Online-Planungsprobleme mit Fristen

—

Algorithmendesign und Analysen

Dissertation

von

Marco Riedel

Schriftliche Arbeit zur Erlangung des Grades
eines Doktors der Naturwissenschaften

Fachbereich 17 (Mathematik-Informatik)
Universität-GH Paderborn

November 2000

gewidmet meinem Mathematiklehrer
Herrn Busch,
der mich in den Klassenstufen 8 bis 10 unterrichtete und mich dabei nicht nur fachlich auf meinen weiteren Weg vorbereitete

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

Steigende Bandbreiten von Kommunikationsnetzwerken haben schon vor über zehn Jahren die Idee aufkommen lassen, Szenarien für multimediale Anwendungen zu entwickeln, in denen u. a. digitalisierte Spielfilme in zentralen Speichern verwaltet und bei Bedarf in Echtzeit zum Nutzer übertragen werden. Beispielsweise könnten damit auf längere Sicht die Videotheken in ihrer heutigen Ausprägung ersetzt werden. Diese Technologie ließe ebenfalls den Aufbau von Filmarchiven mit allgemeinen Zugriffsmöglichkeiten zu.

In diesem Zusammenhang wurde die Frage aufgeworfen, wie derartige Speicher- und Auslieferungssysteme — im folgenden Server genannt — aufgebaut und gesteuert werden sollen. Ein MPEG-komprimierter Video-Datenstrom, wie er z. B. im Bereich des digitalen Fernsehens benutzt wird, hat bei den derzeitig eingesetzten Wiedergabequalitäten eine Bandbreite von ca. 6 MBit/s. Allein das Abspeichern der Videodaten einer mittleren Auswahl von 1000 Titeln in Spielfilmlänge erfordert eine Speicherkapazität von mehreren Terabyte, und darauf müssen Zugriffe mit der geforderten Geschwindigkeit und Bandbreite erzielt werden. Mit dieser Anforderung und dem Wunsch, daß solch ein Server gleichzeitig viele Nutzer bedienen soll, ergibt sich bei den derzeit verfügbaren Speichertechnologien ein System von verteilten Speicherressourcen als grundlegendes Architekturmerkmal. Um mit konkurrierenden Zugriffswünschen der Anwender umgehen zu können, ist zudem der Einsatz von Pufferspeichern sinnvoll. Weiterhin ist die mehrfache Speicherung der Nutzdaten in verschiedenen Speicherressourcen in Erwägung zu ziehen. Durch solche Konstruktionen ergeben sich Wahlmöglichkeiten über den Ort und den exakten Zeitpunkt des Auslesens der angeforderten Daten. Gleichzeitig muß ein internes Planungsproblem gelöst werden, um diese Auswahl zu treffen.

Von Seiten der Forschung in akademischen Einrichtungen und entwicklungsnahen Abteilungen der Industrie wurden Mitte der 90'er Jahre verschiedene Versuchsprojekte durchgeführt. Zu den bekannteren Vorhaben gehören: das „Berkeley Distributed VOD-System“ [BMRC], der „TW/SGI Orlando Trail“, die

Multimediadatenbank der Firma „Oracle“ mit dem massiv parallelen nCUBE-Supercomputer als technischer Basis [Buck94] sowie ein Pilotprojekt des Unternehmens „Siemens“ in Deutschland. Innerhalb dieser Projekte wurden meist prestigeträchtige und hochgesteckte Ziele verfolgt. Teilweise sind für diese Vorhaben nicht einmal die geplanten Anwenderzahlen aufgebracht worden, was in mindestens zwei Fällen zu einem Abbruch des Projektes führte. Bei der Bewertung ist allerdings zu berücksichtigen, daß die Ursache für Fehlschläge nicht notwendigerweise in der technischen Umsetzung zu suchen sind.
In einem anderen Teil der Projekte wurden funktionsfähige Systeme entwickelt und ihr praktischer Einsatz demonstriert. Bisher ist jedoch eine Massenverbreitung dieser Technologien nicht erkennbar.

Neben den konkreten Projekten wurden zur gleichen Zeit auch intensive theoretische Studien zu diesem Thema betrieben. Dabei sind abstrakte Modelle entwickelt und analysiert worden (z. B. [DDM+95, CZ96] sowie [AGH95, BKPS96][1]). Auch in jüngster Zeit bleiben die eingangs erwähnten Fragestellungen für die Forschung interessant, wobei sich mit der weiten Verbreitung des Internets eine Themenverschiebung in Richtung allgemein anwendbarer Datenserver und zu Netzen von verteilten Speichern (Storage Area Networks) vollzieht. Als ein Beispiel dieser Aktivitäten sei das PRESTO-Projekt [BBS99] und die in seinem Umfeld entstandenen theoretischen Modelle mit ihren Analysen genannt.

Vor diesem Hintergrund wird innerhalb der vorliegenden Abhandlung ein Teilaspekt von verteilten Serversystemen für Video-Datenströme, die in Echtzeit auszuliefern sind, untersucht. Das hierfür entwickelte Modell mit dem Namen *Data-Access-Problem* (DAP) berücksichtigt nur die Anfragen nach Datenpaketen mit harten zeitlichen Beschränkungen (Fristen) und die Speicherressourcen. Unter der Annahme, daß die Datenpakete eine Einheitsgröße besitzen, entsteht ein Zuordnungsproblem der Anfragen zu Zeitintervallen der Ressourcen, welches durch einen bipartiten Graphen abgebildet wird. Die Lösung des Planungsproblems ist dann identisch zur Bestimmung eines Matchings in diesem Graphen.
Da diesem Problem ein Zeitmodell zugrunde liegt, wurde das mathematische Modell des DAP so definiert, daß die zur Lösung eingesetzten Algorithmen gezwungen sind, endgültige Entscheidungen über die Nutzung der Ressourcen zu treffen, ohne die zukünftigen Anforderungen zu kennen. Solche Formulierungen, bei denen Algorithmen Teile der finalen Lösung ausgeben, ohne die vollständige Eingabe, d. h. die Beschreibung der kompletten Probleminstanz, zu kennen, werden Online-Probleme genannt.
Ein verteilter Datenserver arbeitet bei Auslieferung von Video-Datenströmen ebenfalls in einer Umgebung, in der die zukünftigen Anforderungen unbekannt sind und die aktuell benötigten Daten mit Hilfe der zur Verfügung stehenden Ressourcen ausgeliefert werden müssen.

[1] In diesen Aufsätzen wurde dieselbe Analysemethode eingesetzt, die auch bei den Studien innerhalb der vorliegenden Arbeit benutzt wird.

In Summe ergibt sich für das untersuchte Modell die folgende Aufgabe: Um möglichst viele der angeforderten Datenpakete auszuliefern muß ein Matching mit großer Kardinalität in einem bipartiten Graphen bestimmt werden, wobei eine Formulierung als Online-Problem zu berücksichtigen ist.

Für Online-Probleme und -Algorithmen sind verschiedene Untersuchungsmethoden bekannt. In dieser Arbeit wird die Technik der *Competitive Analysis* eingesetzt [ST85]. Es erfolgt dabei eine Gegenüberstellung der Leistungsfähigkeit eines Online-Algorithmus zu der Leistung eines optimalen Algorithmus. Für jede mögliche Eingabe wird die Qualität einer optimalen Lösung mit der erzielten Lösungsqualität des Online-Algorithmus verglichen. Das ungünstigste Verhältnis dieser Werte wird bestimmt und trägt die Bezeichnung *Competitive Ratio*. Es handelt sich also um den Typ einer Worst Case Analyse, in welcher der schlimmste Fall der Eingabe herausgearbeitet und bewertet wird. Der Competitive Ratio gibt damit eine Garantie für die Qualität der Lösung, die vom untersuchten Online-Algorithmus berechnet wird.
Die meisten anderen Methoden zum Studium von Online-Problemen gehen von der Existenz einer Eingabeverteilung aus. Dies geschieht implizit auch bei der Konstruktion von randomisierten Lösungsverfahren. Mit einer festgelegten Eingabeverteilung können dann Erwartungswerte sowie erwartete Abweichungen für die Lösungsqualität der Algorithmen bestimmt werden. Die Güte der Untersuchungsergebnisse hängt allerdings in starkem Maße von der richtigen Wahl dieser Eingabeverteilung ab. Die Aussage solcher Resultate ist jedoch prinzipiell eine andere als bei der Competitive Analysis. Einerseits können sie die Beobachtungen beim praktischen Einsatz der Online-Algorithmen stützen oder gut vorhersagen. Andererseits bleibt die Stärke des möglichen Leistungseinbruches bei ungünstigen Eingaben unklar. Die Möglichkeit solche Schwächen aufzudecken zeichnet die Competitive Analysis aus.

Das oben motivierte Modell des DAP besitzt eine komplexe Struktur von Abhängigkeiten. Deshalb wurden die Studien an einem Modell mit vereinfachten Strukturen begonnen. Es trägt den Namen *Online-Request-Server-Matching-Problem* (ORSM-Problem). Gleichzeitig ist dieses Modell eine Generalisierung des DAP und einer Anzahl weiterer untersuchter Modellvarianten. Diese Modellvarianten bilden Konzepte ab, die im DAP zu finden sind, und ermöglichen so das Studium einzelner Strukturelemente dieses Online-Problems.
Daneben wird eine Erweiterung des ORSM-Problems mit Gewichten (wORSM-Problem) untersucht. Diese Gewichte können z. B. Prioritäten der Aufträge abbilden.
Mit den Untersuchungen, der Entwicklung und Analyse entsprechender Online-Algorithmen und den dabei gewonnenen Erkenntnissen ist es möglich, für eine Klasse von Online-Planungsproblemen einige generelle Regeln für das Algorithmendesign aufzustellen.

1.2 Aufbau der Arbeit

An dieser Stelle soll ein grober Überblick über die vorliegende Arbeit und die Aufteilung der Kapitel erfolgen. Jedem einzelnen Kapitel ist eine Inhaltsübersicht vorangestellt, welche zum Detailaufbau und den dargelegten Resultaten Auskunft gibt.

Da in dieser Arbeit ausschließlich die Competitive Analysis als Untersuchungsmethode angewandt wird, gibt das Kapitel 2 dazu eine Einführung mit Beispielen. Sie wird durch die Darstellung und Bewertung von neueren, modifizierten Varianten der Competitive Analysis ergänzt.
Einen Einblick in die Modelle und Resultate von Online-Lastbalancierungs- und Online-Planungsproblemen wird im Kapitel 3: „Vorausgegangene Forschungsergebnisse“ gegeben. Daneben werden die relevanten Vorarbeiten zu Online-Planungsproblemen mit Realzeitanforderungen und zu Online-Matching-Problemen aus der Literatur erläutert.
Im Kapitel 4 werden die verwendeten Notationen eingeführt und die formalen Modelldefinitionen einschließlich aller betrachteten Varianten angegeben. Dieses Kapitel schließt mit Vergleichen und Abgrenzungen zu bekannten Modellen aus der Literatur.
In den darauf folgenden fünf Kapiteln werden die Modelle nacheinander untersucht. Es beginnt mit dem einfachsten Modell des ORSM-Problems in Kapitel 5. Dabei werden grundlegende Eigenschaften und Beweistechniken eingeführt. Kapitel 6 beschäftigt sich mit der gewichteten Modellvariante (wORSM), der einzigen betrachteten Verallgemeinerung. Die Analyse des wORSM-Problems kann zwar die untere und obere Schranke des Competitive Ratios nicht zusammenführen, aber die Bestimmung der Leistungsgarantie eines konkreten Online-Algorithmus ist selbst von Interesse. Dieser Beweis erfordert die Charakterisierung von gewichteten erweiternden Wegen in Graphen, die dynamischen Veränderungen unterworfen sind. Spezialisierte Modelle, die bei Einführung von zusätzlichen Modellkonzepten wie Vorschau, Blockeingabe, Restriktionen in der zugrundeliegenden Graphenstruktur (z. B. konstanter Knotengrad in einer Teilmenge der Knoten) und Kombinationen daraus entstehen, werden in den Kapiteln 7 und 8 behandelt. Diese Modellvarianten stellen Zwischenstufen zum Data-Access-Problem dar. Im Kapitel 7 werden die Problemvarianten studiert, für die exakte Analysen vorliegen. Bei den Beweisen wird in starkem Maße auf die für das ORSM-Modell entwickelten Techniken zurückgegriffen. Dies gilt ebenso für die unteren Schranken der in Kapitel 8 betrachteten Modellvariante. Für einige Wertekombinationen der Parameter dieses Modells sind exakte obere Schranken für den Competitive Ratio durch Computerbeweise bestimmt worden. Der prinzipielle Aufbau dieses Programmes, die Details seiner effizienten Implementierung und der Zusammenhang mit einer Standardbeweistechnik der Competitive Analysis sind ebenfalls im Kapitel 8 zu finden.
Unter den in der Arbeit behandelten Modellen weist das Data-Access-Problem die größte Nähe zu Fragestellungen aus der Praxis auf. Die verfügbaren Untersu-

chungsergebnisse schließen verschiedene untere Schranken und Betrachtungen zur Leistungsfähigkeit einiger Online-Algorithmen ein und sind in Kapitel 9 nachzulesen.
Darauf folgt ein kurzes Kapitel mit Hinweisen, wie die Konstruktionen sämtlicher aufgeführter unterer Schranken zu transformieren sind, um erste untere Schranken für den Competitive Ratio randomisierter Online-Algorithmen zu erhalten.
Es schließen sich einige Bemerkungen zu den offengebliebenen Problemen und zu interessanten weiterführenden Fragestellungen an. Das Resümee beendet diese Arbeit mit einem Rückblick und der Wiederholung von essentiellen Erkenntnissen zum Design von Online-Algorithmen für die behandelte Problemklasse.

Innerhalb der gesamten Arbeit werden die Sätze, Lemmata und Korollare gemeinsam durchgängig numeriert, da der Beweisumfang einiger Lemmata und die Aussagekraft der meisten Korollare denen von Sätzen nicht nachsteht. Diese Art der Numerierung erleichtert m. E. das Auffinden von Bezügen.
Diese Aussage betrifft nicht die Definitionen, die selbstständig durchgezählt werden.

1.3 Publikation von Resultaten

Ein Teil der Ergebnisse, die innerhalb dieser Arbeit beschrieben werden, wurden bereits zuvor unter Mitwirkung von Koautoren oder vom Autor allein auf Konferenzen vorgestellt und in Zeitschriften oder anderen Medien veröffentlicht. Es folgt eine Übersicht dieser Quellen zusammen mit Querbezügen zu den entsprechenden Stellen der vorliegenden Ausarbeitung:

- Die ersten Ergebnisse zum Online-Request-Server-Matching-Problem (ORSM-Problem; Kapitel 5, Sätze 3 und 6) und seiner gewichteten Variante (wORSM-Problem; Kapitel 6, Sätze 7, 8 und 17) wurden für das „Symposium on Theoretical Aspects in Computer Science 1999“ akzeptiert. Wegen restriktiver Druckseitenbegrenzung mußte der umfangreiche Beweis zu Satz 17 entfallen. Die genaue Quellenangabe lautet:

  Riedel, Marco: Online Matching for Scheduling Problems. In: Meinel, Chrisoph (Hrsg.) ; Tison, Sophie (Hrsg.): *Proceedings of the 16th Annual Symposium on Theoretical Aspects in Computer Science —STACS '99, (Trier, Germany, March 4–6, 1999)*. Berlin-Heidelberg-New York : Springer-Verlag, 1999 (Lecture Notes in Computer Science 1563), S. 571–580

- Eine vollständige Ausarbeitung der obigen Resultate ist als Technischer Bericht mit der Nummer tr-ri-98-195 im Fachbereich Mathematik-Informatik der Universität Paderborn erschienen:

  Riedel, Marco: *Online Request Server Matching*, April 1998

- Zu einem späteren Zeitpunkt wurde das einfache ORSM-Modell (Kapitel 5) zusammen mit den Modellmodifikationen aus Kapitel 7 (Satz 20 bis Korollar 27) und eine Darstellung der bis dahin bekannten Resultate zur Modellvariante aus Kapitel 8 für eine Zeitschriftenveröffentlichung eingereicht:

RIEDEL, Marco: Online Request Server Matching. Akzeptiert zur Veröffentlichung im *Journal of Theoretical Computer Science A (Algorithms, automata, complexity and games)*, vorläufig geplant für Band 268 (2001), Nr. 1

- Die in Kapitel 9 angegebenen Forschungsergebnisse zum Data-Access-Problem (DAP) stammen zum Teil (Satz 41 sowie die Angaben im Kapitel 9.2) aus:

  BERENBRINK, Petra ; RIEDEL, Marco ; SCHEIDELER, Christian: Simple Competitive Request Scheduling Strategies. In: *Proceedings of the Eleventh ACM Symposium on Parallel Algorithms and Architectures, (Saint-Malo, France, June 27-30, 1999)*. New York : ACM Press, 1999, S. 33–42

  An diesem Ort sind untere und obere Schranken für den Competitive Ratio von fünf verschiedenen Online-Strategien für das DAP mit $c = 2$ nachgewiesen. Zusätzlich werden lokale Online-Strategien und -Kommunikationsprotokolle aufgestellt und analysiert, in denen das Wissen verteilt ist und die Lieferentscheidungen von Recheneinheiten getroffen werden, welche die einzelnen Speicherressourcen steuern.

2 Einführung in die Competitive Analysis

Übersicht: In diesem Kapitel werden wichtige Grundbegriffe und Notationen definiert. Die Methode der *Competitive Analysis* wird dabei nicht nur einführend erläutert, sondern auch anhand einfacher und klassischer Beispiele vorgeführt. Es werden prinzipielle Beweistechniken aufgezeigt und die Methode wird in verschiedenen Formen interpretiert. Zudem werden Vor- und Nachteile, auch im Vergleich zu anderen Analyseverfahren für Online-Algorithmen, herausgearbeitet.
Ein zweiter Schwerpunkt dieses Kapitels besteht in der Darstellung von Erweiterungen der Competitive Analysis und abgeleiteten Analysetechniken. Diese neueren Ideen versuchen einzelne, schwerwiegende Nachteile der originären Definition zu umgehen. Am Ende des Kapitels soll dann eine kritische Auseinandersetzung mit diesen modifizierten Methoden sowie der Versuch einer Bewertung und eine Abschätzung ihres Potentials erfolgen.

Wie schon mehrfach angedeutet, studiert diese Arbeit verschiedene Online-Probleme und bedient sich dabei der Methode der Competitive Analysis. Das Ziel der Forscher, die Untersuchungen mit Hilfe dieser Technik durchführen, liegt im Aufbau einer Theorie. Dabei sollen abstrakte Problemklassen sowie abstrakte Algorithmenklassen für deren Lösung identifiziert werden. Neben diesen Anstrengungen versucht man generelle Techniken für Design und Analyse derartiger Algorithmen zu entwickeln. Auch der Einfluß auf und die Beziehungen zu anderen Feldern der theoretischen Informatik werden dabei nicht außer Acht gelassen.
Es folgen zunächst formale Definitionen und es werden notwendige Notationen eingeführt. Dabei wird eine möglichst enge Anlehnung an den Notationen des einzig verfügbaren Lehrbuches zum Thema [BEY98] versucht. Es sollte deshalb nicht verwundern, daß man den Inhalt der Kapitel 2.1 und 2.2 an der genannten Stelle nachlesen kann, sofern nicht anders angegeben. Dies gilt auch für die Beispielprobleme und deren Lösungen, die dort z. T. ausführlicher behandelt werden. Die Quelle [BEY98] ist eine kompaktere und strukturiertere Darstellung, als es unzählige Referenzen auf Originalartikel sein können.

2.1 Grundlegende Definitionen und Notationen

In der klassischen Algorithmentheorie werden vorwiegend Offline-Algorithmen betrachtet. Einem solchen Algorithmus steht die Eingabe für ein zu lösendes Problem vollständig zum Lesen zur Verfügung. Durch Operieren auf diesen Eingabedaten wird eine Problemlösung berechnet, die zum Schluß als Resultat ausgegeben wird. Im Unterschied dazu gibt es aber auch Problemklassen, bei denen die Eingabe aus einer *Sequenz* von Eingabeteilen besteht, und die einem Algorithmus nur stückweise enthüllt wird. Trotzdem wird von solch einem Algorithmus verlangt, daß er für Teileingaben schon Teile der endgültigen Lösung bestimmt. Es wird dann von Online-Problemen gesprochen, die im folgenden näher definiert werden.

Jedem Online-Problem liegt ein Zeitmodell zugrunde. Dies kann ein lineares, kontinuierliches Modell einer Realzeit sein. Häufig wird jedoch in den abstrakten Problemklassen ein Zeitmodell eingesetzt, wie es aus ereignisgesteuerten Simulationen bekannt ist: Ereignisse treten in einer Ordnung auf, die die zeitliche Abfolge und das Fortschreiten der Zeit definieren, ohne Rückschlüsse auf abgelaufene Zeitquantitäten zuzulassen. Zu diesen Ereignissen zählen insbesondere das Enthüllen von Elementen der Eingabesequenz, aber auch Änderungen im Systemzustand. Ein Beispiel für Letzteres wäre die Freigabe von Ressourcen nach Bearbeitung eines Auftrages. Das konkrete Online-Problem definiert nun, zu welchen Ereigniszeitpunkten Entscheidungen über die Lösung getroffen werden müssen. Mit diesen Entscheidungen sind Kosten oder Gewinne verbunden und sie können nicht revidiert werden. Die besondere Schwierigkeit von Online-Problemen besteht in der Tatsache, daß zum Entscheidungszeitpunkt zukünftige Teile der Eingabe unbekannt sind, aber die getroffene Entscheidung die Systemkonfiguration und damit auch die zukünftig erzielbare Lösungsqualität beeinflußt. Die Bezeich-

nung Online-Algorithmus wird verwandt, wenn der Algorithmus ein Online-Problem definitionsgemäß löst, also Entscheidungen über Lösungsteile trifft, ohne die zukünftigen Teile der Eingabe zu kennen. Ein Maß zur Bewertung der Qualität solcher Online-Algorithmen stellt der *Competitive Ratio* dar. Vor einer Formalisierung dieses zentralen Begriffs der Competitive Analysis sei noch auf eine wichtige Unterscheidung hingewiesen. Je nach konkretem Problem gibt es eine zu minimierende Kostenfunktion *Cost* oder eine zu maximierende Gewinnfunktion *Perf* (Performance, Leistung). Die Eigenschaft *competitive* und der daraus abgeleitete Competitive Ratio unterliegen, je nach Typ der Problemstellung, verschiedenen Definitionen.

Definition 1: (c-competitive, Competitive Ratio)

Sei für ein Online-Problem:

ALG	–	*ein Online-Algorithmus*
OPT	–	*ein optimaler Offline-Algorithmus*
σ	–	*eine endliche Eingabesequenz (eine Menge von Eingaben über die Laufzeit)*
$Cost_X(\sigma)$	–	*die Kosten des Algorithmus X bei Eingabe σ*
$Perf_X(\sigma)$	–	*die Performance (Gewinn) des Algorithmus X bei Eingabe σ*
a	–	*eine Konstante aus* $\mathbb{R}$

Ein Online-Algorithmus ALG für ein Kostenminimierungsproblem ist c-competitive, falls gilt

$$\exists\, a : \forall\, \sigma : \quad Cost_{ALG}(\sigma) \leqq c \cdot Cost_{OPT}(\sigma) + a \ .$$

Ein Online-Algorithmus ALG für ein Gewinnmaximierungsproblem ist c-competitive, falls gilt

$$\exists\, a : \forall\, \sigma : \quad Perf_{OPT}(\sigma) \leqq c \cdot Perf_{ALG}(\sigma) + a \ .$$

Der Competitive Ratio $\mathcal{R}$ ist das Infimum der Menge aller c, für die der Online-Algorithmus ALG c-competitive ist:

$$\mathcal{R}(ALG) := \inf\{c \mid c \in \mathbb{R}, ALG \text{ ist } c\text{-competitive}\} \ .$$

Der Competitive Ratio stellt folglich den Faktor dar, um den die Kosten einer Lösung des Online-Algorithmus ALG im schlimmsten Fall höher sind als die einer optimalen Offline-Lösung, bzw. um den die Leistung geringer ist. Die Definition stellt sicher, daß dieser Wert stets größer oder gleich 1 ist.
Die additive Konstante a wird insignifikant, wenn die Kosten bzw. der Gewinn der Eingabesequenz unendlich werden können. Entfällt die Konstante a ($a = 0$), so wird auch von *streng competitive* gesprochen. In einem solchen Fall kann der Competitive Ratio äquivalent definiert werden:

$$\mathcal{R}(ALG) = \sup_{\sigma} \frac{Cost_{ALG}(\sigma)}{Cost_{OPT}(\sigma)} \quad \text{bzw.} \quad \mathcal{R}(ALG) = \sup_{\sigma} \frac{Perf_{OPT}(\sigma)}{Perf_{ALG}(\sigma)} \ .$$

Häufig wird eine untere Schranke im Competitive Ratio für ein Online-Problem bestimmt. Es kann dann für dieses Problem kein Online-Algorithmus konstruiert werden, dessen Competitive Ratio diese Schranke unterschreitet.

2.2 Prinzipielle Beweistechniken

Die Methode der Competitive Analysis wird häufig als ein Spiel interpretiert. Ein Online-Algorithmus, welcher seine Entscheidungen nur auf der Grundlage der bisherigen Eingabe trifft, muß sich gegen einen omnipotenten *Gegenspieler* behaupten. Dieser Gegenspieler erzeugt die Eingabesequenz. Dabei kennt er den Online-Algorithmus vollständig und besitzt beliebige Berechnungskraft. So ist er z. B. in der Lage eine schlimmste Eingabesequenz zur Maximierung des Competitive Ratios zu generieren, indem er den Online-Algorithmus auf allen möglichen Eingaben simuliert.
Diese Interpretation ist nicht nur sehr hilfreich, um viele weitere Definitionen im Kapitel 2.5 zu verstehen, sondern auch um untere Schranken für den Competitive Ratio von konkreten Algorithmen oder vollständigen Online-Problemen zu bestimmen. Für letztgenannte Aufgabe werden in der Literatur fast ausnahmslos Gegenspielerstrategien angegeben. Diese beginnen typischerweise mit einem kleinen Stück der Eingabesequenz, die einen Online-Algorithmus in eine Entscheidungssituation bringt. Für jede mögliche Entscheidung dieses Algorithmus wird dann angegeben, wie die Eingabesequenz fortzuschreiben ist, damit sowohl der gewünschte Competitive Ratio erreicht, als auch eine entsprechende neue Entscheidungssituation für den Online-Algorithmus geschaffen wird. Ein einfaches Beispiel soll diese Technik verdeutlichen.

Das k-Paging-Problem

Dieses Problem ist auch als Caching-Problem bekannt und tritt z. B. innerhalb eines Datenverarbeitungssystems mit einer Speicherhierarchie auf. Dabei existiert ein kleiner, schneller Speicher (Cache), der k einheitlich große Speicherseiten aufnehmen kann und ein langsamer, großer Hintergrundspeicher. Die Eingabe besteht aus einer Sequenz von Speicherseiten, auf die Zugriffe erfolgen. Im einfachsten Kostenmodell, dem Seitenfehlermodell, erzeugt ein Zugriff auf den Cache keine Kosten. Ist die benötigte Seite jedoch nur im Hintergrundspeicher vorhanden, so muß sie zuerst in den Cache geladen werden. Dieser Fall wird Seitenfehler genannt und dabei entsteht eine Kosteneinheit. Ein Online-Algorithmus hat in diesem Fall zu entscheiden, welche Speicherseite aus dem Cache zu verdrängen ist.

Schon in [ST85] wurde folgende Strategie angegeben: Der Gegenspieler benutzt nur Anfragen an $k+1$ Speicherseiten. Er fordert im nächsten Schritt immer die Seite an, die ein Online-Algorithmus ALG soeben verdrängt hat. Dabei entstehen in jedem Schritt Kosten von einer Einheit. Dieser Typ eines Gegenspielers, der in jedem Schritt die vorige Entscheidung des Online-Algorithmus bestraft, wird auch „grausam“ genannt.

Die optimale Offline-Lösung wurde von Belady in [Bel66] mit dem Algorithmus MIN angegeben: Es wird immer die Seite verdrängt, die in der Zukunft am längsten nicht benötigt wird. Da vom Gegenspieler nur $k+1$ Seiten benutzt werden, erzeugt MIN höchstens alle k Schritte einen Seitenfehler und somit eine Kosteneinheit. Aus diesen Fakten ergibt sich nun, daß kein Online-Algorithmus einen Competitive Ratio kleiner als k besitzen kann.

Leider sind nur in den wenigsten Fällen die optimalen Offline-Algorithmen bekannt bzw. ihre Leistung auf den Eingaben so einfach zu analysieren. Deshalb schränkt man sich häufig bei der Konstruktion der Gegenspielerstrategie ein und benutzt nur Eingabesequenzen mit wenigen und einfachen Strukturen. Von diesen ist dann der Wert der optimalen Lösung bestimmt worden.
An dieser Stelle soll noch eine weitere allgemeine Technik vorgestellt werden. Statt die optimale Lösung zu berechnen, wird mittels einer Durchschnittsbildung die Existenz einer optimalen Lösung mit einer Mindestqualität gezeigt. Auch dies soll an einem Beispiel veranschaulicht werden.

Das k-Server-Problem

Gegeben sind k *Server* in einem metrischen Raum M. Wie allgemein bekannt, besteht ein metrischer Raum M aus einer Punktmenge P, auf der eine positive, reflexive und symmetrische Abstandsfunktion $d : P^2 \to \mathbb{R}_+$ definiert ist und für die die Dreiecksungleichung gilt; d. h.:

$$
\begin{array}{lrcll}
\forall\, a,b \in P, a \neq b: & d(a,b) & > & 0 & \text{(positiv)} \\
\forall\, a \in P: & d(a,a) & = & 0 & \text{(reflexiv)} \\
\forall\, a,b \in P: & d(a,b) & = & d(b,a) & \text{(symmetrisch)} \\
\forall\, a,b,c \in P: & d(a,c) & \leqq & d(a,b)+d(b,c) & \text{(Dreiecksungleichung)}
\end{array}
$$

Die Eingabesequenz ist eine Folge von Punkten in P. Nach der Eingabe eines Punktes muß ein Algorithmus einen der Server zu diesem Punkt bewegen, falls sich nicht schon einer der Server auf diesem Punkt befindet. Die Summe der zurückgelegten Wege aller Server stellt in diesem Problem die Kosten dar.

Wie aus der Dreiecksungleichung folgt, genügt es für dieses Problem „faule“ Online-Algorithmen zu betrachten. Diese bewegen nur dann einen Server, und zwar genau einen, wenn der Eingabepunkt nicht schon durch einen Server abgedeckt ist.

Das oben eingeführte k-Paging-Problem ist ein Spezialfall des k-Server-Problems, bei dem der metrische Raum uniform ist, d. h. alle Abstände zwischen Punkten (Speicherseiten) sind 1.
Auch die Gegenspielerstrategie ist identisch: Es werden $k+1$ disjunkte Punkte aus einem beliebig vorgegebenen metrischen Raum M verwandt. Der grausame Gegenspieler fordert in der Eingabesequenz immer den Punkt an, der gerade von keinem Server abgedeckt ist. Dabei handelt es sich um den Punkt, von dem der

Online-Algorithmus ALG soeben den Server entfernt hat. Initial seien die k Server auf k verschiedenen Punkten aus der speziell gewählten $(k+1)$-elementigen Punktmenge plaziert.
Es ergeben sich für einen faulen Online-Algorithmus ALG auf einer Eingabesequenz $\sigma = (r_1, r_2, \ldots, r_n)$ der Länge $n \geqq 2$ die Kosten:

$$Cost_{\mathsf{ALG}}(\sigma) \;>\; \sum_{i=1}^{n-1} d(r_{i+1}, r_i) \;, \tag{1}$$

wobei die rechte Ungleichungsseite nur die Kosten der ersten $n-1$ Schritte zählt. Nun müssen jedoch die optimalen Kosten $Cost_{\mathsf{OPT}}(\sigma)$ von oben abgeschätzt werden. Dazu wird eine Menge $\mathcal{A} := \{A_1, A_2, \ldots, A_k\}$ von k faulen Algorithmen betrachtet. Diese k Algorithmen werden derart verwaltet, daß nach jedem Schritt mit einer Anfrage r die folgende Invariante gilt: Alle Algorithmen in $\mathcal{A}$ haben einen Server auf dem Punkt r, wie es die Problemdefinition verlangt, und die Konfigurationen all dieser Algorithmen sind paarweise verschieden. Für jeden Punkt $p \neq r$ gibt es deshalb genau einen Algorithmus in $\mathcal{A}$, der den Punkt p nicht mit einem Server abdeckt. (Es werden immer nur die gewählten $k+1$ Punkte aus M benutzt und betrachtet.) Initial gelte die Invariante, wobei der Punkt r von allen Algorithmen in $\mathcal{A}$ mit einem Server besetzt ist, auf dem die Startkonfiguration des Online-Algorithmus ALG keinen Server hat. Diese Annahmen über die initialen Zustände können ohne Beschränkung getroffen werden, da mit Hilfe der additiven Konstante a in der Definition des Competitive Ratios Abweichungen von diesen Argumentationen aufgefangen werden.
Bewegt im Schritt i der Online-Algorithmus ALG einen Server von Punkt a nach b, da der Punkt b die aktuelle Eingabe darstellt, so wird vom grausamen Gegenspieler im Schritt $i+1$ der Punkt a angefordert. Innerhalb der Algorithmenmenge $\mathcal{A}$ gibt es zu diesem Zeitpunkt $i+1$ nur einen Algorithmus A, der den Punkt a nicht mit einem Server abdeckt. Dieser muß also als einziger einen Server versetzen. Um die oben beschriebene Invariante zu erfüllen, zieht A den Server vom Punkt b, da nach Schritt i alle Algorithmen in $\mathcal{A}$ den Punkt b mit einem Server abdecken. In Summe entstehen so im Schritt $i+1$ für alle k Algorithmen der Menge $\mathcal{A}$ die Kosten $d(b,a)$. Diese Kosten sind identisch mit den entstandenen Kosten $d(a,b)$ des Online-Algorithmus ALG im Schritt i.
Die Gesamtkosten aller k Algorithmen in $\mathcal{A}$ auf die Eingabesequenz σ betragen:

$$\begin{aligned} Cost_{\mathcal{A}}(\sigma) &= \sum_{i=2}^{n} d(r_i, r_{i-1}) \\ &= \sum_{i=1}^{n-1} d(r_{i+1}, r_i) \;. \end{aligned}$$

Da diese Kosten in Summe für k Algorithmen anfallen, muß mindestens einer

dieser Algorithmen Kosten von höchstens dem Durchschnitt

$$\frac{1}{k} \sum_{i=1}^{n-1} d(r_{i+1}, r_i) \tag{2}$$

erzeugen. Damit ist die benötigte obere Schranke für eine optimale Lösung gezeigt. Aus den Kosten (1) und (2) ergibt sich k als untere Schranke für den Competitive Ratio des k-Server-Problems.

Von besonderem Interesse bei der Analyse von konkreten Online-Algorithmen ist deren garantierte Lösungsqualität, d. h. obere Schranken für den Competitive Ratio des Algorithmus. In einigen seltenen Fällen werden dazu ad-hoc-Argumente angegeben. Dies gelingt vorwiegend bei kombinatorischen Problemen. Ein solches Beispiel ist in Kapitel 5.3 auf Seite 83 zu finden. Im allgemeinen wird jedoch eine Potentialfunktionstechnik eingesetzt, deren nähere Erläuterung folgt.
Um zu zeigen, daß ein Online-Algorithmus ALG für ein Gewinnmaximierungsproblem c-competitive ist, muß nach Definition 1 die Gültigkeit der Ungleichung

$$\forall \sigma : \quad Perf_{\mathsf{OPT}}(\sigma) \ \leqq \ \mathsf{c} \cdot Perf_{\mathsf{ALG}}(\sigma) \ + \ \mathsf{a} \tag{3}$$

gezeigt werden. Dies wäre eine einfache Aufgabe, könnte für jeden Schritt i mit Eingabe σ_i einer beliebigen Eingabesequenz σ die Gültigkeit der Ungleichung $Perf_{\mathsf{OPT}}(\sigma_i) \leqq \mathsf{c} \cdot Perf_{\mathsf{ALG}}(\sigma_i)$ gezeigt werden. Im allgemeinen ist dies jedoch nicht möglich, da der Online-Algorithmus ALG zwar häufig einen höheren Gewinn als $Perf_{\mathsf{OPT}}(\sigma_i)/\mathsf{c}$ erreicht, dann aber in einigen Schritten erheblich schlechter abschneidet. Es wird folglich eine Analyse des „mittleren“ bzw. exakter, des *amortisierten* Gewinnes benötigt. Wie derartige Analysen funktionieren, wurde in dem lesenswerten Übersichtsartikel [Tar85] vorgestellt.
Offensichtlich muß der höhere Gewinn einiger Schritte „aufgespart“ werden, um in den Fällen des schlechten Abschneidens diesen aufgesparten Gewinn zum Ausgleichen verwenden zu können. Man kann sich ein Bankkonto für die Verwaltung dieser Gewinne vorstellen, auf das der Online-Algorithmus seine mit c multiplizierten Gewinne „einzahlt“. Der optimale Algorithmus erhält seine Gewinne von diesem Konto. Es muß dann in der Analyse gezeigt werden, daß der Kontostand niemals negativ wird bzw. nach unten durch eine Konstante begrenzt ist, da die Definition des Competitive Ratios eine additive Konstante a zuläßt. Die Betrachtungsweise mit dem Konto wird auch als „Bankiersicht“ bezeichnet.
In der Informatik hat sich allerdings eine andere Interpretation durchgesetzt. Es ist die Sicht eines Physikers, der eine Potentialfunktion Φ eines Feldes betrachtet. Diese nimmt im wesentlichen die Aufgabe des Kontos wahr. Jedoch wird mit dieser Interpretation klarer, daß der Systemzustand, d. h. die Konfiguration des Algorithmus, das zusätzlich aufgebaute Vermögen oder Potential darstellt. Deshalb ist die Potentialfunktion Φ eine Funktion des Systemzustandes in die reellen Zahlen. Bei der Analyse von Online-Algorithmen besteht der Systemzustand aus den Konfigurationen des Online-Algorithmus ALG und des optimalen Offline-Algorithmus OPT, d. h.

$$\Phi : \mathit{Konfiguration}(\mathsf{OPT}) \times \mathit{Konfiguration}(\mathsf{ALG}) \to \mathbb{R} \ .$$

Bei einer geschickten Definition der Funktion Φ kann dann für jeden Schritt i die Ungleichung

$$Perf_{\mathsf{OPT}}(\sigma_i) \quad \leqq \quad \mathsf{c} \cdot Perf_{\mathsf{ALG}}(\sigma_i) + \Delta\Phi \tag{4}$$

gezeigt werden. Dabei wird mit $\Delta\Phi$ die Veränderung der Potentialfunktion Φ zwischen dem Schritt $i-1$ und i notiert. Genauer gesagt, soll Φ_i den Wert der Potentialfunktion nach Bearbeitung der Eingabe σ_i bezeichnen, Φ_0 sei der Initialwert vor der ersten Eingabe und Φ_σ der Potentialfunktionswert nach Bearbeitung der kompletten Sequenz σ. Dann gilt $\Delta\Phi = \Phi_i - \Phi_{i-1}$ und die Addition der Ungleichung (4) für alle Schritte der Eingabesequenz σ ergibt:

$$\begin{aligned}
& \sum_i Perf_{\mathsf{OPT}}(\sigma_i) \quad \leqq \quad \sum_i (\mathsf{c} \cdot Perf_{\mathsf{ALG}}(\sigma_i) + \Delta\Phi) \\
\Leftrightarrow \quad & \sum_i Perf_{\mathsf{OPT}}(\sigma_i) \quad \leqq \quad \mathsf{c} \cdot \sum_i Perf_{\mathsf{ALG}}(\sigma_i) + \sum_i (\Phi_i - \Phi_{i-1}) \\
\Leftrightarrow \quad & Perf_{\mathsf{OPT}}(\sigma) \quad \leqq \quad \mathsf{c} \cdot Perf_{\mathsf{ALG}}(\sigma) + \Phi_\sigma - \Phi_0
\end{aligned} \tag{5}$$

Da hier eine sogenannte Teleskopsumme über die Werte von Φ vorliegt, bleiben zum Schluß nur der letzte (Φ_σ) und der erste Wert (Φ_0) in der Formel (5) übrig. Wird nun zusätzlich die Beschränktheit von $\Phi_\sigma - \Phi_0$ gezeigt, d. h.

$$\exists\, \mathsf{a} \in \mathbb{R} : \ \forall\, \sigma : \quad \Phi_\sigma - \Phi_0 \quad \leqq \quad \mathsf{a} \ , \tag{6}$$

so ist die Ungleichung (3) bewiesen und damit die Tatsache, daß ALG c-competitive ist.

Bei einem Kostenminimierungsproblem verändern sich die zu zeigenden Aussagen leicht. Für jede Eingabe σ_i wird gezeigt:

$$Cost_{\mathsf{ALG}}(\sigma_i) + \Delta\Phi \quad \leqq \quad \mathsf{c} \cdot Cost_{\mathsf{OPT}}(\sigma_i) \ ,$$

woraus sich schlußfolgert:

$$\begin{aligned}
& \sum_i Cost_{\mathsf{ALG}}(\sigma_i) \quad \leqq \quad \sum_i (\mathsf{c} \cdot Cost_{\mathsf{OPT}}(\sigma_i) - (\Phi_i - \Phi_{i-1})) \\
\Leftrightarrow \quad & Cost_{\mathsf{ALG}}(\sigma) \quad \leqq \quad \mathsf{c} \cdot Cost_{\mathsf{OPT}}(\sigma) - \Phi_\sigma + \Phi_0 \ .
\end{aligned}$$

Nach dem Zeigen der Beschränktheit

$$\exists\, \mathsf{a} \in \mathbb{R} : \ \forall\, \sigma : \quad \Phi_0 - \Phi_\sigma \quad \leqq \quad \mathsf{a}$$

folgt ebenfalls, daß ALG c-competitive ist.

Zur Illustration dieser Potentialfunktionstechnik soll das nächste Beispiel dienen.

Das List-Update-Problem

In seiner einfachsten Variante besteht das List-Update-Problem aus einer Menge von Datenblöcken, die mit einem Schlüssel versehen und in einer Liste gespeichert sind. Eine Liste ist ein abstrakter Datentyp, bei dem auf das erste Element direkt zugegriffen werden kann, und dort der Verweis auf das nächste Element zu finden ist, etc. Eine Liste stellt somit eine lineare Struktur dar, in der die Zugriffskosten für das i-te Element i Einheiten betragen. Die Eingabesequenz besteht nun aus einer Folge von Anfragen mit je einem Schlüssel. Nach jeder Anfrage muß auf das Element mit dem selben Schlüssel zugegriffen werden. Dabei fallen Kosten proportional zur Stellung des Elementes in der Liste an. Nach einem solchen Zugriff darf das Element kostenfrei zu einem beliebigen Platz in Richtung des Listenanfangs verschoben werden. Dadurch kann ein nachfolgender Zugriff auf das selbe Element erheblich kostengünstiger werden. Ein Zugriff auf eines der nach hinten verdrängten Elemente wird entsprechend um eine Einheit teurer. Zusätzlich darf ein Algorithmus weitere kostenpflichtige Schritte zur Reorganisation der Liste durchführen: Zwei in der Liste benachbarte Elemente können vertauscht werden. Dabei entsteht auch eine Kosteneinheit. Die Aufgabe eines Algorithmus besteht nun in der Verwaltung dieser Liste, d. h. in der Entscheidung, wohin das aktuelle Element nach seinem Zugriff zu verschieben ist und welche weiteren kostenpflichtigen Tauschoperationen durchzuführen sind. Er hat das Ziel die Summe der Tausch- und Zugriffskosten aller Anfragen der Eingabesequenz zu minimieren. Es sei angemerkt, daß Kombinationen von initialen Listen und Eingabesequenzen bekannt sind, bei denen zur Minimierung der Gesamtkosten bezahlter Elementtausch notwendig ist. Wie kürzlich in [Amb00] gezeigt wurde, ist das zugehörige Offline-Problem $\mathcal{NP}$-schwer.
Prinzipiell ist der abstrakte Datentyp einer Liste dynamisch und unterstützt neben dem Suchen und Zugreifen auch das Einfügen und Löschen von Elementen. Alle bisher durchgeführten Studien zum List-Update-Problem lassen sich unter Beibehaltung der Ergebnisse problemlos erweitern, so daß auch diese dynamischen Operationen unterstützt werden. Deshalb wird meist die vereinfachte Problemvariante mit einer festen Menge von Elementen und Zugriffen ausschließlich darauf untersucht. Soll die Variante des Problems betont werden, so wird von *statischem* (vereinfachtem) bzw. *dynamischem* List-Update-Problem (mit Einfüge- und Löschoperationen) gesprochen.

Der erste Online-Algorithmus für das List-Update-Problem mit konstantem Competitive Ratio wurde in [ST85] untersucht. Er heißt „Move to Front“ (MTF) und benutzt keine kostenpflichtigen Tauschoperationen. Die einzige Regel zur Reorganisation der Liste lautet: Nach dem Zugriff auf ein Element wird selbiges kostenfrei bis zum Listenanfang verschoben.
Nun wird gezeigt, daß MTF für das statische List-Update-Problem 2-competitive ist. Dazu wird eine Potentialfunktion Φ definiert. Sie zählt wieviele Paare (x_i, x_j) von Listenelementen eine Inversion bilden, d. h. x_i steht in der Liste des optimalen Algorithmus OPT vor dem Element x_j, aber x_j befindet sich vor x_i in der durch

MTF verwalteten Liste. Der Wert von Φ ist gleich der benötigten Anzahl von Nachbarelementvertauschungen, um die von OPT verwaltete Liste in die Liste des MTF Algorithmus zu überführen und umgekehrt. Mit dieser Definition kann Φ niemals einen negativen Wert annehmen und zu Beginn gilt $\Phi_0 = 0$, wenn identische Startlisten für beide Algorithmen vorausgesetzt werden. Φ_i soll den Potentialfunktionswert nach Bearbeitung des Eingabeelementes σ_i bezeichnen.

Als nächstes wird behauptet, daß für jede Eingabe σ_i einer beliebigen Eingabesequenz σ die Ungleichung

$$Cost_{\mathsf{MTF}}(\sigma_i) + \Delta\Phi \quad \leqq \quad 2 \cdot Cost_{\mathrm{S}}(\sigma_i) + Cost_{\mathrm{T}}(\sigma_i) \tag{7}$$

gilt, wobei $\Delta\Phi = \Phi_i - \Phi_{i-1}$ bezeichnet und sich die Kosten des optimalen Algorithmus OPT in die Zugriffskosten auf das Element σ_i ($Cost_{\mathrm{S}}(\sigma_i)$ – „Suchkosten“) und die im direkten Anschluß daran anfallenden Kosten für bezahlte Tauschoperationen ($Cost_{\mathrm{T}}(\sigma_i)$) aufteilen:

$$Cost_{\mathsf{OPT}}(\sigma_i) \quad = \quad Cost_{\mathrm{S}}(\sigma_i) + Cost_{\mathrm{T}}(\sigma_i) \ .$$

Um die Ungleichung (7) zu beweisen, werden die beiden von OPT und MTF verwalteten Listen vor der Eingabe von σ_i und nach den Manipulationen durch die beiden Algorithmen betrachtet. Das Element σ_i stehe an Position j in der Liste von OPT und an Position k in MTF's Liste. Damit sind die Zugriffskosten von OPT: $Cost_{\mathrm{S}}(\sigma_i) = j$ und von MTF: $Cost_{\mathsf{MTF}}(\sigma_i) = k$. In MTF's Liste befinden sich vor σ_i weitere $(k-1)$ Elemente. Diese Menge wird aufgeteilt in die τ Elemente, die in der Liste von OPT *nach* dem Element σ_i erscheinen und den restlichen $k - 1 - \tau$ Elementen, die vor σ_i in OPT's Liste stehen. Da σ_i in dieser Liste an Position j steht und mindestens $k - 1 - \tau$ Elemente davor existieren, folgt $k - \tau \leqq j$.
MTF verschiebt das Element σ_i nach seinem Zugriff an den Listenanfang und verändert dabei den Wert der Potentialfunktion Φ. Die τ oben definierten Inversionen zwischen σ_i und Vorgängerelementen in MTF's Liste werden aufgehoben und bezüglich der restlichen $k - 1 - \tau$ Vorgänger werden neue Inversionen geschaffen. Deshalb ist nach der Bearbeitung von σ_i durch MTF der Wert von Φ um $(k - 1 - \tau) - \tau$ angewachsen.
Die Operationen des optimalen Algorithmus OPT auf seiner Liste können auch Auswirkungen auf die Potentialfunktion haben. Bei jeder kostenfreien Nachbarschaftsvertauschung des Elementes σ_i in Richtung Listenanfang wird eine Inversion aufgehoben, da sich σ_i in der Liste von MTF am Anfang befindet. Folglich ist dadurch keine Erhöhung des Potentialfunktionswertes zu verzeichnen.[2] Falls OPT weitere kostenpflichtige Tauschoperationen durchführt, so wird jeweils eine Inversion geschaffen oder aufgehoben, es fällt allerdings immer eine Kosteneinheit an. Die Erhöhung von Φ ist also durch $Cost_{\mathrm{T}}(\sigma_i)$ begrenzt. Damit wird die

[2] In [ST85] wird unter Zuhilfenahme dieser Beobachtungen die schärfere Aussage $Cost_{\mathsf{MTF}}(\sigma) \leqq 2 \cdot Cost_{\mathrm{S}}(\sigma) + Cost_{\mathrm{T}}(\sigma) - Free(\sigma) - n$ gezeigt, wobei $Free(\sigma)$ die Anzahl der von OPT durchgeführten kostenfreien Elementvertauschungen und n die Länge von σ ist.

Ungleichung (7) bewiesen:

$$\begin{aligned} Cost_{\mathsf{MTF}}(\sigma_i) + \Delta\Phi &\leqq k + (k-1-\tau) - \tau + Cost_{\mathrm{T}}(\sigma_i) \\ &\leqq 2(k-\tau) + Cost_{\mathrm{T}}(\sigma_i) \\ &\leqq 2 \cdot Cost_{\mathrm{S}}(\sigma_i) + Cost_{\mathrm{T}}(\sigma_i) \ . \end{aligned}$$

Werden diese Ungleichungen für eine komplette Eingabesequenz σ aufsummiert, so ergibt sich

$$Cost_{\mathsf{MTF}}(\sigma) + \Phi_\sigma - \Phi_0 \leqq 2 \cdot Cost_{\mathrm{S}}(\sigma) + Cost_{\mathrm{T}}(\sigma) \leqq 2 \cdot Cost_{\mathsf{OPT}}(\sigma) \ ,$$

und aus der Beschränktheit von $\Phi_\sigma - \Phi_0$ folgt, daß MTF 2-competitive ist.
An dieser Stelle soll betont werden, daß im vorliegenden Beweis keinerlei Wissen über die Funktionsweise oder Eigenschaften des optimalen Offline-Algorithmus OPT eingegangen ist. Zur klareren Darstellung der Potentialfunktionsmethode wurde eine gegenüber [ST85] vereinfachte Aussage bewiesen und zusätzlich das List-Update-Problem auf seine statische Variante eingeschränkt.
Aus den exakten Aussagen, die in [ST85] bewiesen werden, folgt für die dynamische Problemvariante mit maximaler Listenlänge l, daß MTF $(2 - 1/l)$-competitive ist. Für die statische Problemvariante wurde in [Ira91] das scharfe Resultat $\mathcal{R}(\mathsf{MTF}) = 2 - 2/(l+1)$ bewiesen. Nach [BEY98] überträgt sich dieses Ergebnis auf die dynamische Problemvariante, und MTF ist sogar streng $(2 - 2/(l+1))$-competitive, wenn MTF und OPT auf identischen Listen starten.

Die oben vorgestellte Potentialfunktionstechnik hat sich als eine Art Standard für die Competitive Analysis durchgesetzt und ist der allgemein empfohlene Ansatz. In Kapitel 8.2 wird auch ein Beweis präsentiert, der zeigt, daß die dazu notwendigen Potentialfunktionen de facto immer existieren. Die Schwierigkeit besteht jedoch im Auffinden der Funktionen und dem Nachweis der benötigten Bedingungen (Gleichungen (4) und (6)). So sind in der Literatur auch verschiedene Beweise bekannt, die neben einer Potentialfunktion weitere Eigenschaften des Online-Algorithmus zum Nachweis der gesuchten oberen Schranken ausnutzen. Das wohl prominenteste und sehr komplexe Beispiel einer derartigen Analyse stammt von Elias Koutsoupias und Christos H. Papadimitriou und ist in [KP94b] zu finden. Dort wird die k-Server-Vermutung von Mark S. Manasse, Lyle A. McGeoch und Daniel D. Sleator [MMS88] bis auf einen Faktor von 2 durch die Analyse des Work-Function-Algorithmus (WFA, [CL92] basierend auf Vorarbeiten) bewiesen. Genauer wird gezeigt: WFA ist $(2k-1)$-competitive. Die bekannte untere Schranke ist k und es wird vermutet, daß sie scharf ist.

Neben einer so allgemeinen Beweistechnik wie der Potentialfunktionsmethode wurden für einige wichtige Probleme weitere allgemeine, jedoch auf das Problem zugeschnittene, Techniken entwickelt. Exemplarisch soll hier die *Faktorisierungstechnik* für das soeben behandelte List-Update-Problem vorgestellt werden. Diese Technik (erstmals in [BM85] eingeführt) reduziert die Competitive Analysis eines Online-Algorithmus ALG für das List-Update-Problem unter bestimmten Nebenbedingungen auf die Analyse des Algorithmus ALG für Listen der Länge Zwei.

Das dies eine sehr starke Vereinfachung für die Analyse darstellt ist offensichtlich, denn selbst komplexe Algorithmen werden in ihrem Verhalten und ihren Eigenschaften einfach, wenn sie auf Listen der Länge Zwei operieren. Zudem ist der optimale Offline-Algorithmus OPT bekannt: Er benutzt keine kostenpflichtigen Tauschoperationen und verschiebt das zweite Listenelement nach einem Zugriff genau dann, und nur dann an den Listenanfang, wenn auf selbiges Element unmittelbar ein weiterer Zugriff erfolgt.
Die Grundidee dieser Faktorisierungstechnik besteht in einer abgewandelten Interpretation der anfallenden Kosten. Während man geneigt ist die Kosten von j bei einem Zugriff auf das j-te Element σ_i diesem selbst zuzuschlagen, können diese Kosten auch an den vorhergehenden Elementen festgemacht werden. Jedes davon hat den schnelleren Zugriff auf σ_i blockiert und es wird eine Kosteneinheit für dieses Element gezählt. Das führt zu dem sogenannten partiellen Kostenmodell, in dem nur $j-1$ Kosteneinheiten für den Zugriff auf das j-te Element anfallen und mit $Cost^*_{\mathsf{ALG}}(\sigma_i) = j-1$ bezeichnet werden. Die so vernachlässigten Kosten sind gleich der Länge der Eingabesequenz σ und für jeden Algorithmus identisch. Damit die Analyse auf die reduzierte Listenlänge von Zwei beschränkt werden kann, muß ein Online-Algorithmus ALG zwei Eigenschaften besitzen:

1. Paarweise Stabilität[3]: Sei D eine Elementmenge und $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_n)$ eine Eingabesequenz mit $\sigma_i \in D$, für $1 \leqq i \leqq n$ und sei L eine initiale Liste solcher Elemente. Dann bezeichnet der Term σ_{xy} für zwei beliebige Elemente $x, y \in D$ mit $x \neq y$ eine Eingabesequenz, die aus σ entsteht, wenn alle Einzelelemente verschieden von x oder y gelöscht werden und L_{xy} die Startliste, die aus L durch Entfernen aller Elemente verschieden x oder y entsteht. Damit ist die Länge der Liste L_{xy} durch Zwei begrenzt.
   Ein Algorithmus ALG erfüllt nun die Eigenschaft der paarweisen Stabilität, falls bei Bearbeitung der Eingabesequenz σ (mit Initialliste L) vor und nach den Eingaben $\sigma_i \in \{x, y\}$ die beiden Elemente x und y immer in der selben Ordnung in der Liste auftreten, wie bei den korrespondierenden Eingaben σ_i aus σ_{xy} in der Liste, die ALG bei Bearbeitung von σ_{xy} (mit Initialliste L_{xy}) verwaltet.
   Es wird schnell ersichtlich, daß ein Algorithmus nur dann die paarweise Stabilität garantieren kann, wenn er auf kostenpflichtige Tauschoperationen verzichtet. Dies ist ein Grund, warum der optimale Offline-Algorithmus OPT diese Eigenschaft nicht erfüllt.
2. Kostenunabhängigkeit: Ein Algorithmus ALG heißt kostenunabhängig, wenn seine Entscheidungen unabhängig vom Kostenmodell sind, d. h. ALG führt sowohl im partiellen als auch im vollen Kostenmodell bei gleicher Eingabesequenz σ die selben Operationen aus.

