der Restladezeiten nach der $j$-ten Iteration größer Null ist. Ein $\Delta_j$ gleich der Restzeit führt dazu, dass die mit dieser Iteration erzeugte Entscheidungsmatrix auch für den Baum und die Nutzzeit des Wurzelobjekts R-optimal ist.

Daraus resultiert, dass das Gesamtverfahren für jede Iteration eine R-optimale Matrix aus 4.19, S.121 liefert, somit auch als Endergebnis eine R-optimale Matrix für die Nutzzeit der Wurzel, $u_r = \sum_{i=0}^{j} \Delta_i$, falls die Summe der Restladezeiten aller Knoten des Baums $u_r$ nicht unterschreitet. Sonst wird eine Nullmatrix als R-optimale Matrix geliefert. Satz **4.11** kann bestätigt werden. □

**Satz 4.13** *Mittels* R *für* 2*MSLS kann mit asymptotischem Rechenzeit- und Speicherplatzaufwand von* $O(n^4)$ *eine optimale Lösung berechnet werden.*

**Beweis von 4.13** Zur asymptotischen Aufwandsabschätzung ist die Anzahl der Durchläufe der äußeren Schleife (Zeilen 7 - 27) von besonderer Bedeutung. Der Rechenzeitaufwand der inneren Schleife (Zeilen 20 - 25) ist je Schleifendurchlauf der äußeren Schleife abhängig von $|A^*| = \text{Anzahl(aL)} \leq n-1$ und beträgt $O(n)$.

Mit jedem Schleifendurchlauf der äußeren Schleife steigt entweder die Anzahl Maxima (Fall 1) oder die Restladezeit mindestens eines Knotens sinkt auf Null (Fall 2) und es treten höchstens einmal Fall 3 und einmal Fall 4 ein. Damit steigt die Anzahl der Schleifendurchläufe in Größenordnung $O(n)$. Für die einzelnen Rechenschritte wird nicht mehr Rechenzeitaufwand als $O(n^2)$ benötigt, so dass der Rechenzeitaufwand für R sich mit $O(n^3)$ entwickelt und daraus folgend für OFF unter Verwendung von R{*für 2MSLS*} in $O(n^4)$ wächst.

□

### 4.1.6 MSLS - Minimierung der Summe aller Latenzzeiten der Knoten einer Sequenz über alle Sequenzen

Als letzte Bewertungsfunktion für nicht redundante Navigationsbäume wird die Summe aller Latenzzeiten über alle Sequenzen untersucht.

Das Verfahren R für 2*MSLS* kann ohne Modifikation nicht zur Optimierung genutzt werden. Mit der Diskussion der beiden baumbasierten Verfahren hatte sich gezeigt, dass die Probleme 2*MLB* und *MSLB* so eng miteinander verknüpft sind, dass das Verfahren für 2*MLB* ohne Modifikation auch zur Lösung von *MSLB* eingesetzt werden kann. Die einander ähnlichen Zielstellungen 2*MLB* und 2*MLS* sowie *MSLB* und *MSLS* lassen denselben Zusammenhang vermuten. Jedoch gilt, dass die Anzahl der zu berücksichtigenden Sequenzen direkt proportional zur Anzahl der Blätter des Baums ist. So genannte Sequenzabbrüche, das heißt der Nutzer verwendet nur einen Teil der Objekte auf dem Weg zwischen Wurzel und Blatt, sollen nach Definition 3.7, S.65 unberücksichtigt bleiben. Daraus ergibt sich folgende Umformung der Zielfunktion 3.20, S.68, welche für das folgende Optimierungsverfahren genutzt werden soll, wobei die Funktion $b(a,B)$ die Anzahl der Blätter des Teilbaums aus $B$ liefert, dessen Wurzel der Knoten $a$ ist:

$$\sum_{\forall \sigma \in B} \sum_{\forall a_i \in \sigma} r_i = \sum_{\forall a_i \in B} (r_i \cdot b(a_i, B)) \tag{4.22}$$

Durch das Verfahren für *MSLS* wird zwar wie gefordert die Summe der Restladezeiten in Bezug auf die zur Verfügung stehende Nutzzeit eines Teilbaums minimiert, jedoch bleibt dieser Zusammenhang unberücksichtigt.

Entsprechend wird ein weiteres Verfahren R für *MSLS* vorgeschlagen.

```
PSchlange = PrioSchlange (SElement)
SElement = Verbund( a:AKnoten
                    AnzahlBlätter:Zähler)
```

Die rechts mit den zusätzlichen Datenobjekttypen vorgeschlagene Aktion R ist auf Basis von R für *MSLB* entstanden. Wesentlicher Unterschied zu *MSLB* ist, dass alle Knoten eines Teilbaums exklusive Wurzel, deren Restladezeit größer Null ist, absteigend nach dem Wert der Funktion $b(a,B)$ sortiert in einer Prioritätswarteschlange verwaltet werden.
Einmalig werden mit dem Aufruf von Aktion fülleSchlange(<-> S:PSchlange, -> B:Baum) alle Knoten des Baums B sortiert in die Prioritätswarteschlange eingefügt. [a]

[a] Der asymptotische Rechenzeitaufwand wächst mit $O(n \log n)$.

```
1  Aktion R{für MSLS}(<-> T:Baum)
2  eS:SElement, S:PSchlange, Restzeit,Diff:Zeit
3  Beginn
4    fülleSchlange(S,T)
5    Restzeit=T.a.u
6    solange (Restzeit > 0) und (nicht leer?(S))
7    wiederhole
8      eS = nimmAb(S)
9      Diff = eS.a.r
10     falls Diff >Restzeit dann Diff = Restzeit
11       ReduziereRestladezeit
12                    (T, eS.a.i, Diff)
13       Restzeit=Restzeit-Diff
14     Ende
15 Ende
```

**Alg. 4.9** R(...) *zur Lösung von MSLS*

**Satz 4.14** *Das angegebene Verfahren zur Lösung von MSLS findet eine optimale Lösung.*

**Beweis von 4.14** Das mit R beschriebene Verfahren berechnet, wie die schon vorher beschriebenen Verfahren, für einen Teilbaum die optimale Verwendung der Nutzzeit des Objekts der Wurzel bei gegebenen R-optimalen Entscheidungsmatrizen der Teilbäume und den daraus resultierenden Restladezeiten.

Mit jedem Durchlauf der wiederhole-Schleife, exklusive dem letzten, wird die Summe der Restladezeiten des Gesamtbaums um genau den Wert der Restladezeit eines Knotens abgesenkt.[11] Jedoch jeder Wert der Nutzzeit $w$, welcher zum Laden eines Objekts eines Knotens $a$ der Teilbäume verwendet wird, senkt im Gegensatz zur Problemstellung *MSLB* hier den Zielfunktionswert um $w \cdot b(a,B)$, so dass jeder Knoten in Abhängigkeit von der Funktion $b$ direkt in eine Rangfolge eingeordnet werden kann, wobei Knoten $a_x$ bis $a_y$ eine sortierte Folge aller Knoten der Teilbäume darstellt:

$$rangfolge(T_1,\ldots,T_k) = a_x,\ldots,a_y, \{a_x,\ldots,a_y\} = \{T_1,\ldots,T_k\}, \{a_x,\ldots,a_y\} \in B \qquad (4.23)$$

Die Funktion *rang* liefert die Position eines Knotens in der Rangfolge. Die Funktion *rangfolge* liefert die sortierte Knotenmenge als Folge.

Es gilt für zwei beliebige, voneinander verschiedene Knoten $a_x$, $a_y$ einer beliebigen Folge in Abhängigkeit von der Struktur des Baums:

$$\begin{array}{ll} rang(a_x,B) < rang(a_y,B), & falls\ b(a_x,B) > b(a_y,B) \\ rang(a_x,B) \geq rang(a_y,B), & sonst \end{array} \qquad (4.24)$$

[11] Im letzten Durchlauf kann es sein, dass weniger Restzeit zur Verfügung steht, als die Restladezeit des Knotens beträgt. Entsprechend wurde nur noch um den Wert von Restzeit zu Beginn des letzten Schleifendurchlaufs reduziert.

Weiterhin gilt für jede Entscheidung über die Verwendung der Nutzzeit $u_r$ des Objekts der Wurzel zum Laden eines Objekts des Knotens $a$ von der Dauer $x$:

$$Wert(u_r - x, R(T_1), \ldots, R(T_k)) = Wert(u_r, R(T_1), \ldots, R(T_k)) - b(a,B) \cdot x,\ x \in \mathbb{Z}_0^+ \qquad (4.25)$$

Somit muss jede optimale Lösung $R^*(B)$ auch R-optimal sein.

Das Verfahren R bestimmt in jeder Iteration, in der noch genau *Rest* Zeiteinheiten der Nutzzeit des Objekts der Wurzel zur Verfügung stehen, um wie viele Einheiten der Zielfunktionswert sinkt, wenn die Restladezeit von $a_i$ um $x$ Einheiten gesenkt wird:

$$\begin{gathered} Wert(Rest - x, R(T_1), \ldots, R(T_k)) - b(a_i,B) \cdot x \\ a_i \in \{T_1 \ldots T_k\},\ x = r_i \\ max \Rightarrow \frac{1}{x} \left(Wert(Rest, R(T_1), \ldots, R(T_k)) - Wert(Rest - x, R(T_1), \ldots, R(T_k))\right) \end{gathered} \qquad (4.26)$$

Ziel ist es, den Zielfunktionswert je Zeiteinheit von $x$ höchstmöglich zu senken. Um das zu erreichen, kann die obige Rangfolge verwendet werden, weil der zu maximierende Wert genau dann bei gegebener, bereits verwendeter Nutzzeit $(Rest - x)$ des Objekts der Wurzel und daraus resultierender Entscheidungsmatrix maximal ist, wenn der Knoten $a_i$ mit größtmöglichem Wert $b(a_i,B)$ und $r_i > 0$ verwendet wird.

Damit muss jede aus einer Iteration resultierende Entscheidungsmatrix im Sinne der Zielstellung von *MSLS* R-optimal sein. □

**Satz 4.15** *Das angegebene Verfahren zur Lösung von MSLS arbeitet effizient mit einem asymptotischen Rechenzeit- und Speicherplatzaufwand von $O(n^3)$.*

**Beweis von 4.15** Dazu muss der Rechenzeitaufwand von R für *MSLS* betrachtet werden.

Das Füllen und das sortierte Einfügen in die Prioritätsschlange benötigt, wie schon angesprochen, $O(n\ log\ n)$ Rechenschritte.[12]

Die wiederhole-Schleife wird höchstens $(n-1)$-mal durchlaufen. Dieses erfolgt genau dann, wenn die Restladezeit aller Knoten der Teilbäume größer Null ist. Jeder Knoten, außer der Wurzel, wird mit fülleSchlange genau einmal in die Schlange eingefügt. Mit jedem Schleifendurchlauf wird genau ein Knoten aus dieser entfernt.

Somit ergibt sich für R ein Aufwand für Rechenzeit- und Speicherplatz von $O(n^2)$, woraus für das Gesamtverfahren OFF eine obere Schranke für asymptotischen Rechenzeit- und Speicherplatzaufwand von $O(n^3)$ folgt. □

Zum Schluss soll der Vollständigkeit halber noch das Verfahren zum Ermitteln des Zielfunktionswerts für *MSLS* angegeben werden.

[12] Sukzessive, rekursiv im Baum aufsteigend kann die Anzahl der Blätter zu jedem Knoten ermittelt werden, so dass ein Knoten nur genau zweimal betrachtet werden muss. Das sortierte Einfügen eines jeden Knotens, dessen Restladezeit größer Null ist, kann in die Schlange worst-case-effizient mit $log\ n$ erfolgen, so dass sich für das Füllen eine Gesamtkomplexität von $O(n\ log\ n)$ ergibt.

```
1  Funktion Funktionswert{fürMSLS}
2                ( -> B:Baum):Zeit
3  S:Zeit, a:AKnoten
4  Beginn
5    S=0
6    wiederhole für alle Knoten a in B
7      S=S+(a.r * Blätter(Teilbaum(B,a.i)))
8    liefere(SRestzeit)
9  Ende
```

**Alg. 4.10** Funktionswert(...) *für MSLS*

Der links dargestellte Algorithmus zur Bestimmung des Zielfunktionswerts für *MSLS* aus einer Entscheidungsmatrix iteriert über alle Knoten inklusive Wurzel eines Baums und summiert über das Produkt der Restladezeit eines jeden Knotens und der Anzahl der Blätter des jeweiligen Teilbaums, dessen Wurzel dieser Knoten ist.

### 4.1.7 Überblick zu den Optimierungsverfahren nicht redundanter Navigationsbäume

In Tabelle B.6 sind die Ergebnisse der fünf vorgestellten Optimierungsverfahren für nicht redundante Navigationsbäume dargestellt. Für jede Problemstellung konnte ein effizientes Verfahren angegeben werden. Der Rechenzeitaufwand wächst asymptotisch nie schneller als $O(n^4)$. Der Speicherplatzaufwand, dessen exakte Betrachtung in einigen Abschnitten vernachlässigt wurde, entwickelt sich in jedem Fall unter der jeweiligen oberen Schranke für den Rechenzeitaufwand, so dass die Gesamtkomplexität der Problemstellung die jeweilige Schranke nicht überschreitet.

Bemerkenswert ist, dass sich das Aufwandswachstum unabhängig von den Problemparametern Lade- und Nutzzeit der Knoten entwickelt.

Ein asymptotisches Wachstum von $O(n^4)$ bei hinreichend großen $n$ sollte in der Praxis, insbesondere bei der Unterstützung echtzeitkritischer Entscheidungen, nicht vernachlässigt werden. Werden die vorgestellten Offline-Verfahren zur Lösung von Teilen der Problemstellungen des Online-Modells in Echtzeit verwendet, ist dies unbedingt zu beachten.

Werden jedoch die Offline-Verfahren zur Unterstützung von Entwurfsentscheidungen eines Hypermediums genutzt, können die ermittelten Aufwände als unkritisch bezüglich der praktischen Entscheidbarkeit der Problemstellungen für nahezu beliebige Probleminstanzen angesehen werden.[13]

Für die Zielstellung Minimierung der Summe aller Latenzzeiten eines Baums ist dem Autor, neben dem hier aufgeführten, auch ein alternatives Verfahren bekannt, welches auf der Lösung eines Maximalflussproblems, formuliert in Anlehnung an ein Transportproblem, basiert.[14] Für die anderen Zielstellungen konnte eine solche Formulierung jedoch nicht gefunden werden. Auch kann durch die Art der Lösung kein effizienteres Verfahren als das bereits beschriebene angegeben werden. Das war auch nicht zu erwarten, weil der Aufwand zur Speicherung der Entscheidungsmatrix dem asymptotischen Rechenzeitaufwand entspricht. Das Lösungsverfahren

[13] Ist dem nicht so, fehlt dem Autor wahrscheinlich die Fantasie, sich praktische Problemstellungen im Entwurf von Hypermedien in solchen Größenordnung von $n$ vorzustellen, dass der Entwurfsprozess nennenswert verzögert würde.

[14] Die Grundidee dabei ist, dass die verwendeten Nutzzeiteinheiten der Wurzelknoten eines jeden Teilbaums zu genau den Knoten zu transportieren, welche diese zum Laden eines Objekts verwenden können.

| Problemstellung | Zielfunktion | Rechenzeit-aufwand | lösbar mit |
|---|---|---|---|
| 2*MLB*: Minimierung der maximalen Latenzzeit aller Objekte des Baums | $min \Rightarrow max_{i=1..n}(r_i)$ | $O(n^3)$ | 2MLB |
| *MSLB*: Minimierung der Summe der Latenzzeit aller Objekte des Baums | $min \Rightarrow \sum_{i=1}^{n} r_i$ | $O(n^2)$ | MSLB, 2MLB |
| 2*MLS*: Minimierung der maximalen Latenzzeit der Objekte der durch den Baum repräsentierten Sequenzen | $min \Rightarrow max_{\forall\sigma\in B}(\sum_{\forall a_i\in\sigma} r_i)$ | $O(n^3)$ | 2MLB |
| 2*MSLS*: Minimierung der maximalen Summe aller Latenzzeiten einer beliebigen durch den Baum repräsentierten Sequenz | $min \Rightarrow max_{\forall\sigma\in B}(\sum_{\forall a_i\in\sigma} r_i)$ | $O(n^4)$ | 2MSLS |
| *MSLS*: Minimierung der Summe aller Latenzzeiten der Objekte aller durch den Baum repräsentierten Sequenzen | $min \Rightarrow \sum_{\forall\sigma\in B}\sum_{\forall a_i\in\sigma} r_i$ | $O(n^3)$ | MSLS |

Tabelle 4.1: Optimierungsverfahren des Offline-Modells im Überblick

für *MSLB* ist spätestens dann worst-case effizient, wenn als Lösung nicht nur der Optimalwert, sondern auch die dazugehörige Entscheidungsmatrix gefordert wird.

Weiterhin ist schon mit der Modellformulierung sichtbar, dass für jede der aufgeführten Zielstellungen ein Großteil möglicher Probleminstanzen mehr als eine optimale Lösung zulassen. Hier ist zusätzliches Optimierungspotential für sekundäre Ziele vorhanden, welche im Modell nicht formuliert sind.

Die vorgestellten Verfahren können nicht für redundante Navigationsbäume genutzt werden. Das hängt mit den speziellen Eigenschaften R-optimaler Entscheidungsmatrizen zusammen, welche sich nur auf nicht redundante Bäume anwenden lassen. [15]

[15] Siehe Abschnitt 4.1.1, S.107ff.

## 4.2 Approximierende Offline-Verfahren für redundante Navigationsbäume

Nachdem für die fünf Optimierungsprobleme nicht redundanter Navigationsbäume effiziente Lösungsverfahren gefunden werden konnten, werden in diesem Abschnitt Lösungsverfahren für die Optimierungsprobleme redundanter Navigationsbäume der $*$-Zielstellungen konstruiert. Bereits in Kapitel 3, Abschnitt 3.5, S. 79ff. zur Diskussion grundlegender Modelleigenschaften des Offline-Modells wurden die Optimierungsprobleme als *NP*-hart eingestuft. Damit können höchstwahrscheinlich keine effizient rechenbaren Lösungsverfahren konstruiert werden, welche eine optimale Lösung liefern. Deshalb wird in diesem Abschnitt ein effizient rechenbares Approximationsverfahren basierend auf der Relaxation der Ganzzahligkeitsbedingung der Entscheidungsvariablen vorgeschlagen und die Güte der approximierten Lösung betrachtet.

Dazu wird zuerst das Online-Modell für redundante als auch nicht redundante Navigationsbäume unter der Bedingung reellwertiger Entscheidungsvariablen als lineares Programm formuliert. Dann wird ein Approximationsverfahren dargestellt, welches sowohl für redundante und nicht redundante Navigationsbäume genutzt werden kann, und die Güte des Approximationsverfahrens bewertet. Zur Bewertung wird die approximierte Lösung mit der optimalen Lösung in Abhängigkeit von den Zielfunktionen verglichen und die Komplexität eines Approximationsproblems, welches eine Lösung nicht schlechter als das Optimum zuzüglich einer beliebig großen additiven Konstante fordert, bestimmt.

### 4.2.1 Relaxation der Ganzzahligkeit

Unter der Prämisse reellwertiger Entscheidungsvariablen wird das Offline-Modell aus Kapitel 3, S. 62ff. als lineares Programm formuliert.[16] Im nächsten Abschnitt kann das hier formulierte lineare Programm, dessen optimale Lösung effizient berechenbar ist (vgl. [CM99], S.31-8), zur Konstruktion eines effizienten Approximationsverfahrens verwendet werden.

**Satz 4.16** *Werden für das Offline-Modell an statt ganzzahliger Entscheidungsvariablen reellwertige Entscheidungsvariablen angenommen, dann kann das Offline-Modell als lineares Programm formuliert werden.*

**Beweis von 4.16:** Zum Nachweis des Satzes sind die Nebenbedingungen des Offline-Modells und die Zielfunktionen in Form eines linearen Programms zu formulieren. Dazu wird die im Kapitel 3, S.65 definierte Entscheidungsmatrix $X$ mit den Entscheidungsvariablen $y_{ik}$ für redundante Navigationsbäume inklusive der definierten Knotengruppen $G_k$ genutzt.[17]

Begonnen wird mit der Formulierung der Nebenbedingungen.

Für jeden der $n$ voneinander verschiedenen Teilbäume $T_i$ des Gesamtbaums $B$ mit jeweils Wurzelknoten $a_i$ werden die Nebenbedingungen 4.27 – 4.30 formuliert[18], wobei Nebenbedingung

[16] Konzept und Eigenschaften linearer Programme sowie Lösungsverfahren dafür werden überblickartig in [CM99] dargestellt.

[17] Nicht redundante Navigationsbäume lassen sich als Spezialfall der redundanten navigationsbäume auffassen und können deshalb auf die selbe Art und Weise formal beschrieben werden.

[18] Diese gelten nicht nur für jeden Teilbaum sondern auch für den Gesamtbaum, welcher laut Definition des Navigationsbaums auch als Teilbaum seiner selbst angesehen wird.

4.28 nur eine Verknüpfung der beiden Bedingungen 4.27 und 4.29 mit einer Hilfsvariable $w_{ij}$ in Abhängigkeit vom Wert der Entscheidungsvariablen $y_{ik}$ und der Gruppenzugehörigkeit eines Knotens $a_j$ darstellt. Sie resultieren aus den die Entscheidungsvariablen betreffenden Bedingungen 3.8 bis 3.12, S.65ff. des Offline-Modells.

$$\forall T_i \in B : u_i \geq \left( \sum_{\forall G_k \in B} \left\{ \begin{array}{ll} y_{ik}, & falls\ |G_{ki}| > 0 \\ 0, & sonst \end{array} \right. \right) \tag{4.27}$$

$$w_{ij} = \left\{ \begin{array}{ll} y_{ik}, & falls\ a_j \in G_{ki} \\ 0, & sonst \end{array} \right. \tag{4.28}$$

$$\ell_i \geq \left( \sum_{\forall a_j \in B} \left\{ \begin{array}{ll} w_{ji}, & falls\ a_j \in Weg(a_r, a_i, B)\ und\ (i \neq j) \\ 0, & sonst \end{array} \right. \right) \tag{4.29}$$

$$\forall G_k \in B, \forall T_i \in B : y_{ik} \geq 0 \tag{4.30}$$

Die Funktion $Weg(x,y,B)$ liefert eine Menge von Knoten inklusive $x$ und $y$, welche sich auf dem Weg des Baums $B$ von $x$ nach $y$ befinden.

Die ersten $n$ Ungleichungen aus 4.27 sichern, dass alle Entscheidungen über die Verwendung der Nutzzeit des jeweiligen Objekts der Wurzel eines Teilbaums diese nicht überschreiten. Weiterhin wird gesichert, dass der Wert der Entscheidung für das Laden aller Objekte einer Teilgruppe $G_{ki}$ höchstens einmal in jede der $n$ Ungleichungen eingeht. Und zwar genau dann einmal, wenn mindestens ein Element der Gruppe $G_k$ auch in der Teilmenge enthalten ist[19].

Die Gleichungen aus 4.28 sichern, dass die Werte der Entscheidungen aller Objekte einer Gruppe in den Teilbäumen des jeweiligen Baums $T_i$ identisch sind.

Alle Ungleichungen aus 4.29 sichern, dass ein Objekt eines Knotens nicht länger geladen wird, als dieses mit seiner Ladezeit spezifiziert wurde.

Die letzten $n$ Ungleichungen aus 4.30 sichern, dass die Entscheidungsvariablen entsprechend der Modelldefinition positiv sind.

4.28 und 4.29 können zusammengefasst werden, indem alle die $w_{ij} > 0$ in 4.29 nach Bedingung 4.28 in 4.29 durch die entsprechenden $y_{ik}$ ersetzt werden.

Damit entstehen $2(n-1)$ Ungleichungen mit jeweils höchstens $n-1$ Variablen $y_{ik}$ sowie bis zu $n^2$ Ungleichungen aus 4.30.

Bis zu $(n-1)$ Variablen werden dann für eine der $2(n-1)$ Ungleichungen benötigt, wenn $n$ unterschiedliche Knotengruppen existieren, das heißt ein nicht redundanter Navigationsbaum vorliegt.[20]

[19] Es gilt dann $G_{ki} \subset G_k\ und || G_{ki}| > 0)$

[20] $n-1$ und nicht $n$, weil der Wurzelknoten $a_i$ eines Teilbaums nicht zur jeweiligen Teilgruppe $G_{ki}$ gehört.

Das Gleichungssystem bildet die mit dem Offline-Modell formulierten Nebenbedingungen 3.8 bis 3.12 ab, wenn ganzzahlige $y_{ik}$ angenommen würden. Es gilt:

$$\forall x_{ij} \in X : x_{ij} = \begin{cases} y_{ik}, & falls\ a_j \in G_{ki} \\ 0 & sonst \end{cases}$$
$$y_{ik} \in \mathbb{Z}_0^+ \tag{4.31}$$

Entspricht zum Beispiel ein nicht redundanter Navigationsbaum dem Sonderfall einer linearen Liste und die Indizes sind entsprechend der Reihenfolge der Objekte in dieser Liste vergeben sowie sind die Indizes der einelementigen Knotengruppen gleich den Indizes der Knoten in diesen wie in Abbildung 4.6 angegeben, dann entsteht folgendes Gleichungssystem nach dem Muster eines linearen Programms:

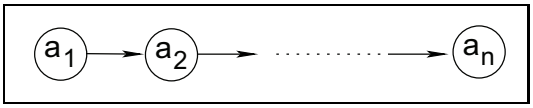



Abbildung 4.6: Navigationsbaum als Liste mit Knoten, indiziert nach Reihenfolge

$$\left(\begin{array}{cccccccccccc}
u_1 & \geq & 0 & + & y_{12} & + & y_{13} & + & y_{14} & + & \dots & + & y_{1n} \\
u_2 & \geq & 0 & + & 0 & + & y_{23} & + & y_{24} & + & \dots & + & y_{2n} \\
u_3 & \geq & 0 & + & 0 & + & 0 & + & y_{34} & + & \dots & + & y_{3n} \\
\vdots & & & & & & & & & & & & \vdots \\
u_{n-1} & \geq & 0 & + & 0 & + & 0 & + & 0 & + & \dots & + & y_{nn} \\
 & & & & & & & & & & & & \\
\ell_2 & \geq & y_{12} & + & \dots & + & 0 & + & 0 & + & 0 & + & 0 \\
\vdots & & & & & & & & & & & & \vdots \\
\ell_{n-2} & \geq & y_{1(n-2)} & + & \dots & + & y_{(n-3)(n-2)} & + & 0 & + & 0 & + & 0 \\
\ell_{n-1} & \geq & y_{1(n-1)} & + & \dots & + & y_{(n-3)(n-1)} & + & y_{(n-2)(n-1)} & + & 0 & + & 0 \\
\ell_n & \geq & y_{1n} & + & \dots & + & y_{(n-3)n} & + & y_{(n-2)n} & + & y_{(n-1)n} & + & 0 \\
 & & & & & & & & & & & & \\
 & & y_{11} \geq 0 & & \dots & & \dots & & \dots & & y_{1n} \geq 0 & & \\
 & & & & & & & & & & & & \\
 & & \vdots & & \vdots & & \vdots & & \vdots & & \vdots & & \\
 & & & & & & & & & & & & \\
 & & y_{1n} \geq 0 & & \dots & & \dots & & \dots & & y_{nn} \geq 0 & &
\end{array}\right)$$

Das Gleichungssystem enthält mindestens eine Lösung. Diese triviale Lösung besteht darin, dass alle $y_{ik}$ den Wert Null erhalten. Die Lösung entspricht dem Umstand, dass keine Objekte während der Nutzzeit irgendeines Aufenthaltsknotens geladen werden.

Je nach Optimierungsziel ergeben sich unterschiedliche Zielfunktionen und zusätzliche Nebenbedingungen:

$2MLB$, $2MLB^*$ können mit Hilfe von 4.32 abgebildet werden.

Es wird eine Hilfsvariable $z_{max}$ eingeführt, welche zu minimieren ist.

$$\forall a_j \in A : z_{max} \geq \left( \ell_j - \sum_{i=1}^{n} y_{ik} \right),\ a_j \in G_k \tag{4.32}$$

$$min \Rightarrow z_{max}$$

Die Differenz $\left( \ell_j - \sum_{i=1}^{n} y_{ik} \right)$ entspricht der Restladezeit des Knotens $a_j$, welcher der Knotengruppe $G_k$ angehört.

$MSLB$, $MSLB^*$ können mit Hilfe von 4.33 gelöst werden.

$$min \Rightarrow \left( \sum_{i=1}^{n} \ell_i \right) - \sum_{\forall G_k \in B} \sum_{i=1}^{n} y_{ik} \tag{4.33}$$

Die Restladezeiten aller $n$ Knoten aller Teilgruppen in den einzelnen Teilbäumen werden aufsummiert.

Zur Abbildung der sessionbasierten Zielstellungen wird eine zusätzliche Dimension mit Index $h$ für eine dreidimensionale Matrix mit konstanten Elementen $C_{hik}$ eingeführt. Die Matrix wird dazu verwendet, alle Knoten des Baums jeweils genau einer der höchstens $n-1$ möglichen Sequenzen zuzuordnen[21], so dass mit jeder der zusätzlichen Nebenbedingungen immer genau eine Sequenz berücksichtigt wird. Jede einzelne Sequenz $\sigma$ wird mit $h$ indiziert und enthält maximal $n$ Objekte.[22]

Die Werte der in Abhängigkeit von der Struktur des Baums konstanten Elemente $C_{hik}$ ergeben sich wie folgt:

$$C_{hik} = \begin{cases} 1, & falls\ \ \exists (a_j \in G_k\ und\ a_j \in \sigma_h) \\ 0, & sonst \end{cases} \tag{4.34}$$

Da im Regelfall nicht $n-1$ unterschiedliche Sequenzen durch den Baum dargestellt werden,[23] gilt für solche Indizes $h$, welche keiner Sequenz zugeordnet sind, dass für die entsprechenden Sequenzen mit diesen Indizes gilt: $s_h = \emptyset$ und damit folgend der obigen Gleichung alle $C_{hik}$ mit demselben Index $h$ auch Null sind.

Auf Basis dieser Matrix werden die sequenzbasierten Zielstellungen als lineares Programm dargestellt.

[21] Ein nicht redundanter Navigationsbaum mit $n-1$ direkten Kindern des Wurzelknotens.

[22] Es gilt $n > 1$. Der triviale Fall für $n = 1$ wird nicht betrachtet.

[23] Dies ist nur der Fall, wenn jede Sequenz des Baums aus genau 2 Knoten besteht.

$2MLS$, $2MLS^*$ können mit Hilfe von 4.35 abgebildet werden.

$$\forall a_j \in A, \forall \sigma_h \in B, a_j \in G_k : z_{max} \geq \left( \ell_j - \left( \sum_{i=1}^{n} C_{hik} \cdot y_{ik} \right) \right) \tag{4.35}$$

$$min \Rightarrow z_{max}$$

$2MSLS$, $2MSLS^*$ können mit Hilfe von 4.36 abgebildet werden. Dabei wird auch hier die zusätzliche Matrix mit den konstanten Elementen $C_{hik}$ verwendet.

$$\forall \sigma_h \in B : z_{max} \geq \left( \left( \sum_{\forall a_i \in \sigma_h} \ell_i \right) - \sum_{\forall G_k \in B} \sum_{i=1}^{n} (C_{hik} \cdot y_{ik}) \right) \tag{4.36}$$

$$min \Rightarrow z_{max}$$

$MSLS$, $MSLS^*$ können im Gegensatz zu den anderen beiden sessionbasierten Zielstellungen ohne die zusätzliche Dimension mit Index $h$ abgebildet werden.

Es kann eine der unter 4.33 angegebenen Zielfunktionen ähnliche Zielstellung, erweitert um die Faktoren $D_i$, verwendet werden. Dabei ist der Wert der $D_i$ genau so zu wählen, dass dieser der Anzahl der Blätter des Teilbaums mit Wurzel $a_i$ entspricht.

Die mögliche vereinfachte Darstellung ergibt sich aus folgendem, später in Abschnitt 4.1.6 noch genauer dargestelltem Zusammenhang:

$$\left\{ \begin{array}{c} min \Rightarrow \sum_{\forall \sigma_h \in B} \left( \left( \sum_{\forall a_i \in \sigma_h} \ell_i \right) - \sum_{\forall G_k \in B} \sum_{i=1}^{n} (C_{hik} \cdot y_{ik}) \right) \\ C_{hik} = \{0,1\} \end{array} \right\} =$$

$$\left\{ \begin{array}{c} min \Rightarrow \left( \sum_{\forall a_i \in \sigma_h} \ell_i \right) - \sum_{\forall G_k \in B} \sum_{i=1}^{n} (D_i \cdot y_{ik}) \\ D_i = \sum_{\forall \sigma_h \in B} \sum_{\forall G_k \in B} C_{hik} \end{array} \right\} \tag{4.37}$$

Aus der Formulierung der Nebenbedingungen sowie der Zielstellungen in Form eines linearen Programms kann Satz 4.16 bestätigt werden. □

So entsteht mit der Lösung des linearen Programms eine Entscheidungsmatrix, welche alle $y_{ik}$ enthält und einfach in die Entscheidungsmatrix $X$ mit ganzzahligen Entscheidungsvariablen nach Bedingung 3.9, S.65 des Offline-Modells überführt werden kann.

Die Größe der mit 4.27 – 4.30 beschriebenen Matrizen, die je Zielfunktion zusätzlich benötigten Gleichungen und die Länge der Zielfunktionen entwickeln sich polynomial in Abhängigkeit

von der Anzahl der Knoten des Navigationsbaums. Lade- und Nutzzeiten der Knoten gestalten sich unabhängig von der Anzahl der Knoten und der Struktur, das heißt der Anzahl Blätter, des Baums. Sie durch den Anwendungszusammenhang des Modells vernachlässigbar klein.[24] Zur Lösung der vorgeschlagenen linearen Programme stehen eine Vielzahl von Verfahren zur Verfügung. Hier kann z.B. die in der Praxis häufig verwendete, im Regelfall sehr schnell arbeitende Simplex-Methode eingesetzt werden. Die Simplex-Methode zeigt zwar ein schlechtes worst-case-Verhalten, jedoch tritt exponentieller Aufwand nur für spezielle Probleminstanzen auf, so genannte Artefakte, welche bei Problemstellungen aus der Praxis nur äußerst selten anzutreffen sind.[25] Die Diskussion des Auftretens solcher Artefakte für die hier vorgeschlagenen linearen Programme soll nicht Gegenstand dieser Arbeit sein. Es sind weitere, im strengen Sinne effiziente Methoden zur Lösung bekannt, die jedoch in ihrer praktischen Anwendung i.d.R. dem Simplexverfahren unterlegen sind.[26] Mit diesen Verfahren können die hier vorgeschlagenen linearen Programme effizient gelöst werden.

### 4.2.2 Approximationsverfahren für redundante und nicht redundante Navigationsbäume

Es werden die im vorherigen Abschnitt für die Optimierungsprobleme formulierten, effizient lösbaren, linearen Programme dazu verwendet, ein effizientes Approximationsverfahren anzugeben. Die linearen Programme liefern als Lösung eine Entscheidungsmatrix $Y$ mit den reellwertigen Entscheidungsvariablen $y_{ik} \in Y$. Der Wert der relaxierten Lösung stellt eine untere Schranke für jede der Zielstellungen des Offline-Modells dar. Durch Abrunden der Entscheidungsvariablen $y_{ik}$ auf den nächsten ganzzahligen Wert, kann eine Nährungslösung konstruiert werden. Die Güte dieser Lösung in Bezug auf den Optimalwert entwickelt sich jedoch polynomial in Abhängigkeit von der Höhe des Navigationsbaums. Die je nach Zielstellung ermittelte Entscheidungsmatrix $Y$ wird nach den Bedingungen des Offline-Modells in eine gültige Lösung mit ganzzahligen Entscheidungsvariablen $x_{ij} \in X$ für nicht redundante sowie redundante Bäume überführt:

$$\forall x_{ij} \in X : x_{ij} = \begin{cases} \lfloor y_{ik} \rfloor, & falls\ a_j \in G_{ki} \\ 0, & sonst \end{cases} \tag{4.38}$$

Der Anteil Nutzzeit $y_{ik}$ des Knotens $a_i$, welcher dazu verwendet wird, um Knoten der Gruppe $G_{ki}$ des Teilbaums $T_i$ mit Wurzel $a_i$ zu laden, wird auf den nächsten ganzzahligen Wert abge-

[24] Eine Nutz- und Ladezeit eines Präsentationsobjekts von mehreren Stunden sind bereits ungewöhnlich. Werden Nutz- und Ladezeiten in Sekunden angegeben und auf zwei Stunden begrenzt, dann ist der Höchstwert 7200 Sekunden.

[25] Die Konstruktion solcher Artefakte, welche zu einem exponentiellen Rechenzeitaufwand für die Simplex-Methode führen, werden ausführlich von *Klee* und *Minty* in [KM72] und nochmals in [Sch00] von *Schrijver* aufgegriffen diskutiert.

[26] Von *Chandru* und *Rao* werden diese Aspekte in [CM99], S.31-32 eingehend diskutiert. Sie untersuchen Verfahren zur Lösung linearer Programme mit reellwertigen Entscheidungsvariablen in Hinblick auf ihre Eignung für unterschiedliche Probleminstanzen. Die Autoren verweisen darauf, dass in Abhängigkeit von der Größe der Inputvariablen, das heißt der Anzahl der Nebenbedingungen, der Anzahl der Entscheidungsvariablen in Nebenbedingungen, der Zielfunktion und der binären Kodierung der Entscheidungsvariablen, im strengen Sinne effiziente Verfahren wie z.B. die Elipsoid-Methode (siehe *Bland*, *Goldfarb* und *Todd* in [BGT81]) existieren.

rundet ($\lfloor y_{ik} \rfloor$). Er muss jedoch für die $x_{ij}$ Null betragen, für welche $a_j$ nicht in $G_{ki}$ enthalten ist und damit auch kein Knoten des Teilbaums $T_i$ ist.

Durch das Abrunden der reellwertigen Entscheidungsvariablen wird eine ganzzahlige Lösung erzeugt, welche für das Offline-Modell unter ganzzahligen Entscheidungsvariablen nicht optimal sein muss, aber immer den Modellbedingungen genügt. Der daraus resultierende Fehler ist abhängig von der Höhe $H(B)$ des Baums,[27] weil das Objekt eines Knotens $a^*$ nur während der Nutzung genau so vieler Objekte anderer Knoten geladen werden kann, wie sich auf dem Weg von der Wurzel des Baums zu $a^*$ befinden. Das sind maximal $H(B)-1$. Deshalb entsteht mit jeder Restladezeit eines Knotens, die in den Zielfunktionswert eingeht, ein kleinerer Fehler als $H(B)-1$. Dieser Sachverhalt ist in Abb. 4.7 dargestellt.

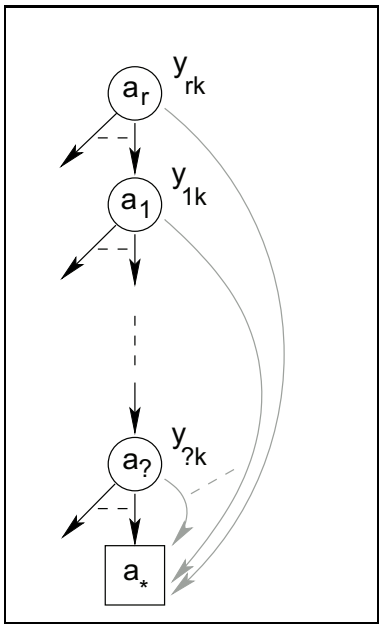



Abbildung 4.7: Entwicklung der Restladezeit eines Knotens $a_*$, welcher Knotengruppe $G_k$ angehört

Die in die Zielfunktionen eingehende Restladezeit eines jeden Knotens setzt sich aus den Ladeentscheidungen der Knoten auf dem Weg von der Wurzel zu diesem Knoten zusammen. Die Länge des Weges ist durch die Höhe des Baums begrenzt. Die Restladezeit eines Blattknotens $a^*$, angehörig der Gruppe $G_k$, in Abb. 4.7 links dargestellt, ergibt sich aus höchstens um eins weniger als der Höhe des Baums entsprechend vielen $y_{ik}$ größer Null eines Teilsbaums $T_i$.

Indirekt hat auch die Anzahl der Knotengruppen sowie die Anzahl der Knoten in den einzelnen Gruppen eines Baums Einfluss auf den Fehler, denn es gilt, dass der Baum nicht höher sein kann, als die maximal mögliche Anzahl nicht redundanter Knoten bzw. die Anzahl $q$ unterschiedlicher Knotengruppen $G_k$ auf einem Weg von der Wurzel hin zu einem Blatt. Da jeder Knoten des Baums in jeder Gruppe genau einmal vermerkt ist, gilt folgende Ungleichung für die Höhe des Baums:

$$H(B) \leq q \leq n+q-\left(\sum_{\forall G_k \in B} |G_k|\right) \leq n \tag{4.39}$$

[27] Die Höhe des Baums entspricht der maximalen Anzahl der Knoten auf einem beliebigen Weg von der Wurzel zu einem Blatt (vgl. z.B.[OW02], S.253).

| Problemstellung | $f$ |
|---|---|
| $2MLB, 2MLB^*$ | $H(B)-1$ |
| $MSLB, MSLB^*$ | $(H(B)-1)(n-1)$ |
| $2MLS, 2MLS^*$ | $H(B)-1$ |
| $2MSLS, 2MSLS^*$ | $(H(B)-1)^2$ |
| $MSLS, MSLS^*$ | $H(B)(H(B)-1)^2$ |

Tabelle 4.2: Fehlerwert $f$ je nach Optimierungsziel in Abhängigkeit von der Höhe $H(B)$ und der Anzahl Knoten $n$ des Navigationsbaums

Für jede der fünf Optimierungsprobleme für redundante sowie für nicht redundante Navigationsbäume soll nun geprüft werden, wie sich der Fehler in Abhängigkeit von der Höhe des Baums entwickelt.

**Satz 4.17** *Der Lösungswert $L^{\mathbb{Z}}(B)$ des ganzzahligen Programms für den Navigationsbaum B kann nicht besser sein als der Lösungswert der korrespondierenden reellwertigen Lösung $L^{\mathbb{R}}(B)$ des linearen Programms. Durch Abrunden der reellwertigen Lösung entsteht ein Fehlerwert $f$, um welchen $L^{\mathbb{R}}(B)$ ansteigt, wobei $f$ je nach Optimierungsziel die Werte der folgenden Tabelle 4.17 annimmt. Es gilt:*

$$L^{\mathbb{R}}(B) \leq L^{\mathbb{Z}}(B) < L^{\mathbb{R}}(B) + f \tag{4.40}$$

Die reellwertige Lösung liefert eine untere Schranke, die Lösung des linearen Programms, und eine obere Schranke, welche sich aus der unteren Schranke zuzüglich des von der Zielfunktion abhängigen Rundungsfehlers ergibt.

**Beweis von 4.17 :** Zum Nachweis ist die Gültigkeit aller mit dem Satz beschriebenen Ungleichungen zu prüfen.

$2MLB$**,** $2MLB^*$**,** $2MLS$**,** $2MLS^*$ Für alle vier Zielstellungen existiert mindestens ein Knoten $a_* \in B$ mit Index $i = *$, dessen Differenz aus Ladezeit $\ell_*$ und Summe aller Entscheidungen für das Laden des Objekts in $a_*$ den optimalen Zielfunktionswert $z^*$ bildet. Dieser minimale Maximalwert ergibt sich aus der folgenden Differenz, wobei $T_j$ ein Teilbaum aus $B$ mit Wurzelknoten $a_j$ ist[28]:

[28]Siehe Abschnitt 4.2.1 „Modellformulierung als lineares Programm“, Nebenbedingungen 4.27 - 4.29, S.131 sowie 4.32, S.133 bzw. 4.35, S.134

$$z^* = r_* = \ell_* - \sum_{j=1}^{n} y'_j, \quad y'_j = \left\{ \begin{array}{ll} y_{jk}, & falls \;\; a^* \in T_j \\ & \\ 0, & sonst \end{array} \right\}, \;\; a_* \in G_k$$
$$f = \left( \sum_{j=1}^{n} y'_j \right) - \left( \sum_{j=1}^{n} \lfloor y'_j \rfloor \right) < H(B) - 1 \tag{4.41}$$

Entsprechend der Höhe des Baums sind bis zu $H(B)-1$ viele $y'_j$ größer Null möglich, aus welchen sich der optimale Funktionswert $r_*$ zusammensetzt. Damit entsteht durch das Abrunden all dieser $y'_j$ je $y'_j > 0$ ein sich kumulierender Fehlerbeitrag von jeweils kleiner 1. Das heißt, $f$ ist kleiner als $(H(B)-1)$.

*MSLB*, *MSLB*$^*$ Der Zielfunktionswert $z^*$ für *MSLB* sowie *MSLB*$^*$ setzt sich wie folgt zusammen:[29]

$$z^* = \left( \sum_{i=1}^{n} \ell_i \right) - \sum_{\forall G_k \in B} \sum_{i=1}^{n} y_{ik}$$
$$f = \left( \sum_{\forall G_k \in B} \sum_{j=1}^{n} y_{ik} \right) - \left( \sum_{\forall G_k \in B} \sum_{j=1}^{n} \lfloor y_{ik} \rfloor \right) < (H(B)-1)(n-1) \tag{4.42}$$

Der Optimalwert ist die Differenz aus der Summe aller Restladezeiten der Knoten und der Summe aller Entscheidungsvariablen $y_{ik}$ bzw. Restladezeiten.

Jeder Knoten $a_i$, welcher kein Blattknoten ist, kann bis zu $H(B)-1$ Entscheidungsvariablen $y_{ik} > 0$ enthalten. Das sind um 1 weniger als $H(B)$, weil während der Nutzzeit des Knotens keine Knoten derselben Gruppe zu laden sind.[30] Für Blattknoten $a_i = \Diamond$ gilt, dass alle $y_{ik}$ Null sind. Es ergibt sich der mit 4.42 angegebene Wert für $f$.

2*MSLS*, 2*MSLS*$^*$ Folgender Fehler ergibt sich für 2*MSLS* und 2*MSLS*$^*$, wobei mindestens eine Sequenz $\sigma^* \in B$ existieren muss, deren Summe Restladezeiten aller Knoten der Sequenz den minimalen Höchstwert $z^*$ bildet. Der Optimalwert der reellwertigen Lösung setzt sich wie folgt zusammen:[31]

[29] Siehe Nebenbedingungen 4.27 - 4.29, S.131 sowie 4.33, S.133

[30] Laut Modellbedingung dürfen auf jedem Weg von der Wurzel bis zu einem Blatt keine zwei Knoten der selben Gruppe enthalten sein (Forderung nach Zyklenfreiheit des Navigationsbaums).

[31] Siehe Nebenbedingungen 4.27 - 4.29, S.131 sowie 4.36, S.134

$$z^* = \left( \sum_{\forall a_i \in \sigma_h} \ell_i \right) - \left( \sum_{\forall G_k \in B} \sum_{i=1}^{n} C_{hik} \cdot y'_{ik} \right), C_{hik} = \{0,1\}, \;\; \sigma_h = \sigma^*$$

$$y'_{ik} = \left\{ \begin{array}{ll} y_{ik}, & falls \; \exists a_j : (a_j \in G_k \; und \; a_j \in \sigma_h \; und \; a_j \in T_i \;) \\ 0, & sonst \end{array} \right\} \tag{4.43}$$

$$f = \left( \sum_{\forall G_k \in B} \sum_{i=1}^{n} C_{hik} \cdot y'_{ik} \right) - \left( \sum_{\forall G_k \in B} \sum_{i=1}^{n} C_{hki} \cdot \lfloor y'_{ik} \rfloor \right) < (H(B)-1)^2$$

Ein Entscheidungswert $y_{ik}$ wird nur dann zur Summe der Restladezeiten einer Sequenz addiert, wenn sich ein Knoten der Gruppe $G_k$ in der Sequenz sowie im Teilbaum mit Wurzel $a_i$ befindet.

Eine Sequenz kann nicht mehr als $H(B)$ Knoten enthalten. Da die Restladezeit des Blattknotens immer ganzzahlig sein muss, ergibt sich der Zielfunktionswert aus höchstens $(H(B)-1)$ nicht ganzzahligen Restladezeiten. Diese resultieren wiederum aus höchstens $(H(B)-1)$ Entscheidungsvariablen größer Null. Es ergibt sich der mit 4.43 angegebene Fehler.

*MSLS*, *MSLS*$^*$ Der Zielfunktionswert setzt sich aus der Summe der Restladezeiten aller Sequenzen des Baums zusammen. Ein Baum kann nach Modelldefinition nicht mehr als $H(B)$ verschiedene Sequenzen enthalten. Der Fehler für $2MSLS$ und $2MSLS^*$ erhöht sich so um das $H(B)$-fache. Es ergibt sich der Fehler:

$$f = H(B) \left( \left( \sum_{\forall G_k \in B} \sum_{i=1}^{n} C_{hik} \cdot y'_{ik} \right) - \left( \sum_{\forall G_k \in B} \sum_{i=1}^{n} C_{hik} \cdot \lfloor y'_{ik} \rfloor \right) \right) < H(B)(H(B)-1)^2 \tag{4.44}$$

Alle in Satz 4.17 angegebenen Fehlerwerte konnten nachgewiesen werden. □

Mit dem angegebenen, sehr einfachen Verfahren kann eine Approximationsgüte für die Optimierungsziele, abhängig von der Knotenanzahl des Baums, angegeben werden. Zwar lassen sich die so erzeugten Näherungslösungen durch „Umverteilung“ der abgerundeten Nutzzeiten der Entscheidungsvariablen noch verbessern, jedoch ist dem Autor kein Verfahren bekannt, welches zu einer Approximationsgüte unabhängig von der Knotenanzahl führt. Aber:

**Bemerkung:** *Der Fehler $f$ ist nicht allein durch die Höhe des Baums sondern auch durch die Summe der Nutzzeiten aller Knoten begrenzt. Außer für MSLS und MSLS$^*$ gilt:*

$$f < \sum_{\forall a_i \in B} u_i \tag{4.45}$$

*Für MSLS und MSLS$^*$ gilt hingegen:*

$$f < \sum_{\forall a_i \in B} \left( u_i^2 \right) \tag{4.46}$$

Die Summe der Ladeentscheidungen $y_{ik}$ der Lösung des linearen Programms kann nicht größer als die Summe bzw. das Quadrat der Summe aller Nutzzeiten des Baums sein. Das heißt, würden Zielstellungen betrachtet, in welchen neben der Restladezeit auch die Nutzzeiten der Knoten Bestandteil des Optimums sind, zum Beispiel die Minimierung der Summe aller Aufenthaltsdauern der Knoten eines Baums (entsprechend *MLB*) oder die Minimierung der höchsten Summe Aufenthaltsdauern einer Sequenz des Baums über alle Sequenzen (entsprechend 2*MSLS*), dann würde das Approximationsverfahren 2-approximative Lösungen liefern:

$$APP(B) < 2\ OPT(B), \tag{4.47}$$

wobei $APP(B)$ den Lösungswert des oben beschriebenen Approximationsverfahren und $OPT(B)$ den dazugehörigen Optimalwert liefern. Es erscheint sinnvoll, die Lösungsgüte des Approximationsverfahrens nicht nur losgelöst von den Nutzzeiten zu betrachten, denn Restladezeiten sind als um so gravierender einzustufen, je kleiner die Nutzzeiten im Verhältnis dazu sind.

Zum Abschluss dieses Abschnitts soll die Komplexität des folgenden Approximationsproblems $APP^+$ betrachtet werden, in welchem ein Approximat für die Optimierungsprobleme des Offline-Modells nicht schlechter als das jeweilige Optimum zuzüglich einer beliebig großen, additiven Konstante gesucht wird. Es kann gezeigt werden, dass $APP^+$ mindestens genauso schwer zu lösen ist, wie das *NP*-vollständige Entscheidungsproblem 3-Dimensionales Matching (3DM).[32]

Das Approximationsproblem $APP^+$ hat folgende Struktur:

Optimierungsproblem $APP^+$

Probleminstanz: Gegeben ist eine Probleminstanz bestehend aus redundantem Navigationsbaum, Bedingungen des Offline-Modells und Zielfunktion eines der fünf Optimierungsprobleme $\{2MLB^*, MSLB^*, 2MLS^*, 2MSLS^*, MSLS^*\}$ und ein beliebig großer, ganzzahliger und positiver Wert $\Delta \in Z_1^+$.

Gesucht ist ein Lösungswert $z$, für welchen gilt:

$$z \leq z^* + \Delta, \tag{4.48}$$

wobei $z^*$ der optimale Zielfunktionswert entsprechend der Zielfunktion ist.

Es wird gezeigt, dass folgender Satz gilt:

**Satz 4.18** $3DM \leq_p APP^+$

Aus Satz 4.18 folgt dann direkt, dass wenn $P \neq NP$ gilt, für $APP^+$ kein Lösungsverfahren existieren kann, welches Lösungen mit polynomialem Aufwand berechnet.

[32] Vgl. zum 3*DM* Kapitel 3, S.82

**Beweis von Satz 4.18:** Zum Nachweis wird die Lösung spezieller Probleminstanzen von $APP^+$ betrachtet. Diese Probleminstanzen werden mit folgendem Optimierungsproblem „Modifiziertes einfaches redundante Ladeproblem“ (*MERL*) zusammengefasst. Es wird gezeigt, dass beliebige Probleminstanzen des Entscheidungsproblems 3*DM* mit polynomiellem Aufwand in Probleminstanzen des Optimierungsproblems *MERL* überführt werden können und aus der Lösung dieser Instanzen mit polynomiellem Aufwand auf eine Lösung für das 3*DM* geschlossen werden kann.

*MERL* ist eine Modifikation des „Einfachen redundanten Ladeproblems“ (*ERL*), auf S.83 angegeben.

Entscheidungsproblem *MERL*

Probleminstanzen von *MERL* stimmen mit Probleminstanzen von *ERL* außer folgenden Veränderungen überein:

- Zusätzlich gegeben ist eine Zielfunktion eines der fünf Optimierungsprobleme $\{2MLB^*, MSLB^*, 2MLS^*, 2MSLS^*, MSLS^*\}$ und ein beliebig großer, ganzzahliger und positiver Wert $\Delta \in Z_1^+$.
- Das Objekt des Wurzelknotens wird nicht $3q+q$ sondern $(3q+q)(1+\Delta)c$ Zeiteinheiten genutzt, wobei $c \in Z_1^+$ eine von der Zielfunktion $Z$ abhängig zu wählende Konstante ist.
- Die Nutzzeiten $(\hat{u}_i^*)$ der Knoten des zweiten Niveaus $w_j, x_j, y_j, j = 1 \ldots q$ entspricht nicht um 1 weniger als die Anzahl der Kindknoten des jeweiligen Knotens $a_i$, sondern dem $(1+\Delta)c-fachen$ dessen.
- Die Ladezeiten der Knoten des zweiten und dritten Niveaus sind nicht 1 sondern $c(1+\Delta)$.

Gesucht ist ein Lösungswert $z$, für welchen gilt:

$$z \leq z^* + \Delta, \tag{4.49}$$

wobei $z^*$ der optimale Zielfunktionswert entsprechend der Zielfunktion ist.

**Bemerkung:** *Alle Probleminstanzen von MERL sind auch Probleminstanzen von* $APP^+$.

Probleminstanzen des Entscheidungsproblems 3*DM* lassen sich auf nahezu identische Weise in Instanzen für *MERL* überführen, wie Instanzen von 3*DM* nach Instanzen von *ERL*.[33] Einziger Unterschied besteht in den veränderten Nutz- und Ladezeiten der Knoten des Baums. Sie sind

[33] Vgl. Problembeschreibung von *ERL*, S.83

abhängig von den Parametern $q$, gegeben durch die Instanz des $3DM$, $\Delta$, welches beliebig groß sein kann[34], und $c$, welches jeweils abhängig von der betrachteten Zielfunktion gewählt wird.

Um von einer Lösung einer Instanz von *MERL* auf eine Lösung einer Instanz von $3DM$ schließen zu können, soll gezeigt werden, dass folgende Aussage für jede der fünf Zielfunktionen bei einem bestimmten $c$ zutrifft:

Ist der Lösungswert $z$ kleiner oder gleich $\Delta$, dann existieren in der Probleminstanz von *MERL* $q$ redundante Knotentripel. Ist der Lösungswert $z$ größer $\Delta$, dann können höchsten $q-1$ redundante Knotentripel existieren.

Wie bereits definiert, besteht ein redundantes Knotentripel aus drei Knoten, welche einer Knotengruppe angehören. Die Objekte aller drei Knoten können während der Nutzdauer einer gemeinsamen Wurzel vollständig geladen werden, wenn mindestens genau so viel Nutzzeit zur Verfügung steht, wie die Ladezeit nur eines Knotens des Tripels beträgt.[35] Auch wurde bereits gezeigt, dass aufgrund der Existenz bzw. nicht Existenz von $q$ solchen redundanten Tripeln auf die Antwort „ja“ bzw. „nein“ für $3DM$ geschlossen werden kann.[36]

Für ein beliebig großes $c$ gilt unabhängig von der Zielfunktion: Existieren $q$ redundante Knotentripel für eine Instanz von *MERL*, dann ist der optimale Lösungswert $z^* = 0$. Für die Lösung von *MERL* muss gelten: $z \leq \Delta$.

Im Folgenden wird je Zielstellung gezeigt, dass ein $c$ so gewählt werden kann, dass wenn höchsten $q-1$ redundante Knotentripel einer Instanz *MERL* existieren, der optimale Zielfunktionswert $\Delta+1$ sein muss. Ein Lösungsverfahren für *MERL* muss dann ein $z$ größer $\Delta$ liefern, wenn keine $q$ redundanten Knotentripel existieren. $3DM$ kann anhand des Werts $z$ gelöst werden.

$2MLB^*$, $c = |S|+q$ , wobei S eine mit der Probleminstanz aus $3DM$ definierte Menge ist, aus welcher die Instanz für *MERL* konstruiert wird.

Existieren nur $q-1$ redundante Knotentripel, dann ist die Summe der Restladezeiten aller Knoten des Baums für eine optimale Lösung mindestens $(\Delta+1)(|S|+q)$ und zwar dann genau $(\Delta+1)(|S|+q)$, wenn das $q$-te Knotentripel zwei zueinander redundante und einen zu diesen beiden Knoten nicht redundanten Knoten enthält.[37] Unter der gegebenen Zielfunktion muss der Opimalwert deshalb mindestens $\Delta+1$ sein, weil der Baum exklusive Wurzel genau $|S|+q$ Knoten enthält. Das heißt, dass eine Lösung von *MERL* größer $\Delta$ sein muss, wenn $q-1$ oder weniger redundante Knotentripel existieren.

$MSLB^*$, $c = 1$

Existieren nur $q-1$ redundante Knotentripel, dann ist die Summe der Restladezeiten aller Knoten des Baums für eine optimale Lösung mindestens $\Delta+1$ und zwar dann genau $\Delta+1$, wenn das $q$-te Knotentripel zwei zueinander redundante und einen zu diesen beiden Knoten nicht redundanten Knoten enthält. Der Zielfunktionswert ist identisch zur Summe der Restladezeiten aller Knoten des Baums. Der Opimalwert ist deshalb mindestens $\Delta+1$. Das heißt, dass das eine Lösung von *MERL* größer $\Delta$ sein muss, wenn $q-1$ oder weniger redundante Knotentripel existieren.

[34] Vgl. die Beschreibung von *APP+*

[35] Vgl. S.84

[36] Vgl. S.3.5.1ff.

[37] Werden $(\Delta+1)(|S|+q)$ Zeiteinheiten auf jeden der $|S|+q$ Knoten gleichmäßig verteilt, werden jedem einzelnen Knoten genau $\Delta+1$ Zeiteinheiten zugeordnet.

$MLS^*,\ c = |S| + q$

Der Nachweis ist identisch zum Nachweis für $2MLB^*$.

$2MSLS^*,\ c = |S| + q$

Existieren nur $q-1$ redundante Knotentripel, dann ist die Summe der Restladezeiten aller Knoten des Baums für eine optimale Lösung mindestens $(\Delta+1)(|S|+q)$ und zwar dann genau $(\Delta+1)(|S|+q)$, wenn das $q$-te Knotentripel zwei zueinander redundante und einen zu diesen beiden Knoten nicht redundanten Knoten enthält. Unter der gegebenen Zielfunktion muss der Opimalwert deshalb mindestens $2(\Delta+1)$ sein, weil der Baum exklusive Wurzel genau $|S|+q$ Knoten enthält.[38] Das heißt, dass eine Lösung von $MERL$ größer $\Delta$ sein muss, wenn $q-1$ oder weniger redundante Knotentripel existieren.

$MSLS^*,\ c = 1$

Existieren nur $q-1$ redundante Knotentripel, dann ist die Summe der Restladezeiten aller Knoten des Baums für eine optimale Lösung mindestens $\Delta+1$ und zwar dann genau $(\Delta+1)$, wenn das $q$-te Knotentripel zwei zueinander redundante und einen zu diesen beiden Knoten nicht redundanten Knoten enthält. Weil jeder Knoten mindestens in einer Session enthalten ist, muss der Optimalwert für Summe der Restladezeiten über alle Session des Baums mindestens $\Delta+1$ betragen. Das heißt, dass das eine Lösung von $MERL$ größer $\Delta$ sein muss, wenn $q-1$ oder weniger redundante Knotentripel existieren.

Für alle fünf Zielstellungen konnte gezeigt werden, dass mit Hilfe der Lösung von $MERL$ auf die Existenz bzw. nicht Existenz von $q$ redundanten Knotentripeln geschlossen werden kann. Aus diesem Wissen kann direkt eine Antwort für $3DM$ abgeleitet werden.

Weil $MERL$ nur Probleminstanzen aus $APP^+$ enthält, ist der Satz 4.18 zu bestätigen. □

## 4.3 Diskussion der Ergebnisse

Alle im vorherigen Kapitel aufgeführten Optimierungsprobleme des Offline-Modells lassen sich jeweils als ganzzahliges lineares Programm formulieren. Die Größe des Programms, das heißt die Anzahl der Entscheidungsvariablen der Ungleichungen und Zielfunktionen, wächst jeweils mit $O(n^2)$.

In Abhängigkeit von der Anzahl Knoten in den knotendisjunkten Knotengruppen $G_k$ des Navigationsbaums zeichnet sich folgendes Bild für die Rechenbarkeit der Optimierungsprobleme ab:

$\forall G_k \in B : |G_k| = 1$

Diese Probleminstanzen sind effizient rechenbar. Je nach Zielstellung gestaltet sich der Aufwand höchstens in der Größenordnung $O(n^4)$.

[38] Werden $(\Delta+1)(|S|+q)$ Zeiteinheiten auf jeden der $|S|+q$ Knoten gleichmäßig verteilt, werden jedem einzelnen Knoten genau $\Delta+1$ Zeiteinheiten zugeordnet. Damit muss die maximale Summe der Restladezeiten einer beliebigen Sequenz des Baums mindestens $2(\Delta+1)$ betragen.

$(\forall G_k \in B : |G_k| \leq 2)$ *und* $(\exists G_k \in B : |G_k| = 2)$

Für beliebige Probleminstanzen dieser Art konnte kein Nachweis für die effiziente Rechenbarkeit oder die *NP*-Vollständigkeit erbracht werden. Es ist zu vermuten, dass diese nicht effizient rechenbar sind.

$\exists G_k \in B : |G_k| \geq 3$

Optimierungsprobleme der $*$-Zielstellungen dieser Art sind *NP*-schwierig. Eine Lösung ist nur für ausgewählte Probleminstanzen praktikabel rechenbar, insbesondere abhängig von der Problemgröße $n$.

Beliebige Probleminstanzen der Optimierungsprobleme lassen sich mit Hilfe des vorgestellten Verfahrens approximieren. Der Fehler der approximierten Lösung entwickelt sich jedoch in Abhängigkeit von der maximalen Tiefe $H(B)$ des Navigationsbaums in Größenordnungen von bis zu $O(H(B)^3)$ bzw. $O(n^3)$. Die Güte der approximierten Lösung ist abhängig von der Höhe des Baums.

Es konnte kein Verfahren gefunden werden, welches approximative Lösungen für eine der $*$-Zielstellungen generiert, deren Güte sich in Relation zur optimalen Lösung, unabhängig von der Zahl der Knoten des Baums, entwickelt. Es ergeben sich zwei interessante weiter zu verfolgende Ansätze, um zu einem solchen Verfahren zu kommen. Einerseits wäre zu prüfen, inwieweit approximierte Lösungen postum durch Anpassen der Ladeentscheidungen zu einer besseren Lösung führen können.[39] Andererseits wäre zu prüfen, inwieweit die gezielte Selektion bestimmter Sequenzen bzw. Sequenzausschnitte aus einem redundanten Navigationsbaum zu gestalten wäre, so dass dieser in einen nicht redundanten Navigationsbaum überführt werden könnte und auf diesem basierend optimale Lösungen als approximierte Lösungen bestimmter Güte für Optimierungsprobleme des redundanten Navigationsbaums Verwendung finden könnten.

Alle vorgeschlagenen Verfahren werden als einfach zu realisieren eingeschätzt. Zur Approximation können frei verfügbare Werkzeuge zur Lösung der linearen Programme, basierend auf standardisierten Modellsprachen, eingesetzt werden.[40] Die Programme selbst lassen sich aus dem jeweiligen Navigationsbaum bzw. -graphen generieren. Die effizient rechenbaren Verfahren für nicht redundante Navigationsbäume werden in ihrer Realisierung auf einer konkreten Maschine als nicht übermäßig kompliziert eingeschätzt. Auf eine Implementation der Verfahren wurde verzichtet, weil deren praktische Anwendung nicht Thema dieser Arbeit ist.

Der Einsatz dieser Verfahren zur Unterstützung der beiden Entscheidungsfelder der Entwurfsphase eines Hypermediums wird im übernächsten Kapitel dargestellt. Der Einsatz der Offline-Verfahren im Rahmen des Online-Modells wurde schon angesprochen und wird im folgenden Kapitel auch untersucht.

[39] Solch eine Diskussion wird in Bezug auf Seiteneffekte der Online-Verfahren, losgelöst von einer zu garantierenden Güte, im nächsten Kapitel noch erfolgen.

[40] Dazu sei auf die standardisierte Modellierungssprache Opl und darauf aufsetzende Werkzeuge verwiesen (vgl. [HL99]) sowie auf die schon älteren, jedoch überblicksartig gestalteten Artikel [Gre93] und [Fou96], welche ausführlich die Werkzeugauswahl und Modellierungsaspekte besprechen.

# Kapitel 5

# Online-Verfahren

*„Die zwei größten Tyrannen der Erde: der Zufall und die Zeit."*
*Johann Gottfried von Herder*

In diesem Kapitel werden die Zielstellungen *MDS* und 2*MDS* sowie die entsprechenden $*$-Ziele für redundante Navigationsbäume untersucht. Zuerst wird ein für die Untersuchungen notwendiges, allgemein gehaltenes Algorithmengerüst zur Lösung von Problemstellungen des Online-Modells vorgeschlagen. Auf dessen Basis wird ein einfaches Lösungsverfahren konstruiert und dessen Güte untersucht. Dann werden im Vergleich zu diesem spezielle Online-Verfahren untersucht, welche sich Lösungsverfahren der Offline-Problemstellungen bedienen.

Die Güte der untersuchten Online-Verfahren wird wiederum mittels Kompetitivitätsanalyse gemessen. Dies erscheint sinnvoll, weil zu den jeweiligen Entscheidungszeitpunkten der Problemstellungen keine Informationen über zukünftige, möglicherweise im Voraus zu stellenden Anfragen verfügbar sind. Es werden nur solche Lösungsverfahren für die vier Zielstellungen als sinnvoll angesehen und in diesem Kapitel betrachtet, welche die in Abschnitt 3.5.2, S.86ff. per Kompetitivitätsanalyse aufgezeigten problemimmanent oberen Schranken nicht überschreiten.

Weiterhin ist der benötigte Rechenzeit- und Speicherplatzaufwand der Verfahren von wichtigerer Bedeutung als im Rahmen des Offline-Modells. Die Online-Verfahren werden dafür eingesetzt, zu bestimmten Zeitpunkten während des Betriebs Ladeentscheidungen zu treffen. Im bereits dargestellten Online-Modell wird jedoch die für eine Ladeentscheidung zur Verfügung stehende Zeitdauer nicht modelliert. Es wird angenommen, dass Entscheidungen in einem Zeitraum von ein bis zwei Sekunden möglich sind, welche mit ineffizienten Verfahren überschritten werden würde, wenn für die Problemstellung angemessene Problemumfänge angenommen werden. Damit ist Grundbedingung die Effizienz der Lösungsverfahren.[1]

[1] Gemeint ist Effizienz im Sinne eines asymptotisch wachsenden Rechenzeit- und Speicherplatzaufwands nach oben beschränkt in Abhängigkeit von der Größe des Navigationsbaums durch ein Polynom. Das garantiert zwar noch nicht eine Entscheidung im vorgegebenen Zeitraum, ist jedoch Grundlage dafür. Die Modellierung von zeitkritischen Entscheidungen in Echtzeit ist nicht Gegenstand dieser Arbeit sein.

## 5.1 Allgemeiner Lösungsansatz

Mit dem folgenden Verfahren wird ein Grobalgorithmus für die zu untersuchenden Online-Algorithmen sowie deren mit der Kompetitivitätsanalyse betrachteten Gegenspielern, den zugehörigen optimalen Offline-Algorithmen, vorgeschlagen. Den einzelnen Zustandsübergängen zwischen den nach Modelldefinition festgelegten Entscheidungszeitpunkten sowie den in unterschiedlichem Umfang zur Verfügung stehenden Informationen wird mit dem folgenden Verfahren entsprochen.

```
   Typ Zustand=Baum
   Typ Sequenz=Schlange(Index)
   Typ Zeitpunkt,Zeitraum=Zähler
   Typ Optimierungsergebnis=Zeit

1  ON( -> Ausgangszustand:Zustand
2       -> Inputsequenz:Sequenz
3       -> von,bis:Zähler
4       -> Aktion informiere(
5                    -> Inputsequenz:Sequenz,
6                    -> jetzt:Zähler,
7                    <-  Wissen:Sequenz )
8     ):Optimierungsergebnis
```

```
9   bekannt:Sequenz, i:Zähler
10  ZAlt,ZNeu:Zustand
11  Beginn
12    i=von
13    ZAlt= Ausgangszustand
14    informiere(Inputsequenz,i,bekannt)
15    solange i < (bis+1)
16    Beginn
17      ALG(ZAlt,ZNeu,bekannt,i)
18      i=i+1
19      ZAlt=ZNeu
20      informiere(Inputsequenz,i,bekannt)
21    Ende
22    liefereErgebnis(ZNeu)
23  Ende
```

**Alg. 5.1** ON(...)

*Algorithmengerüst zur Online-Entscheidungsfindung*[2]

Oben ist das allgemein gehaltene Algorithmengerüst mit Datentypen und Schnittstelle (links) sowie Anweisungsteil (rechts) zur Ermittlung des Optimierungsergebnisses des Online-Entscheidungsprozesses dargestellt.

Für die Entscheidungen der `bis-von+1` Entscheidungsschritte liegen folgende Informationen vor. Über:

- den `Ausgangszustand`

  Dieser ergibt sich aus den bereits vor dem Entscheidungsschritt `von` getroffenen Entscheidungen bzw. aus dem Initialzustand vor der ersten zu treffenden Entscheidung.

Zum Beispiel für ein `von=1` und `bis=3` werden die Entscheidungen der ersten drei Entscheidungszeitpunkte ausgeführt. Als Ergebnis wird das Optimierungsergebnis vom Typ `Zeit` geliefert.

[2] Die Definition der Datenstruktur `Baum` ist identisch zu der bereits im Offline-Modell verwendeten (siehe S. 103, Abschnitt 4.1 „Offline-Verfahren nicht redundanter Navigationsbäume“).

- die Inputsequenz

  Sie enthält die von Entscheidungsschritt zu Entscheidungsschritt zusätzlich verfügbaren Informationen, jeweils in einem Element der Sequenz vermerkt.

- die Funktion informiere(...)

  Sie liefert die je Entscheidungsschritt aus der Inputsequenz bereitzustellenden Elemente in Abhängigkeit vom Datenobjekt Wissen. Mit der Funktion informiere wird das eingeschränkt verfügbare Wissen über die Elemente der Inputsequenz je nach aktuellem Entscheidungsschritt i modelliert.

ON liefert das Optimierungsergebnis der Online-Entscheidungen für einen gegeben Sequenzausschnitt des von-ten bis bis-ten Elements der Inputsequenz, resultierend aus dem Endzustand. Die Funktion liefereErgebnis ermittelt aus diesem Zustand entsprechend der Zielstellung das Endergebnis. Resultiert dieses nicht nur aus dem Endzustand nach der zuletzt getroffenen Entscheidungen, sondern auch aus dem Ablauf der Entscheidungsfindung, dann ist das im Rahmen der den Zustand beschreibenden Datenobjekten entsprechend zu berücksichtigen.

Wann eine Entscheidung zu treffen ist bzw. zu welchem Zeitpunkt Änderungen des Wissens über die Elemente der Sequenz auftreten, wird nicht explizit modelliert. Dies erfolgt über die Inhalte der Elemente der Sequenz. Entscheidungszeitpunkte sind entweder fix, mit jeder Inputsequenz vorgegeben, oder gestalten sich variabel, indem vorher getroffene Entscheidungen sich auf den Zeitpunkt der jeweils folgenden, zu treffenden Entscheidungen auswirken.

In dem hier zu diskutierenden Online-Modell sind nur der Zeitpunkt des Beginns einer Session $t(s_1^1)$ und der folgende Entscheidungszeitpunkt $t(s_2^2))$ bekannt, denn es wird angenommen, dass zeitgleich mit Beginn einer Session auch die erste gestellte Anfrage bzw. das erste Element der Inputsequenz und damit auch die Nutz- und Ladezeit des ersten Aufenthaltknotens bekannt sind.[3] Zu jedem dieser Entscheidungszeitpunkte $t(s_i^i), 1 \leq i \leq m$ ist zusätzliches Wissen über die Inputsequenz gegenüber allen vorherigen Entscheidungszeitpunkten $t(s_j^j), j < i$ verfügbar.[4] Das Verfahren ALG trifft für jeden dieser Entscheidungszeitpunkte Entscheidungen. Dies ist der Entscheidungsschritt, welcher hier auf die Dauer eines Zeitpunkts, dass heißt auf einen Zeitraum der Länge Null, festgelegt wird. Aus den in einem Zeitpunkt $t(s_i^i)$ getroffenen Entscheidungen resultiert der Zustand $Z_{t(s_i^i)}$ eines Entscheidungsschritts. In ON wird dieser im Datenobjekt ZNeu vermerkt, welches das Ergebnis der getroffenen Entscheidungen bzw. die Entscheidungen selbst beinhaltet.

Um das Ergebnis des Online-Verfahrens über einen Entscheidungszeitraum $t(s_i^i)$ bis $t(s_j^j)$ geliefert zu bekommen, sind

- der Ausgangszustand $Z_{t(s_{i-1}^{i-1})}$ bzw. der Initialzustand $Z_0$ vor der ersten Entscheidung,

- der Entscheidungszeitraum durch Angabe der Position der jeweiligen Elemente der Inputsequenz $i$ bis $j$ sowie

[3] Vgl. zur Definition einer Session und der verwendeten Notation Abschnitt 3.3.2 „Das Online-Modell“, S. 69ff..

[4] Es erscheint sinnvoll nur jene Entscheidungszeitpunkte zu modellieren, zu welchen auch zusätzliches Wissen bekannt wird.

- eine Abbildung über die zum jeweiligen Zeitpunkt verfügbaren Informationen mit *informiere*( *Inputsequenz*, *Zeitpunkt*, *Inputsequenzausschnitt*)

zu übergeben.

Wird das Ergebnis jedes einzelnen Entscheidungsschritts gewünscht, ist ON für alle $i = 1 \ldots m$ wie folgt aufzurufen:

$$\mathsf{ON}(Z_{i-1}, s_1^m, i-1, i, \mathsf{informiere}(s_1^m, i, s_h^j), Z_i),$$

wobei informiere in den hier interessierenden Problemstellungen genau den Ausschnitt $s_h^j$ mit $h = 1$ und $j = i$ der Inputsequenz $s_1^m$ für jeden Entscheidungsschritt $i$ liefert.

Zur Lösung der hier beschriebenen Online-Problemstellungen wird der Zustand $Z_i$ mittels des Datenobjekts vom Typ Baum modelliert, welches den Navigationsbaum $B$ sowie die Entscheidungsmatrix $X$ der vorher getroffenen Ladeentscheidungen und daraus resultierenden Restladezeiten enthält. Die Elemente der Inputsequenz und deren Reihenfolge entsprechen den einzelnen während einer Session durch die lokale Anwendungskomponente gestellten Anfragen. Die Sequenz enthält damit die Indizes der Aufenthaltsknoten aller in dieser Session angefragten Objekte $\forall a_i \in s_1^m$. Das jeweilige Ergebnis der Online-Verfahren kann aus dem Ergebnis der letzten Entscheidungen, vermerkt im Baum mit $X_{t(s_m^m)}$, mit den bereits bekannten Funktionen Funktionswert{*für ...*}( -> B:Baum):Zeit der Offline-Algorithmen durch Auswerten der Restladezeiten ermittelt werden.

Entscheidend ist die Wahl bzw. Konstruktion eines Verfahrens für ALG aus ON. Es sollen hier zwei Möglichkeiten untersucht werden: Die Konstruktion eines „neuen“, sehr einfach gehaltenen Verfahrens und die Verwendung von Verfahren, welche schon für das Offline-Modell entworfen worden sind. Jeder einzelne Entscheidungsschritt soll wiederum als Offline-Problem aufgefasst werden. Von Interesse ist jedoch nicht die Güte der einzelnen Entscheidungsschritte, sondern die Güte des Endergebnisses im letzten Entscheidungsschritt bzw. Betrachtungen aller Entscheidungsschritte im zeitlichen Ablauf.

## 5.2 Ein einfaches Online-Verfahren zur Reduktion von Sessiondauer und Aufenthaltsdauer je Knoten

In diesem Abschnitt wird in Anlehnung an das Verfahren R{*für MSLB*} ein Entscheidungsverfahren für ALG entworfen, welches je Entscheidungszeitpunkt $t(s_1^1), \ldots, t(s_m^m)$ einer Session $s_1^m$ die Nutzzeit des jeweiligen Wurzelknotens $a_i$ eines Teilbaums für alle Knoten $a_i \in s_1^m$ dazu verwendet, Objekte der innerhalb des Teilbaums direkten Nachbarknoten zu $a_i$ im Voraus zu laden.

Folgendes Entscheidungsverfahren für die einzelnen Entscheidungsschritte wird vorgeschlagen:

```
1  Aktion ALG( -> BAlt:Baum,<- BNeu:Baum,
2              -> Historie:Teilsequenz, -> i:Zähler )
3  T:Baum, a:AKnoten, aL:AListe, Restzeit, Diff:Zeit
4  Beginn
5    T=liefereTeilbaumAus(BAlt,Historie)
6    aL=liefereDirekteKinder(T,T.a.i)
7    Restzeit=T.a.u
8    solange (Restzeit>0) und (nicht leer?(aL))
9    wiederhole Beginn
10     Diff = liefereDiff(aL,Restzeit)
11     a=entnimm(aL)
12     falls Diff>Restzeit Diff=Restzeit
13     falls Diff>a.l ReduziereRestladezeit(T, a.i, a.l)
14       sonst ReduziereRestladezeit(T, a.i, Diff)
15     Restzeit=Restzeit-Diff
16   Ende wiederhole
17   BNeu=BAlt
18   aktualisiereBaum(BNeu,T)
19 Ende
```

**Alg. 5.2** ALG *für Alg. 5.1*
*Einfache Ladestrategie*

```
AListe=Liste(Index)
Teilsequenz=Sequenz
```

Für ALG benötigte Datenobjekttypen sind oben, das von ON aufzurufende Verfahren links und verwendete Handlungen im Anhang, S.207 dargestellt.

Aus dem Navigationsbaum B, welcher die Restladezeiten der Knoten enthält, wird genau der Teilbaum T mit Wurzelknoten des letzten Elements der Historie, das heißt dem aktuell angefragten Knoten, extrahiert, zu welchem der Weg aus Historie im Gesamtbaum B führt. Für die Dauer der Nutzung des Wurzelknotens von T sind die Ladeentscheidungen zu treffen. Das erfolgt, indem die Nutzzeit zu gleichen Teilen dazu verwendet wird, die direkten Nachbarknoten des Wurzelknotens von T im Voraus zu laden. Mittels liefereDiff (Zeile 10) wird die Dauer des zum Laden im Voraus verfügbaren Zeitraums je Knoten aus aL ermittelt. Die Restladezeit der direkten Nachbarknoten wird entsprechend gemindert.

Mit jeder Iteration der wiederhole-Schleife wird genau eine Ladeentscheidung für einen direkten Kindknoten der Wurzel des Teilbaums getroffen und im Teilbaum vermerkt. Nach Beenden der Schleife werden alle Ladeentscheidungen und Restladezeiten des Teilbaums mit aktualisiere Baum im Gesamtbaum vermerkt. Rechenzeit- wie Speicherplatzaufwand des Verfahrens übersteigen nicht $O(n^2)$.

```
1  wähleDiff( -> aL:Liste(AKnoten),
2             -> Restzeit:Zeit ):Zeit
3  Beginn
4    liefere( Restzeit div Anzahl(aL))
5  Ende
```

**Alg. 5.3** wähleDiff *für* ALG *5.2*

Für dieses Verfahren wurde wähleDiff (links) so konstruiert, dass je Knoten aus aL dieselbe Zeitdauer zum Laden im Voraus zur Verfügung steht. In aL sind all jene direkten Kindknoten des Wurzelknoten des Teilbaums aus ALG enthalten, für welche in der aktuellen Iteration der wiederhole-Schleife noch keine Ladeentscheidungen getroffen wurden.