Kann dann für Eingabesequenzen σ_{xy} an Listen der Länge Zwei

$$Cost^*_{\mathsf{ALG}}(\sigma_{xy}) \quad \leqq \quad \mathsf{c} \cdot Cost^*_{\mathsf{OPT}}(\sigma_{xy}) \tag{8}$$

[3]Diese Eigenschaft wird in der englischen Originalliteratur als „pairwise property“ bezeichnet.

gezeigt werden, so folgt, daß ALG streng c-competitive ist. Wird (8) mit einer additiven Konstante bewiesen:

$$Cost^*_{\mathsf{ALG}}(\sigma_{xy}) \;\leqq\; \mathsf{c}\cdot Cost^*_{\mathsf{OPT}}(\sigma_{xy}) + \mathsf{a}\ ,$$

so ist ALG c-competitive. Das Hauptargument liegt in der abgewandelten Kosteninterpretation. Aus

$$\begin{aligned}
Cost^*_{\mathsf{ALG}}(\sigma) &= \sum_{1\leqq i\leqq n} Cost^*_{\mathsf{ALG}}(\sigma_i) \\
&\qquad\qquad \text{folgt mit der geänderten Kosteninterpretation} \\
&= \sum_{1\leqq i\leqq n}\ \sum_{x\in D} Cost^*_{\mathsf{ALG}}(x,\sigma_i) \\
&\qquad\qquad \text{wobei } Cost^*_{\mathsf{ALG}}(x,\sigma_i) := \begin{cases} 1 & \text{falls } x \text{ vor } \sigma_i \text{ in} \\ & \text{der Liste steht} \\ 0 & \text{sonst} \end{cases} \\
&= \sum_{x\in D}\ \sum_{1\leqq i\leqq n} Cost^*_{\mathsf{ALG}}(x,\sigma_i) \\
&= \sum_{x\in D}\ \sum_{y\in D}\ \sum_{i:\,\sigma_i=y} Cost^*_{\mathsf{ALG}}(x,\sigma_i) \\
&= \sum_{\{x,y\}\subset D}\ \sum_{i:\,\sigma_i\in\{x,y\}} Cost^*_{\mathsf{ALG}}(x,\sigma_i) + Cost^*_{\mathsf{ALG}}(y,\sigma_i) \\
&= \sum_{\{x,y\}\subset D} Cost^*_{\mathsf{ALG}}(\sigma_{xy})\ ,
\end{aligned}$$

wobei für den letzten Umformungsschritt die Eigenschaft der paarweisen Stabilität von ALG essentiell ist.

Die weiteren Details und die Korrektheitsbeweise dieser Listen-Faktorisierungstechnik — insbesondere die Argumentation über OPT — gehen über die einführende Darstellung dieses Kapitels hinaus. Sie können in [BEY98, Seiten 13–17], nachgelesen werden. Dort werden auch weitere Beweistechniken für das List-Update-Problem, wie die auf der Listen-Faktorisierung aufbauende Technik der Phasenunterteilung [Alb98b], vorgestellt.

2.3 Vor- und Nachteile der Competitive Analysis

2.3.1 Vorteile

Das erste und wohl wichtigste Merkmal eines Online-Algorithmus ALG, der als c-competitive nachgewiesen ist, besteht in der *Garantie* der Qualität der von ALG erzeugten Lösungen. Wie am Beispiel des List-Update-Problems demonstriert, erzeugt der MTF-Algorithmus höchstens doppelt so hohe Zugriffskosten,

als bei einer optimalen Lösung entstehen. Dies ist ein sehr zufriedenstellendes Ergebnis, welches auch die empirisch festgestellten Probleme anderer List-Update-Algorithmen überwindet (siehe [ST85] für eine ausführliche Diskussion).
Der wichtigste Grund für solch starke und treffliche Resultate und das Vertrauen in die Analyseergebnisse rührt von der vollständigen Abstraktion jeglicher Eingabeverteilung her. Damit wird die Analyse extrem robust. So wird sichergestellt, daß die Entscheidungen des Online-Algorithmus niemals zu katastrophal sein können. Zudem ist das Verhalten derartiger Online-Algorithmen auch in neuen Situationen mit unbekannten Eingabeverteilungen gut — man befindet sich also „auf der sicheren Seite".
Selbst wenn einige untere Schranken für den Competitive Ratio eines Online-Problems für praktische Anwendungen recht hoch erscheinen, so sind sie doch ohne a priori Wissen entstanden. Falls dann Annahmen oder Wissen über die Eingabeverteilung zur Verfügung stehen, kann eine andere Analysetechnik durchaus Resultate liefern, die das Systemverhalten besser erklären. Stellen sich aber die getroffenen Annahmen als falsch heraus, so stürzt die Lösungsqualität nicht gleich ins Bodenlose ab. Zusätzlich stellt der Competitive Ratio einen guten Maßstab dar, um die eingabeverteilungsabhängigen Analysen zu bewerten. Andererseits kann es vorkommen, daß keine Annahme über die Eingabeverteilung getroffen werden kann, da selbige nicht nur unbekannt, sondern auch schwierig zu analysieren oder zu charakterisieren sein kann oder gar nicht existiert.

Das Modell der Competitive Analysis führt zu Aussagen über Konstruktionsprinzipien von Online-Algorithmen für ein bestimmtes Problem oder eine Problemklasse. Derartiges Wissen ist ein idealer Ausgangspunkt für das Design von Algorithmen für ein konkretes Online-Problem.
Nicht zuletzt soll noch angemerkt werden, daß die Competitive Analysis auf verschiedene Parameter eines Online-Problems simultan angewandt werden kann. Dabei können gegebenenfalls komplexe Abhängigkeiten auftreten.

2.3.2 Nachteile

Die Kritiker dieser Analysemethode werden nicht müde die Nachteile der Competitive Analysis als gravierend darzustellen. Der massivste Vorwurf lautet: Das Verfahren sei zu grob und seine Ergebnisse zu schlecht. Die Competitive Analysis geht davon aus, daß *jede* Eingabeverteilung möglich sei, sucht sich dann die ungünstigste heraus und behauptet, daß diese mit einer Wahrscheinlichkeit von 1 aufträte. So erhält man hohe Werte oder Schranken für den Competitive Ratio auch für Algorithmen, die sich in der Praxis bewährt haben. Das Beispiel des k-Paging-Problems (siehe Seite 10) wird dabei bevorzugt zitiert. Die untere Schranke von k für den Competitive Ratio bedeutet, daß der Einsatz eines Cache-Speichers keine Vorteile erbringt. Das ist unter extrem ungünstigen Bedingungen auch wahr, wird in typischen Anwendungen jedoch nie beobachtet. Die beiden Verdrängungsstrategien LRU (Least-Recently-Used) und FIFO (First-In-First-Out) sind beide k-competitive und somit im Sinne der Competitive Analysis optimal. Trotzdem

erzielt der LRU-Algorithmus in der Praxis erheblich bessere Lösungen als FIFO. Es ergibt sich sofort der nächste Kritikpunkt: Die beiden Algorithmen können von der Competitive Analysis in ihrer Güte nicht unterschieden und klassifiziert werden. Ergo stellen in diesem Beispiel die Resultate keine Entscheidungshilfe dar.

Der Competitive Ratio trifft nur Aussagen relativ zu einer optimalen Lösung. Da jedoch die absoluten Kosten nicht berücksichtigt werden, offenbart sich sofort ein weiteres Problem dieses Analyseergebnisses. Beim Versuch Modellvarianten für ein zu lösendes Online-Problem zu vergleichen, kann schon die Qualität der optimalen Lösungen gravierend verschieden sein. Der Vergleich des garantierten Gütefaktors zur optimalen Lösung läßt dann keine korrekten Rückschlüsse auf die Gesamtlösung zu. Dieser Nachteil tritt schon bei einem einzelnen Online-Algorithmus auf, der je nach Eingabesequenz stark unterschiedliche Kosten verursacht. Die einzig mögliche Interpretation des Competitive Ratios erklärt den maximalen Faktor auch bei den Kosten der Eingabe mit maximalen optimalen Kosten für gültig. Möglicherweise können jedoch derartig hohe absolute Kosten niemals entstehen.

Weitere Kritikpunkte subtilerer Natur lassen sich erst beim Studium von Online-Algorithmen mit guten Competitive Ratios entdecken. Diese Algorithmen sind zum Teil unnatürlich und unpraktikabel, und ihre Arbeitsweise widerspricht der Intuition. So besteht das Ziel einiger Algorithmen nur im Widerstehen des Gegenspielers. Sie versuchen nicht optimal auf der Eingabesequenz zu agieren. Statt dessen verschwenden sie wertvolle Ressourcen, um Aussagen zur optimalen Lösung der bisher bekannten Eingabe zu bestimmen und von diesen Werten ihre Entscheidungen abhängig zu machen.
Im täglichen Leben ist das Lernen aus der Vergangenheit eine richtige Strategie. Auch für einen Online-Algorithmus ist es die einzige Möglichkeit eine Basis zur Entscheidungsfindung aufzubauen. Dabei kann es zum Abspeichern der gesamten bisherigen Eingabe kommen und zu komplexen Rechenoperationen darauf. Der Speicher- und Zeitbedarf kann deshalb, wie oben schon angedeutet, erheblich werden. Ungeachtet dieser Tatsache müssen die zukünftigen Eingaben in keinster Weise mit dem bisher Gesehenen korrelieren. Der Gegenspieler kann das erarbeitete Lernergebnis völlig negieren. Insbesondere hat ein Online-Algorithmus kaum die Möglichkeit eine Option auf die Zukunft zu kaufen, da die Eingabesequenz abrupt enden kann. Deshalb sind Online-Algorithmen dazu verdammt den Geschehnissen hinterherzulaufen und auf die Eingaben in passiver Form zu reagieren. Das ist speziell dann der Fall, wenn ein Algorithmus streng competitive sein muß, wie dies in einigen Online-Problemstellungen natürlicherweise auftritt. Um dieses konterintuitive Verhalten näher zu beleuchten, kann das Versicherungsproblem betrachtet werden: Eine noch so lächerlich geringe Ausgabe zur Versicherung eines neu erworbenen Fahrzeuges gegen Diebstahl stellt ein Verhalten dar, welches nicht competitive ist. Im ungünstigsten Fall wird das Fahrzeug nicht gestohlen! Eine Strategie, die competitive sein soll, kann also frühestens nach dem ersten Totalverlust eine Versicherung abschließen.

Ein weiteres konterintuitives Beispiel wird in Kapitel 5, Seite 30 gezeigt (Vorschau erbringt beim k-Paging-Problem keine Vorteile).

2.4 Eine Gegenüberstellung: Online- zu Approximationsalgorithmen

Die formalen Definitionen der Online-Algorithmen und des Competitive Ratios wurden bereits in Kapitel 2.1 gegeben. Bevor nun ein Vergleich zu Approximationsalgorithmen durchgeführt werden kann, wird auch für diese Algorithmenklasse eine Definition benötigt. In der Literatur sind leider verschiedene formale Definitionen zu finden. Die folgenden Aussagen sind ihnen jedoch gemeinsam:

1. Approximationsalgorithmen sind nicht in der Lage für alle möglichen Eingaben optimale Lösungen zu berechnen. Statt dessen bestimmen sie Näherungslösungen.
2. Die maximale Abweichung der Qualität der Näherungslösung und einer optimalen Lösung für die selbe Eingabe ist durch eine Faktor, genannt Approximationsfaktor, begrenzt.
3. Zur Berechnung der Näherungslösung werden weniger Ressourcen benutzt, als nach bestem Wissensstand zur Bestimmung der optimalen Lösung notwendig sind.

Die beiden ersten Aussagen zeigen einige Ähnlichkeiten zum Konzept des Competitive Ratios. Auch die bisher behandelten Online-Algorithmen sind nicht in der Lage für alle Eingaben eines Problems optimale Lösungen auszurechnen und begnügen sich mit einer Näherungslösung. Ebenso ist die Garantie der Lösungsqualität gleich: Es wird ein Faktor angegeben, um den die Qualität der Näherungslösung maximal von der optimalen Lösung abweichen kann. Bei der formalen Definition des Approximationsfaktors gibt es jedoch einige Varianten: Bezeichne I eine beliebige Eingabe. Für ein Kostenminimierungsproblem nennt man einen Algorithmus ALG ρ-approximativ, falls gilt:

$$\forall I: \quad Cost_{\mathsf{ALG}}(I) \leqq \rho \cdot Cost_{\mathsf{OPT}}(I)$$

Der Faktor ρ wird Approximationsfaktor genannt. Er ist immer größer oder gleich 1. Bei Gewinnmaximierungsproblemen wird diese Definition angepaßt, aber es ist sowohl die Variante mit $\rho_1 \geqq 1$ als auch mit $\rho_2 \leqq 1$ bekannt:

$$\begin{array}{lrcll} & \forall I: & Perf_{\mathsf{OPT}}(I) & \leqq & \rho_1 \cdot Perf_{\mathsf{ALG}}(I) \quad [\text{es folgt } \rho_1 \geqq 1] \\ \text{bzw.} & \forall I: & \rho_2 \cdot Perf_{\mathsf{OPT}}(I) & \leqq & Perf_{\mathsf{ALG}}(I) \quad [\text{es folgt } \rho_2 \leqq 1] \end{array} \tag{9}$$

Es ist offensichtlich, daß dabei $\rho_1 = 1/\rho_2$ gilt. Einschränkend muß bemerkt werden, daß auch für den Competitive Ratio eine Definition mit Werten kleiner oder gleich 1 für Gewinnmaximierungsprobleme bekannt ist (siehe dazu [BKM+92, BL99, KP00]). Allerdings findet man diese Variation ausgesprochen selten und

es hat sich die in dieser Arbeit eingeführte Definition (auf Seite 9) durchgesetzt. So wie in der Competitive Analysis zwischen competitive und streng competitive unterschieden werden kann, gibt es auch für Approximationsalgorithmen entsprechende Definitionsvarianten. Wird auf diese Unterscheidung Wert gelegt, so werden „absoluter“ und „asymptotischer Approximationsfaktor“ definiert. Im asymptotischen Fall wird die Gültigkeit des Approximationsfaktors erst ab einer Mindesteingabelänge gefordert [Hoc96] oder die rechte Ungleichungsseite in (9) wird um eine additive Konstante ergänzt [BEY98]. Mit der letztgenannten Definitionsvariante ist es möglich ρ-approximativ und c-competitive mit identischen Formalismen darzustellen. Darum soll nun der wichtige Unterschied zwischen den beiden Algorithmenklassen herausgearbeitet werden.

Betrachtet man die informelle Aussage 3 oben, so benutzt ein Approximationsalgorithmus weniger Ressourcen — vorwiegend Rechenzeit — um eine Näherungslösung zu bestimmen, als es für eine optimale Lösung notwendig wäre. Häufig werden deshalb Approximationsalgorithmen mit polynomieller Laufzeit zum Lösen von $\mathcal{NP}$-harten Problemen eingesetzt [Hoc96]. In einigen wenigen Fällen wird lediglich das Polynom der Laufzeit verringert (z. B. [Pre99]). Ein Approximationsalgorithmus verfügt folglich über alle notwendigen Informationen (die vollständige Eingabe), um eine optimale Lösung zu bestimmen. Wegen der zu hohen Zeitkomplexität wird jedoch eine schnell zu berechnende und beweisbar gute Näherunglösung bevorzugt.

Die Situation für Online-Algorithmen liegt dagegen völlig konträr. Es stehen nicht die für eine optimale Lösung notwendigen Informationen zur Verfügung, wenn die Entscheidungen über Teillösungen getroffen werden müssen. Deshalb handelt es sich um ein informationstheoretisches Problem. Auf diese strikte Unterscheidung zwischen *komplexitätstheoretischem* (Approximationsalgorithmen) und *informationstheoretischem Maß* (z. B. Online-Algorithmen) soll hier besonders hingewiesen werden.

Nun wird auch verständlich, warum der Ressourcen- bzw. der Zeitverbrauch von Online-Algorithmen eine untergeordnete Rolle spielt und frühestens an zweiter Stelle in einer Untersuchung betrachtet wird. Es wurden in der Literatur mehrfach Online-Algorithmen vorgestellt, die $\mathcal{NP}$-harte Teilprobleme lösen müssen (z. B. in [LMS99]) oder exponentielle Laufzeit haben (z. B. der WFA-Algorithmus in metrischen Räumen mit unendlichen vielen Punkten). Die Fragestellung der Algorithmenuntersuchung lautet eben: Wie gut kann unter einem Informationsregime agiert werden, bei dem die Entscheidungen mit unvollständigem Wissen getroffen werden müssen?

Es kann zudem ausgesagt werden, daß die Competitive Analysis nicht auf Online-Algorithmen beschränkt bleibt. Ein weiteres Beispiel für Entscheidungsfindungen unter unvollständigen Informationen sind verteilte Algorithmen. Eine Arbeit zu dieser Algorithmenklasse stellt [AADW94] dar.

Natürlich sind auch Überschneidungen möglich, wenn ein Algorithmus sowohl in die Klasse der Approximationsalgorithmen fällt, als auch competitive ist. So ist z. B. der Greedy-Algorithmus für das Problem des Maximum-Kardinalitäts-

Matchings 2-approximativ. Da er auch für das Online-Matching-Problem funktioniert, ist dieser Algorithmus ebenfalls 2-competitive. Andererseits zeigt eine einfache Gegenspielerstrategie, daß 2 eine untere Schranke für den Competitive Ratio darstellt [KVV90], während ein 1.5-approximativer Linearzeitalgorithmus bekannt ist [Pre99].

2.5 Modifikationen und neue Analysemethoden

Einige der Resultate, die Untersuchungen mittels der Competitive Analysis hervorgebracht haben, sind in ihren Aussagen deprimierend und werden nicht durch Erfahrungen aus empirischen Studien bestätigt. Deshalb wurde schon kurze Zeit nach der Anerkennung der Competitive Analysis als Untersuchungsmethode nach Auswegen gesucht, um diese und andere im Kapitel 2.3.2 beschriebene Nachteile aufzuheben. Es muß jedoch beachtet werden, daß eine untere Schranke für den Competitive Ratio eines Online-Problems *prinzipiell* von keinem Online-Algorithmus unterboten werden kann, d. h. ein solch schlimmer Fall *kann immer* auftreten.
Um also das Analyseresultat, d. h. die Zahl bzw. Funktion des Competitive Ratios zu „verbessern“, gibt es nur eine begrenzte Anzahl von prinzipiellen Möglichkeiten:

1. Einschränkung bzw. Abschwächung der Omnipotenz von Gegenspielern
2. Stärkung der Informationsbasis für die Online-Algorithmen
3. Einführen von neuen „Maßzahlen“ bzw. „Bewertungsfunktionen“, die, inspiriert von der Competitive Analysis, Abwandlungen der Definition des Competitive Ratios darstellen.

Im folgenden werden die aus der Literatur bekannten Vorschläge dargestellt und diskutiert. Im letzte Abschnitt dieses Teilkapitels wird der Versuch unternommen, diese Ideen zu bewerten.

2.5.1 Randomisierte Online-Algorithmen

Die Anpassung der Methode der Competitive Analysis zur Untersuchung von randomisierten Online-Algorithmen stellt wohl ihre erfolgreichste Modifikation dar. Die grundlegende Arbeit dazu wurde schon im Jahr 1990 veröffentlicht (Journalversion ist [BDBK+94]), und stützt sich z. T. auf erste Überlegungen in [RS89] und in den Vorarbeiten zu [BLS92, FKL+91]. Unterdessen sind gleichwertige Anteile von Untersuchungsergebnissen zu randomisierten und den klassischeren deterministischen Online-Algorithmen in aktuellen Veröffentlichungen zu finden. Deshalb wird nun eine ausführlichere Abhandlung dieser Analysemethode gegeben.

Ein randomisierter Online-Algorithmus benutzt Zufallsexperimente, welche die Entscheidungen beeinflussen. Für eine Eingabesequenz σ ergibt sich damit statt eines Wertes für die Kosten- bzw. Gewinnfunktion ein Erwartungswert. Es gibt

eine weitere Interpretation für das Konzept von randomisierten Online-Algorithmen: Bei jedem Auswerten der Zufallsquelle wird ein deterministischer Online-Algorithmus aus einer Klassen von Algorithmen ausgewählt, der dann das nächste Stück der Eingabesequenz verarbeitet.
Für den omnipotenten Gegenspieler der Competitive Analysis stellt die Klasse der randomisierten Online-Algorithmen in keinster Weise etwas Neues dar. Die Gegenbeispiele, welche den schlechtesten Fall illustrieren und nachweisen, können auch gegen derartige randomisierte Online-Algorithmen konstruiert werden, solange der Algorithmus vollständig bekannt ist. Letzteres schließt ein, daß die Ausgaben der Zufallsquelle mit in der Wissensbasis des Gegenspielers sind. Deshalb wurden Gegenspielertypen entworfen, die nur noch einen beschränkten Zugang zu dem Ausgang der Zufallsexperimente besitzen:

Der blinde Gegenspieler (BL): Diesem Gegenspielertyp wird es nicht erlaubt in irgendeiner Weise von dem Ausgang der Zufallsexperimente des Online-Algorithmus Kenntnis zu erlangen. Ansonsten hat er das vollständige Wissen über den Online-Algorithmus, die darin benutzten Verteilungsfunktionen der Zufallsquelle und ihm steht immer noch beliebige Berechnungskraft zur Verfügung. Er darf auch selbst Zufallsexperimente ausführen, um die Eingabesequenz gegen den randomisierten Online-Algorithmus zu erzeugen. Aus der Beschränkung des Gegenspielers ergibt sich jedoch die folgende Spielregel: Der blinde Gegenspieler bzw. die für ihn angegebene Strategie, muß die Eingabesequenz vollständig a priori aufstellen, d. h. ohne auf die Entscheidungen des Online-Algorithmus reagieren zu können. Im deterministischen Fall konnten durch die vollständige Kenntnis des Online-Algorithmus diese Entscheidungen im voraus bestimmt werden und so in die Konstruktion der Eingabesequenz eingehen. Dies ist nun nicht mehr möglich. Aus diesem Grund wurde für diese Arbeit auch die Bezeichnung *blinder Gegenspieler* gewählt, die von dem im englischen Sprachraum benutzten Begriff „oblivious adversary“ etwas abweicht.
Die Competitive Analysis vergleicht nun den Erwartungswert der Kosten des randomisierten Online-Algorithmus auf eine Eingabesequenz σ — bezeichnet mit $E[Cost_{\mathsf{ALG}}(\sigma)]$ — mit den Kosten der optimalen Lösung für diese Eingabe.

Definition 2: (c-competitive gegen den blinden Gegenspieler)

Ein randomisierter Online-Algorithmus ALG heißt c-competitive gegen den blinden Gegenspieler, falls gilt:

$$\exists\, a \in \mathbb{R} :\ \forall \sigma :\quad E[Cost_{\mathsf{ALG}}(\sigma)] \ - \ c \cdot Cost_{\mathsf{OPT}}(\sigma) \ \leqq \ a \ .$$

Der Competitive Ratio, der sich wie üblich aus der Definition von c-competitive ableitet, erhält eine andere Notation: $\overline{\mathcal{R}}_{\mathsf{BL}}$, wobei der Querstrich den randomisierten Fall verdeutlicht und der Index den Gegenspielertypen angibt.

Die nächsten beiden Gegenspieler sind *adaptiv*, d. h. sie erfahren im nachhinein die Entscheidungen, die der Online-Algorithmus getroffen hat und können dieses Wissen für ihre Zwecke ausnutzen.

Der adaptive Online-Gegenspieler (AON): Dieser Gegenspielertyp läßt sich am besten durch eine Beschreibung des Spiels zwischen Online-Algorithmus ALG und Gegenspieler definieren. Der Gegenspieler konstruiert die nächste Eingabe der Sequenz σ *und* muß selbst die Rolle eines Online-Algorithmus wahrnehmen und eine gültige Entscheidung im Sinne des Online-Problems treffen, ohne sie später revidieren zu dürfen. Erst danach wird dem Gegenspieler die Entscheidung des Online-Algorithmus ALG bekannt gegeben und das Spiel geht in die nächste Runde. Dort darf dann die letzte Ausgabe von ALG für die Konstruktion der weiteren Eingaben durch den Gegenspieler genutzt werden.
Die Kosten des Online-Algorithmus ALG werden in dieser Form der Competitive Analysis nun mit den erreichten Kosten der Lösung des adaptiven Online-Gegenspielers verglichen.

Definition 3: (c-competitive gegen den adaptiven Online-Gegenspieler)

Ein randomisierter Online-Algorithmus ALG heißt c-competitive gegen den adaptiven Online-Gegenspieler, falls gilt:

$$\exists\, a \in \mathbb{R} :\ \forall\, \sigma :\quad E[Cost_{\mathsf{ALG}}(\sigma) \ - \ c \cdot Cost_{\mathsf{AON}}(\sigma)] \ \leqq \ a \ .$$

Da die Eingabesequenz vom Ausgang der Zufallsexperimente in ALG abhängt, ist $Cost_{\mathsf{AON}}(\sigma)$ selbst eine Zufallsvariable. Der Competitive Ratio erhält die Bezeichnung $\overline{\mathcal{R}}_{\mathsf{AON}}$ und ist in üblicher Weise als das Infimum über alle Werte von c definiert, für die ALG c-competitive gegen den adaptiven Online-Gegenspieler ist.

Der adaptive Offline-Gegenspieler (AOF): Dieser Gegenspielertyp verfügt über den Vorteil, nach jeder von ihm bestimmten Eingabe der Sequenz σ, sofort die Reaktion des Online-Algorithmus ALG zu lernen und für die weitere Konstruktion von σ verwenden zu können. Im Gegensatz zum adaptiven Online-Gegenspieler bestimmt er selbst aber erst nach dem vollständigen Aufstellen der Eingabesequenz eine optimale Lösung, was einem Offline-Verhalten entspricht. Die Kosten dieser optimalen Lösung $Cost_{\mathsf{OPT}}(\sigma)$ ist nun ebenfalls eine Zufallsvariable, da die Eingabesequenz σ in Abhängigkeit der zufallsunterstützten Entscheidungen von ALG entworfen wurde. Es ergibt sich:

Definition 4: (c-competitive gegen den adaptiven Offline-Gegenspieler)

Ein randomisierter Online-Algorithmus ALG heißt c-competitive gegen den adaptiven Offline-Gegenspieler, falls gilt:

$$\exists\, a \in \mathbb{R} :\ \forall\, \sigma :\quad E[Cost_{\mathsf{ALG}}(\sigma) \ - \ c \cdot Cost_{\mathsf{OPT}}(\sigma)] \ \leqq \ a \ .$$

Und der entsprechend definierte Competitive Ratio erhält $\overline{\mathcal{R}}_{\mathsf{AOF}}$ als Bezeichnung. Aus Gründen der Übersichtlichkeit wurden oben nur die Definitionen für Kostenminimierungsprobleme angegeben. Für Gewinnmaximierungsprobleme übertragen sich die Definitionen in gleicher Weise, wie schon im Kapitel 2.1 für den Fall der deterministischen Online-Algorithmen geschehen.

Die soeben recht informal eingeführten Gegenspielertypen müssen für jedes zu betrachtende Online-Problem konkretisiert werden. Trotzdem wird schnell ersichtlich, daß für einen randomisierten Online-Algorithmus ALG gilt:

$$\overline{\mathcal{R}}_{\mathsf{BL}}(\mathsf{ALG}) \;\leqq\; \overline{\mathcal{R}}_{\mathsf{AON}}(\mathsf{ALG}) \;\leqq\; \overline{\mathcal{R}}_{\mathsf{AOF}}(\mathsf{ALG}) \;.$$

Die Kraft der Gegenspielertypen steigt also vom blinden Gegenspieler über den adaptiven Online-Gegenspieler bis zum adaptiven Offline-Gegenspieler an. Wird statt eines randomisierten ein deterministischer Online-Algorithmus in die Definitionen 2 bis 4 eingesetzt, so überträgt sich jede der drei Definitionen zu der Definition 1 für deterministische Online-Algorithmen. Diese Tatsache wird als ein Argument dafür angeführt, daß diese Gegenspielertypen für die Analyse von randomisierten Online-Algorithmen eine sinnvolle Erweiterung des ursprünglichen Konzeptes darstellen.

In dem Aufsatz [BDBK+94] wurden weitere wichtige Zusammenhänge der verschiedenen Gegenspielertypen aufgedeckt, die in dieser Arbeit ohne Beweise angegeben werden.

Satz 1: ([BDBK+94])

Ist für ein Problem ein randomisierter Online-Algorithmus bekannt, der c*-competitive gegen jeden adaptiven Offline-Gegenspieler ist, so existiert für dieses Problem auch ein deterministischer Online-Algorithmus, der* c*-competitive ist.*

Mit der Aussage dieses Satzes wird deutlich, daß die Stärke des Gegenspielers im deterministischen Fall und des adaptiven Offline-Gegenspielers gleich ist. Das macht diesen Gegenspielertypen häufig uninteressant für Analysen, da sich oberflächlich betrachtet der Wert des Competitive Ratios nicht vermindern läßt. Der Wert des Competitive Ratios $\overline{\mathcal{R}}_{\mathsf{AOF}}$ ist dann interessant, wenn noch keine gute Abschätzung für den Competitive Ratio $\mathcal{R}$ im deterministischen Fall vorliegt. Leider ist der Beweis von Satz 1 ein reiner *Existenzbeweis*. In [DM97] wurde gezeigt, daß es kein allgemeingültiges Verfahren geben kann, um aus einem randomisierten Online-Algorithmus, der c-competitive gegen den adaptiven Offline-Gegenspieler ist, einen ebenso guten deterministischen Online-Algorithmus zu konstruieren. Jedoch konnten solche Verfahren im Einzelfall für abstrakte Problemklassen angegeben werden. So wird in [BDBK+94] für die sehr mächtige Klasse der Metrischen-Task-Systeme (MTS; u. a. ist das k-Server-Problem in dieser Klasse enthalten) eine explizite Konstruktion von deterministischen Online-Algorithmen mit Competitive Ratio $(1+\varepsilon)\cdot\mathsf{c}$ (für jedes $\varepsilon > 0$) aus randomisierten Online-Algorithmen, die c-competitive gegen jeden adaptiven Offline-Gegenspieler sind, angegeben.

Der zweite Satz aus obig zitierter Arbeit gibt tiefere Einsichten in die Möglichkeiten und Grenzen der randomisierten Online-Algorithmen.

Satz 2: ([BDBK+94])

Sei für ein Online-Problem der randomisierte Online-Algorithmus ALG *c-competitive gegen jeden adaptiven Online-Gegenspieler, und existiere ein randomisierter Online-Algorithmus, der* d*-competitive gegen den blinden Gegenspieler ist, so ist* ALG *$(c \cdot d)$-competitive gegen jeden adaptiven Offline-Gegenspieler.*

Die Sätze 2 und 1 können auch kombiniert werden. Damit folgt z. B., daß der Competitive Ratio $\overline{\mathcal{R}}_{\mathsf{AON}}$ eines randomisierten Online-Algorithmus niemals kleiner als die Quadratwurzel des Competitive Ratios $\mathcal{R}$ eines optimalen deterministischen Online-Algorithmus sein kann, denn $\overline{\mathcal{R}}_{\mathsf{BL}} \leqq \overline{\mathcal{R}}_{\mathsf{AON}}$ und aus Satz 2 und 1 würde sich die Existenz eines besseren deterministischen Online-Algorithmus schlußfolgern.
Wie man den Aussagen der beiden obigen Sätze entnehmen kann, verspricht die Competitive Analysis randomisierter Online-Algorithmen gegen den blinden Gegenspieler die kleinsten Werte für den Competitive Ratio. In der Literatur sind vorwiegend Untersuchungen zu diesem Gegenspielertypen zu finden. Diese Analysen geschehen in der Hoffnung, daß der Competitive Ratio $\overline{\mathcal{R}}_{\mathsf{BL}}$ besser mit den empirischen Studien zusammenpaßt, als ein sehr hoher Competitive Ratio $\mathcal{R}$ für einen deterministischen Online-Algorithmus.

Für das k-Paging-Problem wurden mehrere randomisierte Online-Algorithmen entwickelt und analysiert. Dabei konnten verschiedene Analysetechniken und ein tieferes Verständnis für die verschiedenen Gegenspielertypen entwickelt werden. Es gelang optimale randomisierte Online-Algorithmen für dieses Problem zu entwickeln. Zuerst konnte [MS91] den Algorithmus Partition und später [ACN96] den einfacheren und einfacher zu analysierenden randomisierten Online-Algorithmus Equitable als H_k-competitive gegen den blinden Gegenspieler nachweisen. Dabei ist H_k die k-te harmonische Zahl:

$$H_k \;:=\; \sum_{i=1}^{k} \frac{1}{i} \;.$$

Es ist wohlbekannt, daß für $k \geqq 1$: $\ln k \;<\; H_k \;\leqq\; 1+\ln k$ gilt. Der erste Beweis der unteren Schranke von H_k für den Competitive Ratio randomisierter Online-Algorithmen für das k-Paging-Problem gegen den blinden Gegenspieler ist in [BLS87] gegeben. Dabei wurde eine sehr allgemeine und leicht einsetzbare Technik zum Beweisen unterer Schranken gegen den blinden Gegenspieler entwickelt. Es ist eine Anwendung von spieletheoretischen Aussagen aus [Yao77] und wird deshalb auch als *Yao's Lemma* bezeichnet. Zum Nachlesen und Verstehen der korrekten Anwendung sowie der Einordnung dieser Technik in die Theorie der Competitive Analysis ist unbedingt Kapitel 8 in [BEY98] zu empfehlen.

Auch für das List-Update-Problem wurde Yao's Lemma benutzt, um eine untere Schranke für den Competitive Ratio gegen den blinden Gegenspieler zu erhalten.

[Tei93] zeigt, daß $\overline{\mathcal{R}}_{\mathsf{BL}} \geqq 3/2 - 5/(l+5)$ für Listenlänge l gilt, d. h. die untere Schranke ist de facto 1.5 . Schon für den „mittelstarken" adaptiven Online-Gegenspieler gilt jedoch $\overline{\mathcal{R}}_{\mathsf{AON}} \geqq 2$ [RWS94]. In selbiger Veröffentlichung wird auch der randomisierte Online-Algorithmus BIT angegeben:
Jedes Listenelement ist mit einem zusätzlichen Bit ausgestattet. Initial werden diese Bits durch unabhängige Zufallsexperimente mit gleichförmiger Verteilung gesetzt. Erfolgt ein Zugriff auf ein Listenelement, so wird sein Bit invertiert. Danach wird das Element, wie im MTF-Algorithmus, an den Listenanfang verschoben, wenn sein Bit gleich 1 ist. Anderenfalls verbleibt das Element an seinem Ort. Die Analyse des Algorithmus BIT ergibt im partiellen Kostenmodell $\overline{\mathcal{R}}_{\mathsf{BL}}(\mathsf{BIT}) = 1.75$ und für das volle Kostenmodell ist $1.625 \leqq \overline{\mathcal{R}}_{\mathsf{BL}}(\mathsf{BIT}) \leqq 1.75$ bekannt.
In [Alb98b] wird eine ganze Klasse von List-Update-Algorithmen vorgestellt. Sie heißt Timestamp(p) und ist für $p \in [0,1]$ definiert. Solange $p \in (0,1)$ gilt, handelt es sich um randomisierte Online-Algorithmen, und bei geeigneter Wahl von p ist ein Competitive Ratio gegen den blinden Gegenspieler von $\overline{\mathcal{R}}_{\mathsf{BL}}(\mathsf{Timestamp}(p)) \leqq \phi$ erreichbar, wobei $\phi := (\sqrt{5}+1)/2$ den goldenen Schnitt bezeichnet. In den Grenzwerten von p entspricht Timestamp(1) dem MTF-Algorithmus und Timestamp(0) ist ebenfalls ein deterministischer Online-Algorithmus, der, anstatt die Listenelemente nach dem letzten Zugriffszeitpunkt geordnet zu halten (wie bei MTF), die Elemente innerhalb der Liste nach dem vorletzten Zugriffszeitpunkt ordnet. Dazu sind pro Listenelement einige Speicherbits zum Merken des letzten Zugriffszeitpunktes notwendig. Der Competitive Ratio ist ebenfalls $\mathcal{R}(\mathsf{Timestamp}(0)) = 2$.
In [ASW95] wurden die Algorithmen BIT und Timestamp(0) geschickt zu einem neuen Algorithmus kombiniert. Dieser randomisierte Online-Algorithmus trägt den Namen COMB und wählt vor dem Bearbeiten der Eingabe mit Wahrscheinlichkeit 0.8 den BIT-Algorithmus und mit der Restwahrscheinlichkeit von 0.2 den Algorithmus Timestamp(0) aus. Danach wird die gesamte Eingabesequenz mit dem gewählten Algorithmus bearbeitet. Erstaunlicherweise erreicht COMB den besten bisher bekannten Competitive Ratio für das List-Update-Problem gegen den blinden Gegenspieler von $\overline{\mathcal{R}}_{\mathsf{BL}} = 1.6$. Diese Analyse wurde mittels der im Kapitel 2.2, Seite 17 vorgestellten Listen-Faktorisierungstechnik und der darauf aufbauenden Phasenunterteilungstechnik durchgeführt, mit der sich auch BIT analysieren läßt. Die untere Schranke im Competitive Ratio $\overline{\mathcal{R}}_{\mathsf{BL}}(\mathsf{COMB})$ ist ebenfalls 1.6 und wird unter Anwendung von Yao's Lemma gezeigt.

In der Literatur, auch der soeben zitierten, ist immer wieder der mehr oder weniger direkten Vergleich zwischen dem Competitive Ratio $\mathcal{R}$ für deterministisch Online-Algorithmen und dem Competitive Ratio $\overline{\mathcal{R}}_{\mathsf{BL}}$ randomisierter Algorithmen für das selbe Online-Problem und gegen den blinden Gegenspieler zu finden. Nach Auffassung des Autors entspricht dies einem Vergleich von Äpfeln mit Birnen, da sich die Gegenspieler so stark unterscheiden.
Prof. Allan Borodin hatte im Rahmen eines Tutoriums über randomisierte Online-Algorithmen die Aussage getroffen, daß es eine philosophische Frage sei, ob

man diese Methode der Competitive Analysis für randomisierte Online-Algorithmen als eine Stärkung der Online-Algorithmen oder als eine Abschwächung der Gegenspieler auffaßt. Dabei vertrat er selbst die Ansicht der Stärkung von Online-Algorithmen durch die Hinzunahme von Zufallsexperimenten. Da jedoch durch diese Veränderung die Informationsbasis der Online-Algorithmen in keinster Weise verbessert wird (vergleiche hierzu auch die Anmerkungen im Kapitel 2.4 auf Seite 23), und gegen den omnipotenten Gegenspieler auch keine Absenkung des Wertes des Competitive Ratios möglich ist, plädiert der Autor für die zweite Interpretation: Die Kraft der Gegenspieler wird, besonders stark im Fall des blinden Gegenspielers, eingeschränkt.

Im nächsten Unterkapitel wird ein Analysemodell mit verbesserter Informationsbasis der Online-Algorithmen vorgestellt und danach folgen einige Unterkapitel mit Modellen für abgeschwächte Gegenspieler.

2.5.2 Vorschau

Das Konzept, einen Online-Algorithmus mit einer Vorschau auszustatten, stellt eine Verbesserung seiner Informationsbasis dar, da nun ein Teilwissen über die zukünftigen Anforderungen vorliegt. Albers stellt dazu in [Alb97, Alb98a] zwei Modelle auf.

Definition 5: (ℓ-Vorschau, starke ℓ-Vorschau)

Ein Online-Algorithmus ALG *besitzt eine* ℓ*-Vorschau, wenn ihm bei Bearbeitung der aktuellen Eingabe* σ_i *die weiteren* ℓ *Elemente* $\sigma_{i+1}, \ldots, \sigma_{i+\ell}$ *der Eingabesequenz* σ *bekannt sind.*

Ein Online-Algorithmus ALG *besitzt eine starke* ℓ*-Vorschau, wenn ihm bei Bearbeitung der aktuellen Eingabe* σ_i *die weitere Abfolge der Eingabesequenz* σ *so weit bekannt ist, daß darin* ℓ *paarweise und von* σ_i *verschiedene Elemente auftreten.*

Die Definition der starken ℓ-Vorschau ist durch ein Negativresultat der einfachen ℓ-Vorschau motiviert: Beim k-Paging-Problem bleibt der Competitive Ratio auch bei ℓ-Vorschau durch k nach unten begrenzt. Dazu wird eine bekannte Gegenspielersequenz $\sigma = (\sigma_1, \sigma_2, \sigma_3, \ldots)$ für das k-Paging-Problem umgebaut, um die selbe untere Schranke für Online-Algorithmen mit ℓ-Vorschau zu zeigen. Jedes Element σ_i wird in der neuen Eingabesequenz nacheinander $(\ell+1)$-mal wiederholt:

$$\sigma' = (\underbrace{\sigma_1, \sigma_1, \ldots, \sigma_1}_{\ell+1}, \underbrace{\sigma_2, \ldots, \sigma_2}_{\ell+1}, \underbrace{\sigma_3, \ldots, \sigma_3}_{\ell+1}, \ldots) \ .$$

Es ist leicht zu sehen, daß für die Kosten $Cost_{\mathsf{OPT}}(\sigma') = Cost_{\mathsf{OPT}}(\sigma)$ gilt und ein Online-Algorithmus mit ℓ-Vorschau keine Verbesserung seiner Kosten auf σ erreichen kann, da er zum Zeitpunkt einer Verdrängungsentscheidung keine zusätzlich nutzbare Information in der Vorschau besitzt. Das Konstruktionsprinzip dieser unteren Schranke läßt sich auch auf das allgemeinere k-Server-Problem übertragen. Ein Grund für dieses schlechte Resultat ist das Kostenmodell beider Pro-

bleme: Sobald ein Server bereits auf dem Anfragepunkt steht, bzw. eine Seite im Cache vorhanden ist, fallen bei der Anfrage dieses Elementes *keine* Kosten an.

Die Modellvariante der starken ℓ-Vorschau erzwingt im pathologischen Fall eine beliebig lange Vorschau. Zur Rechtfertigung dieser Idee wird angeführt, daß sich in praktischen Fällen, wenn ganze Blöcke von Anfragen gleichzeitig in einem System eintreffen und somit so etwas wie eine Vorschau gegeben ist, die Elemente innerhalb solcher Blöcke stark unterscheiden.
Im Modell der starken ℓ-Vorschau wird eine Verringerung des Competitive Ratios erreicht. Der LRU-Algorithmus kann einfach für das Vorschaumodell angepaßt werden: LRU(ℓ) verhält sich wie LRU auf den im Cache eingelagerten Seiten, die nicht in der Vorschau auftreten. [Alb97] zeigt für starke ℓ-Vorschau mit $\ell \leqq k-2$ die exakte Analyse: $\mathcal{R}(\mathsf{LRU}(\ell)) = k-\ell$. Für $\ell = k-1$ verhält sich LRU(ℓ) wie der optimale Verdrängungsalgorithmus MIN, denn ab $\ell \geqq k-1$ steht auch im Online-Problem so viel Information zur Verfügung, daß der Algorithmus MIN angewandt werden kann. Dieser ist dann natürlich 1-competitive.

Ein etwas realistischeres Vorschaumodell wird in [Alb97] für den Online-Algorithmus LRU(ℓ)-blocked verwandt. Dabei steht nicht mehr zu jedem Zeitpunkt die starke ℓ-Vorschau zur Verfügung, sondern die Eingabe wird in Blöcken gegeben, so daß die Bedingung der starken ℓ-Vorschau immer nur für das erste Element eines Blockes gilt. LRU(ℓ)-blocked arbeitet innerhalb dieses leicht abgeschwächten Modells von Vorschau wie der Algorithmus LRU(ℓ). Es wird gezeigt, daß für $\ell \leqq k-2$ die Aussage $\mathcal{R}(\mathsf{LRU}(\ell)-\mathsf{blocked}) \leqq k-\ell+1$ gilt. Darüber hinaus wurde im selben Aufsatz eine Variante des randomisierten Marking-Algorithmus mit starker ℓ-Vorschau (Marking(ℓ)) studiert und $H_{k-\ell} \leqq \overline{\mathcal{R}}_{\mathsf{BL}}(\mathsf{Marking}(\ell)) \leqq 2 \cdot H_{k-\ell}$ gezeigt.

Young führt in [You91a] eine weitere Variante von Vorschau für das Paging-Problem ein[4]:

Definition 6: (ressourcen-begrenzte ℓ-Vorschau)

Ein Online-Algorithmus ALG für das (h,k)-Paging-Problem besitzt eine ressourcen-begrenzte ℓ-Vorschau, wenn ihm bei Bearbeitung der aktuellen Eingabe σ_i der maximale Präfix der zukünftigen Abfolge der Eingabesequenz σ bekannt ist, so daß ALG darauf niemals mehr als ℓ Seitenfehler erzeugt.

Die Länge der Vorschau hängt also in diesem Fall vom Verhalten des Online-Algorithmus ab. In [Bre98] werden exakte Resultate für das (h,k)-Paging-Problem in diesem Vorschaumodell gezeigt:

$$\mathcal{R}(\mathsf{LRU}(\ell)) \;=\; \frac{k+\ell}{k-h+\ell+1} \;. \tag{10}$$

[4]Das hier betrachtete (h,k)-Paging-Problem ist eine Verallgemeinerung des k-Paging-Problems, bei dem der Gegenspieler nur $h < k$ Speicherseiten benutzen darf. Es wird genauer in Kapitel 2.5.3 auf Seite 33 definiert.

Am selben Ort wird auch ein weiteres Modell für Vorschau vorgeschlagen, bei welchem die Vorschaulänge ebenfalls vom Verhalten des Online-Algorithmus abhängt:

Definition 7: (natürliche ℓ-Vorschau)

Ein Online-Algorithmus ALG *für das (h,k)-Paging-Problem besitzt eine natürliche ℓ-Vorschau, wenn ihm bei Bearbeitung der aktuellen Eingabe σ_i die weitere Abfolge der Eingabesequenz σ so weit bekannt ist, daß darin ℓ paarweise und von σ_i verschiedene Elemente auftreten, die sich nicht im von* ALG *verwalteten Cache befinden.*

Es gelingt dann für LRU(ℓ) mit natürlicher ℓ-Vorschau auch das Ergebnis (10) nachzuweisen. In den Beweisen dafür werden in beiden Varianten der Vorschau Simulationstechniken verwandt, die für die einfache Struktur des Paging-Problems die Vorschau durch Verwendung einer größeren Anzahl von Seiten im Cache „simulieren“ können. Deshalb läßt sich in diesem Beispielproblem die Vorschau durch zusätzliche Ressourcen ersetzen (und bedingt auch umgekehrt). Das ist auch an der Gleichung (10) erkennbar, denn wird h fixiert, so hängt der Wert des Competitive Ratios von $k+\ell$ ab.
Abgesehen von dem (h,k)-Paging-Problem sind bisher keine weiteren Anwendungen der Modelle ressourcen-begrenzter und natürlicher ℓ-Vorschau in der Literatur aufgetreten.

Dahingegen wurde die ℓ-Vorschau und die starke ℓ-Vorschau auch für das List-Update-Problem [Alb98a] und für graphentheoretische Probleme [Ira94, HS94] untersucht.
Für das List-Update-Problem ist auf Grund des Kostenmodells — im vollen Kostenmodell fallen bei jedem Zugriff Kosten an — die ℓ-Vorschau nicht derart nutzlos wie es beim k-Paging-Problem der Fall ist. Jedoch werden sehr große Werte für ℓ benötigt ($\ell \in \Omega(m^2)$, m bezeichnet dabei die Elementanzahl bzw. Länge der Liste), um asymptotisch eine Verringerung des Competitive Ratios zu erreichen. Auch bei starker ℓ-Vorschau ist immer noch ein $\ell \in \Omega(m)$ notwendig, um den Competitive Ratio von 2 zu unterbieten. Die genaue Studie in [Alb98a], die auch die Konstanten exakt bestimmt, zeigt für sehr kurze Listen und moderate Vorschaulängen signifikante Verbesserungen im Competitive Ratio.
Bei den untersuchten graphentheoretischen Problemen handelt es sich um Online-Varianten des Knotenfärbungsproblems in allgemeinen Graphen [HS94] und in d-induktiven Graphen [Ira94] (in die Klasse der d-induktiven Graphen fallen z. B. planare und chordale Graphen). Dabei werden die Knoten online, d. h. nacheinander, aufgedeckt und ihnen muß sofort und unwiderruflich eine Farbe zugewiesen werden, so daß kein benachbarter Knoten die selbe Farbe besitzt. Für dieses Online-Problem fallen die Definitionen von ℓ-Vorschau und starker ℓ-Vorschau zusammen. Trotzdem wurde in beiden Arbeiten gezeigt, daß sich der Competitive Ratio nicht verbessern läßt, solange die Vorschau nur eine konstante Anzahl von Knoten beinhaltet. Die genauen Ergebnisse sind:

- Für die d-induktiven Graphen wird in [Ira94] der Competitive Ratio nach oben und unten in der Ordnung von $\log(n)$ begrenzt (n ist die Knotenzahl des Graphen) und solange $\ell < n/\log(n)$ gilt, bleibt die untere Schranke gültig.
- [HS94] zeigt untere Schranken für den Competitive Ratio von deterministischen Online-Algorithmen $\mathcal{R} \geqq 2n/\log^2(n)$ und von randomisierten Online-Algorithmen gegen den blinden Gegenspieler $\overline{\mathcal{R}}_{\mathsf{BL}} \geqq n/(16\log^2(n))$, die für Vorschaulängen bis $\ell \in O(\log^2(n))$ Gültigkeit besitzen.

Diese Beispiele mögen ausreichen, um die Anwendungen des Konzeptes Vorschau zu demonstrieren. In der Literatur tritt weiterhin, oft auch in Verbindung mit anderen Konzepten, diese Variation der Competitive Analysis auf.

2.5.3 Ressource-Erweiterung

Bei dieser Variante der Competitive Analysis erhält der Online-Algorithmus mehr Ressourcen als der Gegenspieler. Trotzdem wird die Leistung des Online-Algorithmus, die mittels der vergrößerten Ressourcen erfolgt, mit der optimalen Lösung unter der geringeren Ressourceausstattung verglichen. Schon in der grundlegenden Arbeit [ST85] wird das Paging-Problem unter der Annahme untersucht, daß der Online-Algorithmus über k Seiten und der Gegenspieler nur über h Seiten ($h \leqq k$) verfügt. Diese Variante wird auch (h,k)-Paging-Problem genannt. Es gilt:

$$\mathcal{R}(\mathsf{LRU}) \;=\; \frac{k}{k-h+1}\;.$$

Dieses scharfe Resultat gilt sogar für eine ganze Klasse von Paging-Algorithmen, den sogenannten Markierungsalgorithmen[5] [BIRS95, Tor98], die auch LRU beinhaltet.
Der soeben angeführte Competitive Ratio überträgt sich sogar noch auf eine gewichtete Variante des Online-Problems: Die Kosten des Zugriffes auf den Hintergrundspeicher sind für verschiedene Seiten unterschiedlich (gewichtete Seitenkosten). Young zeigte dies in [You91a] für eine Algorithmenklasse mit dem Namen GreedyDual, welche neben einem verallgemeinerten LRU-Algorithmus auch den ersten optimalen Online-Algorithmus für das gewichtete Paging-Problem Balance [CKPV91] enthält.

Neben den Arbeiten zum Paging-Problem wurde die Methode der Ressource-Erweiterung auch zum Studium anderer Online-Probleme angewandt:

- Kürzlich wurde in [Kou99] eine Analyse des WFA-Algorithmus für das k-Server-Problem vorgestellt, bei dem der Gegenspieler nur $h \leqq k$ Server benutzen darf. Es konnten die folgenden beiden oberen Schranken gezeigt werden, wobei

[5] Ein Markierungsalgorithmus (Marking) markiert jede zugegriffene Seite im Cache (d. h. setzt eine Marke auf „1“) und verdrängt nur unmarkierte Seiten, deren Marke auf „0“ steht. Wird die letzte Seite im Cache markiert, so werden sofort alle Marken auf „0“ zurückgesetzt und der Algorithmus führt seine Strategie fort.

$Cost_{\mathsf{OPT},x}(\sigma)$ die Kosten der optimalen Lösung von σ bei Verwendung von x Servern bezeichnet:

$$\begin{aligned} \forall\,\sigma: \quad Cost_{\mathsf{WFA}}(\sigma) &\leqq k \cdot Cost_{\mathsf{OPT},h}(\sigma) + (h-1) \cdot Cost_{\mathsf{OPT},k}(\sigma) \\ \forall\,\sigma: \quad Cost_{\mathsf{WFA}}(\sigma) &\leqq 2 \cdot h \cdot Cost_{\mathsf{OPT},h}(\sigma) - Cost_{\mathsf{OPT},k}(\sigma) \end{aligned}$$

Diese Beweise basieren nicht auf dem schwierigen Potentialfunktionsansatz aus [KP95c], sondern verwenden sehr anschauliche, neu entwickelte Techniken. Mit ihnen soll ein erneuter Versuch gestartet werden, um die k-Server-Vermutung zu beweisen.

- Eine Arbeit zu einem dem k-Server-Problem ähnlichen Problem stellt [KP95a] dar. Dort wird das Online-Transportation-Problem untersucht, bei dem der Online-Algorithmus doppelt so viele Ressourcen als der Gegenspieler benutzen darf. Für diesen Fall konnte ein konstanter Competitive Ratio gezeigt werden, was ohne Ressource-Erweiterung unmöglich ist. Sowohl in dieser Arbeit, als auch in [Kou99] wird der Gegenspielertyp als „schwacher“ bzw. „abgeschwächter“ Gegenspieler bezeichnet.
- In [KP95b] wird ein Online-Planungsproblem betrachtet, bei dem n unabhängige Berechnungsaufträge mit erlaubten Unterbrechungen auf einem Einzelprozessor so ausgeführt werden sollen, daß sich die durchschnittliche Antwortzeit[6] minimiert. Während in [MPT94] für dieses Problem die unteren Schranken $\mathcal{R} \in \Omega(n^{1/3})$ und $\overline{\mathcal{R}}_{\mathsf{BL}} \in \Omega(\log n)$ gezeigt werden, gelingt es den Autoren mit einem geringfügig schnelleren Prozessor für den Online-Algorithmus konstante Competitive Ratios zu erzielen. Dabei sinkt der Competitive Ratio mit ansteigender Prozessorgeschwindigkeit schnell gegen 1.
- [PSTW97] behandelt das gleiche Modell, jedoch in einem Multiprozessorsystem mit Einzelprozessoraufträgen. Mit doppelter Prozessorgeschwindigkeit im Online-Fall können die optimalen Offline-Resultate für die durchschnittliche Antwortzeit garantiert werden. Gleiches gilt, wenn die Aufträge harten Echtzeit-Bedingungen unterliegen, die unbedingt eingehalten werden müssen. Daneben werden in [PSTW97] auch Aussagen getroffen, wieviele zusätzliche Prozessoren im Online-Fall benötigt werden, um diese optimalen Resultate zu erhalten, wenn die Prozessorgeschwindigkeit für Online-Algorithmus und Gegenspieler gleich ist.

Ist die Variante der Competitive Analysis mit Ressource-Erweiterung nun eine rein theoretische Spielerei, oder können deren Aussagen sinnvoll benutzt werden? Eine vorstellbare Anwendung ist im Bereich von Echtzeit-Systemen zu finden. Ist eine Analyse vorhanden, die unter optimalen Entscheidungen die Funktionsfähigkeit des Systems garantiert sowie ein Online-Algorithmus, der bei vergrößertem Ressourcebedarf 1-competitive ist, so folgt die Garantie für die korrekte Systemfunktion, falls dieser Online-Algorithmus einsetzt und der notwendige Ressourcemehrbedarf bereitstellt wird.

[6] Die Antwortzeit ergibt sich aus dem Fertigstellungszeitpunkt abzüglich dem frühest möglichen Startzeitpunkt eines Auftrages.