Dieses Verfahren wird in Hinblick auf die Zielstellungen *MDS* bzw. *MDS** und 2*MDS* bzw. 2*MDS** untersucht.[5] Es wird geprüft, wie sich der bereits ermittelte Kompetitivitätsfaktor von 2 als problemimmanente Schranke der Optimierungsprobleme verändert.

In den folgenden beiden Abschnitten wird das Ergebnis der „Einfachen Ladestrategie“ in Hin-

[5] *MDS*, *MDS**: Minimierung der Dauer einer Session, 2*MDS*, 2*MDS**: Minimierung der maximalen Dauer über alle Aufenthalte einer Session (siehe Abschnitt 3.3.2, S.69ff.)

blick auf die Ziele des Online-Modells untersucht. Zur vereinfachten Darstellung wird angenommen, dass die Indizes der Knoten des Baums, welche in der Session $s_1^m$ zur Nutzung angefordert werden, genau der Reihenfolge entsprechen, in welcher diese in $s_1^m$ auftreten:
$s_1^1 = a_1, \ldots, s_i^i = a_i, \ldots, s_1^m = a_m$.

### 5.2.1 *MDS* und *MDS**

Für die Zielstellung *MDS* („Minimierung der Dauer einer Session eines nicht redundanten Navigationsbaums“) wird jetzt geprüft, um wie viel die in Abschnitt 3.5.2, S.86ff. für die Zielstellung *MDS* nachgewiesene problemimmanent obere Schranke bei Einsatz sinnvoller Prefetchingverfahren bzw. ohne deren Einsatz abgesenkt werden kann, wenn die oben beschriebene „Einfache Ladestrategie“ in ON zur Entscheidungsfindung eingesetzt wird.

Bei Anwendung der einfachen Ladestrategie muss die obere Schranke der Sessiondauer um genau jenen Teil verringert werden, um welchen garantiert immer, unabhängig von $s_1^m$ ein Teil der direkten Kindknoten im Voraus geladen wird. Es ergibt sich folgender Satz:

**Satz 5.1** *Die problemimmanent obere Schranke des Online-Optimierungsproblems MDS, beschrieben durch den Kompetitivitätsfaktor c*

$$c \leq \begin{cases} \beta \geq 1 : 1 + \dfrac{1}{\beta} \\ \\ \beta \leq 1 : 1 + \beta \end{cases}$$

*sinkt auf den folgenden Wert, abhängig vom bereits bekannten Parameter* $\beta \in \mathbb{R} > 0$ *als Quotient der Summe aus Lade- und Nutzzeiten aller Knoten der Session* $s_1^m$*:*

$$c \leq \begin{cases} \beta \geq 1 : \dfrac{1}{\beta\, k_{max}} \\ \\ \beta \leq 1 : z_2(k_{max}, \beta), \quad 0 < z_2(k_{max}, \beta) \leq \beta, z_2(k_{max}, \beta) \approx \dfrac{1}{k_{max}}, \end{cases}$$

*wobei* $z_2$ *konstant ist, sich* $z_2$ *abhängig von den Konstanten* $\beta$ *und* $k_{max} \in \mathbb{Z}_1^+$ *entwickelt,* $k_{max}$ *die maximale Anzahl direkter Kindknoten aller Knoten der Session angibt sowie* $\beta$ *wie bereits bekannt definiert ist:*

$$\beta = \frac{\sum_{i=1}^{m} \ell_i}{\sum_{i=1}^{m} u_i} \tag{5.1}$$

**Beweis von Satz 5.1:**

Durch die Anwendung der einfachen Ladestrategie ergibt sich die folgende Restladezeit $r_i$ des $i$-ten, in der Session angeforderten Knotens $a_i$ in Abhängigkeit von der Lade- ($\ell_i$) und Nutzzeit ($u_i$):

$$r_i = \begin{cases} \ell_1 & falls \;\; i = 1 \\ \ell_i - \left\lfloor u_{i-1} \frac{1}{k_{i-1}} \right\rfloor & sonst \;\; i > 1 \end{cases} \tag{5.2}$$

$k_i$ bzw. $k_{i-1}$ ist die Anzahl direkter Nachbarknoten des Knotens $a_i$ bzw. des Vorgängerknotens von $a_i$, das heißt $a_{i-1}$, im Navigationsbaum.

Daraus ergibt sich folgende Dauer des $i$-ten Aufenthaltsknotens $d_i$ bzw. $D_{ON}(s_i^i)$ einer Session mit insgesamt $m$ Knoten und demzufolge auch $m$ zueinander nicht redundanten Knoten des Online-Verfahrens:

$$\begin{aligned} \forall i = 2 \ldots m : D_{ON}(s_i^i) &= u_i + \ell_i - u_{i-1} \left\lfloor \tfrac{1}{k_{i-1}} \right\rfloor \\ \forall i = 2 \ldots m : D_{ON}(s_i^i) &\leq u_i + \ell_i - u_{i-1} \left\lfloor \tfrac{1}{k_{max}} \right\rfloor , \end{aligned} \tag{5.3}$$

wobei $k_i$ die Anzahl der Teilbäume des Teilbaums mit Wurzelknoten $a_i$ ausdrückt (s.o.) und $k_{max}$ den maximalen Wert über alle $k_i, i = 1 \ldots m$ des gesamten Navigationsbaums.

Damit ergibt sich unter folgender Nebenbedingung

$$\forall i \geq 2 : \ell_i - \left\lfloor u_{i-1} \frac{1}{k_{max}} \right\rfloor \geq 0 \tag{5.4}$$

die folgende obere Schranke für die Sessiondauer des Online-Verfahrens, welche mit der Sessiondauer des zugehörigen optimalen Offline-Verfahrens verglichen werden soll.

$$D_{ON}(s_1^m) \leq \sum_{i=2}^{m} \left( u_i + \ell_i - \left\lfloor u_{i-1} \frac{1}{k_{max}} \right\rfloor \right) + \ell_1 + u_1, \ell_i - \left\lfloor u_{i-1} \frac{1}{k_{max}} \right\rfloor > 0 \tag{5.5}$$

Zur Vereinfachung der folgenden Analyseschritte erfolgt die Einschränkung auf Probleminstanzen der Session, die Bedingung 5.4 genügen. Sie hat keine Auswirkung auf das Ergebnis der folgende Kompetitivitätsanalyse, weil alle Probleminstanzen, für welche diese Bedingung nicht zu trifft, können in Probleminstanzen transformiert werden, die die Bedingung erfüllen, ohne ein Absenken der oberen Schranke für $ON$ bzw. ein Anheben der unteren Schranke für $OPT$. Die Transformation erfolgt, indem die Nutzzeit $u_{i-1}$ auf genau den Wert $\ell_i - \left\lfloor u_{i-1} \frac{1}{k_{max}} \right\rfloor$ des die Bedingung verletzenden Knotens $a_{i-1}$ reduziert wird.

Die untere Schranke wird durch Reduzieren der Nutzzeit nicht angehoben, weil $u_{i-1}$ in jedem Falle ausreicht, um das jeweils folgende Objekt der Session aus $a_i$ vollständig im Voraus zu laden. Die obere Schranke für den Online-Algorithmus wird nicht abgesenkt, weil die einfache Ladestrategie in jedem Fall nur Objekte der direkten Nachbarknoten des Baums im Voraus lädt. So bleiben obere und untere Schranke des Online- und optimalen Offline-Algorithmus trotz der Transformation erhalten.

Ein Online-Verfahren kann die Sessiondauer eines optimalen Offline-Verfahrens für *MDS*, wie bereits diskutiert,[6] nicht unterschreiten:

$$D_{OPT}(s_1^m) = max\left[(\sum_{i=1}^{m} u_i) + \ell_1, (\sum_{i=1}^{m} \ell_i) + u_m\right] \tag{5.6}$$

Es ergibt sich durch Umformung von $D_{ON}$, im Anhang auf S.209 dargestellt, folgende Ungleichung:

$$\frac{D_{ON}(s_1^m)}{D_{OPT}(s_1^m)} \leq \frac{(1+\beta)\sum_{i=1}^{m} u_i - \left\lfloor \frac{1}{k_{max}} \sum_{i=1}^{m-1} u_i \right\rfloor}{max\left[\left(\sum_{i=1}^{m} u_i\right) + \ell_1, \beta\left(\sum_{i=1}^{m} u_i\right) + u_m\right]} \;,\; \left\lfloor \frac{1}{k_{max}} \sum_{i=1}^{m-1} u_i \right\rfloor \leq \sum_{i=2}^{m} \ell_i \tag{5.7}$$

Die rechts aufgeführte Nebenbedingung ergibt sich aus dem Sachverhalt, dass nicht mehr als das vollständige Objekt während der Nutzzeit des Objekts der Wurzel geladen werden kann[7].

Es ist zu erkennen, dass nicht nur $\beta$, wie schon mit den vorherigen Untersuchungen bestätigt, sondern auch $k_{max}$ in die Kompetitivität des Online-Verfahrens eingeht.

Zur weiteren Vereinfachung wird angenommen, dass die Nutzzeit $u_m$ und die Ladezeit $\ell_1$ Null betragen.[8]

$$\frac{D_{ON}(s_1^m)}{D_{OPT}(s_1^m)} \leq \frac{(1+\beta)\sum_{i=1}^{m-1} u_i - \left\lfloor \frac{1}{k_{max}} \sum_{i=1}^{m-1} u_i \right\rfloor}{max\left(\sum_{i=1}^{m-1} u_i, \beta \sum_{i=1}^{m-1} u_i\right)} \;,\; \left\lfloor \frac{1}{k_{max}} \sum_{i=1}^{m-1} u_i \right\rfloor \leq \sum_{i=2}^{m} \ell_i \tag{5.8}$$

Im Folgenden wird das Verhalten des Quotienten für die zwei Fälle $\beta > 1$ und $\beta \leq 1$ untersucht.

**Fall 1:** $\beta \geq 1$

Durch Umformung der obigen Gleichung 5.8, im Anhang auf S.210 dargestellt, wird folgender Zusammenhang sichtbar, wobei $z_1$, abhängig von $\beta$, $k_{max}$ und der Summe der Nutzzeiten vernachlässigbar klein ist, das heißt wesentlich kleiner als $\frac{1}{\beta k_{max}}$, ist.[9]

$$\frac{D_{ON}(s_1^m)}{D_{OPT}(s_1^m)} \leq \frac{1}{\beta} + 1 - \frac{1}{\beta k_{max}} + z_1 >> \frac{1}{\beta} + 1 - \frac{1}{\beta k_{max}} \leq 2 \tag{5.9}$$

[6] Siehe Abschnitt 3.5.2 „Komparative Analyse des Online-Modells“, S.86ff. bzw. Gleichung 3.34, S.92

[7] Siehe Bedingung 5.4, S.151

[8] Auch diese Annahme hat nicht zur Folge, dass die obere Schranke des Online-Verfahrens sinkt. Die Zulässigkeit dieser Annahme wurde im Rahmen der Kompetitivitätsanalyse von *MDS* und 2*MDS* bereits diskutiert (Abschnitt 3.5.2, S.86ff.).

[9] Vgl. dazu die Umformung im Anhang auf S. 210.

**Fall 2:** $\beta \leq 1$

Wird Gleichung 5.8 auch für Fall 2 umgeformt[10], indem $\beta \leq 1$ und eine Korrekturfunktion $z_2(k_{max}, \beta)$ in die Gleichung eingesetzt wird, ergibt sich folgender Zusammenhang:

$$\frac{D_{ON}(s_1^m)}{D_{OPT}(s_1^m)} \leq 1 + \beta - z_2(k_{max}, \beta), z_2(k_{max}, \beta) \leq \beta \leq 1, z_2 \approx \frac{1}{k_{max}}, \tag{5.10}$$

wobei gilt: $z_2(k_{max}, \beta) \leq \beta$.[11]

Es ergibt sich folgendes Gesamtbild für die Güte des Online-Verfahrens:

$$\frac{D_{ON}(s_1^m)}{D_{OPT}(s_1^m)} \leq \quad c$$

$$c = \begin{cases} \beta \geq 1 : 1 + \dfrac{1}{\beta} - \dfrac{1}{\beta\, k_{max}} \\ \\ \beta \leq 1 : 1 + \beta - z_2(k_{max}, \beta), 0 < z_2(k_{max}, \beta) \leq \beta, z_2(k_{max}, \beta) \approx \dfrac{1}{k_{max}} \end{cases} \tag{5.11}$$

Somit kann Satz 5.1 bestätigt werden. □

**Bemerkung:** *Das Ergebnis gilt für nicht redundante sowie redundante Navigationsbäume. Würden zu einem Knoten $a_i$ Kindknoten existieren, die zueinander redundant wären, dann kann $k_i$ entsprechend um die Anzahl der zueinander redundanten Knotenpaare der Kindknoten reduziert werden bzw. der Wert $k_i$ drückt nicht die Anzahl der direkten Kindknoten, sondern die Anzahl der unterschiedlichen Knotengruppen, zu welchen die Kindknoten gehören, aus.*

Ein Navigationsbaum, in welchem zu einem Knoten $a_i$ Kindknoten der selben Knotengruppe enthalten sind, kann einfach in einen Baum umgewandelt werden, in welchem alle direkten Kindknoten eines jeden Knotens keine zueinander redundanten Knotenpaare bilden, so dass sich die beiden Probleminstanzen in Bezug auf das Optimierungsergebnis nicht unterscheiden:

Angenommen $a_j$ und $a_i$ sind redundant zueinander und die Wurzelknoten der Teilbäume $T_j = \{a_j, T_{j1}, \ldots, T_{jk}\}$ und $T_i = \{a_i, T_{i1}, \ldots, T_{ik}\}$ sowie $T_j$ und $T_i$ sind Teilbäume des Baums $B = \{a_r, T_j, T_i, \ldots\}$, dann können die Teilbäume $T_j$ und $T_i$ in einem Teilbaum $T' = \{a_j, T_{j1}, \ldots, T_{jk}, T_{i1}, \ldots, T_{ik}\}$ zusammengefasst werden, wobei es unerheblich ist, ob $a_j$ oder $a_i$ Wurzelknoten von $T'$ ist, denn $a_i$ und $a_j$ mit selbem Wurzelknoten beschreiben denselben Aufenthalt im Navigationsbaum.

Die obere Schranke ließe sich so für redundante Navigationsbäume weiter durch die genauere Beschreibung von $k_{max}$ präzisieren.

[10] Siehe Anhang, S.153ff.

[11] Das Ergebnis erscheint plausibel, weil $\frac{D_{ON}}{D_{OPT}} \geq 1$ laut Definition gelten muss. Dies kann aufgrund der Nebenbedingung $\left\lfloor \frac{1}{k_{max}} \sum_{i=1}^{m-1} u_i \right\rfloor \leq \sum_{i=2}^{m} \ell_i = \beta \sum_{i=2}^{m} u_i$ auch gezeigt werden: $z_2(k_{max}, \beta) \leq \frac{\beta \sum_{i=2}^{m} u_i}{\sum_{i=1}^{m-1} u_i} = \beta$

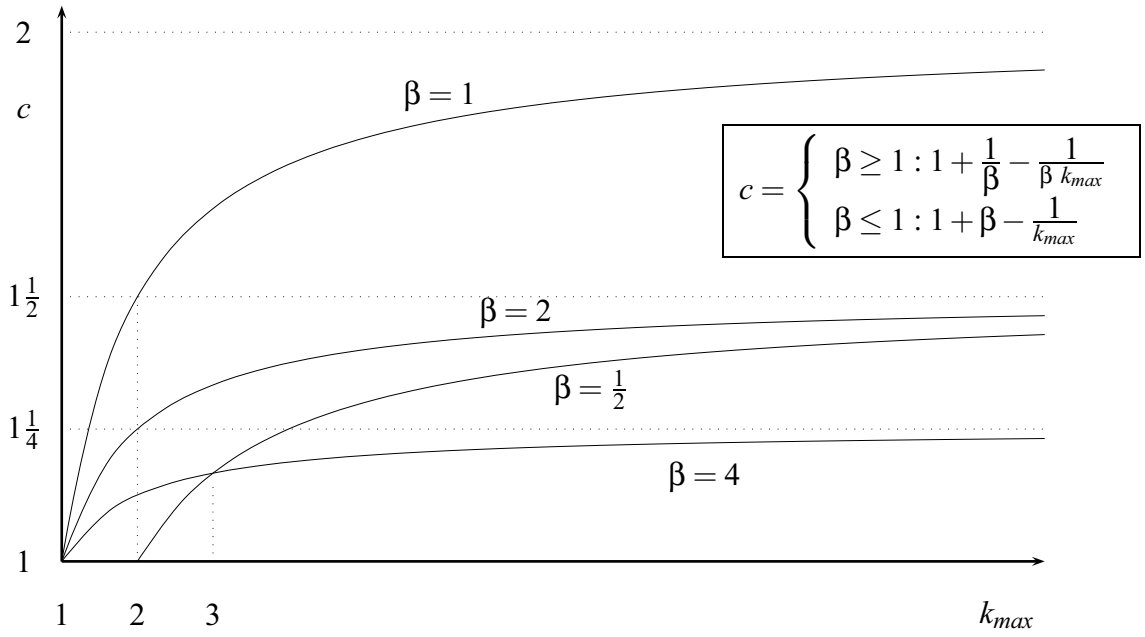



Abbildung 5.1: Funktionaler Zusammenhang zwischen $k_{max}$ und höchstmöglichem Kompetitivitätsfaktor $c$, bei angenommenen β-Werten $\frac{1}{2}, 1, 2, 4$, wobei in der Darstellung die diskret ganzzahligen Ausprägungen von $k_{max}$ nicht berücksichtigt wurden.

Das mit Ungleichung 5.11 erreichte Ergebnis ist nicht überraschend. Im Vergleich zur problemimmanent oberen Schranke des Kompetitivitätsfaktors in Abhängigkeit von β in Ungleichung 3.6, S.93 zeigt sich, dass der dort dargestellte Faktor durch Anwenden der „Einfachen Ladestrategie“ in Abhängigkeit von $k_{max}$ sinkt.

Es erscheint klar, dass $k_i$ bzw. $k_{max}$ wesentlichen Einfluss auf das Anwendungspotential eines solchen Online-Verfahrens haben müssen. Für $k_{max} = 1$ erreicht dieses Verfahren natürlich nahezu dieselben Resultate, wie ein optimales Offline-Verfahren.[12]

Es liegt nahe, zu vermuten, dass der problemimmanente Kompetitivitätsfaktor einer Baumsession von $1\frac{1}{2}$ bei Anwendung der „Einfachen Ladestrategie“ auch um den Term des bestätigten Satzes 5.1, S.150 sinkt.[13] Auf einen Nachweis wird hier jedoch verzichtet.

Jeder Session des Baums kann ein sessionabhängiger Quotient β sowie ein $k_{max}$ zugeordnet werden. Je nach Ausprägung dieser Parameter kann für die jeweils zu betrachtende Session mit unterschiedlichem Anwendungspotential des Online-Verfahrens gerechnet werden (siehe

[12] Die Einschränkung „nahezu“ ist notwendig, weil für Probleminstanzen, welche die Nebenbedingung für 5.5, S.151 verletzen, nicht immer bei $k_{max} = 1$ auch die optimale Offline-Lösung erreicht werden kann, weil nur Objekte direkter Nachbarknoten im Voraus geladen werden.

[13] Der Nachweis der $1\frac{1}{2}$-Kompetitivität von $MDS^*$ für Baumsession ist auf den Seiten 96ff. (Lemma 3.7) erbracht worden. Grundidee des darauf aufsetzenden Beweises für eine weitere Reduktion des Kompetitivitätsfaktors unter Verwendung der „Einfachen Ladestrategie“ ist, die zu betrachtende Session wiederum in zwei Knotenmengen zu zerlegen, so dass im ersten Teil der Session alle Knoten des zweiten Teils der Session vollständig geladen werden. Die Dauer des ersten Sessionteils als Summe aus Nutz- und Ladezeiten der Knoten muss sich um den Faktor $\frac{1}{k_{max}}$ reduzieren, weil während der Nutzdauer der Knoten des ersten Sessionteils mindestens der Anteil $\frac{1}{k_{max}}$ dieser Nutzzeit dazu verwendet wird, Objekte der Knoten des ersten Sessionteils auch im Voraus zu laden.

Abbildung 5.1). Je kleiner $k_{max}$ und je größer $|1-\beta|$ sind, desto dichter ist die obere Schranke des Online-Verfahrens am bestmöglichen Optimierungsresultat unter Kenntnis der Zukunft.

### 5.2.2 $2MDS$ und $2MDS^*$

Das dargestellte Ergebnis mit der Betrachtung der Sessiondauer kann nun einfach auf die zweite zu betrachtende Zielstellung übertragen werden.

Die Nutzzeit des Objekts eines Knotens wird allein zum Laden der jeweiligen direkten Kindknoten verwendet. Die Restladezeit eines Knotens kann gegenüber seiner Ladezeit nur um genau diesen, jeweils zum Laden im Voraus verwendeten Anteil Nutzzeit kleiner sein. Damit kann je Knoten der Session die Sessiondauer eines Ausschnitts der Session, bestehend aus jeweils zwei aufeinander folgenden Entscheidungspunkten $i-1$ und $i$, betrachtet werden. In Anlehnung an die Gleichung 5.3 und 5.5 des vorherigen Abschnitts entstehen folgende Gleichungen

$$\begin{gathered} d_{ON}(s_1^m) = max\left[max_{i=2}^{m}\left(u_i+\ell_i-u_{i-1}\left\lfloor\frac{1}{k_{i-1}}\right\rfloor\right), l_1+u_1\right] \\ \textit{unter der Bedingung} \\ \forall i = 2\ldots m : \ell_i - \left\lfloor u_{i-1}\frac{1}{k_{i-1}}\right\rfloor > 0 \end{gathered} \tag{5.12}$$

Im Gegensatz zu $MDS$ und $MDS^*$ kann jedoch der von einem optimalen Offline-Algorithmus gelieferte Optimalwert für $2MDS$ und $2MDS^*$ genau der Nutzzeit eines Knotens entsprechen.[14] Damit ergibt sich folgende obere Schranke für das Verhältnis des Ergebnisses der einfachen Ladestrategie und eines optimalen Offline-Algorithmus:

$$\begin{gathered} \frac{d_{ON}(s_i^i)}{d_{OPT}(s_i^i)} \leq max\left[max_{i=2}^{m}\left(1+\frac{\ell_i-u_{i-1}\left\lfloor\frac{1}{k_i}\right\rfloor}{u_i}\right), l_1\right] \\ \textit{unter der Bedingung} \\ \forall i = 2\ldots m : \ell_i - \left\lfloor u_{i-1}\frac{1}{k_{i-1}}\right\rfloor > 0 \end{gathered} \tag{5.13}$$

Da das Verhältnis $\frac{l_i}{u_i}$ beliebig groß werden kann, ist die einfache Ladestrategie nicht $c$-kompetitiv für die Zielstellungen $2MDS$ und $2MDS^*$. Dies war nach der Diskussion zur problemimmanenten Schranke der Ziele in Kapitel 3 auch nicht zu erwarten. Dem Autor ist kein Online-Verfahren bekannt, welches eine $c$-kompetitive Lösung liefert. Es liegt die Vermutung sehr nahe, dass ein Verfahren ohne Kenntnis über zukünftige Anfragen nicht existieren kann.[15]

[14] Vgl. Kompetitivitätsanalyse zu $2MDS/2MDS^*$, Bemerkung S.99.

[15] Mit der Diskussion über problemimmanente Schranken wurden bereits Probleminstanzen aufgeführt, welche zu beliebig schlechten Lösungen ohne Kenntnis über zukünfte Anfragen führen. Es stellt sich die Frage, welche zusätzlichen Modellannahmen getroffen werden müssten, um eine kompetitive Lösung erreichen zu können, ohne sich zu weit von den realen Gegebenheiten zu entfernen.

## 5.3 Einsatz der Offline-Verfahren im Online-Modell

In diesem Abschnitt wird die Verwendung der im vorherigen Kapitel entwickelten Optimierungsverfahren für die sequenzbasierten Zielstellungen des Offline-Modells ($2MLS/2MLS^*$, $2MSLS/2MSLS^*$ und $MSLS/MSLS^*$) diskutiert. Zu berücksichtigen ist, dass für die $*$-Zielstellungen keine effizienten Optimierungsverfahren konstruiert werden konnten. Jedoch wird erst einmal angenommen, dass solche auch für diese Verfahren existieren und später auf die Problematik noch genauer eingegangen.

Die die Offline-Zielstellungen optimal lösenden Verfahren sollen zur Entscheidungsfindung in den einzelnen Entscheidungsschritten der Online-Situation eingesetzt werden. Diese treten an die Stelle von ALG in dem für ON beschriebenen Algorithmengerüst S.146. Jeder einzelne Entscheidungsschritt wird als ein separates Offline-Optimierungsproblem aufgefasst. Der Navigationsbaum mit den einzelnen Ladezeiten der Knoten ergibt sich aus dem jeweils zu betrachtenden Teilbaum $T_i$ des Gesamtbaums $B$, dessen Wurzel der im Entscheidungsschritt aktuell angeforderte Knoten ist. Die Ladezeiten aller Knoten des Teilbaums sind gleich den Restladezeiten, sich ergebend aus bereits getroffenen Entscheidungen. Der Teilbaum ist, wie schon mit der einfachen Ladestrategie angesprochen (S.149), aus dem Gesamtbaum zu extrahieren. Für jeden einzelnen Entscheidungsschritt ergeben sich so durch das Offline-Verfahren erzeugte Entscheidungsmatrizen $X_1, \ldots, X_m$ für eine Session $s_1^m$.

Es soll der Einsatz eines optimalen Lösungsverfahrens für die Ziele $2MLS$ und $2MLS^*$ sowie für $2MSLS$ und $2MSLS^*$ zur Lösung von $2MDS$ und $2MDS^*$ untersucht werden sowie der Einsatz eines optimalen Verfahrens für $MSLS$ und $MSLS^*$ für $MDS$ und $MDS^*$.[16]

Vorweg genommen sei, dass sich aus den Nachweisen der beiden nächsten Abschnitte folgendes Bild ergibt:

**Satz 5.2** *Für jedes der oben genannten Verwendungsszenarien der optimalen Offline-Verfahren zur Lösung der Online-Problemstellungen können jeweils Probleminstanzen konstruiert werden, welche zeigen, dass trotz Einsatz der Verfahren in jedem Entscheidungsschritt der Session die jeweils problemimmanent vorhandene, obere Schranke erreicht wird.*

Damit sind die entsprechenden Online-Verfahren unter Verwendung der Offline-Verfahren für die Zielstellungen $2MDS$ und $2MDS^*$ nicht kompetitiv und für $MDS$ und $MDS^*$ strikt 2-kompetitiv. Sie sind aus Sicht einer kompetitiven Analyse, für ein $k_{max} > 1$ einer Session schlechter zu bewerten als die vorher diskutierte einfache Ladestrategie.

**Beweis von Satz 5.2:** Der Satz wird mit den beiden folgenden Abschnitten nachgewiesen.

### 5.3.1 $2MDS$ und $2MDS^*$

Kann für die beiden Ziele $2MDS$ und $2MDS^*$ jeweils in Abhängigkeit vom eingesetzten, optimalen Offline-Verfahren gezeigt werden, dass Session existieren, in welcher die Restladezeit

[16] Letztendlich könnte der Einsatz aller die Offline-Ziele optimal lösenden Verfahren für alle Online-Zielstellungen untersucht werden. Die Anzahl möglicher Kombinationen würde jedoch den Umfang der Arbeit sprengen. So hat sich der Autor auf die Verknüpfung von Zielen beschränkt, welche in ihrer Semantik zueinander nahe liegen.

aller Knoten der Session deren Ladezeit entspricht, dann gilt Satz 5.2 für die beiden Ziele.[17]
Zuerst wird der Einsatz von $2MLS$ und $2MLS^*$ lösenden Verfahren untersucht.

**Lemma 5.3** *Es lassen sich Probleminstanzen für $2MDS$ und $2MDS^*$ konstruieren, welche für den Einsatz optimal lösender Verfahren von $2MLS$ bzw. $2MLS^*$ für ALG in ON zeigen, dass die problemimmanent obere Schranke erreicht wird.*

**Beweis von Lemma 5.3:** Die Probleminstanzen haben folgende Struktur:

Für jeden Teilbaum $T_j = \{a_j, \ldots\}$ des Gesamtbaums $B$ einer solchen Probleminstanz, dessen Wurzelknoten $a_j$ auch in Session $s_1^m \in B, s_1^m = \{a_1, \ldots, a_m\}$ enthalten ist, sowie für die Session selbst und den Navigationsbaum $B$ gilt:

$$\exists s_1^m \in B : \begin{cases} a_i = s_{m-1}^{m-1} \textit{ und } \exists a_j \in T_i = \{a_i, T_1, \ldots T_k\} \textit{ und } a_j \notin s_1^m \\ \textit{und} \\ \nexists a_h \in s_1^{m-1} : red(a_j, a_h) = wahr \\ \textit{und} \\ \forall a_h \in s_1^{m-1} : \ell_h \leq \ell_j - \sum_{\forall a_k \in s_1^{m-1}} u_k \\ \textit{und} \\ l_1 = 0 \textit{ und } u_m = 0 \end{cases} \tag{5.14}$$

Es existiert ein Knoten $a_j$, dessen Ladezeit so groß ist, dass die Restladezeit dieses Knotens die Ladezeiten aller Knoten der Session übersteigt, trotzdem die Nutzzeiten aller Knoten der Session exklusive des letzten Knotens dafür verwendet werden, Teile des Objekts aus $a_j$ zu laden.

Optimale Entscheidungsmatrizen $X_1, \ldots X_m$ für $2MLS$ bzw. $2MLS^*$ haben unter den in 5.14 getroffenen Annahmen zur Folge, dass die Restladezeit aller Knoten der Session $s_1^m$ zu jedem beliebigen Entscheidungszeitpunkt auch deren jeweiliger Ladezeit entspricht. Die Ladezeit eines anderen, nicht in der Session enthaltenen Knotens, dessen Objekt zu jedem Entscheidungszeitpunkt geladen werden kann, ist so groß, dass kein Teil eines Objekts der Knoten aus der Session zu irgendeinem Entscheidungszeitpunkt geladen wird. Weil die Ladezeit des ersten und die Nutzzeit des letzten Knotens der Session von Null gewählt wurden, ergibt sich eine Sessiondauer entsprechend dem Doppelten des Bestmöglichen eines optimalen Offline-Verfahrens. □

[17] Die problemimmanent obere Schranke für $2MDS$ und $2MDS^*$ wurde in Kapitel 3, Abschnitt „Komparative Analyse des Online-Modells“, S.99 diskutiert. Es wurde gezeigt, dass Probleminstanzen existieren, in welchen die Restladezeit eines beliebigen Knotens einer Session, auch nach deren Beendigung, gleich der Ladezeit des Knotens sein kann, wobei Restladezeiten von Null optimal möglich wären.

Auch für 2*MSLS* bzw. 2*MSLS*$^*$ zur Lösung von *MDS* bzw. *MDS*$^*$ kann folgendes adäquate Lemma formuliert werden:

**Lemma 5.4** *Es lassen sich Probleminstanzen für 2MDS und 2MDS$^*$ konstruieren, welche für den Einsatz optimal lösender Verfahren von 2MSLS bzw. 2MSLS$^*$ für* ALG *in* ON *zeigen, dass die problemimmanent obere Schranke erreicht wird.*

**Beweis von Lemma 5.4:** Die Probleminstanzen haben die folgende Struktur.

Für jeden Teilbaum $T_i = \{a_i, T_1, \ldots, T_k\}$ einer solchen Probleminstanz, dessen Wurzelknoten $a_i$ auch in Session $s_1^m \in B, s_1^m = \{a_1, \ldots, a_m\}$ enthalten ist, sowie für die Session selbst und den Navigationsbaum $B$ gilt:

$$\exists \sigma_2^q \in T_i : \left\{ \begin{array}{c} \sigma_2^q \neq s_i^m \\ \textit{und} \\ \sum\limits_{\forall a_j \in s_{i+1}^m} l_j \leq \left( \sum\limits_{\forall a_j \in \sigma_2^q} l_j \right) - u_i \\ \textit{und} \\ \forall (a_i, a_j), a_i \in \sigma_2^q, a_j \in s_{i+1}^m : red(a_i, a_j) = falsch \\ \textit{und} \\ l_1 = 0 \textit{ und } u_m = 0 \end{array} \right\} \tag{5.15}$$

In jedem dieser Teilbäume $T_i$ existiert neben dem im Teilbaum enthaltenen Teilabschnitt $s_i^m$ der Session $s_1^m$ eine weitere Teilsequenz $\sigma_2^q$. Die Summe der Ladezeiten aller Knoten dieser Teilsequenz ist jeweils so groß, dass die Summe der Ladezeiten abzüglich der Nutzzeit des Objekts im Wurzelknoten, die Summe der Ladezeiten aller Knoten im Teilabschnitt $s_{i+1}^m$ übersteigt.[18]

Die von Lösungsverfahren für 2*MSLS* bzw. 2*MSLS*$^*$ getroffenen Entscheidungen sind der Art, dass die Nutzzeit der Objekte aus Knoten des Sessionausschnitts $s_1^{m-1}$ nur zum Laden von Objekten verwendet wird, die in Knoten enthalten sind, welche nicht der Session $s_1^m$ angehören. So unterscheidet sich zu keinem Entscheidungszeitpunkt der Wert der Restladezeit vom Wert der Ladezeit eines jeden Knotens der Session.

Weil Ladezeit des ersten und Nutzzeit des letzten Knotens der Session Null sind, ergibt sich eine Sessiondauer entsprechend dem Doppelten des Bestmöglichen eines optimalen Offline-Verfahrens. □

[18] Die Ladezeit des Wurzelknotens ist nicht zu berücksichtigen, weil laut Modellannahmen mit Beginn dessen Nutzung, die Restladezeit Null betragen muss. Zusätzlich beträgt die Ladezeit des ersten Knotens der Session Null. Damit werden nur $\sigma_2^q$ und $s_{i+1}^m$ betrachtet.

### 5.3.2 *MDS* und *MDS*$^*$

Auch für ein Lösungsverfahren *MSLS* bzw. *MSLS*$^*$ zur Lösung von *MDS* bzw. *MDS*$^*$ können Probleminstanzen formuliert werden, die die problemimmanent obere Schranke erreichen. Diese Schranke ist jedoch 2-kompetitiv.

**Lemma 5.5** *Es lassen sich Probleminstanzen für MDS und MDS*$^*$ *konstruieren, welche für den Einsatz optimal lösender Verfahren von MSLS bzw. MSLS*$^*$ *für* ALG *in* ON *zeigen, dass die problemimmanent obere Schranke, entsprechend einem Kompetitivitätsfaktor von 2, erreicht wird.*

**Beweis von Lemma 5.5:** Die Probleminstanzen haben die folgende Struktur.

Für einen Navigationsbaum $B$, bestehend aus den Teilbäumen $T_i = \{a_i, T_1, \ldots, T_k\}$ gilt:

$$\forall a_i \in s_1^{m-1} = \{a_1, \ldots, a_{m-1}\} : \left\{ \begin{array}{c} T_i = \{a_i, T_h, \ldots, T_j\} \; und \; a_h \in T_h \; und \; a_j \in T_j = \{a_j, \ldots\} \\ und \\ a_h = s_{i+1}^{i+1} \\ und \\ \forall a_i \in s_1^m : red(a_i, a_j) = falsch \\ und \\ l_j \geq u_i \\ und \\ AnzahlKnoten(T_j) > AnzahlKnoten(T_h) \\ l_1 = 0 \; und \; u_m = 0 \end{array} \right. , \tag{5.16}$$

wobei die Funktion $AnzahlKnoten(T_i)$ die Anzahl der in einem Teilbaum enthaltenen Knoten liefert.

Die Probleminstanzen entsprechen genau der Struktur, dass in jedem Teilbaum $T_i$ mit Wurzelknoten $a_i$ eines Knotens aus der Session $s_1^m$ jeweils ein spezieller Teilbaum $T_j$ existiert, der mehr Knoten enthält, als ein anderer auch in $T_i$ enthaltener Teilbaum $T_h$. Zusätzlich ist die Ladezeit des Wurzelknotens des Baums $T_j$ so groß, dass die Nutzzeit des Wurzelknotens von $T_i$ vollständig zum Laden des Objekts dieses Wurzelknotens von $T_j$ verwendet wird. Das jeweils in der Nutzzeit des Wurzelknotens von $T_i$ im Voraus geladene Objekt ist in keinem der Knoten der Sequenz $s_1^m$ enthalten. Daraus folgt, dass ein Lösungsverfahren für *MSLS* bzw. *MSLS*$^*$ zu keinem der Entscheidungsschritte $t(s_1^1), \ldots . t(s_m^m)$ optimale Entscheidungsmatrizen $X_1, \ldots, X_m$

generiert, welche dazu führen, dass die Restladezeit eines der Knoten aus $s_1^m$ zum Zeitpunkt der Anfrage kleiner als seine Ladezeit ist.

Weil Ladezeit des ersten und Nutzzeit des letzten Knotens der Session Null sind, ergibt sich eine Sessiondauer entsprechend dem Doppelten des Bestmöglichen eines optimalen Offline-Verfahrens.

Mit der Bestätigung des dritten Lemmas ist auch Satz 5.2, S.156 zu bestätigen. □

### 5.3.3 Ergebnisse der Analyse

Hier zeigen sich die Grenzen der kompetitiven Analyse sowie der Offline-Verfahren. Es ist zu vermuten, dass der Einsatz anderer, die Ziele des Offline-Modells optimal lösender Verfahren, zum selben Ergebnis führt. Ursache hierfür ist die Annahme von Unkenntnis über das Eintreten zukünftiger Anfragen. Mit der Kompetitivitätsanalyse wird genau die Unkenntnis bzw. Kenntnis von Informationen bezüglich zukünftiger Anfragen bewertet, indem solche Probleminstanzen in die Wertung einfließen, welche durch die Unkenntnis dazu führen, dass das Optimierungsergebnis in Bezug auf das Bestmögliche besonders schlecht ist. Es wird vermutet, dass für jedes hier vorgeschlagene Verfahren des Offline-Modells, eingesetzt in der Online-Situation, eine Probleminstanz konstruiert werden kann, die zur Folge hat, dass die Restladezeiten einer Sequenz von Knoten bzw. eines Knotens genau deren Ladezeit entsprechen, trotzdem ein in dem jeweiligen Entscheidungszeitpunkt optimales Offline-Verfahren eingesetzt wurde.[19]

Die einfache Ladestrategie ist zwar den Offline-Verfahren im Einsatz des Online-Modells überlegen, das wird aber durch folgenden Sachverhalt relativiert.

Die einzelnen Entscheidungsschritte des Online-Modells beziehen sich jeweils immer nur auf Informationen eines Teilbaums des gesamten Navigationsbaums $B = \{a_r, \ldots\}$. Diese Teilbäume $T_i$ haben für jeden Knoten $a_i \in s_1^m = \{a_1, \ldots, a_m\}$ folgende Eigenschaften:

- $T_i = \{a_i, \ldots\}$
- $T_i \in B$
- $\exists(s_1^i \in B \text{ und } s_1^1 = a_r \text{ und } s_i^i = a_i)$

Wird nun angenommen, dass die Entscheidungen in jedem einzelnen Entscheidungsschritt $t(s_i^i)$ des Online-Modells für $T_i$ und $X_1, \ldots, X_{i-1}$ für eine Session $s_1^m = \{a_1, \ldots, a_m\}$ mittels eines Verfahrens getroffen werden, welches eine der Zielstellungen des Offline-Modells optimal löst, dann gilt:

$$OPT(T_1, X_1) \geq OPT(T_2, X_1, X_2) \geq \ldots \geq OPT(T_m, X_1, \ldots, X_m) \tag{5.17}$$

Da $T_1 = B$ ist, muss auch gelten:

$$OPT(B) \geq OPT(T_m, X_1, \ldots, X_m) \tag{5.18}$$

[19] Weil es sich um ein rein deterministische Verfahren handelt, ist es besonders einfach, solche Probleminstanzen zu konstruieren.

Das heißt, bei Einsatz eines der optimalen Offline-Verfahren für ALG in ON wird keinesfalls das Ergebnis der optimalen Offline-Lösung für den Gesamtbaum überschritten. Dies kann die einfache Ladestrategie nicht garantieren.

Unter Berücksichtigung der Entwurfsentscheidungen für Navigationsbaum bzw. Hypermedium ist das als besonders interessanter Umstand zu bewerten, denn Entwurfsentscheidungen können deshalb aus Optimierungsergebnissen der vorgestellten Offline-Verfahren abgeleitet werden.[20] Die durch die Offline-Verfahren ermittelten Restladezeiten für die Knoten werden jedoch nur dann erreicht, wenn entweder die von den Verfahren festgelegten Ladeentscheidungen auch im Betrieb getroffen werden oder wenn ein Online-Verfahren eingesetzt wird, welches trotz Abweichen von den Ladeentscheidungen des Offline-Verfahrens diese Restladezeiten garantieren kann und bestenfalls niedrigere Restladezeiten erreicht.[21]

Aufgrund der prognostizierbaren Ergebnisse der einfachen Ladestrategie können zwar auch Entwurfsentscheidungen des Navigationsbaums unterstützt werden, aber Ziele wie die Minimierung der maximalen Dauer einer Session oder eines Aufenthaltknotens fließen nicht in die Entscheidung ein, spielen jedoch für Lernanwendungen eine herausragende Rolle. Eine große Zahl von inhaltsbasierten Prefetchingverfahren bedienen sich dem mit der „Einfachen Ladestrategie“ verfolgten Ansatz, jeweils immer nur solche Datenobjekte im Voraus zu laden, auf welche Links der aktuell genutzten Datenobjekte des Hypermediums verweisen.[22] Diese Verfahren sind für Lernanwendungen nur bedingt einsetzbar.

Zusätzlich muss bedacht werden, dass für redundante Navigationsbäume nur approximierende Verfahren bezüglich der Zielstellungen des Offline-Modells verfügbar sind. Da die Approximationsgüte erheblich von der Zahl der Knoten des Baums beeinflusst wird, ist für besonders hohe Navigationsbäume damit zu rechnen, dass die Lösung erheblich vom Bestmöglichen abweicht.

### 5.3.4 Eigenschaft des „Späten Ladens“

Zum Schluss dieses Kapitels sei noch auf eine Eigenschaft der Offline-Verfahren für nicht redundante Navigationsbäume hingewiesen, welche den Einsatz dieser Verfahren im Rahmen der Online-Zielstellungen besonders interessant macht. Diese Eigenschaft des so genannten „Späten Ladens“ hebt die aus den Offline-Verfahren konstruierten Online-Verfahren von einer Vielzahl anderer, in der Literatur vorgestellter Prefetchingverfahren ab. Sie resultiert aus der R-Optimalität der Verfahren und hat positive Auswirkungen auf die zu erwartenden, negativen Seiteneffekte des Prefetchingverfahrens, welches sich in den einzelnen Entscheidungsschritten der R-optimalen Offline-Lösungen bedient.

**Definition:** Einem Verfahren des Offline-Modells, welches eine Entscheidungsmatrix $X$ mit folgender Eigenschaft erzeugt, kann die Eigenschaft des „Späten Ladens“ zugeschrieben werden:

Für die Ladeentscheidungen $x \in X$ aller Knoten $a_1, \ldots, a_m$ einer beliebigen Sequenz $\sigma_1^m$ des Navigationsbaums $B$ gilt:

[20] Hierzu genauer im folgenden Kapitel, welches sich mit der Auswertung der Entscheidungsmatrizen für Entscheidungen in der Phase des Entwurfs und Betriebs des Hypermediums beschäftigt.

[21] Ein solches Verfahren konnte nicht konstruiert werden.

[22] Diese Verfahren laden die Datenobjekte vollständig im Voraus, so dass sie damit auch den hier nachgewiesenen Kompetitivitätsfaktor für die „Einfache Ladestrategie“ überschreiten. Solche Verfahren werden zum Beispiel in [Kle99] und [IX00] dargestellt und im Gegensatz zu den Erkenntnissen des Autors als sehr effektiv für WWW-Anwendungen beschrieben.

$$\forall \sigma \in B : \forall (a_i, a_j, i < j) \in \sigma_1^m : \begin{cases} \sum_{h=1}^{n} x_{ih} = u_i \\ \quad oder \\ \sum_{h=i}^{j-1} x_{hj} = l_j \end{cases} \tag{5.19}$$

Entweder die Nutzzeit des Knotens $a_i$ wird vollständig zum Laden von Objekten im Voraus verwendet oder das Objekt aus Knoten $a_j$ wird während der Nutzzeiten aller Knoten des Weges von $a_i$ nach $a_j$ vollständig im Voraus geladen.

**Bemerkung:** *Die einzelnen Offline-Verfahren zur Konstruktion R-optimaler Entscheidungsmatrizen nicht redundanter Navigationsbäume liefern Entscheidungsmatrizen, die der Eigenschaft des „Späten Ladens“ entsprechen.*

Die einzelnen Ladeentscheidungen $x_{hj}, h = 1, \ldots, m$ können jeweils als Ladeentscheidungen im Zeitablauf einer Session $s_1^m$ für das Objekt des Knotens $a_h$ und die einzelnen Entscheidungsschritte zu den Zeitpunkten $t(s_h^h), h = 1 \ldots m$ betrachtet werden. In diesem Zeitablauf wird mit den R-optimal bestimmten Entscheidungsmatrizen der Teilbäume erst dann und nicht früher (also spät) ein Objekt der Sequenz bzw. ein Teil dessen im Voraus geladen, wenn es in den jeweils folgenden Entscheidungsschritten $(t(s_{h+1}^{h+1}), \ldots, t(s_m^m))$ unmöglich wäre, die Restladezeit dieses und aller anderen Knoten entsprechend der R-optimalen Entscheidungsmatrix zu erreichen, ohne dass die Ladeentscheidungen in $t(s_h^h)$ getroffen worden wären.

Trotz der Unsicherheitssituation, welche Sequenz des Baums jeweils eine Session eines Nutzers abbildet, kann mittels R-optimaler Entscheidungsmatrizen der Anteil im Voraus geladener, aber im Nachhinein nicht genutzter Daten drastisch gesenkt und die angestrebten Restladezeiten der Offline-Lösung garantiert werden. Diese Eigenschaft des „Späten Ladens“ ist für die Online-Verfahren von besonderer Bedeutung. Die Nutzzeit der Knoten wird nicht „blind“ zum Laden im Voraus verwendet. „Blind“ heißt, dass diese jeweils vollständig dazu verwendet wird, alles das zu laden, was in dieser Zeit möglich wäre und verfügbar ist, jedoch zum Erreichen der Zielstellung nicht unbedingt nötig ist.[23] Zwar ist es aufgrund der Unsicherheit in der Online-Situation nahezu unmöglich, zu keinem Zeitpunkt Objekte zu laden, welche zukünftig nicht genutzt werden, jedoch sollten zu diesem Zeitpunkt nur Ladeentscheidungen für solche Objekte getroffen werden, welche auch zwingend notwendig sind, um die festgelegten Verzögerungen nicht zu überschreiten.

**Bemerkung:** *Die „Einfache Ladestrategie“ setzt auch die Eigenschaft des „Späten Ladens“ um.*

Zwar wird, wie bereits angemerkt, mit der „Einfachen Ladestrategie“ in keiner Weise eines der Optimierungsziele des Off- oder Online-Modells angestrebt, jedoch wird ein Teil eines Objekts eines Knotens $a_i$ der Session erst dann geladen, wenn das zwingend erforderlich ist, um dessen Restladezeiten jeweils in Höhe von $r_i \leq l_i - \lfloor u_{i-1} \frac{1}{k_{i-1}} \rfloor$ nicht zu überschreiten, wobei $a_{i-1}$ der jeweilige Vorgängerknoten der Session ist.

[23] Siehe Diskussion blinder Prefetchingverfahren S.38ff..

**Bemerkung:** *Die erzeugten Entscheidungsmatrizen des in Abschnitt 4.2.2, S.135ff. angegebenen Verfahrens zur Approximation optimaler Lösungen für redundante sowie nicht redundante Navigationsbäume entsprechen nicht der Eigenschaft des „Späten Ladens“.*

Mit der Formulierung der linearen Programme des Offline-Modells wurde diese Eigenschaft nicht berücksichtigt. Sie ist nicht Gegenstand der Optimierung und ergab sich als „Nebenprodukt“ der effizienten Offline-Verfahren. Um den vorgestellten Approximationsansatz sinnvoll für Online-Verfahren einsetzen zu können, ist ein effizientes Verfahren zu konstruieren, welches eine gegebene, approximierende Entscheidungsmatrix so umformt, dass die Eigenschaft des „Späten Laden“ erfüllt ist.

**Satz 5.6** *Je nach angestrebter Restladezeit eines jeden Knotens kann auf Basis einer gegebenen Entscheidungsmatrix und der damit festgelegten Restladezeiten aller Knoten eine Entscheidungsmatrix konstruiert werden, die die Eigenschaft des „Späten Ladens“ erfüllt.*

Ein solches Verfahren wird im Folgenden kurz skizziert. Auf einen ausführlichen Nachweis der Korrektheit und Effizienz und damit dem Nachweis der Gültigkeit des Satzes, wird aufgrund der untergeordneten Bedeutung verzichtet.

Die Restladezeiten der einzelnen Knoten ergeben sich aus der vorgegebenen Entscheidungsmatrix, welche in eine Matrix umzuwandeln ist, die der Eigenschaft des „Späten Ladens“ entspricht. Die Transformation einer Entscheidungsmatrix ist nicht trivial, weil die möglichen Ladeentscheidungen der Knoten durch die begrenzten Nutzzeiten und festgelegte Verknüpfung mittels Navigationsbaum limitiert werden.

Folgendes effiziente Verfahren liefert für einen redundanten wie nicht redundanten Baum $B$ und dessen Entscheidungsmatrix $X$ in Form der Datenstruktur Baum eine Entscheidungsmatrix, die die Eigenschaft des „Späten Ladens“ erfüllt:

```
1  transformiereMatrix( <-> B:Baum)
2  aL:Liste(Index), Min:Zeit, i,j:Index
3  Beginn
4     falls nicht Blatt?(B)
5        wiederhole für j=1 bis AnzElemente(B.a.x)
6        falls j <> B.a.i
7        Beginn
8           aL=liefereRedundanteKnoten(j,B)
9           Min=liefereMinSummeRestnutzzeit(aL,B)
10          fuer alle TeilBaeume T in B.Teilbaeume
11             VerteileUm(T,aL,Min)
12          für alle i in aL
13             B.a.x[i]=B.a.x[i]-Min
14       Ende
15    fuer alle TeilBaeume T in B.Teilbaeume
16       transformiereMatrix(T,aL,Min)
17 Ende
```

**Alg. 5.4** transformiereMatrix(B)

```
1  verteileUm(<-> B:Baum,
2                -> aL:Liste(Index),  -> Min:Zeit)
3  i:Index
4  Beginn
5     falls B nicht Blatt Beginn
6        für alle i in aL
7           falls nicht inBaum?(B,i) entferne(aL,i)
8        falls nicht leer?(aL) Beginn
9           Rest=Restnutzzeit(B.a.u-Summe(B.a.x)
10          falls Min < Rest Rest=Min
11          für alle i in aL
12             B.a.x[i]=B.a.x[i]-Rest
13          Min=Min-Rest
14          für alle Teilbäume T in B.Teilbäume
15             verteileUm(T,aL,Min)
16       Ende
17    Ende
18 Ende
```

**Alg. 5.5** verteileUm(...)

Die Schnittstellen und Funktionalitäten der genutzten Handlungen werden im Folgenden kurz umrissen und sind im Anhang, S.207 aufgeführt.

Das dargestellte Verfahren transformiereMatrix ermittelt für alle jeweils in B enthaltenen zueinander redundanten Knoten, deren Objekte während der Nutzzeit des Wurzelknotens geladen werden, eine Zeitdauer (Min). Das ist genau die Zeitdauer, während welcher die zueinander redundanten Knoten in der bisher nicht verwendeten Nutzzeit anderer, in B enthaltener Knoten im Voraus geladen werden können. Sie wird ermittelt, indem alle Wege vom Wurzelknoten zu den jeweiligen betreffenden Knoten betrachtet werden und die Summe der nicht verwendeten Nutzzeiten der auf den Wegen befindlichen Knoten bestimmt wird (Zeile 9). Das Minimum dieser Summen entspricht genau der Zeitdauer, um welche diese Knotenmenge nicht während der Nutzung des Wurzelknotens von B im Voraus geladen wurden muss. Auf den ermittelten Wegen zwischen Wurzel und diesen Knoten ist ausreichend bisher nicht zum Laden im Voraus verwendete Nutzzeit verfügbar. Die Ladeentscheidungen auf diesen Wegen sowie des Wurzelknotens werden aufgrund des ermittelten Werts mit den Zeilen 10-13 modifiziert. VerteileUm nimmt diese Modifikation für alle diese Wege vor, indem rekursiv die Ladeentscheidungen der Wurzelknoten in den Teilbäumen von B, in welchen Knoten aus der Knotenmenge existieren, angepasst werden.

Das Verfahren transformiereMatrix wird rekursiv, beginnend mit dem Wurzelknoten des Gesamtbaums, absteigend für alle Teilbäume folgend den Niveaus der jeweiligen Wurzelknoten der Teilbäume aufgerufen, so dass letztendlich eine Entscheidungsmatrix geliefert werden kann, deren Ladeentscheidungen die Eigenschaft des „Späten Ladens“ erfüllen.

Auf einen Korrektheitsnachweis sowie die Bestimmung von Rechenzeit- und Speicherplatzaufwand sei hier verzichtet, da das Verfahren „nur“ eine bekannte Lösung in Hinblick auf Ziele, die so nicht im On- wie Offline-Modell verfolgt werden, anpasst.

Das Verfahren arbeitet mit einem Rechenzeitaufwand von $O(n^4)$, wobei verteileUm einen Rechenzeitaufwand von $O(n^3)$ verursacht und $O(n)$-mal durch transformiereMatrix aufgerufen wird. Die Funktion liefereMinSummeRestnutzzeit kann mit einem Rechenzeitaufwand von $O(n^2)$ realisiert werden.

Die Eigenschaft des „Späten Ladens“ ist wesentlich schwächer als die Forderung nach einer R-optimalen Entscheidungsmatrix. Sie garantiert keine optimale Verwendung der Nutzzeiten in Hinblick auf die jeweilige Zielstellung. Es existiert wahrscheinlich kein Verfahren, mit welchem effizient geprüft werden kann, ob für eine Menge vorgegebener Restladezeiten eine Entscheidungsmatrix existiert, die diese Restladezeiten unter den gegebenen Nebenbedingungen realisiert. Mit einem solchen Verfahren wären die als *NP*-vollständig klassifizierten Optimierungsprobleme effizient lösbar.

Die zu erwartenden Seiteneffekte resultieren aus Entscheidungsmatrizen, die der Eigenschaft des „Späten Ladens“ genügen. Sie sind jedoch wesentlich geringer, als wenn scheinbar willkürlich Ladeentscheidungen in Hinblick auf die Zielstellung getroffen werden würden.

## 5.4 Diskussion der Online-Verfahren

Die konstruierten Offline-Verfahren lassen sich auch zur Konstruktion von Online-Algorithmen sinnvoll einsetzen. Jedoch allein unter Gesichtspunkten der vergleichenden Analyse mittels

Kompetitivitätsanalyse sind die so erzeugten Optimierungsergebnisse schlechter als die einer „Einfachen Ladestrategie“, ein sehr einfaches und leicht zu implementierendes Verfahren, zu bewerten.

Hinzu kommt die Problematik, dass für redundante Navigationsbäume nur approximierende Offline-Verfahren zur Verfügung stehen. Die approximierte Lösung könnte mittels Heuristiken weiter verbessert werden, jedoch ist die verfügbare Rechenzeit stark begrenzt. Die Rechenzeit ist in der Online-Situation durch die Restladezeit des durch die lokale Anwendungskomponente angeforderten Objekts, über dessen Verwendung der Nutzzeit zu entscheiden ist, beschränkt. Wird die für die Berechnung der Entscheidung zur Verfügung stehende Restladezeit überschritten, steigt zwangsläufig die Aufenthaltsdauer eines Knotens über $r_i + u_i$, weil in der Zeitdauer $r_i$ nicht über die Verwendung von $u_i$ entschieden werden konnte. In Abhängigkeit vom verwendeten Approximationsverfahren, der Größe des Navigationsbaums sowie der verfügbaren Rechenzeit ist die benötigte Zeit zur Entscheidung über die Verwendung der Nutzzeit zu berücksichtigen. In diesem Kapitel wurde die Dauer der Entscheidung auf einen Zeitpunkt reduziert. Weitere experimentell angelegte Arbeiten sind zur Abschätzung der benötigten Zeitdauer notwendig.[24] Innerhalb des Modells kann notwendige Entscheidungszeit einfach berücksichtigt werden, indem die Nutzzeiten der Knoten um den entsprechend benötigten Wert reduziert werden. Übersteigt die von einem Entscheidungsverfahren benötigte Rechenzeit die Nutzzeit eines Knotens, dann können vom Verfahren Entscheidungen geliefert werden, denen nicht oder nur teilweise sinnvoll für diesen Knoten entsprochen werden kann. Verfahren zur Abschätzung der höchstens benötigten Rechenzeit je Knoten im Baum sind wünschenswert.

Alles in allem sind diese Ergebnisse des Kapitels ernüchternd. Eine wesentliche Hoffnung des Autors in der Modellformulierung bestand darin, nicht „nur“ die besonderen Anforderungen verteilter Lernanwendungen im Entwurf unterstützen zu können,[25] sondern auch Verfahren unter Berücksichtigung der Nutzzeiten in den Aufenthaltsknoten für den Betrieb von Lernanwendungen zu finden, deren Kompetitivität sich wesentlich günstiger entwickelt, als das mit den schon diskutierten problemimmanent oberen Schranken basierend auf der Modellformulierung gegeben ist. Solche Verfahren konnten nicht konstruiert werden. Die „Einfache Ladestrategie“ senkt zwar die problemimmanent obere Schranke für die Online-Ziele, aber nur in Abhängigkeit vom Parameter $k_{max}$ in relativ geringem Umfang. Es erscheint so, dass die Offline-Verfahren nur allein in Offline-Situationen sinnvoll einsetzbar sind. Jedoch muss bedacht werden, dass mit dem Einsatz der in dieser Arbeit entworfenen Offline-Verfahren in der Online-Situation im voraus bekannte Restladezeiten garantiert werden können. Diese Restladezeiten sind vor dem Betrieb der Anwendung bekannt. Sie können im Entwurf berücksichtigt werden und mit der Spezifikation verglichen werden. Gegebenenfalls ist der Entwurf zu überarbeiten bzw. die Umsetzbarkeit der Spezifikation zu diskutieren. Ein niedriger Kompetitivitätsfaktor ist zwar wünschenswert, jedoch nicht zwingend kritisch für das Anwendungsverhalten. Viel wichtiger ist die Kenntnis bereits in der Entwurfsphase über das Antwortzeitverhalten bei Einsatz von Prefetching-Verfahren.

Es zeigt sich, dass mit der komparativen Analyse auf Basis der Betrachtungen der Kompetiti-

[24] Allein eine Betrachtung der asymptotischen Wachstumsordnungen für Rechenzeitaufwände erscheint nicht ausreichend. Für eine sekundengenaue Abschätzung ist eine wesentlich exaktere Betrachtung unter zusätzlichen Einflussfaktoren notwendig, welche das Off- wie Online-Modell nicht berücksichtigen. Asymptotische Rechenzeitaufwände in Größenordnungen von bis zu $O(n^4)$ sind hier nicht zu vernachlässigen.

[25] Dazu im nächsten Teil der Arbeit mehr.

vitätsfaktoren, sich ergebend aus Online- und optimalen Offline-Verfahren, für die dargestellte Problemsituation nur begrenzt sinnvolle Aussagen treffen lassen. Andere Analysetechniken, wie zum Beispiel die Betrachtung durchschnittlicher Fälle oder in der Literatur häufig betrachteter Fälle, basierend auf empirisch Untersuchungen, sind aufgrund der Unkenntnis über die Verteilung zukünftiger Anfragen hier nicht oder nur geringfügig hilfreich. Auch hätten die Aussagen solcher Ansätze nur begrenzten Wert auf die ganz konkret beobachtbare Anwendungssituation.[26]

Der durch die „Einfache Ladestrategie“ erreichte Kompetitivitätsfaktor war zu erwarten. Dieser entspricht einem Verfahren, welches keine Kenntnis über die Zukunft hat, außer dass es alle, jeweils als nächstes stellbaren Anfragen der lokalen Anwendungskomponente kennt, wie es für inhaltsbasierte Prefetchingverfahren typisch ist.

Als Nebenprodukt der Analysen wird die für die Online-Situation interessante Eigenschaft des „Späten Ladens“ betrachtet. Von weiterem Interesse wäre eine Abschätzung der Minderung von negativen Seiteneffekten durch die Eigenschaft des „Späten Ladens“. In Abhängigkeit vom Optimierungsziel sind Entscheidungsmatrizen, die der Eigenschaft des „Späten Ladens“ entsprechen, mit solchen zu vergleichen, die der Eigenschaft nicht entsprechen.

[26]Dieses Problem wurde schon im Grundlagenkapitel, Kapitel 2 in Abschnitt 2.3.3, S.41ff. zur Bewertung der Güte von Prefetchingverfahren diskutiert.

# Kapitel 6

# Entscheidungsunterstützung

*„Wenn ein Kapitän nicht weiß,*
*welches Ufer er ansteuern soll,*
*dann ist kein Wind der richtige.“*
*Lucius Annaeus Seneca*

In diesem Kapitel wird ein Ausblick über den möglichen Einsatz der vorgestellten Verfahren in Entwicklung und Betrieb verteilter hypermedialer Lernanwendungen gegeben. Ziel dieses Kapitels ist es, Vorstellungen zu entwickeln, wie die Ansätze in die leistungsorientierte Entwicklung und den Betrieb von zeitkritischen Lernanwendungen einfließen könnten, und welche weiteren Forschungsarbeiten diesbezüglich von Interesse sind. Die Notwendigkeit des leistungsorientierten Vorgehens im Sinne des Software-Performance-Engineering und -Tuning wurde bereits im Kapitel 2, S.21ff. mit den Besonderheiten verteilter Lernanwendungen und S.28ff. mit den speziellen Erfordernissen einer leistungsorientierten Entwicklung ausführlich diskutiert.

Mit dieser Arbeit wird nicht das Ziel verfolgt, die konstruierten Ansätze in die Menge der vielzählig vorgeschlagenen, teilweise sehr stark je nach Anwendungsbezug differierenden Vorgehensmodelle im Rahmen des Web Engineerings bzw. der Entwicklung von Lernsoftware für Hypermedien einzubetten. Hier sei dazu nur kritisch angemerkt, dass ein gezielt leistungsorientierter Entwurf von jenen Modellen nicht explizit unterstützt wird. Das klassische Vorgehen, das Performance-Tuning während der Testaktivitäten und des Betriebs einzusetzen, ist dominant (vgl. [ELW04], S.242ff.). Frühzeitige Ausrichtung der Entwicklungsaktivitäten an Leistungskriterien der Anwendungen spielt keine bzw. eine stark untergeordnete Rolle. *Lang* erkennt für die Entwicklung von Hypermedien, dass aufgrund der überwiegend evolutionär getriebenen Entwicklung ein Mangel an entscheidungsunterstützenden Werkzeugen existiert; insbesondere in der Entwurfsphase besteht ein Mangel an Werkzeugen, welche zeitnah Konsequenzen der Entwurfsentscheidungen sowie Einstellungen von Betriebsparametern darstellen und auch vorzeitig abschätzen können (vgl. [Lan02]). Aufwändige Simulationen und analytische Betrachtungen sind zwar in Hinblick auf die zu erwartende Qualität wünschenswert, aber aufgrund der schlechten Handhabbarkeit solcher Methoden in der praktischen Entwicklungsarbeit nicht sehr erfolgsversprechend (vgl. [Lan04] und [GM01]).

*Bulterman* beschreibt einen für die vorliegende Arbeit sehr interessant erscheinenden und sehr einfach gehaltenen Ansatz zur Visualisierung der zu erwartenden Performance-Probleme für

den rechnergestützten Entwurf linear strukturierter Präsentationssequenzen, deren darzustellende Inhalte über ein verteiltes Rechnernetz ausgeliefert werden. Abhängig von der zur Verfügung stehenden Bandbreite der Verbindung zwischen Präsentationskomponente und Datenquelle werden Verzögerungen während der Präsentation kalkuliert und für den Entwickler visuell mittels eines so genannten „timeline interface“ aufbereitet dargestellt (vgl. [Bul04]). Über diese Benutzerschnittstelle kann der Entwickler auch einmal getroffene Segmentierungsentscheidungen der Präsentationsinhalte schnell überarbeiten und bekommt zeitnah eine Rückkopplung zu seinen Entwurfsentscheidungen bezüglich zu erwartender zeitlicher Verzögerungen geliefert.[1]

Die in dieser Arbeit vorgestellte Problematik stellt sich jedoch wesentlich komplizierter dar als das Abschätzen von Verzögerungen in einer verteilt realisierten, linear strukturierten Präsentation. Die Interpretation der von den Optimierungsverfahren gelieferten Entscheidungsmatrizen und daraus ableitbaren Prefetchingentscheidungen sowie Verzögerungen ist komplexer. Von den Ergebnissen ist nicht nur die Segmentierung, sondern auch die Sequenzierung der Lerninhalte betroffen. Technisch orientierte Anforderungen an den Entwurfsprozess von zu präsentierenden Lerninhalten spielen im Regelfall eine untergeordnete Rolle.[2] Für den Entwurf verteilter Lernanwendungen sind unbedingt Kompromissentscheidungen zwischen dem nach der didaktischen Konzeption noch Sinnvollen und dem technisch Machbaren zu unterstützen.

Dieses Kapitel beschäftigt sich mit der Interpretation der Optimierungsergebnisse zur Steuerung des Entwurfsprozesses sowie mit der Einbettung eines Online-Prefetchingverfahrens in eine Lernanwendung zu deren Betrieb. Die Ergebnisse könnten in einem ähnlichen wie von *Bulterman* beschriebenen Werkzeug münden, welches nicht nur entwurfs- sondern auch betriebsbedingte Entscheidungen stützt. Das Kapitel ist entsprechend in zwei Teile gegliedert.

## 6.1 Entwurf

Für den Entwurf des Hypermediums werden mit Kapitel 3 zur „Modellformulierung“ zwei Entscheidungsfelder dargestellt. Die Entscheidung über den notwendigen Einsatz eines Prefetchingverfahrens, die so genannte „Verwendungsentscheidung“, und die Entscheidung über gezielte Anpassung von Eigenschaften des Hypermediums, welche laut den hier dargestellten Modellen Berücksichtigung in den Prefetchingverfahren finden - die so genannten „Entwurfsentscheidungen“ bezüglich des Hypermediums unter Berücksichtigung des angewendeten Prefetchingverfahrens.

Der Entwurf wird von zwei Aktivitäten geprägt: der Segmentierung und der Sequenzierung. Die Segmentierung der rechnergestützt zu präsentierenden Lerninhalte liefert Präsentationseinheiten, das heißt die Objekte der Aufenthaltsknoten des Navigationsbaums. Die Sequenzierung liefert die Reihenfolge, in welcher die Präsentationseinheiten angefragt werden können. Beide Aktivitäten wirken auf Lade- und Nutzzeiten der Aufenthaltsknoten sowie auf die im Navigati-

[1] Dieses Werkzeug wurde zur Unterstützung der Bearbeitung von Video-, Audio- und/oder Bildsequenzen für SMIL-Präsentationen entwickelt (SMIL - Synchronized Multimedia Integration Language, vgl. zu SMIL [BGJ+05]).

[2] Beispielsweise beschreiben *Niegemann* und weitere Autoren in [NHHM+04], S.51ff. ausführlich Vorgehensweisen zur Entwicklung von Lernanwendungen. Technische Probleme in Hinblick auf die Antwortzeiten werden erkannt, aber nur am Rande benannt und fließen in keiner Weise in das Vorgehensmodell mit ein.

onsbaum enthaltenen Knotensequenzen.[3]

In Abb. 6.1 ist die Wirkung von Segmentierung und Sequenzierung auf die Eigenschaften des Navigationsbaums zusammengefasst. Die Ladezeiten der Objekte werden nicht von Sequenzierungsentscheidungen betroffen. Sie ergeben sich aus der Segmentierung der Lerninhalte sowie den technischen Rahmenbedingungen, insbesondere der zur Verfügung stehenden Verbindung zur Übertragung der Daten zwischen den verteilten Anwendungskomponenten. Die Nutzzeiten hingegen resultieren einerseits aus den Segmentierungsentscheidungen sowie andererseits auch aus der Reihenfolge, in welcher die Objekte angefordert werden, den Sequenzierungsentscheidungen.

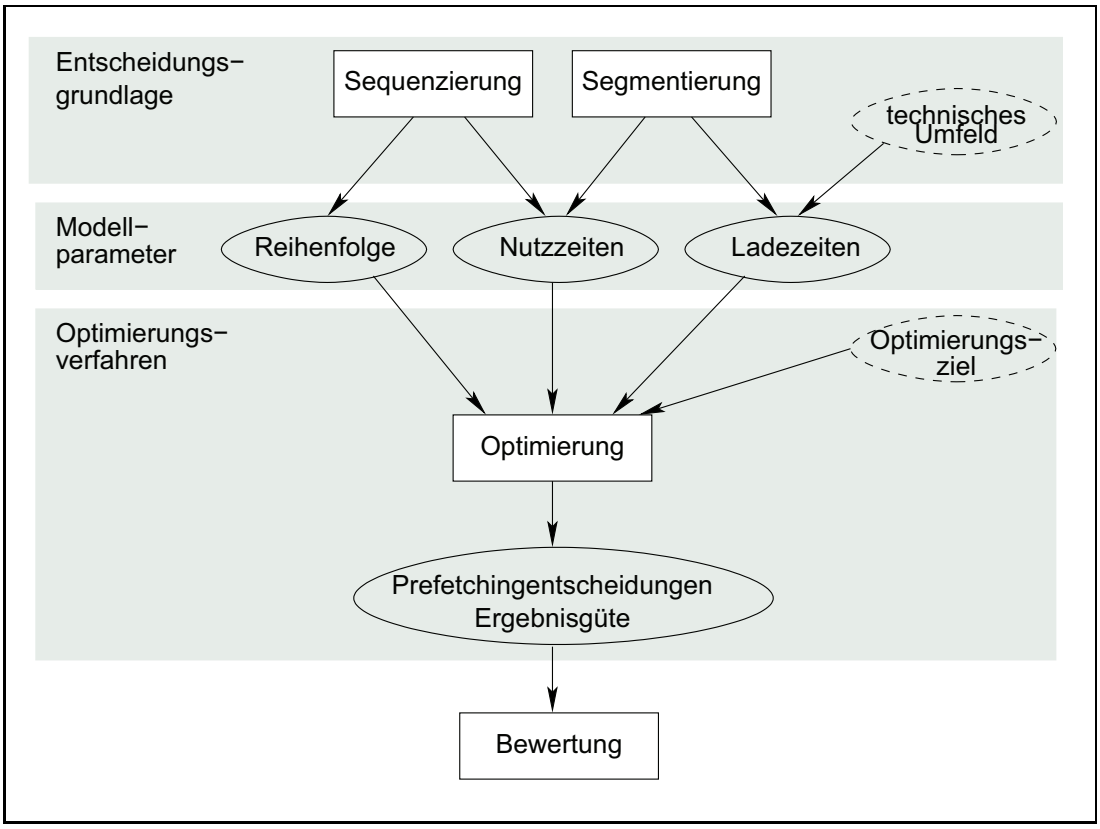



Abbildung 6.1: Wirkung von Segmentierungs- und Sequenzierungsentscheidungen der zu präsentierenden Lerninhalte. Die Aktivitäten wirken auf die wesentlichen Parameter des Off- und Onlinemodells. Sie bestimmen die Eigenschaften des Navigationsbaums.

Segmentierung und Sequenzierung wirken damit je nach verfolgtem Optimierungsziel auf das Bestmögliche sowie im schlechtesten Fall zu Erwartende Optimierungsergebnis der in den vorangegangenen Kapiteln vorgestellten Offline- und Online-Verfahren.[4] Das Ergebnis ist bezüglich der getroffenen Segmentierungs- und Sequenzierungsentscheidungen zu bewerten. Die

[3] In Kapitel 2, S.27ff. sowie Kapitel 3, S.78 und S.57 ff. wurde bereits auf die Segmentierung und Sequenzierung von Lerninhalten als Resultat der navigationsbezogenen und didaktischen Anforderungen eingegangen.

[4] Wie im vorherigen Kapitel ausführlich nachgewiesen, haben Nutz-, Ladezeiten sowie die Struktur des Navigationsbaums wesentlichen Einfluss, auf Optimierungsergebnis in der Online-Entscheidungssituation. Der für eine bestimmte Instanz eines Navigationsbaums geltende Kompetitivitätsfakor der in der vorliegenden Arbeit untersuchten Optimierungsziele, wird von allen drei Parametern maßgeblich beeinflusst.

genauere Betrachtung des Zusammenspiels von Segmentierungs- und Sequenzierungsentscheidungen mit Prefetching wird in den folgenden beiden Abschnitten dargestellt.

### 6.1.1 Entscheidung für bzw. gegen Prefetching

In einem ersten Schritt ist über den Einsatz eines Prefetchingverfahrens zu entscheiden. Dabei ist vordergründig das Antwortzeitverhalten der Lernanwendung ohne Einsatz eines Prefetchingverfahrens zu bewerten. Bevor der Einsatz eines Prefetchingverfahrens in Erwägung gezogen wird, ist folgende Frage zu beantworten:

Kann das spezifizierte Antwortzeitverhalten der Anwendung auch ohne Einsatz eines Prefetchingverfahrens erreicht werden?

Unter der Annahme, dass kein Prefetchingverfahren eingesetzt wird, ist der Navigationsbaum zu prüfen auf:

- die zu erwartenden Ladezeiten der einzelnen Aufenthaltsknoten des Navigationsbaums.

  Gegebenenfalls sind Segmentierungsentscheidungen zu überarbeiten.

- die zu erwartende Lerndauer je Sequenz aller Sequenzen des Navigationsbaums.

  Gegebenenfalls sind Sequenzierungsentscheidungen zu überarbeiten.

Sind die erreichten Resultate nicht zufriedenstellend, ist der Einsatz eines Prefetchingverfahrens in Erwägung zu ziehen. Dabei ist jedoch in einem ersten Schritt das Risiko des Einsatzes eines solchen Verfahrens, insbesondere bezogen auf den zu erwartenden Entwicklungs- und Betriebsaufwand der gesamten Lernanwendung, abzuschätzen. Bei erhöhtem Risiko ist auch eine Anpassung der spezifizierten Anforderungen bzw. der Abbruch der Entwicklung sinnvoll.[5]

Ist der Einsatz eines Verfahrens zur Verbesserung der Antwortzeit der Lernanwendung notwendig, sind aus den Anforderungen der Spezifikation heraus die Ziele eines solchen Verfahrens festzulegen.

Im Weiteren sollen die in dieser Arbeit entwickelten Online-Verfahren betrachtet werden, die sich auf die Offline-Verfahren zur Entscheidungsfindung in den einzelnen Entscheidungsschritten stützen. Sie sind dadurch gekennzeichnet, dass für jedes dieser Verfahren und gegebenen Navigationsbaum eine obere Schranke für das Optimierungsergebnis mit Hilfe des zum Online-Verfahren korrespondierenden Offline-Verfahrens angegeben werden kann.[6] Diese wird im Regelfall die mit der Kompetitivitätsanalyse ermittelte, problemimmanente, von der Struktur des Navigationsbaums abhängige, obere Schranke unterschreiten. Aus der oberen Schranke ergibt sich eine Mindestleistung des gewählten Prefetchingverfahrens bzw. ein garantiertes Antwortzeitverhalten der Lernanwendung. Diese Schranke ist mit den spezifizierten Anforderungen an das Antwortzeitverhalten abzugleichen. Wird dem nicht entsprochen, ist zu prüfen, ob im Rahmen der Sequenzierung und Segmentierung das Optimierungsergebnis beeinflusst werden kann,

[5] Spätestens hier hat ein gezieltes Risikomanagement zu greifen. Performance-Risiken sind mit dem Entwicklungsaufwand abzuwägen und zu kontrollieren.

[6] Siehe Diskussion Einsatz Offline-Verfahren zur Abschätzung der Güte von Online-Verfahren S.160.

so dass die Spezifikation des Antwortzeitverhaltens[7] unter den Anforderungen an die Struktur der Lerninhalte gemäß dem didaktischen Konzept erfüllt werden kann.

Mit der Beurteilung des Anwendungspotentials eines Prefetchingverfahrens ist jedoch nicht nur die bestimmbare obere Schranke, sondern auch deren Nähe zur unteren Schranke von Interesse. Diese kann einfach mit Hilfe des Ergebnisses des zum Online-Verfahren korrespondieren optimalen Offline-Verfahrens bestimmt werden. Eine große Differenz zwischen oberer und unterer Schranke lässt vermuten, dass das Prefetchingverfahren im praktischen Betrieb wahrscheinlich wesentlich bessere Ergebnisse liefern wird, als das nach der Mindestleistung zu erwarten ist.

Aus dieser Sachlage ergibt sich, dass über die Verwendung eines bestimmten Prefetchingverfahrens ohne weiteres entschieden werden kann. Segmentierung und Sequenzierung beeinflussen das Optimierungsergebnis nachhaltig. Es ist in einen entsprechenden Entwurfsprozess einzutreten, in welchem Segmentierungs- als auch Sequenzierungsentscheidungen je nach Optimierungsergebnis des entsprechend der verfolgten Zielstellung verwendeten Prefetchingverfahrens zu überarbeiten sind.

### 6.1.2 Performance Engineering

Ist ein Prefetchingverfahren des Online-Modells zu bewerten, welches sich in den einzelnen Entscheidungsschritten eines Offline-Verfahrens bedient, dann können anhand der Entscheidungsmatrix des Offline-Verfahrens für den gesamten Navigationsbaum, das heißt anhand des Ergebnisses des ersten Entscheidungsschritts der Online-Situation, die Eigenschaften des Navigationsbaums schon in der Entwurfsphase geprüft werden.

Dabei soll hier zwischen zwei möglichen Problemstellungen unterschieden werden:

- die Restladezeit einzelner Knoten des Navigationsbaums entspricht nicht den Anforderungen oder
- der durch die gesamte Entscheidungsmatrix repräsentierte Zielfunktionswert entspricht nicht den Anforderungen an das Antwortzeitverhalten.

Ziel ist es, den Navigationsbaum so anzupassen, dass die durch das Optimierungsverfahren gelieferte Entscheidungsmatrix den Anforderungen entspricht.

Eine Unterscheidung zwischen diesen beiden Problemstellungen erscheint sinnvoll, weil mit den bisher hier modellierten Zielfunktionen des Off- wie Online-Modells jeweils nur das zweitgenannte Problem verfolgt werden kann. Praktisch ist aber davon auszugehen, dass Aufenthaltsdauern von Sequenzen wie auch einzelner Knoten von Interesse sind und die Minimierung maximal möglicher Latenzzeiten nicht kompromisslos über Latenzzeiten einzelner Aufenthaltsknoten gestellt werden kann. Es sind zum Beispiel spezifizierte Anforderungen der Art denkbar, dass zwar die Minimierung der maximal möglichen Restladezeit ($2MLB/2MLB^*$) primär verfolgtes Ziel ist, diese jedoch unter der Maßgabe erreicht werden soll, dass die Restladezeit bei keinem Knotens einen vorgegebenen Wert überschreiten darf. Das nach $2MLB^*$ optimale bzw. approximierte Ergebnis wäre anzupassen.

[7] Anforderungen an Lerndauer und Latenzzeiten einzelner Aufenthaltsknoten

Nachträgliche Anpassungen der Eigenschaften des Navigationsbaums bzw. vom Entwickler festgelegte Vorgaben für Ladeentscheidungen können erheblich den praktischen Nutzen des Optimierungsergebnisses erhöhen. Es zeigt sich jedoch in den folgenden Abschnitten, dass sich eine Modifikation des Baums mit dem Ziel, einen bestimmten Optimalwert zu erreichen, äußerst komplex gestalten kann. Das Festlegen einzelner Ladeentscheidungen zum Mindern der Restladezeit einzelner Knoten erweist sich hingegen als verhältnismäßig einfach.

**Reduktion der Restladezeit eines Knotens**

Eine typische Aufgabe in der Entwurfsphase könnte sein, die Eigenschaften eines Navigationsbaums in Hinblick auf das zu erwartende Antwortzeitverhalten bei Einsatz eines bestimmten Prefetchingverfahrens anzupassen: Es sei das zu erwartende Antwortzeitverhalten für einen Navigationsbaum als Ergebnis eines der Offline-Optimierungsverfahren gegeben. Es liegt in Form einer Entscheidungsmatrix $Y$ mit den Entscheidungen $y_{ik} \in Y$ für einen redundanten oder auch nicht redundanten Navigationsbaum entsprechend der Definition des Offline-Modells in Kapitel 3, Abschnitt 3.3.1 „Das Offline-Modell“, S. 65 vor. Die Restladezeit $r_i$ eines Knotens $a_i$ dieser Matrix, angehörig der Knotengruppe $G_k$ aller zu $a_i$ redundanten Knoten des Gesamtbaums, ist auf einen vorgegeben Wert $r^*$ zu reduzieren, weil für diesen Knoten die Restladezeit des Optimierungsergebnisses nicht die Spezifikation erfüllt.