2.5.4 Comparative Ratio

Diese Analysemethode stellt eine Verallgemeinerung des Competitive Ratios dar und wurde in [KP94a] vorgestellt. Sie ermöglicht den Online-Algorithmus auch gegen weniger starke Gegenspieler zu vergleichen. Dazu werden zwei Algorithmenklassen $\mathcal{A}$ und $\mathcal{B}$ definiert. Typischerweise gilt $\mathcal{A} \subsetneq \mathcal{B}$, dies stellt jedoch keine Notwendigkeit dar. $\mathcal{B}$ ist dabei die Klasse mit der umfangreicheren Informationsbasis oder den leistungsfähigeren Ressourcen.
Als Spiel von Online- und Gegenspieler interpretiert, läuft in der Analyse das Folgende ab: Der Gegenspieler wählt einen Algorithmus $B \in \mathcal{B}$ und der Online-Spieler reagiert mit der Wahl eines Algorithmus $A \in \mathcal{A}$. Dann konstruiert der Gegenspieler die Eingabesequenz σ und die Kosten von A und B auf σ werden ins Verhältnis gesetzt.

Definition 8: (Comparative Ratio)

Seien $\mathcal{A}$ und $\mathcal{B}$ zwei Algorithmenklassen für ein Kostenminimierungsproblem. Der Comparative Ratio $\mathcal{C}(\mathcal{A}, \mathcal{B})$ zwischen beiden Klassen ist definiert als:

$$\mathcal{C}(\mathcal{A}, \mathcal{B}) \;:=\; \max_{B \in \mathcal{B}} \min_{A \in \mathcal{A}} \max_{\sigma} \frac{Cost_A(\sigma)}{Cost_B(\sigma)} \;.$$

In [KP94a] wurde diese Definition nur im Zusammenhang mit dem k-Paging-Problem angegeben. Entsprechende Anpassungen sind für Gewinnmaximierungsprobleme und Eingaben bzw. Bewertungsfunktionen aus kontinuierlichen Zahlenbereichen notwendig und leicht durchführbar.
Werden in der Klasse $\mathcal{B}$ alle Algorithmen zugelassen, also auch der optimale Offline-Algorithmus, so mutiert die Definition 8 zur bekannten Definition 1 des Competitive Ratios, wobei entweder $\mathcal{A}$ auf die Klasse der Online-Algorithmen eingeschränkt wird (das definiert den Competitive Ratio eines Online-Problems), oder $\mathcal{A}$ nur aus einem einzigen zu analysierenden Online-Algorithmus ALG besteht. Aus diesen Fakten ergibt sich, daß der Comparative Ratio eine echte Verallgemeinerung des Konzeptes des Competitive Ratios darstellt.
Als Anwendung wird in [KP94a] der Einfluß von ℓ-Vorschau in Metrischen-Task-Systemen und im k-Paging-Problem untersucht. Bezeichne $\mathcal{L}_i$ die Klasse von Online-Algorithmen mit i-Vorschau für das k-Paging-Problem, so wird ein Comparative Ratio von

$$\mathcal{C}(\mathcal{L}_0, \mathcal{L}_\ell) \;=\; \min\{\ell + 1, k\}$$

gezeigt. Leider erfährt dieses Konzept bisher in der Literatur wenig Anwendung, obwohl es mehrfach ausführlich in Übersichtsartikeln dargestellt ist (z. B. [FW98, Kapitel 3]).

2.5.5 Accommodating Ratio

Diese Variation der Competitive Analysis wurde in der Gruppe um Joan Boyar, Kim S. Larsen und Morton N. Nielsen entwickelt [BL99, BN99, BLN99a, BLN99b]. Um die Motivation nachvollziehen zu können, ist es sinnvoll das erste von ihnen untersuchte Problem zu betrachten.

Das Platzkarten-Problem

Gegeben ist eine Eisenbahnstrecke mit k Stationen (Stationen $1, \ldots, k$, einschließlich der Anfangs- und Endstation). Für einen Zug mit n Plätzen sollen Platzkarten verkauft werden. Die Kunden erscheinen nacheinander und möchten Plätze von Station s bis t ($1 \leqq s < t \leqq k$) reservieren. Die Entscheidung über die Ausstellung von Platzkarten mit eindeutigen und für den Streckenabschnitt noch nicht besetzten Platznummern bzw. der Ablehnung des Kundenwunsches muß sofort und unwiderruflich geschehen. Ziel ist die Maximierung der Anzahl verkaufter Platzkarten (Einheitspreismodell[7]). Die Autoren schränken die Online-Algorithmen zur Lösung dieses Problems auf *faire* Algorithmen ein, d. h. der Algorithmus darf nie eine realisierbare Reservierung ablehnen[8].

Dieses Online-Problem hat Ähnlichkeiten zum Online-Knotenfärbungsproblem von Intervallgraphen [KT92], dem Online-Intervallscheduling [LT94] und zu Zuweisungsproblemen von Verbindungsanforderungen in (optischen) Netzwerken [ABFR94, AAF+96, GGK+97]. Im Detail sind jedoch immer einige entscheidende Unterschiede zu finden.
Die untere Schranke im Competitive Ratio für das Platzkarten-Problem ist $\Omega(k)$. Dabei stellt der Gegenspieler in einer ersten Phase n Anforderungen von Reservierungswünschen über die Gesamtstrecke $(0, k)$. Da der Online-Algorithmus fair sein soll, muß er all diese Anfragen akzeptieren. In einer zweiten Phase werden dann vom Gegenspieler nur noch Anfragen für Platzkarten zwischen zwei benachbarten Stationen generiert, d. h. es werden je n Anfragen für den Streckenabschnitt $(i, i+1)$, $\forall\ 0 \leqq i < k$ erzeugt. In der optimalen Lösung werden alle n Anfragen der ersten Phase abgelehnt und alle Anfragen der zweiten Phase akzeptiert.
Das unfaire an dieser Konstruktion der unteren Schranke besteht in einer Eingabesequenz des Gegenspielers, bei der auch die optimale Lösung nicht alle Kundenwünsche erfüllen kann und die somit einen optimalen Algorithmus impliziert, der selbst nicht fair ist. Der Accommodating Ratio verbietet nun Eingaben mit dieser Eigenschaft.

Definition 9: (Accommodating Ratio)

Sei ALG ein Online-Algorithmus für ein Gewinnmaximierungsproblem mit parametrisierter Ressourcemenge, dessen einzelne Bearbeitungsaufträge abgelehnt

[7] Es wird in der Arbeit [BL99] auch die Modellvariante mit Platzkartenpreisen proportional zur Streckenlänge untersucht.
[8] Eine Form einer Greedy-Strategie.

werden können oder bei Akzeptanz einen Gewinn erbringen. Sei $\mathcal{S}$ die Menge aller Eingaben, für die eine korrekte Lösung mit den vorhandenen Ressourcen und ohne abgelehnte Aufträge existiert. Der Accommodating Ratio von ALG ist das Infimum aller Werte $\mathsf{m} \in \mathbb{R}$, für die gilt:

$$\exists\, \mathsf{a} \in \mathbb{R} :\ \forall\, \sigma \in \mathcal{S} :\qquad Perf_{\mathsf{OPT}}(\sigma) \ \leqq\ \mathsf{m} \cdot Perf_{\mathsf{ALG}}(\sigma) \ +\ \mathsf{a}\ .$$

In [BL99] wird gezeigt, daß jeder faire Online-Algorithmus einen Accommodating Ratio von 2 besitzt. Für spezielle Online-Strategien wie First-Fit und Best-Fit werden kleinere Accommodating Ratios bestimmt.
An diesem Beispiel ist ein fundamentaler Unterschied zwischen dem Competitive und dem Accommodating Ratio erkennbar, der in [BLN99b] noch verstärkt wird: Bei der Untersuchung des Bin-Packing-Problems mit Einheitspreisen werden die Leistungen der Online-Strategien First-Fit und Worst-Fit voneinander separiert. Bei der Competitive Analysis ist Worst-Fit echt besser als First-Fit, bei der Bestimmung des Accommodating Ratios ist es genau umgekehrt.
Auf der anderen Seite ist der Accommodating Ratio ein Spezialfall des Comparative Ratios (Kapitel 2.5.4), da in der Definition des Accommodating Ratios die Gegenspieler in ihren Möglichkeiten beschränkt werden.

Das zentrale Problem, welches in den späteren Kapiteln dieser Arbeit untersucht wird, ist ebenfalls ein Ressourcen-Zuordnungsproblem. Wie sich zeigt, verwenden die Gegenspielerstrategien in den Beweisen der unteren Schranken keine Eingaben, die nicht vollständig vom System erfüllt werden können. Deshalb gelten die dort gezeigten unteren Schranken für den Competitive Ratio auch für das schwächere Maß des Accommodating Ratios. Für die untersuchten Modelle, für die exakte Analysen vorliegen, ergibt sich daraus die spezielle Situation der Gleichheit von Competitive und Accommodating Ratio.

Die Grundidee des Accommodating Ratios wurde in [BLN99a] zu einer allgemeingültigeren Theorie weiterentwickelt, in der der zentrale Begriff die *Accommodating Funktion* ist.

2.5.6 Der diffuse bzw. statistische Gegenspieler

Eine Mischform zwischen der klassischen Variante einen Online-Algorithmus unter einer bekannten Eingabeverteilung zu analysieren und der Competitive Analysis, die sämtliche Verteilungen für die Eingabe zuläßt, ist das Modell des diffusen Gegenspielers [KP94a] bzw. des statistischen Gegenspielers [Rag92][9]. Dabei wird die Eingabe gemäß einer Verteilung D generiert, die aus einer Klasse $\mathcal{D}$ von Verteilungen stammt. Die Lage des Online-Algorithmus wird verbessert, da ihm $\mathcal{D}$ bekannt ist.

[9] Dort wird nur die Grundidee beschrieben und ein Beispiel gegeben. Eine formale und allgemeingültige Definition fehlt.

Definition 10: (c-competitive gegen die Klasse $\mathcal{D}$)

Sei $\mathcal{D}$ eine Klasse von Verteilungen und bezeichne $E_{\sigma\in D}[Cost_X(\sigma)]$ den Erwartungswert der Kosten des Algorithmus X, wenn die Eingabesequenz σ unter der Verteilung D generiert wird. Ein Online-Algorithmus ALG für ein Kostenminimierungsproblem ist c-competitive gegen die Klasse $\mathcal{D}$, falls gilt:

$$\exists\, \mathsf{a} \in \mathbb{R} :\ \forall\, D \in \mathcal{D} :\quad E_{\sigma\in D}[Cost_{\mathsf{ALG}}(\sigma)] \ \leqq\ \mathsf{c}\cdot E_{\sigma\in D}[Cost_{\mathsf{OPT}}(\sigma)] \ +\ \mathsf{a}\ .$$

Die so formulierte Definition entstammt der Darstellung in [BEY98, Seite 74]. Dort wird auch der „$\mathcal{D}$-beschränkte Competitive Ratio“ als das Infimum der Menge aller c, für die ALG c-competitive gegen $\mathcal{D}$ ist, definiert.

Die Definition 10 stellt eine weitere Verallgemeinerung des Competitive Ratios dar und entspricht ihm (Definition 1), falls die Klasse $\mathcal{D}$ sämtliche Verteilungen umfaßt. Weiterhin umfaßt diese Definition andere Konzepte, die vorwiegend in Verbindung mit dem k-Paging-Problem vorgeschlagen wurden, wie z. B. das k-Paging-Problem mit Zugriffsgraphen [BIRS95, IKP96, FR97] (siehe dazu auch das Unterkapitel 2.5.8).

2.5.7 Max/Max-Ratio

Ein völlig neues Maß zur Bewertung von Online-Algorithmen wird in [BDB94] vorgeschlagen. Es versucht die Passivität der Online-Algorithmen zu überwinden und so deren Rolle in der Analyse zu stärken. Wie auch in der Competitive Analysis muß sich ein Online-Algorithmus gegen einen omnipotenten Gegenspieler behaupten, d. h. auch in diesem Fall wird die Analyse eines schlimmsten Falles durchgeführt. Der Gegenspieler konstruiert wiederum die Eingabesequenz σ und die Leistung eines Online-Algorithmus ALG wird durch die verursachten Kosten $Cost_{\mathsf{ALG}}(\sigma)$ bemessen. Zur Verteidigung darf ALG dem Gegenspieler eine Anfragesequenz $\overline{\sigma}$ entgegensetzen, die die selbe Länge wie σ hat ($|\overline{\sigma}| = |\sigma|$), und die vom Gegenspieler optimal gelöst wird. Der Vergleich der Leistung von ALG auf σ findet dann mit den optimalen Kosten auf $\overline{\sigma}$ statt. Formal wird definiert:

Definition 11: (Max/Max-Ratio)

Der Max/Max-Ratio eines Online-Algorithmus ALG für ein Kostenminimierungsproblem ist das Infimum aller Werte $\mathsf{m} \in \mathbb{R}$, für die gilt:

$$\exists\, \mathsf{a} \in \mathbb{R} :\ \forall\, \sigma :\ \exists\, \overline{\sigma}, |\overline{\sigma}| = |\sigma| :\quad Cost_{\mathsf{ALG}}(\sigma) \ \leqq\ \mathsf{m}\cdot Cost_{\mathsf{OPT}}(\overline{\sigma}) \ +\ \mathsf{a}\ .$$

Der Charakter dieser Definition unterscheidet sich gravierend vom Competitive Ratio. Sie impliziert, daß die schlechteste Leistung von ALG mit der schlechtesten Leistung der optimalen Lösung auf der selben Problemgröße verglichen und dieser

Faktor asymptotisch für wachsende Eingabelängen bestimmt wird. Darum kann der Max/Max-Ratio alternativ auch als

$$\limsup_{n\to\infty} \frac{\max_{\sigma,|\sigma|=n} Cost_{\mathsf{ALG}}(\sigma)}{\max_{\overline{\sigma},|\overline{\sigma}|=n} Cost_{\mathsf{OPT}}(\overline{\sigma})}$$

definiert werden. In [BDB94] wird vorwiegend das k-Server-Problem in verschiedenen Varianten mit dieser Methode untersucht. Dabei werden unter anderem die folgende Ergebnisse erzielt:

- Die einfache ℓ-Vorschau im k-Paging-Problem wirkt sich auf den Max/Max-Ratio aus. Er ist bei m verschiedenen Seiten in der Eingabesequenz σ

  $$\frac{m-1}{m-k} \qquad \text{für} \qquad 1 < \ell \leqq m-k \qquad \text{und}$$

  $$\frac{m-1}{\ell} \qquad \text{für} \quad m-k < \ell \leqq m-1$$

  Es bleibt jedoch zu beachten, daß sich der Max/Max-Ratio für Vorschaulängen bis $m-k$ nicht verbessert.

- Für das k-Server-Problem in metrischen Räumen mit begrenzter Punkteanzahl wird ein speicherloser Online-Algorithmus (K-Center) angegeben, dessen Max/Max-Ratio nach oben durch $2k$ beschränkt und der damit höchstens um den Faktor 2 vom Optimum entfernt ist.

Es wird weiterhin vorgeschlagen, mittels der Definition 11, einen Online-Algorithmus, statt mit einem optimalen Offline-Algorithmus, mit einem anderen Online-Algorithmus oder einer Klasse von Online-Algorithmen zu vergleichen. Dabei könnten verschiedene Algorithmen für ein Online-Problem direkt miteinander verglichen werden, und zwar ohne etwas über die optimale Lösung wissen zu müssen. Es ergibt sich zudem die Möglichkeit die Online-Algorithmen bezüglich des Maßes des Max/Max-Ratios zu ordnen.
Bisher sind jedoch keine weiteren Veröffentlichungen bekannt, die den Max/Max-Ratio für Analysen anwenden.

2.5.8 Abschlußbetrachtungen zu den Analysevarianten

Im Zusammenhang mit dem k-Paging-Problem wurden weitere Modifikationen der Analyse vorgeschlagen, von denen in diesem Kapitel nur noch zwei angesprochen werden. Young führt in [You91a] das Konzept des *lockeren Competitive Ratios* ein. Für $\varepsilon, \delta \in \mathbb{R}, 0 < \varepsilon, \delta < 1$ ist ein Online-Algorithmus ALG für das k-Paging-Problem (ε, δ)-locker c-competitive falls für jede Eingabe σ und jedes $n \in \mathbb{N}$ für mindestens $(1-\delta)n$ der Werte von $k \in \{1, 2, \ldots, n\}$ gilt:

$$Cost_{\mathsf{ALG}}(\sigma) \leqq \max\{\mathsf{c} \cdot Cost_{\mathsf{OPT}}(\sigma), \varepsilon \cdot |\sigma|\} \ .$$

Diese Definition bedeutet erstens, daß das Leistungsverhältnis zwischen dem optimalen Algorithmus und ALG auf einer fixierten Eingabe nur noch für die meisten Werte von k gelten muß, womit spezielle Gegenspielereingaben, die für eine bestimmte Modellkonfiguration (in diesem Beispiel der Parameter k) maßgeschneidert wurden, nicht mehr das Ergebnis dominieren. Zweitens interessiert der Faktor c nicht mehr, falls die absoluten Kosten sowieso sehr klein sind, d. h. einen ε-Bruchteil der Eingabenlänge nicht überschreiten. Die von Young durchgeführten Studien zeigen eine dramatische Verbesserung im (ε, δ)-lockeren Competitive Ratio auf, der nun nicht mehr vom Parameter k abhängt.
Dieses Analysemodell hat das Potential auch für einige Online-Planungsprobleme benutzt zu werden, da bei einigen Problemen dieser Problemklasse die Eingabesequenzen zum Beweis der unteren Schranken ebenfalls sehr exakt auf die Ressourcemenge (in dem Fall die Prozessorzahl) ausgelegt sind.

Torng untersucht in [Tor98] das k-Paging-Modell mit totalen Speicherzugriffskosten, d. h. bei jedem Zugriff entstehen Kosten: geringe Kosten von einer Einheit, falls sich die Seite im Cache befindet und größere Kosten von s Einheiten, falls diese Seite nur im Hintergrundspeicher vorhanden ist. In der Analyse geht dann der Parameter s, welcher das Verhältnis zwischen den Zugriffskosten auf den Hintergrundspeicher und den Cache repräsentiert, ein. So erreicht jeder Online-Paging-Algorithmus aus der Klasse der Markierungsalgorithmen (z. B. LRU) einen Competitive Ratio von

$$\mathcal{R}(\mathsf{Marking}) \;=\; \frac{k(s+1)}{k+s} \quad \approx \min\{k, s+1\} \;\; .$$

Die Antriebskraft für die Entwicklung derartig vieler Varianten und Modifikationen des Analysemodells stellt eindeutig das Negativresultat des k-Paging-Problems dar. Dieses besagt einerseits, daß bezüglich der Competitive Analysis gute und schlechte Online-Seitenverdrängungsstrategien wie LRU und FIFO gleich gut sind und so in ihrer Leistung nicht getrennt werden. Andererseits zeigt die untere Schranke von k für den Competitive Ratio, daß das Konzept des Cache-Speichers nicht nutzbringend ist.
Bei genauerer Betrachtung des k-Paging-Problems verwundern diese Resultate jedoch nicht. Schließlich wird in der Competitive Analysis der schlimmste Fall bestimmt und seit dem Aufkommen der Idee von Caches ist bekannt, daß sie in extremen Fällen nicht helfen. Allerdings war die Motivation für den Einsatz von Caches auch eine andere, wie sie in jedem Lehrbuch über Rechnerarchitektur nachgelesen werden kann (z. B. [Hay88, Seite 442]): Die Beobachtung von Programmabläufen zeigte starke Lokalitäten der Datenzugriffe in Zeit und Ort auf. Das Konzept des Caches wurde unter dieser Annahme entwickelt und hat sich bis heute erfolgreich durchgesetzt. Da die abstrakte Formulierung des k-Paging-Problems diese Annahmen unberücksichtigt läßt und trotzdem eine Analyse des schlimmsten Falles durchgeführt wird, sind keine besseren Ergebnisse, d. h. Aussagen, die die guten praktischen Erfahrungen mit Caches widerspiegeln können, zu erwarten.

Viele der in diesem Kapitel vorgestellten Modifikationen der Analysemethode versuchen sich durch Manipulation und Einschränkung der Gegenspieler stückweise der Annahmen des Cache-Konzeptes zu nähern. Teilweise sind die dazu vorgeschlagenen Ideen auch deshalb nicht weiter verfolgt worden, weil sie nur schwierig auf andere Online-Probleme übertragbar sind. Nach Meinung des Autors ist es aber ein sinnvollerer Weg das Online-Problem exakter zu modellieren, statt an der Competitive Analysis Kritik zu üben und selbige dann zu amputieren, um die numerischen Analyseresultate aufzubessern. Auch in dieser Richtung gibt es Arbeiten, die nicht unerwähnt bleiben sollen.

So wird in [BIRS91] erstmals eine Variante des k-Paging-Problems eingeführt, welche die Lokalität der Zugriffe modelliert. Man bedient sich dabei eines sogenannten Zugriffsgraphen. Dieser Graph ist gerichtet und besitzt Schleifen[10]. Jeder Knoten repräsentiert eine Speicherseite, und eine Eingabesequenz muß einem Weg in diesem Zugriffsgraphen entsprechen, d. h. dem Aufruf einer Speicherseite u darf nur der Zugriff auf eine Seite v folgen, falls die Kante (u, v) im Zugriffsgraphen enthalten ist.
Die umfangreiche Studie von LRU in [BIRS91] kann die Güte dieses Algorithmus für das Zugriffsgraphenmodell des k-Paging-Problems mit beliebigen vorgegebenen Graphen bis auf einen Faktor von 2 charakterisieren. Zudem wird gezeigt, daß LRU optimal für Bäume ist. Weiterhin ist der Competitive Ratio von LRU, wie [CN99] zeigt, niemals größer als der von FIFO, egal welcher Zugriffsgraph den Eingabesequenzen zugrunde liegt. [BIRS91] stellt einen weiteren Online-Algorithmus vor, der ebenfalls aus der Klasse der Markierungsalgorithmen stammt. Er heißt FAR und verdrängt diejenige unmarkierte Seite aus dem Cache, die im Zugriffsgraphen den längsten Weg zur aktuellen Seite aufweist. Damit versucht FAR in gewisser Weise den optimalen Algorithmus MIN zu imitieren. [IKP96] gelingt eine verbesserte Analyse von FAR, in der nachgewiesen wird, daß er höchstens einen konstanten Faktor vom optimalen Competitive Ratio des k-Paging-Problems mit ungerichteten Zugriffsgraphen entfernt ist. Für eine verallgemeinerte Problemvariante kann [FK95] das selbe Ergebnis erzielen. Dabei werden sogar Mehrfachreferenzen im Graphen zugelassen, wie sie in einer Multiprogrammumgebung mit mehreren nebenläufig und unabhängigen Flüssen von Speicherzugriffen auftritt. [FR97] ergänzt die theoretischen Untersuchungen mit einer empirischen Studie. „Wahre“ Online-Algorithmen werden in [FM97] vorgestellt. Sie benötigen kein a priori Wissen der Graphenstruktur und können sogar auf sich dynamisch ändernde Zugriffsgraphen adaptieren, solange die Veränderungen moderat bleiben. Diese Algorithmenvarianten erreichen ebenfalls den optimalen Competitive Ratio bis auf einen konstanten Faktor und sind sogar streng competitive.

Das Modell des k-Paging-Problems mit Zugriffsgraphen wird in [KPR92] nochmals ergänzt. Den Kanten des Zugriffsgraphen werden nun Wahrscheinlichkeiten zugeordnet womit dieser Graph zur Beschreibung einer Markowschen Kette wird,

[10] Eine Schleife ist eine Kante, die einen Knoten mit sich selbst verbindet.

die die Eingabesequenzen generiert. Damit stellen diese Studien eine Abkehr von der Competitive Analysis hin zur „klassischeren“ probabilistischen Analyse dar. Obwohl diese Idee sicherlich einen interessanten Weg aufzeigt, um reales Systemverhalten möglichst exakt zu beschreiben und dafür effektive Algorithmen zu entwickeln, geht dieser Ansatz weit über den Inhalt dieses Kapitels hinaus und soll deshalb an dieser Stelle nicht weiter vertieft werden.

Wie man den Kapiteln 2.5.1 bis 2.5.6 ebenfalls entnehmen kann, wenden die Wissenschaftler viel Kreativität auf, um den Wert des Competitive Ratios eines Problems durch Modifikation der Analysetechnik zu verringern. Daran sind folgende zwei Punkte zu kritisieren:

1. Die Zahlenwerte der Untersuchungsresultate mittels modifizierter Analysetechniken liegen typischerweise unterhalb der unteren Schranken für den Wert des Competitive Ratios. Was auf den ersten Blick widersprüchlich erscheint, offenbart bei genauerer Betrachtung ein schwerwiegendes Problem: Zu oft werden die numerischen Ergebnisse gegen verschieden mächtige Gegenspielermodelle bedenkenlos miteinander verglichen. Schon eine Gegenüberstellung solcher Resultate bedarf m. E. einer sehr sensiblen Interpretation. Dahingegen ist der direkte Vergleich kaum in der Lage neue und tiefere Einblicke in das untersuchte Problem und die entworfenen Algorithmen zu erbringen.
2. Falls die Analyse eines Online-Problems einen kleinen und konstanten Competitive Ratio bestimmt (wie im Fall des List-Update-Problems und dem Algorithmus MTF), kann von diesen Aussagen auch in der Praxis profitiert und die vom Competitive Ratio übernommene Leistungsgarantie ausgenutzt werden. Erbringt die Analyse jedoch eine hohe untere Schranke, so ist der Einsatz anderer Untersuchungsmethoden nicht zwingend notwendig. Prof. Amos Fiat sagte dazu einmal: „These numbers means nothing.“ Er wollte damit zum Ausdruck bringen, daß die absoluten Werte der Competitive Ratios nur eine untergeordnete Rolle spielen. Ziel ist es vielmehr mit Hilfe der Methode der Competitive Analysis Online-Algorithmen zu bewerten und zu vergleichen. Es ist anzustreben, unter Zuhilfenahme einer Analysetechnik, das Funktionsprinzip eines guten Algorithmus zu bestimmen und die korrekte Vorhersage zu treffen, daß er eine bessere Leistung als andere Algorithmen für das selbe Problem erbringt. Auf eine überproportionale Betonung einzelner Zahlenwerte soll und kann dann verzichtet werden.

3 Vorausgegangene Forschungsergebnisse

Übersicht: In der Literatur sind vielfältige Modelle studiert worden, die im engeren oder weiteren Sinne in Beziehung zu den späteren Untersuchungen innerhalb dieser Arbeit stehen. Ein Überblick über die relevantesten und als Vorarbeiten wichtigsten Modelle sowie über ihre Untersuchungsergebnisse wird in diesem Kapitel gegeben. Dazu werden zuerst die großen Klassen der Lastbalancierungs- und Planungsprobleme in ihren Online-Varianten behandelt. Dies schließt eine Herausarbeitung des wichtigen Unterscheidungsmerkmals beider Online-Problemklassen ein. In der Vergangenheit wurden jedoch Dutzende von Varianten solcher Modelle analysiert, so daß an dieser Stelle eine Einschränkung notwendig ist. Modellkonzepte, die in den Studien der späteren Kapitel keinen Eingang finden, werden in diesem Kapitel nicht betrachtet.

Der Übergang zur danach behandelten Klasse von Online-Planungsproblemen mit Deadlines (Terminzwang) führt zur Motivation einer neuen Zielfunktion und ändert damit die Problemstruktur. Das letzte Teilkapitel widmet sich Zuordnungsproblemen bzw. Online-Matching-Problemen. Vertreter dieser Problemklasse entstehen teilweise bei der Modellierung von Planungsproblemen mit Deadlines, und sie bilden die direkte Grundlage für die später folgenden Studien.

Eine Diskussion und Abgrenzung der aus der Literatur bekannten Modelle zu denen, die in dieser Arbeit untersucht werden, wird auf das Ende des Kapitels 4 verschoben. Erst dann stehen die vollständigen und formalen Definitionen letztgenannter Modelle zur Verfügung.

Innerhalb dieses Kapitels werden Probleme aus der Literatur aufgearbeitet, in denen vorhandene Ressourcen an Aufträge mit Ressourceanforderungen vergeben werden. Im Online-Szenario treffen diese Aufträge nacheinander und über die Zeit verteilt ein. Es gilt jedoch zwei Klassen solcher Problemstellungen strikt voneinander zu unterscheiden. Bei *Online-Lastverteilungsproblemen* werden die Aufträge sofort den Ressourcen zugeordnet. Dies geschieht unabhängig davon, ob diese Ressource überlastet wird, und die Bearbeitung des Auftrages erst in der Zukunft durchgeführt werden kann. Je nach der zugrundeliegenden praktischen Aufgabenstellung können durchaus derartige Modelle entstehen. Im Gegensatz dazu stehen *Online-Planungsprobleme*[11]. Bei ihnen werden die eingehenden Aufträge in einen Puffer gelegt und die Zuordnungsentscheidungen werden erst getroffen, wenn Systemressourcen frei sind. Daraus ergibt sich der Vorteil, daß Entscheidungen hinausgezögert werden und unter Umständen schon weitere Teile der Eingabesequenz bekannt sind, die in der Lösungsbestimmung genutzt werden können.

3.1 Online-Lastbalancierung

Für die Problemklasse der Lastbalancierung hat sich eine eigene Terminologie herausgebildet, die dem Bereich des parallelen Rechnens entlehnt ist. Die Ressourcen eines Systems werden durch m Maschinen verkörpert. Die Aufträge werden von 1 bis n durchgezählt und dies soll auch ihre Reihenfolge innerhalb der Eingabesequenz definieren. Jeder Auftrag $i \in \{1, \ldots, n\}$ wird durch zwei Werte charakterisiert. Einerseits ist es die Dauer, die zur Bearbeitung des Auftrages benötigt wird und zum anderen gibt es einen Lastvektor p_i. Dieser beschreibt für jede Maschine $j \in \{1, \ldots, m\}$ die Erhöhung ihrer Last, falls der Auftrag i durch j bearbeitet wird. Die Elemente $p_{i,j}$ dieses Vektors sind nichtnegative Zahlenwerte. Die Last einer Maschine j zu einem Zeitpunkt t ist definiert als die Summe aller Einzellasten $p_{i,j}$ von Aufträgen i, die zum Zeitpunkt t der Maschine j zugeordnet sind.
Die Aufgabe eines Online-Algorithmus besteht darin, die eingehenden Aufträge so den Maschinen zuzuordnen, daß die maximale Last, bestimmt über alle m Maschinen und alle Zeitpunkte t, minimiert wird. Der Online-Algorithmus nimmt dabei die Rolle einer zentralen Steuereinheit wahr, da ihm der komplette Systemzustand bekannt ist.

Zur weiteren Unterteilung der Lastbalancierungsprobleme werden folgende Eigenschaften der Maschinen und Aufträge benutzt:

- **unterbrechbare / nichtunterbrechbare Aufträge:** In dieser Abhandlung wird nur der Fall nichtunterbrechbarer Aufträge behandelt, d. h. nach Zuweisung des Auftrages i an eine Maschine j verbleibt i bei j und wird ausschließlich von j bearbeitet. Im Gegensatz dazu gibt es auch das Modell mit unterbrechbaren Aufträgen. Dort ist eine Neuzuweisung eines Auftrages i an eine andere Ma-

[11] Im englischen als „Scheduling"-Probleme bezeichnet.

schine j' erlaubt, wobei sich die Dauer von i entsprechend der schon erhaltenen Laufzeit auf j verringert. Werden beliebig viele Neuzuordnungen zugelassen, so kann ein Online-Algorithmus nach jedem Eintreffen eines neuen Auftrages alle noch vorhandenen Aufträge den Maschinen neu zuordnen und so eine optimale Lösung erzielen, d. h. 1-competitive werden. Deshalb machen Untersuchungen dieser Modellvariante nur Sinn, wenn die Anzahl der Neuzuordnungen begrenzt wird. Resultate zu diesen Modellen sind in [AAPW94, PW98, Wes00] zu finden.

- **Aufträge bekannter / unbekannter Dauer:** Die Dauer eines Auftrages kann Teil der Eingabe sein, die dem Online-Algorithmus beim Aufdecken des Auftrages mitgeteilt wird. Im anderen Fall bleibt die Dauer eines Auftrages so lange unbekannt, bis dieser vollständig abgearbeitet ist und dem Online-Algorithmus das Ereignis der Ressourcefreigabe mitgeteilt wird. Natürlich ist in der letztgenannten Modellvariante die Informationsbasis des Online-Algorithmus schlechter und deshalb sind keine besseren Resultate als in dem Modell mit bekannter Auftragsdauer möglich.
  Im Falle der bekannten Ausführungsdauer wird weiterhin wie folgt unterschieden:

- **permanente / temporäre Aufträge:** In einem Modell mit temporären Aufträgen besitzt jeder Auftrag i eine endliche Verweildauer im System. Von permanenten Aufträgen wird gesprochen, wenn für alle Aufträge die Dauer auf Unendlich gesetzt wird. Mit der Last eines permanenten Auftrages wird dann häufig seine Laufzeit identifiziert, und das Lastbalancierungsproblem wird benutzt, um eine Zuordnung mit minimaler Gesamtlaufzeit des Systems zu bestimmen.
  Da in einem Modell mit permanenten Aufträgen zusätzliches a priori Wissen über die Struktur der Aufträge vorhanden ist, übertragen sich Algorithmen und deren obere Schranken von dem Modell mit temporären Aufträgen, und Online-Algorithmen können u. U. im Modell mit permanenten Aufträgen einen besseren Competitive Ratio erreichen.

- **Maschinencharakteristik:** Die Eigenschaft der m Maschinen bildet sich in der Struktur der Lastvektoren p_i ab. Alle Maschinen können gleich sein und damit sind die einzelnen Einträge eines Lastvektors auch gleich. Dann wird vom Modell mit *identischen* Maschinen gesprochen.
  Sind alle m Maschinen gleichartig gebaut, werden aber mit unterschiedlichen Geschwindigkeiten v_j ($j \in \{1, \ldots, m\}$) betrieben, so führt das zum Modell mit *ähnlichen* Maschinen. Jeder Auftrag i besitzt dabei ein Arbeitsvolumen w_i und die Einträge im Lastvektor p_i sind durch $p_{i,j} = w_i/v_j$ gegeben.
  Von einem Modell mit *beschränkter Zuordnung* wird gesprochen, wenn ein Auftrag nur auf einer Teilmenge $S \subset \{1, \ldots, m\}$ der Maschinen ausgeführt werden darf. Dann sind die Lastvektoreinträge $p_{i,j}$ gleich für alle $j \in S$ und

unendlich für alle anderen Maschinen:

$$p_{i,j} = \begin{cases} w_i & \text{für } j \in S \\ \infty & \text{für } j \notin S \end{cases}$$

Im allgemeinsten Fall sind die Einträge der Lastvektoren beliebige nichtnegative Zahlenwerte. Dies wird als das Modell mit nicht in Beziehung stehenden Maschinen bezeichnet oder kürzer mit *beliebigen* Maschinen.
Das Modell mit identischen Maschinen ist ein Spezialfall der beiden Modelle mit ähnlichen Maschinen und beschränkter Zuordnung. Diese zwei Modelle sind selbst nicht miteinander vergleichbar, aber der Fall mit beliebigen Maschinen ist eine Verallgemeinerung beider. Mit Hilfe dieser Beobachtungen lassen sich Ergebnisse über untere und obere Schranken zwischen den Maschinenmodellen transferieren.

- Es verbleibt eine noch umfassendere Generalisierung der Modelle einzuführen. Sie ist unter dem Namen *Virtual Circuit Routing* (VCR) bekannt. Dabei ist ein Netzwerk mit Knoten und Verbindungskanten gegeben. Diese Verbindungskanten stellen die Ressourcen dar und entsprechen somit den bisherigen Maschinen. Ein Auftrag besteht aus einem Start- und einem Zielknoten sowie der Dauer und dem Lastvektor. Jedoch müssen im VCR-Problem einem Auftrag mehrere Ressourcen zugewiesen werden, und zwar derart, daß diese Kanten einen Weg zwischen dem Start- und dem Zielknoten bilden. D. h. ein Auftrag kann als Bandweitenanforderung für eine Kommunikation zwischen dem Start- und dem Zielknoten interpretiert werden. Nach solch einer Zuweisung steigt die Last auf allen Kanten des Weges entsprechend dem Lastvektor. Das VCR-Modell soll jedoch nicht vertiefend betrachtet werden. Der interessierte Leser sei auf den Übersichtsartikel [Leo98] und die dort zitierten Literaturstellen verwiesen.

Bevor auf die Untersuchungsresultate der Lastbalancierungsmodelle im Detail eingegangen wird, werden zwei allgemeingültige Techniken, die bei der Lösung von Problemen aus dieser Klasse mehrfach eingesetzt wurden, vorgestellt.

Die Verdoppelungstechnik

Sei ein Online-Algorithmus ALG gegeben, der c-competitive ist und für seine Berechnungen das Wissen über die optimalen Kosten Λ benutzt. Die im folgenden beschriebene Verdoppelungstechnik bietet einen algorithmischen Rahmen, um aus ALG einen 4c-competitiven Online-Algorithmus zu konstruieren. Auf das zusätzliche Wissen von Λ kann also verzichtet werden und der dadurch entstehende Verlust im Competitive Ratio ist nicht größer als der konstante Faktor 4. Dieser neue Online-Algorithmus arbeitet in Phasen. Am Anfang wird ein ausreichend kleiner Wert λ als Annahme für Λ gewählt, und zu Beginn jeder weiteren Phase wird dieser Wert λ verdoppelt. Innerhalb der Phase arbeitet ALG unter der Annahme λ wäre der korrekte Wert für die optimale Lösung und ALG ignoriert die zugewiesenen Aufträge und Entscheidungen aller früheren Phasen. Führen

die von ALG in der aktuellen Phase getroffenen Entscheidungen zu einem Widerspruch in der Annahme $\Lambda \leqq \lambda$, indem ALG eine Lösung mit Kosten größer als $c\lambda$ konstruieren würde, so endet diese Phase. In der letzten Phase wird der wirkliche optimale Wert Λ maximal um den Faktor 2 überschätzt, weshalb die Kosten dieser Phase nicht größer als $2c\Lambda$ sein können. Da sich die Kosten in jeder Phase höchstens verdoppelt haben, so können die Kosten aller früheren Phasen zusammen den Wert $2c\Lambda$ nicht übersteigen. Die Gesamtkosten sind also durch $4c\Lambda$ begrenzt und aus diesem Grund geht bei dieser Konstruktion des Online-Algorithmus höchstens der Faktor 4 gegenüber dem c-competitiven Algorithmus ALG verloren.

Die Exponentialfunktionstechnik

Dabei handelt es sich um ein algorithmisches Prinzip, welches schon mehrfach für $O(\log m)$-competitive Algorithmen eingesetzt wurde. Es kann nur angewandt werden, wenn der maximale Ressourceverbrauch oder ein maximaler Wert für die vorhandenen Ressourcen bekannt ist. Dies kann z. B. über die Annahme eines Wertes der optimalen Lösung (s. o.) geschehen. Bei der Exponentialfunktionstechnik werden die noch vorhandenen Ressourceanteile mit Kosten belegt. Je weniger von einer Ressource übrig ist, desto teurer wird sie. Und diese Kosten steigen mit einer Exponentialfunktion an. Der Online-Algorithmus bestimmt in einer Entscheidungssituation die Kosten für jede Möglichkeit der Zuordnung des Auftrages zu einer Ressource, indem die Ressourcekosten mit dem Ressourceverbrauch (Lastvektor) verknüpft werden. Die billigste Variante wird dann gewählt.

Innerhalb der folgenden Darstellung von Ergebnissen zu Lastbalancierungsproblemen werden nur die Modelle mit permanenten Aufträgen und temporären Aufträgen bekannter Dauer aufgeführt.

Das Modell mit identischen Maschinen und permanenten Aufträgen wurde schon 1966 von R. L. Graham betrachtet [Gra66]. Diese Arbeit stellt die erste bekannte Untersuchung vom Typ der Competitive Analysis dar, obwohl dieser Begriff damals noch unbekannt war, und die Idee einer derartigen Analyse erst ca. 18 Jahre später wieder aufgenommen bzw. neu entwickelt wurde. Graham zeigte in [Gra69], daß ein einfacher Greedy-Algorithmus, der einen Auftrag immer der Maschine mit der minimalen Last zuweist, genau $(2 - 1/m)$-competitive ist.
Für maximal vier Maschinen konnten in [GW93, CVW94] bessere Algorithmen gezeigt werden. Der erste allgemeingültige Online-Algorithmus mit einem Competitive Ratio strikt kleiner als 2 wurde in [BFKV95] vorgestellt. Dabei wird ein Auftrag nicht immer der Maschine mit geringster Last zugeordnet, damit diese für das Auftreten sehr stark belastender Aufträge frei bleibt. Die besten derzeit bekannten Ergebnisse für deterministische Online-Algorithmen stammen von Albers [Alb00] und lauten für dieses Lastbalancierungsproblem:

$$1.852 \leqq \mathcal{R} \leqq 1.923 \ .$$

Werden statt der identischen nun ähnliche Maschinen betrachtet, d. h. in diesem Modell haben die Maschinen verschiedene Geschwindigkeiten, so ist eine andere algorithmische Idee notwendig. Eingebettet in die Verdoppelungstechnik arbeitet der folgende Algorithmus [AAF+97]: Der aktuelle Auftrag wird der Maschine mit minimalem Index zugeordnet, bei der ein Lastwert von 2λ nicht überschritten wird. Die nächste Phase beginnt, falls eine solche Maschine nicht existiert. Dieser Algorithmus ist damit 8-competitive.
In [BCK00] wurde die Verdoppelungstechnik durch eine stark verfeinerte Methode ersetzt und ein Competitive Ratio von $3+\sqrt{8} \approx 5.828$ gezeigt. Am gleichen Ort ist die beste bekannte untere Schranke $\mathcal{R} \geqq 2.438$ für deterministische Online-Algorithmen, die dieses Problem lösen, angegeben.

Für die Modellvariante permanenter Aufträge mit beschränkter Zuordnung ist in [ANR95] ein bis auf Konstanten optimaler Online-Algorithmus vorgeschlagen und analysiert worden. Der Algorithmus weist den aktuellen Auftrag der Maschine aus der Menge aller zulässigen Maschinen zu, deren Belastung minimal ist. Diese einfache Online-Strategie ist $(\lceil \log_2 m \rceil + 1)$-competitive, und als untere Schranke für den Competitive Ratio wurde $\mathcal{R} \geqq \lceil \log_2 m \rceil$ nachgewiesen.

Für das allgemeine Modell von permanenten Aufträgen mit beliebigen nichtnegativen Lastvektoren (beliebige Maschinen) wurde in [AAF+97] ebenfalls ein $O(\log m)$-competitiver deterministischer Online-Algorithmus gezeigt. Er basiert auf der Exponentialfunktionstechnik unter Annahme des Wertes Λ der optimalen Lösung. Diese Entscheidungsfunktion ist dann in die Verdoppelungstechnik eingehüllt. Eine zu dem Ergebnis passende untere Schranke von $\mathcal{R} \in \Omega(\log m)$ wird ebenfalls in [AAF+97] angegeben.

Interessanterweise läßt sich der soeben angedeutete Algorithmus mit minimalen Modifikationen auch für das generalisierte Modell des VCR-Problems einsetzen. Statt die Kosten für die Zuordnung eines Auftrages zu einer Ressource zu bestimmen, wird der Kostenzuwachs bezüglich der Exponentialfunktion über alle Kanten eines zulässigen Weges im Netzwerk aufsummiert. Gewählt wird dann der billigste aller möglichen Wege von dem geforderten Start- zum Zielknoten. In [AAF+97] wird dieser Online-Algorithmus als $O(\log m)$-competitive nachgewiesen. [BL97] zeigt auch für ungerichtete Netzwerke die untere Schranke $\mathcal{R} \in \Omega(\log m)$ für das VCR-Problem.

Für die Lastbalancierungsprobleme mit Aufträgen von endlicher Dauer ist es nicht ausreichend die Diskussion auf die Modelle mit temporären Aufträgen bekannter Ausführungsdauer zu beschränken, da einige der Resultate zu diesen Modellen von der Modellvariante mit unbekannter Auftragsdauer übernommen werden. So ist es bisher noch nicht klar, ob bei identischen und ähnlichen Maschinen das zusätzliche Wissen um die Dauer der Aufträge zu besseren Lösungen führen kann. Im Fall identischer Maschinen wird in [AE97] der Greedy-Algorithmus nach Graham als optimal (mit $\mathcal{R} = 2 - 1/m$) nachgewiesen. Für das Modell mit Maschinen verschiedener Geschwindigkeiten bestimmt [AKP+97] die Schranken $3 - o(1) \leqq \mathcal{R} \leqq 20$, d. h. $\mathcal{R} \in \Theta(1)$.

[AKP+97] behandelt auch die Lastbalancierungsprobleme von Aufträgen mit bekannter Dauer und beschränkter Zuordnung, beliebigen Lastvektoren und des VCR-Problems. Dabei wird der Parameter T, der das Verhältnis der maximalen zur minimalen Dauer eines Auftrages bezeichnet, innerhalb der Analyse benötigt. Die Autorengruppe gibt dann einen Online-Algorithmus an, der $O(\log(mT))$-competitive ist, wobei dem Algorithmus die minimale Dauer a priori bekannt sein muß. Es ist bisher unklar, ob der $\log(T)$-Term im Competitive Ratio für diese Problemklassen wirklich notwendig ist.
Am gleichen Ort wird für das Modell von temporären Aufträgen mit unbekannter Dauer und beschränkter Zuordnung eine Verschärfung der Ergebnisse[12] auf $\mathcal{R} \in \Theta(\sqrt{m})$ erreicht, indem ein $O(\sqrt{m})$-competitiver Algorithmus angegeben wird. Dieses Resultat überträgt sich auch auf die Modellvariante mit bekannter Auftragsdauer und ist für sehr große Werte von T besser.

Die soeben beschriebenen Ergebnisse für den Competitive Ratio der verschiedenen hier abgehandelten Lastbalancierungsmodelle sind in Tabelle 1 zusammengefaßt.

Maschinen-charakteristik	permanente Aufträge	temporäre Aufträge bekannter Dauer
identisch	1.923	$2 - o(1)$
ähnlich	$\Theta(1)$	$\Theta(1)$
beschränkte Zuordnung	$\Theta(\log m)$	$O(\log mT)$, $O(\sqrt{m})$
beliebig; VCR	$\Theta(\log m)$	$O(\log mT)$

Tabelle 1: Zusammenfassung der Competitive Ratios für Lastbalancierung

3.2 Online-Planungsprobleme

In der Klasse der Planungsprobleme (Scheduling-Probleme) geht es ebenfalls darum Aufträge mit temporären Ressourcebedarf den vorhandenen Ressourcen sowie Zeitpunkten bzw. -intervallen zuzuordnen. Es ist also ein Zeitplan aufzustellen. Allerdings gibt es sehr viele Modelle, um die Ressourcen und die Aufträge sowie weitere Nebenbedingungen und die Zielfunktion zu charakterisieren. Das führt zu einer unüberschaubaren Variantenvielfalt von Planungsproblemen. Um in dieser Menge etwas Struktur zu verankern wurde schon frühzeitig ein Klassifikationsschema entwickelt [CMM67]. Es wird auch weiterhin benutzt (siehe [GLLRK79, Bru95, TGS94, TSS94]) und erweitert (z. B. in [BLRK83, Tim93, BESW94, BJK97]). Da dieses Schema mehrere orthogonal zueinander stehende Dimensionen besitzt, führt es jedoch auch zu Studien von theoretischen Modellen,

[12] Die untere Schranke $\mathcal{R} \in \Omega(\sqrt{m})$ stammt aus [ABK94]. Dort konnte jedoch nur ein $O(m^{2/3})$-competitiver Algorithmus gezeigt werden.

für die keine Motivation aus der Praxis vorliegen. Diese Tatsache stellt einen Kritikpunkt dar. Auf der anderen Seite kann dieses Klassifikationsschema nur einen Teil der Planungsprobleme abbilden und jedes Jahr werden neue Modelle aus praktischen Fragestellungen heraus entwickelt und untersucht. Eine Abhandlung der bisherigen Studien in der Literatur würde eine mehrbändige Enzyklopädie füllen und damit den Rahmen dieser Arbeit deutlich überschreiten. Deshalb werden in diesem Teilkapitel ausschließlich die Planungsprobleme vom Online-Typ vorgestellt, deren Auftrags- und Maschinencharakteristik schon im Kapitel 3.1 eingeführt wurden.
Bei der Untersuchung von Offline-Planungsproblemen wird als erstes die Frage nach der Zugehörigkeit zu den Komplexitätsklassen $\mathcal{P}$ oder $\mathcal{NP}$-schwer beantwortet. Im Falle der Klasse $\mathcal{P}$ wird gleichzeitig nach einem effizienten Algorithmus gesucht. Für Modelle, die als $\mathcal{NP}$-schwer eingestuft sind, besteht die Herausforderung im Auffinden von Polynomialzeit-Approximationsschemata (PTAS, FPTAS)[13], von Approximationsalgorithmen und Nichtapproximierbarkeitsresultaten sowie unter praktischen Aspekten in der Konstruktion von gut funktionierenden Heuristiken.

Es soll nochmals betont werden, daß in der Online-Variante von Planungsproblemen die Aufträge a priori unbekannt sind, jedoch nach ihrer Aufdeckung so lange gespeichert werden und eine Zuordnungsentscheidung hinausgezögert wird, bis die benötigten Ressourcen frei sind. Dieser Vorteil gegenüber dem Lastbalancierungsproblem in gleicher Umgebung führt zu besseren Competitive Ratios. Natürlich lassen sich weiterhin die Algorithmen und oberen Schranken der entsprechenden Lastbalancierungsprobleme übertragen.

Zuerst wird wiederum ein Modell mit m identischen Maschinen und n unabhängigen, nicht unterbrechbaren Aufträgen mit einem Bedarf von einer Maschine und vorgegebener Laufzeit betrachtet. Eine Maschine kann zu einem Zeitpunkt höchstens einen Auftrag bearbeiten. In [CV97] wird für die Zielfunktion der Minimierung des Fertigstellungszeitpunktes aller Aufträge der Online-Algorithmus LPT („longest processing time“) angegeben und als 1.5-competitive nachgewiesen. Sobald eine Maschine frei wird, wählt diese Strategie den längsten derzeit bekannten und noch nicht bearbeiteten Auftrag aus und startet ihn. Eine untere Schranke von $\mathcal{R} \geqq 1.3473$ wird am gleichen Ort gezeigt.
Für Modelle mit ähnlichen und beliebigen Maschinen ergeben sich Resultate aus einer sehr allgemeinen Aussage, die in [SWW95] bewiesen wurde. Dort wird eine Transformationstechnik angegeben, die unter Verlust eines Gütefaktors von 2 aus einem Planungsalgorithmus, der alle Aufträge zu Beginn kennt, einen Online-Algorithmus konstruiert, dem die Aufträge erst zu ihrem frühesten Startzeitpunkt bekannt gemacht werden müssen. Für die hier betrachteten Modelle, bei denen die Auftragsdauer mit dem Erhalt des Auftrages bekannt ist, können in diese Technik Offline-Algorithmen eingesetzt werden.
Das soeben vorgestellte Planungsproblem ist in allen drei Varianten von Ma-

[13] polynomial time approximation scheme; fully polynomial time approximation scheme

schinenmodellen $\mathcal{NP}$-schwer [GJ79]. Für identische und ähnliche Maschinen sind in [HS87, HS88] PTAS gezeigt worden. Mit beliebigen Maschinen ist eine 1.5-Approximation $\mathcal{NP}$-schwer und ein 2-approximativer Algorithmus wurde entwickelt [LST90]. Aus diesen Studien ergibt sich unter Anwendung der Transformationstechnik aus [SWW95] die Existenz von Online-Algorithmen, die 2-competitive sind. Außerdem folgern sich auch Online-Algorithmen mit polynomieller Laufzeit, welche für das Modell mit Maschinen verschiedener Geschwindigkeit $(2+\varepsilon)$-competitive und für das Modell mit beliebigen Maschinen 4-competitive sind.

Werden Unterbrechungen und Neuzuordnungen von Aufträgen zugelassen, so wurden 1-competitive Online-Algorithmen für identische Maschinen und für ähnliche Maschinen mit einer Beschränkung in der Geschwindigkeitsdifferenz gezeigt [HL92, Ves97]. Dabei sind jedoch bis zu $n \cdot m$ Neuzuordnungen notwendig.

Eine weitere populäre Zielfunktion, die es in Planungsproblemen zu minimieren gilt, ist der durchschnittliche Fertigstellungszeitpunkt der Aufträge. Dabei ist die Summe der Endzeitpunkte der Auftragsbearbeitungen klein zu halten. Besitzen die Aufträge zusätzliche Prioritäten, so werden die Fertigstellungszeitpunkte innerhalb der Summenbildung mit einem Gewichtungsfaktor versehen. Für beide Versionen dieser Zielfunktion wurde in [CPS+96] eine Transformationstechnik angegeben, die eine ähnliche Leistung wie die in [SWW95] erbringt. Diese Technik stellt einen algorithmischen Rahmen namens Greedy-Interval zur Verfügung. Er arbeitet in Phasen, deren Zeitdauer sich immer verdoppeln. Zu Beginn einer Phase i zum Zeitpunkt $t = 2^i$ werden alle schon bekannten und noch nicht bearbeiteten Aufträge betrachtet. Für diese wird — mittels eines an dieser Stelle einzusetzenden, z. B. optimalen Offline-Algorithmus — ein Plan bestimmt, in dem die Aufträge bearbeitet werden, deren Gewichte in Summe maximal sind und die bis zum Ende der Phase bei $t = 2^{i+1}$ beendet werden können.
Die Strategie Greedy-Interval in Verbindung mit optimalen Offline-Planungsalgorithmen ergibt dann Online-Algorithmen für alle in diesem Abschnitt betrachteten Modelle, die 4-competitive bezüglich der gewichteten durchschnittlichen Fertigstellungszeit und *gleichzeitig* 3-competitive bezüglich der gesamten Planlänge sind. Weitere Arbeiten zu dieser Zielfunktion mit geringeren Competitive Ratios für Modelle mit unterbrechbaren Aufträgen, einer Einzelmaschine bzw. Online-Algorithmen mit polynomieller Laufzeit sind in [HSW96, HSSW97, PSW95, Goe97] zu finden.

Soll hingegen die durchschnittliche Verweildauer der Aufträge im System zu minimiert werden, erhöhen sich die Schwierigkeiten gewaltig. Die Verweildauer eines Auftrages ist definiert als sein Fertigstellungszeitpunkt abzüglich seines frühesten Startzeitpunktes, also im Online-Fall des Eingabezeitpunktes dieses Auftrages. Wird das Unterbrechen oder das Neustarten von Aufträgen verboten, so ist schon für ein System mit nur einer Einzelmaschine eine untere Schranke von $\mathcal{R} \geqq n$ für den Competitive Ratio beweisbar. Dazu wird ein Auftrag mit der Laufzeit 1 eingegeben und direkt danach für den selben Startzeitpunkt $n-1$ Aufträge mit

Laufzeit 0. Allerdings ist auch schon das Offline-Problem sehr komplex, denn eine $n^{1/2-\varepsilon}$-Approximation ist für jedes $\varepsilon > 0$ $\mathcal{NP}$-schwer [KTW99].
Werden Unterbrechungen bei der Bearbeitung der Aufträge zugelassen, so ist in [LR97] $\mathcal{R} \in \Theta(\log(n/m))$ gezeigt worden. Auch bei diesem Ergebnis geht die Anzahl der Aufträge als Funktion in den Competitive Ratio ein. Nur wenn das Verhältnis von maximaler zu minimaler Auftragsdauer durch eine Konstante T begrenzt ist, gilt $\mathcal{R} \in \Theta(\log T)$, wie ebenfalls in [LR97] bewiesen.

3.3 Planungsprobleme in Realzeitsystemen

Im Unterschied zu den oben diskutierten Modellen haben die Aufträge nun zusätzlich eine Frist, d. h. einen Zeitpunkt bis zu dem sie ausgeführt sein müssen. Diese Eigenschaft wird durch den Namen *Realzeitsystem* oder durch Hinzufügen des Begriffes *Deadline*[14] zum Ausdruck gebracht. In der Offline-Variante dieser Problemklasse ändert sich die grundlegende Fragestellung. Man ist nun primär nicht mehr daran interessiert eine Funktion in den Fertigstellungszeitpunkten der Aufträge zu optimieren, sondern fragt sich, ob ein Zeitplan existiert, in dem alle Aufträge innerhalb ihrer Frist bearbeitet werden können. Einen Zeitplan, der diese Forderung erfüllt, wird *zulässig* genannt. Vom komplexitätstheoretischen Betrachtungsstandpunkt liegt also ein Entscheidungsproblem vor.