Grundsätzlich existieren fünf, sich in ihrer Schwierigkeit unterscheidende Ansätze zur Bewältigung dieser Aufgabe:

- Knotensplitt

  Der Aufenthaltsknoten $a_i$ könnte in zwei oder mehr Knoten zerlegt und miteinander verbunden werden, so dass die Lade- bzw. Restladezeit eines jeden der für $a_i$ neuen Knoten den Wert $r^*$ nicht überschreitet.

  Diese Entscheidung betrifft die Segmentierung des Hypermediums. Präsentationseinheiten werden in kleinere, miteinander linear verknüpfte Einheiten zerlegt. Dabei ist jedoch zu beachten, dass die daraus resultierenden Knoten nicht mehr redundant zu Knoten aus $G_k$ sind. Damit kann sich das Optimierungsergebnis des modifizierten Baums verschlechtern.[8]

- Knoten einfügen

  Ein oder mehrere Knoten mit hohen Nutzzeiten und niedrigen Ladezeiten werden in die Teilsequenz des Navigationsbaums, bestehend aus Knoten auf dem Weg von der Wurzel bis zu $a_i$, eingefügt. Im Extremfall handelt es sich um Knoten mit einer Ladezeit von annähernd Null. Die Nutzzeit dieser Knoten wird dazu verwendet, Teile des Objekts aus $a_i$ zu laden. Es handelt sich dabei um Knoten, deren Hauptfunktionalität darin besteht, die auf diese folgenden Objekte zu laden, so genannte Preloader, wie sie häufig in webbasierten Anwendungen zum Einsatz kommen. Die Nutzzeit dieser Knoten ist als Lernpause bzw. gezielte Unterbrechung des Lernprozesses anzusehen.

[8] Das Off- wie Online-Modell bildet eine solche Art der Redundanz, dass Objekte eines Knotens nur teilweise redundant zu einem anderen Knoten sind, nicht ab.

- Festlegen einer Menge von Ladeentscheidungen durch den Entwerfenden

  Der Entwerfende kann vor der Optimierung eine Menge von Ladeentscheidungen $y_{jk} \in Y$ „manuell“ festlegen. Das betrifft jedoch nur solche Knoten $a_j$, welche sich auf dem Weg der Teilsequenz $\sigma_1^{i-1}$ zwischen Wurzelknoten $a_1$ des Gesamtbaums und $a_i$ befinden.[9]

  Um den angestrebten Wert $r^*$ zu erreichen, muss dann gelten:

$$r^* = \left( \sum_{\forall a_j \in \sigma_1^{i-1}} y_{jk} \right), \quad a_i \in G_k \tag{6.1}$$

  Für eine Optimierung auf Basis dieser festgelegten Prefetchingentscheidungen ist der Navigationsbaum so zu modifizieren, dass die Nebenbedingungen des Modells eingehalten werden können. Es sind Nutzzeiten entsprechend anzuheben bzw. Ladezeiten entsprechend zu senken:

  Für jedes $y_{jk}$ der Summe aus obiger Gleichung ist die Nutzzeit des Knotens $a_j$ um genau den Wert $y_{jk}$ zu senken und die Ladezeit von $l_i$ ist genau mit $r^*$ festzusetzen. In jedem Fall liefert ein Optimierungsverfahren für den so modifizierten Navigationsbaum keine Entscheidungsmatrix $Y^*$, mit welcher $r_i > r^*$ ist, weil $l_i = r^*$ nach der Modifikation gilt.

  Zu berücksichtigen ist, dass für den Optimalwert des modifizierten Baums ($Wert(Y^*)$) im Vergleich zum Optimalwert vor der Modifikation ($Wert(Y)$) gilt:

$$Wert(Y) \leq Wert(Y^*) \tag{6.2}$$

  Durch Mindern der Nutzzeiten wird der Entscheidungsspielraum eingeschränkt. Es ist leicht einzusehen, dass das Optimierungsergebnis $Wert(Y^*)$ keinesfalls besser sein kann als das Ergebnis $Wert(Y)$ vor der Modifikation. Grundsätzlich gilt aber, dass der Entscheidungsspielraum umso geringfügiger eingeschränkt wird und damit die Differenz aus $Wert(Y^*) - Wert(Y)$ tendenziell kleiner sein wird, je kleiner die Anzahl aller Kanten auf den Wegen von den Knoten $a_j$ nach $a_i$ im Navigationsbaum ist. Dabei sind die Knoten $a_j$ all jene Knoten, deren Nutzzeit durch die Modifikation reduziert wurde. Dieser Zusammenhang entsteht aus dem Sachverhalt, dass die Nutzzeit eines Knotens zum Laden von um so mehr Knoten des Baums verwendet werden kann, je kleiner das Niveau dieses Knotens im Baum ist, das heißt je näher der Knoten sich an der Wurzel befindet.

  **Bemerkung:** *Soll das selbe Optimierungsergebnis vor wie nach der Modifikation erreicht werden ($Wert(Y) = Wert(Y^*)$), dann stehen zwei einfache Möglichkeiten zur Anpassung der Segmentierung des Baums zur Verfügung:*

  – Die Nutzzeit all jener Knoten $a_j$, deren Nutzzeit mit der Modifikation reduziert wurde, ist wieder auf den Wert vor der Modifikation anzuheben bzw. die Nutzzeit von Knoten dichter zur Wurzel als $a_j$ können im selben Maß, das heißt um die Differenz zum Wert vor der Modifikation angehoben werden.

[9] Zur vereinfachten Darstellung wird wiederum angenommen, dass die Indizes der Knoten in der Sequenz entsprechend in deren auftretenden Reihenfolge vergeben sind.

– Weiterhin können die Ladezeiten all jener Knoten, welche mit $Y$ während der Nutzung von $a_j$ jeweils im Voraus geladen werden, in der Summe um jene $y_{ik}$ reduziert werden.

Beide Möglichkeiten können auch kombiniert angewendet werden. Sie bewirken denselben Optimalwert.

- Modifizieren der Lade- und Nutzzeiten von Knoten des Baums, so dass das Optimierungsverfahren für den Knoten die angestrebte Restladezeit ermittelt.
- Modifizieren der Reihefolge der Aufenthaltsknoten des Baums, so dass das Optimierungsverfahren für den Knoten die angestrebte Restladezeit ermittelt

Zur Diskussion der letzten beiden von den fünf Ansätzen zum Senken der Restladezeit eines Knotens ist folgender Satz von Bedeutung:

**Satz 6.1** *Es existiert kein effizientes Verfahren, welches angibt, in welcher Höhe Nutz- bzw. Ladezeiten wie auch Reihenfolge der Knoten eines redundanten Navigationsbaums zu verändern wären, so dass ein bestimmter Optimalwert x von einem der Optimierungsverfahren des Offline-Modells für den modifizierten Navigationsbaum geliefert wird, wenn gilt $P \notin NP$.*

Gilt dieser Satz, kann kein effizientes Verfahren für die beiden zuletzt genannten Ansätze konstruiert werden, denn die jeweils durch die Veränderungen zu erreichende Restladezeit des Knotens könnte auch der Optimalwert für Optimierungsprobleme der Zielstellungen $2MLB^*$ oder $2MLS^*$ sein bzw. die Restladezeit könnte neben anderen in die Summe des optimalen Zielfunktionswerte eingehen.

**Beweis von Satz 6.1:** Zum Nachweis des Satzes wird folgendes Entscheidungsproblem formuliert:

Entscheidungsproblem „Modifikation möglich?“

Probleminstanz: Gegeben ist ein redundanter Navigationsbaum mit mindestens einer Knotengruppe $G_k \geq 3$ sowie Modifikationen folgender Art:

- Die Lade- oder Nutzzeiten eines oder mehrerer Aufenthaltsknoten des Baums werden verändert.
- Die Reihenfolge der Knoten in einer oder mehreren Sequenzen des Baums werden verändert.

Fragestellung:
Existieren Modifikationen des redundanten Navigationsbaums, so dass ein Optimierungsverfahren für die $*$-Ziele des Offline-Modells eine Entscheidungsmatrix $Y^*$ mit dem Lösungswert $x = Wert(Y^*)$ liefern kann?

Würde ein Verfahren existieren, welches effizient in Abhängigkeit von der Anzahl der Knoten des Navigationsbaums $B$ eine Antwort auf das beschriebene Entscheidungsproblem liefern könnte, dann wäre auch eine effiziente Lösung für die bereits als $NP$-vollständig eingestuften Problemstellungen der $*$-Ziele des Offline-Modells möglich, weil:

Je nach Zielstellung umfasst der Lösungsraum für die optimalen Werte weniger als $max_{\forall a_i \in B}(l_i)$ ($2MLB^*$, $2MLS^*$) bis $n^2 \cdot max_{\forall a_i \in B}(l_i)$ ($MSLS^*$) unterschiedliche Ergebniswerte. Absteigend kann für jeden Wert $x$ in diesem Bereich das imaginär effiziente Verfahren iterativ zur Beantwortung des Entscheidungsproblems verwendet werden.[10] Liefert das Verfahren die Antwort „nein“ im $i$-ten Iterationsschritt und im $i+1$-ten Schritt „ja“, dann ist das optimale Ergebnis für den gegebenen Navigationsbaum der Wert von $x$ der $i$-ten Iteration.

Würde ein effizientes Verfahren existieren, welches eine Antwort auf das Entscheidungsproblem liefert, dann muss gelten $P = NP$. Im Umkehrschluss muss unter der Vorraussetzung $P \notin NP$ gelten (siehe Satz), dass kein solches Verfahren existiert. Der Satz ist zu bestätigen. □

Dieses Ergebnis hat zur Folge, dass es unmöglich erscheint, für die beiden Ansätze zur Reduktion der Restladezeit eines Knotens unter der Maßgabe eines Zielfunktionswerts ein effizientes Verfahren zu konstruieren. Es werden spezielle Näherungsverfahren zur Entscheidungsunterstützung benötigt. Diese sollen nicht Gegenstand dieser Arbeit sein. Es ist weiter zu vermuten, dass diese aus den bereits diskutierten Approximationsverfahren abgeleitet werden können. Es können einfache Heuristiken in Form von Faustregeln für den Entwerfenden angegeben werden, um die Restladezeit eines Knotens durch Modifikation des Navigationsbaums zu senken.

Aus folgenden Heuristiken können Entscheidungen zur Segmentierung und Sequenzierung abgeleitet werden. Sie verfolgen das Ziel, den Navigationsbaums so zu modifizieren, dass die Restladezeit eines Knotens $a_i$ gesenkt wird.

**Segmentierungsentscheidungen**

- Ein oder mehrere Knoten auf dem Weg vom Wurzelknoten des Baums nach $a_i$ einfügen, deren Ladezeit wesentlich geringer ist als die Ladezeit von $a_i$ und welche eine verhältnismäßig große Nutzzeit besitzen
- Die Nutzzeit eines oder mehrerer Knoten auf dem Weg zwischen Wurzelknoten des Baums und $a_i$ erhöhen
- Die Ladezeit eines oder mehrerer Knoten des Teilbaums mit Wurzel $a_i$ verringern
- Die Ladezeit eines oder mehrerer Knoten, welche vor der Modifikation des Baums im Voraus während der Nutzung von Knoten auf dem Weg zwischen Wurzel und $a_i$ geladen werden, reduzieren

**Sequenzierungsentscheidungen**

Die Platzierung zweier Knoten $a_x$ und $a_y$ einer Sequenz, welche die Knoten $a_i, a_x, a_y$ enthält, kann getauscht werden, wenn

[10] Effizienter noch mittels binärer Suche im Ergebnisraum

- die Ladezeit von $a_y$ kleiner oder gleich sowie die Nutzzeit größer ist als die von $a_x$ oder
- die Ladezeit von $a_y$ kleiner sowie die Nutzzeit größer oder gleich ist als die von $a_x$,

wobei $a_x$ in der Sequenz vor $a_i$ und $a_y$ nach $a_i$ platziert sind. Ein solcher Tausch ist in Abbildung 6.2 dargestellt.

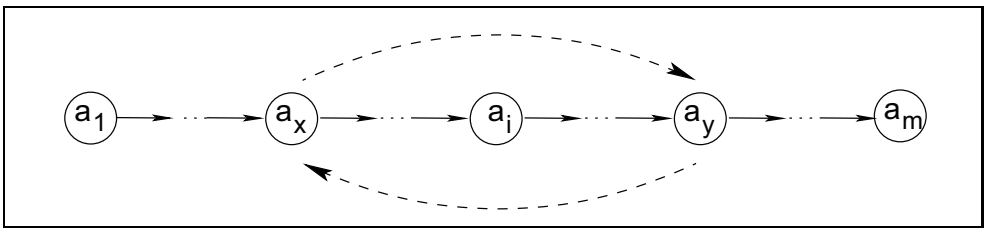



Abbildung 6.2: Die Sequenzierungsentscheidung „Knoten $a_x$ tritt an Stelle des Knotens $a_y$ und umgekehrt“ kann Wirkung auf die Restladezeit von $a_i$ haben.

Werden die Entscheidungsmatrizen der optimalen Ergebnisse vor und nach der Anwendung dieser Heuristiken betrachtet, dann sinkt zwar nicht zwangsläufig die Restladezeit des Knotens $a_i$, jedoch wird die Restladezeit von $a_i$ keines Falls ansteigen. Die Heuristiken können keine Garantie für ein Sinken der Restladezeit liefern. Sie sind aber trotzdem hilfreich, um effizient rechnergestützte Lösungsvorschläge zur Hilfestellung für Segmentierungs- und Sequenzierungentscheidungen des Entwerfenden zur Reduktion der Restladezeit eines Knotens zu generieren.

**Bemerkung:** *Die Restladezeit eines Knotens $a_i$ kann nur dann zwingend mittels zusätzlich verfügbarer Nutzzeit gesenkt werden, wenn alle Entscheidungen zum Laden im Voraus des Objekts aus Knoten $a_i$ explizit durch den Entwerfenden fixiert werden. Das heißt, es sind nicht nur Ladeentscheidungen zum Laden von $a_i$ festzulegen, welche vor der Modifikation vom Optimierungsverfahren vorgeschlagen wurden, sondern es sind weitere zusätzlich notwendige Ladeentscheidungen festzulegen, um nach der Modifikation zusätzlich verfügbare Nutzzeit zum Laden im Voraus auf $a_i$ zu verwenden.*

### Reduktion der Restladezeit mehrerer Knoten zur Verbesserung des Optimalwerts

Als zweiter großer Problembereich stellt sich die Modifikation von Eigenschaften des Gesamtbaums dar, um eine angestrebte Lösung mit Hilfe des eingesetzten Prefetchingverfahrens erreichen zu können.

Um den Entwurfsprozess gezielt stützen zu können, wären effiziente Verfahren wünschenswert, welche für gegebenen Navigationsbaum, Zielfunktion und Entscheidungsmatrix sowie angestrebten Optimalwert Modifikationsvorschläge für Eigenschaften des Gesamtbaums generieren. So kann der Entwerfende erkennen, welche Kompromisse er mit den Eigenschaften des Hypermediums eingehen müsste, um ein bestimmtes, anzustrebendes Antwortzeitverhalten erreichen zu können. Da die Konstruktion solcher Verfahren für die Modifikation des Baums mit der Zielstellung, eine bestimmte Restladezeit eines Einzelknotens zu erreichen, unmöglich erscheint,

ist der Wunsch nach einem solchen Verfahren zwar legitim, aber als reine Illusion anzusehen. Wie schon im vorherigen Abschnitt angedeutet, ist zu vermuten, dass Approximationsverfahren konstruiert werden können, die den Entscheidungsprozess teilweise unterstützen können. Solche Verfahren wurden im Rahmen dieser Arbeit jedoch nicht entwickelt. Die Konstruktion der Verfahren sollte auch nicht Gegenstand dieser Arbeit sein. Nach jetzigem Stand können nur die optimalen bzw. approximierten Ergebnisse für einen Navigationsbaum entsprechend der gelieferten Entscheidungsmatrix vor und nach Modifikationen ermittelt und miteinander verglichen werden. Empfehlungen für die Art und Weise der Modifikationen können nur mittels einfacher Heuristiken und durch den Vergleich derer Ergebnisse generiert werden. Hier spielen menschliche Intuition und analytisches Denken des Entwerfenden eine wichtige Rolle. Damit der Entwerfende gezielt Entscheidungen im Rahmen des angestrebten Optimierungsziels treffen kann, ist jedoch detailliertes Wissen über Funktions- und Wirkungsweise des jeweiligen Prefetchingverfahrens notwendig. Davon ist aber grundsätzlich nicht auszugehen.

Um den Entwurfsprozess besser rechnerunterstützen zu können, sind weitere Arbeiten notwendig. Der hier vorgestellte Ansatz zur Reduktion der Restladezeiten eines oder mehrerer Knoten ist aufgrund der Komplexität der Optimierungsprobleme für redundante Navigationsbäume nur bedingt nutzbar. Weiterhin vermutet der Autor, dass für die dargestellten Entwurfsprobleme nicht redundanter Bäume effiziente Verfahren konstruiert werden können. Das wäre in weiterführenden Arbeiten zu prüfen.

## 6.2 Laufzeit

Mit den Prefetchingverfahren des Online-Modells werden Entscheidungsmechanismen für die Laufzeit der Lernanwendung unter Berücksichtigung der Entwurfsentscheidungen geliefert. Die Online-Modelle bilden aber nur zum Teil die realen Gegebenheiten während der Laufzeit nach. Aufgrund der Abstraktion von vielzähligen Einflussfaktoren auf die Leistung des Prefetchingverfahrens ist mit Abweichungen von der im Entwurf erwarteten Leistung zu erwarten. Die im Entwurf angenommene, eher theoretisch betrachtete Leistungsfähigkeit ist deshalb zu relativieren. In diesem Abschnitt wird auf diese Problematik eingegangen, indem ein kurzer Überblick zur Erfassung von Leistungsabweichungen, möglichen Anpassungen des Modells sowie auf die Leistungssteigerung durch Kombination mit anderen Prefetchingansätzen eingegangen wird.

Die Entwurfsentscheidungen des vorherigen Kapitels zielen darauf ab, dass unter Einsatz eines bestimmten Online-Verfahrens ein bestimmtes, zur Spezifikation passendes Antwortzeitverhalten garantiert wird. Da jedoch von den realen Bedingungen während der Laufzeit abstrahiert wurde, das heißt nur ausgewählten Gegebenheiten mit den Modellannahmen und Parametereinstellungen entsprochen wurde, ist unbedingt mit Abweichungen vom im Entwurf angenommenen Verhalten zu rechnen. Solche Abweichungen sind mit dem dynamischen Verhalten der Lernanwendung zu erkennen, zu bewerten und, falls notwendig, entsprechende Maßnahmen in Form des Performance-Tunings einzuleiten.[11]

Negativ auf die Leistung im Betrieb der Anwendung wirken neben weiteren noch zu diskutierenden Faktoren:

[11] Vgl. dazu die Einbettung des Performance Tunings in den Rahmenprozess zum leistungsorientierten Betrieb, bestehend aus den Teilprozessen Performanceüberwachung, Performance Tuning und Performance Validierung (vgl. [RS99]).

- kürzere Nutzzeiten sowie
- längere Ladezeiten der Aufenthaltsknoten als im Entwurf angenommen und
- mit dem Navigationsbaum nicht berücksichtigte Anfragefolgen der lokalen Anwendungskomponenten.

An diese Abweichungen kann das Modell durch eine veränderte Parametereinstellung, das heißt durch einen modifizierten Navigationsbaum, angepasst werden. Deshalb sind diese Abweichungen mit der Laufzeit im Betrieb oder schon in der Testphase der Lernanwendung zu identifizieren und gegebenenfalls Tuningmaßnahmen aufgrund veränderter Modellparameter und daraus resultierender Leistungsfähigkeit des Prefetchingverfahrens einzuleiten.

### 6.2.1 Erfassung von Abweichungen zu den Modellparametern

Die Erfassung der Nutz-, Ladezeiten und Anfragefolgen erfolgt mit dem Ziel, aus den erfassten Daten Informationen zur Modifikation des Navigationsbaums abzuleiten. Für Lernanwendungen ist damit eine sekundengenaue Erfassung der Nutz- und Ladezeiten notwendig. Nutzzeiten und Reihenfolge der angefragten Aufenthaltsknoten variieren je nach Navigationsverhalten der Lernenden. Ladezeiten schwanken in Abhängigkeit von der verfügbaren Bandbreite der Netzverbindung zwischen entfernten Ressourcen und lokalen Anwendungskomponenten. Damit ist einerseits das Navigationsverhalten, andererseits die verfügbare Bandbreite der Verbindungen zwischen den verteilten Anwendungskomponenten zu erfassen. Zur automatisierten Erfassung der benötigten Daten eignen sich Logmechanismen, installiert bei der lokalen Anwendungskomponente. Diese protokollieren aus Sicht der Komponente Anfragezeitpunkt und Lieferzeitpunkt eines Objekts. Daraus können unmittelbar Nutz- und Ladezeiten der Aufenthaltsknoten abgeleitet werden. Durch die lokale Nähe eines Logmechansimus zur lokalen Anwendungskomponente wird eine systematisch verfälschte Erfassung der Daten bezüglich Anfragezeitpunkte und Dauer bis zur Lieferung der Objekte ausgeschlossen. Zur genauen Erfassung des Navigationsverhaltens des Lernenden wurde bereits im Kapitel 2, Abschnitt „Architekturkomponenten“, S.39 der lokale Beobachter vorgeschlagen. Die Erfassung der cachebezogenen Daten ist allein bei diesem zu realisieren. Sie lassen sich verwenden, um Rückschlüsse auf die Qualität der Verbindungen zu entfernten Ressourcen zu gewinnen.

Die Erfassung von Navigationsverhalten, das so genannte Lerner-Tracking, gehört zu den Funktionalitäten moderner Lernplattformen. Sie kann jedoch nur eingeschränkt zur Erfassung der benötigten Daten eingesetzt werden. Lerner-Tracking wird durchgeführt, um Lernerfolg und Lerneffizienz zu beurteilen. Dazu ist jedoch eine sekundengenaue Erfassung der Nutzzeiten nicht notwendig. Im Vordergrund steht die Bewertung von Lernpfaden und die Protokollierung von Aufgabenlösungen (vgl. [SHF03] und [Ker01b], S.212). Deshalb werden zur Umsetzung des Lerner-Trackings Logmechanismen angewendet, die „nur“ Zugriffe auf entfernte Ressourcen direkt bei den entfernten Ressourcen protokollieren.[12]

[12] Vergleiche dazu die Diskussion von *Zaiane*, welcher ausführlich die Grenzen der Informationsgewinnung aus Logdateien in Bezug auf die Auswertung des Navigationsverhaltens in Lernanwendungen darstellt (vgl. [Zai01]), sowie den detaillierten, sehr kritisch angelegten Überblick von *Davidson* in [Dav99] über die Grenzen der Informationsgewinnung aus Logdaten aus dem Blickwinkel von Prefetching, sowie *A. Joshi*, *K. Joshi* und *Krishnapruram* in [JK00] mit der Diskussion von Dataminingtechniken zur Informationsgewinnung aus Logdaten für webbasierte Anwendungen.

Die Vor- und Nachteile einzelner Mechanismen zum Protokollieren sollen hier nicht diskutiert werden. Spezielle Anforderungen sind nur bezüglich zeitlicher Genauigkeit der Daten zu erfüllen. Um Entscheidungen des Entwurfs anpassen zu können, sind die protokollierten Daten aller lokalen Anwendungskomponenten und Caches aggregiert zentral zur Verfügung zu stellen. Dies kann erfolgen, indem mit dem Verbindungsaufbau zwischen lokalen Anwendungskomponenten und entfernten Ressourcen auch die protokollierten Daten an einen Erfassungsdienst ausgeliefert werden. Es ist nicht sinnvoll, die Daten erst dann vom lokalen Beobachter und Cache anzufordern, wenn sie der Analyse unterzogen werden sollen. Die mögliche zeitlich divergierende Nutzung der Lernanwendung durch die Lernenden verhindert einen zeitunabhängigen Zugriff auf die lokal gesammelten Daten.

### 6.2.2 Anpassung des Navigationsbaums

Für die Anpassung des Navigationsbaums sind zwei Bereiche von Interesse, die „automatische", das heißt rein durch Algorithmen gesteuerte, Modifikation der Modellparameter während der Laufzeit, welches zu einem adaptiven Prefetchingverfahren führen würde, und die Anpassung durch Überarbeitung der Entwurfsentscheidungen, das eigentliche Performance-Tuning durch reaktive Maßnahmen zur Leistungsanpassung.

#### Performance-Tuning

Aus den protokollierten Daten kann für jeden Aufenthaltsknoten die mit dem Entwurf unterstellte Hypothese über mindestens anfallende Nutz- und höchstens anfallende Ladezeit eines jeden Knotens geprüft werden. Ein häufiges Unter- bzw. Überschreiten der Nutz- bzw. Ladezeiten muss eine entsprechende Anpassung der Nutz- und Ladezeiten im Navigationsbaum nach sich ziehen, weil dadurch die mit dem Entwurf unterstellte Leistung des Prefetchingverfahrens nicht mehr erreicht werden könnte. Das ist am entsprechend durch Anpassung der Nutz- und Ladezeiten modifizierten Navigationsbaum zu prüfen.

Weiterhin können mit den erfassten Daten von den Lernenden angeforderte Knotensequenzen, die einzelnen Session eines Lernprozesses, rekonstruiert werden. Bei der Rekonstruktion ist auf Besonderheiten der Lernenden gegenüber allgemein üblichem, durchschnittlichem Verhalten im WWW zu achten. *Zaiane* streicht insbesondere längere Zeiträume zwischen Datenanfragen und sehr lange Session der Lernenden über Tage und Monate heraus.[13] Hier ist zur Prüfung korrekter Modellannahmen von Interesse, ob das Hypermedium Sequenzen zulässt, welche nicht mit dem Navigationsbaum berücksichtigt wurden. Um diese ist der Navigationsbaum zu erweitern. Genauso sind ungenutzte Sequenzen zu entfernen. Sie reduzieren sonst das zu erwartende Leistungspotential. Dadurch ist eine kleinere, sich aus dem Navigationsbaum ergebende obere Schranke für den Optimalwert des Prefetchingverfahrens zu erwarten.

Kann aus den Anpassungen auf eine kritische Leistungsdifferenz geschlossen werden, sind die Entwurfsentscheidungen des Hypermediums zu überarbeiten. Die Parameterabweichungen sowie Änderungen in der Struktur des Hypermediums sind beim Prefetchingverfahren der jewei-

[13] In Bezug auf das WWW wird mit einer Session eher eine Menge von Anfragen in einem zeitlich eng gesteckten Rahmen von wenigen Sekunden bis zu wenigen Stunden, in der Regel initiiert durch einen Nutzer, beschrieben (vgl. [HG00]).

ligen lokalen Anwendungskomponente abzugleichen. Dazu ist eine softwaretechnische Umsetzung der lokalen Prefetchingkomponente notwendig, die Schnittstellen zu Anpassung des Navigationsbaums zur Verfügung stellt.[14]

### Adaption

Eine Leistungssteigerung der Prefetchingverfahren ist auch möglich, indem der Navigationsbaum nicht nur an die sich ändernden Gesamtbedingungen mittels Performance-Tuning angepasst wird, sondern indem je nach Rahmenbedingungen unterschiedliche Navigationsbäume konstruiert werden bzw. Transformationsregeln entwickelt werden, welche eine laufzeitnahe Anpassung des Baums ermöglichen.

Eine Adaption des Navigationsbaums ist aber nur dann sinnvoll möglich, wenn frühzeitig während einer Session aus dem erkennbaren Verhalten und Eigenschaften des Lernenden sowie den technischen Rahmenbedingungen, in welchen die lokale Anwendungskomponente betrieben wird, auf zukünftiges Navigationsverhalten und Ladezeiten geschlossen werden kann. Anhaltspunkte dafür kann der gezielte Einsatz von statistischen Verfahren im Rahmen des Dataminings in den erfassten Daten zum Nutzerverhalten und zu den Ladezeiten liefern.[15] Durch die detaillierten Vorstellungen über die Zusammenhänge zwischen Persönlichkeitseigenschaften, Lernstile, -strategien und Navigationsverhalten der Lernenden[16] sowie eine unmittelbare Korrelation der Ladezeiten mit der verfügbaren Bandbreite von Netzanbindungen zu den entfernten Ressourcen erscheinen dynamische als auch statische Adaptionstechniken für Prefetchingverfahren in Lernanwendungen besonders viel versprechend zur Leistungssteigerung.[17] Untersuchungen hierzu existieren jedoch noch nicht. Die Vermutung des Autors resultiert aus den vielzähligen, in den letzten Jahren entstandenen Veröffentlichungen zur Personalisierung und Adaption von Lernanwendungen. Diese zielen aber nicht auf die Anpassung von Antwortzeitverhalten beeinflussenden Parametern ab, sondern auf die Anpassung von Navigationfunktionalitäten, Präsentation der Lerninhalte und Verfügbarkeit von Funktionalitäten. Die Adaption erfolgt mit dem Ziel, die Lerneffizienz zu steigern (vgl. [KA03] und [SCB04]). Der Umstand, dass scheinbar eine sinnvolle Adaption überhaupt möglich ist, lässt durch die enge Verknüpfung von Lernstil und Navigationsverhalten vermuten, dass auch für Prefetchingverfahren Adaptionsmechanismen sinnvoll eingesetzt werden können. Kern der Konzepte zur Adaption ist die Betrachtung des Nutzerverhaltens und ein Zuschnitt der Anwendungseigenschaften auf einzelne Personen oder Personengruppen mit Hilfe verschiedener Adaptionsdienste (vgl. [DHNS04]). Hier ist ein Zuschnitt des Prefetchingverfahrens auf die Nutzer und Rahmenbedingungen der jeweiligen lokalen Anwendungskomponente notwendig. Für die hier betrachteten Prefetchingverfahren wäre

[14] Gegebenfalls ist nicht nur eine kleine Modifikation, sondern ein kompletter Wechsel von einem Verfahren zu einem anderen notwendig. Hierfür sind Techniken zum Einsatz variierender Prefetching- und Cachestrategien in der Betriebsphase notwendig. In [SDMML03b] und [BP99] werden dazu verwendbare Konzepte für variierende Cache- und Prefetchingstrategien dargestellt.

[15] Vgl. dazu [Zai01] und [SCB04], die Dataminingtechniken zur Beurteilung von Lernverhalten und darauf basierende Anpassung von Lernanwendungen nutzen. Auf die einzelnen statistischen Methoden, das Vorgehen und Architekturanpassungen zur möglichen Adaption wird hier nicht weiter eingegangen.

[16] Vgl. Kapitel 2 Abschnitt „Navigationsverhalten“, S.23ff..

[17] Statisch und dynamisch adaptierende Verfahren unterscheiden sich durch den Zeitpunkt der Erstellung unterschiedlicher Ausprägungsvarianten dessen, was zu adaptieren ist (vor bzw. während der Laufzeit). Dynamische Adaption erfolgt nahezu ausschließlich mit algorithmisch realisierten Transformationen einer Ausprägung der Anwendung in weitere, angepasste Varianten (vgl. [KPRR04], S.65ff.).

neben einer Adaption an das Nutzerverhalten auch eine Adaption an die Netzanbindung der lokalen Anwendungskomponente notwendig.

Es erscheinen beispielsweise folgende, einfach realisierbare Anpassungen möglich:

- Anpassen der Nutzzeiten an die Lerngeschwindigkeit des Lernenden durch einen variierenden linearen Faktor

  Je nach Dauer des Lernprozesses kann zwischen „langsamen“ und „schnellen“ Lernern unterschieden werden. Auf die Lerngeschwindigkeit kann anhand der Nutzzeit bereits besuchter Aufenthaltsknoten geschlossen werden.

- Anpassen der Ladezeiten an die jeweils aktuell verfügbare Bandbreite zum Beispiel durch einen variierenden linearen Faktor

  Die Bandbreite kann aus der Ladedauer und dem Umfang der bereits an den Cache übertragenen Daten abgeleitet werden.

- Anpassen der mit dem Navigationsbaum für das Prefetchingverfahren betrachteten, navigationsbezogenen Links

  Der Baum könnte beispielsweise in seiner Höhe variabel gestaltet werden. Anhand der Nutzzeiten sowie Reihenfolge besuchter Aufenthaltsknoten kann auf ein tiefen- oder oberflächenstrategisches Vorgehen beim Lernen geschlossen werden.[18]

Die drei einfach gehaltenen Ansätze können leicht algorithmisch beschrieben werden und eignen sich damit für eine dynamische wie auch statische Adaption in Form der Anpassung des Navigationsbaums.

Im Rahmen weiterer Forschungsarbeiten wären eine systematische Untersuchung von Anpassungsmöglichkeiten der hier konstruierten Prefetchingverfahren und deren Realisierung mittels dynamischer und statischer Adaption sowie die Beurteilung von Lösungsvarianten notwendig.

### 6.2.3 Kritische Modellannahmen

Ein weiteres Problem sind Leistungsabweichungen durch drastische Abweichungen der modellimmanenten Annahmen von den realen Gegebenheiten. Dabei handelt es sich um Annahmen, welche nicht oder nur in sehr geringem Maß mit angepassten Parameterausprägungen entsprochen werden kann.[19] Sie sind bei der Leistungsbewertung im Betrieb der Anwendung unbedingt zu berücksichtigen und, falls technisch wie organisatorisch machbar, zu protokollieren, um dem Betreiber der Lernanwendung Rückkopplung über ein etwaig bestehendes Missverhältnis zwischen Modellannahmen und realen Gegebenheiten liefern zu können.

Folgende, durch das Modell nicht berücksichtigte Faktoren wirken drastisch auf die Leistung des Prefetchingverfahrens, wobei unterschieden wird in tendenziell gut und schlecht durch Entwickler und Anbieter der Lerninhalte beeinflussbare Faktoren.

[18] Siehe Kapitel 2, Abschnitt „Navigationsverhalten“, S.23

[19] In Kapitel 3 wurden mit den Abschnitten „Modellannahmen“, S.59ff. und „Diskussion der Modellannahmen“, S.71ff. die mit dem On- und Offline-Modell getroffenen restriktiv wirkenden Einschränkungen ausführlich diskutiert.

- Tendenziell gut beeinflussbare Faktoren:

  **Prefetchability** der entfernt gespeicherten Lerninhalte

  Es ist auf den verbreiteten Einsatz von Technologien und Funktionalitäten zu verzichten, die zu großen, entfernt gespeicherten, schlecht im Voraus ladbaren Datenmengen führen.

  Jüngere Forschungsarbeiten beschäftigen sich damit, scheinbar schlecht im Voraus ladbare Daten wie zum Beispiel Anfrageergebnisse von Suchmaschinen auf komplexe Anfragemuster vorherzusagen (vgl. zum Beispiel [LM03]). Eine Hoffnung für einen breiter möglichen Einsatz von Prefetching besteht darin, dass heute als nicht oder nur schlecht im Voraus ladbar geltende Daten durch weiterentwickelte, spezialisierte Vorhersagemethoden, bezogen auf Anwendungsgebiete wie hier die Unterstützung von Lernprozessen, zukünftig als gut „prefetchable“ gelten.

  **Geschlossenes Hypermedium**

  Es ist auf den exzessiven Einsatz von navigationsbezogenen Links zu anwendungsextern gespeicherten Lerninhalten zu verzichten. Das heißt, wenn möglich, ein Verzicht auf Lerninhalte, welche sich einer Modellierung mittels Navigationsbaum entziehen.

  Die Funktionalität des Prefetchings von Daten hat einerseits Middleware-Charakter, sie wirkt anwendungsübergreifend. Mit Prefetchingverfahren, realisiert auf der Transportschicht eines verteilten Rechnernetzes, zum Beispiel verbunden mit Proxydiensten, wird versucht, diesem Charakter gerecht zu werden (vgl. [AFJ99], [CZB00], [LL02] und [SP03]). Andererseits wirken eine Vielzahl von anwendungsspezifischen Faktoren auf Prefetchingverfahren. Diesen kann mit einer alleinigen Realisierung als Middleware-Komponente nicht entsprochen werden. Das wurde besonders in dieser Arbeit bei der Betrachtung von Prefetchingverfahren für Lernanwendungen sichtbar. Eine Vielzahl von Forschungsarbeiten beschäftigt sich damit, den anwendungsbezogenen wie auch anwendungsübergreifenden Charakter durch spezielle Architekturen für Prefetchingverfahren gerecht zu werden. Zielrichtung ist die Identifikation und Realisierung von Basisfunktionalitäten für Middleware-Komponenten in Kombination mit austauschbaren Prefetchingstrategien auf Anwendungsebene. Zukünftige Entwicklungen dieser Art werden jedoch durch fehlende, standardisierte Schnittstellen für Cache- und Prefetchingtechnologien behindert. Erste Entwicklungen in diese Richtung zeichneten sich mit der Standardisierung von Schnittstellen für Prefetchingverfahren durch das W3C-Konsortium ab.[20]

Mit einer Vielzahl kleiner Veränderungen der Inhalte über den Lebenszyklus der Lernanwendung ist unbedingt darauf zu achten, dass die Leistungsfähigkeit des Prefetchingverfahrens nicht sukzessive und dadurch unbemerkt eingeschränkt wird. Hier ist, wie schon mit dem Entwurf diskutiert, ein günstiger Kompromiss zwischen Anpassung der Lerninhalte und Einschränkung der Leistung zu finden. Der Kompromiss ist geprägt durch das technisch und methodisch zurzeit Mögliche an realisierbarer Leistung und machbaren Freiheitsgraden in der Entwicklung und Präsentation von Lerninhalten. Mit vielzähligen marginalen Veränderungen über längere Zeiträume besteht die Gefahr, die

[20] Siehe Kapitel 2, Abschnitt „Web-Prefetching“, S.19

Leistungsfähigkeit zu vernachlässigen  insbesondere dann, wenn die Autoren der Lerninhalte, völlig verständlich und nachvollziehbar, aus rein technischer Sicht, anwendungskritischen Aspekten wie dem Antwortzeitverhalten nur eine untergeordnete Bedeutung beimessen. Im Betriebsprozess werden Autoren häufig auch mit softwaretechnisch gerichteten Entscheidungen wie zum Beispiel der Präsentationsqualität von Daten konfrontiert, ohne explizite Beratung durch Experten. Deshalb sind Kontrollmechanismen in die Lernanwendung bzw. das Prefetchingverfahren zu integrieren, welche die Resultate dieser Entscheidungen abschätzen können und gegebenenfalls alarmierend wirken.

- Tendenziell schlecht beeinflussbare Faktoren, die im Regelfall der Kontrolle des Lernenden oder Dritten unterliegen:

  **Verfügbare Cachekapazität** Mit den hier entworfenen Verfahren wird ein beliebig großer Cache angenommen. Eine Einschränkung der Cachegröße, kleiner als der Speicherplatzumfang aller von entfernten Ressourcen angeforderten Daten, kann negativ auf das Antwortzeitverhalten wirken.

  Die von der lokalen Anwendungskomponente nutzbare Cachegröße ist von der Konfiguration des Anwendungssystems, auf welchem die lokale Anwendungskomponente betrieben wird, als auch von weiteren Anwendungen, welche denselben Cache nutzen, abhängig. Eine Mindestcachegröße sowie Folgen bestimmter Cachestrategien zur Verwaltung des limitierten Speichers sind an den Anwender zu kommunizieren.

  **Verfügbare Bandbreite** Die Verbindung zwischen lokaler Anwendungskomponente und entfernten Ressourcen unterliegt nur teilweise dem Betreiber der Lernanwendung. Häufig wird der Fall anzutreffen sein, dass die Entscheidung für oder gegen eine bestimmte Bandbreite der Verbindung zwischen lokaler Anwendungskomponente und entfernter Ressource vom Lernenden getroffen wird. Kostenaspekte spielen eine dominante Rolle. Auch muss berücksichtigt werden, dass die nutzbare Bandbreite von der vorherrschenden Netzlast auf der Verbindung beeinflusst wird. Die Netzlast und damit auch die verfügbare Bandbreite können starken Schwankungen unterliegen, die sich der Kontrolle des Lernenden und genauso des Betreibers der Lernanwendung entziehen. Eine minimale Bandbreite ist trotz Einsatz von Prefetching zu gewährleisten. Deren Unterschreitung ist mit den daraus resultierenden Einschränkungen an den Lernenden zu kommunizieren.