Jiří Sgall äußerte in seinem Aufsatz [Sga98] die Meinung, daß Planungsprobleme mit Fristen im Online-Szenario wenig sinnvoll sind. Er begründet dies damit, daß zur Lösung des Entscheidungsproblems der Existenz eines zulässigen Planes im Online-Fall 1-competitive Algorithmen notwendig sind, d. h. ein Online-Algorithmus muß die gleiche Lösungsqualität garantieren, wie ein optimaler Offline-Algorithmus. Für einige sehr triviale Planungsprobleme ist das möglich. Erhöht sich jedoch die Komplexität der Modelle marginal, so lassen sich schnell untere Schranken größer als 1 für den Competitive Ratio zeigen, wie weiter unten bei der Besprechung der Resultate aus [HL92] zu sehen ist.
Allerdings stimmt der Autor mit dieser negativen Auffassung Sgall's nicht überein, wie auch einige andere Autoren, deren Arbeiten in diesem Kapitel diskutiert werden. Wird in einem Realzeitsystem die Frage nach der Anzahl der Aufträge, die fristgerecht bearbeitet wurden oder nach der Gewichtssumme dieser Auftragsmenge gestellt, so ergibt sich wiederum ein Optimierungsproblem. Dann ist auch die Entwicklung von Online-Algorithmen mit Competitive Ratios größer als 1 interessant. Diese veränderte Zielfunktion impliziert allerdings auch, daß innerhalb eines derartigen Systems Aufträge, für die ihre Fristen nicht eingehalten werden, nicht bearbeitet werden müssen oder sie werden sofort nach ihrem Erscheinen abgelehnt.
Solche Modelle stellen nicht nur eine Erweiterung für theoretische Untersuchungen dar, sondern sie sind auch durch Probleme aus der Praxis motiviert. So macht

[14] In diesem Zusammenhang am besten übersetzt mit Ablauffrist oder spätestem Fertigstellungszeitpunkt.

es bei der Realzeitauslieferung von Daten für einen Datenstrom eines kontinuierlichen Mediums (z. B. Video, Audio) keinen Sinn ein Datenpaket, welches nicht fristgerecht ausgeliefert werden konnte, später noch nachzusenden. Das Fehlen dieses Paketes führt zwar zu einer Störung bei der Wiedergabe, ein kurzzeitiges Anhalten der Medienwiedergabe wird jedoch subjektiv als viel unangenehmer oder gar als unzumutbar empfunden.
Auch im Bereich der eingebetteten Kontroll- und Steuersysteme ist ein entsprechendes Szenario vorstellbar. Die typischen Aufträge zur Auswertung und Aufbereitung von Sensordaten werden zyklisch wiederholt. Im Falle eines Fehlverhaltens oder einer Ausnahmesituation, die zu einer Systemüberlastung führt, werden nur noch wichtige, d. h. hoch priorisierte Aufträge bearbeitet. Aufträge von untergeordneter Bedeutung, deren Nichtausführung die Systemintegrität nicht gefährdet, wie z. B. Messungen von Parametern zu Informations- oder Protokollzwecken, werden später nicht nachgeholt. Zum einen können Messungen nicht auf den Systemzustand der Vergangenheit zurückgreifen, und zum anderen stabilisiert sich das Steuersystem schneller durch Weglassen dieser Last.
Planungsprobleme mit Fristen können in bestimmten abstrakten Modellvarianten (z. B. diskretes Zeitmodell, Aufträge mit einheitlichen Ausführungszeiten, etc.) als ein Zuordnungsproblem der Aufträge zu a priori abgegrenzten äquidistanten Zeitabschnitten der Ressourcen modelliert werden. Die daraus resultierende Struktur von bipartiten Graphen und die Transformation des Planungs- zu einem Matchingproblem wird im Kapitel 3.4 erörtert. Es folgt nun die Diskussion von Arbeiten zu Planungsproblemen mit Intervallaufträgen und Modellen, die als Realzeitplanungsprobleme bezeichnet werden.

3.3.1 Intervallaufträge

Den Modellen von Planungsproblemen mit Intervallaufträgen ist folgende spezielle Charakteristik der Aufträge gemeinsam: Ein Auftrag besitzt einen Startzeitpunkt, der im Online-Szenario mit dem Zeitpunkt seiner Eingabe zusammenfällt, und eine Laufzeit. Damit der Auftrag erfolgreich beendet wird, muß er zu seinem Startzeitpunkt einer freien Ressource zugeordnet werden und solange unterbrechungsfrei bearbeitet werden, bis seine Laufzeit beendet ist. Dieser Zeitpunkt stellt gleichzeitig den spätesten akzeptablen Fertigstellungszeitpunkt des Auftrages dar. Ein Auftrag spezifiziert somit ein unverrückbares Zeitintervall (vom Startzeitpunkt bis zum Startzeitpunkt plus der Laufzeit) für eine Ressourceanforderung.

Faigle und Nawijn untersuchen in [FN95] ein Modell mit m identischen Maschinen. Ein Auftrag wird durch ein Zeitintervall spezifiziert und ist abbrechbar, d. h. die Bearbeitung des Auftrages kann zu seinem Eingabe- bzw. Startzeitpunkt begonnen werden, darf aber später zugunsten eines anderen Auftrages abgebrochen werden. Auch in solch einem Fall ist der frühere Auftrag nicht erfolgreich beendet worden. Das Ziel ist die Online-Bestimmung eines Zeitplanes, so daß möglichst viele Aufträge erfolgreich sind oder formaler ausgedrückt, die Kardinalität der

Menge erfolgreicher Aufträge ist zu maximieren. Wird aus den Intervallen der Eingabe ein Intervallgraphen[15] gebildet, so entspricht diese Aufgabenstellung der Färbung eines maximalen Teilgraphen mit m verschiedenen Farben.
In diesem Modell ist der folgende Greedy-Algorithmus 1-competitive: Beim Eintreffen eines Auftrages j wird dieser einer freien Maschine zugeordnet, falls eine solche Ressource vorhanden ist. Ansonsten wird der Auftrag j' bestimmt, der bearbeitet wird und die längste verbleibende Restlaufzeit aufweist. Ist die Laufzeit von j kleiner als die noch benötigte Restlaufzeit von j', so wird j' abgebrochen und j wird dieser Maschine zugewiesen.

In [Woe94] analysiert Woeginger für eine Einzelressource das gleiche Modell von Aufträgen, die jedoch um Gewichtswerte ergänzt sind. Es ist nun die Gewichtssumme aller erfolgreich bearbeiteten Aufträge zu maximieren. Dürfen den Aufträgen beliebige Gewichte zugeordnet werden, so können keine Online-Algorithmen mit begrenztem Competitive Ratio existieren. Deshalb werden in [Woe94] die Gewichte als Funktion der Intervallänge definiert. Für den Fall, daß diese Funktion konvex oder monoton fallend ist, werden Online-Algorithmen vorgestellt, die 4-competitive sind. Diese sind optimal, da auch als allgemeingültige untere Schranke $\mathcal{R} \geqq 4$ gilt. Für eine Modellvariante mit Aufträgen gleicher Intervallänge (Einheitsintervalle) und beliebiger Gewichtsfunktion wird ein weiterer Online-Algorithmus als ebenfalls 4-competitive nachgewiesen.

Ein etwas anderes Modell wird in [LT94] betrachtet. Auch dabei existiert nur eine Ressource, aber das Gewicht der Aufträge ist uniform, d. h. der Gewichtswert eines Auftrages ist identisch mit seiner Intervallänge. Die Maximierung der Gewichtssumme entspricht dann dem Ziel der maximalen Auslastung der Ressource. Im Unterschied zu den beiden vorigen Modellen darf ein einmal begonnener Auftrag nicht abgebrochen werden. Lipton und Tomkins untersuchen für dieses Problem randomisierte Online-Algorithmen, und analysieren sie gegen den blinden Gegenspieler. Für nur zwei Intervalltypen der Länge 1 und $k \gg 1$ wird ein optimaler Online-Algorithmus mit $\overline{\mathcal{R}}_{\mathsf{BL}} = 2$ gezeigt. Der Parameter T, der das Verhältnis der längsten zur kürzesten auftretenden Intervallänge beschreibt, wird in der Analyse der Modellvariante mit beliebigen Intervallängen benötigt. Damit wird eine untere Schranke von $\overline{\mathcal{R}}_{\mathsf{BL}} \in \omega(\log T)$ bewiesen und ein randomisierter Online-Algorithmus mit $\overline{\mathcal{R}}_{\mathsf{BL}} \in O((\log T)^{1+\varepsilon})$ vorgestellt.

3.3.2 Realzeitsysteme

Im Gegensatz zu der Modellvariante des vorigen Teilkapitels steht in den nun zu behandelnden Planungsproblemen der Startzeitpunkt eines erfolgreich bearbeiteten Auftrages nicht von vornherein fest. Ein Modell, welches aus m identischen Maschinen besteht, und dessen Aufträge kostenfrei in ihrer Bearbeitung unterbrochen werden können, wird in [HL92] untersucht. Ab dem Auftreten eines

[15] Intervallgraphen gehören zur Klasse der Schnittgraphen (intersection graphs): Zwei Konten sind durch eine Kante verbunden, wenn sich ihre korrespondierenden Intervalle überlappen.

Auftrages darf dieser von einer Maschine ausgeführt werden. Seine Bearbeitungszeit ist ebenfalls sofort bekannt. Wird eine gemeinsame Frist angegeben, innerhalb der alle Aufträge der Eingabesequenz fertiggestellt werden müssen, so gibt es einen Online-Algorithmus, der einen zulässigen Zeitplan entwirft, falls es für die Eingabe einen solchen gibt. Er unterbricht alle in Bearbeitung befindlichen Aufträge beim Eintreffen eines neuen Auftrages und bestimmt einen neuen, optimierten Zeitplan, der bis zur nächsten Eingabe realisiert wird. Dabei basiert dieser Online-Algorithmus auf McNaughton's Regel [McN59]. Wird hingegen für jeden Auftrag eine individuelle Bearbeitungsfrist angegeben, so zeigt [HL92] die Nichtexistenz von derartig optimalen Online-Algorithmen.

Einen wichtigen Beitrag zu Online-Realzeitproblemen hat die Autorengruppe der Artikel [BKM+91a, WM91, BKM+92, KS92, KS94] geleistet. In dem von ihnen studierten Modell gibt es m identische Maschinen ($m \in \mathbb{N}$) mit den Spezialfällen $m = 1$ (Uniprozessor) und $m = 2$. Jeder Auftrag j wird durch die folgenden Parameter charakterisiert:

- t_j der früheste Startzeitpunkt, der identisch mit dem Eingabezeitpunkt im Online-Szenario ist,
- p_j die Laufzeit,
- d_j der späteste Fertigstellungszeitpunkt (Deadline) und
- v_j der Wert des Auftrages j.

Die Aufträge dürfen kostenfrei unterbrochen oder abgelehnt werden. Das Ziel eines Algorithmus besteht in der Maximierung der Wertesumme $\sum_{j \in S} v_j$ aller erfolgreich bearbeiteten Aufträge S. Eine sekundäre Eigenschaft von Aufträgen ist ihre Wertdichte v_j/p_j. Für die Analysen und Resultate ist der Parameter k von Bedeutung, der als das Verhältnis der maximalen zur minimalen Wertdichte aller Aufträge der Eingabesequenz definiert ist.
Die Vorarbeiten [BKM+91a] und [WM91] kumulieren in dem Artikel [BKM+92] und zeigen für ein Uniprozessorsystem ($m = 1$)

- eine untere Schranke von $\mathcal{R} \geqq (1 + \sqrt{k})^2$,
- einen Online-Algorithmus für uniforme Aufträge ($k = 1$), der 4-competitive ist und damit die bestmögliche Leistungsgarantie gibt

sowie für ein Zweiprozessorsystem ($m = 2$) und uniforme Aufträge ($k = 1$) mit Intervallcharakter ($p_j = d_j - t_j$) einen Algorithmus mit einem Competitive Ratio von 2. [KS94] berichtet über ein identisches Resultat für Aufträge ohne die Intervalleinschränkung in den Modellvarianten mit und ohne Auftragsmigration[16] zwischen den Maschinen.

[16] D. h. ein Auftrag darf kostenfrei unterbrochen und zu einem späteren Zeitpunkt mit einer anderen Maschine weiter bearbeitet werden.

[KS92] ergänzt die Resultate zum Uniprozessor um den Online-Algorithmus $\mathsf{D}^{\mathsf{over}}$ mit $\mathcal{R}(\mathsf{D}^{\mathsf{over}}) = (1+\sqrt{k})^2$, der eine weitere interessante Eigenschaft aufweist. Solange das System nicht überlastet ist, d. h. ein zulässiger Zeitplan für die Eingabe existiert, wird von $\mathsf{D}^{\mathsf{over}}$ ein derartiger Zeitplan aufgestellt.
Das Modell mit beliebiger Maschinenanzahl $m \in \mathbb{N}$ wird in [KS94] untersucht. Die detailierten Ergebnisse für den Competitive Ratio hängen von m ab und können nur mit sehr komplizierten Formeln ausgedrückt werden. Diese vereinfachen sich im Grenzwert für $m \to \infty$ bei $k > 1$ zu den Aussagen

- einer unteren Schranke für den Competitive Ratio von $k\ln(k)/(k-1)$, die auch gilt, wenn Aufträge zwischen Maschinen migriert werden dürfen und
- einem Online-Algorithmus namens MOCA, der ohne Migration auskommt und $(2\ln(k)+3)$-competitive ist.

Über eine weitere Untersuchung dieses Realzeitplanungsproblems für eine Maschine ($m = 1$) wird in [CSZ95] berichtet. Dabei wird die Einschränkung getroffen, daß $\forall\, j:\ d_j = t_j + 2p_j$ gilt, also das Zeitfenster, in dem ein Auftrag j bearbeitet werden kann, exakt doppelt so lang wie seine Laufzeit ist. Für $k = 1$ wird ein einfacher 2-competitiver Online-Algorithmus angegeben. Er verzichtet auf Unterbrechungen von Aufträgen und weist der freigewordenen Ressource sofort einen Auftrag zu, der noch erfolgreich ausgeführt werden kann. Es werden in [CSZ95] weitere Online-Algorithmen vorgeschlagen, die Unterbrechungen von Aufträgen vornehmen und für allgemeine $k \geqq 1$ arbeiten. Die Beweise für untere Schranken im Competitive Ratio von k bzw. $\log_2 k$ werden lediglich durch Simulationsergebnisse zur Beurteilung der erreichbaren Lösungsqualität ergänzt.

Ein Planungsproblem mit einer aufwendigeren Bewertungsfunktion und einem abweichenden Typ von Aufträgen wird in [BLMS+00] vorgestellt und analysiert. Das Modell besteht wiederum aus m identischen Maschinen, aber die Aufträge dürfen nicht unterbrochen werden. Sie besitzen eine Laufzeit p_j und einen Strafwert $\bar{v}_j$, jedoch keine Frist. Sofort nach dem Eintreffen eines Auftrages muß entschieden werden, wie er im Zeitplan einzuordnen ist oder ob seine Bearbeitung abgelehnt wird. Die Zielfunktion besteht aus der Minimierung des letzten Fertigstellungszeitpunktes der bearbeiteten Aufträge plus der Summe der Strafwerte aller abgelehnten Aufträge. Als untere Schranke für den Competitive Ratio wird in [BLMS+00] der Wert $1+\phi \approx 2.618$ bestimmt. Dabei bezeichnet ϕ den Goldenen Schnitt $(1+\sqrt{5})/2$. Der passende Online-Algorithmus, der diese Lösungsqualität garantieren kann, heißt Reject-Total-Penalty, $\mathsf{RTP}(\alpha)$. Seine Arbeitsweise wird im folgenden beschrieben.
Sei J die Menge aller Aufträge der Eingabesequenz. Dann wird mit der Menge $B := \{j \in J \mid \bar{v}_j \leqq p_j/m\}$ die Teilmenge von Aufträgen definiert, deren Strafwerte kleiner sind als ihre Laufzeitkosten in einem ideal dichten Zeitplan. Deshalb werden Aufträge der Menge B prinzipiell abgewiesen. Ansonsten wird

ein Auftrag j abgelehnt, wenn

$$\sum_{\substack{i \in J \setminus B \\ i \text{ abgelehnt}}} \bar{v}_i \; + \; \bar{v}_j \;\; \leqq \;\; \alpha \cdot p_j$$

gilt, d. h. wenn die Summe der Strafwerte aller bisher zurückgewiesener Aufträge aus $J \setminus B$ und dem Strafwert des aktuellen Auftrages j geringer als α-mal die Laufzeit von j ist. Ein akzeptierter Auftrag wird der Maschine mit geringster Belastung zugeordnet. Dieser Teil entspricht Graham's List-Scheduling-Algorithmus [Gra66].
Die Analyse in [BLMS$^+$00] setzt $\alpha = \phi - 1$ und erhält damit das Resultat $\mathcal{R}(\mathsf{RTP}(\phi - 1)) = 1 + \phi$. Interessanterweise ist innerhalb dieses Modells Graham's List-Scheduling-Algorithmus optimal, während er dies in einem Modell ohne Abweisung von Aufträgen nicht ist (siehe [Alb00]).
Für Modellvarianten mit fixierter Maschinenzahl m sind in [BLMS$^+$00] auch bessere Resultate angegeben. So wird ein optimaler Online-Algorithmus für den Fall $m = 2$ gezeigt, der ϕ-competitive ist. Zusätzlich wird für das $\mathcal{NP}$-schwere Offline-Problem ein vollständiges Polynomialzeit-Approximationsschema (FPTAS) und ein $(2 - 1/m)$-approximativer Algorithmus mit $O(n \log n)$ Laufzeit angegeben.

Die letzte in diesem Teilkapitel zu behandelnde Arbeit [BNGNS99] betrachtet ein ähnliches Modell für ein Realzeitsystem, die Algorithmen sind aber nicht primär für das Online-Szenario konstruiert. Trotzdem soll diese Arbeit erwähnt werden, da sie ein weiteres, eindrucksvolles Beispiel für den neuen Typ von Zielfunktionen ist, der über das Entscheidungsproblem der Existenz von zulässigen Zeitplänen hinausgeht.
Das System besteht aus m identischen oder beliebigen Maschinen und jeder Auftrag j ist durch vier Parameter charakterisiert: dem frühesten Startzeitpunkt, dem spätesten Fertigstellungszeitpunkt, dem Vektor der Bearbeitungszeiten für die verschiedenen Maschinen und dem Wert des Auftrages. Die Aufträge dürfen in ihrer Bearbeitung nicht unterbrochen werden. Das Ziel besteht in der Maximierung der Summe aller Werte von erfolgreich bearbeiteten Aufträgen.
Zuerst wird ein Greedy-Algorithmus für die Modellvariante mit identischen Auftragswerten vorgestellt. Dabei ist nur die Anzahl der erfolgreichen Aufträge zu maximieren. Für beliebige Maschinen im System ist dieser Greedy-Algorithmus 2-approximativ. Bei identischen Maschinen wird ein Approximationsfaktor von $(1 + 1/m)^m/((1 + 1/m)^m - 1)$ gezeigt. Für einen Uniprozessor ($m = 1$) bedeutet das einen Wert von 2, der mit wachsender Maschinenzahl auf den Grenzwert ($m \to \infty$) von $e/(e - 1) \approx 1.582$ fällt.
In der Modellvariante mit beliebigen Auftragswerten werden zwei Fälle unterschieden. Sind alle Parameter, die die Aufträge beschreiben, ganze Zahlen, und ist die Länge der Eingabe durch ein Polynom in n, der Auftragsanzahl, beschränkt, so wird vom integralen Fall gesprochen. Im allgemeinen Fall gelten diese Einschränkungen nicht. Ein Algorithmus, der auf der Lösung eines linearen Programmes basiert, ist 3-approximativ im integralen Fall und 4-approximativ bei

allgemeiner Eingabe. Für Uniprozessorsysteme verringern sich diese Werte auf 2 bzw. 3 .
Der Algorithmus für identische Maschinen ($m \geqq 2$) und beliebigen Auftragswerten ist komplizierter. Er benutzt ebenfalls den LP-Ansatz, der für jede Maschine wiederholt angewandt wird. Für den integralen Fall ist dieser Algorithmus ebenfalls — wie der Greedy-Algorithmus — $((1 + 1/m)^m/((1 + 1/m)^m - 1))$-approximativ. Bei allgemeiner Eingabe verschlechtert sich der Approximationsfaktor auf $(1 + 1/2m)^m/((1 + 1/2m)^m - 1)$. Für den hypothetischen, da ausgeschlossenen Fall von $m = 1$ ist der letzte Term 3, und er fällt mit wachsender Maschinenzahl bis zum Grenzwert für $m \to \infty$ von $\sqrt{e}/(\sqrt{e} - 1) \approx 2.5415$.
Baruch Schieber äußerte in einem persönlichen Gespräch[17] mit dem Autor sein Erstaunen, daß dieser Zielfunktionstyp bisher im Offline-Fall nicht eingehend betrachtet wurde. Aber an der Relevanz dieser Betrachtungsweise habe er und seine Mitautoren keine Zweifel.

3.4 Online Matching-Probleme

Der Begriff *Matching* soll innerhalb dieser Arbeit in seiner englischen Form beibehalten werden. Die passende Übersetzung in der Informatik wäre Paarbildung. Einem Problem dieser Klasse liegt immer ein Graph $G = (V, E)$ zugrunde. Er besteht aus einer Menge von Knoten V und einer Kantenmenge E, wobei eine Kante zwei verschiedene Knoten verbindet, d. h. $E \subset V \times V$. Bei Matching-Problemen sollen nun disjunkte Paare von Knoten gebildet werden, die jeweils durch eine Kante verbunden sind. Formal heißt das, ein Matching M ist eine Teilmenge der Kantenmenge E ($M \subset E$), in der jeder Knoten höchstens an einer Kante aus M beteiligt ist.
Bestimmte eingeschränkte Modelle von Planungsproblemen lassen sich durch solche Graphen modellieren, und die Berechnung eines Matchings — einer Paarung zwischen Aufträgen und Zeitabschnitten von Ressourcen — stellt dann eine Lösung dar (siehe z. B. auch [ABK94, PW98]). Da die innerhalb dieser Arbeit untersuchten Planungsprobleme in diese Klasse fallen, werden in diesem Kapitel die Resultate der Studien zu Online-Varianten derartiger Matching-Probleme vorgestellt.

3.4.1 Komplexität der Offline-Version

Das Problem der Bestimmung von Matchings wurde intensiv untersucht. Schon aus einem Theorem von C. Berge [Ber57] ergibt sich ein Polynomialzeit-Algorithmus für das Matching maximaler Kardinalität in bipartiten Graphen. Der Aufsatz [Edm65b] von J. Edmonds stellt das erste tiefgreifende Verständnis für die Besonderheiten von Matching-Problemen in allgemeinen Graphen dar.
Werden die Kanten des Graphen durch Gewichte ergänzt, so ist die Bestimmung von Matchings mit maximaler Gewichtssumme oder von Matchings maximaler

[17] Im Rahmen des Dagstuhlseminars „Competitive Algorithms“ vom 20. bis 25. Juni 1999.

Kardinalität und minimaler Gewichtssumme interessant. Diese Probleme sind in der Literatur der mathematischen Optimierung unter dem Begriff *Assignment* bekannt.
Bis heute wurden permanent algorithmische Verbesserungen und Untersuchungen zu effizienteren Implementierungen publiziert. Tabelle 2 gibt einen Überblick über die ersten Ergebnisse und die aktuell schnellsten Offline-Algorithmen für die verschiedenen Graphvarianten des Matching-Problems. Die frühen Arbeiten werden hier zitiert, weil darin Techniken entwickelt wurden, auf die Beweise innerhalb dieser Arbeit zurückgreifen. Auf einen historischen Abriß, der die vielen Schritte zur Verringerung der algorithmischen Komplexität aufzeigt, wird in dieser Tabelle verzichtet. Er ist z. B. in [LP86] gut aufbereitet.
Anmerkungen zu den in Tabelle 2 zitierten Quellen: Für ungewichtete, bipartite Graphen ist der Algorithmus nach [ABMP91] nur dann besser als $O(\sqrt{n}\,m)$, falls der Graph dicht ist (z. B. $m \in \Theta(n^2)$). Dem Artikel [MV80] folgten [PL88, Vaz94] mit ausführlichen Erläuterungen, effizienteren Implementierungen und der Beseitigung kleinerer Fehler beim Matchingalgorithmus für allgemeine Graphen. Fredman und Tarjan führen in [FT84] Fibonacci-Heaps ein und erhalten mit dieser Datenstruktur das beste Resultat für gewichtete bipartite Graphen. Der Arbeit von [GT89] folgt [OA92] mit einer effizienteren Implementierung bei gleicher theoretischer Komplexität. Alle Algorithmen in [GT89, OA92, GT91] basieren auf der Skalierungstechnik, wobei ganzzahlige Kantengewichte angenommen werden und N das größte vorkommende Gewicht beschreibt.

Graph:	bipartit		allgemein	
ungewichtet	[Ber57]	$O(n\,m)$	[Edm65b]	$O(n^4)$
	[HK73]	$O(\sqrt{n}\,m)$	[MV80]	$O(\sqrt{n}\,m)$
	[ABMP91]	$O(n^{3/2}\sqrt{m/\log n})$		
gewichtet	[Kuh55]	$O(n^2 m)$	[Edm65a]	$O(n^4)$
	[FT84]	$O(n\,m + n^2 \log n)$	[Gab90]	$O(n\,m + n^2 \log n)$
	[GT89]	$O(\sqrt{n}\,m \log(nN))$	[GT91]	$O(\sqrt{n\,\alpha(m,n)\log n}\,m \log(nN))$

Tabelle 2: Erste Polynomialzeitalgorithmen und beste Laufzeitresultate für Grundvarianten des Matching-Problems. Es gilt $n := |V|$, $m := |E|$, N ist die Größenordnung des maximalen Gewichtes bei ganzzahligen Kantengewichten, und $\alpha(m,n)$ symbolisiert die inverse Ackermann-Funktion.

3.4.2 Online Bipartite Matching

In diesem Fall ist der zugrundeliegende Graph bipartit, d. h. $G = (U \dot\cup V, E)$, wobei U und V disjunkte Mengen von Knoten sind und für die Kantenmenge

gilt $E \subset U \times V$. Weiterhin ist auf V eine totale Ordnung $\prec$ gegeben, die das Online-Zeitmodell definiert. In jedem Schritt i werden die Kanten des nächsten Knotens $v_i \in V$ (bezüglich der Ordnung $\prec$) aufgedeckt. Ein Online-Algorithmus muß dann die unwiderrufliche Entscheidung treffen, welche Kante in das Matching M aufgenommen wird, falls dies überhaupt noch möglich ist.
Die erste Untersuchung dieses Modells erfolgte in [KVV90]. Dort wurde gezeigt, daß der einfache Greedy-Algorithmus 2-competitive ist. Greedy betrachtet nacheinander die Kanten des aktuellen Knotens $v_i \in V$ und nimmt die erste dieser Kanten in das Matching M auf, für die es legal ist. Auf diese Weise bestimmt er ein maximales Matching, d. h. M kann nicht durch Hinzunahme einer weiteren Kante vergrößert werden. Es ist wohlbekannt, daß ein maximales Matching mindestens halb so viele Kanten enthalten muß, als ein optimales Maximum-Kardinalitäts-Matching[18]. Dieser Greedy-Algorithmus ist optimal, wie folgende Gegenspielerstrategie zeigt: Zuerst wird ein Knoten $v \in V$ mit den Kanten $\{v, u_1\}$ und $\{v, u_2\}$ $(u_1, u_2 \in U)$ eingegeben. Entscheidet sich ein Online-Algorithmus für $\{v, u_1\} \in M$, so ist die nächste Eingabe des Gegenspielers ein Knoten $v' \in V$ mit der Kante $\{v', u_1\}$, anderenfalls wird $\{v', u_2\}$ vom Gegenspieler erzeugt. Der Online-Algorithmus kann nur eine Kante in das Matching M einfügen, während die optimale Lösung um zwei Kanten vergrößert wird. Dieses Spiel wird mit neuen Knoten, die die Rollen von v, v', u_1 und u_2 übernehmen, ständig wiederholt. Damit ist die untere Schranke für den Competitive Ratio $\mathcal{R} \geqq 2$ gezeigt.
Der Hauptbeitrag des Aufsatzes [KVV90] liegt allerdings in der Entwicklung und Analyse eines randomisierten Online-Algorithmus namens Ranking, für den $\overline{\mathcal{R}}_{\mathsf{BL}}(\mathsf{Ranking}) = e/(e-1) \approx 1.582$ gilt. Dieses Ergebnis ist ebenfalls optimal (d. h. $\overline{\mathcal{R}}_{\mathsf{BL}} \geqq e/(e-1)$).

Das selbe Modell wurde in [KT91] mit Blockeingabe studiert, d. h. in jedem Schritt werden die nächsten k Knoten der Eingabe aufgedeckt. Für $k = 1$ ergibt sich das soeben betrachtete Modell und für $k = n$ erhält man die Offline-Variante des Problems. Die Blockeingabe ist jedoch von wenig Nutzen, da sich die oben ausgeführte Gegenspielerstrategie leicht modifizieren läßt, um für $k \leqq n/2$ die untere Schranke von 2 aufrecht zu erhalten.

3.4.3 Gewichtetes Online Bipartite Matching

In diesem Modell erhalten die Kanten des bipartiten Graphen zusätzlich ein nichtnegatives Gewicht, d. h. es ist eine Gewichtsfunktion $w : E \to \mathbb{R}_+$ gegeben. Es ist leicht zu erkennen, daß ohne Anforderungen an die Graphenstruktur und Einschränkungen in der Gewichtsfunktion keine competitiven Online-Algorithmen existieren können, falls eine Funktion in der Gewichtssumme der Matchingkanten zu optimieren ist. Deshalb ist in der Literatur das Modell derart eingeschränkt

[18] Beweis: Alle $2 \cdot |M|$ Knoten eines nicht erweiterbaren Matchings M bilden ein Vertex Cover VC; ein minimales Vertex Cover $VC_{\min}$ muß auch alle Kanten eines Maximum-Kardinalitäts-Matchings M_{opt} abdecken, hat also eine Größe von mindestens $|M_{\text{opt}}|$; es folgt $2 \cdot |M| \geqq |VC_{\min}| \geqq |M_{\text{opt}}| \Rightarrow |M_{\text{opt}}|/|M| \leqq 2$.

worden, daß der bipartite Graph vollständig ist und die Gewichtsfunktion von solcher Struktur, daß der Graph einen metrischen Raum darstellt. Insbesondere muß dann für die gewichteten Wege in diesem Graphen die Dreiecksungleichung gelten.
In [KP93] wurde für das Ziel ein Matching mit maximaler Gewichtssumme online zu berechnen ein Competitive Ratio von exakt 3 festgestellt.
Gilt für den vollständigen bipartiten Graphen $|U| = |V| = n$, d. h. die beiden Knotenpartitionen sind von identischer Kardinalität, so existiert immer ein perfektes Matching. Das ist ein Matching M, in dem alle Knoten V mit Matchingkanten inzident sind bzw. $|M| = n$ gilt. Das Optimierungsziel kann auch in der Bestimmung eines perfekten Matchings mit minimaler Gewichtssumme bestehen. Für diese Aufgabe wurde unabhängig voneinander in [KP93] und [KMV94] der Competitive Ratio von $2n - 1$ bestimmt. In der letztgenannten Quelle wird auch das Problem der stabilen Ehen [GS62, GI89] untersucht. Innerhalb des vollständigen bipartiten Graphen sind die Präferenzen der einen Seite a priori bekannt, die der anderen Seite wird im Online-Stil eingegeben. Der FCFS-Algorithmus[19] erreicht durchschnittlich $O(n \log n)$ instabile Paare in seiner Lösung. Für beliebige Online-Algorithmen wurde der schlimmste Fall jedoch mit $\Omega(n^2)$ instabilen Paaren nachgewiesen.

3.4.4 Online b-Matching

Diese Variante des Matching-Problems in bipartiten Graphen $G = (U \dot{\cup} V, E)$ läßt die Zuordnung von bis zu b Matchingkanten zu Knoten aus der a priori bekannten Partition U zu. Die Knoten aus V werden wiederum im Online-Stil eingegeben. Die Entscheidung eine Kante von $v_i \in V$ in das Matching M aufzunehmen muß instantan nach Bekanntgabe der Kanten von v_i erfolgen. Ziel ist die Bestimmung einer großen Anzahl von Matchingkanten.
[KP00] untersucht dieses Problem mit einem Gegenspielermodell, welches pro Knoten $u \in U$ nur bis zu a Matchingkanten zuordnen darf. Dieses Matching-Problem wird optimal durch den Online-Algorithmus Balance gelöst. Er nimmt die Kante des aktuellen Knotens v_i in das Matching auf, welche zu einem Knoten aus U mit minimaler Anzahl bereits bestimmter Matchingkanten adjazent ist. Sollten schon alle Nachbarknoten von v_i mit b Matchingkanten gesättigt sein, so kann Balance keine Kante von v_i in M aufnehmen. Die Analyse dieses Online-Algorithmus ergibt einen Competitive Ratio von

$$\mathcal{R}(\mathsf{Balance}) \;=\; \frac{(1+\frac{1}{a})^b}{(1+\frac{1}{a})^b - 1}\,. \tag{11}$$

In [KP00] wird für die allgemeine untere Schranke im Competitive Ratio dieses Matching-Problems der selbe Term bewiesen.
Der spezielle Fall für $a = b$ wurde schon in [KP96] analysiert. Wird $a = b = 1$ gesetzt, so ergibt sich der 2-competitive Greedy-Algorithmus aus [KVV90]. Für

[19] first come, first serve

den Grenzwert von $a = b$ gegen Unendlich ergibt sich aus Gleichung (11)

$$\lim_{b\to\infty} \frac{(1+\frac{1}{b})^b}{(1+\frac{1}{b})^b - 1} \;=\; \frac{e}{e-1} \;\approx\; 1.582 \;,$$

womit der Competitive Ratio für dieses Problem auf ein Intervall eingegrenzt ist.

Das b-Matching-Problem läßt sich auch so auffassen, daß die Knotenpartition U eine Ressourcemenge darstellt und V eine Menge von Aufträgen. Es handelt sich dabei um eine spezielle Form des Lastbalancierungsproblems mit permanenten Aufträgen und Beschränkung der Maximallast. So stellt es eine Verbindung zu Lastbalancierungsproblemen mit Einheitsaufträgen und beschränkter Zuordnung her. Werden nun Kantengewichte eingeführt und wird die Begrenzung von b, der Anzahl Matchingkanten eines Ressourceknotens, fallen gelassen, so ergibt sich das Lastbalancierungsmodell aus [ANR95], welches schon auf Seite 48 diskutiert wurde.

3.4.5 Online Perfektes Matching

In diesem Modell ist ein bipartiter Graph $G = (U \dot\cup V, E)$ gegeben, und die a priori bekannte, fixierte Partition U wird als Menge von Ressourcen interpretiert. Sobald ein Knoten $v_i \in V$ aufgedeckt wird, muß er mit einem Knoten aus U verbunden werden. Es ist also die Invariante eines perfekten Matchings bezüglich der Partition V zu erhalten. Um v_i mit einem Knoten $u \in U$ zu verbinden, kann es notwendig werden frühere Knoten aus V den Ressourcen U neu zuzuordnen.[20] Die Anzahl solcher Neuzuordnungen ist gering zu halten, und die $|V| = n$ Zuordnungen einer optimalen Offline-Lösung wird mit der von einem Online-Algorithmus benötigten Zuordnungsanzahl verglichen.
In [GKKV95] wird eine Greedy-Strategie vorgeschlagen, die in jedem Online-Schritt eine Teillösung bestimmt, in der die Anzahl der Neuzuordnungen minimiert wird.[21] Dieser Online-Algorithmus ist $O(\log n)$-competitive und die allgemeine untere Schranke für dieses Modell wird mit $\mathcal{R} \in \Omega(\log n)$ bestimmt.

3.4.6 Das Online-Problem der Zimmerpartner

Das Problem der Zimmerpartner (Roommates-Problem) ist ein Matching-Problem in allgemeinen Graphen. Die Online-Formulierung stammt aus [BR93] und wird wie folgt interpretiert: Gegeben sei ein Konferenzhotel, welches nur aus Doppelzimmern besteht. Die Gäste der Konferenz treffen nacheinander ein und bringen eine Liste mit, in der sie mögliche Zimmerpartner benennen. Es wird angenommen, daß diese Listen symmetrisch sind, d. h. wenn Person A bereit ist sich ein Zimmer mit Person B zu teilen, so gilt das auch umgekehrt. Die Gäste können dann als Knoten eines ungerichteten Graphen modelliert werden, und die

[20] D. h. das Matching wird durch einen erweiternden Weg vergrößert.
[21] Es wird also ein kürzester, erweiternder Weg berechnet.

Paare, die bereit sind sich ein Zimmer zu teilen, sind im Graphen durch eine Kante verbunden. Es wird auch die Variante eines gewichteten Graphen vorgestellt. Dann entsprechen die Kantengewichte einem „Zufriedenheitswert“ mit dem Zimmerpartner oder einer Priorität.
Nachdem ein Gast im Hotel angekommen ist, muß ihm der Hotelmanager sofort ein Zimmer zuweisen. Das Ziel des Managers besteht aus wirtschaftlichen Gründen darin, möglichst wenige Zimmer zu belegen, d. h. ein Matching größtmöglicher Kardinalität zu finden. Im Falle der priorisierten Partnerlisten — also des gewichteten Graphen — möchte der Manager die totale Zufriedenheit maximieren, d. h. ein Matching mit maximaler Gewichtssumme bestimmen.
In der graphentheoretischen Betrachtung tritt im Online-Fall folgende Besonderheit auf: Bei der Eingabe des Knotens $v_i \in V$ im i-ten Schritt werden alle seine Nachbarknoten bekannt. Eine Zuordnung ist in diesem Moment jedoch nur zu den schon aufgedeckten Knoten v_j mit $j < i$ möglich. Sollte der Knoten v_i nicht im Schritt i in das Matching aufgenommen werden, so ergibt sich später u. U. nochmals diese Möglichkeit, und zwar bei jedem Nachbarknoten, der nach Eingabe von v_i aufgedeckt wird. Eine kanonische Interpretation im Sinne von Ressourcen und Aufträgen, wie in bipartiten Graphen möglich, scheitert in diesem Modell, da im Schritt i der Knoten v_i die Rolle eines Auftrages hat und im Falle des Nichtzuordnens später die Rolle einer Ressource einnimmt.
In [BR93] werden für beide Varianten des gewichteten und ungewichteten Online-Problems der Zimmerpartner untere Schranken für den Competitive Ratio bestimmt und obere Schranken durch Analysen von Online-Algorithmen angegeben. Im ungewichteten Fall kann der Competitive Ratio auf exakt 1.5 festgelegt werden. Für die gewichtete Variante bleibt eine Lücke, da die allgemeingültige untere Schranke im Competitive Ratio mit 3 bestimmt und ein Online-Algorithmus als 4-competitive nachgewiesen wird.

4 Die Modelle

Übersicht: Dieses Kapitel ist den formalen Definitionen und Modellbeschreibungen gewidmet. Im ersten Teil werden Basiskonzepte, wie Graphen und einige ihrer Eigenschaften, wiederholt. Daneben wird das erste, sehr grundlegende Online-Modell eingeführt sowie eine graphische Notation. Sie unterstützt in den späteren Kapiteln die Argumentation innerhalb der Beweise.
Danach werden verschiedene konzeptionelle Erweiterungen des ersten Modells eingeführt, motiviert und als Generalisierung oder Spezialisierung charakterisiert. Ein weiteres Unterkapitel beschäftigt sich mit einem Vergleich dieser neu formulierten und aus der Literatur bekannten Modellen. Dabei werden besonders Unterschiede herausgearbeitet, aber es werden auch Fälle aufgeführt, in denen sich Modelle aus der Literatur als Generalisierungen der in dieser Arbeit betrachteten Probleme herausstellen.

4.1 Das grundlegende Modell

Bevor nun die Modelle vorgestellt werden, die der Untersuchungsgegenstand der weiteren Arbeit sind, sollen im ersten Teilkapitel einige Definitionen wiederholt und Notationen geklärt und eingeführt werden.

4.1.1 Einige Notationen und graphentheoretische Definitionen

Innerhalb dieser Abhandlung wird vorausgesetzt, daß der Leser mit den Grundkonzepten der theoretischen Informatik und der Graphentheorie vertraut ist. Da jedoch einige Begriffe häufig in leicht unterschiedlichen Bedeutungen verwandt werden, sollen die folgenden formalen Definitionen Unklarheiten beseitigen. Zudem werden einige Notationen eingeführt, die später exzessiv benutzt werden.

Definition 12: (Gewichteter Graph)

$G = (V, E, w)$ *heißt gewichteter Graph genau dann, wenn gilt:*

V *ist eine endliche Menge von Knoten,*

$E \subset \{\{u, v\} \mid u, v \in V \wedge u \neq v\}$ *ist eine Menge von Kanten, und*

$w : E \to \mathbb{R}_+$ *ist eine Gewichtsfunktion.*

Falls die Einheitsgewichtsfunktion $w : E \to 1$ benutzt wird, so entsteht ein ungewichteter Graph $G = (V, E)$. Dabei wird die Gewichtsfunktion w aus der Notation entfernt. Diese Transformation von einem gewichteten zu einem ungewichteten Graphen überträgt sich auch auf die nächsten Definitionen von Graphen und Matchings.
Bipartite Graphen bestehen aus zwei disjunkten Knotenmengen, die innerhalb der Mengen keine Kanten besitzen:

Definition 13: (Bipartiter gewichteter Graph)

$G = (U \dot\cup V, E, w)$ *heißt bipartiter gewichteter Graph genau dann, wenn gilt:*

U *und* V *sind endliche Menge von Knoten,*

$E \subset \{\{u, v\} \mid u \in U \wedge v \in V\}$ *ist eine Menge von Kanten, und*

$w : E \to \mathbb{R}_+$ *ist eine Gewichtsfunktion.*

Definition 14: (Knoteninduzierter Teilgraph)

Sei $G = (V, E, w)$ *ein gewichteter Graph und* $V_S \subset V$. *Dann ist*

$$G\big|_{V_S} := (V_S, \{\{u, v\} \mid u, v \in V_S, \{u, v\} \in E\}, w)$$

der durch die Knotenmenge V_S *induzierte Teilgraph von* G.

Definition 15: (Matching)

Sei $G = (V, E, w)$ ein gewichteter Graph. M heißt Matching in G genau dann, wenn gilt:

$$M \subset E \text{ und}$$
$$\{u, v\} \in M \;\Rightarrow\; \nexists\, x \in V : \{u, x\} \in M \wedge x \neq v \;.$$

Definition 16: (Gewicht eines Matchings, $|M|$)

Sei $G = (V, E, w)$ ein gewichteter Graph und sei M ein Matching in G.

$$|M| \;:=\; \sum_{\{u,v\} \in M} w(\{u, v\})$$

heißt Gewicht des Matchings M.

Es ist offensichtlich, daß $|M|$ die Anzahl der Kanten in M zählt, wenn diese Definition auf einen ungewichteten Graphen angewandt wird. In einem solchen Fall wird auch von der *Kardinalität* oder *Größe* des Matchings M gesprochen. Ist ein Matching M in einem Graphen G gegeben, so werden *freie* und *gebundene* Knoten unterschieden. Ein Knoten v ist frei, falls er nicht inzident zu einer Matchingkante aus M ist. Anderenfalls ist der Knoten v ein Endknoten einer eindeutigen Matchingkante und wird als gebunden bezeichnet. Um diese Zugehörigkeit eines Knotens v zu einer Kante im Matching M auszudrücken, wird folgende Kurzschreibweise eingeführt:

Definition 17: (Notation: $\mathring{\in}$)

Sei $G = (V, E, w)$ ein Graph und sei $M \subset E$ ein Matching. Für jedes $v \in V$ gilt:

$$v \mathring{\in} M \;:\Leftrightarrow\; \exists\, u \in V : \{v, u\} \in M \text{ und}$$
$$v \mathring{\notin} M \;:\Leftrightarrow\; \nexists\, u \in V : \{v, u\} \in M \;.$$

Definition 18: (Maximum-gewichtetes Matching, $\mathcal{M}(G)$)

Sei $G = (V, E, w)$ ein gewichteter Graph. Dann bezeichnet $\mathcal{M}(G)$ ein maximum-gewichtetes Matching:

$$\mathcal{M}(G) \;\in\; \{M \mid \nexists\, M' : |M| < |M'| \wedge M, M' \text{ sind Matchings in } G\} \;.$$

Mit dieser Definition wird $\mathcal{M}(G)$ nicht eindeutig angegeben, da in G mehrere Matchings mit der Extremal-Charakteristik existieren können. Wichtig ist jedoch diese spezielle Eigenschaft, die betont werden soll. Bei der späteren Anwendung

der Notation $\mathcal{M}(G)$ wird typischerweise ein beliebiges, aber festes Matching aus dieser Menge verstanden.

Definition 19: (Symmetrische Differenz, $\oplus$)

Seien A und B zwei Mengen. Dann bezeichnet $A \oplus B$ die symmetrische Differenz:

$$A \oplus B \;:=\; (A \cup B) \setminus (A \cap B) \;=\; (A \setminus B) \cup (B \setminus A) \;.$$

Definition 20: (alternierender Weg)

Sei $G = (V, E, w)$ ein gewichteter Graph und M ein Matching in G. Ein Weg $P = (v_1, v_2, \ldots, v_p)$ mit $v_i \in V$ heißt alternierender Weg in M genau dann, wenn die aufeinanderfolgenden Kanten $\{v_1, v_2\}, \{v_2, v_3\}, \ldots, \{v_{p-1}, v_p\}$ im Weg P abwechselnd dem Matching M angehören und nicht angehören.

Es wechseln sich also entlang des Weges P die Matchingkanten und Nichtmatchingkanten ab.

Definition 21: (erweiternder Weg)

Sei $G = (V, E)$ ein ungewichteter Graph und $P = (v_1, v_2, \ldots, v_p)$ ein alternierender Weg im Matching M von G. P heißt erweiternder Weg in M genau dann, wenn die beiden Endknoten v_1 und v_p freie Knoten in M sind ($v_1, v_p \mathring{\notin} M$).

Ein erweiternder Weg P kann benutzt werden, um ein vorhandenes Matching M um eine Kante zu vergrößern. Da beide Endknoten des Weges frei sind, können auf dem Weg P alle Nichtmatchingkanten in das Matching M aufgenommen werden und dafür werden alle originalen Matchingkanten des Weges P aus M entfernt. Da die Kantenanzahl in P ungerade ist, und er eine Nichtmatchingkante mehr als Matchingkanten besitzt, so wird durch diese Operation die Kardinalität des Matchings um Eins erhöht.
Sei M ein beliebiges Matching in G und $\mathcal{M}(G)$ ein Matching maximaler Kardinalität im selben Graphen G. Die symmetrische Differenz $M \oplus \mathcal{M}(G)$ beschreibt eine Kantenmenge in G, die aus alternierenden Wegen oder Kreisen in G bezüglich der Matchings M und $\mathcal{M}(G)$ besteht. Die Kreise besitzen gerade Länge und zeigen ebenso wie die Wege gerader Länge nur strukturelle Unterschiede zwischen M und $\mathcal{M}(G)$ auf. Die Wege ungerader Länge in $M \oplus \mathcal{M}(G)$ sind jedoch erweiternde Wege in M, und die Erweiterung dieser disjunkten Wege transformiert M in ein Matching maximaler Kardinalität. Der formale Beweis, daß ein Matching in einem ungewichteten Graphen genau dann und nur dann von maximaler Kardinalität ist, wenn keine erweiternden Wege existieren, wurde bereits in [Ber57] geführt. Da in bipartiten Graphen keine Kreise ungerader Länge existieren, führt dieses Theorem zu einem ersten effizienten Algorithmus, der in bipartiten Graphen Matchings maximaler Kardinalität bestimmt. Für allgemeine ungewichtete Graphen wurde dieses Problem erst in [Edm65b] gelöst. Letztere Arbeit sei zum vertiefenden Nachlesen der oben dargestellten Sachverhalte empfohlen.

Die Definition eines erweiternden Weges in einem gewichteten Graphen ist etwas komplizierter. Die Grundlage bildet ebenfalls ein alternierender Weg. Die Endknoten müssen dabei nicht frei sein, aber bei einem gebundenen Endknoten muß die inzidente Matchingkante Teil des Weges sein. Ein so beschriebener alternierender Weg ist dann ein erweiternder Weg, wenn die Gewichtssumme seiner Nichtmatchingkanten größer ist als die Summe der Gewichte der Matchingkanten dieses Weges. Dann kann ebenfalls durch das Austauschen der Matching- und Nichtmatchingeigenschaft der Kanten des Weges eine Vergrößerung des Gewichtes des Matchings erreicht werden.
Eine Konsequenz dieser Definition besteht darin, daß ein erweiternder Weg in einem gewichteten Graphen auch eine gerade Anzahl von Kanten besitzen kann bzw. daß bei der Erweiterung des Weges die Anzahl der Kanten im Matching konstant bleiben oder um Eins zu-, aber auch abnehmen kann.

4.1.2 Das Online-Request-Server-Matching-Problem

Der Name *Online-Request-Server-Matching-Problem* oder verkürzt ORSM-Problem, ergibt sich aus der Motivation und Interpretation des Modells. Dem sehr abstrakten Modell selbst liegt ein ungewichteter bipartiter Graph $G = (R \dot\cup S, E)$ zugrunde. Auf beiden Knotenmengen R und S ist jeweils eine totale Ordnung $\prec$ gegeben. Für die Knoten werden die Bezeichner $r_1, r_2, r_3, \ldots$ und $s_1, s_2, s_3, \ldots$ benutzt, wobei $r_i \in R$, $s_j \in S$, $i, j \in \mathbb{N}$ gilt, und die Indizes die Stellung der Knoten innerhalb der Ordnung $\prec$ angeben. Diese totale Ordnung stellt ein diskretes Zeitmodell dar. Die Knoten der Menge S sollten als eine Ressource, *Server* genannt, interpretiert werden. Diese Ressource steht in jedem Zeitschritt i für eine Bearbeitungseinheit zur Verfügung. Der Knoten s_i bildet diese Tatsache ab. Die Knotenmenge R wird am besten als eine Menge von Aufträgen, hier *Requests* genannt, aufgefaßt. Pro Zeitschritt i kann ein Auftrag r_i auftreten, der den Ressourcebedarf nach einer Bearbeitungseinheit hat. Eine Kante $\{r_i, s_j\}$ bedeutet, daß der Auftrag r_i von der Ressource zum Zeitpunkt j — also mittels s_j — bearbeitet werden kann. Damit stellen die zu einem Requestknoten r_i benachbarte Serverknoten $\{s_j \mid s_j \in S, \{r_i, s_j\} \in E\}$ die Menge aller Zeitpunkte dar, zu denen der Auftrag bearbeitet werden kann. Die vollständige Kantenmenge $E \subset R \times S$ unterliegt der Einschränkung

$$\{r_i, s_j\} \in E \;\Rightarrow\; i \leqq j \quad . \tag{12}$$

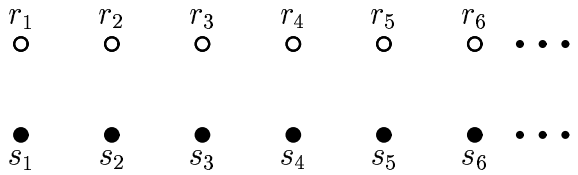
Das bedeutet nichts anderes, als daß ein Request keinen Servicezeitpunkt spezifizieren darf, der vor seinem eigenen Auftreten liegt. Dies würde für eine Anwendung auch keinen Sinn machen und Online-Algorithmen mit einem beschränkten Competitive Ratio könnten nicht mehr existieren. Es ist zu beachten, daß diese Definition der Kantenmenge nicht die Spezifikation eines geschlossenen Zeitintervalls für die Bearbeitung eines Requests fordert. Ebenso muß der Zeitpunkt i eines Requestknotens r_i nicht als Servicezeitpunkt zulässig sein, d. h. die Kante $\{r_i, s_i\}$ muß nicht existieren. Würden solche Forderungen gestellt, so ergäben sich sehr

einfache Modelle mit trivialen, optimalen Lösungsalgorithmen. Die Einführung dieser allgemeineren und vielleicht unmotiviert anmutenden Definition der Kantenmenge hat den folgenden Grund. Dieses Modell stellt eine Vorstufe zu komplexeren Modellen dar, wie sie in Kapitel 4.3 definiert werden. Die für das ORSM-Problem entwickelten algorithmischen Ideen und Analyseergebnisse können dann übertragen werden.
Es ergibt sich nun ein Planungsproblem, welches zum Ziel hat möglichst viele der Requests zu bearbeiten. Jedes Matching M in dem bipartiten Graphen stellt einen gültigen Bearbeitungsplan dar, da jede Matchingkante $\{r_i, s_j\} \in M$ den Auftrag r_i dem Zeitpunkt j der Serverressource eindeutig zuordnet. Das Maximierungsziel des Planungsproblems ist dann identisch mit dem Problem der Bestimmung eines Matchings maximaler Kardinalität.
Nach der Beschreibung der Modellstruktur muß nun die Frage geklärt werden, wie sich das System im Online-Fall verhält. Zum Startzeitpunkt ist die vollständige Knotenpartition S bekannt. In jedem Zeitschritt i wird der nächste Knoten r_i der Requestpartition einschließlich seiner Kanten aufgedeckt. Dieser Knoten war vorher unbekannt.[22] Tritt keine neue Anfrage auf, so bleibt r_i isoliert.
Nach der Eingabe von r_i hat ein Online-Algorithmus zu entscheiden, welcher Auftrag r_k im Schritt i mit der Ressource bearbeitet wird. Dabei muß gelten: $\{r_k, s_i\} \in E$ und $r_k \mathring{\notin} M_{i-1}$, dem bis Schritt $i-1$ bestimmten Matching. Das heißt s_i kann zusammen mit einer inzidenten Kante unwiderruflich in das Matching M der Lösung aufgenommen werden oder es gilt $s_i \mathring{\notin} M$. Durch die Restriktion (12) ist außerdem sichergestellt, daß zu diesem Entscheidungszeitpunkt *alle* zu s_i inzidenten Kanten offengelegt sind.
Nach der Entscheidung, ob und wie der Knoten s_i in das Matching M aufgenommen wird, schreitet das Zeitmodell einen Schritt fort und es erfolgt die Eingabe des Requestknotens r_{i+1}.

4.1.3 Die graphische Notation

Zur Illustration der Graphen und Matchings wird eine bildhafte Darstellung benutzt. Sie ist völlig äquivalent zur mengentheoretischen Repräsentation. Die Knoten der beiden Mengen R und S werden durch kleine Kreise ○ bzw. Kreisscheiben ● dargestellt. Die Kreise stehen für Knoten der Requestpartition R, und die Kreisscheiben repräsentieren die Ressource S. Falls erforderlich, werden zusätzlich die Bezeichner an die Knoten geschrieben.
Soll das Bild eine Instanz des ORSM-Problems beschreiben, so werden die Knoten der Requestpartition, gemäß ihrer Ordnung, in äquidistanten Abständen von links nach rechts horizontal aufgetragen. Die Knoten der Serverpartition werden in gleicher Weise in einigem Abstand darunter angeordnet, und zwar so, daß Knoten mit identischem Zeitindex übereinander stehen:

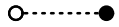
[22] Es ist zwar offensichtlich, daß pro Zeitschritt i ein Knoten r_i aus R existiert, die Menge seiner inzidenten Kanten sind jedoch vor Schritt i unbekannt. Zur Vereinfachung der Darstellung soll deshalb die Interpretation der Eingabe von Requestknoten beibehalten werden.



Die Kanten werden als Linien zwischen zwei Knoten dargestellt. Da der Graph bipartit ist, hat eine Linie immer einen Kreis und eine Kreisscheibe als Endpunkt:

Um Kanten eines Matchings auszuzeichnen, werden sie als Doppellinie symbolisiert:

Falls eine Kante nicht mehr als Matchingkante ausgewählt werden darf, da sonst die Matchingbedingung verletzt würde oder der inzidente Serverknoten einen abgelaufenen Zeitindex besitzt, so wird sie gestrichelt dargestellt:



4.2 Die Modellerweiterung mit Gewichten

Das Hinzufügen einer Gewichtsfunktion $w : E \to \mathbb{R}_+$ zum bipartiten Graphen G des ORSM-Problems führt zu einer Generalisierung des Modells. Es wird *gewichtetes Online-Request-Server-Matching* Problem genannt und mit wORSM-Problem abgekürzt. Nun besteht das Ziel eines Online-Algorithmus nicht mehr in der Maximierung der Kardinalität des berechneten Matchings, sondern in der Maximierung seines Gesamtgewichtes $|M|$.
Die Gewichtsfunktion w kann als Auftragspriorität aufgefaßt werden oder das Kantengewicht $w(\{r_i, s_j\})$ wird als verdienter Gewinn für den Fall der Aufnahme der Kante $\{r_i, s_j\}$ in das Matching M betrachtet.
Innerhalb der graphischen Notation werden die Kantengewichte, falls erforderlich, an die Linien der Kanten geschrieben.

4.3 Modellvarianten mit Restriktionen

Die sehr einfache Struktur des ORSM-Problems kann mit zusätzlichen Konzepten angereichert werden, um somit realitätsnähere und für Anwendungen interessantere Modelle zu erhalten. Diese Modellerweiterungen umfassen Anfragen, die je ein Zeitintervall für die Bearbeitung spezifizieren, Vorschau für den Online-Algorithmus, die Eingabe der Requestknoten in Blöcken und zwei strukturelle Beschränkungen der Kantenmenge: Pro gestellter Anfrage wird eine identische Anzahl von Servicezeitpunkten spezifiziert sowie eine a priori Beschränkung der Zeitspanne zwischen einer Anfrage und ihrem letzten zulässigen Servicezeitpunkt. Außerdem lassen sich einzelne dieser Basiskonzepte in einem Modell kombinieren. Wie sich herausstellt können all diese Ideen durch das ORSM-Modell mit zusätzlichen Beschränkungen in der Kantenmenge abgebildet werden. Daraus folgt,

daß ein Online-Algorithmus über mehr Wissen in der Problemstruktur verfügt und dieses ausnutzen kann. Da diese Modellvarianten echte Spezialisierungen des ORSM-Ausgangsproblems sind, lassen sich einige Resultate der ORSM-Analyse übertragen.
Im folgenden werden die einzelnen Konzepte erläutert und ihre Darstellung als ORSM-Modell mit zusätzlichen Restriktionen an die Kantenmenge aufgezeigt.

4.3.1 Das Modell mit Intervalleingaben

In dieser Version des ORSM-Problems darf jeder Request r_i nur ein Zeitintervall $[t_i, d_i]$ angeben. Es gilt dabei $i \leqq t_i \leqq d_i$, $t_i, d_i \in \mathbb{N}$. Zu jedem Zeitpunkt innerhalb des Intervalls wird die Bearbeitung des Auftrages r_i akzeptiert, d. h. die Menge der zu r_i adjazenten Serverknoten ist $\{s_j \mid t_i \leqq j \leqq d_i\}$. Natürlich können auch in dieser Modellvariante isolierte Requestknoten r_i existieren, wenn zum Zeitpunkt i keine Anfrage gestellt wird.
Die Definition dieses Modells mit Intervalleingabe ist ein direkter Spezialfall des ORSM-Problems.

4.3.2 Das Modell mit Vorschau

Die Abwandlung eines Online-Problems zu einem Problem mit ℓ-Vorschau wurde schon in Kapitel 2.5.2 vorgestellt. Auch für das ORSM-Problem ist diese Variation einfach zu übertragen. Dabei sind zum Zeitpunkt i, an dem über die Benutzung des Serverknotens s_i entschieden wird, schon alle Requestknoten bis einschließlich $r_{i+\ell}$ aufgedeckt.
Um dieses zusätzliche Wissen im Standard-ORSM-Modell abzubilden, genügt es die Knoten der Requestpartition um ℓ Zeitschritte vorzuziehen. Diese Ansicht ist identisch zu einer Erhöhung der Indizes der Serverknoten um ℓ. Es gibt dann keine Kante $\{r_i, s_j\}$ mit $j - i < \ell$. Ergo ist das ORSM-Modell mit ℓ-Vorschau äquivalent zu dem ORSM-Problem mit der Einschränkung $\{r_i, s_j\} \in E \Rightarrow i + \ell \leqq j$.

4.3.3 Das Modell mit Blockeingabe

Auch das Konzept von Blockeingabe (siehe dazu das Beispiel am Ende des Kapitels 3.4.2) läßt sich auf das ORSM-Problem übertragen. Dabei werden in jedem Zeitschritt i die nächsten b ($b \in \mathbb{N}, b \geqq 2$) Aufträge $(r_{ib+1}, \ldots, r_{(i+1)b})$ eingegeben und es wird über die Nutzung der nächsten b Serverknoten $(s_{ib+1}, \ldots, s_{(i+1)b})$ entschieden. Somit wird die totale Ordnung auf R und S aufgehoben und in eine spezielle, partielle Ordnung transformiert. Diese Variante des Modells kann auch als ein ORSM-Modell mit b parallelen Serverressourcen oder einer Ressource mit Bearbeitungskapazität von b Einheiten pro Zeitschritt aufgefaßt werden.
Eine Modellierung dieser verbesserten Informationsbasis ist durch das ORSM-Problem wie folgt möglich. Die Indizes der totalen Ordnung auf R und S werden in durchnumerierte Blöcke der Länge b aufgeteilt, d. h. Block j beinhaltet die Indizes $jb+1, \ldots, (j+1)b$. Die b Requestknoten des Zeitschrittes i werden durch die

Requests des Blockes $2i-1$ dargestellt und die Serverknoten des selben Schrittes durch die Serverknoten des Blockes $2i$. Es existieren dann nur noch Kanten zwischen Requestknoten aus ungeraden Blöcken zu Serverknoten aus geraden Blocknummern.
Mit dieser Restriktion der Kantenmenge ist wiederum eine äquivalente Formulierung des ORSM-Modells mit Blockeingabe durch das Basismodell des ORSM-Problems gelungen.

4.3.4 Das Modell mit konstantem Requestknotengrad

In dieser Modellvariante wird verlangt, daß ein Requestknoten r_i entweder *genau* g ($g \in \mathbb{N}, g \geqq 2$) Kanten zu Serverknoten besitzt oder r_i isoliert ist. Damit wird eine Begrenzung von möglichen Servicezeitpunkten der Aufträge abgebildet. Wie sich jedoch herausgestellt hat, ist eine bloße Beschränkung des Requestknotengrades nicht sinnvoll, da die Gegenspielerstrategien zum Aufbau extremer Überlastpunkte immer wieder von Requests mit kleinem Knotengrad Gebrauch machen.
Diese Definition impliziert, daß das Modell mit konstantem Requestknotengrad ein Spezialfall des ORSM-Problems ist.

4.3.5 Das Modell mit begrenzter Kantenlänge

Dieses Modell stellt ein erster Schritt in Richtung eines Planungsproblems mit Realzeitanforderungen dar. Ein Auftrag r_i muß alle akzeptierten Servicezeitpunkte innerhalb des Zeitfensters $[i, i+d]$ spezifizieren. So besitzt jeder Auftrag von vorn herein einen individuellen spätesten Fertigstellungszeitpunkt (Deadline) von d ($d \in \mathbb{N}, d \geqq 2$). Diese Eigenschaft des Gesamtsystems ist dem Online-Planungsalgorithmus bekannt.
Formal läßt sich diese Variante des ORSM-Problems mit der Restriktion der Kantenmenge: $\{r_i, s_j\} \in E \Rightarrow j \leqq i+d$ beschreiben.

Für die Parameterwahl $b = 1$ ergibt sich das ORSM-Basismodell. Bei $g = 1$ oder $d = 1$ ergibt sich ein ORSM-Modell mit Intervalleingabe, da durch diese Beschränkungen eine Anfrage keine zwei getrennten Zeitintervalle für ihre Bearbeitung konstruieren kann. Deshalb wurden diese Werte für die Parameter schon bei der Modelldefinition ausgeschlossen.

4.3.6 Kombinationen der zusätzlichen Konzepte und das Zugriffsproblem in einem parallelen Realzeit-Datenserver (DAP)

Die bisher definierten Einzelkonzepte können auch zu weiteren Modellvarianten kombiniert werden. Es ist jedoch vorteilhaft, diese Ideen zuerst allein zu untersuchen und danach schrittweise verschiedene Kombinationen zu betrachten.
Mehrere der oben definierten Konzepte finden sich in dem folgenden komplexeren Planungsproblem wieder. Es ist in der Literatur [MBLR97] unter dem Namen *Data-Access-Problem* (bzw. DAP als Kurzschreibweise) eingeführt wor-

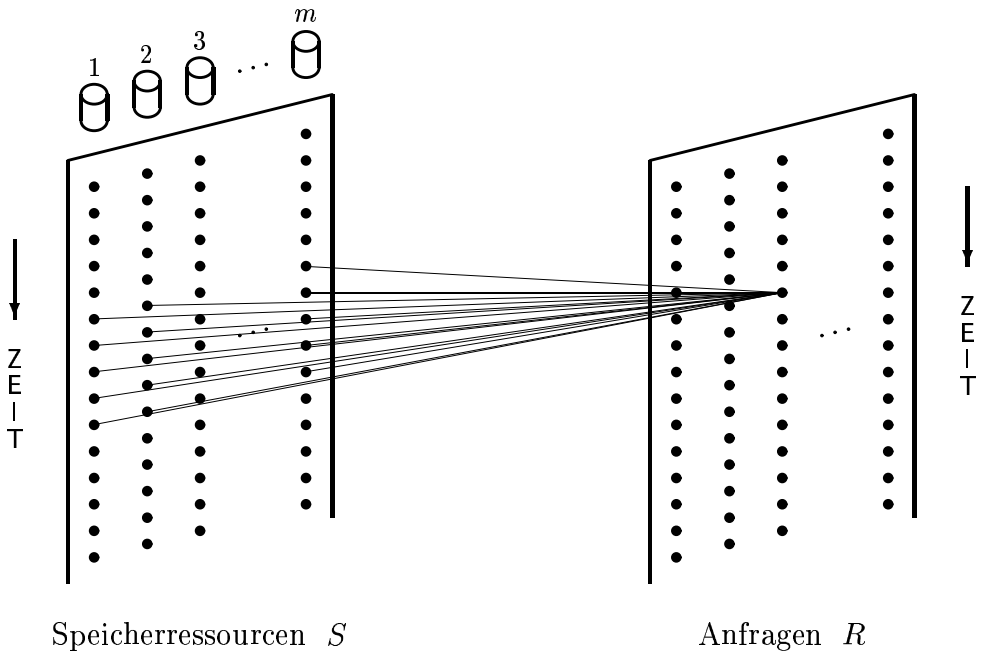
den und stellt ein abstraktes Zugriffsproblem in einem Datenserver mit parallelen Ressourcen und Realzeitanforderungen dar. In einem solchen System seien m Speicherressourcen gegeben und wiederum ein diskretes Zeitmodell. Da jede Speicherressource pro Zeitschritt ein Datenpaket in Einheitsgröße ausliefern kann, werden die Ressourcen mit $S := \{s_{j,i} \mid j \in \{1, \ldots, m\}, i \in \mathbb{N}\}$ beschrieben. Das Element $s_{j,i}$ repräsentiert dabei die Ressource j zum Zeitpunkt i.
Innerhalb des Datenservers seien kontinuierliche Medienströme (z. B. Videos) abgespeichert, die in Echtzeit an Konsumenten ausgeliefert werden sollen. Dazu ist die große Datenmenge eines Medienstromes in Pakete etwa gleicher Größe aufgeteilt. Um Konflikte beim Ressourcezugriff und damit einhergehende lokale Überlastungen zu verringern, wird jedes Datenpaket in c Kopien ($c \in \mathbb{N}, c \geqq 2$) auf c verschiedene Speicherressourcen abgelegt. Die Wirksamkeit dieser Maßnahme wurde in anderen Modellen mehrfach gezeigt (siehe z. B. [ABKU99]). Es sollen im DAP keine Annahmen gemacht werden, nach welchen Regeln die Datenpakete und ihre Kopien den Speicherressourcen zugeordnet werden.
Ein Konsument, welcher einen Medienstrom empfangen möchte, benötigt in regelmäßigen Zeitabständen das nächste Datenpaket. Daraus ergibt sich eine Abfolge von Anfragen nach Paketen. Für das DAP gilt, daß pro Zeitschritt maximal ein Paket benötigt wird. Für die korrekte Wiedergabe des kontinuierlichen Medienstromes unterliegt die Paketlieferung Echtzeitbedingungen. Motiviert durch die Existenz von Puffern, genügt es ein zum Zeitpunkt i bestelltes Paket erst zum Zeitpunkt $i + d - 1$ auszuliefern. Dabei stellt d ($d \in \mathbb{N}$) den zweiten wichtigen Modellparameter dar. Innerhalb der d Zeitschritte des Intervalls $[i, i+d-1]$ wird das Paket entgegengenommen. Danach ist es zu spät und wertlos, d. h. es muß auch nicht mehr ausgeliefert werden.
Die derzeit eingesetzten komprimierenden Datenformate für Medienströme (z. B. MPEG für Videodaten) verfügen zwar nicht über redundante Speicherung, um ein Ausbleiben einiger Datenanteile zu kompensieren, aber ein Fehlen eines kleinen Teiles der Daten scheint nur kleine und akzeptable Störungen auszulösen.
Da beim Konsumenten die Möglichkeit der Interaktion, d. h. des Unterbrechens und Springens im Medienstrom gegeben werden soll, ist nicht bekannt, welche Teile des Datenstromes demnächst angefordert werden. Um die Gesamtbandbreite des Systems nicht zu überlasten, wird die Anzahl der gleichzeitig zugelassenen Konsumenten auf m begrenzt.

Für das abstrakte Modell des DAP ergibt sich damit auf der Seite der Aufträge das folgende Bild. Pro Zeitschritt i werden bis zu m Aufträge $r_{j,i}$ an das System gestellt. Diese bilden die Menge $R := \{r_{j,i} \mid j \in \{1, \ldots, m\}, i \in \mathbb{N}\}$. Jeder Auftrag $r_{j,i}$ spezifiziert c Speicherressourcen $\mathcal{S}_{r_{j,i}}$, in denen das benötigte Datenpaket abgelegt ist. Es gilt $\mathcal{S}_{r_{j,i}} \in \{a \mid a \in \wp(\{1, \ldots, m\}) \wedge |a| = c\}$, wobei $\wp(A)$ die Potenzmenge der Menge A bezeichnet und $\mathcal{S}_{r_{j,i}}$ die c verschiedenen Indizes der Speicherressourcen beinhaltet, in denen das gewünschte Datenpaket enthalten ist. Das von $r_{j,i}$ angeforderte Datenpaket soll innerhalb des Zeitintervalls $[i, i + d - 1]$ ausgeliefert werden. Diese Aufgabe kann somit durch die Ressourcen $\{s_{p,q} \mid p \in \mathcal{S}_{r_{j,i}}, q \in \{i, \ldots, i + d - 1\}\}$ erfüllt werden.

Es ergibt sich wieder ein Modell, welches als bipartiter Graph $G = (R \dot{\cup} S, E)$ dargestellt werden kann. Dabei sind die Mengen R und S wie oben definiert und die Kantenmenge E wird unter Zuhilfenahme der Speicherstellen $\mathcal{S}_{r_{j,i}}$ gebildet:

$$E := \{\{r_{j,i}, s_{p,q}\} \mid r_{j,i} \in R, j \in \{1, \ldots, m\}, i \in \mathbb{N}, \\ s_{p,q} \in S, p \in \mathcal{S}_{r_{j,i}}, q \in \{i, \ldots, i+d-1\}\} \ .$$

Die Skizze 1 soll diese Situation verdeutlichen, wobei nur das „Kantenbüschel“ einer Anfrage $r_{j,i}$ eingezeichnet ist.



Skizze 1: Visualisierung der Struktur des DAP als bipartiter Graph; die Mengen S und R sind zweidimensional ausgelegt und das diskrete Zeitmodell verläuft von oben nach unten; beispielhaft ist für die Parameter $c = 3$ und $d = 5$ die Kantenmenge der Anfrage $r_{3,6}$ mit $\mathcal{S}_{r_{3,6}} = \{1, 2, m\}$ eingezeichnet.

Auch für dieses Modell definiert jedes Matching in dem bipartiten Graphen G einen gültigen Bearbeitungsplan. Da so viele Datenpakete wie möglich ausgeliefert werden sollen, ist das Optimierungsziel die Bestimmung eines Matchings maximaler Kardinalität. Für den im folgenden beschriebenen Online-Fall wird dann untersucht, wie gut dieses Ziel bei ungünstigsten Eingabestrukturen erreicht werden kann.
Das DAP in einer Online-Umgebung wird durch eine Menge von Ereignissen und ihre Abfolge definiert. Zu Beginn des Zeitschrittes i werden alle Anfragen $r_{j,i}$ mit $j \in \{1, \ldots, m\}$ als Eingabe offengelegt. Alle Anfragen $r_{j,i'}$ mit $i < i'$ bleiben unbekannt. Ein Online-Algorithmus muß dann über die Verwendung der Ressourcen $s_{j,i}$, $j \in \{1, \ldots, m\}$ entscheiden. Diese Teillösung wird dem globalen Online-Matching unwiderruflich hinzugefügt und der nächste Zeitschritt beginnt.

Auch das DAP ist ein Spezialfall des ORSM-Modells. Um diese Tatsache zu sehen, wird die zweidimensionalen Anfrage- bzw. Ressourcemengen in je eine Sequenz überführt, um so die Ordnung innerhalb des ORSM-Modells zu definieren. Dazu werden die Elemente $r_{j,i}$ lexikographisch nach $i \circ j$, also dem primären Schlüssel i und dem sekundären Schlüssel j, sortiert. Das gleiche geschieht mit der Ressourcemenge $S = \{s_{j,i} \mid j \in \{1, \ldots, m\}, i \in \mathbb{N}\}$ um die totale Ordnung $\prec$ des ORSM-Modells herzustellen. Nach dieser Transformation wird auch der Einsatz anderer oben genannter Konzepte deutlich. Die Eingabe von m Anfragen pro Zeitschritt entspricht einer Blockeingabe mit Blockgröße m. Da jede Anfrage durch genau $c \cdot d$ Ressourcen bearbeitet werden kann, ist das Konzept des konstanten Requestknotengrades implementiert. Durch den Parameter d wird auch die Kantenlänge begrenzt. Das Konzept der Intervalleingaben ist in Teilen wiederzuerkennen, denn jede Anfrage $r_{j,i}$ kann im gesamten Zeitintervall $[i, i + d - 1]$ bearbeitet werden. In der auf das ORSM-Modell transformierten Version entsteht jedoch statt einem Intervall ein Muster, welches sich d-mal im Abstand m wiederholt. Natürlich ist die Kantenstruktur des DAP mit diesen Konzepten nicht vollständig beschrieben. Das in [MBLR97] eingeführte Modell besitzt einen weiteren Parameter b ($b \in \mathbb{N}$), der die Bandweite der Ressourcen angibt. Jede Ressource aus S kann dort b Datenpakete pro Zeitschritt ausliefern und es werden bis zu bm Konsumenten zugelassen. Eine Modellvariante mit $b > 1$ kann durch Erhöhung des Parameters m mit dem hier definierten DAP simuliert werden. Auf Grund von mehr a priori Wissen über die Systemstruktur bzw. geringeren Abhängigkeiten im Datenserver könnten sich die Resultate etwas verbessern. Da sich jedoch die Prinzipien der Problemstellung, der Online-Algorithmen und ihrer Analyse nicht ändern, soll innerhalb dieser Arbeit auf den Parameter der Bandbreite verzichtet werden.

4.4 Abgrenzung zu Modellen aus der Literatur

Die meisten Arbeiten zu Online-Planungsproblemen beziehen sich auf die Optimierung von Funktionen in den Fertigstellungszeitpunkten der Aufträge, die alle bearbeitet werden müssen. Da das ORSM-Problem einen anderen Typ einer Zielfunktion hat, bleiben nur noch wenige Modelle der Literatur übrig, mit denen ein Vergleich sinnvoll erscheint. Die Auswertung der Literatur in den Kapiteln 3.1 und 3.2 zeigt dieses Faktum auf. Es verbleiben deshalb an dieser Stelle nur die Betrachtung einiger der in Kapitel 3.3 und 3.4 eingeführten Arbeiten. Da das ORSM-Problem ein Matchingproblem in einem bipartiten Graphen darstellt, soll mit einem Vergleich zu bekannten Online-Matchingproblemen begonnen werden.

Das ORSM-Modell basiert auf einer ähnlich einfachen Struktur wie das Problem des Online Bipartite Matchings aus [KVV90]. Dabei wird ebenfalls pro Zeitschritt ein neuer Knoten samt seinen inzidenten Kanten offengelegt. Während beim Online Bipartite Matching unter den soeben veröffentlichten Kanten vom Online-Algorithmus maximal eine Kante für das Matching ausgewählt werden muß, so geschieht diese Entscheidung im ORSM-Problem bezüglich der inzidenten Kan-

tenmenge eines Knotens der *anderen* Partition. Deshalb sind im ORSM-Modell auch beide Knotenpartitionen R und S mit einer totalen Ordnung bzw. mit dem Zeitmodell versehen, und es gibt die Einschränkung an die Kantenmenge (Aussage (12) auf Seite 68). Im ORSM-Problem findet somit in der Betrachtung ein permanenter Wechsel zwischen den Partitionen statt. Bezüglich der R-Partition erfolgt die Eingabe und daraufhin wird bezüglich eines Knotens der S-Partition eine Entscheidung getroffen.
Da zum Zeitpunkt i der Matchingentscheidung bezüglich des Knotens s_i alle seine inzidenten Kanten bekannt sind, kann das ORSM-Modell auch als einen Spezialfall des Online Bipartite Matchings aufgefaßt werden.[23] Dazu wird die S-Partition als die „unbekannte“ Partition betrachtet, bezüglich der die Kantenmengen offengelegt und über die Entscheidungen getroffen werden müssen. Das zusätzliche Wissen über weitere schon bekannte Kanten und die eingeschränkte Problemstruktur wird ignoriert. Allerdings ist eine Übertragung von Methoden und Resultaten des Online Bipartite Matchings auf das ORSM-Problem nicht sinnvoll, da selbst die untere Schranke für den Competitive Ratio randomisierter Online-Algorithmen gegen den blinden Gegenspieler höher liegt, als der Competitive Ratio eines deterministischen Online-Algorithmus für das ORSM-Problem.

Die geänderte Aufgabenstellung der Online-Entscheidung des ORSM-Problems führt ebenfalls dazu, daß die Erweiterung mit Gewichten (wORSM) keine zusätzliche Beschränkungen in der Struktur der bipartiten Graphen benötigt. Dies ist gleichfalls ein gravierender Unterschied zu den gewichteten Varianten des Online Bipartite Matchings.

Das Online-Matching-Problem in allgemeinen Graphen, welches unter dem Namen ‚Problem der Zimmerpartner‘ eingeführt wurde, bildet nicht die klare Aufteilung von Ressourcen und Aufträgen ab. Trotzdem kann die Struktur der Kantenmenge so eingeschränkt werden, daß jedes wORSM-Problem abgebildet bzw. simuliert werden kann. Dazu wird der Graph $G_{\mathrm{R}} = (V_{\mathrm{R}}, E_{\mathrm{R}}, w_{\mathrm{R}})$ und die totale Ordnung $\prec$ auf V_{R} wie folgt aus einem wORSM-Problem mit dem Graphen $G = (R \dot\cup S, E, w)$ konstruiert:

$$\begin{aligned} V_{\mathrm{R}} &:= R \cup S \ , \\ E_{\mathrm{R}} &:= E \ , \\ w_{\mathrm{R}} &:= w \ . \end{aligned}$$

Die durch die Knotenindizes gegebene Ordnung auf R und S wird so in die benötigte Ordnung $\prec$ umgewandelt, daß gilt:

$$r_1 \prec s_1 \prec r_2 \prec s_2 \prec r_3 \prec s_3 \prec r_4 \prec s_4 \prec \ldots$$

Die nachfolgenden Fakten zeigen, daß nach dieser Transformation des wORSM-Problems das Online-Problem der Zimmerpartner die Online-Ereignissequenz des

[23] Diese Aussage verdankt der Autor einem Hinweis von Yossi Azar.

wORSM-Modells simuliert. Sobald ein Requestknoten r_i aufgedeckt wird, kann das Zimmerpartnerproblem keine Matchingkante bestimmen, da für alle Nachbarknoten $s_j \quad i < j$ gilt, und all diese Serverknoten erst später als aktuelle Eingabeknoten auftreten. Dies gilt wegen der Einschränkung der Kantenmengenstruktur (Aussage (12)). Bei der Eingabe eines Serverknotens s_i in das Problem der Zimmerpartner sind, wie bei Bearbeitung von s_i im wORSM-Modell, alle inzidenten Kanten bekannt. Nun muß eine Entscheidung über die Aufnahme von s_i in das Matching getroffen werden. Da später keine neuen Nachbarknoten aufgedeckt werden — dafür ist wiederum Restriktion (12) verantwortlich — erhält ein Online-Algorithmus des Zimmerpartnerproblems später nicht noch einmal eine Möglichkeit eine zu s_i inzidente Kante in das Online-Matching aufzunehmen.
Die beiden Modelle sind jedoch keineswegs äquivalent, da das wORSM-Modell das Online-Problem der Zimmerpartner nicht simulieren kann. Deshalb lassen sich die Analyseergebnisse aus [BR93] nicht einfach auf den Spezialfall des wORSM-Problems übertragen. Einerseits müssen die allgemeingültigen unteren Schranken für den Competitive Ratio neu entwickelt werden, und zum anderen ist der Competitive Ratio für das wORSM-Problem besser, als für das Online-Problem der Zimmerpartner. Dieses Ergebnis resultiert offensichtlich aus der Tatsache, daß im wORSM-Modell mehr Wissen über die Graphenstruktur bekannt ist. Lediglich in den ungewichteten Varianten ist für beide Modelle der Competitive Ratio gleich. Für das ORSM-Problem wird jedoch innerhalb dieser Arbeit ein speziell zugeschnittener Online-Algorithmus einschließlich entsprechender Analyse vorgestellt. Die dazu entwickelten Argumente sind später auch auf die Modellvarianten des ORSM-Problems übertragbar.

Die weiteren in Kapitel 3.4 vorgestellten Online-Varianten von Matchingproblemen unterscheiden sich vom ORSM-Modell so grundlegend, daß ein Vergleich nicht möglich ist. Beim b-Matching werden mehrere Anfragen der selben Ressource zugeordnet. Nur durch diese Vereinfachung der Problemstruktur — es werden weniger kombinatorische Varianten und damit einhergehende Abhängigkeiten erzeugt — wird der Competitive Ratio verringert. Im Falle $b = 1$ entspricht sowohl das Modell, als auch seine Analyseresultate dem oben diskutierten Online Bipartite Matching.
Im Modell des Online Perfekten Matchings werden Neuzuordnungen innerhalb der schon bestimmten Lösung zugelassen, und die Zielfunktion ist ebenfalls völlig von der des ORSM-Problems verschieden.

Die in Kapitel 3.3.1 vorgestellten Modelle für Planungsprobleme in Realzeitsystemen mit Intervallaufträgen besitzen Aufträge verschiedener Länge (Ressourcebedarf). Zudem sind diese Aufträge unterbrechbar [FN95, Woe94] oder es ist nur eine einzige Ressource vorhanden [LT94]. In all diesen Fällen ist die Struktur der Probleme derart verschieden zum ORSM-Problem, daß eine Übertragung von algorithmischen Ideen, Analysetechniken oder Ergebnissen nicht möglich ist.
Auch die weiteren Modelle von Realzeitsystemen (Kapitel 3.3.2) verwenden Aufträge mit umfangreichen Parametern. Die für diese Modelle gezeigten konstanten unteren Schranken für den Competitive Ratio übersteigen jedoch deutlich die

Werte der Resultate für das ORSM-Problem. Einzig das in [BNGNS99] vorgestellte und motivierte Problem (siehe dazu Seite 57) soll noch einmal näher betrachtet werden. Dieses Modell beschreibt das DAP, falls es m identische Maschinen besitzt und für jeden Auftrag die folgenden Festlegungen gelten:

- die Differenz zwischen spätestem Fertigstellungs- und frühestem Startzeitpunkt beträgt d,
- die Bearbeitungszeit ist 1 für die c Maschinen, welche die Speicherressourcen $\mathcal{S}_{r_{j,i}}$ darstellen und $+\infty$ für alle anderen Maschinen und
- der Auftragswert ist 1 .

Die verschiedenen Zeitmodelle (kontinuierlich bzw. diskret im DAP) führen unter diesen Nebenbedingungen nicht zu unterschiedlichen Lösungen. Die in [BNGNS99] bestimmten Approximationsfaktoren, die z. T. durch einen Online-Greedy-Algorithmus gezeigt werden, liegen jedoch ebenfalls deutlich über den Werten des Competitive Ratios für das DAP.

5 Analyse des ungewichteten Modells

Übersicht: Die Analyse des einfachsten und abstraktesten Modells, des ungewichteten Online-Request-Server-Matching-Problems, ist Gegenstand dieses Kapitels. Der Competitive Ratio für dieses Modell wird dabei auf exakt 1.5 bestimmt. Die dazu notwendigen Untersuchungen beinhalten eine allgemeingültige untere Schranke für deterministische Online-Algorithmen und die Vorstellung sowie die Untersuchung des Online-Algorithmus LMM, für den eine passende obere Schranke gezeigt wird.

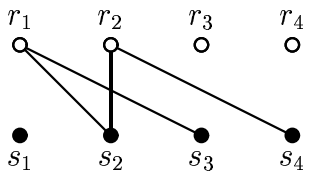
5.1 Die Untere Schranke

Zum Bestimmen der allgemeinen unteren Schranke für den Competitive Ratio des ORSM-Problems wird eine Gegenspielerstrategie entwickelt. Sie beginnt mit einer Eingabestruktur, die einem Online-Algorithmus zu einer Entscheidung zwischen strukturell verschiedenen Nachfolgekonfigurationen zwingt. Danach wird die Eingabe derart fortgesetzt, daß sich die getroffene Entscheidung als ungünstig herausstellt. Durch Wiederholung dieser Struktur ergibt sich die gewünschte Aussage.

Satz 3:

Jeder deterministische Online-Algorithmus für das ORSM-Problem besitzt einen Competitive Ratio von mindestens 1.5.

Beweis: Eine Gegenspielerstrategie beginnt mit der folgenden Eingabe: Zum Startzeitpunkt $i = 1$ werden die Kanten $\{r_1, s_2\}$ und $\{r_1, s_3\}$ präsentiert und im nächsten Zeitschritt $i + 1 = 2$ die Kanten $\{r_2, s_2\}$ und $\{r_2, s_4\}$.



Skizze 2: Situation zum Zeitpunkt $i + 1 = 2$

Ein deterministischer Online-Algorithmus ALG kann auf diese Situation im Zeitpunkt $i + 1 = 2$ mit drei verschiedenen Entscheidungen reagieren:

Fall 1 (s. Skizze 3, Teil a)**:** ALG nimmt die Kante $\{r_1, s_2\}$ in das Online-Matching M_{ALG} auf.

Danach präsentiert der Gegenspieler die Kante $\{r_3, s_4\}$; ALG kann nun den Serverknoten s_3 nicht nutzen und erreicht eine Matching-Größe $|M_{\mathsf{ALG}}| \leqq 2$, während die optimale Lösung $|M_{\mathsf{OPT}}| = 3$ beträgt.

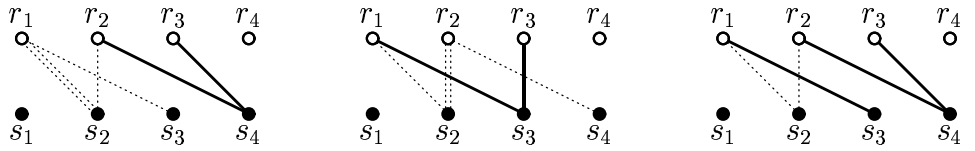
Fall 2 (s. Skizze 3, Teil b)**:** ALG nimmt die Kante $\{r_2, s_2\}$ in das Online-Matching M_{ALG} auf.

Danach präsentiert der Gegenspieler nur noch die Kante $\{r_3, s_3\}$; ALG kann nun den Serverknoten s_4 nicht nutzen und erreicht eine Matching-Größe $|M_{\mathsf{ALG}}| \leqq 2$, während die optimale Lösung $|M_{\mathsf{OPT}}| = 3$ beträgt.

Fall 3 (s. Skizze 3, Teil c)**:** ALG nimmt keine Kante und somit auch nicht den Knoten s_2 in das Online-Matching M_{ALG} auf.

Dann kann der Gegenspieler die Eingabe des Falles 1 präsentieren (die Eingabe des Falles 2 funktioniert ebenso). Da ALG den Serverknoten s_2 nicht benutzt, kann er nur noch ein Matching der Größe $|M_{\mathsf{ALG}}| \leqq 2$ erreichen,

während die optimale Lösung $|M_{\mathsf{OPT}}| = 3$ beträgt. Es zudem leicht ersichtlich, daß ein Nichtbenutzen der aktuellen Ressource in diesem Modell keinen Vorteil erbringen kann.



a: $\{r_1, s_2\} \in M_{\mathsf{ALG}}$ **b:** $\{r_2, s_2\} \in M_{\mathsf{ALG}}$ **c:** $M_{\mathsf{ALG}} = \varnothing$

Skizze 3: Die Teilskizzen a bis c zeigen die Konfigurationen zum Zeitschritt $i = 3$, nach den verschiedenen möglichen Entscheidungen eines Online-Algorithmus ALG, und den Reaktionen des Gegenspielers darauf.

Der Gegenspieler kann diese Strategie alle vier Zeitschritte wiederholen (zu allen Zeitpunkten $i \equiv 0 \mod 4$) und somit das Verhältnis zwischen der optimalen Lösung zur Online-Lösung von

$$\frac{|M_{\mathsf{OPT}}|}{|M_{\mathsf{ALG}}|} \geqq \frac{3}{2} = 1.5$$

zeigen. Da Eingabesequenzen beliebiger Länge konstruiert werden können, für die das Verhältnis der Lösungsqualitäten 1.5 nicht unterschreitet, folgt die Aussage des Satzes. ■ Satz 3

5.2 Der Algorithmus LMM

Der Online-Algorithmus, welcher im folgenden vorgestellt wird, löst das ORSM-Problem und gibt dabei eine bestmögliche Garantie für die Lösungsqualität. Er agiert im Zeitschritt i auf einem Teilgraphen B_i des zugrundeliegenden Graphen $G = (R \dot\cup S, E)$ der Probleminstanz. B_i enthält alle Teile der Eingabe, einschließlich der zu Knoten r_i inzidenten Kanten, abzüglich aller Kanten, die durch irreversible Entscheidungen der zuvor liegenden Zeitschritte schon fixiert wurden. Genauer gesagt ist B_i ein zeitlich lokaler Teilgraph, der durch die Knotenmenge V_i induziert wird:

$$V_i = \{r_k |\ r_k \in R, r_k \mathring{\notin} M_{\mathsf{ALG}}, 1 \leqq k \leqq i\} \ \dot\cup\ \{s_k |\ s_k \in S, k \geqq i\} \ .$$

Der Online-Algorithmus bestimmt auf B_i ein Matching $\mathcal{M}(B_i)$ maximaler Kardinalität. Er trägt deshalb den Namen LMM (local maximum matching). Aus Effizienz- und technischen Gründen wird die schon berechnete Lösung für B_{i-1} mit Hilfe des wohlbekannten Konzepts der erweiternden Wege [Edm65b] verbessert. Außerdem wird sichergestellt, daß der Knoten s_i der aktuellen Ressource benutzt wird, falls dies möglich ist.

Die Funktion des LMM-Algorithmus ist formal in folgendem Pseudocode gegeben:

```
1: loop {für alle Zeitschritte i}
2:    lies die Eingabe des Schrittes i und baue B_i auf
3:    bestimme ein Maximum-Matching M(B_i) in B_i :
          beginne mit allen Matchingkanten aus M(B_{i-1}),
          welche auch Kanten in B_i sind;
4:        suche einen erweiternden Weg vom Knoten r_i aus und
          führe bei Erfolg die Erweiterung durch
          {Wegen des Kardinalitätsmaximums von M(B_{i-1}),
          muß jeder erweiternde Weg r_i als Endknoten beinhalten.}
5:    if s_i ∈̊ M(B_i) then
6:       füge die Matchingkante von s_i dem Online-Matching M_LMM hinzu
             {d. h. {s_i, r} ∈ M(B_i) ⟹ {s_i, r} ∈ M_LMM}
7:    else if s_i ist in B_i nicht isoliert then {alle Nachbarn von s_i sind in M(B_i)}
8:       füge eine beliebige Kante {s_i, r} aus B_i dem Matching M_LMM hinzu und
         lösche die Matchingkante von r in M(B_i)
9:    end if
10: end loop
```

Algorithmus 1: LMM

Die Zeile 8 dieses Algorithmus ist essentiell und stellt die Nutzung von s_i sicher, falls dies zum Zeitpunkt i noch möglich ist.[24]

5.3 Die Obere Schranke

Um die Leistungsfähigkeit des Algorithmus LMM zu analysieren, werden zwei Beobachtungen benötigt.

Lemma 4:

Nachdem ein Requestknoten r_i im Teilgraph B_i in das Matching $\mathcal{M}(B_i)$ aufgenommen wurde ($r_i \mathring{\in} \mathcal{M}(B_i)$), bleibt er in allen folgenden lokalen Maximum-Matchings $\mathcal{M}(B_{i'})$ gebunden, d. h. Teil einer Matchingkannte, bis er mit einer solchen in das Online-Matching M_{LMM} aufgenommen wird.

Beweis: In Zeile 4 des LMM-Algorithmus wird r_i in das Matching $\mathcal{M}(B_i)$ aufgenommen, falls dies möglich ist. In den folgenden Zeitschritten i' ($i' \geqq i$) garantiert die Zeile 3 von LMM, daß die Matchingkante von r_i in das neue Matching von $B_{i'}$ übernommen wird, und in Zeile 4 wird zur Bestimmung von $\mathcal{M}(B_{i'})$ nur eine Erweiterung durchgeführt. Bei der Vergrößerung des Matchings über einen

[24] Würde der Algorithmus auf das bevorzugte Benutzen des aktuellen Serverknotens verzichten, so kann nur ein Competitive Ratio von 2 (und zwar als obere und untere Schranke) gezeigt werden. Dieser Wert ist nicht besser, als beim GREEDY-Algorithmus für das Online Bipartite Matching Problem und die Schranken werden ebenfalls wie in [KVV90] bewiesen.

erweiternden Weg bleiben jedoch alle gebundenen Knoten des Matchings weiterhin gebunden, also Endknoten von Matchingkanten. Damit überträgt sich diese Eigenschaft des Knotens r_i in alle lokalen Maximum-Matchings $\mathcal{M}(B_{i'})$ bis zu einem Zeitpunkt j ($i \leqq i' \leqq j$), in dem die aktuelle Matchingkante $\{r_i, s_j\} \in \mathcal{M}(B_j)$ zu dem Online-Matching M_{LMM} hinzugefügt wird (Zeile 6 von LMM) bzw. Zeile 8 von LMM die Kante $\{r_i, s_j\}$ zur Aufnahme in M_{LMM} auswählt. ■ Lemma 4

Lemma 5:

Falls s_i in B_i nicht isoliert ist, so gilt $s_i \mathring{\in} M_{\mathsf{LMM}}$.

Beweis: Die Zeilen 5 bis 8 in LMM garantieren, daß s_i in das Matching M_{LMM} aufgenommen wird, falls in B_i eine zu s_i inzidente Kante existiert. Die spezielle Behandlung von s_i in den Zeilen 7 und 8 sorgt für diese Tatsache auch dann, wenn der Knoten s_i nach der Bestimmung des lokalen Maximum-Matchings $\mathcal{M}(B_i)$ nicht im Matching enthalten ist. ■ Lemma 5

Unter Anwendung dieser beiden Lemmata kann der folgende Satz gezeigt werden.

Satz 6:

Der deterministische Online-Algorithmus LMM für das ORSM-Problem ist 1.5-competitive.

Beweis: Es wird gezeigt, daß ein fertiggestelltes Online-Matching M_{LMM} keine erweiternden Wege der Länge Eins oder Drei besitzt. Deshalb haben die kürzesten, erweiternden Wege, über die das Matching M_{LMM} vergrößert werden kann, eine Länge von Fünf. Bei einer Erweiterung solcher Wege werden jeweils zwei Matchingkanten der Online-Lösung in drei Matchingkanten einer optimalen Lösung transformiert. Diese Tatsache führt sofort zur Aussage des Satzes, da das Kantenverhältnis längerer, erweiternder Wege geringer ist und Matchingkanten in M_{LMM}, die nicht Teil erweiternder Wege sind, das Gesamtverhältnis ebenfalls vermindern. Durch vollständige Fallunterscheidung und Widerspruchsbeweise wird nun die Nichtexistenz erweiternder Wege der Länge Eins und Drei gezeigt.

Skizze 4: Die Struktur erweiternder Wege der Länge Eins und Drei.

Fall 1 Erweiternder Weg der Länge Eins, $\{s_i, r_a\} \notin M_{\mathsf{LMM}}$:

Der Knoten r_a wurde niemals Teil einer Matchingkante in einem lokalen Maximum-Matching, da $r_a \mathring{\notin} M_{\mathsf{LMM}}$ gilt (Umkehrung von Lemma 4). Damit ist aber die Kante $\{s_i, r_a\}$ in B_i und die Tatsache $s_i \mathring{\notin} M_{\mathsf{LMM}}$ widerspricht Lemma 5.

Fall 2 Erweiternder Weg der Länge Drei und $i < j$:

Der Knoten r_a wurde erst zum Zeitpunkt j in die Online-Lösung M_{LMM} aufgenommen und deshalb war die Kante $\{s_i, r_a\}$ in B_i vorhanden. Dann steht $s_i \mathring{\notin} M_{\mathsf{LMM}}$ im Widerspruch zu Lemma 5.

Fall 3 Erweiternder Weg der Länge Drei und $i > j$:

Aus $r_b \mathring{\notin} M_{\mathsf{LMM}}$ folgt unter Umkehrung des Lemmas 4, daß r_b niemals Teil eines lokalen Matchings war. Die Kante $\{r_a, s_j\}$ wird erst im Schritt j dem Matching M_{LMM} hinzugefügt. Deshalb sind die Requestknoten r_a und r_b im Schritt j nicht im Online-Matching M_{LMM} und somit ist der gesamte Weg $P := \{\{s_i, r_a\}, \{r_a, s_j\}, \{s_j, r_b\}\}$ Teil des lokalen Graphen B_j.

Die Entscheidung $\{r_a, s_j\} \in M_{\mathsf{LMM}}$ kann nicht durch Zeile 8 des LMM-Algorithmus getroffen worden sein. Dieser Spezialfall verlangt als Vorbedingung, daß $s_j \mathring{\notin} \mathcal{M}(B_j)$, aber dann sind s_j und r_b frei und $\{r_b, s_j\}$ müßte zur Wahrung der Optimalität in $\mathcal{M}(B_j)$ aufgenommen werden.

Der Weg P ist ein erweiternder Weg, und damit ein Widerspruch zur Optimalität von $\mathcal{M}(B_j)$, solange s_i frei ist. Damit läßt sich auf die Existenz eines Requestknotens r_c mit $\{r_c, s_i\} \in \mathcal{M}(B_j)$ schließen. Nur die spezielle Behandlung in Zeile 8 des LMM-Algorithmus kann einen vorher gebundenen Serverknoten (hier s_i) in einen freien Knoten ($s_i \mathring{\notin} M_{\mathsf{LMM}}$) überführen. Unter welchen Bedingungen kann dies passieren?

Zum Beispiel folgt aus dem Fakt $s_i \mathring{\notin} M_{\mathsf{LMM}}$, daß in einem Zeitschritt k mit $j < k < i$ die Matchingkante $\{r_c, s_i\}$ aus $\mathcal{M}(B_k)$ durch Zeile 8 von LMM entfernt und dafür $\{r_c, s_k\} \in M_{\mathsf{LMM}}$ gesetzt wurde. Dieses Verhalten setzt jedoch zum Zeitpunkt k einen freien Knoten s_k voraus, d. h. $s_k \mathring{\notin} \mathcal{M}(B_k)$. Es ergibt sich die Struktur eines alternierenden Weges der Länge Fünf:

Skizze 5: Alternierender Weg der Länge Fünf mit freiem Knoten r_b.

Nach Definition des ORSM-Problems (alle zu einem Requestknoten inzidenten Kanten werden gleichzeitig veröffentlicht) sind die *beiden* Kanten $\{r_c, s_i\}$ und $\{r_c, s_k\}$ schon zum Zeitpunkt j bekannt und deshalb ist der Weg $P^+ := \{\{s_k, r_c\}, \{r_c, s_i\}, \{s_i, r_a\}, \{r_a, s_j\}, \{s_j, r_b\}\}$ ein alternierender Weg in $\mathcal{M}(B_j)$. Falls s_k ein freier Knoten ist, so ist der Weg P^+ ein erweiternder Weg, was ein weiteres Mal im Widerspruch zur Optimalität von $\mathcal{M}(B_j)$ steht. Anderenfalls muß $s_k \mathring{\in} \mathcal{M}(B_j)$ gelten. Und nun kann die soeben geführte Argumentation über s_i, quasi rekursiv, auf s_k angewandt werden. Da B_j ein endlicher Graph ist, muß die wiederholte Anwendung der Argumentationsfolge in einem Widerspruch zur Existenz eines erweiternden Weges der Länge Drei in M_{LMM} enden.

Im allgemeinen wird der Teilgraph A betrachtet, der aus dem System aller alternierender Wege, die mit s_i beginnen und r_a nicht beinhalten, besteht. Innerhalb des Zeitintervalls $[j, i)$ verändert sich A dynamisch. Durch Einfügen eines Requestknotens r_x mit seinen inzidenten Kanten und der Erweiterung des lokalen Matchings zu $\mathcal{M}(B_x)$ kann sich auch A vergrößern. Dabei wird ein freier Knoten s_f von A gebunden. Z. B. gilt $\{r_x, s_f\} \in \mathcal{M}(B_x)$ oder allgemeiner, nach dem Einfügen von r_x ist s_f Endknoten eines erweiternden Weges in A. Andererseits können mehrere freie Knoten in der Nachbarschaft des gerade eingefügten Knotens r_x und damit auch in der vergrößerten Struktur A entstehen. Beim Entfernen einer Matchingkante $\{r_x, s\}$ aus A durch Überführung in das Online-Matching ($\{r_x, s\} \in M_{\mathsf{LMM}}$) wird die Struktur von A verkleinert. Da aber alle zu r_x und s inzidenten Kanten und möglicherweise Teilstrukturen von A mit entfernt werden, können durch diese Operation keine freien Knoten in A entstehen. Wird eine Matchingkante $\{r_x, s_x\}$ durch die Sonderbehandlung von Zeile 8 des Algorithmus LMM aus dem Teilgraphen A entfernt ($\{r_x, s_y\} \in M_{\mathsf{LMM}}$, s_y ist freier Knoten in A), so entsteht der neue freie Knoten s_x auf Kosten des zuvor in A vorhandenen freien Knotens s_y.

Aus dieser Analyse aller möglichen Operationen, die den Teilgraphen A der alternierenden Wege verändern, folgt, daß s_i nur dann als (letzter!) freier Knoten in A zurückbleibt, wenn permanent mindestens ein freier Serverknoten in A vorhanden war. Das gilt auch schon für den Zeitschritt j. Sei s_{free} ein solcher freier Knoten im Teilgraph A zum Zeitpunkt j. Dann ist der alternierende Weg von s_{free} über s_i bis zu r_a ein erweiternder Weg. Dies steht im Widerspruch zur Optimalität von $\mathcal{M}(B_j)$ und beendet damit den Beweis unter Anwendung der oben aufgeführten Argumentationskette.

■ Satz 6

Zusammen mit Satz 3 ergibt sich die scharfe Analyse des Online-Algorithmus LMM, der 1.5-competitive ist.

6 Analyse des Modells mit Gewichten

Übersicht: Nachdem im vorigen Kapitel das ORSM-Problem untersucht und sein Competitive Ratio exakt bestimmt wurde, folgt nun die Auseinandersetzung mit einer Generalisierung: dem gewichteten Online-Request-Server-Matching-Problem (wORSM). Ähnlich dem Kapitel 5 wird zuerst eine allgemeingültige untere Schranke gezeigt. Danach folgt die Darstellung und Diskussion des Online-Algorithmus wLMM. Er stellt eine natürliche Erweiterung des LMM-Algorithmus für die gewichtete Modellvariante dar. Leider ist die Leistungsfähigkeit des wLMM-Algorithmus schlechter, als die allgemeine untere Schranke für den Competitive Ratio zeigt. Deshalb wird eine spezielle Gegenspielerstrategie gegen wLMM, die eine untere Schranke für die Qualitätsgarantie der Lösung *dieses* Online-Algorithmus nachweist, angegeben. Den umfangreichsten Teil dieses Kapitels nimmt der Beweis des — zur speziellen unteren Schranke exakt passenden — Competitive Ratios von wLMM ein. Dazu ist eine Charakterisierung von erweiternden Wegen in gewichteten bipartiten Graphen unter einem eingeschränkten Satz von dynamischen Operationen auf diesen Graphen notwendig.
Trotz der exakten Analyse von wLMM verbleibt eine Lücke in den Resultaten des Competitive Ratios deterministischer Online-Algorithmen für das wORSM-Modell. Die Möglichkeiten und Grenzen der Verbesserung des Online-Algorithmus und der entwickelten Analysetechnik, einschließlich eines konkreten Vorschlages für einen neuen, leider noch nicht analysierten Online-Algorithmus (PHI), bilden den letzten Teil dieses Kapitels.

In diesem Kapitel spielt der *Goldene Schnitt* mehrfach eine Rolle. Er ist seit der Antike bekannt und stellt die *stetige Streckenteilung* dar, die als besonders harmonisch gilt. Wird eine Strecke $\overline{AB}$ durch einen Punkt C in einen größeren Abschnitt $\overline{AC}$ (Major) und einen kleineren Abschnitt $\overline{CB}$ (Minor) geteilt, und verhält sich der Major zum Minor ($\overline{AC} : \overline{CB}$) genauso, wie die Gesamtstrecke zum Major ($\overline{AB} : \overline{AC}$), so nennt man dieses Verhältnis Goldener Schnitt. Sein Wert läßt sich auch algebraisch angeben.

Definition 22: (Goldener Schnitt)

Der Goldene Schnitt ϕ ist definiert als

$$\phi := \frac{\sqrt{5}+1}{2} .$$

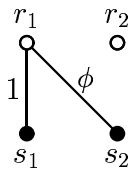
Dieser Term ist numerisch in etwa 1.618034 .

6.1 Die Allgemeine Untere Schranke

Satz 7:

Jeder deterministische Online-Algorithmus für das wORSM-Problem besitzt einen Competitive Ratio von mindestens $\phi = (\sqrt{5}+1)/2$.

Beweis: Die folgende Gegenspielerstrategie zeigt die untere Schranke.
Zum Startzeitpunkt $i = 1$ werden die Kanten $\{r_1, s_1\}$ und $\{r_1, s_2\}$ mit den Gewichten $w(\{r_1, s_1\}) = 1$ und $w(\{r_1, s_2\}) = \phi$ präsentiert.



Skizze 6: Situation zum Zeitpunkt $i = 1$

Ein deterministischer Online-Algorithmus ALG für das wORSM-Problem kann auf diese Situation im Zeitpunkt $i = 1$ mit zwei verschiedenen Entscheidungen reagieren:

Fall 1 (s. Skizze 7, Teil a)**:** ALG nimmt die Kante $\{r_1, s_1\}$ in das Online-Matching auf.

Danach präsentiert der Gegenspieler keine weitere zu s_2 inzidente Kante. Es folgt $|M_{\mathsf{ALG}}| = 1$ und $|M_{\mathsf{OPT}}| = \phi$.

Fall 2 (s. Skizze 7, Teil b)**:** ALG verzichtet auf die Aufnahme der Kante $\{r_1, s_1\}$ in das Online-Matching M_{ALG}.

Danach präsentiert der Gegenspieler die Kante $\{r_2, s_2\}$ mit dem Gewicht $w(\{r_2, s_2\}) = \phi$. Nun kann ALG höchstens noch ein Matching des Gewichtes $|M_{\mathsf{ALG}}| \leqq \phi$ erreichen, während $|M_{\mathsf{OPT}}| = 1 + \phi$ gilt. Das Verhältnis dieser Gewichte ist

$$\frac{|M_{\mathsf{OPT}}|}{|M_{\mathsf{ALG}}|} \;=\; \frac{1+\phi}{\phi} \;=\; \phi \;.$$

a: $\{r_1, s_2\} \in M_{\mathsf{ALG}}$ **b:** $M_{\mathsf{ALG}} = \varnothing$

Skizze 7: Die Teilskizzen a und b zeigen die Konfigurationen zum Zeitschritt $i = 2$, nach den beiden möglichen Entscheidungen eines Online-Algorithmus ALG, und den Reaktionen des Gegenspielers darauf.

Der Gegenspieler kann diese Strategie alle zwei Zeitschritte wiederholen und somit die untere Schranke im Competitive Ratio von $\phi = (\sqrt{5}+1)/2$ sicherstellen.

■ Satz 7

6.2 Der Algorithmus wLMM

Die Abkürzung wLMM steht für den Algorithmus „weighted local maximum matching“ (lokales maximum-*gewichtetes* Matching). Dieser Online-Algorithmus arbeitet auch sehr ähnlich dem Algorithmus LMM. Jedoch bestimmt wLMM im Zeitschritt i ein maximum-gewichtetes Matching ($\mathcal{M}(B_i)$) auf dem lokalen Graphen B_i. Unter dieser natürlichen und einfachen Anpassung des LMM-Algorithmus auf die gewichtete Modellerweiterung, macht eine erzwungene Bevorzugung des aktuellen Serverknotens s_i (wie in Zeile 8 von LMM) keinen Sinn. Ein Gegenspieler kann durch abziehen eines ε-großen Betrages von den Gewichten der zu s_i inzidenten Kanten die Auswahl einer anderen Kante erzwingen, ohne einen spürbaren Einfluß auf den Wert des Competitive Ratios auszuüben. Die mit dieser Beobachtung verbundenen Probleme und Ideen zu ihrer Überwindung werden im letzten Teil dieses Kapitels diskutiert.

Auf der nächsten Seite ist der Online-Algorithmus wLMM in Pseudocode formal dargestellt und die nächsten Teilkapitel beinhalten seine exakte Analyse.

1: **loop** {für alle Zeitschritte i}
2: lies die Eingabe des Schrittes i und baue B_i auf
3: bestimme ein maximum-gewichtetes Matching $\mathcal{M}(B_i)$ in B_i
4: **if** $s_i \mathring{\in} \mathcal{M}(B_i)$ **then**
5: füge die Matchingkante von s_i dem Online-Matching M_{wLMM} hinzu
{d. h. $\{s_i, r\} \in \mathcal{M}(B_i) \implies \{s_i, r\} \in M_{\mathsf{wLMM}}$}
6: **end if**
7: **end loop**

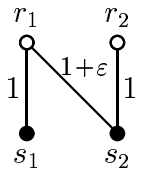
Algorithmus 2: wLMM

6.3 Die Untere Schranke von wLMM

Satz 8:

Der Online-Algorithmus wLMM besitzt einen Competitive Ratio von mindestens 2 .

Beweis: Die folgende Struktur für ein Eingabeproblem beweist die Aussage. Im Schritt $i = 1$ werden die Kanten $w(\{r_1, s_1\}) = 1$, $w(\{r_1, s_2\}) = 1 + \varepsilon$ ($\varepsilon \in \mathbb{R}_+$) und im nächsten Zeitschritt die Kante $w(\{r_2, s_2\}) = 1$ veröffentlicht.



Skizze 8: Eingabestruktur für die untere Schranke von wLMM.

Der Online-Algorithmus wLMM bestimmt für diese Eingabestruktur das Matching $M_{\mathsf{wLMM}} = \{\{r_1, s_2\}\}$, da im ersten Schritt die Kante $\{r_1, s_1\}$ nicht ausgewählt wird. Es besitzt das Gewicht $|M_{\mathsf{wLMM}}| = 1 + \varepsilon$. Die optimale Lösung ist hingegen $M_{\mathsf{OPT}} = \{\{r_1, s_1\}, \{r_2, s_2\}\}$ mit dem Gewicht $|M_{\mathsf{OPT}}| = 2$. Daraus ergibt sich das Verhältnis

$$\frac{|M_{\mathsf{OPT}}|}{|M_{\mathsf{wLMM}}|} = \frac{2}{1+\varepsilon}$$

und für ε gegen Null folgt der Grenzwert:

$$\lim_{\varepsilon \to 0} \frac{2}{1+\varepsilon} = 2 \ .$$

Da diese Struktur alle zwei Zeitschritte erneut als Eingabe fungieren kann, ist gezeigt, daß wLMM keinen besseren Competitive Ratio als 2 besitzt. ■ Satz 8

6.4 Der Competitive Ratio von wLMM

Die Analyse des Online-Algorithmus wLMM erfordert einige Vorarbeiten. Zuerst müssen einige neue Definitionen und Notationen eingeführt und häufig gebrauchte Definitionen sollen kurz wiederholt werden. Danach werden Eigenschaften von gewichteten, erweiternden Wegen in (bipartiten) Graphen unter einer Menge von dynamischen Graphmanipulationen herausgearbeitet. Diese Erkenntnisse bilden die Grundlage für Beweise zweier Hauptlemmata. Zum Schluß kann der eigentliche Satz unter Anwendung dieser Hauptlemmata mit recht knapper Argumentation gezeigt werden. Die in diesem Teil der Arbeit benutzten Notationen und Sprechweisen lehnen sich an den Aufsatz [BR93] an, in dem eine ähnliche Argumentationskette eingesetzt wird.