  **Verbindungskosten** Je nach gewähltem Kostenmodell für die Rechnernetzverbindungen zwischen entfernten Ressourcen und lokalen Anwendungskomponenten können die hier entwickelten Prefetchingverfahren erhebliche Kosten verursachen, die ohne Einsatz dieser Verfahren nicht anfallen würden. Optimierungsentscheidungen der Verfahren werden völlig losgelöst von Kostenaspekten getroffen. Die Garantie für ein bestimmtes Antwortzeitverhalten hat aufgrund der Unsicherheit über den gewählten Lernpfad des Lernenden durch das Hypermedium zur Folge, dass die Möglichkeit besteht, eine große Menge von Daten anzufordern, ohne dass diese zukünftig durch den Lernenden genutzt werden. Sind die Kosten an das übertragene Datenvolumen gekoppelt, entstehen mit jeder Ladeentscheidung für das Laden im Voraus Kosten, welche sich nicht zwangsläufig amortisieren müssen. An den Lernenden sind diese Folgen von Prefetching zu kommunizieren. Hier sind Verfahren

wünschenswert, welche Kosten- und Vorhersagemodelle mit den hier dargestellten, zeitbezogen optimierenden Verfahren verbinden.

Faktoren, die aus der Gestaltung des Hypermediums resultieren, können durch Entwickler bzw. Betreiber der Lernanwendung tendenziell gut beeinflusst werden, andere Faktoren sind hingegen schlecht beeinflussbar. Sie unterliegen der Kontrolle der Lernenden oder Dritten. Indirekter Einfluss darauf ist nur durch Kommunizieren der Bedeutung dieser Faktoren auf das Antwortzeitverhalten möglich. Gegebenenfalls sind die hier konstruierten Prefetchingverfahren nur optional je nach Rahmenbedingungen anzuwenden. Auch dem Lernenden kann die Entscheidung über das Verwenden eines Prefetchingverfahrens überlassen werden. Das würde aber die Transparenz der verteilten Lernanwendung negativ beeinflussen.

### 6.2.4 Kombinationsmöglichkeiten mit anderen Prefetchingansätzen

Die hier konstruierten Verfahren unterliegen einer Vielzahl von Einschränkungen. Sie liefern aber eine wichtige Eigenschaft, die sie von anderen Prefetchingverfahren unterscheidet. Sie können auf Grundlage der getroffenen Entscheidungen und des Anwendungspotentials eines Verfahrens für einen bestimmten Navigationsbaum ein Antwortzeitverhalten im schlechtesten Fall garantieren. Die verbreiteten vorhersage- oder schwellenwertbasierten Verfahren begnügen sich damit, das Antwortzeitverhalten in durchschnittlichen Fällen zu betrachten.[21] Sie unterliegen damit nicht der Annahme, vollständige Kenntnis über die im Navigationsbaum modellierten Informationen zu besitzen.

So lassen sich kostenorientierte, meist in Form von schwellenbasierten Verfahren realisiert, vorhersagebasierte und die hier dargestellten Verfahren zur Garantie eines Antwortzeitverhaltens miteinander kombinieren.[22] Zur Illustration der Kombinationsmöglichkeiten sei angenommen, dass die Ergebnisse folgender drei Verfahren $VOR(T, s_1^i)$, $SCH(T, s_1^i)$ und $GAR(T, S, s_1^i)$ miteinander „verschmolzen“ werden sollen, um Ladeentscheidungen, basierend auf allen drei Verfahren, einem vorhersage-, einem schwellenbasierten und einem der hier konstruierten Verfahren, zu generieren. Die Verfahren erzeugen ihre Ergebnisse für einen Entscheidungszeitpunkt auf Basis der bereits durch die lokale Anwendungskomponente angeforderten Aufenthaltsknoten $\sigma_1^i$ und unter Kenntnis eines Ausschnitts des Navigationsbaums $B$ in Form des Baumausschnitts $T$. Es sind Ladeentscheidungen zum Laden im Voraus während der Nutzung des angeforderten Knotens $a_i$ zu treffen (der letzten Knoten in $s_1^i$).

- $VOR(T, s_1^i)$ liefert die Rangfolge $V$ aller Knoten aus $T$

  $VOR$ ist ein vorhersagebasiertes Verfahren, welches unter der Kenntnis bereits angeforderter Objekte der Aufenthaltsknoten $s_1^i$ sowie Kenntnis über zukünftig mögliche Anfragen $T$ die Wahrscheinlichkeit je Knoten vorhersagt, mit welcher das Ereignis eintritt,

[21] Es besteht in der Literatur teilweise der Irrglaube, dass unter der Annahme schlechtester Fälle für das Prefetchingergebnis die gelieferte Leistung dem Einsatz keines Prefetchingverfahrens gleich kommt. Für bestimmte Zielstellungen mag das zutreffen, kann aber nicht für beliebige Zielstellungen verallgemeinert werden.

[22] Vorhersage- und schwellenbasierte Verfahren wurden in Kapitel 2 im Abschnitt „Prefetchingverfahren und Prefetchingkonzepte im Überblick“, S.29 vorgestellt und eine Vielzahl von Publikationen mit konkreten Verfahren angegeben.

dass dieser Knoten zukünftig angefragt wird. Entsprechend dieser Wahrscheinlichkeiten werden die Knoten in eine Rangfolge gebracht.

- $SCH(T, s_1^i)$ liefert ausgewählte Knoten $S$ aus $T$.

  Dabei handelt es sich um jene Aufenthaltsknoten des bekannten Ausschnitts $T$ aus dem Gesamtbaum $B$, für welche es abhängig von der Bewertung mittels einer Schwellenfunktion sinnvoll erscheint, die in diesen enthaltenen Objekte im Voraus zu laden.

- $GAR(T, S, s_1^i)$ liefert mit $G$ für alle Knoten aus $S$ unter Berücksichtigung von $T$ die Zeit, mit welcher die Objekte der Knoten im Voraus zu laden sind, um ein verfolgtes, das Antwortzeitverhalten betreffendes Optimierungsziel erreichen zu können. Knoten, die nicht in $S$ aber in $T$ sind, werden nicht zum Gegenstand der Optimierung gemacht. Sie sind nicht „prefetchable“. Solche Aufenthaltsknoten erhalten eine Ladezeit von Null, so dass für diese keine Prefetchingentscheidungen generiert werden.

Es sei ein weiteres Verfahren angenommen, welches die Ergebnisse $V$, $S$ und $G$ der drei methodisch unterschiedlich ansetzenden Verfahren miteinander vereint. Dieses Verfahren lädt in der Nutzzeit von $a_i$ Objekte aus Knoten von $S$ in der mit $V$ gelieferten Reihenfolge und in der mit $G$ angegebenen Zeitdauer im Voraus. Das Resultat sind Ladeentscheidungen im durch $T$ gesteckten Rahmen, welche dazu führen, dass nur solche Objekte bzw. Teile davon im Voraus geladen werden, die

- zwingend zum Erreichen des angestrebten Antwortzeitverhaltens notwendig sind,
- unter der Schwellenbedingung nicht zu laden sind, weil sie sich zum im Voraus Laden nicht eignen,
- als am wahrscheinlichsten in ihrer zukünftigen Anfrage eingestuft wurden.[23]

Der Ausschnitt $T$ des gesamten Baums kann je nach Kenntnis der Verfahren über den Gesamtbaum eingeschränkt werden. Sinnvoll erscheint für besonders tiefe Navigationsbäume eine Einschränkung der Höhe für $T$ vorzunehmen. Dies ist auch dann sinnvoll, wenn der Baum sehr breit ist und damit die Wahrscheinlichkeit der Anfrage eines einzelnen Knotens aus $T$, geliefert von $VOR$, mit zunehmender Tiefe drastisch abnimmt.

Durch die Kombination der drei Ansätze zum Fällen von Prefetchingentscheidungen für die Lernanwendung steigt der Schwierigkeitsgrad der Entwicklung eines Gesamtverfahrens. In Kapitel 2, Abschnitt „Komponenten“, S.39 wird bereits ein grobes Schichtenmodell zur Modularisierung der einzelnen Funktionalitätsbereiche in notwendige Komponenten für eine vereinfachte Entwicklung eines solch komplexen Verfahrens vorgeschlagen. Forschungsarbeiten zur sinnvollen Umsetzung, zur Gesamtgüte, resultierend aus den Synergien der Ansätze, zu Auswirkungen auf Seiteneffekte im realen Betrieb sind durchzuführen. Weiterführende Arbeiten zur Kombination der in dieser Arbeit vorgestellten optimierenden Ansätze mit vorhersage- und schwellenbasierten Verfahren erscheinen sehr viel versprechend.

[23] Der letzte Punkt ist insbesondere dann interessant, wenn die angenommenen Nutzzeiten des Navigationsbaums im realen Betrieb unterschritten werden.

# Kapitel 7

# Resümee

*„Es gibt eine Theorie, die besagt, wenn jemals irgendwer genau herausfindet,*
*wozu das Universum da ist und warum es da ist, dann verschwindet es auf der Stelle*
*und wird durch noch etwas Bizarreres und Unbegreiflicheres ersetzt.*
*Es gibt eine andere Theorie, nach der das schon passiert ist.“*
*Douglas Adams*

Zum Abschluss der vorliegenden Arbeit sind die vorgelegten Ergebnisse zu bewerten. Dazu werden die erzielten Ergebnisse im Überblick beschrieben, anhand der im ersten Kapitel dargestellten Fragestellungen und Hypothesen sowie der Vorgehensweise der Untersuchungen bewertet, die Übertragbarkeit der Ergebnisse auf andere Anwendungsbereiche umrissen sowie ein vorausschauender Ausblick für weitere Forschungstätigkeiten gegeben.

## 7.1 Diskussion der Ergebnisse

Im Folgenden werden die Ergebnisse kurz aufgelistet, dann kritisiert und deren Bedeutung diskutiert.

### 7.1.1 Ergebnisse im Überblick

*Erstmalig werden die Themen Lernverhalten in Lernanwendungen, leistungsorientierter Entwurf und Betrieb verteilter hypermedialer Lernanwendungen und Prefetching gezielt miteinander verknüpft und deren Wechselwirkungen zueinander betrachtet.*

Die vorgestellte Klassifikation von Prefetchingverfahren liefert Ansatzpunkte zur Auswahl sinnvoll verwendbarer Prefetchingkonzepte für verteilte Lernanwendungen. Es wird herausgearbeitet, dass der aus dem Einsatz eines Prefetchingverfahrens zu erwartende Nutzen differenziert zu bewerten ist. Es bestehen zu berücksichtigende Interdependenzen zu den durch das Lernszenario gegebenen technischen Rahmenbedingungen, zur Struktur des Hypermediums und zu den Eigenschaften der Lernobjekte.

*Es wird ein formaler Rahmen entwickelt, welcher den Erfordernissen in der Entwicklung und dem Betrieb verteilter hypermedialer Lernanwendungen im Besonderen entspricht und gleichzeitig den Entscheidungsspielraum möglicher Prefetchingverfahren modelliert.*

Dazu werden drei grundlegende Entscheidungsfelder in Entwurf und Betrieb aufgezeigt. Diese werden, basierend auf einem Navigationsbaum, beschrieben, welcher sich direkt aus Entwurfsentscheidungen zur Struktur des Hypermediums sowie aus Eigenschaften der Lernobjekte ergibt. Die Darstellung von Navigationsmöglichkeiten in Form eines Baums ist in der Literatur häufig anzutreffen, jedoch mit der Unterscheidung von redundanten und nicht redundanten Bäumen, um die Problematik der zyklenbehafteten Navigationswege differenziert betrachten zu können, wird ein neuartiger Ansatz verfolgt. Mit dem vorgeschlagenen Modell ist es möglich, Entscheidungen der Entwurfsphase an Prefetchingentscheidungen des Betriebs sowie umgekehrt Prefetchingentscheidungen an Entwurfsentscheidungen zu koppeln. Das ermöglicht, Maßnahmen des Performance Engineerings und Tunings für Hypermedien in Hinblick auf Prefetching als eine Einheit zu betrachten. Weiterhin kann mit dem Modell die rechnergestützte Entscheidungsfindung bzw. die Darstellung des Entscheidungsspielraums völlig losgelöst von speziellen Technologien untersucht werden.

*Es werden verschiedene Zielstellungen zur Beeinflussung des Antwortzeitverhaltens verteilter hypermedialer Lernanwendungen mittels Prefetching betrachtet.*

Allein Kenntnisse von Navigationswegen durch das Hypermedium sowie von Lade- und Nutzzeiten der Objekte des Mediums reichen aus, um sinnvolle Aussagen über die Leistungsfähigkeit von Prefetchingverfahren treffen zu können. Die Aussagen betreffen unterschiedliche Ziele zur Beeinflussung des Antwortzeitverhaltens. Es sind Ziele gewählt worden, die den Besonderheiten in der Entwicklung verteilter Lernanwendungen besonders entsprechen.

Mit der angewendeten komparativen Analysetechnik zur Untersuchung der Problemeigenschaften und Verfahren wird insbesondere dem spekulativen Charakter von Prefetching und der Entscheidung unter Unsicherheit im Entwurf und Betrieb entsprochen. Neu ist hier die konsequente Betrachtung der Leistungsfähigkeit, entkoppelt von häufig nur ungenau bzw. nicht sinnvoll festlegbaren stochastischen Parametern bezüglich des Navigationsverhaltens. Auch wird eine Beurteilung der Leistungsfähigkeit in Abhängigkeit von der Gesamtdauer einer Session bzw. eines Lernprozesses als wesentlich herausgearbeitet. Es ergeben sich dadurch je nach Zielstellung allein von der Problemstellung abhängige obere Schranken für potentielle Optimierungsverfahren, die zwingend zu strikt kompetitiven Verfahren führen.

Erstmalig wird ein Optimierungsmodell zur Konstruktion von Prefetchingverfahren formuliert, welches allein auf das Antwortzeitverhalten im „worst-case“ abzielt. Die Minimierung der Wirkung von schlechtest möglichen Verhaltensweisen ist wiederum eine Konsequenz der Unsicherheitssituation im Entscheidungsprozess von Entwurf und Betrieb.

*Es werden praktisch sinnvoll einsetzbare Prefetchingverfahren zur Entscheidungsunterstützung für alle modellierten Optimierungsziele entwickelt, welche je nach Modelleigenschaften eine optimale bzw. approximierte Lösung liefern.*

Es wird ein allgemeiner Lösungsansatz für alle fünf Zielstellungen des Offline-Modells für nicht redundante Navigationsbäume zum Finden einer optimalen Lösung entwickelt. Der Ansatz ermöglicht es, effiziente Entscheidungsverfahren für alle fünf Zielstellungen zu entwickeln. Für die fünf Ziele des Offline-Modells, beruhend auf redundanten Navigationsbäumen, konnte

gezeigt werden, dass es als nahezu unmöglich gilt, effiziente Verfahren zur optimalen Lösung anzugeben.

Hingegen sind Approximationsverfahren, basierend auf effizient lösbaren linearen Programmen, entwickelt worden, welche Näherungslösungen mit einer von der Anzahl der Knoten des Baums abhängigen Güte liefern. Mit dem entwickelten Näherungsverfahren können sinnvoll Entwurf und Betrieb verteilter hypermedialer Lernanwendungen unterstützt werden. Es lassen sich Entwurfswerkzeuge entwickeln, mit welchen eine Bewertung der Entwurfsentscheidungen rechnerunterstützt werden kann. Auch ist es durch dynamische Anpassung der Modellparameter möglich, Lösungsergebnisse an sich ständig ändernde Gegebenheiten anzupassen. Es lassen sich „adaptive“ Prefetchingverfahren konstruieren, welche Entscheidungen abhängig von variierenden technischen Rahmenbedingungen und vom Lernverhalten generieren.

*Die Verfahren des Online-Modells lassen sich sinnvoll mit herkömmlichen Prefetchingansätzen kombinieren.*

Diese Arbeit entstand aus der Vision heraus, optimierende Verfahren zu entwickeln, die einerseits allein auf Basis eines Optimalitätskriteriums zwischen im Voraus zu ladenden und nicht im Voraus zu ladenden Daten unterscheiden, andererseits aber die Menge der im Voraus zu ladenden Daten mit Hilfe bereits erfolgreich eingesetzter Prefetchingansätze zusätzlich bewerten. Es ist gelungen, einen Prefetchingansatz zu entwickeln, der einerseits der Unsicherheitssituation besonders gerecht wird, andererseits mit stochastischen Ansätzen in Form von vorhersage- und schwellenbasierten Verfahren kombiniert werden kann und dadurch auch Entscheidungen unter Risiko anstatt allein unter Unsicherheit berücksichtigt. Das ermöglicht die Vielzahl herkömmlicher, bereits für webbasierte Anwendungen erfolgreich eingesetzten Ansätze auf die Besonderheiten verteilter hypermedialer Lernanwendungen zuzuschneiden. Damit wird in Kombination mit dem hier präsentierten Ansatz eine Vielzahl von Prefetchingansätzen auch für verteilte hypermediale Lernanwendungen gut nutzbar.

### 7.1.2 Bewertung der Ergebnisse

Mit den gelieferten Ergebnissen können die drei formulierten Hypothesen bestätigt werden: Mit Entwurf und Betrieb verteilter hypermedialer Lernanwendungen sind bei Einsatz von Prefetching eine Vielzahl von Besonderheiten zu beachten. Bisher publizierte Prefetchingkonzepte entsprechen diesen Besonderheiten nur in begrenztem Maß. Mit den hier präsentierten Ansätzen sind die Besonderheiten aufgegriffen worden und konnten zum großen Teil berücksichtigt werden.[1] Die beiden Untersuchungsziele der Arbeit sind weitestgehend erreicht worden: Es konnten Prefetchingansätze entwickelt werden, die den Besonderheiten entsprechen und die im Entwurf sowie Betrieb sinnvoll Verwendung finden können.[2] Die Ergebnisse sind aber unbedingt kritisch zu reflektieren. Eine Vielzahl von Teilproblemen musste offen bleiben. Deren Lösung erwies sich als äußerst kompliziert bzw. umfangreich. Deshalb sollen einige kritische Anmerkungen zu den Betrachtungen des Lern- und Navigationsverhaltens, zur vorgeschlagenen Klassifikation existierender Prefetchingansätze, zu gewählten Modellannahmen, zur Komplexität der formulierten Optimierungsprobleme, zu den konstruierten Entscheidungsverfahren sowie zur rechnergestützten Entscheidung im Entwurfsprozess folgen:

[1] Vgl. Abschnitt 1.1, S.4
[2] Vgl. Abschnitt 1.2, S.4

### Verknüpfung von Lern- und Navigationsverhalten

Es zeigt sich bei der Betrachtung vielzähliger empirischer Untersuchungen zum Lernverhalten in hypermedialen Lernanwendungen, dass sie nur teilweise Rückschlüsse auf das Navigationsverhalten und damit auch auf die Leistungsfähigkeit von Prefetchingverfahren in verteilten hypermedialen Lernanwendungen zulassen. Die Arbeit ist bewusst nicht als empirische Studie angelegt, um gezielt auf das Navigationsverhalten bezüglich Prefetching schließen zu können. Es sind „nur" gezielt bereits existierende empirische Studien über das Lernverhalten, Lerneffizienz und Lerneffekte ausgewählt worden, um tendenziell vorhandene Zusammenhänge zwischen Lern- und Navigationsverhalten aufzuzeigen und diskutieren zu können. Die betrachteten Studien lassen nur teilweise Schlussfolgerungen zu plausibel erscheinenden Zusammenhängen zu. Sie wurden nicht dazu konzipiert, Zusammenhänge zwischen der Dauer von Lernprozessen, Störungen während Lernprozessen und nichtlinearem Navigationsverhalten derart zu erklären, dass gezielt Effekte von und auf die Leistung von Prefetchingverfahren abgeleitet werden könnten.

### Klassifikation der Prefetchingverfahren

Um die große Zahl veröffentlichter Prefetchingansätze in Bezug auf die Verwendbarkeit für die hier beschriebene Problemstellung untersuchen zu können, war es unbedingt notwendig, eine Einteilung der vielzähligen Ansätze nach deren zu erwartenden Eigenschaften zu finden. Aufgrund der Vielfalt war es im Rahmen der Arbeit nahezu unmöglich und auch nicht gewollt, alle publizierten Verfahren und deren teilweise nur marginalen Unterschiede zu erfassen. Deshalb konnte hier nur eine grobe Unterteilung geliefert werden, welche aber für die vorgegebene Problemstellung völlig ausreichend erscheint. Es konnte nicht geprüft werden, ob jedes beliebige Prefetchingverfahren bzw. alle publizierten Verfahren in die vorgeschlagene Klassifikation eingeordnet werden können. Ein Klassifizieren der hier entwickelten, neuartigen Ansätze nach diesem Schema erscheint problematisch.

### Modellannahmen und -parametrisierung

Die hier angenommenen Modellrestriktionen sind hauptsächlich aus praktischen Problemstellungen heraus, das heißt mit dem Blickwinkel auf bestimmte Lernszenarien, entstanden. Um das zu motivieren, sind Besonderheiten der Entwicklung hypermedialer Lernanwendungen zusätzlich, ausführlich in Form eines der Praxis entnommenen Anwendungsszenarios dargestellt worden. Mögliche Restriktionen wurden nur teilweise systematisiert betrachtet. Wünschenswert ist eine Systematik möglicher Modellannahmen, das heißt eine Systematik zu sinnvollen Modellkonstanten, -parametern und Entscheidungsvariablen für Prefetchingverfahren. Eine solche konnte mit dieser Arbeit nicht geliefert werden. In der Arbeit wird jedoch eine ausführliche, aber allein nach der hier unterstellten Problemsituation bewertete Liste von möglichen Modellannahmen präsentiert.[3]

Einige Modellannahmen sind, wie bereits diskutiert, als äußert streitbar einzustufen. Dazu gehört insbesondere die Annahme über bekannte, fixe Nutzdauern der Lerninhalte als eine der zentralen Annahmen des Modells, welche die beschriebene Art und Weise der Optimierung ermöglicht.

[3] Allein eine solche umfangreiche Liste stellt schon einen Forschritt dar.

Daran streitbar ist, dass zwar einerseits von einer Unsicherheit über verwendete Navigationswege ausgegangen wird das betrifft insbesondere die Entwurfsphase andererseits aber sehr detaillierte Vorstellungen über die Nutzdauern der segmentierten Lerninhalte bestehen.[4]

**Komplexität der Optimierungsprobleme**

Durch den neuartigen Ansatz des Unterscheidens zwischen redundanten und nicht redundanten Navigationsbäumen kann die Komplexität der Optimierungsprobleme differenziert betrachtet werden. Sie entwickelt sich nicht abhängig von den fünf sich unterscheidenden Zielfunktionen des Offline-Modells, sondern ist abhängig von der Struktur des angenommenen Navigationsbaums. In der Praxis sind nicht redundante Navigationsbäume die Ausnahme, so dass im Regelfall mit schwer lösbaren Optimierungsproblemen zu rechnen ist. Wünschenswert ist eine Methode, mit welcher redundante Bäume in nicht redundante Bäume überführt werden können, ohne dass für den Optimierungsprozess wesentliche Informationen verloren gehen bzw. auf eine Näherungslösung mit unabhängig von der Baumtiefe garantierter Güte geschlossen werden kann. Hierfür wurde in der Arbeit kein Ansatz gefunden.

**Konstruierte Verfahren**

Erstmalig wird mit der Arbeit gezielt zwischen Offline-Entscheidungssituationen in der Entwurfsphase und der Online-Entscheidungssituationen während des Betriebs der Prefetchingverfahren unterschieden. Die konstruierten Offline-Verfahren lassen sich in der Online-Situation so wieder verwenden, dass die gelieferten Optimalwerte der Offline-Lösungen zwar nicht dem bestmöglichen Wert der Online-Situation entsprechen, aber Rückschlüsse auf die Leistungsfähigkeit sowie die zu erwartenden Seiteneffekte zulassen. Die vorgeschlagene Verflechtung von On- und Offline-Entscheidungssituationen ermöglicht es, Probleme des Performance Engineerings im Entwurf und des Performance Tunings im Test und Betrieb miteinander zu verknüpfen.

Bemerkenswert ist, dass die Verfahren mit schwellen- und vorhersagebasierten Ansätzen einfach kombiniert werden können. Durch die Kombination mit anderen Verfahren ist es einfach möglich, die während der Online-Entscheidungssituation in den einzelnen Entscheidungszeitpunkten generierten Offline-Lösungen einer Session über das mit den Offline-Verfahren garantierte Maß hinaus zu verbessern. Die Dauer für eine Entscheidungsfindung zu einem Entscheidungszeitpunkt wurde nicht explizit modelliert und nur äußerst knapp bei der Analyse der Verfahren diskutiert. Dafür sind zwingend zusätzliche Betrachtungen des Rechenzeitaufwands und -speicherplatzes, korrespondierend zu speziellen Implementierungsdetails, wie zum Beispiel den verwendeten Standardverfahren für die Lösung linearer Programme, notwendig. Auf eine konkrete Implementierung der vorgeschlagenen Verfahren sowie auf praktische Erprobungen zur Beurteilung der Leistungsfähigkeit und der benötigten Ressourcen wurde hier verzichtet.

Enttäuschend ist die Lösungsgüte der konstruierten Online-Verfahren in Bezug auf das jeweils Bestmögliche. Mit der Arbeit konnten keine Optimierungsverfahren für die Online-Entscheidungssituation gefunden werden, welche Lösungen garantieren, die wesentlich besser sind als schon durch die problemimmanenten Schranken vorgegeben. Aber gerade hier liegt der primäre Anwendungsbereich der Prefetchingverfahren.

[4] Genau dieser Umstand entspricht der gängigen Praxis. Es ist eigentlich nicht die vorgeschlagene Methodik, sondern das übliche praktische Vorgehen in der Entwicklung strittig.

**Entscheidungsunterstützung im Entwurf**

Die Möglichkeiten der Entscheidungsunterstützung in der Entwurfsphase wurden nur umrissen. Es wurde allein gezeigt, dass der Prozess zur Strukturierung eines Hypermediums für verteilte Lernanwendungen mit den konstruierten Verfahren sinnvoll unterstützt werden kann. Die unterschiedlichen Varianten der Unterstützung wurden aber keinesfalls systematisch betrachtet. Wünschenswert sind Verfahren, welche gezielt Vorschläge zum Abändern eines bestehenden Entwurfs unter bestimmten Zielstellungen liefern. In der Arbeit konnten nur heuristische Ansätze, die keine optimalen oder approximierten Lösungen bezüglich der Zielkriterien liefern, präsentiert werden. Es handelt sich dabei allein um Ansätze, die innerhalb eines durch eine Lösung gegebenen Rahmens gewünschte Veränderungen unter Einhalten bestimmter Eigenschaften der Lösung ermöglichen.

Ein weiterer Schwerpunkt in der Entscheidungsunterstützung des Entwurfs ist die Visualisierung von Konsequenzen von Entwurfsentscheidungen. Der ganze Bereich der sinnvollen Visualisierung von Optimierungsergebnissen wurde hier völlig außen vor gelassen, stellt aber einen wesentlichen Erfolgsfaktor für ein zukünftiges Werkzeug zur Entwurfsunterstützung dar.

Auch wurde die Anpassung von Navigationsbäumen in variierende Rahmenbedingungen nur umrissen. Die Konstruktion adaptiver Navigationsbäume, welche dann zu „adaptiven Optimierungsergebnissen“ führen, ist von wichtiger Bedeutung für den zukünftigen Einsatz der vorgeschlagenen Prefetchingverfahren.

Die vielzähligen Kritikpunkte ergeben sich insbesondere aus der Facettenvielfalt des Themas. Mit Beginn dieser Arbeit wurde der Umfang der Problemstellung unterschätzt, welches teilweise nur zu einem Ausblick auf und Anreißen von bestimmten Problemlösungen geführt hat.[5] Der Kern der Arbeit wird davon aber nicht berührt.

## 7.2 Ausblick

Zum Abschluss der Arbeit wird diskutiert, inwieweit die Ergebnisse auf andere Anwendungsbereiche übertragen werden können und welche zukünftigen Forschungsarbeiten als sinnvoll erachtet werden.

### 7.2.1 Übertragbarkeit der Ergebnisse auf andere Anwendungsbereiche

Die präsentierten Ergebnisse sind nicht allein für verteilte, hypermediale Lernanwendungen von Interesse. Sie sind insbesondere auf rechnergestützte Anwendungen übertragbar, für welche folgende Punkte zutreffen:

- Die dem Benutzer zu präsentierenden Daten sind verteilt gespeichert.

[5] Da die Arbeit mehrere Wissenschaftsgebiete bedient, ist das nicht verwunderlich. Das zeigt sich insbesondere in den Kapiteln 2 und 6.

- Mindestens eines der hier betrachteten Optimierungsziele des Offline- bzw. Online-Modells zur Gestaltung des Antwortzeitverhalten in der Anwendungsentwicklung und -betrieb ist sinnvoll.

- Die Daten bzw. deren Eigenschaften lassen sich in Form des hier beschriebenen Navigationsbaums modellieren. Dazu ist es notwendig, dass

  - die verteilt gespeicherten Daten mittels eines Hypermediums verknüpft sind,
  - die zu präsentierenden Daten sich in Form von Präsentationseinheiten zusammenfassen lassen,[6]
  - die Nutzdauer je Präsentationseinheit aufgrund des zu erwartenden bzw. beobachtbaren Navigationsverhaltens der Nutzer sinnvoll abgeschätzt werden kann,
  - die voraussichtliche Ladedauer der Präsentationseinheiten abgeschätzt werden kann und sie nicht starken Schwankungen innerhalb verhältnismäßig kurzer Zeitintervalle während des Betriebs unterliegt sowie
  - die Präsentationseinheiten mittels navigationsbezogenen Links sinnvoll verknüpft werden können.

Damit sind die Ergebnisse auch auf solche Anwendungsbereiche übertragbar, in welchen:

- verteilte Datenspeicherung zu störenden Verzögerungen während der Navigation führt,
- solchen Störungen mittels Prefetching sinnvoll begegnet werden kann,
- Navigationsmöglichkeiten mit Hilfe eines Hypermediums abgebildet werden[7] und
- schon mit der Anwendungsentwicklung genaue Vorstellungen über das Navigationsverhalten der Nutzer existieren.

Die Präsentation von Informationen in Form hypermedial strukturierter Daten findet weite Verbreitung auch in anderen Anwendungsbereichen verteilter Anwendungen. Genaue Vorstellungen über das Navigationsverhalten existieren häufig, wenn detaillierte Informationen über Nutzergruppen vorhanden sind bzw. erhoben werden können. Die Vorhersagbarkeit von Verhalten wird aber je nach konkreter Anwendungssituationen divergieren. Zum Beispiel die Unterstützung einer allgemein gehaltenen, webbasierten Firmenpräsentation erscheint aufgrund der schlecht bestimmbaren bzw. sehr heterogenen Zielgruppe ungeeignet. Dagegen sind im elektronischen Verkauf im Consumer-to-Consumer Bereich wie auch Business-to-Business Bereich eine Vielzahl geeignete Anwendungsszenarien denkbar.

[6] Zur Erinnerung: Ein Objekt eines Knotens des Navigationsbaums umfasst jeweils genau eine solche Präsentationseinheit.

[7] Sind nur lineare Verknüpfungen zum Beispiel in Form einer Filmsequenz charakteristisch für die Anwendung, dann ist der hier betriebene Aufwand nicht vertretbar. Andere Konzepte sind nutzbar.

### 7.2.2 Zukünftige Forschungsschwerpunkte

Aus der Kritik der vorgelegten Ergebnisse lassen sich Schwerpunkte für Zielsetzungen zur Bearbeitung des Themas ableiten. Folgende Untersuchungsgegenstände werden, in Reihenfolge nach ihrer Bedeutung für den zukünftig praktischen Einsatz der Ansätze, vorgeschlagen:

- Entwicklung eines Werkzeugs zum Entwurf von Hypermedien für verteilte Lernanwendungen

  Es werden Ansätze der Visualisierung von Entscheidungsspielräumen und aus Entscheidungen resultierenden Konsequenzen für den rechnergestützten Entwurf benötigt. Weiterhin sind die hier präsentierten Prefetchingansätze so zu modifizieren, dass Entwurfsentscheidungen je nach Entwurfsziel gezielt und automatisiert generiert werden können.

- Abschätzen der Leistung im Betrieb, abhängig von den softwaretechnischen Rahmenbedingungen

  Die vorgeschlagenen Verfahren sind auf einer konkreten Maschine zu implementieren. Es sind detaillierte Abschätzungen über Leistungsfähigkeit und Aufwand über die hier präsentierten rein asymptotischen Aufwandsabschätzungen hinaus notwendig. Das Echtzeitverhalten während der Laufzeit ist zu untersuchen.

- Rechnergestützte Konstruktion des Navigationsbaums

  Mit dem Entwurf des Hypermediums entstehen eine Vielzahl zusätzlicher, maschinell auswertbarer Informationen. Es ist zu untersuchen, ob und wie es möglich ist, aus Lerninhalten, Lernverhalten und die Lernende beschreibenden Daten[8] rechnergestützt einen Navigationsbaum derart zu erzeugen, wie er hier zur Entscheidungsfindung benötigt wird.

- Kombination mit anderen Verfahren

  Es ist zu prüfen, welche Kombinationen mit anderen bereits publizierten Prefetchingverfahren sinnvoll sind und wie dadurch die Leistungsfähigkeit beeinflusst wird.

- Verfeinerung der Approximationsansätze

  Die hier dargestellten Approximationsverfahren bedienen sich einer sehr einfachen Methode. Es wird vermutet, dass Näherungslösungen einer besseren, von der Knotenanzahl des Baums unabhängigen Güte für bestimmte Optimierungsziele konstruiert werden können. Auch wird angenommen, wie bereits teilweise nachgewiesen, dass in Abhängigkeit von der geforderten Güte auch von einer schweren Lösbarkeit des Approximationsproblems auszugehen ist. Das ist zu untersuchen und entsprechende Approximationsverfahren mit besserer als der hier angegebenen Güte sind zu konstruieren.

- Modellerweiterungen

  Die hier getroffenen Modellannahmen können vielfältig variiert werden. Je nach Anwendungsszenario erscheinen andere Annahmen wesentlich sinnvoller. Es ist zu prüfen, inwieweit die hier konstruierten Verfahren auch für andere Modellannahmen korrekt arbeiten bzw. welche Modifikationen einfach möglich sind, um anderen Modellannahmen

[8] Zum Beispiel aus im Entwurf erfassten Metadaten zu den einzelnen Lernobjekten oder Logdateien

gerecht zu werden. Von besonderem Interesse ist die Annahme eines beliebig großen lokalen Cachespeichers.

**Fazit**

Grundsätzlich sind wir als Anbieter von Lerninhalten und Entwickler von verteilten Lernanwendungen dazu verpflichtet, von uns, den Betreibern und den Lernenden bereitgestellte Ressourcen so effizient zu nutzen, wie es mit den gegeben Methoden und technischen Rahmenbedingungen zu einem vertretbaren Aufwand möglich ist. Die Option Prefetching einzusetzen, sollte in der professionellen Entwicklung verteilter hypermedialer Lernanwendungen eine zentralere Rolle einnehmen als das bisher der Fall ist. Diese Arbeit liefert dafür einen Beitrag.

# Anhang A

# Beispiele

## A.1 Offline-Modell

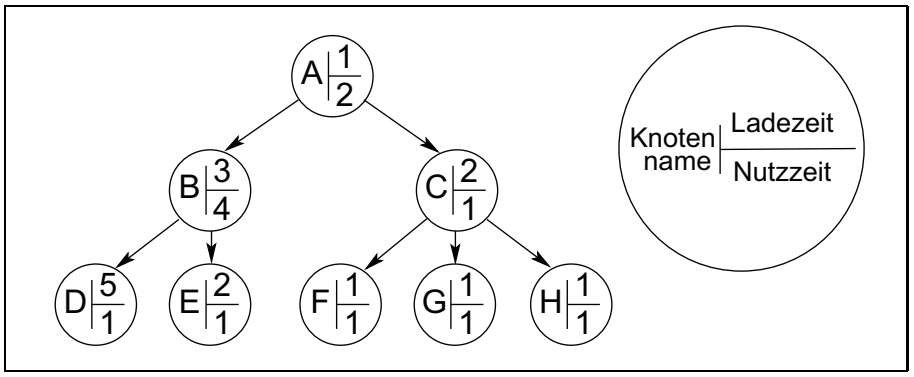


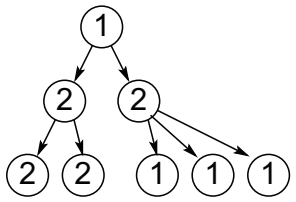

Abbildung A.1: Instanz eines nicht redundanten Navigationsbaums
Zu jedem Knoten existieren Informationen über Nutz- und Ladezeit

Aus der Probleminstanz in Abbildung A.1 ergeben sich entsprechend den Bedingungen des Offline-Modells[1] für die Optimierungsziele 2*MLB*, *MSLB*, 2*MLS*, 2*MSLS* und *MSLS* die folgenden im Baum dargestellten Restladezeiten, basierend auf den jeweils links daneben angegebenen Entscheidungsmatrizen. In den Matrizen sind nur Entscheidungswerte für $x_{ij} > 0$ angegeben. Entscheidungsoptionen mit „-“ markiert, stellen keine „echten“ Entscheidungsmöglichkeiten dar, weil die entsprechenden Objekte vorher schon genutzt wurden und im lokalen Speicher verfügbar sind.

Die Zielstellungen 2*MLB* und 2*MLS* führen zum selben Ergebnis. Auch stellt die hier angegebene Lösungsmenge eine von mehreren möglichen dieser Probleminstanz dar. Hier wurden aus der Menge optimaler Lösungen jene angegeben, mit welchen die verwendete Nutzzeit zum Laden von Objekten möglichst klein ist.[2] Dieses ist jedoch rein zufällig, weil der Aspekt nicht Optimierungsgegenstand des Modells ist.

[1] Vgl. Abschnitt 3.3.1, S.62ff.
[2] Trotzdem ist nicht immer nur eine Lösung möglich.

2*MLB* und 2*MLS*




$OPT(B) = 2$

X

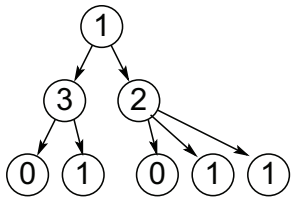
| $x_{ij}$ | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| A | - | - | - | - | - | - | - | - |
| B | 1 | - | | - | - | | | |
| C | | | - | | | - | - | - |
| D | | 3 | | - | | | | |
| E | | | | | - | | | |
| F | | | | | | - | | |
| G | | | | | | | - | |
| H | | | | | | | | - |

i

j

---

*MSLB*




$OPT(B) = 7$

X

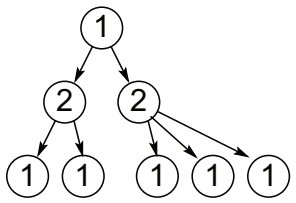
| $x_{ij}$ | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| A | - | - | - | - | - | - | - | - |
| B | | - | | - | - | | | |
| C | | | - | | | - | - | - |
| D | 2 | 3 | | - | | | | |
| E | | 1 | | | - | | | |
| F | | | 1 | | | - | | |
| G | | | | | | | - | |
| H | | | | | | | | - |

i

j

---

2*MSLS*




$OPT(B) = 4$

X

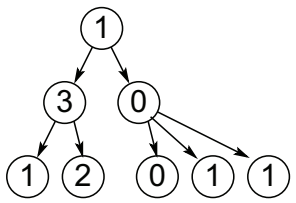
| $x_{ij}$ | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| A | - | - | - | - | - | - | - | - |
| B | 1 | - | | - | - | | | |
| C | | | - | | | - | - | - |
| D | 1 | 3 | | - | | | | |
| E | | 1 | | | - | | | |
| F | | | | | | - | | |
| G | | | | | | | - | |
| H | | | | | | | | - |

i

j

---

*MSLS*




$OPT(B) = 15$

X

| $x_{ij}$ | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| A | - | - | - | - | - | - | - | - |
| B | | - | | - | - | | | |
| C | 2 | | - | | | - | - | - |
| D | | 4 | | - | | | | |
| E | | | | | - | | | |
| F | | | 1 | | | - | | |
| G | | | | | | | - | |
| H | | | | | | | | - |

i

j

# A.2 Kompetitivitätsanalyse

## A.2.1 Drei Beispiele im Rahmen des Nachweises der 2-Kompetitivität

Zur Illustration der Entwicklung oberer und unterer Schranken eines Online-Verfahrens seien für die folgenden drei Probleminstanzen Lösungen eines optimalen Offline-Verfahrens (*OPT*) denen eines Online-Verfahrens, welches keine Objekte im voraus anfragt, gegenübergestellt.

| | Knoten in Reihenfolge der Indizierung dieser angefragt | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Bsp. 1 | | | Bsp. 2 | | | Bsp. 3 | | |
| | $a_1$ | $a_2$ | $a_3$ | $a_1$ | $a_2$ | $a_3$ | $a_1$ | $a_2$ | $a_3$ |
| $\ell$ | 0 | 1 | 1 | 0 | 1 | 4 | 4 | 1 | 0 |
| $u$ | 1 | 1 | 0 | 3 | 2 | 0 | 0 | 2 | 3 |

angefragte Knotensequenz

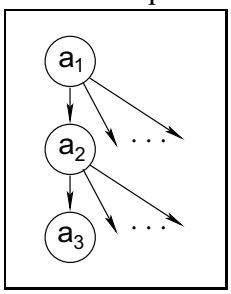



Abbildung A.2: Drei Session mit jeweils drei zueinander nicht redundanten Knoten $a_1, a_2, a_3$ unterschiedlicher Ausprägung der Nutz- und Ladezeit

Die Beispiele unterscheiden sich wie folgt voneinander:

**Beispiel 1** Identische Nutz- und Ladezeiten größer Null, ausgenommen Ladezeit des ersten und Nutzzeit des letzten Knotens der Session.

**Beispiel 2** Sich unterscheidende Lade- und Nutzzeiten. Die Nutzzeit und Ladezeiten der Vorgängerknoten ermöglichen es, jedes Objekt eines Knotens mit einer Restladezeit von Null auszuliefern.

**Beispiel 3** Beispiel drei in umgekehrter Reihefolge. Während keiner Nutzung eines Objekts kann ein anderes angefordert werden.

Die Ergebnisse der „Optimierungsverfahren“ für *OPT* und *ON* auf derfolgenden Seite:

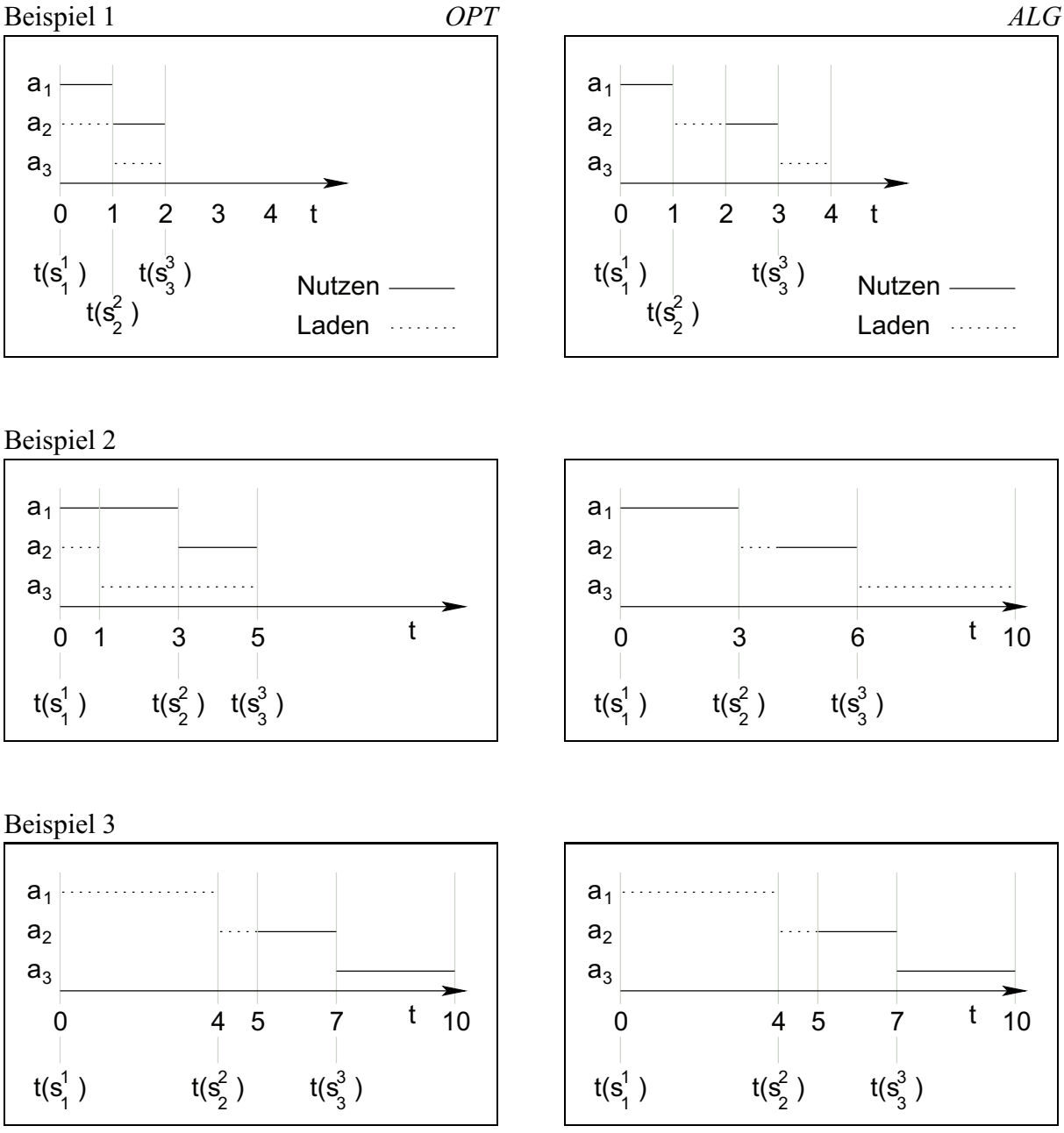



Abbildung A.3: Drei Beispiele für die Abfolge von Laden und Nutzen, *OPT* (links), *ALG* ohne paralleles Laden und Nutzen (rechts) für in Abb. A.2 vorgegebene Lade-, Nutzzeiten und Reihenfolge

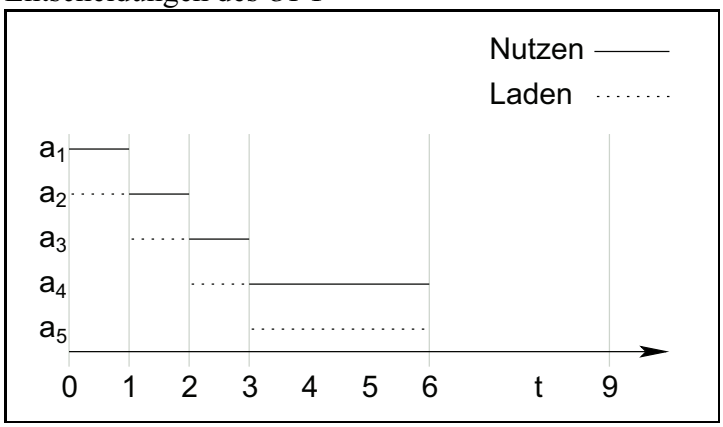
### A.2.2 Ein Beispiel im Rahmen des Nachweises der 1.5-Kompetitivität

Wird eine Baumsession angenommen, dann kann für *ON* die 1.5. Kompetitivität nachgewiesen werden. Das folgende Beispiel steht für die Ausnutzung der unteren und oberen Schranke durch *OPT* und *ON*.

Folgende Probleminstanz sei gegeben, wobei die Anfragereihenfolge der Knoten der aufsteigenden Indizierung dieser entspricht:

| | $a_1$ | $a_2$ | $a_3$ | $a_4$ | $a_5$ |
|---|---|---|---|---|---|
| $\ell$ | 0 | 1 | 1 | 1 | 3 |
| $u$ | 1 | 1 | 1 | 3 | 0 |

Entscheidungen des *OPT*




Entscheidungen des *ON*

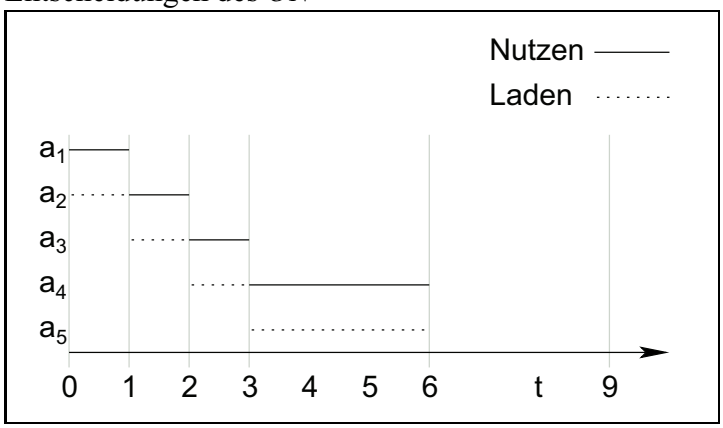



Abbildung A.4: Entscheidungen des *ON* im Vergleich zu *OPT*, die zum 1.5-fachen der Sessiondauer führen.

Im obigen Beispiel wird nochmals sichtbar, dass die Schranken nur unter Annahme von Lade- und Nutzzeiten von Null erreicht werden können.

# Anhang B

# Elementarhandlungen

Im Folgenden werden als elementar eingestufte Handlungen zum Umgang mit Aufenthaltsknoten und Navigationsbaum aufgelistet. Der asymptotische Speicherplatzaufwand des Navigationsbaums inklusive Entscheidungsmatrix (Datenobjekttyp Baum) wächst mit $O(n^2)$.

## B.1 Handlungen für Algorithmen *OFF* und *InitOFF*

Beschreibung der Algorithmen auf S.103

| Schnittstelle | Funktionalität | Zeit, Speicher |
|---|---|---|
| leer?( -> Aggregat(Typ)):WW | Liefert wahr, falls kein Element im Aggregat enthalten ist, sonst falsch. | $O(1)$, $O(n^2)$ |
| nimmAb (<-> PrioSchlange(Typ)) | Entfernt und liefert das erste Element der sortierten Schlange. | $O(1)$, $O(n)$ |
| hängeAn (<-> PrioSchlange(Typ), -> e:Typ) | Hängt ein Element an die sortierte Schlange an. | $O(1)$, $O(n)$ |
| Teilbaum ( -> B:Baum, -> i:Index):Baum | Liefert einen Teilbaum aus B, dessen Wurzelknoten Index i hat. Dieser enthält alle Kindknoten, auch indirekte Kinder, des Knotens mit Index i im Baum B. | $O(n^2)$, $O(n^2)$ |
| aktualisiereBaum (<-> B:Baum, -> T:Baum) | Überschreibt den Teilbaum in B mit den Werten aus T, welcher den selben Wurzelknoten enthält. | $O(n^2)$, $O(n^2)$ |
| KnotenAufWeg ( -> B:Baum, -> i,j:Index):Zähler | Liefert die Anzahl der Knoten des Baums, welche sich auf dem Weg zwischen Knoten mit Index i und j befinden. | $O(n)$, $O(n^2)$ |

Tabelle B.1: Elementarhandlungen für OFF

## B.2 Handlungen für Algorithmen zur Lösung von $2MLB$

Beschreibung des Algorithmus auf S.112

| Schnittstelle | Funktionalität | Zeit, Speicher |
|---|---|---|
| Anzahl<br>(BL:Liste(Baum):Zähler | Liefert die Anzahl der in BL enthaltenen Teilbäume. | $O(n)$, $O(n^2)$ |
| AnzahlMaximaRestladezeit<br>( -> BL:Liste(Baum):Zähler | Ermittelt, wie viele Knoten in der Baumliste mit dem selben maximalen Wert der Komponente r enthalten sind. | $O(n)$, $O(n^2)$ |
| MaxRestladezeit<br>( -> B:Baum):Zeit | Liefert den Wert der größten Restladezeit aller in B enthaltenen Knoten. | $O(log\ n), O(n^2)$ |
| MaxRestladezeit<br>( -> BL:Liste(Baum)):Zeit | Liefert den Wert der größten Restladezeit aller in der Baumliste enthaltenen Knoten. | $O(log\ n), O(n^2)$ |
| Max2Restladezeit<br>( -> B:Baum):Zeit | Liefert den Wert der zweitgrößten Restladezeit aller in B enthaltenen Knoten. Falls dieser Wert nicht existiert, wird Null geliefert. | $O(log\ n), O(n^2)$ |
| ReduziereMaxRestladezeit<br>(<-> T:Baum, Wert:Zeit) | Ermittelt einen Knoten j des Baums mit der größten Komponente r, subtrahiert von dieser Wert und liefert den Index dieses Knotens zurück sowie aktualisiert die Entscheidungsvariable T.a.x[j]. | $O(log\ n), O(n^2)$ |

Tabelle B.2: Zusätzliche Elementarhandlungen für R für $2MLB$

## B.3 Handlungen zur Lösung von *MSLB*

Beschreibung des Algorithmus auf S.116

| Schnittstelle | Funktionalität | Zeit, Speicher |
|---|---|---|
| fülleListe<br>(<-> L:Liste(AKnoten), -> B:Baum) | Fügt in eine leere Liste alle die Knoten des Baums B ein, deren Restladezeit $> 0$ ist. | $O(n)$, $O(n)$ |
| leer?( -> L:Liste(Element):WW | Liefert wahr, falls kein Element in der Liste enthalten ist, sonst falsch. | $O(1)$, $O(n)$ |
| ReduziereRestladezeit<br>(<-> T:(Baum), -> j:Index, -> um:Zeit) | Reduziert die Restladezeit des Knotens mit Index j um den Wert um sowie aktualisiert die Entscheidungsvariablen entsprechend für T.a.x[j]. | $O(1)$, $O(n)$ |

Tabelle B.3: Elementarhandlungen für R zur Lösung von *MSLB*

## B.4 Handlungen zur Lösung von $2MSLS$

Beschreibung des Algorithmus auf S.119

| Schnittstelle | Funktionalität | Zeit, Speicher |
|---|---|---|
| MaxSumRestladezeit<br>( -> B:Baum):Zeit | Liefert den Wert der größten Summe aller Restladezeiten je Sequenz aller Sequenzen des Baums. | $O(n)$, $O(n)$ |
| SeqMaxSumRestladezeit<br>( -> B:Baum):<br>Liste(Liste(AKnoten)) | Liefert alle Sequenzen in einer Liste, die den Maximalwert erreichen. Die Knoten, exklusive der Wurzel von B, jeder dieser Sequenzen werden in einer Liste, entsprechend ihrer Reihenfolge, in dieser Reihenfolge jeweils vermerkt. | $O(n^2)$, $O(n^2)$ |
| KnotenJeSeqMinRest<br>( -> sL:Liste(Liste(AKnoten)))<br>:Liste(AKnoten) | Liefert je Liste in sL höchstens einen Knoten. Dieser Knoten ist der Knoten einer Sequenz, welcher von den Knoten, deren Restladzeit größer Null ist, sich in der Liste am „weitesten vorn“ befindet. Die Knoten jeder Liste in sL sind nach der Reihenfolge ihres Auftretens in der Sequenz angeordnet. | $O(n^2)$, $O(n^2)$ |
| entferneDublikate<br>(<-> L:Liste(AKnoten)) | Liefert eine Knotenliste, in welcher jeder Knoten aus L nur genau einmal vorkommt. | $O(n \log n)$, $O(n)$ |
| MinRestladezeit<br>(<-> L:Liste(AKnoten)) | Liefert den Wert der kleinsten Restladezeit eines Knotens aus L. | $O(n)$, $O(n)$ |
| MaxSumRestladezeitOhne<br>( -> L:Liste(AKnoten),<br>-> B:Baum):Zeit | Liefert den Maximalwert entsprechend MaxSumRestladezeit, jedoch nur für Sequenzen aus B, welche nicht Knoten aus L enthalten. | $O(n^2)$, $O(n)$ |
| entnimm<br>(<-> L:Liste(Element)):Element | Liefert ein beliebiges Element aus aL und entfernt dieses aus aL. | $O(n)$, $O(n)$ |
| ReduziereRestladezeit<br>(<-> T:(Baum),<br>-> j:Index, -> um:Zeit) | Reduziert die Restladezeit des Knotens mit Index j um den Wert um sowie aktualisiert die Entscheidungsvariablen entsprechend für T.a.x[j]. | $O(1)$, $O(n)$ |

Tabelle B.4: Elementarhandlungen für R zur Lösung von $2MSLS$

## B.5 Handlungen zur Lösung von *MSLS*

Beschreibung des Algorithmus auf S.126

| Schnittstelle | Funktionalität | Zeit, Speicher |
|---|---|---|
| fülleSchlange<br>(<-> S:Schlange(Element),<br>-> B:Baum) | Fügt sortiert in eine leere Schlange Elemente für alle Knoten des Baums B mit einer Restladezeit größer Null ein, wobei jedes Element den Index eines Knotens sowie sein Niveau im Baum enthält. Sortiert sind die Elemente absteigend nach ihrem Niveau in B. | $O(n\ log\ n)$, $O(n)$ |
| leer?( -> S:Schlange(Element):WW | Liefert wahr, falls kein Element in der Schlange enthalten ist, sonst falsch. | $O(1)$, $O(n)$ |
| ReduziereRestladezeit<br>(<-> T:(Baum),<br>-> j:Index, -> um:Zeit) | Reduziert die Restladezeit des Knotens mit Index j um den Wert um sowie aktualisiert die Entscheidungsvariablen entsprechend für T.a.x[j]. | $O(1)$, $O(n)$ |
| Blätter( -> B:Baum):Zahl | Liefert die Anzahl der Blätter eines Baums. | $O(n)$, $O(n)$ |

Tabelle B.5: Elementarhandlungen für R zur Lösung von *MSLS*

## B.6 Handlungen für *ALG* und *transformiereMatrix* des Online-Verfahrens

Beschreibungen der Algorithmen sind auf S.149 („Einfache Ladestrategie“) und S.163 transformiereMatrix(...) zu finden.

| Schnittstelle | Funktionalität | Zeit, Speicher |
|---|---|---|
| liefereTeilbaumAus<br>( -> B:Baum,<br>-> L:Liste(Index):Baum | Liefert einen Teilbaum aus B mit genau dem Wurzelknoten, zu welchem der Weg L in B, beginnend mit der Wurzel aus B, führt. | $O(n^2)$, $O(n^2)$ |
| liefereDirekteKinder<br>( -> B:Baum,<br>-> i:Index):Liste(Index) | Liefert die Indizes der direkten Nachbarknoten des Knotens mit Index i von Baum B. | $O(n)$, $O(n^2)$ |
| entnimm<br>( -> L:Liste(Index)):Index | Entfernt ein beliebiges Element aus der Liste und liefert dieses zurück. | $O(1)$, $O(n)$ |
| ReduziereRestladezeit<br>(<-> T:(Baum),<br>-> j:Index, -> um:Zeit) | Reduziert die Restladezeit des Knotens mit Index j um den Wert um sowie aktualisiert die Entscheidungsvariablen entsprechend für T.a.x[j]. | $O(1)$, $O(n)$ |
| aktualisiereBaum<br>(<-> B:Baum, -> T:Baum) | Überschreibt den Teilbaum in B mit den Werten aus T, welcher den selben Wurzelknoten enthält. | $O(n^2)$, $O(n^2)$ |
| liefereRedundanteKnoten<br>( -> j:Index,<br>-> B:Baum):Liste(Index) | Liefert die Indizes aller zu Knoten mit Index j redundanten sowie in B enthaltenen Knoten inklusive j. | $O(n)$, $O(n^2)$ |
| liefereMinSummeRestnutzzeit<br>( -> L:Liste(Index),<br>-> B:Baum):Zeittyp | Ermittelt jeweils die Summe der Restladezeiten auf allen Wegen von der Wurzel aus B bis zu jedem Knoten mit Index aus L und liefert den kleinsten Wert der Summen zurück. | $O(n^2)$, $O(n^2)$ |
| AnzElemente<br>( -> r:Reihe(Element)):Zahl | Liefert die Anzahl der Elemente von Reihe r. | $O(1)$, $O(n)$ |
| leer?( -> L:Liste(Element):WW | Liefert wahr, falls kein Element in L ist, ansonsten falsch zurück. | $O(1)$, $O(n)$ |

Tabelle B.6: Elementarhandlungen für ALG und transformiereMatrix

# Anhang C

# Umformungen

## C.1 Umformen von $D_{ON}$ durch Einsetzen von $\beta$

Umformung verwendet auf S.152

$$\beta = \frac{\sum_{i=1}^{m} \ell_i}{\sum_{i=1}^{m} u_i}$$

$$\begin{aligned}
D_{ON}(s_1^m) &\leq \sum_{i=2}^{m} \left( u_i + \ell_i - \left\lfloor u_{i-1} \frac{1}{k_{max}} \right\rfloor \right) + \ell_1 + u_1 \\
&\leq \sum_{i=1}^{m} u_i + \sum_{i=1}^{m} \ell_i - \sum_{i=1}^{m-1} \left( \left\lfloor u_i \frac{1}{k_{max}} \right\rfloor \right) \\
&\leq \left( \sum_{i=1}^{m} u_i \right) + \beta \left( \sum_{i=1}^{m} u_i \right) - \left\lfloor \frac{1}{k_{max}} \sum_{i=1}^{m-1} u_i \right\rfloor \quad | \sum_{i=1}^{m} \ell_i = \beta \sum_{i=1}^{m} u_i \\
&\leq (1+\beta) \sum_{i=1}^{m} u_i - \left\lfloor \frac{1}{k_{max}} \sum_{i=1}^{m-1} u_i \right\rfloor
\end{aligned}$$

## C.2 Umformung von $\frac{D_{ON}(s_1^m)}{D_{OPT}(s_1^m)}$ für Fall 1

Folgende Umformung wird auf 152ff. verwendet.

Zur Umformung wird ein Korrekturfaktor $f$ eingeführt: $f = \frac{1}{k_{max}} \sum_{i=1}^{m-1} u_i - \left\lfloor \frac{1}{k_{max}} \sum_{i=1}^{m-1} u_i \right\rfloor$

Folgende Umformungsschritte wurden durchgeführt:

$$\left\lfloor \frac{1}{k_{max}} \sum_{i=1}^{m-1} u_i \right\rfloor = \frac{1}{k_{max}} \sum_{i=1}^{m-1} u_i - f$$

Weil $\beta \geq 1$ für Fall 1 gilt:

$$\frac{D_{ON}(s_1^m)}{D_{OPT}(s_1^m)} \leq \frac{\sum_{i=1}^{m-1} u_i + \beta \sum_{i=1}^{m-1} u_i - \left[ \frac{1}{k_{max}} \sum_{i=1}^{m-1} u_i - f \right]}{\beta \sum_{i=1}^{m-1} u_i} = \frac{1}{\beta} + 1 - \frac{1}{\beta k_{max}} + \frac{f}{\beta \sum_{i=1}^{m-1} u_i}$$

$$z_1 = \frac{f}{\beta \sum_{i=1}^{m-1} u_i} = \frac{\frac{1}{k_{max}} \sum_{i=1}^{m-1} u_i - \left\lfloor \frac{1}{k_{max}} \sum_{i=1}^{m-1} u_i \right\rfloor}{\sum_{i=1}^{m-1} u_i}$$

$$\lim_{\sum_{i=1}^{m-1} u_i \to \infty} (z_1) = \lim_{k_{max} \to \infty} (z_1) = 0$$

$$\frac{D_{ON}(s_1^m)}{D_{OPT}(s_1^m)} \leq \frac{1}{\beta} + 1 - \frac{1}{\beta k_{max}} + z_1 \approx \frac{1}{\beta} + 1 - \frac{1}{\beta k_{max}} \leq 2$$

## C.3 Umformung von $\frac{D_{ON}(s_1^m)}{D_{OPT}(s_1^m)}$ für Fall 2

Folgende Umformung wird auf S.153 verwendet.
Weil $\beta \leq 1$ für Fall 2 gilt:

$$\frac{D_{ON}(s_1^m)}{D_{OPT}(s_1^m)} \leq \frac{(1+\beta)\sum_{i=1}^{m-1} u_i - \left\lfloor \frac{1}{k_{max}} \sum_{i=1}^{m-1} u_i \right\rfloor}{\sum_{i=1}^{m-1} u_i}, \left\lfloor \frac{1}{k_{max}} \sum_{i=1}^{m-1} u_i \right\rfloor \leq \beta \sum_{i=2}^{m} u_i \tag{C.1}$$

Diese Gleichung kann wie folgt umgeformt werden, so dass die Abhängigkeit des Quotienten von $\beta$ und $k_{max}$ sichtbar wird:

$$\frac{D_{ON}(s_1^m)}{D_{OPT}(s_1^m)} = 1+\beta - \frac{\left\lfloor \frac{1}{k_{max}} \sum_{i=1}^{m-1} u_i \right\rfloor}{\sum_{i=1}^{m-1} u_i} = 1+\beta - \frac{\frac{1}{k_{max}} \sum_{i=1}^{m-1} u_i - f}{\sum_{i=1}^{m-1} u_i} = 1+\beta - \frac{1}{k_{max}} + \frac{f}{\sum_{i=1}^{m-1} u_i} \tag{C.2}$$

Zur Vereinfachung wird zu eine Korrekturfunktion in Abhängigkeit von $k_{max}$ und $\beta$ eingeführt:

$$z_2(k_{max},\beta) = \frac{1}{k_{max}} - \frac{f}{\sum_{i=1}^{m-1} u_i} = \frac{1}{k_{max}} - \frac{\frac{1}{k_{max}} \sum_{i=1}^{m-1} u_i - \left\lfloor \frac{1}{k_{max}} \sum_{i=1}^{m-1} u_i \right\rfloor}{\sum_{i=1}^{m-1} u_i} = \frac{\left\lfloor \frac{1}{k_{max}} \sum_{i=1}^{m-1} u_i \right\rfloor}{\sum_{i=1}^{m-1} u_i} \leq 1 \tag{C.3}$$

$$\lim_{\sum_{i=1}^{m-1} u_i \to \infty} \left( \frac{\left\lfloor \frac{1}{k_{max}} \sum_{i=1}^{m-1} u_i \right\rfloor}{\sum_{i=1}^{m-1} u_i} \right) = \frac{1}{k_{max}} \tag{C.4}$$

und in C.2 eingesetzt:

$$\begin{aligned} &\frac{D_{ON}(s_1^m)}{D_{OPT}(s_1^m)} \leq 1+\beta - z_2(k_{max},\beta), z_2(k_{max},\beta) \leq \beta \leq 1, z_2 \approx \frac{1}{k_{max}} \\ &z_2(k_{max},\beta) \leq \beta \end{aligned} \tag{C.5}$$

# Literaturverzeichnis


[Abd98] ABDULLA, G.: *Analysis and Modeling of World Wide Web Traffic*, Virginia Polytechnic Institute and State University, Blacksburg, VA., Diss., 1998

[ADW01] ANDERSON, C. ; DOMINGOS, P. ; WELD, D.: Adaptive web navigation for wireless devices. In: *Proceedings of the 17th International Joint Conference on Artifical Intelligence*, 2001

[AFJ99] ARLITT, Martin ; FRIEDRICH, Rich ; JIN, Tai: Workload Characterization of a Web Proxy in a Cable Modem Environment. In: *ACM SIGMETRICS Performance Evaluation Review* 27 (1999), Nr. 5, S. 25 36

[AJ00] ARLITT, M. ; JIN, T.: Workload Characterization of the 1998 World Cup Web Site. In: *IEEE Network Magazine* 14 (2000), Nr. 3, S. 30 37

[Ale01] ALEXANDER, S.: E-learning Developments and Experiences. In: *Education and Training* 43 (2001), S. 240 248

[Alo04] ALOMYAN, Hesham: Individual Differences: Implications for Web-based Learning Design. In: *International Education Journal, Educational Research Conference 2003 Special Issue* 4 (2004), Nr. 4, S. 188 195

[ALR99] ALLENDER, Eric ; LOUI, Michael C. ; REGEAN, Kenneth W.: Reducibility and Completeness. In: *Mikhail J. Atallah (Hrsg.): Algorithms and Theory of Computation Handbook*. London : CRC Press LLC, 1999

[Ang03] ANGERMANN, Michael: Differences in Cost and Benefit of Prefetching in Circuit-Switched and Packet-Switched Networks. In: *10th International Conference on Telecommunications IEEE ICT'2003*. Tahiti, Papeete, 2003

[Ast01] ASTLEITNER, Hermann: Web-basierte Lernumgebungen: Forschung und Praxis. In: *Handbuch Hochschullehre* 1.20 (2001), S. 1 28

[Ata99] ATALLAH, Mikhail J.: *Algorithms and Theory of Computation Handbook*. London : CRC Press LLC, 1999

[AZN99] ALBRECHT, David W. ; ZUKERMAN, Ingrid ; NICHOLSON, Ann E.: Presending Documents on the WWW: A Comparative Study. In: *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence IJCAI*, 1999, S. 1274 1279

[BB96] BALLIN, Dieter ; BRATER, Michael: *Dieter Blume (Hrsg.): Handlungsorientiert Lernen mit Multimedia*. Nürnberg : Bildung und Wissen, 1996

[BB98] BERGNER, K. ; BRUEGGE, B.: Teaching and Learning with the Internet. In: *Proceedings of the International Conference Software Engineering: Education and Practice*. Dunidin, New Zealand, 1998, S. 26 29

[BB01] BERENDT, Bettina ; BRENSTEIN, Elke: Visualizing individual differences in Web navigation: STRATDYN, a tool for analyzing navigation patterns. In: *Behavior Research Methods, Instruments, and Computers* 33 (2001), Nr. 2, S. 243 257

[BBK00] BHATTI, Nina ; BOUCH, Anna ; KUCHINSKY, Allan: Integrating user-perceived quality into Web server design. In: *Proceedings of the 9th International World Wide Web Conference*. Amsterdam, 2000

[BE98] BORODIN, Allan ; ELYANIV, Ran: *Online Computation and Competetive Analysis*. Cambridge : Cambridge University Press, 1998

[Bes96] BESTAVROS, Azer: Speculative Data Dissemination and Service to Reduce Server Load, Network Traffic and Service Time for Distributed Information Systems. In: *Proceedings of the 1996 International Conference on Data Engineering (ICDE-96)*. New Orleans, Louisiana, 1996

[BGJ+05] BULTERMAN, Dick ; GRASSEL, Guido ; JANSEN, Jack ; KOIVISTO, Antti ; LAYAIDA, Nabil ; MICHEL, Thierry ; MULLENDER, Sjoerd ; ZUCKER, Daniel: Synchronized Multimedia Integration Language (SMIL 2.0) Specification. 2005. Forschungsbericht. W3C Empfehlung. verfügbar unter http://www.w3.org/TR/2005/REC-SMIL2-20050107/Paper (Abruf 1.4.2005) bzw. auf CD unter BGJ+05.html

[BGT81] BLAND, R. ; GOLDFARB, D. ; TODD, M.J.: The ellipsoid method: a survey. In: *Operations Research* Bd. 29, 1981, S. 1039 1091

[BHS02] BERENDT, Bettina ; HOTHO, Andreas ; STUMME, Gerd: Towards semantic web mining. In: *International Semantic Web Conference 2002 (ISWC02)*. Sardinien, Italien, 2002

[BKK03] BOURAS, Christos ; KONIDARIS, Agisilaos ; KOSTOULAS, Dionysios: Efficient Reduction of Web Latency through Predictive Prefetching on a WAN. In: *Fourth International Conference on Web-Age Information Management*. Chengdu/China, 2003

[BKK04] BOURAS, Christos ; KONIDARIS, Agisilaos ; KOSTOULAS, Dionysios: Predictive Prefetching on the Web and Its Potential Impact in the Wide Area. In: *World Wide Web archive* 7 (2004), Nr. 2, S. 11 30

[BL01] BARRY, C. ; LANG, M.: A Survey of Multimedia and Web Development Techniques and Methodology Usage. In: *IEEE Multimedia 8*, 2001, S. 52 60

[Blu98] BLUMENSTENGEL, Astrid: *Entwicklung hypermedialer Lernsysteme*. Berlin : Wissenschaftlicher Verlag, 1998

[BMB05] BMBF: Report des Bundesministeriums für Bildung und Forschung: Angebote zur digitalen Hochschullehre. Bonn : Bundesministerium für Bildung und Forschung, 2005. Forschungsbericht

[BP99] BARNES, J. F. ; PANDEY, Raju: CacheL: Language Support for Customizable Caching Policies. In: *Proceedings of the 4th International Web Caching Workshop*, 1999

[BP02] BOLCHINI, Davide ; PAOLINI, Paolo: Capturing Web Apllications: New issues for Conceptual Modeling. In: *Proceedings of 5th Workshop on Requirements Engineering*. Valencia, 2002

[BPR03] BOLCHINI, Davide ; PAOLINI, Paolo ; RANDAZZO, Giovanni: Adding Hypermedia Requirements to Goal-Driven Analysis. In: *Proceedings of the 11th IEEE International Requirements Engineering Conference (RE'03)*. Monterey Bay, USA, 2003, S. 127 137

[Bra86] BRADY, James T.: A theory of productivity in the creative process. In: *IEEE Computer Graphics and Applications* 6 (1986), Nr. 5, S. 25 34

[Bre96] BRENSTEIN, Elke: Untersuchungsmöglichkeiten von Lernverhalten in hypermedialen Lernumgebungen / Interdisziplinäres Zentrum für Lern- und Lehrforschung Potsdam. 1996. Forschungsbericht. LLF-Berichte, Nr. 16, S.45-55. Potsdam

[Bre04] BREMER, Claudia: E-Learning-Strategien im Spannungsfeld von Hochschulentwicklung, Kompetenzansätze und Anreizsysteme. In: *Claudia Bremer und Kerstin E. Kohl (Hrsg.): E-Learning-Strategien und E-Learning-Kompetenzen an Hochschulen* (2004), S. 9 32

[BRS03] BADACH, Anatol ; RIEGER, Sebastian ; SCHMAUCH, Matthias: *Web-Technologien*. München : Hanser, 2003

[Bru04] BRUSILOVSKY, P.: KnowledgeTree: A Distributed Architecture for Adaptive E-Learning. In: *Proceedings of the 14th International World Wide Web Conference*. New York, 2004

[BS00] BERENDT, Bettina ; SPILIOPOUPOU, Myra: Analysis of navigation behaviour in web sites integrating multiple information. In: *The Very Large Database Journal* 9 (2000), S. 56 75

[Bul04] BULTERMAN, Dick C.: Engineering Aspects of Web Hypermedia: Examples and Lessons from the GRiNS Editor. In: *Proceedings of the ACM Hypertext 2004*. Santa Cruz, 2004

[BYE03] BAEZA-YATES, Ricardo ; ENCALADA, Blanco: Evolution of the Web Structure. In: *The Twelfth International World Wide Web Conference*. Budapest, Hungary, 5 2003

[CB98] CROVELLA, Mark ; BARFORD, Paul: The Network Effects of Prefetching. In: *Proceedings of the Infocom 1998. IEEE*. San Francisco, CA, 1998

[CDF+98] CACERES, R. ; DOUGLIS, F. ; FELDMANN, A. ; GLASS, G. ; RABINOVICH, M.: Web proxy caching: the devil is in the details. In: *ACM Performance Evaluation Review* 26(3) (1998), S. 11 15

[CDK02] COULOURIS, George ; DOLLIMORE, Jean ; KINDBERG, Tim: *Verteilte Systeme*. München : Pearson Studium, 2002

[CF00] CHEN, S. ; FORD, N.: Individual Differences, hypermedia navigation, and learning: An empirical study. In: *Journal of Educational Multimedia and hypermedia* 9 (2000), Nr. 4, S. 281 311

[CFB+03] CERI, Stefano ; FRATERNALI, Piero ; BONGIO, Aldo ; BRAMBILLA, Marco ; COMAI, Sara ; MATERA, Maristella: *Designing Data-Intensive Web Applications*. Amsterdam : Morgan Kaufmann, 2003

[CH03] CHEN, Xuan ; HEIDEMANN, John: Preferential Treatment for Short Flows to Reduce Web Latency. In: *Computer Networks* 41 (2003), Nr. 6, S. 779 794

[CHR00] CLOSE, R. ; HUMPHREYS, R. ; RUTTENBUR, B.: E-learning and Knowledge Technology. In: *Sun Trust Equitable Securities* (2000), S. 1 195

[CK00] COHEN, Edith ; KAPLAN, Haim: Prefetching the Means for Document Transfer: A New Approach for Reducing Web Latency. In: *INFOCOM (2)*, 2000, S. 854 863

[CK01a] COHEN, Edith ; KAPLAN, Haim: Proactive Caching of DNS Records: Addressing a Performance Bottleneck. In: *Proceedings of the 2001 Symposium on Applications and the Internet (SAINT 2001)*, 2001, S. 85

[CK01b] COHEN, Edith ; KAPLAN, Haim: Refreshment Policies for Web Content Caches. In: *INFOCOM*, 2001, S. 1398 1406

[CKR99] COHEN, Edith ; KRISHNAMURTHY, Balachander ; REXFORD, Jennifer: Efficient Algorithms for Predicting Requests to Web Servers. In: *INFOCOM (1)*, 1999, S. 284 293

[CKZ03] COHEN, Edith ; KAPLAN, Haim ; ZWICK, Uri: Connection caching: model and algorithms. In: *Source Journal of Computer and System Sciences archive* 67 (2003), Nr. 1, S. 92 126

[CLR98] CORMEN, Thomas H. ; LEISERSON, Charless E. ; RIVEST, Ronald L.: *Introduction to Algorithms*. 20th. Cambridge Massachusetts : MIT Press, 1998

[CM99] CHANDRU, Vijay ; M.R.RAO: Linear Programming. In: *Mikhail J. Atallah (Hrsg.): Algorithms and Theory Computation Handbook* (1999), S. 31/1 31/37

[CM02] CHEN, S.Y. ; MACREDIE, R.D.: Cognitive styles and hypermedia navigation: Development of a learning model. In: *Journal of the American Society for Information Science and Technology* (2002), Nr. 53, S. 3 15

[CSM97] COOLEY, R. ; SRIVASTAVA, J. ; MOBASHER, B.: Web Mining: Information and Pattern Discovery on the World Wide Web. In: *Proceedings of the 9th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'97)*, 1997

[CT02] CASTELEYN, Sven ; DE TROYER, Olga: Exploiting Link Types during the Web Site Design Process to Enhance Usability of Web Sites. In: *2nd International-Workshop on Web Oriented Software Technology*. Malaga, 2002

[CW99] CHIU, B. C. ; WEB, G.: Using decision trees for agent modeling: improving prediction performance. In: *User Modeling and User-Adapted Interaction* Bd. 8, 1999, S. 131 152

[CY97] CHINEN, Kenichi ; YAMAGUCHI, Suguru: An interactive prefetching proxy server for improvement of WWW latency. In: *Proceedings of the Seventh Annual Con- ference of the Internet Society (INET'97)*. Kuala Lumpur, 1997

[CZ03] CHEN, Xin ; ZHANG, Xiaodong: A Popularity-Based Prediction Model for Web Prefetching. In: *IEEE Computer Society* Bd. 36, 2003, S. 63 70

[CZB00] CAO, Pei ; ZHANG, Jin ; BEACH, Kevin: Active Cache: Caching Dynamic Contents on the Web. In: *International Conference on Distributed Systems Platforms and Open Middleware*. Hudson River Valley, New York, 2000, S. 373 388

[Dav99] DAVIDSON, Brian D.: Web traffic logs: An imperfect resource for evaluation. In: *Proceedings of th INET'99 Conference*, 1999

[Dav02a] DAVIDSON, Brian D.: *The Design and Evaluation of Web Prefetching and Caching Techniques*. New Jersey, Graduate School New Brunswick RutgersState, University of New Jersey, Diss., 2002

[Dav02b] DAVIDSON, Brian D.: Predicting Web Actions from HTML Content. In: *Proceedings of the Thirtheenth ACM Conference on Hypertext and Hypermedia*, 2002

[Dav04] DAVISON, Brian: Learning Web Request Patterns. In: *A. Poulovassilis and M. Levene (Hrsg.): Web Dynamics*. Boston : Springer, 2004, S. 435 460

[DFKM97] DOUGLIS, Fred ; FELDMANN, Anja ; KRISHNAMURTHY, Balachander ; MOGUL, J.: Rate of change and other metrics: a live study of the world wide web. In: *Proceedings of the 1st USENIX Symposium on Internet Technologies and Systems*, 1997

[DHNS04] DOLOG, Peter ; HENZE, Nicola ; NEJDL, Wolfgang ; SINTEK, Michael: Personalization in distributed e-learning environments. In: *Proceedings of the 13th international World Wide Web Conference*. New York, NY, USA, 2004, S. 170 179

[DJX02] DONGSHAN, Xing ; JIAOTONG, Xian ; XIAN, Shen J.: A New Markov Model for Web Access Prediction. In: *Computing in Science and Engineering* 4 (2002), Nr. 6, S. 34 39

[DMF97] DUSKA, Bradley M. ; MARWOOD, David ; FREELEY, Michael J.: The Measured Access Characteristics of World-Wide-Web Client Proxy Caches. In: *Proceedings of the 1997 Usenix Symposium on Internet Technologies and Systems (USITS-97)*. Monterey, CA, 1997

[DP96] DINGLE, Adam ; PARTL, Tomas: Web cache coherence. In: *Proceedings Computer Networks and ISDN Systems*, 1996, S. 28(7 11):907{920}

[DS01] DÖRR, Günther ; STRITTMATTER, Peter: Multimedia aus pädagogischer Sicht. In: *L.J.Issing and P.Klimsa (Hrsg.): Information und Lernen mit Multimedia* (2001), S. 29 42