6.4.1 Definitionen und Notationen

Zum schnellen Auffinden folgen zuerst einige Wiederholungen von Definitionen.

$G = (R \dot\cup S, E, w)$ ist der gewichtete bipartite Graph des wORSM-Problems mit der Gewichtsfunktion $w: E \to \mathbb{R}_+$. Er wird als Eingabe online präsentiert.

M_i ist das Online-Matching, welches der Algorithmus wLMM bis zum Zeitschritt i bestimmt hat. Das heißt M_i ist der Teil des Online-Matchings M_{wLMM}, welcher nach vollständiger Bearbeitung des Zeitschrittes $i-1$ und vor der Eingabe des Schrittes i bestimmt wurde.

$B_i := G\big|_{R_i \dot\cup S_i}$ ist der lokale Graph des Zeitschrittes i. Dieser Graph ist der knoteninduzierte Teilgraph von G, welcher durch alle bekannten und nicht in M_i benutzten Requestknoten $R_i \subset R$, $r_j \in R_i \Rightarrow (j \leqq i \ \wedge \ r_j \mathring{\notin} M_i)$ und alle noch nicht entschiedenen Serverknoten $S_i \subset S$, $s_j \in S_i \Rightarrow i \leqq j$ gebildet wird. Somit enthält er die Kantenmenge $E_i = (R_i \times S_i) \cap E$.

$\mathcal{M}(B)$ bezeichnet ein maximum-gewichtetes Matching im Graphen B.

Damit können die folgenden Kurzschreibweisen eingeführt werden.

Definition 23: (Kurzschreibweisen, Teil 1)

(I) $m(B) := |\mathcal{M}(B)| = \sum_{\{r,s\} \in \mathcal{M}(B)} w(\{r,s\})$
ist das (eindeutige) Gesamtgewicht eines maximum-gewichteten Matchings im Graphen B.

(II) $B_{i \leftarrow v} := G\big|_{R_i \dot\cup S_i \dot\cup \{v\}}$, $v \in (R \dot\cup S) \setminus (R_i \dot\cup S_i)$
ist der um den Knoten v und seinen Kanten erweiterte Graph B_i.

(III) $B_i{}^{\rightarrow v} := G\big|_{R_i \dot\cup S_i \setminus \{v\}}$, $v \in (R_i \dot\cup S_i)$
ist der um den Knoten v verminderte Graph B_i.

Die beiden letzten Notationen können auch kombiniert auftreten. Für die Knotenbezeichner aus R und S werden indizierte Buchstaben r und s verwandt, so daß die Herkunftspartition der Knoten eindeutig sichtbar ist. Zusätzlich werden Listen von eingefügten oder gelöschten Knoten benutzt, falls dies notwendig ist. Eine typische Anwendung dieser Notation ist dann $B_i \overset{\rightarrow r_k, s_i}{\leftarrow r_{i+1}}$, welche den knoteninduzierten Teilgraphen $G\big|_{R_i \dot\cup S_i \dot\cup \{r_{i+1}\} \setminus \{r_k\} \setminus \{s_i\}}$, $r_{i+1} \in R \setminus R_i$, $r_k \in R_i$, $s_i \in S_i$ bezeichnet.

Sei $\mathcal{M}(B_i)$ ein fixiertes maximum-gewichtetes Matching von B_i und $\mathcal{M}(B_i{}^{\rightarrow s})$ ein solches Matching, welches nach dem Entfernen des Knotens s aus B_i unter minimaler Veränderung aus $\mathcal{M}(B_i)$ entsteht. Mit dieser Definition ergibt sich $\mathcal{M}(B_i{}^{\rightarrow s})$ aus $\mathcal{M}(B_i)$ auf $B_i{}^{\rightarrow s}$ durch das Erweitern maximal eines gewichteten, erweiternden Weges. Dieser erweiternde Weg ist auch Teil der symmetrischen Differenz $\mathcal{M}(B_i) \oplus \mathcal{M}(B_i{}^{\rightarrow s})$, welche einen Weg P bestimmt. P ist ein erweiternder Weg in B_i bezüglich des Matchings $\mathcal{M}(B_i{}^{\rightarrow s})$, und er ist möglicherweise leer. Die Notation P soll als Synonym für den Graphen $P = (V_P, E_P, w_P)$, bestehend aus den Kanten E_P des Weges P und der Menge V_P von inzidenten Knoten, gelten. Mit dieser Betrachtung wird der Sinn der folgenden Definitionen klar.

Definition 24: (Wert $\beta_i(s)$ eines Serverknotens)

Sei eine Probleminstanz $G = (R \dot\cup S, E, w)$ des wORSM-Problems und ein Zeitschritt i gegeben. Dann ist der Wert $\beta_i(s)$ eines Serverknotens $s \in S_i$ im Zeitschritt i definiert als

$$\beta_i(s) \;:=\; m(B_i) - m(B_i{}^{\rightarrow s}) \;.$$

Es wird also festgestellt, wie stark das maximum-gewichtete Matching im Graphen B_i absinkt, falls der Knoten s aus B_i entfernt wird. Somit stellt $\beta_i(s)$ eine Bewertung bzw. den Wert des Knotens s in B_i dar. Es ist sofort einsichtig, daß $\beta_i(s) = m(B_i) - m(B_i{}^{\rightarrow s}) = m(P) - m(P^{\rightarrow s})$ gilt, wenn die oben fixierten Matchings $\mathcal{M}(B_i)$, $\mathcal{M}(B_i{}^{\rightarrow s})$ und die Definition für P benutzt werden. Deshalb wird manchmal $\beta_i(s)$ auch das *Gewicht des Weges P* genannt, weil dieser Wert zum Gesamtgewicht des Matchings addiert wird, wenn der Weg P in $\mathcal{M}(P^{\rightarrow s})$ erweitert wird.

Nun kann, in Anlehnung an die Potentialfunktionstechnik aus Kapitel 2.2, eine globale Bewertungsfunktion definiert werden. Sie soll ebenfalls Potentialfunktion genannt werden, obwohl ihre Aufgabe etwas von der in der Potentialfunktionstechnik benutzten Definition abweicht.

Definition 25: (Potentialfunktion Φ_i)

Sei eine Probleminstanz G des wORSM-Problems und ein Zeitschritt i gegeben. Dann ist die Potentialfunktion Φ_i im Zeitschritt i definiert als

$$\Phi_i \;:=\; 2\,|M_i| + m(B_i) \;.$$

Diese Potentialfunktion bewertet das Gesamtgewicht des bisherigen Online-Matchings M_i doppelt und fügt den Wert des maximum-gewichteten Matchings des lokalen Graphen B_i hinzu. Der letzte Summand beschreibt das zusätzliche Matchinggewicht, welches garantiert noch erzielt werden kann.

Zum Schluß folgen einige weitere Kurzschreibweisen, die für einen Ausdruck f benutzt werden. Der Ausdruck f steht für eine der soeben definierten Notationen.

Definition 26: (Kurzschreibweisen, Teil 2)

Sei f ein Ausdruck, wie oben definiert. Dann gilt:

(I) $f_i^+ := f_{i+1}$

(II) $\Delta f_i := f_i^+ - f_i$

(III) $f_{|v}$ *ist der Wert des Ausdruckes* f_i *mit* $v \in \{r_i, s_i\}$.

(IV) $\Delta f_{|v}$ *ist die Wertedifferenz des Ausdruckes* f *vor und nach Bearbeitung des Knotens* v.

Der Index i wird für Ausdrücke immer weggelassen, wenn der Zeitschritt i aus dem Kontext eindeutig hervorgeht. Deshalb ergeben sich mit f^+ und Δf sinnvolle, abkürzende Notationen. In der Schreibweise $f_{|v}$ wird der Zeitschritt i für f_i durch den Zeitindex des angegebenen Knotens v definiert. In der Definition von $\Delta f_{|v}$ wird, abweichend vom eigentlich definierten Zeitmodell (Differenz bezüglich eines Zeitschrittes), die Veränderung des Ausdruckes f in einer zeitlich enger gefaßten Betrachtungsweise benutzt.

6.4.2 Grundlegende Eigenschaften von gewichteten, erweiternden Wegen, die durch $\beta(v)$ beschrieben werden

Im folgenden werden einige wichtige Eigenschaften von gewichteten Wegen P, die unter entsprechender Interpretation (s. o.) zu $\beta(v)$ korrespondieren, aufgelistet. Wird von einem durch $\beta(v)$ implizierten, *erweiternden Weg* P gesprochen, so ist damit gemeint, daß dieser Weg bezüglich $\mathcal{M}(P^{\rightarrow v})$ ein erweiternder Weg ist.
Innerhalb dieses Teilkapitels kann die Beschränkung auf bipartite Graphen fallengelassen werden. Alle bisher getroffenen Definitionen übertragen sich direkt auf einfache ungerichtete gewichtete Graphen.

Lemma 9:

Sei $G = (V, E, w)$ ein einfacher ungerichteter gewichteter Graph. Dann gilt:

$$\forall\, v \in V: \quad \beta(v) \;\geqq\; 0 \;.$$

Beweis: $G^{\to v}$ ist ein Teilgraph von G und sowohl $\mathcal{M}(G)$ als auch $\mathcal{M}(G^{\to v})$ sind optimal. Deshalb gilt $m(G) \geqq m(G^{\to v})$ und es folgt:

$$\beta(v) \;=\; m(G) - m(G^{\to v}) \;\geqq\; 0 \;.$$

■ Lemma 9

Lemma 10:

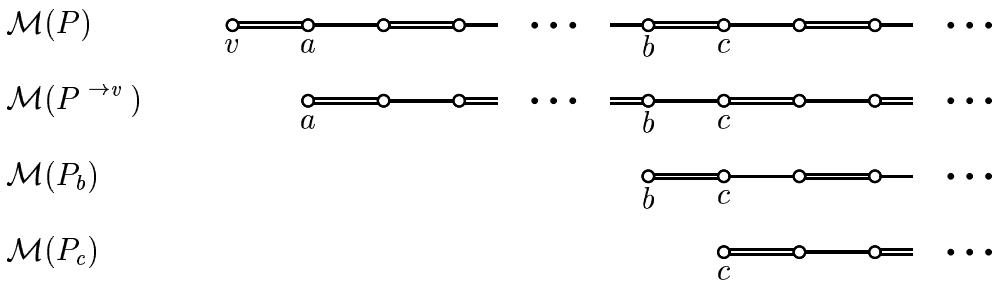
Sei $G = (V, E, w)$ ein einfacher ungerichteter gewichteter Graph und P der erweiternde Weg, der durch $\mathcal{M}(G) \oplus \mathcal{M}(G^{\to v})$ für ein festes $v \in V$ beschrieben wird.

(I) *Jeder Teilweg P_b, der aus P durch das Entfernen einer* geraden *Anzahl von Kanten ab dem Startknoten v entsteht und mit dem Knoten b beginnt, besitzt ein maximum-gewichtetes Matching $\mathcal{M}(P_b)$, welches eine Teilmenge des optimalen Matchings $\mathcal{M}(P)$ von P ist. Es gilt:*

$$\mathcal{M}(P_b) \;=\; \mathcal{M}(P) \cap P_b \;.$$

(II) *Jeder Teilweg P_c, der aus P durch das Entfernen einer* ungeraden *Anzahl von Kanten ab dem Startknoten v entsteht und mit dem Knoten c beginnt, besitzt ein maximum-gewichtetes Matching $\mathcal{M}(P_c)$, welches eine Teilmenge des optimalen Matchings $\mathcal{M}(P^{\to v})$ von P ist. Es gilt:*

$$\mathcal{M}(P_c) \;=\; \mathcal{M}(P^{\to v}) \cap P_c \;.$$



Skizze 9: Darstellung der Wege aus Lemma 10 mit deren optimalen Matchings

Beweis: Durch Widerspruch wird Teil (I) bewiesen:
Angenommen $\mathcal{M}(P) \cap P_b$ ist kein maximal-gewichtetes Matching für P_b:

$$|\mathcal{M}(P) \cap P_b| \;<\; |\mathcal{M}(P_b)| \;=\; m(P_b) \;. \qquad (13)$$

Die Matchingkante des Knotens b in $\mathcal{M}(P)$ ist Teil des Weges P_b, $\{b,c\} \in \mathcal{M}(P)$ und $\{b,c\} \in P_b$, da $\{v,a\} \in \mathcal{M}(P)$ und der Knoten b in einem Abstand von *gerader* Kantenanzahl vom Startknoten v entfernt liegt. Der Weg P kann am Knoten b in zwei Teilwege zerlegt werden und es gilt:

$$\mathcal{M}(P) \;=\; (\mathcal{M}(P) \setminus P_b) \,\dot\cup\, (\mathcal{M}(P) \cap P_b) \;.$$

Dann folgt aus der Annahme (13):

$$m(P) \;=\; |\mathcal{M}(P)| \;<\; |\mathcal{M}(P) \setminus P_b \,\dot\cup\, \mathcal{M}(P_b)| \;=\; |\mathcal{M}(P) \setminus P_b| + |\mathcal{M}(P_b)| \;,$$

ein Widerspruch zur Optimalität von $\mathcal{M}(P)$.

In gleicher Weise wird Teil (II) bewiesen, wobei über $P^{\rightarrow v}$ und P_c argumentiert wird. Dabei ist zu beachten, daß per Definition von P in $\mathcal{M}(P^{\rightarrow v})$ die Matching- und Nichtmatchingkanten bezüglich $\mathcal{M}(P)$ vertauscht sind. ■ Lemma 10

Lemma 11:

Seien der Weg P und die Teilwege P_b und P_c wie in Lemma 10 definiert. Dann kann der Wert $\beta(v)$ ab dem Knoten b bzw. c wie folgt durch $\beta(b)$ bzw. $\beta(c)$ ausgedrückt werden:

(I) $\beta(v) \;=\; m(P \setminus P_b) - m((P \setminus P_b)^{\rightarrow v}_{\leftarrow b}) \;+ \beta(b)$

(II) $\beta(v) \;=\; m((P \setminus P_c)_{\leftarrow c}) - m((P \setminus P_c)^{\rightarrow v}) \;- \beta(c)$

Beweis:

(I) Das Matching $\mathcal{M}(P)$ läßt sich nach Lemma 10, Teil (I) in $\mathcal{M}(P \setminus P_b)$ und $\mathcal{M}(P_b)$ zerlegen. Ebenso es ist nach Lemma 10, Teil (II) möglich $\mathcal{M}(P^{\rightarrow v})$ in $\mathcal{M}((P \setminus P_b)^{\rightarrow v}_{\leftarrow b})$ und $\mathcal{M}(P_b{}^{\rightarrow b})$ zu zerlegen. Damit gilt:

$$\begin{aligned}
\beta(v) &= m(P) - m(P^{\rightarrow v}) \\
&= m(P \setminus P_b) + m(P_b) - [m((P \setminus P_b)^{\rightarrow v}_{\leftarrow b}) + m(P_b{}^{\rightarrow b})] \\
&= m(P \setminus P_b) - m((P \setminus P_b)^{\rightarrow v}_{\leftarrow b}) \;+ \underbrace{m(P_b) - m(P_b{}^{\rightarrow b})}_{\beta(b)}
\end{aligned}$$

(II) Das Matching $\mathcal{M}(P)$ läßt sich nach Lemma 10, Teil (I) in $\mathcal{M}((P \setminus P_c)_{\leftarrow c})$ und $\mathcal{M}(P_c{}^{\rightarrow c})$ zerlegen. Ebenso es ist nach Lemma 10, Teil (II) möglich $\mathcal{M}(P^{\rightarrow v})$ in $\mathcal{M}((P \setminus P_c)^{\rightarrow v})$ und $\mathcal{M}(P_c)$ zu zerlegen. Damit gilt:

$$\begin{aligned}
\beta(v) &= m(P) - m(P^{\rightarrow v}) \\
&= m((P \setminus P_c)_{\leftarrow c}) + m(P_c{}^{\rightarrow c}) - [m((P \setminus P_c)^{\rightarrow v}) + m(P_c)] \\
&= m((P \setminus P_c)_{\leftarrow c}) + m((P \setminus P_c)^{\rightarrow v}) \;- \underbrace{[m(P_c) - m(P_c{}^{\rightarrow c})]}_{\beta(c)}
\end{aligned}$$

■ Lemma 11

Lemma 12:

$$\forall \{v,u\} \in \mathcal{M}(G): \quad w(\{v,u\}) \;\geqq\; \beta(v) \;.$$

Beweis: Folgt direkt aus dem Lemma 11, Teil (II), wobei $\beta(v)$ einen nichtleeren, erweiternden Weg P impliziert und P_c mit dem Knoten u beginnt. Dann ist $m((P \setminus P_u)_{\leftarrow u}) = w(\{v,u\})$, und $(P \setminus P_u)^{\rightarrow v}$ stellt den leeren Graph mit $m((P \setminus P_u)^{\rightarrow v}) = 0$ dar. Damit gilt:

$$\beta(v) \;=\; w(\{v,u\}) - \beta(u)$$

und unter Anwendung des Lemmas 9 ($\beta(u) \geqq 0$) folgt die Aussage.

■ Lemma 12

Lemma 13:

Sei $G = (V,E,w)$ ein einfacher ungerichteter gewichteter Graph und $v \in V$. P sei der durch $\beta(v)$ implizit definierte, erweiternde Weg von gerader *Länge. Wird der Weg P durch eine dynamische Veränderung am Graphen G verlängert, so verringert sich der Wert von $\beta(v)$ niemals, d. h.:*

$$\Delta\beta(v) \;\geqq\; 0 \;.$$

Beweis: Sei b der letzte Knoten des Weges P und P_b die Verlängerung von P. Da P von gerader Länge ist, kann das Lemma 11, Teil (I) angewandt werden. Damit gilt:

$$\begin{aligned}
\beta^+(v) &= m(P^+ \setminus P_b) - m(P^+ \setminus P_b{}^{\rightarrow v}_{\leftarrow b}) + \beta^+(b) \\
&= m(P) - m(P^{\rightarrow v}) + \beta^+(b) \\
&= \beta(v) + \beta^+(b)
\end{aligned}$$

sowie

$$\begin{aligned}
\Delta\beta(v) &= \beta^+(v) - \beta(v) \\
&= \beta(v) + \beta^+(b) - \beta(v) \\
&= \beta^+(b)
\end{aligned}$$

und unter Nutzung des Lemmas 9 folgt:

$$\Delta\beta(v) \;=\; \beta^+(b) \;\geqq\; 0 \;.$$

■ Lemma 13

Lemma 14:

Sei $G = (V, E, w)$ ein einfacher ungerichteter gewichteter Graph und ein Knoten $v \in V$ gegeben. P sei der durch $\beta(v)$ implizit definierte, erweiternde Weg. Wird der Weg P durch eine dynamische Veränderung am Graphen G so verkürzt, daß der zu $\beta^+(v)$ korrespondierende Weg P^+ ungerade *Länge besitzt, so verringert sich der Wert von $\beta(v)$ niemals, d. h.:*

$$\Delta\beta(v) \geqq 0 \ .$$

Beweis: Sei c der letzte Knoten des verkürzten Weges P^+ und $P_c{}^{\rightarrow c}$ die Verkürzung von P (d. h. $P = P^+ \dot{\cup} P_c{}^{\rightarrow c}$). Da P^+ von ungerader Länge ist, kann das Lemma 11, Teil (II) angewandt werden. Damit gilt:

$$\begin{aligned} \beta(v) &= m((P \setminus P_c)_{\leftarrow c}) - m((P \setminus P_c)^{\rightarrow v}) - \beta(c) \\ &= m(P^+) - m(P^{+\,\rightarrow v}) - \beta(c) \\ &= \beta^+(v) - \beta(c) \end{aligned}$$

sowie

$$\begin{aligned} \Delta\beta(v) &= \beta^+(v) - \beta(v) \\ &= \beta(v) + \beta(c) - \beta(v) \\ &= \beta(c) \end{aligned}$$

und unter Nutzung des Lemmas 9 folgt:

$$\Delta\beta(v) = \beta(c) \geqq 0 \ .$$

■ Lemma 14

Die einfachen Aussagen der Lemmata 9 bis 14 ermöglichen nun den Beweis zweier wichtiger Hilfssätze. Dabei erhalten diese Lemmata die gleichen Namen wie in [BR93], da sie später die gleichen Funktionen übernehmen, wie in den Beweisen am angegebenen Ort.

6.4.3 Die Hauptlemmata

Das erste Hauptlemma garantiert für jeden Serverknoten $s \in S$, daß sich sein Wert $\beta(s)$ niemals durch die Entscheidungen von wLMM und die vom wORSM-Problem vorgegebenen Veränderungen am lokalen Graphen B verringert. Dies gilt für $s_i \in S$ natürlich nur nur bis zum Zeitschritt i, da der Ausdruck $\beta(s)$ später nicht mehr definiert ist.

Lemma 15: (Stabilitätslemma)

Während der Ausführung von wLMM vermindert sich der Wert eines Serverknotens niemals, d. h.:

$$\forall\, s \in S_i :\ \forall\, j < i :\quad \Delta\beta_j(s) \ \geqq\ 0\ .$$

Beweis: Während des Ausführens von wLMM können die folgenden Veränderungen am lokalen Graphen B auftreten:

Fall 1 „Zuordnung“: Bei der Entscheidung über den aktuellen Serverknoten $s_i \in S$ wird die Matchingkante $\{r, s_i\} \in \mathcal{M}(B_i)$ in das Online-Matching M_{wLMM} aufgenommen und zusammen mit ihren Endknoten und allen adjazenten Kanten aus B_i entfernt:

$$B_i^+ \ = \ B_i \,^{\rightarrow r, s_i}\ .$$

Fall 2 „Nichtbenutzung“: Der aktuelle Serverknoten s_i wird durch wLMM nicht in $\mathcal{M}(B_i)$ und damit auch nicht in M_{wLMM} aufgenommen ($s_i \mathring{\notin} \mathcal{M}(B_i)$ und $s_i \mathring{\notin} M_{\mathsf{wLMM}}$), sondern nur samt seinen Kanten aus B_i entfernt:

$$B_i^+ \ = \ B_i \,^{\rightarrow s_i}\ .$$

Dieser Fall kann auftreten, wenn die Gewichte der zu s_i inzidenten Kanten zu klein sind.

Fall 3 „Eingabe“: Ein neuer Anfrageknoten $r_i \in R$ wird in den lokalen Graphen aufgenommen:

$$\begin{aligned} B_i &= B_{i-1} \,_{\leftarrow r_i} \qquad \text{bzw. verkürzt} \\ B^+ &= B \,_{\leftarrow r_i}\ . \end{aligned}$$

Es wird nun gezeigt, daß unter diesen drei Manipulationen am lokalen Graphen B die $\beta(s)$-Werte ($s \in S_i$) stabil bleiben. Für ein beliebiges, fest gewähltes $s \in S_i$ sei der durch $\beta(s)$ implizit beschriebene, erweiternde Weg mit Q bezeichnet. Solange dieser Weg Q durch die dynamischen Veränderungen an B nicht betroffen ist, gilt $\beta^+(s) = \beta(s)$ und trivialerweise $\Delta\beta(s) \geqq 0$. Deshalb ist es ausreichend eine Veränderung in der Struktur des Weges Q zu untersuchen.
Um die Notationen einfach zu halten wird in den folgenden Untersuchungen ohne Beschränkung der Allgemeinheit angenommen, daß pro Zeitschritt nur eine Manipulation am Graphen B auftritt. Eine Kombination des Falles 3 mit dem Fall 1 oder 2 im selben Zeitschritt führt zu keiner Änderung an der Stabilitätsaussage.

Fall 1 ($\{r, s_i\} \in \mathcal{M}(B_i)$)**:** Der Weg Q wird durch die Entnahme einer seiner Matchingkanten, einschließlich der dazu adjazenten Nichtmatchingkanten, verkürzt. Es verbleibt ein erweiternder Weg Q^+ ungerader Länge, denn er besitzt in $\mathcal{M}(B_i)$ an beiden Enden eine Matchingkante. Die Anwendung von Lemma 14 ergibt $\Delta\beta(s) \geqq 0$.

Fall 2 ($s_i \mathring{\notin} \mathcal{M}(B_i)$)**:** Der Weg Q kann nur durch Entfernen seines letzten Knotens (s_i) verkürzt werden, denn alle anderen Knoten in Q sind im Matching $\mathcal{M}(B_i)$. Wiederum besitzt der verbleibende Weg Q^+ an beiden Enden eine Matchingkante und ist deshalb von ungerader Länge. Somit folgt aus Lemma 14 die Aussage $\Delta\beta(s) \geqq 0$.

Fall 3 (Einfügen von $r_i \in R$)**:** Als erstes wird die Möglichkeit untersucht, daß der neue Requestknoten r_i nicht Teil des lokalen, maximum-gewichteten Matchings wird, $r_i \mathring{\notin} \mathcal{M}(B^+)$. Da r_i frei bleibt, könnte er nur zu einem freien Endknoten eines Weges Q^+ werden, der dann von gerader Länge sein muß. Da Q per Definition mit einem Serverknoten $s \in S_i$ beginnt, liegen alle Requestknoten von Q in ungeradem Abstand vom Startknoten s. Aus diesem Widerspruch folgt, daß sich die Struktur eines Weges Q bei $r_i \mathring{\notin} \mathcal{M}(B^+)$ nicht verändert und $\forall\, s \in S_i\, :\, \Delta\beta(s) = 0$ gilt.

Somit darf angenommen werden, daß r_i zu einem Serverknoten s_j im vergrößerten Graphen $B_{\leftarrow r_i}$ gebunden ist, $\{r_i, s_j\} \in \mathcal{M}(B_{\leftarrow r_i})$. Dann wurde ein Weg P erweitert, der exakt durch $\mathcal{M}(B_{\leftarrow r_i}) \oplus \mathcal{M}(B)$ beschrieben ist. Das Gewicht dieses Weges P, der mit r_i beginnt, soll wie der Wert eines Serverknotens definiert werden[25]:

$$\beta'(r_i) \;:=\; m(B_{\leftarrow r_i}) - m(B) \;>\; 0\ .$$

Der Knoten s_j kann zuvor schon im Matching $\mathcal{M}(B)$ gebunden sein oder auch nicht. Dafür müssen zwei weitere Fälle unterschieden werden.

Fall 3.a ($s_j \mathring{\notin} \mathcal{M}(B)$)**:** Durch die Kante $\{r_i, s_j\} \in \mathcal{M}(B_{\leftarrow r_i})$ kann kein erweiternder Weg Q ungerader Länge verlängert worden sein, da derartige Wege zwei gebundene Endknoten besitzen (Widerspruch zur Annahme des Falles 3.a). Es verbleibt nur die Variante, daß ein Weg Q gerader Länge durch $\{r_i, s_j\}$ verlängert wurde. Mit Lemma 13 folgt dann $\Delta\beta(s) \geqq 0$.

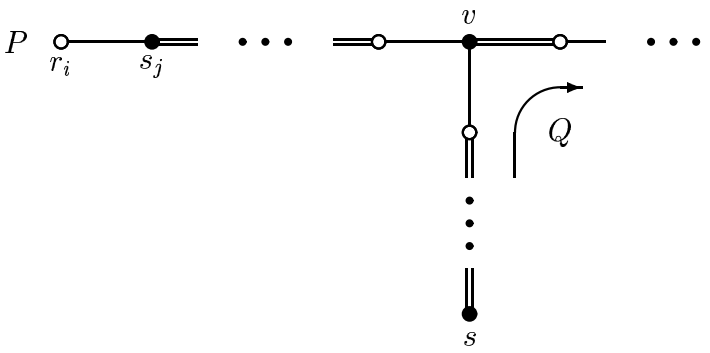
Fall 3.b ($s_j \mathring{\in} \mathcal{M}(B)$)**:** Aus den Voraussetzungen dieses Unterfalles folgt:

- P ist ein erweiternder Weg mit Länge echt größer Eins, da der adjazente Knoten s_j schon zuvor gebunden war.
- Es existiert ein Knoten v, welcher der erste gemeinsame Knoten der Wege P und Q ist. An diesem Knoten treffen sich die Wege zum ersten Mal bezüglich ihrer Startknoten r_i bzw. s.

[25] Beachte, daß der Ausdruck $\beta(s)$ nur für Serverknoten $s \in S$ definiert ist.

Aus diesen Tatsachen läßt sich schlußfolgern:

- $v \in S_i$, da anderenfalls v in $\mathcal{M}(B)$ mit einem Knoten in Q, der nicht auf P liegt *und* mit einem weiteren Knoten in P verbunden wäre. Das widerspricht der Eigenschaft eines Matchings.



Skizze 10: Situation vor der Erweiterung von P.

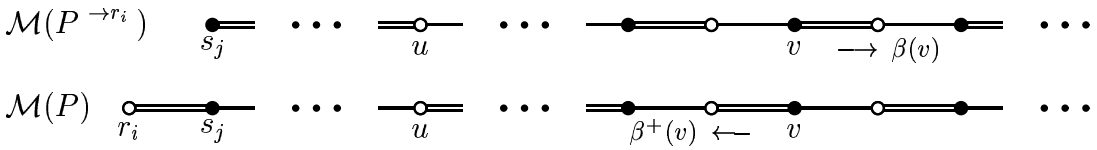
- Ohne Beschränkung der Allgemeinheit verlaufen die Wege P und Q ab dem Knoten v gemeinsam bis zum Ende, da nach Lemma 11, Teil (I) der Wert von $\beta(v)$ in beiden Wegen identisch ist.

  Bemerkung: Der Wert $\beta(v)$ ist optimal und für beide Wege P und Q gleich. Trotzdem gibt es die Möglichkeit, daß die mit v beginnenden Endstücke P_{tail} und Q_{tail} von P und Q verschieden sind. Dann lassen sich jedoch maximum-gewichtete Matchings finden, in denen Q_{tail} der Teil von P ab dem Knoten v, also P_{tail} ist. Es ist offensichtlich, daß diese Matchings existieren und damit geeignete Definitionen für P, $\beta(s_j)$ und $\beta'(r_i)$.

Das Lemma 11, Teil (I) erlaubt es, den Wert von $\beta(s)$ mit $\beta(v)$ auszudrücken. Da zwischen den Knoten s und v des Weges Q keine Matchingkanten verändert werden (siehe auch Skizze 10), gilt:

$$\Delta\beta(s) \;=\; \Delta\beta(v) \;.$$

Damit ist es ausreichend $\Delta\beta(v)$ zu untersuchen. Vor der Matchingerweiterung mittels des erweiternden Weges P bildet der Knoten v mit dem in P bezüglich r_i entfernteren Requestknoten eine Matchingkante. Beim Erweitern von P werden alle Matching- und Nichtmatchingkanten vertauscht. Danach befindet sich der Knoten v in einer Matchingkante mit dem Requestknoten, der sich in P näher an r_i befindet. Desweiteren hat sich der zu $\beta^+(v)$ korrespondierende, erweiternde Weg verändert. Er läuft jetzt auf dem Weg P in der „entgegengesetzten Richtung“, d. h. er startet mit dem Knoten v und endet bei einem Knoten u, der sich auf P zwischen v und r_i befindet (siehe Skizze 11).
Dieser Knoten u muß ein Requestknoten sein ($u \in R_i$), da sonst seine Matchingkante in $\mathcal{M}(P)$ außerhalb des zu $\beta^+(v)$ korrespondierenden Weges zwischen v und u liegen würde. Der Knoten u kann auch mit r_i zusammenfallen.



Skizze 11: Strukturelle Veränderung von $\beta(v)$ durch die Erweiterung des Weges P.

Aus der Definition von $\Delta\beta(v)$ ergibt sich:

$$\begin{aligned}\Delta\beta(v) &= \beta^+(v) - \beta(v) \\ &= m(P) - m(P^{\to v}) - [m(P^{\to r_i}) - m(P^{\to r_i,v})] \\ &= \underbrace{m(P) - m(P^{\to r_i})}_{=\beta'(r_i)} - [m(P^{\to v}) - m(P^{\to r_i,v})] \qquad (14)\end{aligned}$$

Zwischenbehauptung:

$$(P^{\to v}) - m(P^{\to r_i,v}) \;=\; \beta'(r_i) - \beta'(u)\ .$$

Mit dieser Zwischenbehauptung ergibt sich nach Gleichung (14) und dem Lemma 9 ($\beta'(u) \geqq 0$):

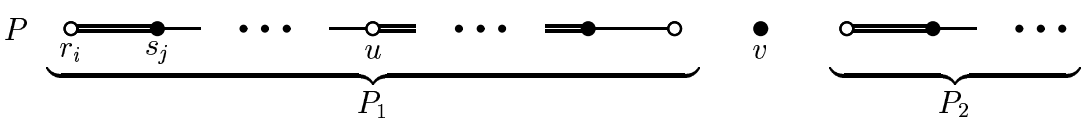
$$\Delta\beta(v) \;=\; \beta'(r_i) - [\beta'(r_i) - \beta'(u)] \;=\; \beta'(u) \;\geqq\; 0\ ,$$

woraus die behauptete Stabilität folgt. Das Lemma 9 kann hier benutzt werden, da es für den Wert $\beta(\cdot)$ eines beliebigen Knotens in einem allgemeinen gewichteten Graphen gilt.

Beweis der Zwischenbehauptung:

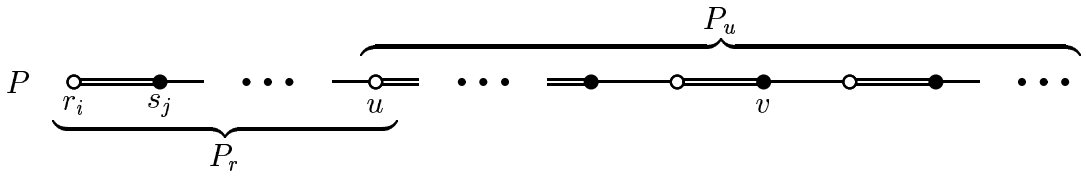
$$(P^{\to v}) - m(P^{\to r_i,v}) \;=\; \beta'(r_i) - \beta'(u)\ .$$

In der linken Gleichungsseite ist der Knoten v in beiden Termen aus dem Weg P entfernt. Deshalb ist P in zwei Teilwege $P_1 \dot\cup \{v\} \dot\cup P_2$ aufgespaltet, wobei sich r_i in P_1 befindet:



Das maximum-gewichtete Matching von P_2 ist in $\mathcal{M}(P^{\to v})$ und in $\mathcal{M}(P^{\to r_i,v})$ identisch und hat deshalb auf die Differenz $m(P^{\to v}) - m(P^{\to r_i,v})$ keinen Einfluß.

Sei P_r der Teilweg von P zwischen r_i und u und sei P_u der Teilweg von P zwischen u und dem Ende von P:



Nach Definition von $\beta'(r_i)$ und Lemma 10 gilt damit:

$$\begin{aligned}\beta'(r_i) &= m(P) - m(P^{\to r_i}) \\ &= m(P_r) + m(P_u) - [m(P_r{}^{\to r_i}) + m(P_u{}^{\to u})] \\ &= \underbrace{m(P_r) - m(P_r{}^{\to r_i})}_{m(P^{\to v}) - m(P^{\to r_i,v})} + \underbrace{m(P_u) - m(P_u{}^{\to u})}_{\beta'(u)}\end{aligned}$$

Um die Aussage $m(P_r) - m(P_r{}^{\to r_i}) = m(P^{\to v}) - m(P^{\to r_i,v})$ zu zeigen, genügt es P_1 zu betrachten. Durch die Erweiterung des Weges P (also dem Übergang von $\mathcal{M}(P^{\to r_i})$ zu $\mathcal{M}(P)$) wurden alle Matching- und Nichtmatchingkanten in P vertauscht. Nach dem Entfernen des Knotens v wird das Matching von P_1 vergrößert, indem alle Matching- und Nichtmatchingkanten im Teilweg zwischen v und u, der durch $\beta^+(v)$ beschrieben ist, ein weiteres Mal vertauscht werden. Deshalb liegt in diesem Teilweg die gleiche Matchingsituation wie vor dem Einfügen des Knotens r_i vor. Die Differenz zwischen $\mathcal{M}(P_1)$ und $\mathcal{M}(P_1{}^{\to r_i})$ kann nur im Teilweg P_r vorgefunden werden (genauer gesagt $\mathcal{M}(P_1) \oplus \mathcal{M}(P_1{}^{\to r_i}) = P_r$).
Und damit folgt aus den obigen Gleichungen:

$$\begin{aligned}\beta'(r_i) &= m(P^{\to v}) - m(P^{\to r_i,v}) + \beta'(u) \\ \Leftrightarrow \quad m(P^{\to v}) - m(P^{\to r_i,v}) &= \beta'(r_i) - \beta'(u)\ .\end{aligned}$$

■ Lemma 15

Das zweite Hauptlemma betrachtet die Veränderung der Potentialfunktion beim Behandeln von Server- und Requestknoten. Der Zuwachs im Potentialfunktionswert kann dabei für jeden Knotentyp von unten abgeschätzt werden.

Lemma 16: (Auszahlungslemma)

Während der Ausführung von wLMM gilt:

(I) $\Delta\Phi_{|s} \geqq \beta(s)_{|s}$, $s \in S$

(II) $\Delta\Phi_{|r} \geqq w(\{r,s\}) - \beta(s)_{|r} \qquad \forall\, s:\ \{r,s\} \in E$, $r \in R$

Bevor das Lemma bewiesen wird, sollen die Formeln kurz interpretiert werden:

(I) Die Potentialfunktion wächst bei der Entscheidung über den Serverknoten s_i um mindestens seinen aktuellen Wert $\beta_i(s_i)$.

(II) Wenn der Requestknoten r_i in den lokalen Graphen B_i eingefügt wird, stellt die Aussage (II) sicher, daß der schwerste, erweiternde Weg ab r_i zur Bestimmung des maximum-gewichteten Matchings $\mathcal{M}(B_i)$ erweitert wird. Die linke Seite der Ungleichung beschreibt das Gewicht des ausgewählten, erweiternden Weges (der möglicherweise leer ist) und auf der rechten Ungleichungsseite sind die Gewichte aller möglichen erweiternden Wege mit Startknoten r_i beschrieben (siehe dazu auch das Lemma 11, Teil (II)).

Beweis:
Zur Vereinfachung wird in allen Notationen auf den Index i verzichtet.

Teil (I):

Fall 1: Der Serverknoten s wurde in das Online-Matching aufgenommen, d. h. $s \mathring{\in} M_{\mathsf{wLMM}}$. Dann läßt sich über den Schritt i folgern:

$$\begin{aligned} M^+ &= M \cup \{r,s\} \\ \Rightarrow\ |M^+| &= |M| + w(\{r,s\}) \end{aligned}$$

und

$$\begin{aligned} B^+ &= B^{\rightarrow r,s} \\ \Rightarrow\ m(B^+) &= m(B) - w(\{r,s\})\ , \end{aligned}$$

eingesetzt in die Definition von $\Delta\Phi_{|s}$ ergibt sich:

$$\begin{aligned} \Delta\Phi_{|s} &= 2\,|M^+| + m(B^+) - [2\,|M| + m(B)] \\ &= 2\,|M| + 2w(\{r,s\}) + m(B) - w(\{r,s\}) - 2\,|M| - m(B) \\ &= w(\{r,s\}) \end{aligned}$$

und mit Lemma 12 folgt:

$$\Delta\Phi_{|s} = w(\{r,s\}) \geqq \beta(s)_{|s}\ .$$

Fall 2: Der Serverknoten s wurde nicht in das Online-Matching aufgenommen, $s \mathring{\notin} M_{\mathsf{wLMM}}$. Dann läßt sich über den Schritt i folgern:

$$\begin{aligned} M^+ &= M \\ \Rightarrow\ |M^+| &= |M| \end{aligned}$$

und

$$\begin{aligned} B^+ &= B^{\rightarrow s} \\ \Rightarrow\ m(B^+) &= m(B)\ , \end{aligned}$$

eingesetzt in die Definition von $\Delta\Phi_{|s}$ ergibt sich:

$$\begin{aligned}\Delta\Phi_{|s} &= 2\,|M^+| + m(B^+) - [2\,|M| + m(B)] \\ &= 2\,|M| + m(B) - 2\,|M| - m(B) \\ &= 0\ .\end{aligned}$$

Da $s \mathring{\notin} \mathcal{M}(B)$, gilt:

$$\beta(s)_{|s} \;=\; m(B) - m(B^{\rightarrow s}) \;=\; 0$$

und zusammen ergibt sich:

$$\Delta\Phi_{|s} \;=\; 0 \;=\; \beta(s)_{|s}\ .$$

Teil (II): Beweis durch Widerspruch.
Sei $r \in R$ und $m(B_{\leftarrow r})$ das Gesamtgewicht des optimalen Matchings von B mit eingefügtem Knoten r. Annahme:

$$\exists\, s \in S_i: \quad \Delta\Phi_{|r} \;<\; w(\{r,s\}) - \beta(s)_{|r}\ .$$

Da sich am Online-Matching M durch das Einfügen des Knotens r nichts ändert, gilt:

$$\Delta\Phi_{|r} \;=\; m(B_{\leftarrow r}) - m(B)\ .$$

In die Annahme eingesetzt, ergibt sich:

$$\begin{aligned} & m(B_{\leftarrow r}) - m(B) \;<\; w(\{r,s\}) - \beta(s)_{|r} \\ \Leftrightarrow\quad & m(B_{\leftarrow r}) - m(B) \;<\; w(\{r,s\}) - [m(B) - m(B^{\rightarrow s})] \\ \Leftrightarrow\quad & m(B_{\leftarrow r}) \;<\; w(\{r,s\}) + m(B^{\rightarrow s})\ . \end{aligned} \tag{15}$$

Der Term $w(\{r,s\}) + m(B^{\rightarrow s})$ ist das Gewicht eines Matchings im Graphen $B_{\leftarrow r}$, da in $B^{\rightarrow s}$ die Knoten r und s nicht enthalten sind. Damit steht jedoch die Aussage (15) im Widerspruch zur vorausgesetzten Optimalität von $m(B_{\leftarrow r})$. So folgt die Behauptung. ■ Lemma 16

6.4.4 Der Beweis des Competitive Ratios

Satz 17:

Der Online-Algorithmus wLMM für das wORSM-Problem ist 2-competitive.

Beweis: Für alle Requestknoten $r \in R$ gilt Auszahlungslemma 16, Teil (II):

$$\Delta\Phi_{|r} \;\geqq\; w(\{r,s\}) - \beta(s)_{|r} \qquad \forall s \in S,\ \{r,s\} \in E\ , \tag{16}$$

wobei die Definition des wORSM-Problems sicherstellt, daß der Requestknoten r in den Graphen eingefügt wird, bevor der Serverknoten s bearbeitet wird.

Für alle Serverknoten $s \in S$ gilt nach dem Auszahlungslemma 16, Teil (I):

$$\Delta\Phi_{|s} \quad \geqq \quad \beta(s)_{|s} \ , \tag{17}$$

und das Stabilitätslemma 15 ergibt für den hier vorliegenden Fall, daß s nach r behandelt wird:

$$\beta(s)_{|s} \quad \geqq \quad \beta(s)_{|r} \ . \tag{18}$$

Die Addition der drei Gleichungen (16), (17) und (18) ergibt:

$$\Delta\Phi_{|r} + \Delta\Phi_{|s} \quad \geqq \quad w(\{r,s\}) \ .$$

Damit gilt die folgende Aussage für ein *beliebiges* Matching M_G von G, wobei Φ_{final} der Potentialfunktionswert nach vollständiger Bearbeitung der Eingabe durch wLMM ist:

$$\Phi_{\text{final}} = \sum_{r \in R} \Delta\Phi_{|r} + \sum_{s \in S} \Delta\Phi_{|s} \geqq \sum_{\{r,s\} \in M_G} \Delta\Phi_{|r} + \Delta\Phi_{|s} \geqq \sum_{\{r,s\} \in M_G} w(\{r,s\}) = |M_G| \ .$$

Deshalb gilt auch für ein maximum-gewichtetes Matching M_{OPT}:

$$\Phi_{\text{final}} \quad \geqq \quad |M_{\mathsf{OPT}}|$$

und aus der Definition von Φ und der Tatsache, daß der lokale bipartite Graph nach Bearbeitung des Gesamtproblems leer ist ($B_{\text{final}} = \varnothing$), folgt:

$$\Phi_{\text{final}} \quad = \quad 2\,|M_{\mathsf{wLMM}}| \quad \geqq \quad |M_{\mathsf{OPT}}| \ ,$$

womit die obere Schranke im Competitive Ratio von 2 gezeigt ist. ■ Satz 17

6.5 Der Algorithmus PHI

In der Analyse des LMM-Algorithmus hat sich gezeigt, daß die bevorzugte Nutzung der aktuellen Ressource s_i ein wesentlicher Beitrag ist, um die optimale Leistungsfähigkeit im Sinne der Competitive Analysis zu gewährleisten. Wie schon auf Seite 88 kurz angedeutet, verbessert eine solche Regelung den Competitive Ratio von wLMM nicht. Formal läßt sich ein Online-Algorithmus wLMM* definieren, der bezüglich des wLMM-Algorithmus (siehe Seite 89) um eine Zeile 5a ergänzt wird. Darin wird geprüft, ob ein alternierender Weg P mit Startknoten s_i existiert, der mit einem freien Knoten oder einer Matchingkante endet, und dessen Matchingkanten die selbe Gewichtssumme wie seine Nichtmatchingkanten besitzen. Dann kann das Matching $\mathcal{M}(B_i)$ durch Invertieren der Matchingzugehörigkeit der Kanten des Weges P in ein maximum-gewichtetes Matching $\mathcal{M}'(B_i)$ mit

$s_i \mathring{\in} \mathcal{M}'(B_i)$ transformiert werden. Zuletzt wird die zu s_i inzidente Matchingkante in das Online-Matching M_{wLMM^*} aufgenommen.

Sei $G = (V, E, w)$ der Eingabegraph des Gegenspielers für wLMM, mit dem eine untere Schranke im Competitive Ratio von wLMM gezeigt wird. Sei weiterhin i der erste Zeitschritt, in dem wLMM und wLMM^* verschiedene Entscheidungen treffen, d. h. $s_i \mathring{\notin} M_{\mathsf{wLMM}}$ und $s_i \mathring{\in} M_{\mathsf{wLMM}^*}$. Dann wird der Gegenspieler alle in B_i zu s_i inzidenten Kanten um eine kleine Konstante ε ($\varepsilon \in \mathbb{R}_+$) in ihrem Gewicht vermindern und die Bedingung für den Weg P ist nicht mehr gegeben, woraus $s_i \mathring{\notin} M_{\mathsf{wLMM}^*}$ folgt. Dieses Konstruktionsschema wird iterativ angewandt und der entstehende Graph $G^* = (V, E, w^*)$ erzeugt ein Matching $M^*_{\mathsf{wLMM}^*}$, welches aus den selben Kanten wie M_{wLMM} besteht. Da sich die Matchinggewichte $|M_{\mathsf{wLMM}}|$ und $|M^*_{\mathsf{wLMM}^*}|$ für $\varepsilon \to 0$ beliebig annähern lassen, und ebenso die Gewichte der zugehörigen optimalen Lösungen, beweist die Eingabe G^* die selbe untere Schranke für den Competitive Ratio des wLMM^*-Algorithmus, wie G für wLMM.

Sollte deshalb im Falle von $s_i \mathring{\notin} \mathcal{M}(B_i)$ die Bedeutung der Ressource s_i durch Anhebung der Gewichte aller zu s_i in B_i inzidenten Kanten um einen Faktor erhöht werden? Und welchen Wert sollte dieser Faktor haben?
Eine Antwort auf die zweite Frage wird durch Betrachtung der Konstruktion der allgemeinen unteren Schranke in Satz 7 nahegelegt. Wenn der Faktor ϕ ist, so sind beide Auswahlmöglichkeiten in der vom Gegenspieler erzeugten Entscheidungssituation gleichwertig.
Die erste Frage läßt sich verneinen, wie die folgende untere Schranke gegen den Algorithmus zeigt. Er soll den Namen wLMM^ϕ erhalten. Wie oben angesprochen, modifiziert er den Graphen B_i im Falle $s_i \mathring{\notin} \mathcal{M}(B_i)$ zu dem Graphen B_i^ϕ, indem alle zu s_i inzidenten Kanten in ihrem Gewicht um den Faktor ϕ erhöht werden ($\forall \{r, s_i\} \in B_i : w^\phi(\{r, s_i\}) := \phi \cdot w(\{r, s_i\})$). Mit dieser Notation kann wLMM^ϕ nun formal in Pseudocode beschrieben werden:

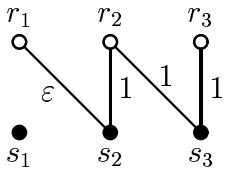
```
1: loop {für alle Zeitschritte i}
2:    lies die Eingabe des Schrittes i und baue B_i auf
3:    bestimme ein maximum-gewichtetes Matching M(B_i) in B_i
4:    if s_i ∉̊ M(B_i) then
5:       konstruiere B_i^φ {d. h. ∀{r, s_i} ∈ B_i : w({r, s_i} := φ·w({r, s_i})}
6:       bestimme ein maximum-gewichtetes Matching M(B_i^φ) in B_i^φ
7:    end if
8:    if s_i ∈̊ M(B_i) oder s_i ∈̊ M(B_i^φ) then
9:       füge die Matchingkante von s_i dem Online-Matching M_wLMM^φ hinzu
10:   end if
11: end loop
```

Algorithmus 3: wLMM^ϕ

Satz 18:

Der Competitive Ratio des Online-Algorithmus wLMM$^{\phi}$ *beträgt mindestens 2.*

Beweis: Die folgende Eingabe des Gegenspielers zeigt die Aussage: $w(\{r_1, s_2\}) = \varepsilon$ und $w(\{r_2, s_2\}) = w(\{r_2, s_3\}) = w(\{r_3, s_3\}) = 1$.



Skizze 12: Eingabestruktur für die untere Schranke von wLMM$^{\phi}$.

Der Online-Algorithmus wLMM$^{\phi}$ bestimmt für diesen Eingabegraphen das Matching $M_{\mathsf{wLMM}^{\phi}} = \{\{r_1, s_2\}, \{r_2, s_3\}\}$, da eindeutig $\mathcal{M}(B_2) = \{\{r_1, s_2\}, \{r_2, s_3\}\}$ und $s_2 \mathring{\in} \mathcal{M}(B_2)$ gilt, weshalb B_2^{ϕ} nicht gebildet und $\mathcal{M}(B_2^{\phi})$ nicht untersucht wird. Das Online-Matching $M_{\mathsf{wLMM}^{\phi}}$ besitzt das Gewicht $|M_{\mathsf{wLMM}^{\phi}}| = 1 + \varepsilon$. Die optimale Lösung ist hingegen $M_{\mathsf{OPT}} = \{\{r_2, s_2\}, \{r_3, s_3\}\}$ mit dem Gewicht $|M_{\mathsf{OPT}}| = 2$. Daraus ergibt sich wiederum das Verhältnis:

$$\frac{|M_{\mathsf{OPT}}|}{|M_{\mathsf{wLMM}}|} = \frac{2}{1+\varepsilon}$$

und für ε gegen Null folgt der Grenzwert:

$$\lim_{\varepsilon \to 0} \frac{2}{1+\varepsilon} = 2 \ .$$

Diese Struktur kann aller drei Zeitschritte erneut als Eingabe fungieren, womit gezeigt ist, daß wLMM$^{\phi}$ keinen besseren Competitive Ratio als 2 besitzt.

■ Satz 18

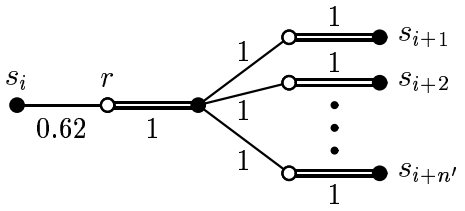
Da die untere Schranke im Competitive Ratio ebenfalls 2 ist, wie auch für den Algorithmus wLMM, hat die Modifikation, die wLMM$^{\phi}$ darstellt, keine Vorteile. Andererseits ist in der Gegenspielerstrategie deutlich zu sehen, warum wLMM$^{\phi}$ keine verbesserte Leistungsgarantie geben kann. Die prinzipielle Idee, die aktuelle Ressource s_i in ihrer Wichtigkeit anzuheben, ist jedoch die einzige Chance einen verbesserten Online-Algorithmus zu schaffen. Darum ist ein Vorschlag der Algorithmus PHI, der die zu s_i inzidenten Kanten in B_i *immer* um den Faktor ϕ in ihrem Gewicht aufwertet. Dieser Algorithmus ist auf Seite 107 in Pseudocodedarstellung zu finden.

Ein bisher ungelöstes Problem ist die Analyse des Online-Algorithmus PHI. Die in Satz 17 genutzte Technik kann dazu nicht angewandt oder einfach modifiziert werden, da das Stabilitätslemma 15 unter den von PHI an B_i durchgeführten dynamischen Veränderungen zusammenbricht. Dieser Zusammenbruch bleibt dabei nicht lokal auf eine oder einige wenige Ressourceknoten beschränkt. Es läßt sich

1: **loop** {für alle Zeitschritte i}
2: lies die Eingabe des Schrittes i und baue B_i auf
3: konstruiere B_i^ϕ {d. h. $\forall \{r, s_i\} \in B_i : w(\{r, s_i\} := \phi \cdot w(\{r, s_i\})\}$
4: bestimme ein maximum-gewichtetes Matching $\mathcal{M}(B_i^\phi)$ in B_i^ϕ
5: **if** $s_i \mathring{\in} \mathcal{M}(B_i^\phi)$ **then**
6: füge die Matchingkante von s_i dem Online-Matching M_{PHI} hinzu
{d. h. $\{s_i, r\} \in \mathcal{M}(B_i^\phi) \implies \{s_i, r\} \in M_{\mathsf{PHI}}\}$
7: **end if**
8: **end loop**

Algorithmus 4: PHI

vielmehr ein sternförmiger Graph, bestehend aus Wegen der Länge Zwei, konstruieren, in dem beliebig viele Ressourceknoten einen Wert $\beta > 0$ besitzen, und in dem nach der Konstruktion von B_i^ϕ diese Werte simultan auf Null fallen.



Skizze 13: Im gezeigten sternförmigen Graphen ist $\mathcal{M}(B_i)$ eingetragen. Es existieren n' Ressourceknoten s_j ($j \in \{i+1, i+2, \ldots, i+n'\}$) mit $\beta(s_j) = 0.38 > 0$. Nach der Konstruktion von B_i^ϕ wird die Kante $\{s_i, r\}$ wichtiger ($0.62 > 1/\phi \Rightarrow \phi \cdot 0.62 > 1$) und es folgt $\beta(s_i) > 0$ aber $\forall\, j \in \{i+1, i+2, \ldots, i+n'\} : \beta(s_j) = 0$.

Anmerkung

Der Algorithmus LMM kann als eine effiziente Spezialimplementierung des Algorithmus PHI für ungewichtete bipartite Graphen aufgefaßt werden. Im Falle von Einheitsgewichten der Kanten, entspricht das Erhöhen der Gewichte von zu s_i inzidenten Kanten um den Faktor ϕ und die Berechnung des maximum-gewichteten Matchings in diesem Graphen B_i^ϕ genau der Bevorzugung von zu s_i inzidenten Kanten im ungewichteten Maximum-Matching.

7 Exakte Analysen einiger Modellvarianten

Übersicht: In diesem Kapitel erfolgt ein Rückgriff auf das ORSM-Problem aus Kapitel 5, d. h. die Kanten der bipartiten Graphen sollen nun wieder Einheitsgewichte besitzen. Es werden neue Konzepte in das schon analysierte ORSM-Problem eingeführt, die zuerst einzeln untersucht werden. Eine Kombination dieser Konzepte führt danach zu Modellvorstufen des Data-Access-Problems (DAP). Soweit vollständige Analysen für derartige ORSM-Modellvarianten vorliegen, werden sie in diesem Abschnitt dargestellt. Die vorhandenen Resultate einer weiteren Modellvariante und des DAP werden in den späteren Kapiteln 8 und 9 behandelt.

7.1 Vorbemerkungen und Ergebnisübersicht

Obwohl in Kapitel 4 schon einige Aussagen zur Motivation der in diesem Kapitel behandelten Modellvarianten getroffen wurden, soll zuvor noch einmal der Zweck der Untersuchungen dieser Modelle verdeutlicht werden.
Das ORSM-Problem ist ein hochgradig abstraktes und anwendungsfernes Modell, welches den Untersuchungen relativ leicht zugänglich ist. Ein anderes Extrem stellt das DAP dar, welches immer noch eine starke Vereinfachung für ein verteilten Datenserver beschreibt bzw. nur einige Teilaspekte berücksichtigt, dessen Strukturaufklärung und -auswertung jedoch schwierig ist. Um ein Verständnis für die Strukturen des letztgenannten Problems zu erhalten, ist die Betrachtung von Modellvarianten zwischen diesen beiden Extremen sinnvoll. Dazu werden die Konzepte wie Blockeingabe, konstanter Requestknotengrad und beschränkte Kantenlängen (maximale individuelle Fristen der Aufträge) dem einfachen ORSM-Modell hinzugefügt, da sie durch das DAP impliziert werden. Außerdem sollen noch Vorschau in der Eingabesequenz und Intervalleingaben als zusätzliche Konzepte untersucht werden. Wie schon in Kapitel 4 aufgezeigt, stellen diese „Modellerweiterungen“ in Wahrheit Einschränkungen in der Eingabestruktur des ORSM-Problems dar bzw. ein zusätzliches a priori Wissen. Deshalb bleibt die obere Schranke von 1.5 im Competitive Ratio (Satz 6) für alle Modelle gültig.

Für sämtliche Modellvarianten, deren Competitive Ratio genau 1.5 ist, genügt es Gegenspielerstrategien für untere Schranken anzugeben. Dabei handelt es sich nur um leichte Anpassungen der Gegenspielerstrategie aus Satz 3. Die oberen Schranken für Resultate kleiner als 1.5 werden durch den Online-Algorithmus LMM gezeigt. Diese Analysen folgen ebenfalls der bekannten Argumentation aus Satz 6, in dem LMM für das einfache ORSM-Problem untersucht wird.

In den folgenden Teilkapiteln werden zuerst die Modellvarianten untersucht, die dem ORSM-Modell lediglich ein zusätzliches Konzept hinzufügen. Danach werden Kombinationen solcher Konzepte studiert, solange für die Modellvarianten exakte Analysen vorliegen. Die Tabelle 3 gibt eine Übersicht der in diesem Kapitel bewiesenen Resultate.

7.2 Analysen der Modellvarianten

7.2.1 Das Modell mit Intervalleingaben

Jede Anfrage in dieser Modellvariante spezifiziert ein ununterbrochenes Intervall von Zeitpunkten bzw. Serverknoten für ihre Bearbeitung. Mit dieser speziellen Struktur der Eingabe wird es möglich zwei Anfragen, die beide im aktuellen Zeitschritt bedienbar sind, zu vergleichen. Die Anfrage mit der kürzeren Frist (Deadline) ist dringender und damit wichtiger. Sie sollte deshalb in der Bearbeitung bevorzugt werden. Innerhalb dieser früheren Frist verhalten sich beide Anfragen im System völlig identisch und sind daher austauschbar.

Intervall-eingabe	Vorschau ℓ	Block-eingabe b	Requestknotengrad g	Kantenlänge d	Competitive Ratio	Kommentar
×					1	EDF-Algorithmus
	×				1.5	
		×			1.5	
			×		1.5	$g \geqq 2$
				×	1.5	$d \geqq 2$
	×	×	×		1.5	$g \geqq 2$
		×		×	1.5	$b \geqq 2, d \geqq 1$
	×			×	$\frac{2\lfloor \ell/d \rfloor + 3}{2\lfloor \ell/d \rfloor + 2}$	für $\ell + 1 \not\equiv 0 \bmod d$
					$\frac{2\lceil \ell/d \rceil + 2}{2\lceil \ell/d \rceil + 1}$	für $\ell + 1 \equiv 0 \bmod d$
	×	×		×	$\frac{2\lfloor \ell/d \rfloor + 3}{2\lfloor \ell/d \rfloor + 2}$	$b \geqq 2, d \geqq 1$

Tabelle 3: Das ORSM-Problem mit verschiedenen Kombinationen von zusätzlichen Konzepten und die resultierenden Competitive Ratios.

J. R. Jackson untersuchte in [Jac55] ein Offline-Planungsproblem, in dem die Aufträge keine einzuhaltenden Fristen, sondern gewünschte Fertigstellungszeitpunkte[26] besitzen. Das Ziel in diesem Planungsproblem ist die Minimierung der Summe aller Verspätungen der Aufträge bezüglich ihrer Fertigstellungswünsche. Die optimale Lösung ist als Jackson's Regel bekannt und bedient sich einer Beobachtung die sehr ähnlich der oben getroffenen ist: Alle Aufträge werden gemäß nicht fallender gewünschter Endzeitpunkte sortiert und in dieser Reihenfolge abgearbeitet. Der so definierte Algorithmus besitzt auch den Namen *Earliest-Due-Dates* (EDD) und stellt eine spezielle Variante des List-Scheduling dar.
Stellen die Fertigstellungszeitpunkte harte Fristen dar, so verwandelt sich Jackson's Regel in den Earliest-Deadline-First-Algorithmus (EDF). Er ist in der Lage in Zeit $O(n \log n)$ das Entscheidungsproblem, ob ein Plan existiert, der alle Aufträge einer Probleminstanz erfüllt, zu lösen (siehe dazu z. B. [BESW94, Seite 52] oder [Cof76, Seite 13 f.]).

Für ein Online-Planungsproblem mit Fristen, bei dem die veränderte Zielfunktion der Anzahl erfolgreicher Aufträge zu maximieren ist, funktionieren die Beweise zur Optimalität von EDF aus der Literatur nicht. Im Offline-Problem sind alle Aufträge zu bearbeiten und es kann ein einfaches Vertauschungsargument benutzt werden, um jede optimale Lösung in die vom EDF-Algorithmus erzeugte

[26] Im Englischen als *due dates* bezeichnet.

Lösung zu transformieren, ohne dabei die Lösungsqualität zu beeinträchtigen. Da im Online-Modell verschiedene Mengen erfolgreicher Aufträge in der optimalen bzw. durch EDF erzeugten Lösung enthalten sein können, und die Aufträge individuelle früheste Startzeitpunkte besitzen, ist eine aufwendigere Beweisführung notwendig.
Vor dieser Analyse soll jedoch erst die exakte Funktion des EDF-Algorithmus für das Planungsproblem, welches dem ORSM-Modell mit Intervalleingaben entspricht, geklärt werden: In jedem Zeitschritt i wird als erstes der mögliche neue Auftrag eingelesen und in einem Auftragspuffer zwischengespeichert. Danach wird der Auftrag r mit kürzester noch verbleibender Frist bestimmt, der im Zeitschritt i bearbeitet werden kann. Existiert ein solcher ausführbarer Auftrag im Puffer, so wird er daraus entfernt und bearbeitet. Zum Schluß werden alle Aufträge mit abgelaufener Frist i aus dem Puffer gelöscht und der nächste Zeitschritt beginnt.

Auf Seite 112 ist eine detailliertere Pseudocodedarstellung des EDF-Algorithmus für das ORSM-Problem mit Intervalleingabe unter der graphentheoretischen Interpretation angegeben. Dabei ist ein Requestknoten r durch seinen frühesten und spätesten Servicezeitpunkt r_{start} und r_{end} beschrieben, d. h. r_{start} und r_{end} stellen die Zeitindizes des ersten und letzten zu r adjazenten Serverknotens dar. Per Definition sei ein isolierter Requestknoten r durch $r_{\text{end}} = 0$ charakterisiert. In der Algorithmusdarstellung wird auf den abstrakten Datentyp der linearen Liste mit den Operationen:

$initialize(L)$	–	richtet eine leere Liste L ein; insbesondere wird ein Schlußelement erzeugt, das sich selbst als Nachfolger besitzt
$head(L)$	–	gibt das erste Listenelement aus L zurück
$end(L)$	–	bezeichnet das Schlußelement von L
$next(r, L)$	–	gibt das in L auf r nachfolgende Listenelement zurück
$delete(r, L)$	–	entfernt das Element r aus L

und einer Einfügeoperation zurückgegriffen.

Satz 19:

Der Online-Algorithmus EDF für das ORSM-Problem mit Intervalleingaben ist 1-competitive.

Beweis: Die von EDF konstruierte Lösung wird in Phasen eingeteilt. Eine Phase beginnt mit einem Intervall, in dem alle Serverknoten im Matching M_{EDF} sind und es folgt ein Intervall, dessen Serverknoten bezüglich M_{EDF} frei sind. Die Phasen seien mit P_j durchnumeriert, und P_1 beginnt mit dem ersten gebundenen Serverknoten. P_0 entspricht dem möglicherweise vorhandenen Intervall von freien Serverknoten am Anfang der Eingabe. In der letzten Phase folgt auf das Intervall gebundener Serverknoten eine unendliche Menge weiterer Servicezeitpunkte.
Eine Phase P_j wird durch ihren Startzeitpunkt b_j, den Zeitpunkt des letzten benutzten Servicezeitpunktes u_j und den letzten Zeitpunkt e_j charakterisiert. Mit dieser Phasendefinition wird die Requestknotenmenge in disjunkte Teilmengen aufgegliedert. Jeder Phase P_j wird dabei die Menge $R_j \subset R$ zugeordnet, die alle

```
 1: initialize(L)
 2: loop {für alle Zeitschritte i}
 3:    lies die Eingabe r_i des Schrittes i
 4:    if r_{i,end} > 0 then {r_i ist ein nichtisolierter Requestknoten}
 5:       sortiere r_i in L nach (r_{i,end}, r_{i,start}) lexikographisch aufsteigend ein
 6:    end if
 7:    r := head(L)
 8:    repeat {sucht zu s_i adjazenten Requestknoten mit kürzester Frist}
 9:       if r_start ≦ i then {entsprechender Requestknoten gefunden}
10:          füge die Kante {r, s_i} dem Online-Matching M_EDF hinzu
11:          delete(r, L)
12:          break {Schleifenabbruch}
13:       else
14:          r := next(r, L)
15:       end if
16:    until r = end(L)
17:    r := head(L) {entfernt im folgenden alle Elemente mit Frist i aus L}
18:    while r ≠ end(L) und r_end = i do
19:       delete(r, L)
20:       r := head(L)
21:    end while
22: end loop
```

Algorithmus 5: EDF (anstatt einer linearen Liste kann auch eine effizientere Datenstruktur — beispielsweise auf balancierten Bäumen basierend — eingesetzt werden)

Requestknoten r beinhaltet, deren früheste Servicezeitpunkte innerhalb der Phase P_j liegen ($b_j \leqq r_{\text{start}} \leqq e_j$). Der Phase P_0 ist die leere Menge R_0 zugeordnet.

Bemerkung: Der EDF-Algorithmus sortiert die Requestknoten in seiner Liste L primär nach nichtabsteigenden Fristen und sekundär nach nichtabsteigenden frühesten Servicezeitpunkten. Sind beide Werte für zwei Knoten identisch, so sind sie nicht unterscheidbar und deshalb ist ihre Stellung zueinander in L egal. Wie man sich leicht überzeugen kann, entspricht die Menge R_j genau der Menge von Requestknoten, die im Intervall $[b_j, u_j]$ der Phase P_j von EDF aus seiner internen Datenstruktur L entnommen wird und die frühesten Servicezeitpunkte der Requestknoten aus R_j übersteigen niemals den Wert u_j. Eine davon abweichende Eigenschaft würde dem Intervall freier Serverknoten $\{s_{u_j+1}, \ldots, s_{e_j}\}$ unter der EDF-Regel widersprechen. Es gilt also:

$$R \;=\; \dot{\bigcup_{j \in \mathbb{N}}} R_j$$

sowie

$$\forall r \in R_j : \quad b_j \;\leqq\; r_{\text{start}} \;\leqq\; u_j \;.$$

Die Berechnung der Matchings der Phasen ist mit obiger Definition voneinander unabhängig, da das Zuordnen eines Requestknotens $r \in R_j$ zu einem Serverknoten aus $\{s_{b_{j+1}}, \ldots, s_{e_{j+1}}\}$ der nächsten Phase P_{j+1} die Kardinalität der Gesamtlösung M_{EDF} nicht erhöhen kann, da die Phase P_j selbst mit freien Serverknoten und damit unbenutzten Kapazitäten endet. Weiterhin ist es nach Definition der Phasen unmöglich einen Requestknoten mit einem Serverknoten der vorherigen Phase zu paaren. Wird nun gezeigt, daß EDF die Requestknoten R_j innerhalb der Phase P_j optimal den Serverknoten zuordnet, so folgert sich daraus die globale Optimalität des Online-Algorithmus EDF.
Die Phase P_0 wird offensichtlich durch EDF optimal gehandhabt, da im definierten Intervall alle Serverknoten isoliert sind. Innerhalb einer Phase P_j ($j \geqq 1$) kann nun die spezielle Struktur der Lösung, d. h. alle gebundenen Serverknoten befinden sich konsekutiv am Anfang der Phase, für die Beweisführung genutzt werden. Gilt $|R_j| = u_j - b_j + 1$, d. h. die Requestknotenmenge R_j besitzt genauso viele Elemente, wie die Anzahl der in Phase P_j gebundenen Serverknoten $\{s_{b_j}, \ldots, s_{u_j}\}$, so liegt ebenfalls eine optimale Lösung für P_j vor.
Anderenfalls sei r_f der erste Knoten aus R_j, der von EDF aus der Liste L entfernt und nicht dem Matching M_{EDF} hinzugefügt wurde. Dann hat EDF alle Requestknoten[27] aus R_j, die vor r_f in L sind und deshalb keine späteren Fristen als $r_{f,\mathrm{end}}$ aufweisen, bezüglich M_{EDF} gebunden. Zusätzlich können Requestknoten mit längeren Fristen den Serverknoten aus $\{s_{b_j}, \ldots, s_{r_{f,\mathrm{end}}-1}\}$ zugeordnet sein, falls zu diesen Zeitpunkten keine adjazenten Requestknoten mit kürzeren Fristen zur Verfügung standen. EDF ordnet niemals Requestknoten unnötigerweise früheren Serverknoten zu, und deshalb kann die Matchingkardinalität durch Zuordnung solcher Requestknoten mit Fristen größer als $r_{f,\mathrm{end}}$ zu späteren Serverknoten nicht erhöht werden. Mithin ist die Teillösung im Intervall $[b_j, r_{f,\mathrm{end}}]$ optimal, da alle Ressourcen ausgenutzt werden und es keine Möglichkeit gibt den abgewiesenen Requestknoten innerhalb des Intervalls $[b_j, r_{f,\mathrm{end}}]$ einem Serverknoten zuzuweisen ohne im Gegenzug einen anderen Requestknoten mit kürzerer oder gleicher Frist aus dem Matching M_{EDF} zu entfernen. Nun läßt sich das Matchingproblem der Phase P_j zu einem Restproblem P'_j reduzieren, indem alle Requestknoten mit einer Frist bis $r_{f,\mathrm{end}}$ aus R_j entfernt werden, d. h.:

$$R'_j \;:=\; \{r \in R_j \mid r_{\mathrm{end}} > r_{f,\mathrm{end}}\} \;.$$

Die Zeitpunkte des Restproblems werden auf das Intervall $[r_{f,\mathrm{end}} + 1, e_j]$ und die Servicezeitpunkte aller Requestknoten aus R'_j reduziert:

$$S'_j \;:=\; \{s_k \in S \mid (\{r, s_k\} \in M_{\mathsf{EDF}} \;\Rightarrow\; r_{\mathrm{end}} > r_{f,\mathrm{end}}) \;\vee\; (r_{f,\mathrm{end}} < k \leqq e_j)\} \;.$$

Mit dieser Definition erfüllt der Rest P'_j einer Phase P_j die gleichen Voraussetzungen, wie sie an eine Phase gestellt werden: Kein Requestknoten $r \in R'_j$ kann sinnvoll einem Serverknoten verschieden S'_j zugeordnet werden, denn eine Paarung von r mit einem Knoten aus $S_j \setminus S'_j$ würde zu einer unumgänglichen Verdrängung

[27] maximal $r_{f,\mathrm{end}} - b_j + 1$ Stück

eines anderen Requestknotens führen, was die Kardinalität des Matchings M_{EDF} nicht erhöhen kann. Andere Zuordnungen sind aus den oben zu einer Phase genannten Gründen nicht von Vorteil.

Das Restproblem P'_j ist strikt kleiner als das Ausgangsproblem der Phase P_j. Es entfällt mindestens der Zeitschritt $r_{f,\text{end}}$, zwei Requestknoten (r_f sowie der $s_{r_{f,\text{end}}}$ zugeordnete Requestknoten) und der Zeitpunkt des Fristablaufes des nächsten erfolglosen Requestknotens wird ebenfalls erhöht.
Die Lösung, die EDF auf dem isolierten Problem P'_j erzeugt, besteht aus den selben Entscheidungen, wie in der EDF-Lösung der gesamten Phase P_j. Alle Zuordnungen von Requestknoten vor dem Zeitpunkt $r_{f,\text{end}}$ werden für das Restproblem P'_j ebenso durchgeführt, wie innerhalb der Phase P_j, da zu diesen Zeitpunkten in P_j keine Requestknoten aus der Menge $R_j \setminus R'_j$ zur Verfügung stehen. Ab dem Zeitpunkt $r_{f,\text{end}} + 1$ sind zusätzlich die internen Konfigurationen der Datenstrukturen von EDF in beiden Problemvarianten identisch. Mit diesen Beobachtungen kann nun formal ein Induktionsbeweis über die Zeitpunkte geführt werden, in denen Requestknoten endgültig als frei bezüglich M_{EDF} festgelegt werden. Der Induktionsabbruch erfolgt bei einem der beiden trivialen Fälle einer leeren Requestknotenmenge R'_j oder dem Zuordnen aller Requestknoten des Restproblems.
Somit ist die oben gesuchte Optimalitätseigenschaft der von EDF bestimmten Lösung für eine beliebige Phase P_j nachgewiesen, woraus schließlich die Aussage des Satzes folgt. ■ Satz 19

7.2.2 Das Modell mit Vorschau

Eine Vorschau in die zukünftigen Teile der Eingabesequenz könnte zusätzliche Information aufdecken, die für eine bessere Entscheidung benutzt werden könnte. Wie schon in [Alb97] festgestellt und auf Seite 30 besprochen, folgt dies nicht notwendigerweise. Auch im ORSM-Modell kann der Gegenspieler Eingaben erzeugen, die trotz Vorschau von ℓ Zeitschritten keine zusätzlich verwertbare Information offenbaren. Dazu wird die Strategie aus Satz 3 nur leicht modifiziert.

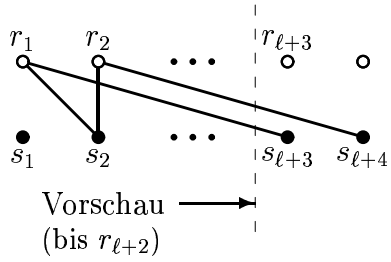
Satz 20:

Jeder deterministische Online-Algorithmus für das ORSM-Problem mit Vorschau von ℓ Zeitschritten besitzt einen Competitive Ratio von mindestens 1.5 .

Beweis: Folgende Eingabe ist die Grundlage der Gegenspielerstrategie (siehe auch Skizze 14):

$$E \;=\; \{\{r_1, s_2\}, \{r_1, s_{\ell+3}\}, \{r_2, s_2\}, \{r_2, s_{\ell+4}\}\} \;.$$

Je nach Entscheidung des Online-Algorithmus ($\{r_1, s_2\} \in M$ oder $\{r_2, s_2\} \in M$) wird im Zeitschritt $i = \ell + 3$ die Kante $\{r_{\ell+3}, s_{\ell+4}\}$ oder $\{r_{\ell+3}, s_{\ell+3}\}$ eingefügt. Diese letzte Kante ist zum Entscheidungszeitpunkt $i = 2$ noch nicht bekannt.



Skizze 14: Der Eingabegraph und die Entscheidungssituation im Zeitschritt $i = 2$. Die gestrichelte Linie begrenzt die Vorschau.

Deshalb kann die Argumentation aus dem Beweis zu Satz 3 auf diesen modifizierten Eingabegraphen übernommen werden. Es folgt eine unter Schranke im Competitive Ratio von 1.5 . ■ Satz 20

Zusammen mit der oberen Schranke aus Satz 6, die sich auf diese Modellvariante überträgt, ist die Analyse vollständig.

7.2.3 Das Modell mit Blockeingabe

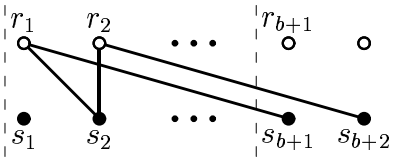
Auch in dieser Modellvariante genügt eine geringfügige Modifikation der Gegenspielereingabe, um eine untere Schranke von 1.5 für den Competitive Ratio zu zeigen. Der LMM-Algorithmus mit seiner Leistungsgarantie aus Satz 6 vervollständigt die exakte Analyse.

Satz 21:

Jeder deterministische Online-Algorithmus für das ORSM-Problem mit Blockeingabe von b Knoten pro Zeitschritt ($b \in \mathbb{N}, b \geqq 2$) besitzt einen Competitive Ratio von mindestens 1.5.

Beweis: Der Gegenspieler konstruiert als Eingabe die Kantenmenge

$$E \;=\; \{\{r_1, s_2\}, \{r_1, s_{b+1}\}, \{r_2, s_2\}, \{r_2, s_{b+2}\}\} \;.$$



Skizze 15: Der Eingabegraph und die Entscheidungssituation im ersten Zeitschritt. Die gestrichelten Linien begrenzen die Eingabeblöcke.

Im zweiten Zeitschritt wird dann die Kante $\{r_{b+1}, s_{b+1}\}$ oder $\{r_{b+1}, s_{b+2}\}$ hinzugefügt. Sie ist bei der Entscheidung über die Nutzung der Ressource s_2 noch

nicht bekannt. Die Argumentation aus Satz 3 kann deshalb benutzt werden und vervollständigt den Beweis. ■ Satz 21

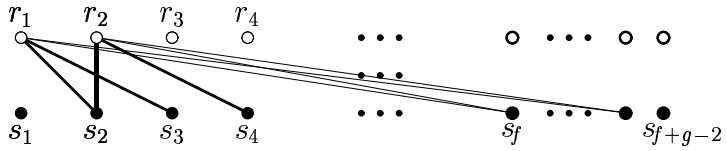
7.2.4 Das Modell mit konstantem Requestknotengrad

Die Grundidee einer Gegenspielerstrategie, die für diese Modellvariante ebenfalls einen Competitive Ratio von mindestens 1.5 zeigt, besteht darin, alle „überflüssigen" Kanten der Requestknoten beliebig weit in der Zukunft auf $g-1$ Serverknoten zu versammeln. Wird die Grundstruktur der Gegenspielereingabe, wie in Satz 3, sehr häufig angewandt, so verschwindet der Einfluß dieser zusätzlichen $g-1$ Matchingkanten.

Satz 22:

Jeder deterministische Online-Algorithmus ALG für das ORSM-Problem mit konstantem Requestknotengrad von g ($g \in \mathbb{N}, g \geqq 2$) besitzt einen Competitive Ratio von mindestens 1.5.

Beweis: Der Beweis benutzt die Gegenspielerstrategie aus Satz 3 mit einer kleinen Anpassung: Alle $g-2$ bzw. $g-1$ Kanten von Anfrageknoten, die in der Gegenspielereingabe des Satzes 3 nicht benutzt werden, haben die Serverknoten $\{s_f, s_{f+1}, \ldots, s_{f+g-2}\}$ als Endpunkte. Diese Menge bleibt auch für die Wiederholungen der Grundstruktur die selbe und ihr erster Serverknoten s_f besitzt einen Zeitindex f, der nach allen in den Wiederholungen der Grundstruktur benutzten Knotenindizes liegt.



Skizze 16: Die Eingabe und die Entscheidungssituation zum Zeitpunkt $i = 2$.

Ein Online-Algorithmus ALG hat für den Serverknoten s_2 eine Entscheidung zu treffen, die der Situation im Beweis des Satzes 3 gleich kommt. Die Gegenspielerreaktion ist ebenfalls identisch. Je nach Wahl von $\{r_1, s_2\} \in M_{\mathsf{ALG}}$ oder $\{r_2, s_2\} \in M_{\mathsf{ALG}}$ wird der Gegenspieler die Kante $\{r_3, s_4\}$ oder $\{r_3, s_3\}$ einfügen. Die restlichen $g-1$ Kanten des Requestknotens r_3 werden ebenfalls mit den Serverknoten $\{s_f, s_{f+1}, \ldots, s_{f+g-2}\}$ verbunden.
Für eine beliebige Anzahl k von Wiederholungen dieser Strategie gilt $|M_{\mathsf{OPT}}| = 3k$ und $|M_{\mathsf{ALG}}| = 2k+g-1$, wobei g eine Konstante unabhängig von der Eingabelänge ist. Daraus ergibt sich 1.5 auch als untere Schranke für den Competitive Ratio dieser Modellvariante. ■ Satz 22

Die passende obere Schranke für die exakte Bestimmung des Competitive Ratios wird ein weiteres Mal durch den Satz 6 gegeben.

7.2.5 Das Modell mit begrenzter Kantenlänge

Wird mit dem Parameter d die Länge der Kanten begrenzt, d. h. die Zeitintervalle, die Kanten im Eingabegraphen überbrücken dürfen, so ist sinnvollerweise $d \geqq 2$ zu fordert. Für $d < 2$ können nur noch Anfragen formuliert werden, die Intervallcharakter besitzen. Damit würde sich diese Modellvariante zum ORSM-Modell mit Intervalleingabe vereinfachen.
Die Eingabe der Gegenspielerstrategie in Satz 3 macht aber keinen Gebrauch von Kanten, deren Länge Zwei übersteigt. Deshalb funktioniert diese Gegenspielerstrategie auch für das ORSM-Modell mit begrenzter Kantenlänge. Zusammen mit der Anwendbarkeit des Satzes 6 auf diese Modellvariante ergibt sich das folgende Korollar.

Korollar 23:

Das ORSM-Problem mit begrenzter Kantenlänge von d Zeitschritten ($d \in \mathbb{N}$, $d \geqq 2$) hat einen Competitive Ratio von genau 1.5.

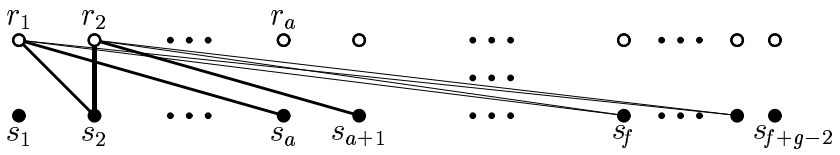
7.2.6 Die Kombination von Vorschau, Blockeingabe und konstantem Requestknotengrad

Auch in dieser Modellvariante ist keine Verbesserung des Competitive Ratios möglich. Dazu werden die Modifikationen der Gegenspielerstrategie aus Satz 3, die in den Beweisen zu den Sätzen 20, 21 und 22 vorgenommen wurden, verbunden. Für diese Modellvariante gilt es zu beachten, daß pro Zeitschritt b Requestknoten eingefügt und b Serverknoten behandelt werden. Deshalb bezieht sich die Vorschau von ℓ zusätzlichen Zeitschritten auf $\ell \cdot b$ Knoten in die Zukunft.

Satz 24:

Das ORSM-Problem mit Vorschau von ℓ Zeitschritten, Blockeingabe mit Blocklänge b und konstantem Requestknotengrad von g hat einen Competitive Ratio von genau 1.5.

Beweis: Entsprechend der Basisfunktion der Gegenspielerstrategie aus Satz 3 und den Anpassungen in den Sätzen 20, 21 und 22 benutzt der Gegenspieler die in Skizze 17 dargestellte Eingabestruktur.



Skizze 17: Der Eingabegraph und die Entscheidungssituation im ersten Zeitschritt.

Der Gegenspieler setzt $a = (\ell + 1) \cdot b + 1$ und ein Online-Algorithmus ALG besitzt im ersten Zeitschritt (in dem auch über s_2 entschieden wird, da $b \geqq 2$) nicht

mehr Information als im einfachen ORSM-Modell. Die weitere Argumentation folgt der im Beweis zu Satz 22 und mündet in der unteren Schranke von 1.5 für den Competitive Ratio.
Die passende obere Schranke ist schon durch Satz 6 bewiesen. ■ Satz 24

7.2.7 Die Kombination von Blockeingabe und begrenzter Kantenlänge

In dieser Variante des ORSM-Problems ist die maximale Kantenlänge d in Zeitschritten gegeben. Da pro Zeitschritt b Knoten jeder Partition berücksichtigt werden, ergibt sich eine leicht abgewandelte und nichtuniforme Definition für die Maximallänge der Kanten. Der Parameter d muß aber mindestens 1 sein, damit die Anfragen Servicezeitpunkte im nächsten Zeitschritt bzw. Block spezifizieren können. Anderenfalls würde dieses Online-Problem in ein optimal lösbares Offline-Problem pro Zeitschritt zerfallen, denn mit $d = 0$ existieren keine Abhängigkeiten zwischen zwei Zeitschritten.
Die vom Gegenspieler erzeugte Eingabe im Beweis des Satzes 21 benutzt keine Kanten, die mehr als einen Zeitschritt (Block) in die Zukunft reichen. Deshalb funktioniert diese Gegenspielerstrategie für die hier betrachtete Modellvariante ohne Änderungen und es folgt das Korollar:

Korollar 25:

Das ORSM-Problem mit Blockeingabe der Länge b und begrenzter Kantenlänge von d Zeitschritten ($d \in \mathbb{N}$, $d \geqq 1$) hat einen Competitive Ratio von genau 1.5.

7.2.8 Die Kombination von Vorschau und begrenzter Kantenlänge

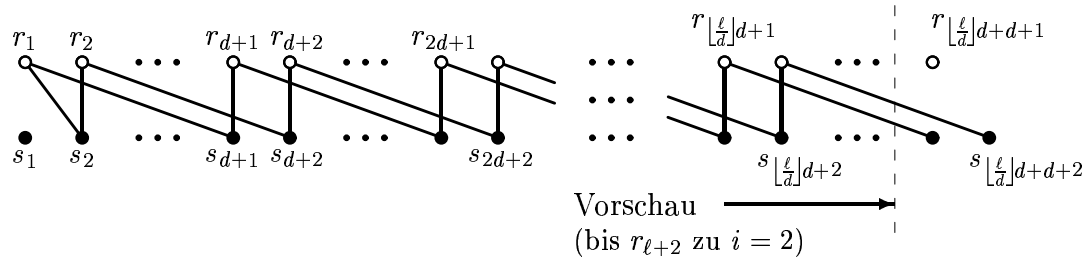
Wenn die Konzepte von Vorschau um ℓ Zeitschritte und auf eine Länge von d Zeitschritten begrenzte Kanten gemeinsam dem ORSM-Modell hinzugefügt werden, so verringert sich der Competitive Ratio, sobald ℓ ein Mehrfaches von d ist. Dabei tritt der Fall ein, daß ein Online-Algorithmus zu einem Entscheidungszeitpunkt garantiert mehr Information über die Struktur des Eingabegraphen besitzt; genauer gesagt ist eine weitere Nachbarschaft des aktuellen Serverknotens vollständig bekannt. Dieses zusätzliche Wissen wird ohne weitere Anpassung vom Algorithmus LMM ausgenutzt.

Satz 26:

Der Competitive Ratio $\mathcal{R}$ des ORSM-Problems mit ℓ-Vorschau und begrenzter Kantenlänge von d Zeitschritten ($d \geqq 2$) beträgt:

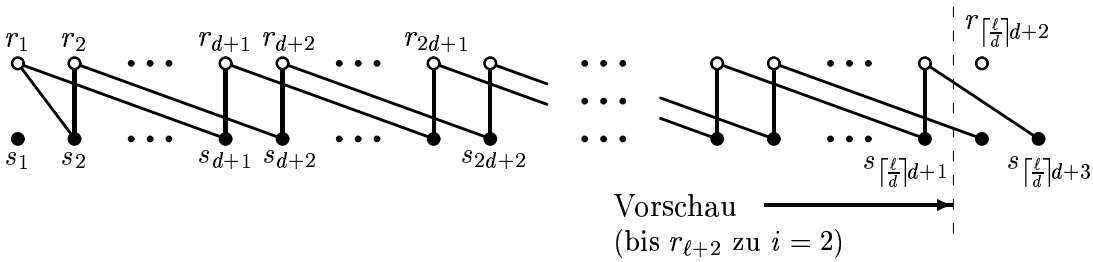
$$\mathcal{R} = \begin{cases} \dfrac{2\lfloor \ell/d \rfloor + 3}{2\lfloor \ell/d \rfloor + 2} & \text{für} \quad \ell + 1 \not\equiv 0 \bmod d \\ \dfrac{2\lceil \ell/d \rceil + 2}{2\lceil \ell/d \rceil + 1} & \text{für} \quad \ell + 1 \equiv 0 \bmod d \end{cases} .$$

Beweis: Der Beweis der unteren Schranke wird in die beiden dem Resultat entsprechenden Fälle untergliedert. Der grundlegende Eingabegraph der Gegenspielerstrategie benutzt für die Hälfte aller Kanten die maximal mögliche Länge. Er implementiert die Entscheidungssituation ähnlich des Satzes 3, muß jedoch eine „Verlängerung“ durchführen, um die am weitesten in der Zukunft liegenden und mit Kanten inzidenten Serverknoten außerhalb des Vorschaubereiches zu plazieren. Dies geschieht für den Fall $\ell+1 \not\equiv 0 \bmod d$, wie in Skizze 18 gezeigt.



Skizze 18: Situation im Zeitschritt $i = 2$ mit Vorschau der ℓ nächsten Knoten.

Der Gegenspieler reagiert auf die Entscheidung eines Online-Algorithmus ALG über die Nutzung von s_2 mit dem Einfügen der Kante $\{r_{\lfloor \ell/d \rfloor d+d+1}, s_{\lfloor \ell/d \rfloor d+d+1}\}$ bzw. der Kante $\{r_{\lfloor \ell/d \rfloor d+d+1}, s_{\lfloor \ell/d \rfloor d+d+2}\}$. Damit gilt für diesen Eingabegraphen $|M_{\mathsf{ALG}}| \leqq 2\lfloor \ell/d \rfloor + 2$ und $|M_{\mathsf{OPT}}| = 2\lfloor \ell/d \rfloor + 3$. Die weiteren Schritte der Argumentation sind identisch zu denen im Beweis zu Satz 3.
Für den Fall, daß $\ell+1 \equiv 0 \bmod d$ gilt, wird die Situation ein wenig asymmetrisch und es wird eine Eingabestruktur notwendig, wie sie in Skizze 19 zu sehen ist.



Skizze 19: Situation des asymmetrischen Falls im Zeitschritt $i = 2$ mit Vorschau der ℓ nächsten Knoten.

Auf die Entscheidung des Online-Algorithmus ALG über den Serverknoten s_2 ($\{r_1, s_2\} \in M_{\mathsf{ALG}}$ oder $\{r_2, s_2\} \in M_{\mathsf{ALG}}$) antwortet der Gegenspieler mit der Kante $\{r_{\lceil \ell/d \rceil d+2}, s_{\lceil \ell/d \rceil d+3}\}$ oder $\{r_{\lceil \ell/d \rceil d+2}, s_{\lceil \ell/d \rceil d+2}\}$. Dann gilt für diesen Eingabeteil $|M_{\mathsf{ALG}}| \leqq 2\lceil \ell/d \rceil + 1$ und $|M_{\mathsf{OPT}}| = 2\lceil \ell/d \rceil + 2$, und die beliebige Wiederholung dieser Strategie zeigt die gesuchte untere Schranke auf.

Der Online-Algorithmus LMM erreicht die passenden oberen Schranken. Dazu wird die Argumentation aus Satz 6 angewandt. Im Fall $\ell + 1 \not\equiv 0 \bmod d$ kennt LMM zum Zeitpunkt i die vollständige $(2\lfloor \ell/d \rfloor + 2)$-Nachbarschaft des Knotens s_i im Graphen G. Diese Tatsache folgt aus der begrenzten Kantenlänge und dem Wert ℓ der Vorschau. Die obere Schranke für den Competitive Ratio ergibt sich nun aus der Beobachtung, daß in der durch LMM erzeugten Lösung M_{LMM} keine erweiternden Wege der Länge $4\lfloor l/d \rfloor + 3$ oder kürzer existieren können.
Mit einer zusätzlichen Fallunterscheidung wird für den asymmetrischen Fall (bei $\ell + 1 \equiv 0 \bmod d$) die Nichtexistenz von erweiternden Wegen der Länge $4\lceil \ell/d \rceil + 1$ und kürzerer Länge in M_{LMM} gezeigt. Die dazu notwendigen Argumente und Schlußfolgerungen entsprechen ebenfalls denen im Beweis zu Satz 6 angegebenen.

■ Satz 26

Wird die soeben untersuchten Variante des ORSM-Problems mit ℓ-Vorschau und begrenzter Kantenlänge um das Konzept der Blockeingabe erweitert, so bezieht sich die Vorschau von ℓ Zeitschritten und die maximale Kantenlänge von d Zeitschritten auf jeweils einen Block mit b Knoten jeder Partition pro Zeitschritt. Da $b \geqq 2$ entfällt der asymmetrische Spezialfall ($\ell + 1 \equiv 0 \bmod d$) aus dem vorherigen Satz. Die übrigen Teile des Beweises übertragen sich jedoch exakt auf diese Modellvariante und implizieren damit das folgende Korollar.

Korollar 27:

Das ORSM-Problem mit ℓ-Vorschau, Blockeingabe mit Blocklänge b und begrenzter Kantenlänge von d Zeitschritten ($d \geqq 1$) hat einen Competitive Ratio von

$$\frac{2\lfloor \ell/d \rfloor + 3}{2\lfloor \ell/d \rfloor + 2} \ .$$

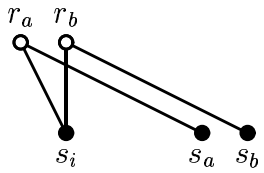
8 Die Modellvariante mit konstantem Knotengrad und begrenzter Kantenlänge

Übersicht: Dieses Kapitel widmet sich den Studien einer einzigen Variante des ORSM-Problems. In dieser Modellvariation spezifizieren nichtisolierte Requestknoten (d. h. aufgetretene Anfragen) eine konstante Anzahl g von Servicezeitpunkten innerhalb eines Zeitfensters, welches bis zu d Schritte nach dem Eingabezeitpunkt des Knotens reicht.
Die erzielten Resultate der Analysen zeigen eine Abhängigkeit des Competitive Ratios von dem Verhältnis der Parameter d und g auf. Deshalb gelten die Konstruktionen zur allgemeinen unteren Schranke nur für begrenzte Wertebereiche, die jedoch in ihrer Vereinigung alle Parameterkombinationen abdecken. Die Darstellung und Auswertung der Gegenspielerstrategien umfaßt den ersten Teil dieses Kapitels.
Für einen Bereich von Parameterkombinationen sind bessere Competitive Ratios als der Wert von 1.5 bekannt, der sich aus dem einfachen ORSM-Problem ergibt. Leider sind die kombinatorische Vielfalt und die auftretenden Abhängigkeiten so umfangreich, daß für die Modellinstanzen keine theoretischen Analysen gelungen sind. Es war jedoch möglich für Modellinstanzen mit kleinen Werten der Parameter alle möglichen Eingaben auf geschickte Weise zu prüfen und somit die Leistungsfähigkeit von Online-Algorithmen für diese Modelle mittels Computereinsatz zu ermitteln. Wie dieser Ansatz funktioniert und mit welchen Techniken die Implementierung so effizient gestaltet wurde, daß für mehr als nur die minimalen Parameter Resultate erzielt werden konnten, ist Gegenstand des zweiten Teilkapitels. Dem schließt sich eine Darstellung der untersuchten Online-Algorithmen mit den Ergebnissen ihrer Computeranalyse an. Zusammenfassend werden die Erkenntnisse über Designprinzipien für Online-Algorithmen der in dieser Arbeit behandelten Problemklasse diskutiert.

8.1 Untere Schranken

Wie sich im Laufe der Untersuchungen der ORSM-Modellvariante mit konstantem Knotengrad g und begrenzter Kantenlänge d gezeigt hat, ist der Competitive Ratio in hohem Maße von dem Verhältnis der beiden Modellparameter g und d abhängig. Dieses Verhalten spiegelt sich auch in der Tatsache wider, daß eine Gegenspielerstrategie zum Nachweisen von unteren Schranken verschiedene Eingabestrukturen verwendet. Deshalb wird an dieser Stelle ein System, bestehend aus verschiedenen Konstruktionen für untere Schranken, vorgestellt, welches die besten bekannten Resultate erreicht. Die verschiedenen Eingabestrukturen basieren auf Verallgemeinerungen der Beweisidee zu Satz 3, die in unterschiedlicher Form miteinander kombiniert werden. Zuerst werden verallgemeinerte Grundstrukturen definiert, um innerhalb der darauf folgenden Beweise einfacher und deutlicher argumentieren zu können.

Ein *Block* besteht aus einer Menge von Requestknoten und einer gleichmächtigen Menge von Serverknoten, die so durch Kanten verbunden sind, daß für diese Graphenstruktur ein perfektes Matching existiert. Alle zu den Requestknoten adjazenten Serverknoten sind Elemente dieser Serverknotenmenge. Deshalb muß die Requestknotenmenge mindestens g Knoten beinhalten.
Mit Hilfe derartiger Strukturen können die Kanten anderer Requestknoten, die ebenfalls zu Knoten der Servermenge des Blockes inzident sind, „blockiert“ werden. Sie können also nicht in das Online-Matching aufgenommen werden bzw. nur unter Verlust einer Matchingkante aus dem Block, so daß die Kardinalität des Gesamtmatchings dadurch nicht erhöht wird.
Die nächste Grundstruktur ist eine *Entscheidungssituation*, wie sie schon aus Satz 3 bekannt ist (siehe Skizze 20).

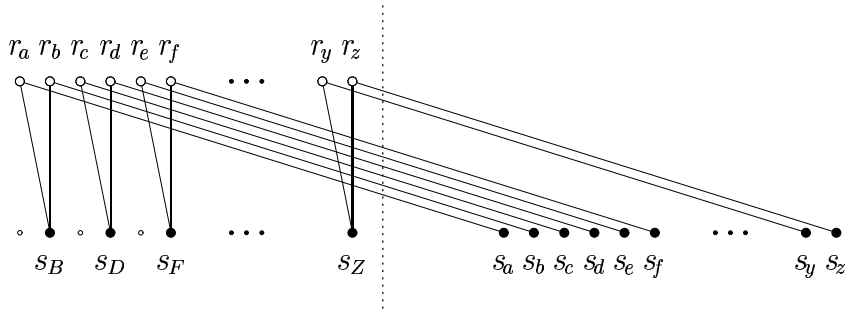


Skizze 20: Eine einfache Entscheidungssituation zum Zeitpunkt i; s_i ist nur zu r_a und r_b adjazent.

Zum Zeitpunkt i muß ein Online-Algorithmus entscheiden, ob die Kante $\{r_a, s_i\}$ oder $\{r_b, s_i\}$ in das Online-Matching aufgenommen wird. Im ersten Fall bleibt der Serverknoten s_a und im zweiten Fall s_b frei, wenn später keine weiteren Kanten zu diesen Knoten hinzukommen. Gleichzeitig kann der jeweils andere Serverknoten blockiert werden, woraus sich ein Verlust von einer Matchingkante gegenüber einer optimalen Lösung ergibt (siehe auch den Beweis zu Satz 3). Der Sonderfall, daß s_i bezüglich des Online-Matchings frei bleibt, hat ebenfalls den Verzicht auf

eine Matchingkante zur Folge. Alle $g-2$ Kanten der Requestknoten, die nicht Teil der oben gezeigten Struktur sind, müssen blockiert werden.

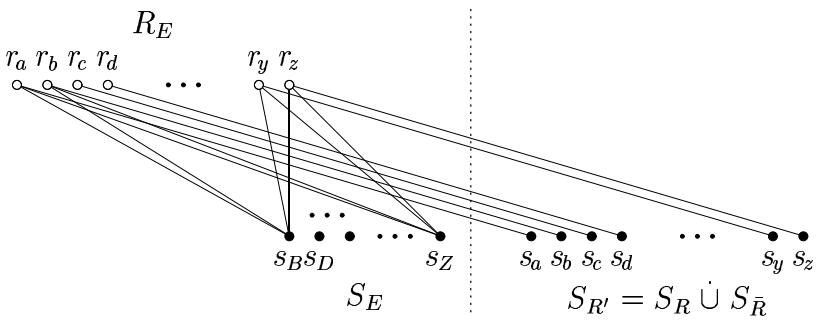
Bei ausreichend großem Wert für den Parameter d, ist es sinnvoll mehrere einfache Entscheidungssituationen ineinander zu verschränken, um mit einem Block möglichst geringer Größe alle übriggebliebenen Kanten zu blockieren. Dadurch wird die untere Schranke im Competitive Ratio verbessert, denn das Verhältnis der Entscheidungssituationen (entspricht den vom Online-Algorithmus verlorenen Matchingkanten) zu der Knotenanzahl in Blöcken erhöht sich.
Solche *ineinander verschränkte Entscheidungssituationen* haben einen Aufbau, wie ihn Skizze 21 zeigt.



Skizze 21: Die Konstruktion ineinander verschränkter Entscheidungssituationen.

Alle weiteren $g-2$ Kanten der Requestknotenmenge $\{r_a, r_b, \ldots, r_z\}$ müssen blockiert werden. Dabei können jedoch einige Kanten zusätzlich zu Knoten der Menge $\{s_a, s_b, \ldots, s_z\}$ inzident sein. Nachdem die Entscheidung über den Serverknoten s_B gefallen ist, besitzt nur noch einer der Knoten s_a und s_b eine Kante zu einem freien Requestknoten. Zu diesem Serverknoten kann nun je eine Kante der Requestknoten r_c und r_d (sowie aller weiteren Requestknoten $\{r_e, \ldots, r_z\}$) inzident sein, die somit ebenfalls blockiert sind. Nach Bearbeitung des Knotens s_D stehen dann zwei Knoten aus $\{s_a, s_b, s_c, s_d\}$ als Nachbarknoten der weiteren Requestknoten $\{r_e, r_f, \ldots, r_z\}$ zum Blockieren zur Verfügung, etc. Es ist garantiert, daß die Hälfte der Knoten der Menge $\{s_a, s_b, \ldots, s_z\}$ durch einen Online-Algorithmus nicht genutzt werden können (oder alternativ im Austausch einige der Knoten $\{s_B, s_D, \ldots, s_Z\}$ nicht in Matchingkanten auftreten) und die anderen Serverknoten nach Bearbeitung von s_Z blockiert werden. Die optimale Lösung besitzt jedoch alle nichtisolierten Serverknoten als Endknoten im Online-Matching. In der Skizze wird das Ende der Entscheidungen und damit der Zeitpunkt, ab dem die Blockade der übriggebliebenen Kanten beginnen kann, durch eine gestrichelte, vertikale Linie symbolisiert.

Eine weitere Variante einer Verallgemeinerung der Entscheidung ist eine *Multi-Entscheidungssituation*. Ihr Grundaufbau ist in Skizze 22 zu sehen.



Skizze 22: Aufbau einer Multi-Entscheidungssituation.

Sie besteht aus einer geraden Anzahl von Requestknoten $R_E := \{r_a, r_b, \ldots, r_z\}$, die mit einer halb so großen Menge von Serverknoten $S_E := \{s_B, s_D, \ldots, s_Z\}$ mit bis zu $g-1$ Kanten pro Requestknoten verbunden sind. Eine Kante pro Requestknoten wird durch das Knotenpaar $\{r_\alpha, s_\alpha\}$ mit $\alpha \in \{a, b, \ldots, z\}$ gebildet. Sollte die Knotenmenge S_E weniger als $g-1$ Elemente aufweisen, so müssen die weiteren Kanten der Requestknoten aus R_E mit Serverknoten in Blöcken verschieden von $S_{R'} := \{s_a, s_b, \ldots, s_z\}$ verbunden werden. Falls $|S_E| \geqq g$ gilt, so kann die Knotenmenge S_E auch früher beginnen, d. h. der Zeitpunkt von r_z liegt nach dem Zeitpunkt des Knotens s_B.
Nachdem die Matchingentscheidungen über die Knotenmenge S_E getroffen wurden, besitzt die Hälfte der Knoten aus $S_{R'}$ keine freien Nachbarn aus R_E, oder es sind einige Knoten aus S_E nicht für das Matching genutzt worden. Diese Menge von nun isolierten Serverknoten sei mit $S_{\bar R}$ bezeichnet, während die möglichen Matchingpartner der noch freien Knoten aus R_E durch die Menge S_R notiert werden ($S_{R'} = S_R \dot\cup S_{\bar R}$; S_R sei genau $|S_{R'}|/2 = |R_E|/2$ groß, möglicherweise überzählige Elemente werden mit in $S_{\bar R}$ aufgenommen). Wird die Menge S_R Teil eines Blockes, der nach Abschluß der Entscheidung über die Nutzung von s_Z aufgebaut wird, so werden $|R_E|/2$ Kanten verschiedener Knoten der Menge R_E blockiert. Ein Gegenspieler garantiert mit dieser Konstruktion, daß in dem Online-Matching $|R_E|/2 = |S_R|$ weniger Matchingkanten existieren als in der optimalen Lösung.

Bei der zuletzt vorgestellten Verallgemeinerung wurde gleichzeitig eine im folgenden häufig benutzte Notation eingeführt. Knotenmengen mit identischer Funktion werden auch in ihrer Bezeichnung zu Mengen zusammengefaßt, und zwar mit

R_E – Menge von Requestknoten in Entscheidungssituationen,

S_E – Menge von Serverknoten, bei denen die Matchingentscheidungen getroffen werden müssen ($|S_E| = |R_E|/2$),

$S_{R'}$ – Resultatmenge von Serverknoten der Entscheidungssituation ($|S_{R'}| = |R_E|$), unterteilt in:

S_R – Menge von Serverknoten, die nach den Entscheidungen zu jeweils einem freien Requestknoten der Menge R_E adjazent sind und später durch den Gegenspieler blockiert werden ($|S_R| = |R_E|/2$) und

$S_{\bar{R}} := S_{R'} \setminus S_R$ – Serverknotenmenge, die vom Online-Algorithmus nicht genutzt wird, obwohl alle diese Elemente im optimalen Matching enthalten sind bzw. $\forall s \in S_{\bar{R}} : \quad s \mathring{\in} M_{\mathsf{ALG}} \Rightarrow \exists! \, s' \in S_E : \; s' \mathring{\notin} M_{\mathsf{ALG}}$.

Hierzu kommen noch die Bezeichnungen R_B und S_B für Request- bzw. Serverknotenmengen in Blöcken und R_F bzw. S_F für Mengen von isolierten Knoten.

Nach diesen Vorbereitungen werden nun die Sätze vorgestellt, die das System der Gegenspielerstrategie für die unteren Schranken des Competitive Ratios des ORSM-Modells mit konstantem Knotengrad g und begrenzter Kantenlänge d aufspannen.
Jeder Wert einer unteren Schranke gilt dabei ab der notwendigen unteren Begrenzung von d für *alle* größeren Parameterwerte, obwohl die Gegenspielerkonstruktion, so wie im folgenden angegeben, nur bis zu einer Obergrenze des Parameters d aufgebaut werden kann. Da für größere Parameterwerte andere und schärfere Gegenspielerkonstruktionen vorhanden sind, ist es jedoch nicht zweckmäßig die angegebenen Obergrenzen zu überschreiten.

Satz 28:

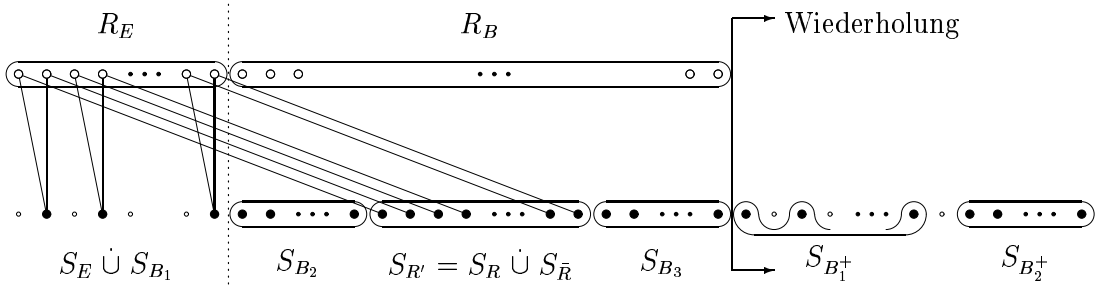
Jeder deterministische Online-Algorithmus ALG für das ORSM-Problem mit konstantem Knotengrad g und begrenzter Kantenlänge d besitzt für $g \geqq 2$ und $d \in [g, 2g-2]$ einen Competitive Ratio von mindestens

$$\frac{2d+1}{d+g} \; .$$

Beweis: Die Gegenspielerstrategie basiert auf sich wiederholenden und überlappenden Phasen. Jede Phase beginnt mit ineinander verschränkten Entscheidungssituationen und hat einen Aufbau, wie ihn Skizze 23 zeigt. Dabei gelten die folgenden Grenzen der Parameter und Größen der Mengen:

$$g, d \in \mathbb{N}, \quad g \geqq 2, \quad d \in [g, 2g-2]$$

$$\begin{aligned} |R_E| = |S_{R'}| &= 2(d-g+1) \\ |R_B| &= 2g-1 \\ |S_E| = |S_R| = |S_{\bar{R}}| = |S_{B_1}| &= d-g+1 \\ |S_{B_2}| &= 2g-d-2 \\ |S_{B_3}| &= 2g-d-1 \end{aligned}$$



Skizze 23: Struktur der Gegenspielereingabe für $g \geqq 2$, $d \in [g, 2g-2]$.

Die Mengen S_E und S_{B_1} sind im Reißverschlußprinzip ineinander verzahnt, d. h. die ungeraden Knotennummern gehören der Menge S_{B_1} an und die geraden Knotennummern stellen die Menge S_E dar. Die Mengen $S_{B_1^+}$ und $S_{B_2^+}$ sind die korrespondierenden Block-Servermengen der folgenden Phase, die jedoch schon in der gezeigten Phase mitbenutzt werden.
Die in Skizze 23 nicht dargestellten Kanten verbinden die Knotenmenge R_E mit Knoten aus S_{B_1}, S_{B_2} und Elementen von S_R, soweit diese Menge schon durch getroffene Entscheidungen innerhalb der Phase definiert ist sowie Knoten der Menge R_B mit Knoten aus S_{B_2}, S_R, S_{B_3}, $S_{B_1^+}$ und $S_{B_2^+}$.
Somit werden zu den Zeitpunkten der Serverknoten aus S_E vom Online-Algorithmus ALG $d-g+1$ Entscheidungen getroffen, aus denen gemäß dieser Gegenspielerstrategie die Menge $S_{\bar R}$ von $d-g+1$ nicht genutzten Serverknoten folgt. Die $d-g+1$ Serverknoten des initialen Blockes S_{B_1} der ersten Phase können durch einen vorgelagerten Block der Größe g dargestellt werden.[28] Da dies nur ein einziges Mal geschieht, sich die Phasen aber immer wiederholen, haben die daraus entstehenden zusätzlichen g Matchingkanten keinen Einfluß auf den Competitive Ratio. Um ihn zu bestimmen, genügt die Auswertung einer einzelnen Phase. Diese besteht aus $|R_E| + |R_B| = 2d+1$ Request- und ebensovielen Serverknoten, von denen ALG jedoch $|S_{\bar R}| = d-g+1$ nicht in das Matching M_{ALG} aufnehmen kann, obwohl ein perfektes Matching existiert. Daraus folgt:

$$\mathcal{R}(\mathsf{ALG}) \quad \geqq \quad \frac{2d+1}{d+g} \quad .$$

■ Satz 28

Satz 29:

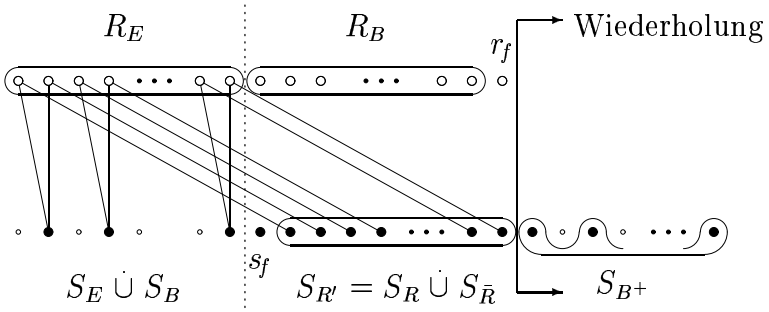
Jeder deterministische Online-Algorithmus ALG für das ORSM-Problem mit konstantem Knotengrad g und begrenzter Kantenlänge d besitzt für $g \geqq 2$ und $d \geqq 2g-1$ einen Competitive Ratio von mindestens $4/3$.