[DT82] DOHERTY, Walter J. ; THADANI, Ahrvind J.: The economic value of rapid response time / IBM. White Plains, NY, USA, 1982. technical report GE20-0752-0

[Duc99] DUCHAMP, Dan: Prefetching Hyperlinks. In: *Proceedings USENIX Internet Technologies & Systems 99*, 1999

[EJM00] EDEN, Avinoam N. ; JOH, Brian W. ; MUDGE, Trevor: Web Latency Reduction Via Client-Side Prefetching. In: *Proceedings of the International Symposium on Performance Analysis of Systems and Software*, 2000, S. 193 200

[ELW04] ENGELS, Gregor ; LOHMANN, Marc ; WAGNER, Annika: Entwicklungsprozess von Web-Anwendungen. In: *G. Kappel (Hrsg.): Web-Engineering* (2004), S. 239 263

[Epp99] EPPLER, Martin J.: Qualitätsstandards- Ein Instrument zur Sicherung der Informationsqualität in Multmedia-Produktionen. In: *Oliver Merx (Hrsg.): Qualitätssicherung bei Multimediaprojekten* (1999), S. 129 149

[ESGS98] EL-SADDIK, Abdulmotaleb ; GRIWODZ, Carsten ; STEINMETZ, Ralf: Exploiting User Behaviour in Prefetching WWW Documents. Oslo : Springer, 1998, S. 302 311

[EW04] EBNER, Arno ; WERTHNER, Hannes: Betrieb und Wartung von Web-Anwendungen. In: *G. Kappel (Hrsg.): Web-Engineering* (2004), S. 187 206

[FC02] FORD, Nigel ; CHEN, Sherry Y.: Individual Differences, hypermedia Navigation and Learning: An Emprirical Study. In: *Journal of Educational Multmedia and Hypermedia* Bd. 9(4), 2002, S. 281 311

[FCD+99] FELDMANN, Anja ; CACERES, Ramon ; DOUGLIS, Fred ; GLASS, Gideon ; RABINOVICH, Michael: Performance of Web proxy caching in heterogeneous bandwidth environments. In: *Proceedings of the INFOCOM '99 conference*, 1999

[FCJ99] FAN, Li ; CAO, Pei ; JACOBSON, Quinn: Web Prefetching Between Low Bandwith Clients and Proxies: Potential and Performance. In: *Measurement and Modeling of Computer Systems (SIGMETRICS '99)*. Atlanta, GA, 1999, S. 178 187

[FGM+97] FIELDING, R. ; GETTYS, J. ; MOGUL, J. ; FRYSTYK, H. ; BERNERS-LEE, T.: Hypertext Transfer Protocol, HTTP 1.1, http://www.w3.org/Protocols/rfc2068/rfc2068.txt (Abruf 20.12.04). 1997. Forschungsbericht. (auf CD unter RFC2068.txt)

[FMK02] FRIAS-MARTINEZ, E. ; KARAMCHETI, V.: A Prediction Model for User Access Sequences. In: *International Conference on Knowledge Discovery and Data Mining (ACM SIGKDD)*, 2002

[For01] FORSYTH, I.: *Teaching and Learning Materials and the Internet*. 3. Aufl. UK : Kogan Page, 2001

[Fou96] FOURER, L.B.J.: Software for Optimization: A Buyer's Guide (Parts I and II). In: *INFORMS Computer Science* 17(1/2) (1996)

[FR04] FREIRE, Manuel ; RODRÃGUEZ, Pilar: A graph-based interface to complex hypermedia structure visualization. In: *Proceedings of the working conference on Advanced visual interfaces*. Gallipoli, Italy, 2004, S. 163 166

[Fri01] FRIESEN, Norm: What are Educational Objects? In: *Interactive Learning Environments* 9 (2001), 10, Nr. 3

[FS00] FOYGEL, Dan ; STRELOW, Dennis: Reducing Web Latency with Hierarchical Cache-Based Prefetching. In: *Proceedings of the International Workshop on Scalable Web Services (ICPP 2000)*. Toronto, 2000

[FS02] FRIESEN, Karsten ; SCHMITZ, Hans: Applying Software Engineering Managing the Development of an E-Learning Product. In: *Proceedings of the International NAISO Congress on Networked Learning in a Global Environment, Challenges and Solutions for Virtual Education*, ICSC-NAISO Academic Press, Canada/The Netherlands, 5 2002

[FS03] FISHER, D. ; SAKSENA, G.: Link prefetching in Mozilla: A server-driven approach. In: *Proceedings of the Eighth International Workshop on Web Content Caching and Distribution*. Hawthorne, NY, 2003

[FW03] FELDMANN, Martin ; WAGNER, Ralf: Strukturieren mit Multitrees: Ein Fachkonzept zur verbesserten Navigation in Hypermedia. In: *Wirtschaftsinformatik* 45 (2003), Nr. 6, S. 589 598

[GCP00] GOMEZ, Jaime ; CACHERO, Christina ; PASTOR, Oscar: Extending a Conceptual Modelling Approach to Web Application Design. In: *Proceedings of the 12th International Conference on Advanced Information Systems*. Valencia, Spain : Springer-Verlag, 2000, S. 79 93

[GJ79] GAREY, Michael R. ; JOHNSON, David S.: *Computers and Intractability*. Murray Hill, New Jersey : Bell Laboratories, 1979

[Gla01] GLASSON, B.: E-Education: Two Challenges presented in the panel section e-Education: Problems, Challenges, Solutions. In: *Fourtheenth Bled Electronic Commerce Conference*. Bled, Slovenia, 2001

[GM01] GINIGE, Athula ; MURUGESAN, San: The essence of web engineering - managing the diversity and complexity of web application development. In: *IEEE Multimedia* 8 (2001), Nr. 2, S. 22 25. ISSN 1070 986X

[Gra04] GRAF, Maja: *eModeration, Lernende im Netz begleiten*. Bern : hep, 2004

[Gre93] GREENBERG, H.J.: *A Computer-Assisted Analysis System for Mathematical Programming Models and Solutions*. Boston : Kluwer, 1993

[GSV00] GÜELL, Natacha ; SCHWABE, Daniel ; VILAIN, Patricia: Modeling Interactions and Navigation in Web Applications. In: *Proceedings of the Workshops on Conceptual Modeling Approaches for E-Business and The World Wide Web and Conceptual Modeling*, Springer, 2000, S. 115 127

[GZ01] GELENBE, Erol ; ZHU, Qi: Adaptive control of pre-fetching. In: *Performance Evaluation* 46 (2001), Nr. 2-3, S. 177 192

[Han04] HAND, Klaus G.: *Kursbuch eLearning 2004: Produkte aus dem Förderprogramm*. Bonn : Bundesministerium für Bildung und Forschung, 2004

[Har99] HARASIM, L.: A Framework for Online learning: The Virtual-U1. In: *Computers*, 1999, S. 44 49

[HBD00] HALL, Richard H. ; BALESTRA, Joel ; DAVIS, Miles: A Navigational Analysis of Linear and Non-Linear Hypermedia Interfaces / American Educational Research Association. New Orelans, LA, 2000. Forschungsbericht

[Her02] HERDER, E.: Metrics for the Adaptation of Site Structure. In: *Proceedings of the German Workshop on Adaptivity und User Modeling in Interactive Systems*. Hanover, 2002, S. 22 26

[HG97] HEWETT, W.G. ; GOODWIN, C.: The Use of Internet Technology to Enhance the Learning Experience for Students and Provide a Richer Teaching Environment. In: *Proceedings of the 3rd Pacific Asia Conference on Information Systems*. Brisbane, 1997, S. 347 356

[HG00] HE, D. ; GÖKER, A.: Detecting Session Boundaries from Web User Logs. In: *Proceedings of of the BCS-IRSG 22nd Annual Colloquium on Information Retrieval Research*, 2000

[HL99] HENTENRYCK, Pascal V. ; LUSTIG, Irvin: *The Opl Optimization Programming Language*. Cambridge : MIT Press, 1999

[HRBA99] HONG, O K. ; ROBERT, Fiona ; BIUK-AGHAI, P.: A Web Prefetching Model Based on Content Analysis. In: *Proceedings of Macau IT Congress 1999*. Macau, 1999, S. 61 66

[IK02] ISSING, L. J. ; KLIMSA, P.: *Information und Lernen mit Multimedia*. Weinheim : Beltz, 2002

[Isk02] ISKE, Stefan: *Vernetztes Wissen. Hypertext Strategien im Internet*. Bielefeld : Bertelsmann, 2002

[IX00] IBRAHIM, Tamer I. ; XU, Cheng-Zhong: Neuronal net based pre-fetching to tolerate WWW latency. In: *Proceedings of the 20th International Conference on Distributed Computing Systems (ICDCS2000)*, 2000

[JC98] JACOBSON, Quinn ; CAO, Pei: Potential and Limits of Web Prefetching Between Low-Bandwidth Clients and Proxies. In: *Proceedings of the Third International WWW Caching Workshop*. Manchester, 1998

[JFM97] JOACHIMS, Thorsten ; FREITAG, Dayne ; MITCHELL, Tom M.: Web Watcher: A Tour Guide for the World Wide Web. In: *Proceedings of the Fifteenth International Conference on Artificial Intelligence IJCAI*, 1997, S. 770 777

[JK98] JIANG, Zhinei ; KLEINROCK, Leonard: An adaptive network prefetch scheme. In: *IEEE Journal on Selected Areas in Communications* Bd. 16, 1998, S. 358 368

[JK00] JOSHI, A. ; KRISHNAPURAM, R.: On Mining Web Access Logs. In: *Proceedings of ACM SIGMOD Workshop on Research Issues in Datamining and Knowledge Discovery*. Dallas, Texas, 2000, S. 63 69

[JL04] JARCHOW, Silvia ; LABADIE, Christian: Hat E-Learning bzw. Multimedia eine längerfristige Perspektive in der Hochschullehre? Erfahrungen zur Hochschulentwicklung durch digitale Medien aus dem IMUNHO-Projekt der Universität Bremen. In: *C. Bremer, K. E. Kohl (Hrsg.): E-Learning Strategien und E-Learning-Kompetenzen an Hochschulen* (2004), S. 139 153

[JLC00] JUNG, Jaeyeon ; LEE, Dongman ; CHON, Kilnam: Proactive Web caching with cumulative prefetching for large multimedia data. In: *Proceedings of the 9th international World Wide Web conference on Computer networks*. Amsterdam, 2000, S. 645 655

[Jon96] JONASSEN, D.H.: *Handbook of research for educational communications and technology (Hrsg.)*. New York : Simon and Schuster Macmillan, 1996

[JWS02] JANG, Yingyin ; WU, Min-You ; SHU, Wei: Web Prefetching: Costs, Benefits and Performance. In: *7th International Workshop on Web Content Caching and Distribution (WCW)*. Boulder, Colorado, 2002

[KA03] KALAYDIJEV, Ognian ; ANGELOVA, Galia: Adaptive Hypermedia in eLearning. 2003. Forschungsbericht

[Kel99] KELTER, U.: Ein Referenzmodell für multimediale Lehr- und Lernsysteme. In: *M. Nagl, A. Behle, B. Westfechtel, H. Balzert, C. Weidauer, H.-W. Six, P. Pauen, J. Voss, W. Schäfer, J. Wadsack, U. Kelter (Hrsg.): Softwaretechnische Anforderungen an multimediale Lehr- und Lernsysteme, Report* (1999), S. S.113 145. www.informatik.fernuni-hagen.de/ import/pi3/PDFs/SMIL.pdf, Abruf 1.4.2005

[Ker99] KERRES, Michael: Didaktische Konzeption multimedialer und telemedialer Lernumgebungen. In: *HMD - Praxis der Wirtschaftsinformatik* 36(1) (1999), S. 9 21

[Ker00] KERRES, Michael: Computergestütztes Lernen als Element hybrider Lernarrangements. In: *Rudolf Kammerl (Hrsg.): Computergestütztes Lernen*. München : Oldenbourg Verlag, 2000, S. 23 39

[Ker01a] KERRES, Michael: Mediendidaktische Professionalität bei der Konzeption und Entwicklung technologiebasierter Lernszenarien. In: *Herzig, Bardo (Hrsg): Medien machen Schule. Grundlagen, Konzepte und Erfahrungen der Medienbildung* (2001)

[Ker01b] KERRES, Michael: *Multimediale und telemediale Lernumgebungen*. 2. Aufl. München : Oldenbourg Verlag, 2001

[Ker01c] KERRES, Michael: Technische Aspekte multi- und telemedialer Lernangebote. In: *L.J.Issing and P.Klimsa (Hrsg.): Information und Lernen mit Multimedia*. Weinheim: Beltz : Oldenbourg Verlag, 2001, S. 19 27

[KJ99] KERRES, Michael ; JECHLE, Thomas: Hybride Lernarrangements: Personale Dienstleistungen in multimedialen und telemedialen Lernumgebungen. In: *Jahrbuch Arbeit - Bildung - Kultur* Bd. 17, 1999, S. 21 39

[KJ01] KERRES, Michael ; JECHLE, Thomas: Didaktische Konzeption des Tele-Lernens. In: *L.J.Issing and P.Klimsa (Hrsg.): Information und Lernen mit Multimedia*. Weinheim: Beltz : Oldenbourg Verlag, 2001, S. 267 282

[KK02] KOCH, Nora ; KRAUS, Andreas: The Expressive Power of UML-based Web Engineering. In: *D. Schwabe, O. Pastor, G. Rossi, and L. Olsina (Hrsg.): Proceedings of the 2nd International Workshop Web-Oriented Software Technology*. Malaga, Spain, 2002

[KL97] KROEGER, Thomas M. ; LONG, Darrell D. E.: Exploring the Bounds of Web Latency Reduction from Caching and Prefetching. In: *Proceedings of the Symposium on Interworking Systems and technologies USNIX*, 1997, S. 13 22

[Kle99] KLEMM, Reinhard P.: WebCompanion: A friendly client-side Web prefetching agent. IEEE Transactions on Knowledge and Data Engineering. In: *Journal of Educational Computing Research* 11(4) (1999), S. 577 594

[KLM97] KROEGER, Thomas M. ; LONG, Darrell D. E. ; MOGUL, Jeffrey C.: Exploring the Bounds of Web Latency Reduction from Caching and Prefetching. In: *USENIX Symposium on Internet Technologies and Systems*, 1997

[KM72] KLEE, V. ; MINTY, G.J.: How good is the simplex algorithm? In: *O. Sisha(Hrsg.): Inequalities III, Academic Press*, 1972

[KM04] KOPER, Rob ; MANDERVELD, Jocelyn: Educational modelling language: modelling reusable, interoperablke, rich and personalised units of learning. In: *British Journal of Educational Technology* 35 (2004), Nr. 5, S. 537 551

[KPRR04] KAPPEL, Gerit ; PRÖLL, Birgit ; REICH, Siegfried ; RETSCHITZEGGER, Werner: *Web Engineering*. Heidelberg : dpunkt.verlag, 2004

[KR01] KRISHNAMURTHY, Balachander ; REXFORD, Jennifer: *Web Protocols and Practice: HTTP 1.1, Networking Protocols, Caching, and Traffic Measurements*. Boston : Addison-Wesley, 2001

[KT01] KHAN, Javed I. ; TAO, Quingping: Partial Prefetch for Faster Surfing in Composite Hypermedia. In: *3rd USENIX Symposium on Internet Technologies and Systems (USITS 2001)*. San Francisco, 2001, S. 13 24

[KT04] KUNDT, Günther ; TAMM, Jan: Erfahrungen mit der Lernsoftware Anwendung statistischer Methoden in der klinisch experimentellen Forschung. In: *S. Pöppl and J. Bernauer (Hrsg.): Rechnergestützte Lehr- und Lernsysteme in der Medizin. Proceedings zum 8. Workshop der GMDS AG Computergestützte Lehr- und Lernsysteme in der Medizin*. Lübeck : Shaker-Verlag, 2004, S. 185 191

[KTS05] KUNDT, Günther ; TAMM, Jan ; SANDER, Martin: Die Lernsoftware ¨ Anwendung statistischer Methoden in der klinisch experimentellen Forschung¨ für die Aus- und Weiterbildung in Medizinischer Biometrie. In: *S. Pöppl and J. Bernauer (Hrsg.): Rechnergestützte Lehr- und Lernsysteme in der Medizin. Proceedings zum 9. Workshop der GMDS AG Computergestützte Lehr- und Lernsysteme in der Medizin*. Lübeck : Shaker, 2005

[KV98] KRISHNAN, P. ; VITTER, Jeffrey S.: Optimal Prediction for Prefetching in the Worst Case. In: *Society for Industrial and Applied Mathematics 006* 27 (1998), S. 1617 1636

[Lan02] LANG, Michael: Hypermedia Systems Development: Do We Really Need New Methods? In: *Proceedings of the Informing Science and IT Education Conference (IS2002)*. Cork, Ireland, 2002, S. S.883 891

[Lan04] LANG, Michael: A Critical Review of Challenges in Hypermedia Systems Development. In: *Information Systems Development Advances in Theory, Practice and Education 13th International Conference on Information Systems Development, ISD'2004 Proceedings* 7 (2004), Nr. 1, S. 51 69

[LB97] LOON, Tong S. ; BHARGHAVAN, Vaduvur: Alleviating the Latency and Bandwidth Problems in WWW Browsing. In: *Usenix Symposium on Internet Technologies and Systems*. Monterey, 1997

[LBO99] LAN, Bin ; BRESSAN, Stephane ; OOI, Beng C.: Making Web servers pushier. In: *Proceedings of the Workshop on Web Usage Analysis and User Profiling*. San Diego, CA, 1999

[LD97] LEI, Hui ; DUCHAMP, Dan: An Analytical Approach to File Prefetching. In: *USENIX Annual Technical Conference*. Anaheim, 1997

[LH99] LEE, M.J. ; HARVEY, F.: The releationships between navigational patterns and information processing styles of hypermedia users. In: *Proceedings of Selected Research and Development Papers Presented at the National Convention of the Association for Educational Communications und Technologie*. Houston, Texas, 1999

[Lie95] LIEBERMAN, Henry: Letizia: An Agent That Assists Web Browsing. In: *Proceedings International Joint Conference on Artifical Intelligence (IJCAI 95)*, AAAI Press, 1995

[Lie97] LIEBERMAN, Henry: Autonomous interface agents. In: *Proceedings of the ACM SIGCHI'97 Conference on Human Factors in Computing Systems*. Atlanta, GA, 1997

[LK98] LAWLESS, Kimberley A. ; KULIKOWICH, Jonna M.: Domain knowledge, interest, and hypertext navigation: A study of individual differences. In: *Journal of educational computing research* 7 (1998), Nr. 1, S. 51 69

[LL02] LOU, Wenwu ; LU, Hongjun: Efficient Prediction of Web Accesses on a Proxy Server. In: *Proceedings of the eleventh international conference on Information and knowledge management*. McLean, Virginia, USA, 2002, S. 169 176

[LM03] LEMPEL, Ronny ; MORAN, Shlomo: Predictive Caching and Prefetching of Query Results in Search Engines. In: *The Twelfth International World Wide Web Conference (WWW2003)*. Budapest, 2003

[LP01] LYTRAS, Miltiadis D. ; POULOUDI, Nancy: Exanding e-learning effectiveness. The shift from content orientation to knowledge managment utilization. In: *Word Conference on Educational Multimedia, Hypermedia and Telecommunications (ED-MEDIA 2001)*. Tampere, Finnland, 2001

[LWP+01] LIM, Lipyeow ; WANG, Min ; PADMANABHAN, Sriram ; VITTER, Jeffrey S. ; AGARWAL, Ramesh: Characterizing Web Document Change. In: *Lecture Notes in Computer Science* 2118 (2001), S. 133 146

[MA02] MENACE, Daniel A. ; ALMEIDA, Virgilio A. F.: *Capacity Planing for Web Services*. Upper Saddle River, New Jersey : Prentice Hall, 2002

[Mac99] MACGREGOR, S. K.: Hypermedia Navigation Profiles: Cognitive Characteristics and Information Processing Strategies. In: *Journal of Educational Computing Research* 20(2) (1999), S. 189 206

[Mel96] MELARA, Gloria: Investigating learning styles on different Hypertext environments: Hierachical-like and network-like structures. In: *Journal of educational computing research* 14 (1996), Nr. 4, S. 313 328. ISSN 0735 6331

[Mer99] MERX, Oliver: *Qualitätssicherung bei Multimedia-Projekten*. Berlin : Springer, 1999

[Mit97] MITCHELL, Tom M.: *Machine Learning*. Upper Saddle River, New Jersey : McGraw-Hill, 1997

[MS02] MAYR, Peter ; SEUFERT, Sabine: *Fachlexikon e-Learning: Wegweiser durch das e-Vokabular*. Bonn : Gerhard May Verlags GmbH, 2002

[Mur00] MURUGESAN, San: Web Engineering for Successful Software Development, Tutorial. In: *Asia Pacific Web Conference*. Xian, China, 10/2000

[NBW+99] NAGL, M. ; BEHLE, A. ; WESTFECHTEL, B. ; BALZERT, H. ; WEIDAUER, C. ; SIX, H.-W. ; PAUEN, P. ; VOSS, J. ; SCHÄFER, W. ; WADSACK, J. ; U. KELTER, U.: Studie über Softwaretechnische Anforderungen an multimediale Lehr- und Lernsysteme / Forschergruppe Softtec NRW. 1999. Forschungsbericht. (auf CD unter NBW+99.pdf)

[NCO04] NTOULAS, Alexandros ; CHO, Junghoo ; OLSTON, Christopher: What s New on the Web? The Evolution of the Web from a Search Engine Perspective. In: *Proceedings of the ACM WWW 2004 Conference*. New York, 2004

[NGBS+97] NIELSEN, Henrik F. ; GETTYS, Jim ; BAIRD-SMITH, Anselm ; PRUDHOMMEAUX, Eric ; LIE, Hakon W. ; LILLEY, Chris: Network Performance Effects of HTTP/1.1, CSS1, and PNG. In: *Proceedings of the ACM SIGCOMM 97 Conference*. Cannes, France, 1997

[NHHM+04] NIEGEMANN, H.M. ; HESSEL, S. ; HOCHSCHEID-MAUEL, D. ; ASLANSKI, K. ; DEICHMANN, M. ; KREUZBERGER, G.: *E-Learning Kompendium*. Berlin : Springer, 2004

[NKM03] NANOPOULOS, A. ; KATSAROS, D. ; MANOLOPOULOS, Y.: A Data Mining Algorithm for Generalized Web Prefetching. In: *IEEE Transactions on Knowledge and Data Engineering* 15 (2003), Nr. 5, S. 11 30

[NZA98] NICHOLSON, Ann E. ; ZUKERMAN, Ingrid ; ALBRECHT, David W.: A Decision-Theoretic Approach for Pre-sending Information on the WWW. In: *Pacific Rim International Conference on Artificial Intelligence*, 1998, S. 575 586

[OR99] OUGHTON, J. ; REED, W.: The effect of hypermedia knowledge and learning style on student-centered concept maps about hypermedia. In: *Journal of Research on Computing in Education* 32 (1999), Nr. 3, S. 366 384

[Ore00] OREN, Nir: A Survey of prefetching techniques / University of Aberdeen, Scotland, UK. 2000. Forschungsbericht. (auf CD unter Ore00.pdf)

[OW02] OTTMANN, Thomas ; WIDMAYER, Peter: *Algorithmen und Datenstrukturen*. 4. Aufl. Heidelberg : Spektrum, 2002

[Pad95] PADMANABHAN, Venkata N.: Improving World Wide Web Latency / UC Berkeley. 1995 ( Report UCB/CSD-95-875). Forschungsbericht. (auf CD unter Pad95.pdf)

[PD03] PETERSON, Larray Y. ; DAVIE, Bruce S.: *Computer Networks A System Approach*. 3. Aufl. San Francisco : Morgan Kaufmann, 2003

[Pet99] PETERS, Heinz: Qualitätssicherung im Bereich des Computer-Based Training. In: *Interacting with ComputersOliver Merx (Hrsg.): Qualitätssicherung bei Multimediaprojekten* (1999), S. 243 254

[Pet00] PETERS, Otto: Ein didaktisches Modell für den virtuellen Lernraum. In: *Uwe Sanders (Hrsg.): Zum Bildungswert des Internet* (2000), S. 159 187

[PHMaZ00] PEI, Jian ; HAN, Jiawei ; MORTAZAVI-ASL, Behzad ; ZHU, Hua: Mining Access Patterns Efficiently from Web Logs. In: *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 2000, S. 396 407

[Pit99] PITKOW, James E.: Summary of WWW characterization. In: *WWW Journal* 2 (1999), S. 3 13

[PM96] PADMANABHAN, Venkata N. ; MOGUL, Jeffrey C.: Using Predictive Prefetching to Improve World-Wide Web Latency. In: *Computer Communication Review. Proceedings of the ACM Special Interest Group on Data Communications Conference (SIGCOMM '96)* 26 (1996), Nr. 3, S. S.22 36

[PM99a] PALPANAS, Themistoklis ; MENDELZON, Alberto: Web Prefetching Using Partial Match Prediction. In: *Proceedings of the 4th International Web Caching Workshop*, 1999

[PM99b] PALPANAZ, Themistokelis ; MENDELZON, Alberto: Web prefetching using partial match prediction. In: *Proceedings of the 4th International Web Caching Workshop*, 1999

[PMC99] PETERSON, R.W. ; MAROSTICA, M.A. ; CALLAHAN, L.M.: E-learning: helping Investors Climb the E-learning Curve / U.S. Bancorp Piper Jaffray Equity Research Report. 1999. Forschungsbericht

[PP99] PIROLLI, Peter L. ; PITKOW, James E.: Distributions of surfers' paths through the World Wide Web. In: *World Wide Web* 2 (1999), S. 29 45

[PRW04] PARKINSON, Adrian ; REDMOND, James A. ; WALSH, Cathal: Accommodating field-dependence: a cross-over study. In: *Proceedings of the 9th annual SIGCSE conference on Innovation and technology in computer science*. Leeds, UK, 2004, S. 72 76

[PS99] PAUEN, P. ; SIX, H.-W.: Defizite und Lösungsansätze. In: *M. Nagl, A. Behle, B. Westfechtel, H. Balzert, C. Weidauer, H.-W. Six, P. Pauen, J. Voss, W. Schäfer, J. Wadsack, U. Kelter (Forschergruppe SoftTec NRW): Softwaretechnische Anforderungen an multimediale Lehr- und Lernsysteme, Report* (1999), S. S.146 154. (auf CD unter NRW+99.pdf)

[PW99] PHILLIPS, Steven ; WESTBROOK, Jevery: On-line Algorithms: Competitive Analyse and Beyond. In: *Mikhail J. Atallah (Hrsg.): Algorithms and Theory of Computation Handbook* (1999), S. 10 1 10 22

[QB99] QUENTIN-BAXTER, Megan: Quantitative evidence for differences between learners making use of passive hypermedia learning environments. In: *ACM Computing Survey* 31 (1999), Nr. 4, S. 26

[RAL96] REED, W. M. ; AYERSMAN, David J. ; LIU, Min: The Effects of Students Computer-based Prior Experiences and Instructional Exposured on the Apllication of Hypermedia-Related Mental Models. In: *Journal of Educational Computing Research* 14 (1996), Nr. 2, S. 185 207

[RBP98] RAMSAY, Judith ; BARBESI, Alessandro ; PREECE, Jenny: A psychological investigation of long retrieval times on the World Wide Web. In: *Interacting with Computers* 10 (1998), S. 77 86

[RCM+96] REVEL, Dan ; COWAN, Crispin ; MCNAMEE, Dylan ; PU, Calton ; WALPOLE, Jonathan: An Architecture for Flexible Multimedia Prefetching. In: *Kevin Jeffay (Hrsg.): IEEE Real-Time Systems Symposium Workshop on Resource Allocation Problems in Multimedia Systems*. Washington, DC, 1996

[RHJ99] RAGGET, Dave ; HORS, Arnaud L. ; JACOBS, Ian: HTML 4.01 Specification.W3C Recommendation / W3C. 1999. Forschungsbericht. (auf CD unter RHJ99.txt)

[RKFH02] RISER, U. ; KEUNEKE, J. ; FREIBICHLER, H. ; HOFFMANN, B.: *Konzeption und Entwicklung interaktiver Lernprogramme*. Upper Saddle River, New Jersey : Springer, 2002

[Roa98] ROAST, Chris: Designing for delay in interactive information retrieval. In: *Interacting with Computers* 10 (1998), S. 87 104

[Röc05] RÖCK, Hans: *Arbeitsmaterial Vorlesung Grundlagen der Wirtschaftsinformatik, Teil I, Kapitel 3, Lehrstuhl für Wirtschaftsinformatik, Universität Rostock*. 2005. (auf CD unter Roe05.pdf)

[Röl03] RÖLL, Franz J.: *Pädagogik der Navigation: Selbstgesteuertes Lernen durch Neue Medien*. München : Kopaed, 2003

[RS99] RAUTENSTRAUCH, C. ; SCHOLZ, A.: Improving the Performance of a Database-based Information system. A Hierarchical Approach. In: *Proceedings of the 17th International Conference of the Association of Management*. San Diego, CA, 1999, S. S.153 160

[RS02] RODRIGUES, Rogerio F. ; SOARES, Luiz Fernando G.: Framework for Prefetching Mechanisms in Hypermedia Presentations. In: *Proceedings of the Fourth International Symposium on Multimedia Software Engineering*. New Port, California, USA, 2002, S. 278 285

[RS03] RODRIGUES, Rogerio F. ; SOARES, Luiz Fernando G.: Inter and intra media-object QoS provisioning in adaptive formatters. In: *Proceedings of the 2003 ACM Symposium on Document Engineering*. Grenoble, France, 2003, S. 78 87

[Sar00] SARUKKAI, Ramesh R.: Link Prediction and Path Analysis Using Markov Chains. In: *9th WWW Confernce* Bd. 33. Amsterdam, 2000, S. S.377 386

[SCB04] STACH, Natalia ; CRISTEA, Alexandra I. ; BRA, Paul D.: Authoring of Learning Styles in Adaptive Hypermedia: Problems and Solutions. In: *13th International World Wide Web Conference*. New York, 17.-22. Mai 2004

[Sch99] SCHADER, Michael: Lernerprofilierung - Wissen, was der Lerner weiss. In: *Proceedings GI-Workshop-Tage Lernen Wissensentdeckung and Adaptivität*. Magdeburg, 1999

[Sch00] SCHRIJVER, Alexander: *A Theory of Linear and Integer Programming*. Weinheim : John Wiley and Sohns, 2000. Reprint

[Sch01] SCHOLZ, Andre: *Performance-orientierte Systementwicklung am Beispiel datenbankbasierter integrierter Anwendungssysteme*. Aachen, Universität Magdeburg, Institut für Technische und Betriebliche Informationssysteme, Diss., 2001

[Sch02] SCHULMEISTER, Rolf: Taxonomie der Interaktivität von Multimedia. In: *it+ti* Bd. 4(2002), 2002, S. 193 199

[Sch03] SCHULMEISTER, Rolf: *Lernplattformen für das virtuelle Lernen. Evaluation und Didaktik*. München, Wien : Oldenbourg Verlag, 2003

[SDMML03a] SEGURA-DEVILLECHAISE, Marc ; MENAUD, Jean-Marc ; MULLER, Gilles ; LAWALL, Julia L.: Web cache prefetching as an aspect: towards a dynamic-weaving based solution. In: *Proceedings of the 2nd international conference on Aspect-oriented software development*. Boston, 2003, S. 110 119

[SDMML03b] SEGURA-DEVILLECHAISE, Marc ; MENAUD, Jean M. ; MULLER, Gilles ; LAWALL, Julia L.: Web cache prefetching as an aspect: towards a dynamic-weaving based solution. In: *Proceedings of the 2nd international conference on Aspect-oriented software development*, ACM Press, 2003. ISBN 1 58113 660 9, S. 110 119

[SH03] SEN, Rituparna ; HANSEN, Mark H.: Predicting aWeb user's next request based on log data. In: *Journal of Computational and Graphical Statistics* 12 (2003), Nr. 1

[SHF03] STEINKE, Matthias ; HUK, Thomas ; FLOTO, Christian: The Process of Learning with Hypermedia Systems: Linking Learner Characteristics, Software Design and Log Files. In: *IWF Knowledge and Media* (2003), S. S. 2744 2747

[SK99] SZABO, Michael ; KANUKA, Heather: Effects of Violating Screen Design Principles of Balance, Unity, and Focus on recall Learning, Study Time, and Completion Rates. In: *Journal of Educational Multmedia and Hypermedia* Bd. 8(1), 1999, S. 23 42

[SK04] SCHWINGER, Wieland ; KOCH, Nora: Modellierung von Webanwendungen. In: *Gerti Kappel, Birgit Pröll, Siegfried Reich und Werner Retschitzegger (Hrsg.): Web Engineering*. Heidelberg : dpunkt.verlag, 2004

[SLH04] SIMOES, D. ; LUÃS, R. ; HORTA, N.: Enhancing the SCORM Metadata Model. In: *Proceedings of the 14th International World Wide Web Conference*. New York, 2004

[SOBB03] SOW, Daby M. ; OLSHEFSKI, David P. ; BEIGI, Mandis ; BANAVAR, Guruduth: Prefetching based on Web usage mining. In: *M. Endler and D. Schmidt (Hrsg.): Middleware 2003*, 2003, S. 262 281

[SP03] SAFRONOV, Victor ; PARASHAR, Manish: Optimizing Web servers using Page rank prefetching for clustered accesses. In: *Information Science* 150 (2003), Nr. 3-4, S. 165 176

[Spi99] SPILIOPOULOU, Myra: A Data Miner analyzing the Navigational Behaviour of Web Users. In: *Principles of Data Mining and Knowledge Discovery*, 1999, S. 588 589

[SR00] SWAMINATHAN, N. ; RAGHAVAN, S. V.: Intelligent prefetching in WWW using client behavior characterization. In: *Proceedings of the Eighth International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)*, 2000

[ST85] SLEATOR, Daniel D. ; TARJAN, Robert E.: Amortized efficiency of list update and paging rules. In: *Communication of the Association for Computing Machinery*, 1985, S. 28(2):202 208

[SW02] SMITH, Connie U. ; WILLIAMS, Lloyd G.: *Performance Solutions*. Boston : Pearson Education, 2002

[Tam05] TAMM, Jan: Prefetching für verteilte, hypermediale Lernanwendungen unter Berücksichtigung der Lerndauer, 2005. angenommen zur Fachtagung Delphi der Gesellschaft für Informatik, September 2005, Rostock, auf CD unter Tam05.pdf

[Tan02] TANENBAUM, Andrew: *Distributed Systems*. Upper Saddle River, New Jersey : Pearson Education, 2002

[Tan03] TANENBAUM, Andrew: *Computer Networks*. 4. Aufl. Upper Saddle River, New Jersey : Pearson Education, 2003

[Thi02] THISSEN, Frank: *Screen-Design Handbuch*. 2. Aufl. Berlin : Springer, 2002

[TKG02] TAMM, Jan ; KUNDT, Günther ; GIERL, Lothar: Entwicklung eines Methodenlehre-Baukasten zur Unterstützung der Biometrieausbildung. In: *Biometrie und Epidemiologie in Medizin und Biologie* Bd. 33(2002), 2002, S. 132

[TKG03] TAMM, Jan ; KUNDT, Günther ; GIERL, Lothar: Lernsoftware Anwendung statistischer Methoden in der klinisch experimentellen Forschung. In: *Biometrie und Epidemiologie in Medizin und Biologie (GMDS2003)* Bd. 34(2003), 2003, S. 389 390

[TKG04] TAMM, Jan ; KUNDT, Günther ; GIERL, Lothar: Pre-Fetching-Verfahren für Entwurf und Betrieb webbasierter, hypermedialer Lernanwendungen. In: *Proceedings GMDS 2004*. Innsbruck, 2004

[TVB+04] TATTERSALL, Colin ; VOGTEN, Hubert ; BROUNS, Francis ; KOPER, Rob ; VAN ROSMALEN, Peter ; SLOEP, Peter ; VAN BRUGGEN, Jan: *Delivering courses modelled using IMS Learning Design*. 2004. Preprint für Journal of Computer Assisted Learning, Herausgeber Open Universiteit Nederland, Quelle http://dspace.learningnetworks.org/handle/1820/35, Abruf 1.3.2005

[TX00] TAMER, Ibrahim I. ; XU, Cheng-Zhong: Neural net based pre-fetching to tolerate WWW latency. In: *Proceedings of the 20th International Conference on Distributed Computing Systems (ICDCS2000)*. Taipei, Taiwan, 2000

[UAD+02] ULBRICH, Armin ; AUSSERHOFER, Andreas ; DIETINGER, Thomas ; RABACK, Wolfgang ; HOITSCH, Patrick: Presentational and Interactive Elements for eLearning Courses Considering Behavioral and Constructivist Learning Models. In: *ED-Media 2002, Association for the Advancement of Computing in Education (AACE)* (2002)

[UCS03] UNNITHAN, Chandana R. ; CHAN, Elsie S. K. ; SWATMAN, Paula M. C.: Applying external solutions to organisational development: eLearning as a platform for internal growth. In: *Proceedings of the International Federation for Information Processing (IFIP 13E)*. Kopenhagen, 2003

[Vit96] VITTER, Jeffrey S.: Optimal prefetching via data compression. In: *Journal of the ACM* 43 (1996), S. 771 793

[VL96] VANDERWIEL, Steve ; LILJA, David J.: A Survey of Data Prefetching Techniques / University of Minnesota. 1996. Forschungsbericht. Nr. HPPC-96-05 (auf CD unter VL96.pdf)

[VYK+02] VENKATARAMANI, A. ; YALAGANDULA, P. ; KOKKU, R. ; SHARIF, S. ; DAHLIN, M.: The potential costs and benefits of long term prefetching for content distribution. In: *Elsevier Computer Communications* 25(4) (2002), S. 267 375

[Wan99] WAN, Jia: A survey of web caching schemes for the Internet. In: *ACM Computer Communication Review*, 1999, S. 29(5):36–46

[Weg93] WEGENER, Ingo: *Theoretische Informatik*. Stuttgart : B.G. Teubner, 1993

[Wei99] WEIDAUER, C.: Ein Vorgehensmodell für die industrielle Entwicklung multimedialer Lehr- und Lernsysteme Spezifikationsansätze. In: *M. Nagl, A. Behle, B. Westfechtel, H. Balzert, C. Weidauer, H.-W. Six, P. Pauen, J. Voss, W. Schäfer, J. Wadsack, U. Kelter (Hrsg.): Softwaretechnische Anforderungen an multimediale Lehr- und Lernsysteme, Report* (1999), S. S.113–145. – (auf CD unter NBW+99.pdf)

[Wil04] WILKENS, Ulrike: E-Learning: Strategie und Umsetzung an der Hochschule Bremen. Entfaltung pädagogischer Gestaltungsräume durch die Vereinnahmung informationstechnischer Infrastrukturen. In: *Claudia Bremer und Kerstin E. Kohl (Hrsg.): E-Learning-Strategien und E-Learning-Kompetenzen an Hochschulen* (2004), S. 111–122

[WMGC77] WITKIN, H.A. ; MOORE, C.A ; GOODENOUGH, D.R. ; COX, P.W.: Field dependent and field independent cognitive styles and their educational implications. In: *Review of Educational Research* 47 (1977), S. 1–64

[YHN03] YANG, Qiang ; HUANG, Joshua Z. ; NG, Michael: A data cube model for prediction-based web prefetching. In: *Journal of Intelligent Information Systems archive* 20 (2003), Nr. 1, S. 11–30

[YK00] YU, S-Z. ; KOBAYASHI, H.: A New Prefetch Cache Scheme. In: *IEEE Globecom 2000*. San Francisco, 2000

[ZA01] ZUKERMAN, Ingrid ; ALBRECHT, David W.: Predictive Statistical Models for User Modeling. In: *User Modeling and User-Adapted Interaction* 11 (2001), Nr. 1-2, S. 5–18

[Zai01] ZAIANE, Osmar R.: Web Usage Mining for a Better Web-Based Learning Environment. In: *Proceedings of Conference on Advanced Technology for Education (CATE2001)*, 2001

[ZAN99] ZUKERMAN, I. ; ALBRECHT, D. ; NICHOLSON, A.: Predicting users' requests on the WWW. In: *Proceedings of the seventh international conference on User modeling (UM99)*. Banff, Kanada, 1999, S. 275–284

[Zha01] ZHANG, Hainning: *Improving Performance on WWW using path-based predictive caching and prefetching*. South China, Simon Fraser University, School of Computing Science, Diss., 2001

[ZL02] ZAIANE, Osmar R. ; LUO, Jun: Towards Evaluating Learners' Behaviour in a Web-Based Distance Learning Environment. In: *Proc. of IEEE International Conference on Advanced Learning Technologies*, 2002, S. 357–360

[Zon99] ZONA: The economic impact of unaccessible Web site download speeds / Zona Research Center. 1999. – White Papers Zona Research Center 1999