[28] Es gilt: $d-g+1 < g$ für $g \leqq d \leqq 2g-2$.

Beweis: Die Gegenspielerstrategie basiert auf sich wiederholenden und überlappenden Phasen. Jede Phase beginnt mit ineinander verschränkten Entscheidungssituationen und hat einen Aufbau, wie ihn Skizze 24 zeigt. Dabei gelten die folgenden Grenzen der Parameter und Größen der Mengen:

$$g, d \in \mathbb{N}, \quad g \geqq 2, \quad d = 2g - 1$$

$$\begin{array}{rcccccc} & |R_E| & = & |R_B| & = & |S_{R'}| & = & 2g-2 \\ |S_E| & = & |S_R| & = & |S_{\bar{R}}| & = & |S_B| & = & g-1 \end{array}$$



Skizze 24: Struktur der Gegenspielereingabe für $g \geqq 2$, $d = 2g - 1$.

Die Mengen S_E und S_B sind ineinander verzahnt, d. h. die ungeraden Knotennummern gehören der Menge S_B an und die geraden Knotennummern stellen die Menge S_E dar. Die Menge S_{B^+} ist die korrespondierende Block-Servermenge der folgenden Phase, die jedoch schon in der gezeigten Phase mitbenutzt wird. Die Knoten r_f und s_f bleiben isoliert.
Die in Skizze 24 nicht dargestellten Kanten verbinden die Knotenmenge R_E mit Knoten aus S_B und Elementen von S_R, soweit diese Menge schon durch getroffene Entscheidungen innerhalb der Phase definiert ist. Weiterhin werden Knoten der Menge R_B mit Knoten aus S_R und S_{B^+} verbunden.
Somit werden zu den Zeitpunkten der Serverknoten aus S_E vom Online-Algorithmus ALG $g - 1$ Entscheidungen getroffen, aus denen gemäß dieser Gegenspielerstrategie die Menge $S_{\bar{R}}$ von $g - 1$ nicht genutzten Serverknoten folgt. Die $g - 1$ Serverknoten des initialen Blockes S_B der ersten Phase können durch einen vorgelagerten Block der Größe g dargestellt werden. Da dies nur ein einziges Mal geschieht, sich die Phasen aber immer wiederholen, haben die daraus entstehenden zusätzlichen g Matchingkanten keinen Einfluß auf den Competitive Ratio. Um ihn zu bestimmen, genügt die Auswertung einer einzelnen Phase. Diese besteht aus $|R_E| + |R_B| = 4g - 4$ nichtisolierten Request- und ebensovielen nichtisolierten Serverknoten, von denen ALG jedoch $|S_{\bar{R}}| = g - 1$ nicht in das Matching M_{ALG} aufnehmen kann, obwohl ein perfektes Matching existiert. Daraus folgt:

$$\mathcal{R}(\mathsf{ALG}) \geqq \frac{4g-4}{3g-3} = \frac{4}{3}. \qquad \blacksquare \text{ Satz 29}$$

Satz 30:

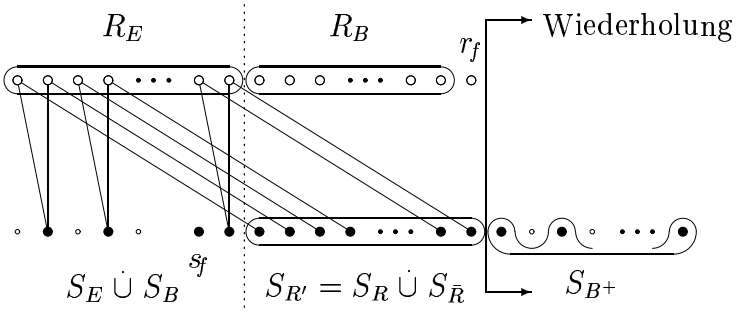
Jeder deterministische Online-Algorithmus ALG für das ORSM-Problem mit konstantem Knotengrad g und begrenzter Kantenlänge d besitzt für $g \geqq 2$ und $d \geqq 2g$ einen Competitive Ratio von mindestens

$$\frac{4g-1}{3g-1} \ .$$

Beweis: Die Gegenspielerstrategie basiert auf sich wiederholenden und überlappenden Phasen. Jede Phase beginnt mit ineinander verschränkten Entscheidungssituationen und hat einen Aufbau, wie ihn Skizze 25 zeigt. Dabei gelten die folgenden Grenzen der Parameter und Größen der Mengen:

$$g, d \in \mathbb{N}, \quad g \geqq 2, \quad d = 2g$$

$$\begin{aligned} |R_E| = |S_{R'}| &= 2g \\ |R_B| &= 2g-1 \\ |S_E| = |S_R| = |S_{\bar{R}}| &= g \\ |S_B| &= g-1 \end{aligned}$$



Skizze 25: Struktur der Gegenspielereingabe für $g \geqq 2$, $d = 2g$.

Die Mengen S_E und S_B sind ineinander verzahnt, d. h. die ungeraden Knotennummern gehören der Menge S_B an und die geraden Knotennummern stellen die Menge S_E dar. Die Menge S_{B^+} ist die korrespondierende Block-Servermenge der folgenden Phase, die jedoch schon in der gezeigten Phase mitbenutzt wird. Die Knoten r_f und s_f bleiben isoliert.
Die in Skizze 25 nicht dargestellten Kanten verbinden die Knotenmenge R_E mit Knoten aus S_B und Elementen von S_R, soweit diese Menge schon durch getroffene Entscheidungen innerhalb der Phase definiert ist. Weiterhin werden Knoten der Menge R_B mit Knoten aus S_R und S_{B^+} verbunden.

Somit werden zu den Zeitpunkten der Serverknoten aus S_E vom Online-Algorithmus ALG g Entscheidungen getroffen, aus denen gemäß dieser Gegenspielerstrategie die Menge $S_{\bar{R}}$ von g nicht genutzten Serverknoten folgt. Die $g-1$ Serverknoten des initialen Blockes S_B der ersten Phase können durch einen vorgelagerten Block der Größe g dargestellt werden. Da dies nur ein einziges Mal geschieht, sich die Phasen aber permanent wiederholen, haben die daraus entstehenden zusätzlichen g Matchingkanten keinen Einfluß auf den Competitive Ratio. Um ihn zu bestimmen, genügt die Auswertung einer einzelnen Phase. Diese besteht aus $|R_E| + |R_B| = 4g-1$ nichtisolierten Request- und ebensovielen nichtisolierten Serverknoten, von denen ALG jedoch $|S_{\bar{R}}| = g$ nicht in das Matching M_{ALG} aufnehmen kann, obwohl ein perfektes Matching existiert. Daraus folgt:

$$\mathcal{R}(\mathsf{ALG}) \geqq \frac{4g-1}{3g-1} .$$

■ Satz 30

Satz 31:

Jeder deterministische Online-Algorithmus ALG für das ORSM-Problem mit konstantem Knotengrad g und begrenzter Kantenlänge d besitzt für $g > 2$ und $d > 2g$ einen Competitive Ratio von mindestens

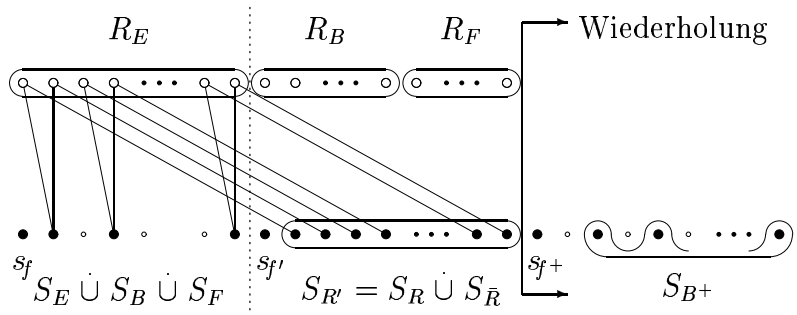
$$\frac{3\lfloor d/2 \rfloor + g - 2}{2\lfloor d/2 \rfloor + g - 2} .$$

Beweis: Die Gegenspielerstrategie basiert auf sich wiederholenden und überlappenden Phasen. Jede Phase beginnt mit ineinander verschränkten Entscheidungssituationen und hat einen Aufbau, wie ihn Skizze 26 zeigt. Dabei gelten die folgenden Grenzen der Parameter und Größen der Mengen:

$$g, d \in \mathbb{N}, \quad g > 2, \quad d > 2g$$

$$\begin{aligned} |R_E| = |S_{R'}| &= 2\lfloor d/2 \rfloor \\ |R_B| &= \lfloor d/2 \rfloor + g - 2 \\ |R_F| &= \lceil d/2 \rceil - g + 2 \\ |S_E| = |S_R| = |S_{\bar{R}}| &= \lfloor d/2 \rfloor \\ |S_B| &= g - 2 \\ |S_F| &= \lfloor d/2 \rfloor - g + 1 \end{aligned}$$

Die Knoten der Menge S_E sind mit dem Knoten s_f, den Knoten der Menge S_B und darauf folgend den Knoten der Menge S_F verzahnt, d. h. die ungeraden Knotennummern gehören sukzessive zu s_f, der Menge S_B und S_F an und die geraden



Skizze 26: Struktur der Gegenspielereingabe für $g > 2$, $d > 2g$; der Knoten s_f existiert nur, falls $d \equiv 1 \bmod 2$.

Knotennummern stellen die Menge S_E dar. Die Menge S_{B+} ist die korrespondierende Block-Servermenge der folgenden Phase, die jedoch schon in der gezeigten Phase mitbenutzt wird. Die Knoten s_f, $s_{f'}$ und die Knoten der Mengen R_F und S_F bleiben isoliert. Der Knoten $s_{f'}$ existiert in der Konstruktion nur, falls der Parameter d ungerade ist, denn $|R_B| + |R_F| = d$ und $|S_{R'}| = 2\lfloor d/2 \rfloor$.
Die in Skizze 26 nicht dargestellten Kanten verbinden die Knotenmenge R_E mit Knoten aus S_B und Elementen von S_R, soweit diese Menge schon durch getroffene Entscheidungen innerhalb der Phase definiert ist. Weiterhin werden die Knoten der Blockmenge R_B mit Knoten aus S_R und S_{B+} verbunden.
Somit werden zu den Zeitpunkten der Serverknoten aus S_E vom Online-Algorithmus ALG $\lfloor d/2 \rfloor$ Entscheidungen getroffen, aus denen gemäß dieser Gegenspielerstrategie die Menge $S_{\bar R}$ von $\lfloor d/2 \rfloor$ nicht genutzten Serverknoten folgt. Die $g-2$ Serverknoten des initialen Blockes S_B der ersten Phase können durch einen vorgelagerten Block der Größe g dargestellt werden. Da dies nur ein einziges Mal geschieht, sich die Phasen aber immer wiederholen, haben die daraus entstehenden zusätzlichen g Matchingkanten keinen Einfluß auf den Competitive Ratio. Um ihn zu bestimmen, genügt die Auswertung einer einzelnen Phase. Diese besteht aus $|R_E| + |R_B| = 3\lfloor d/2 \rfloor + g - 2$ nichtisolierten Request- und ebensovielen nichtisolierten Serverknoten, von denen ALG jedoch $|S_{\bar R}| = \lfloor d/2 \rfloor$ nicht in das Matching M_{ALG} aufnehmen kann, obwohl ein perfektes Matching existiert. Daraus folgt:

$$\mathcal{R}(\mathsf{ALG}) \;\geqq\; \frac{3\lfloor d/2 \rfloor + g - 2}{2\lfloor d/2 \rfloor + g - 2} \;.$$

■ Satz 31

Bei genauer Betrachtung der Konstruktionen der Eingabegraphen in den Beweisen zu den Sätzen 28, 29, 30 und 31 läßt sich feststellen, daß es sich um Fortsetzungen einer Idee unter verschiedenen Randbedingungen handelt. Besonders deutlich wird dieser Fakt, wenn der Parameter g konstant gehalten wird, und die

Graphkonstruktionen der jeweils besten Schranke für wachsende Werte von d beobachtet werden. Dann gehen sowohl die Strukturen der verschiedenen Graphen, als auch die Werte der damit gezeigten Competitive Ratios sanft ineinander über. Eine etwas andere Graphkonstruktion liegt der Gegenspielerstrategie des nächsten Satzes zugrunde.

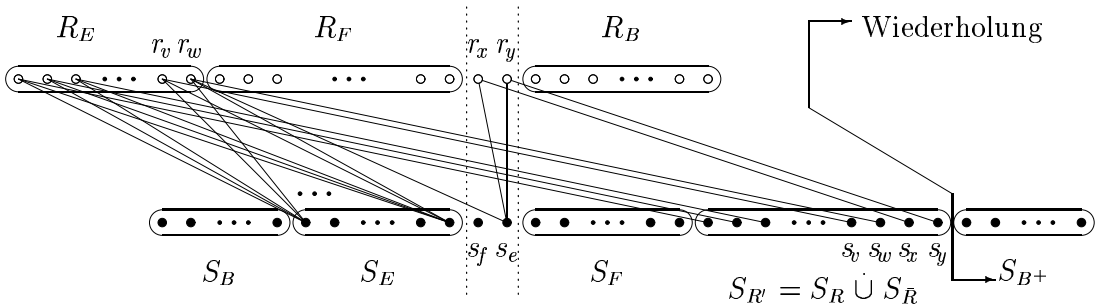
Satz 32:

Jeder deterministische Online-Algorithmus ALG *für das* ORSM*-Problem mit konstantem Knotengrad g und begrenzter Kantenlänge d besitzt für $g \geqq 4$ und $d \in [2g+3, 4g-4]$ einen Competitive Ratio von mindestens*

$$\frac{2\lfloor d/2 \rfloor - g + 4}{\lfloor d/2 \rfloor + 2} \quad .$$

Beweis: Die Gegenspielerstrategie basiert ein weiteres Mal auf sich wiederholenden und überlappenden Phasen. Jede Phase beginnt mit einer Multi-Entscheidungssituation und hat einen Aufbau, wie ihn Skizze 27 zeigt. Dabei gelten die folgenden Grenzen der Parameter und Größen der Mengen:

$$\begin{aligned} g, d \in \mathbb{N}, \quad g \geqq 4, \quad & d \in [2g+3, 4g-4] \\ |R_E| &= 2\lfloor d/2 \rfloor - 2g + 2 \\ |R_F| &= g - 3 \\ |R_B| &= g \\ |S_B| &= 2g - \lfloor d/2 \rfloor - 2 \\ |S_E| &= \lfloor d/2 \rfloor - g + 1 \\ |S_F| &= g - 1 \\ |S_{R'}| &= 2\lfloor d/2 \rfloor - 2g + 4 \\ |S_R| = |S_{\bar{R}}| &= \lfloor d/2 \rfloor - g + 2 \end{aligned}$$



Skizze 27: Struktur der Gegenspielereingabe für $g \geqq 4$, $d \in [2g+3, 4g-4]$.

Die Menge S_{B^+} ist die korrespondierende Block-Servermenge der folgenden Phase, die jedoch schon in der gezeigten Phase mitbenutzt wird. Die Knotenmengen R_F und S_F sowie die Requestknoten, die auf R_B bis zum Ende der Phase folgen, und s_f bleiben isoliert.
Die in Skizze 27 nicht dargestellten bzw. angedeuteten Kanten verbinden die Knoten der Menge R_B mit den Knoten der Mengen S_R und S_{B^+}. Die Knoten der Mengen R_E und S_E bilden einen vollständigen, bipartiten Teilgraphen. Zur Sicherstellung des Grades g der Knoten aus R_E werden selbige auch mit Knoten der Menge S_B durch Kanten verbunden. Nachdem der Online-Algorithmus ALG über alle Serverknoten aus S_E die Entscheidung bezüglich des Matchings M_{ALG} getroffen hat, veröffentlicht der Gegenspieler die Knoten r_x und r_y. Deren weitere $g-2$ Kanten sind mit den Knoten der zu diesem Zeitpunkt eindeutig definierten Menge $S_R \backslash \{s_w, s_x, s_y\}$ und mit Knoten der Menge S_{B^+} inzident. Eine Verbindung von $\{r_x, r_y\}$ mit s_w durch eine Kante muß unterbunden werden, um das Resultat eines nicht genutzten Serverknotens aus $\{s_x, s_y\}$ durch die Entscheidung an s_e zu garantieren. Es gilt weiterhin $|S_R \setminus \{s_w, s_x, s_y\}| \geqq \lfloor d/2 \rfloor - g$.
Mit dieser Konstruktion stellt ein Gegenspieler sicher, daß aus den Entscheidungen des Online-Algorithmus ALG zu den Zeitpunkten der Serverknoten aus S_E und des Knotens s_e die Menge $S_{\bar{R}}$ von $\lfloor d/2 \rfloor - g + 2$ nicht genutzten Serverknoten folgt. Die $2g - \lfloor d/2 \rfloor - 2$ Serverknoten[29] des initialen Blockes S_B der ersten Phase können durch einen vorgelagerten Block der Größe g dargestellt werden. Da dies nur ein einziges Mal geschieht, sich die Phasen aber immer wiederholen, haben die daraus entstehenden zusätzlichen g Matchingkanten keinen Einfluß auf den Competitive Ratio. Um ihn zu bestimmen, genügt die Auswertung einer einzelnen Phase. Diese besteht aus $|R_E| + |\{r_x, r_y\}| + |R_B| = 2\lfloor d/2 \rfloor - g + 4$ nichtisolierten Request- und ebensovielen nichtisolierten Serverknoten ($|S_E| + |\{s_e\}| + |S_B| + |S_{R'}|$), von denen ALG jedoch $|S_{\bar{R}}| = \lfloor d/2 \rfloor - g + 2$ nicht in das Matching M_{ALG} aufnehmen kann, obwohl ein perfektes Matching existiert. Daraus folgt:

$$\mathcal{R}(\mathsf{ALG}) \geqq \frac{2\lfloor d/2 \rfloor - g + 4}{\lfloor d/2 \rfloor + 2} .$$

■ Satz 32

Bei den Sätzen 28 bis 31 steigen die Terme der unteren Schranken des Competitive Ratios immer an, d. h. ein Satz, dessen Gültigkeitsbereich einen größeren Wert für d fordert, dominiert in seinem Ergebnis alle Gegenspielerkonstruktionen für kleinere Parameter d, auch wenn diese Konstruktionen weiterhin möglich sind. Eine derartige Dominanzbeziehung liegt zwischen den beiden Termen der unteren Schranken aus den Sätzen 31 und 32 nicht vor. Erst durch Auflösen der quadratischen Ungleichung

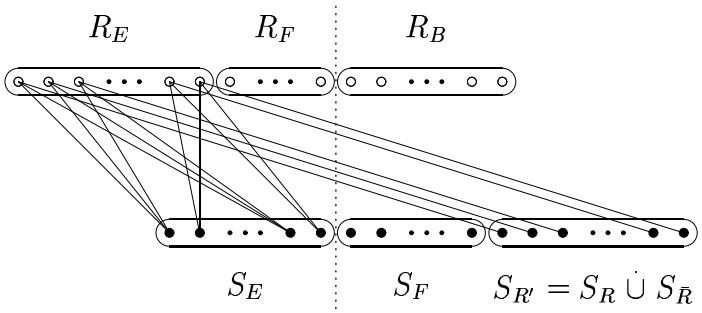
$$\frac{3\lfloor d/2 \rfloor + g - 2}{2\lfloor d/2 \rfloor + g - 2} \leqq \frac{2\lfloor d/2 \rfloor - g + 4}{\lfloor d/2 \rfloor + 2}$$

[29] Unter den Nebenbedingungen der Parameter ist diese Anzahl zwischen 0 und g.

nach dem Parameter d ergibt sich, daß die Gegenspielerstrategie nach Satz 32 mindestens so hohe Werte für die untere Schranke wie Satz 31 ergibt, sobald gilt:

$$d \geqq 2 \left\lceil \frac{g}{2} + \sqrt{\frac{5}{4}g^2 - 4g + 4} \right\rceil .$$

Wird in den Term der unteren Schranke aus Satz 32 der Wert $d = 4g-4$ eingesetzt, so ergibt sich 1.5 als Resultat. Diese untere Schranke gilt ebenfalls für alle größeren Werte von d, denn ein Gegenspieler muß von der maximalen Kantenlänge keinen Gebrauch machen. Da aus Satz 6 auch für Modellvariante dieses Kapitels eine obere Schranke im Competitive Ratio von 1.5 folgt, liegt folglich für $d \geqq 4g - 4$ (und $g \geqq 4$) eine exakte Analyse vor.
Für die beiden kleineren Werte $g \in \{2, 3\}$ werden in den Sätzen 34 und 35 noch spezielle Gegenspielerstrategien folgen. Zuvor soll jedoch eine gegenüber Satz 32 stark vereinfachte Eingabe gezeigt werden, die sehr deutlich die Basisstruktur, wie im Beweis zu Satz 3, aufzeigt. Sie beweist ab $d \geqq 4g - 3$ und $g \geqq 2$ eine untere Schranke im Competitive Ratio von 1.5 .
Ein Gegenspieler wiederholt permanent die Eingabe, wie in Skizze 28 zu sehen. Sie besteht aus einer Multi-Entscheidungssituation (Knotenmenge R_E und S_E sowie den resultierenden Mengen S_R und $S_{\bar{R}}$), der ein einfacher Block der Größe g (vollständiger bipartiter Teilgraph zwischen den Knotenmengen R_B und S_R) folgt.



Skizze 28: Struktur der Gegenspielereingabe für $g \geqq 2$, $d \geqq 4g - 3$.

Es gilt $|R_E| = |S_{R'}| = 2g$, $|S_E| = |S_R| = |S_{\bar{R}}| = g$, $|S_F| = g - 1$ und $|R_F| = g - 2$, woraus folgt, daß $|R_E| + |R_B| = 3g$ nichtisolierte Requestknoten und ebensoviele nichtisolierte Serverknoten ($|S_E| + |S_{R'}|$) in der Konstruktion des Eingabegraphen vorhanden sind. Der Gegenspieler stellt nach Bearbeitung der Menge $|S_E|$ sicher, daß alle Kanten zu Knoten aus S_R blockiert werden. Während die optimale Lösung ein perfektes Matching der Kardinalität $3g$ in der durch Kanten verbundenen Graphkomponente bestimmt, kann ein Online-Algorithmus ALG $|S_{\bar{R}}| = g$ nichtisolierte Serverknoten nicht nutzen. Da diese Konstruktion beliebig oft wiederholt werden kann, ergibt sich das Korollar:

Korollar 33:

Jeder deterministische Online-Algorithmus ALG *für das* ORSM*-Problem mit konstantem Knotengrad g und begrenzter Kantenlänge d besitzt für $g \geqq 2$ und $d \geqq 4g-3$ einen Competitive Ratio von mindestens $3/2 = 1.5$.*

Für kleine Werte des Parameters g sind noch zwei besondere Konstruktionen von Gegenspielern bekannt, die die Werte der unteren Schranke nochmals verbessern.

Satz 34:

Jeder deterministische Online-Algorithmus ALG *für das* ORSM*-Problem mit konstantem Knotengrad $g = 2$ und begrenzter Kantenlänge $d = 4$ besitzt einen Competitive Ratio von mindestens $10/7$.*

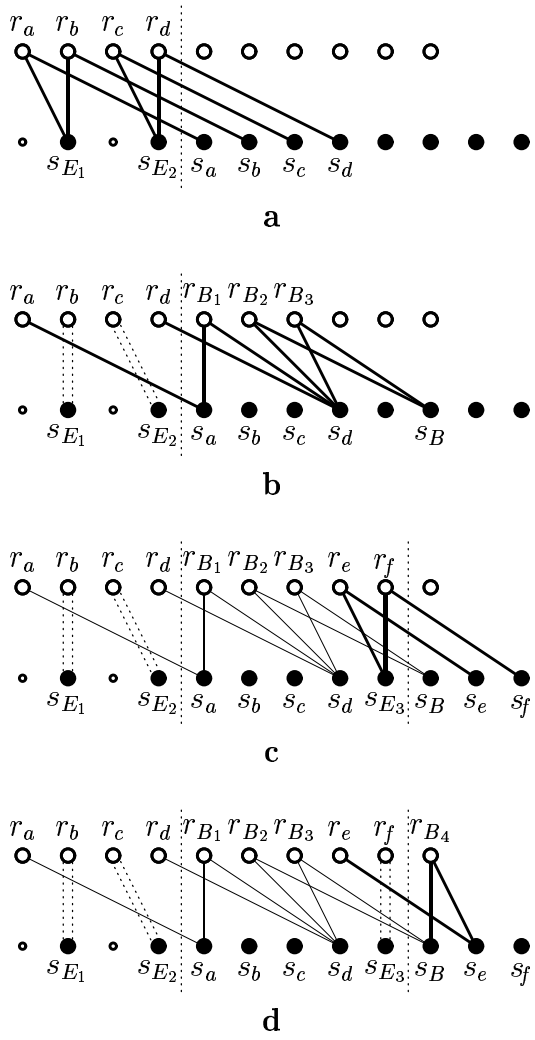
Beweis: Die Gegenspielerstrategie beginnt die Eingabe mit zwei ineinander verschränkten Entscheidungssituationen, wie in Skizze 29, Teil a zu sehen. Der Online-Algorithmus ALG bestimmt die Nutzung der Serverknoten s_{E_1} und s_{E_2}, worauf mindestens zwei der vier Serverknoten $\{s_a, s_b, s_c, s_d\}$ weiterhin mit einem freien Requestknoten adjazent sind. Für die folgenden Skizzen sei diese Menge S_R o. B. d. A. aus den Knoten s_a und s_d bestehend. Der Gegenspieler erweitert den Eingabegraphen daraufhin um einen Block der Größe Drei (siehe Skizze 29, Teil b) und ergänzt eine weiter Entscheidungssituation am Knoten s_{E_3} (Skizze 29, Teil c). Deren resultierende Restkante (hier sei o. B. d. A. $\{r_e, s_e\}$ angenommen) wird am Ende durch den Requestknoten r_{B_4} blockiert (Skizze 29, Teil d).
Ein perfektes Matching dieser Graphkomponente hat eine Kardinalität von 10, während der Online-Algorithmus ALG drei Serverknoten — je einer aus den Mengen $\{s_a, s_b\}$, $\{s_c, s_d\}$ und $\{s_e, s_f\}$ — nicht in das Online-Matching M_{ALG} aufnehmen kann. Durch beliebig häufige Wiederholung dieser Eingabestrategie ergibt sich die untere Schranke im Competitive Ratio von

$$\mathcal{R}(\mathsf{ALG}) \;\geqq\; \frac{10}{7} \;.$$

■ Satz 34

Für die Kantenlänge von $d = 5$ erreicht die Eingabestrategie nach Korollar 33 für $g = 2$ die maximal mögliche untere Schranke des Competitive Ratios von 1.5 .

Auch im Falle $g = 3$ kann die 1.5-Schranke schon für die geringere Kantenlänge von $d = 8$ erzielt werden. Dazu wird die Gegenspielerstrategie aus Satz 32 benutzt, die bei dieser Parameterkombination den Sonderfall $|S_B| = 0$ aufweist. Deshalb benötigt der kritische Knoten r_x keine Kante mehr, die einen Serverknoten aus S_{B^+} erreicht (siehe Skizze 27) und die untere Grenze für d verringert sich um Eins auf $d \geqq 2g+2 = 8$. Anderenfalls wäre die Konstruktion nach Satz 32 nicht möglich, da für $g = 3$ und $d \in [2g+3, 4g-4]$ ein undefiniertes Intervall für den Parameter d mit unterer Grenze von 9 und oberer Grenze von 8 folgt.



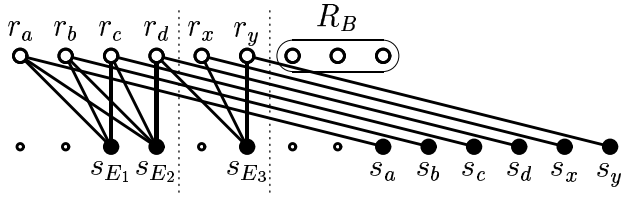
Skizze 29: Gegenspielereingabe für $g = 2$, $d = 4$. Teil a: erste Entscheidungssituationen, im Bsp. gilt $\{r_b, s_{B_1}\}, \{r_c, s_{B_2}\} \in M_{\mathsf{ALG}}$; Teil b: erste Blockierungen; Teil c: weitere Entscheidungssituation, im Bsp. gilt $\{r_f, s_{B_3}\} \in M_{\mathsf{ALG}}$; Teil d: restliche Blockierung.

Satz 35:

Jeder deterministische Online-Algorithmus ALG für das ORSM-Problem mit konstantem Knotengrad $g = 3$ und begrenzter Kantenlänge $d = 8$ besitzt einen Competitive Ratio von mindestens $3/2 = 1.5$.

Beweis: Identisch zu der Konstruktion im Beweis zu Satz 32 beginnt ein Gegenspieler mit einer Multi-Entscheidungssituation, der sofort eine einfache Entscheidungssituation folgt, wie in Skizze 30 zu sehen ist. Nach Bearbeitung der Serverknoten s_{E_1} und s_{E_2} ist sichergestellt, daß mindestens ein Serverknoten s aus der Menge $\{s_a, s_b, s_c\}$ noch einen Requestknoten als Nachbarn hat, der nicht in das Matching M_{ALG} aufgenommen wurde. Die jeweils dritte Kante der Knoten r_x und r_y sind dann inzident zu s. Nachdem auch der Knoten s_{E_3} bearbeitet

wurde, ergänzt der Gegenspieler den Eingabegraphen um einen Block der Größe Drei, der aus R_B sowie der entsprechenden Menge S_R, die s beinhaltet, besteht.



Skizze 30: Struktur der Gegenspielereingabe für $g = 3$, $d = 8$.

Das optimale Matching hat eine Größe von neun Kanten und der Online-Algorithmus ALG kann drei Serverknoten der Menge $S_{R'} = \{s_a, s_b, s_c, s_d, s_x, s_y\}$ nicht nutzen. Da der Gegenspieler diese Grundkonstruktion aller 14 Zeitschritte wiederholen kann, folgt eine untere Schranke für den Competitive Ratio von

$$\mathcal{R}(\mathsf{ALG}) \geqq \frac{3}{2} = 1.5 \ .$$

■ Satz 35

Die Verallgemeinerung dieses Spezialfalles mit $|S_B| = 0$ und $d \geqq 2g + 2$ für größere Werte von g ist zwar möglich, erzielt jedoch gegenüber dem Satz 31 keine verbesserten unteren Schranken für den Competitive Ratio.

Zusammenfassend werden in Tabelle 4 die Gültigkeitsbereiche und die Resultate in den unteren Schranken des Competitive Ratios für das ORSM-Modell mit konstantem Knotengrad g und begrenzter Kantenlänge d der Sätze 28 bis 32 dargestellt.

Satz	28	29	30	31	32
Bereich	$g \leqq d \leqq 2g-2$	$d = 2g-1$	$d = 2g$	$d > 2g$	$2g+3 \leqq d \leqq 4g-3$
untere Schranke	$\frac{2d+1}{d+g}$	$\frac{4}{3}$	$\frac{4g-1}{3g-1}$	$\frac{3\lfloor\frac{d}{2}\rfloor+g-2}{2\lfloor\frac{d}{2}\rfloor+g-2}$	$\frac{2\lfloor\frac{d}{2}\rfloor-g+4}{\lfloor\frac{d}{2}\rfloor+2}$

Tabelle 4: Übersicht der allgemeingültigen Resultate für die unteren Schranken und den Bereich ihrer Gegenspielerkonstruktionen.

Weiterhin gibt Tabelle 5 einen Eindruck davon, wie die in den Sätzen 28 bis 35 vorgestellten Gegenspielerstrategien zusammenwirken, und wie sich die Werte der unteren Schranken entwickeln. Pro Zeile wird der Parameter g um Eins erhöht. In einer Spalte wird nicht der Parameter d konstant gehalten, sondern die Differenz $d - g$. Somit erscheinen in der Tabelle keine Parameterkombinationen mit $g > d$. Dies würde im Widerspruch zum Modell stehen.

$d-g=$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
$g=2$	1.250 5/4	**1.333** 4/3	1.429 10/7	*1.500* 3/2																	
3	1.167 7/6	1.286 9/7	**1.333** 4/3	**1.375** 11/8	1.429 10/7	*1.500* 3/2															
4	1.125 9/8	1.222 11/9	1.300 13/10	**1.333** 4/3	**1.364** 15/11	1.400 7/5	1.417 17/12	*1.429* 10/7	*1.500* 3/2												
5	1.100 11/10	1.182 13/11	1.250 5/4	1.308 17/13	**1.333** 4/3	**1.357** 19/14	1.385 18/13	1.400 7/5	1.400 7/5	*1.444* 13/9	*1.444* 13/9	*1.500* 3/2									
6	1.083 13/12	1.154 15/13	1.214 17/14	1.267 19/15	1.312 21/16	**1.333** 4/3	**1.353** 23/17	1.375 11/8	1.389 25/18	1.389 25/18	*1.400* 7/5	*1.400* 7/5	*1.455* 16/11	*1.455* 16/11	*1.500* 3/2						
7	1.071 15/14	1.133 17/15	1.188 19/16	1.235 21/17	1.278 23/18	1.316 25/19	**1.333** 4/3	**1.350** 27/20	1.368 26/19	1.381 29/21	1.381 29/21	1.391 32/23	1.391 32/23	*1.417* 17/12	*1.417* 17/12	*1.462* 19/13	*1.462* 19/13	*1.500* 3/2			
8	1.062 17/16	1.118 19/17	1.167 7/6	1.211 23/19	1.250 5/4	1.286 9/7	1.318 29/22	**1.333** 4/3	**1.348** 31/23	1.364 15/11	1.375 11/8	1.375 11/8	1.385 18/13	1.385 18/13	1.393 39/28	1.393 39/28	*1.429* 10/7	*1.429* 10/7	*1.467* 22/15	*1.467* 22/15	*1.500* 3/2
9	1.056 19/18	1.105 21/19	1.150 23/20	1.190 25/21	1.227 27/22	1.261 29/23	1.292 31/24	1.320 33/25	**1.333** 4/3	**1.346** 35/26	1.360 34/25	1.370 37/27	1.370 37/27	1.379 40/29	1.379 40/29	1.387 43/31	1.387 43/31	*1.400* 7/5	*1.400* 7/5	*1.438* 23/16	*1.438* 23/16
10	1.050 21/20	1.095 23/21	1.136 25/22	1.174 27/23	1.208 29/24	1.240 31/25	1.269 33/26	1.296 35/27	1.321 37/28	**1.333** 4/3	**1.345** 39/29	1.357 19/14	1.367 41/30	1.367 41/30	1.375 11/8	1.375 11/8	1.382 47/34	1.382 47/34	1.389 25/18	1.389 25/18	*1.412* 24/17
11	1.045 23/22	1.087 25/23	1.125 9/8	1.160 29/25	1.192 31/26	1.222 11/9	1.250 5/4	1.276 37/29	1.300 13/10	1.323 41/31	**1.333** 4/3	**1.344** 43/32	1.355 42/31	1.364 15/11	1.364 15/11	1.371 48/35	1.371 48/35	1.378 51/37	1.378 51/37	1.385 18/13	1.385 18/13
12	1.042 25/24	1.080 27/25	1.115 29/26	1.148 31/27	1.179 33/28	1.207 35/29	1.233 37/30	1.258 39/31	1.281 41/32	1.303 43/33	1.324 45/34	**1.333** 4/3	**1.343** 47/35	1.353 23/17	1.361 49/36	1.361 49/36	1.368 26/19	1.368 26/19	1.375 11/8	1.375 11/8	1.381 29/21
13	1.038 27/26	1.074 29/27	1.107 31/28	1.138 33/29	1.167 7/6	1.194 37/31	1.219 39/32	1.242 41/33	1.265 43/34	1.286 9/7	1.306 47/36	1.324 49/37	**1.333** 4/3	**1.342** 51/38	1.351 50/37	1.359 53/39	1.359 53/39	1.366 56/41	1.366 56/41	1.372 59/43	1.372 59/43
14	1.036 29/28	1.069 31/29	1.100 11/10	1.129 35/31	1.156 37/32	1.182 13/11	1.206 41/34	1.229 43/35	1.250 5/4	1.270 47/37	1.289 49/38	1.308 17/13	1.325 53/40	**1.333** 4/3	**1.341** 55/41	1.350 27/20	1.357 19/14	1.357 19/14	1.364 15/11	1.364 15/11	1.370 63/46
15	1.033 31/30	1.065 33/31	1.094 35/32	1.121 37/33	1.147 39/34	1.171 41/35	1.194 43/36	1.216 45/37	1.237 47/38	1.256 49/39	1.275 51/40	1.293 53/41	1.310 55/42	1.326 57/43	**1.333** 4/3	**1.341** 59/44	1.349 58/43	1.356 61/45	1.356 61/45	1.362 64/47	1.362 64/47
16	1.031 33/32	1.061 35/33	1.088 37/34	1.114 39/35	1.139 41/36	1.162 43/37	1.184 45/38	1.205 47/39	1.225 49/40	1.244 51/41	1.262 53/42	1.279 55/43	1.295 57/44	1.311 59/45	1.326 61/46	**1.333** 4/3	**1.340** 63/47	1.348 31/23	1.354 65/48	1.354 65/48	1.360 34/25
17	1.029 35/34	1.057 37/35	1.083 13/12	1.108 41/37	1.132 43/38	1.154 15/13	1.175 47/40	1.195 49/41	1.214 17/14	1.233 53/43	1.250 5/4	1.267 19/15	1.283 59/46	1.298 61/47	1.312 21/16	1.327 65/49	**1.333** 4/3	**1.340** 67/50	1.347 66/49	1.353 23/17	1.353 23/17

Tabelle 5: Übersicht der Werte der unteren Schranken. Die Einträge in den Spalten sind nach der *Differenz* der Parameter $d-g$ aufgetragen. Zu den verschiedenen Schrifttypen siehe die Anmerkungen im Text auf Seite 138.

Die Tabelleneinträge bestehen aus einem auf drei Nachkommastellen gerundeten Dezimalbruch und einem exakten Wert als gemeinem Bruch darunter. Es ist offensichtlich, daß innerhalb einer Zeile die Werte der unteren Schranken nicht absinken können, denn eine Gegenspielerkonstruktion für einen Wert von d ist auch für beliebig größere Werte der maximalen Kantenlänge gültig. Deshalb wurde in der Tabelle auf die redundanten Einträge von 1.5 im rechten oberen Bereich verzichtet. Innerhalb einer Spalte läßt sich ein Abfallen der unteren Schranken erkennen. Die daraus schlußfolgerbare Verringerung des Competitive Ratios ist plausibel, wenn folgende Tatsachen berücksichtigt werden:
Die Parameterdifferenz $d - g$ gibt an, wieviele Serverknoten im Intervall der Länge $d+1$ möglicher Servicezeitpunkte nicht zur Bearbeitung einer durch einen Requestknoten dargestellten Anfrage benutzt werden können. Mit wachsendem Parameter g wird dann die Bedeutung dieser „Löcher“ im Serviceintervall immer geringer und man nähert sich einem Modell mit dem Charakter von Intervalleingaben. Gleichzeitig wird das sichere Wissen über die Struktur zukünftiger Eingaben verstärkt, da deren Variationsvielfalt relativ zum Anstieg der kombinatorischen Möglichkeiten innerhalb der größeren Intervalle abnimmt.

In Tabelle 5 wird außerdem deutlich, in welchen Parameterbereichen die verschiedenen Gegenspielerstrategien die besten unteren Schranken aufzeigen. Die Wirkung der Schranke von $4/3$ nach Satz 29 für $d = 2g - 1$ ist als Diagonale zu erkennen, die zusätzlich im Fettdruck steht. Unterhalb dieser Diagonalen folgen die Resultate aus Satz 28. Rechts daneben sind die Ergebnisse von Satz 30 ebenfalls im Fettdruck dargestellt. Weiter rechts befinden sich Zahlen in aufrechter Schrift, die aus Satz 31 folgen. Die kursiven Werte im rechten oberen Bereich werden von Satz 32 gezeigt. Die unterstrichenen Zahlen sind die schon behandelten Sonderfälle aus Korollar 33 (für $g = 2$ und $d = 5$) und den Sätzen 34 und 35.

8.2 Bestimmung oberer Schranken durch experimentelle Untersuchungen

8.2.1 Allgemeiner Ansatz

Die Analyse der Leistungsfähigkeit von Online-Algorithmen und damit die Bestimmung oberer Schranken für den Competitive Ratio eines Online-Problems ist nach Definition 1 und den Gleichungen (3) bis (6) eine einfache Aufgabe. Erstens wird eine Potentialfunktion Φ mit der Eigenschaft bestimmt, die Gleichung (4) zu erfüllen. Und zweitens muß diese Eigenschaft sowie die Gültigkeit der Gleichung (6) (siehe Seite 14) nachgewiesen werden. Beide Aufgaben sind jedoch gar nicht leicht zu lösen und es stellt sich ebenfalls die Frage: Gibt es für jedes Online-Problem und jeden -Algorithmus eine Potentialfunktion, mit Hilfe derer eine Analyse nach der Potentialfunktionsmethode möglich ist?

Für die ORSM-Modellvariante mit konstantem Requestknotengrad g und durch d beschränkte Kantenlänge läßt sich feststellen, daß es nur eine begrenzte An-

zahl von verschiedenen Eingaben gibt. Der Teilgraph, welcher in einer Entscheidungssituation als Wissensbasis zur Verfügung steht, ist ebenfalls durch die Kantenlängenbegrenzung in seiner Größe beschränkt. Die sich daraus ergebende endliche Menge von Situationen ermöglicht eine vollständige Modellierung des Spieles zwischen einem Online-Algorithmus und einem Gegenspieler. Weiterhin wird damit die Bestimmung des Competitive Ratios eines Online-Algorithmus möglich. Derartige Computerexperimente für Modellinstanzen des ORSM-Problems mit kleinen Parameterwerten g und d sind geeignet, die Analyse dieser ORSM-Modellvariante zu unterstützen.
Bei genauer Betrachtung fällt auf, daß die durch den Teilgraphen beschriebene Entscheidungssituation auch dem Zustand bzw. der Konfiguration des Online-Algorithmus und des Gegenspielers entspricht. Der Zustand, in dem sich das Gesamtssystem befindet, ist somit ein Paar $(Z_{\mathsf{ALG}}, Z_{\mathsf{ADV}})$, bestehend aus einem Einzelzustand Z_{ALG} des Online-Algorithmus ALG und einem Zustand Z_{ADV} des Gegenspielers. Die Menge aller Zustandspaare seien die Knoten eines Graphen. Seine Kanten werden durch die Entscheidungen der Algorithmen und die damit verbundenen Zustandsübergänge gebildet. Es existiert eine gerichtete Kante $e = ((Z_{\mathsf{ALG},1}, Z_{\mathsf{ADV},1}), (Z_{\mathsf{ALG},2}, Z_{\mathsf{ADV},2}))$, falls es eine Eingabe σ_i gibt, unter der der Online-Algorithmus ALG aus Zustand $Z_{\mathsf{ALG},1}$ in den Zustand $Z_{\mathsf{ALG},2}$ übergeht und es eine Entscheidungsmöglichkeit für den Gegenspieler gibt, bei der er aus $Z_{\mathsf{ADV},1}$ unter der Eingabe σ_i in den Zustand $Z_{\mathsf{ADV},2}$ wechselt. Gleichzeitig sind jeder Kante e die Leistungswerte $Perf_{\mathsf{ALG}}(e)$ und $Perf_{\mathsf{ADV}}(e)$ zugeordnet, die bei den entspechenden Zustandsübergängen der Algorithmen erreicht werden. In diesem Konfigurationsübergangsgraphen gibt es das Paar $(Z_{\mathsf{ALG},0}, Z_{\mathsf{ADV},0})$ der Startzustände, die die leeren Teilgraphen im ORSM-Problem repräsentieren. Für die Modellierung des Systemverhaltens unter allen möglichen Eingabesequenzen ist nur die erste Erreichbarkeitskomponente des gerichteten Graphen, beginnend vom Startzustandspaar $(Z_{\mathsf{ALG},0}, Z_{\mathsf{ADV},0})$, von Interesse. Alle von diesem Knotenpaar nicht erreichbaren Zustandspaare können in dem Spiel zwischen ALG und dem Gegenspieler nie gleichzeitig auftreten, da die beiden Zustände eines solchen Paares verschiedene Eingabesequenzen voraussetzen würden. Wie eine einfache Überlegung zeigt, gibt es von jedem Zustandspaar einen Weg zur Startkonfiguration $(Z_{\mathsf{ALG},0}, Z_{\mathsf{ADV},0})$, da diese bei fortwärender leerer Eingabe immer erreicht wird. Deshalb entspricht der von $(Z_{\mathsf{ALG},0}, Z_{\mathsf{ADV},0})$ erreichbare Teil des Konfigurationsübergangsgraphen seiner ersten starken Zusammenhangskomponente.

Der so definierte Graph modelliert vollständig das Verhalten eines Online-Algorithmus ALG auf allen möglichen Eingabesequenzen und sämtliche mögliche Verhaltensweisen des Gegenspielers auf der selben Eingabesequenz. Um den Competitive Ratio von ALG zu bestimmen, genügt es einen Kreis $circ = (e_1, e_2, \ldots)$ im Konfigurationsübergangsgraphen zu berechnen, für den das Verhältnis der Leistung des Gegenspielers $\sum_{e \in circ} Perf_{\mathsf{ADV}}(e)$ zur Gesamtleistung von ALG auf dieser Eingabe $\sum_{e \in circ} Perf_{\mathsf{ALG}}(e)$ maximiert wird. Dieses Verhältnis ist der exakte Competitive Ratio des Online-Algorithmus ALG, da die mit den Kreiskanten implizit definierte Eingabesequenz einen Zeugen für die untere Schranke darstellt

und durch die Extremaleigenschaft des Kreises dieses Verhältnis zu maximieren, gleichzeitig die schlimmste vom Gegenspieler erzeugbare Eingabe bestimmt wird. Es gilt deshalb:

$$\mathcal{R}(\mathsf{ALG}) \quad = \quad \max_{circ} \frac{\sum_{e \in circ} Perf_{\mathsf{ADV}}(e)}{\sum_{e \in circ} Perf_{\mathsf{ALG}}(e)} \quad .$$

Das Berechnen des gesuchten Kreises *circ* erweist sich als schwierig. Da in einem Graphen exponentiell viele Kreise (bezüglich der Beschreibungsgröße des Graphen) existieren können, kann ein Untersuchen jedes Kreises des Konfigurationsübergangsgraphen zu einer inakzeptablen Laufzeit führen. Ein algorithmischer Ansatz, bei dem optimale Teillösungen bestimmt und später miteinander verknüpft werden, erscheint ebenfalls als nicht erfolgversprechend. Der Grund liegt in den ungünstigen Eigenschaften der Verknüpfungsoperation, die die Verhältniswerte zweier Teilwege zur Berechnung des Gesamtverhältnisses getrennt im Zähler und Nenner addiert. So ist beispielsweise nicht bestimmbar, ob der Weg (a) von u nach v mit dem Leistungsverhältnis $5:3$ oder der Weg (b) mit $16:9$ zu bevorzugen ist, denn je nach Leistungsverhältnis des Rückweges von $5:2$ oder $3:5$ bildet (a) oder (b) den Kreis mit größerem Gesamtverhältnis.

In der Literatur wurde dieses Problem gelöst und es ist dort unter dem Namen *Tramp-Steamer-Problem* [AMO93] oder auch *Minimum-Cost-to-Time-Ratio-Cycle-Problem* bekannt. Die Lösungsprozedur verlangt jedoch, daß das Verhältnis der beiden Summen im oben definierten Kreis geraten wird. Unter der Annahme das gesuchte maximale Verhältnis sei c, wird aus den beiden Leistungswerten einer jeden Kante e ein Gewicht gemäß der Formel

$$w(e) \quad := \quad c \cdot Perf_{\mathsf{ALG}}(e) - Perf_{\mathsf{ADV}}(e)$$

berechnet. Innerhalb dieses gewichteten gerichteten Graphen wird dann einen Kreis mit minimaler Gewichtssumme $W_{\min}$ bestimmt. Gilt $W_{\min} = 0$, dann ist das gesuchte Verhältnis exakt c. Hat der berechnete Kreis eine negative Gewichtssumme, so ist c eine echte untere Schranke für das gesuchte Verhältnis und falls $W_{\min} > 0$, dann verkörpert c eine echte obere Schranke. In [AMO93] wird zur Berechnung des Kreises mit geringstem Gewicht der Floyd-Warshall-Algorithmus empfohlen. Er berechnet für alle Knotenpaare den Weg mit geringstem Gewicht („kürzester Weg"), der keine Kreise, d. h. Knotenwiederholungen, besitzt. Danach kann für jedes Knotenpaar (u, v) die Summe der kürzesten Wege von u nach v und zurück von v nach u gebildet werden. Über diese Gewichtswerte von Kreisen, die die Knoten u und v enthalten, wird minimiert. Sei n die Anzahl von Knoten im Konfigurationsübergangsgraphen. Dann benötigt der Floyd-Warshall-Algorithmus eine Laufzeit von $\Theta(n^3)$. Die darauf folgende Operation zur Bestimmung des minimalen Kreisgewichtes erfordert nur noch eine Laufzeit von $\Theta(n^2)$ und spielt deshalb im asymptotischen Aufwand keine Rolle.

Die bisher beschriebene Lösungsprozedur stellt lediglich einen Test dar, der für einen vorgegebenen Wert c ausgibt, ob das gesuchte Verhältnis getroffen wurde, oder ob c eine untere bzw. obere Schranke dafür darstellt. Der Wert 1 ist eine triviale untere Schranke für jedes zu untersuchende Online-Problem. Beginnend mit 2 kann durch Werteverdoppelung schnell eine obere Schranke gefunden werden, bzw. für die konkrete ORSM-Modellvariante ist schon 1.5 als obere Schranke bekannt. Dann kann mittels einer Binärsuche der exakte Wert für den Competitive Ratio effizient mit der benötigten Genauigkeit berechnet werden.

Im Falle der zu untersuchenden ORSM-Variante können die Leistungswerte eines Schrittes $Perf_{\mathsf{ALG}}(e)$ und $Perf_{\mathsf{ADV}}(e)$ nur die Werte 0 oder 1 annehmen. Der längste, theoretisch mögliche Kreis im Konfigurationsübergangsgraphen hat eine Länge von n Kanten, welche der Knotenanzahl entspricht. Deshalb muß das exakte Verhältnis, das den Competitive Ratio darstellt, ein Bruch der Form a/b mit $a, b \in \{1, 2, \ldots, n\}$ sein. In der Zahlentheorie ist die sortierte Menge $\mathcal{F}_n$ von gekürzten gemeinen Brüchen a/b mit $a < b$ und $a, b \in \{1, 2, \ldots, n\}$ unter dem Begriff der *Farey-Reihe* bekannt ([Far16, Gla79][30]). Die Elemente einer Farey-Reihe liegen zwischen 0 und 1 und stellen einen Teilbaum des *Stern-Brocot-Baumes*[31] dar. Für das konkrete Online-Problem sind nun die Kehrwerte der Farey-Reihe $\mathcal{F}_n$ interessant, die kleiner oder gleich 3/2 sind. Nur aus dieser Menge kann ein Wert für den Competitive Ratio stammen. Aus dem Zusammenhang der Elemente der Farey-Reihe mit dem Stern-Brocot-Baum läßt sich ein Schema ableiten, mit dessen Hilfe der exakte Competitive Ratio durch möglichst wenige Tests bestimmt werden kann.

Die zu Beginn gestellte Frage nach der Existenz von Potentialfunktionen für jedes Online-Problem und jeden -Algorithmus läßt sich auf Basis der vorgestellten Idee positiv beantworten. In [IK96][32] wird ein kurzer Beweis angegeben, der an dieser Stelle wiederholt wird:
Sei ein Online-Problem Π und ein Online-Algorithmus ALG gegeben. Zu jedem Zeitpunkt läßt sich eine Konfiguration bestehend aus dem Zustand von Π (z. B. durch Aufzählung aller Werte von Variablen des Problems Π) und allen Variablen von ALG bestimmen. Jedes Konfigurationspaar $(Z_{\mathsf{ALG}}, Z_{\mathsf{OPT}})$, in dem Z_{ALG} eine Konfiguration von Π während der Ausführung des Online-Algorithmus ALG und Z_{OPT} eine mögliche Problemkonfiguration des optimalen Algorithmus ist, stellt einen Knoten in einem Graphen G dar. Die Graphkanten werden durch die Tupel $((Z_1, Z_2), (Z_3, Z_4))$ gebildet, falls ALG aus dem Zustand Z_1 unter einer Eingabe σ in den Zustand Z_3 wechselt und ein optimaler Algorithmus aus Z_2 unter der

[30] Neuere Darstellungen, z. T. mit historischen Einordnungen und weiteren Referenzen sind in [HW38, Bei66, Dic71, CG96] zu finden.
[31] Der Stern-Brocot-Baum ist eine elegante Form, um alle positiven gekürzten gemeinen Brüche und damit die positiven rationalen Zahlen $\mathbb{Q}_+$ ohne Doppelungen aufzuzählen. Siehe dazu [Ste58, Bro60] oder z. B. [GKP94].
[32] Der Ursprung dieser Überlegung war nicht zurückzuverfolgen. Die Recherchen des Autors innerhalb der aktiven Forschergruppen ergaben keine eindeutige Quellen. Vergleiche auch die Fußnote in [IK96, Seite 534].

selben Eingabe σ in die Konfiguration Z_4 übergehen kann.
Um für ein Profitmaximierungsproblem Π nachzuweisen, daß ALG c-competitive ist, erhält jede Kante $e = ((Z_1, Z_2), (Z_3, Z_4))$ ein Gewicht $w(e)$, welches c-mal dem Profit von ALG beim Übergang von Z_1 nach Z_3 unter σ abzüglich dem vom OPT erreichten Profit ist, falls OPT unter σ von Z_2 nach Z_4 wechselt. Sei (Z_0, Z_0) das Ausgangskonfigurationspaar. Falls ALG c-competitive ist, so kann die erste von (Z_0, Z_0) erreichbare Komponente des Graphen G keinen Kreis mit negativem Gesamtgewicht besitzen. Anderenfalls wäre eine Sequenz von mit den Graphkanten korrespondieren Eingaben bestimmt worden, bei der die Leistung von ALG weniger als 1/c-mal der optimale Profit ist, falls diese Eingabe bei den entsprechenden Systemzuständen erfolgt und beliebig oft wiederholt wird. Das entspräche einer Verletzung der zu beweisende Aussage.
Schon in [EK70] wurde gezeigt, daß für einen gerichteten gewichteten Graphen $G = (V, E, w)$ mit negativen Kantengewichten, aber ohne negative Kreise, eine Potentialfunktion $\Phi : V \to \mathbb{R}$ existiert, so daß sämtliche Kantengewichte $w((u, v))$ durch nichtnegative Gewichte

$$w_{\text{neu}}((u, v)) \quad = \quad \Phi(u) + w((u, v)) - \Phi(v)$$

ersetzt werden können. Die Gewichtssumme von Kreisen bleibt dabei erhalten, ebenso weitere Eigenschaften von Wegen, die bei der Bestimmung des minimalen Kostenflusses[33] von Bedeutung sind. Es ist leicht zu erkennen, daß eine derartige Funktion Φ für den oben definierten Graphen G genau die Eigenschaften besitzt, die für die gesuchte Potentialfunktion in Gleichung (3) gefordert wird. Die algorithmische Lösung dieses Problems wird zu einem späteren Zeitpunkt innerhalb dieses Kapitels diskutiert. Mit den obigen Überlegungen folgt jedoch aus der Existenz des Graphen G für jedes Online-Problem Π und jeden Online-Algorithmus ALG — der u. U. unendlich groß sein kann — die Existenz einer Potentialfunktion für einen Beweis der Aussage: „ALG ist c-competitive."

Weiterhin läßt sich aus diesem Ansatz ableiten, daß das Problem der Analyse von ALG auch durch das Lösen eines linearen Programmes durchgeführt werden kann. Für ein Online-Problem mit endlicher Konfigurationsmenge werden dazu die oben definierten Gleichungen sowie die Gleichungen zum Nachweis der Eigenschaft von ALG aufgestellt (die Potentialfunktionswerte $\Phi(v)$ stellen die Variablen dar) und die Variable c wird minimiert. Für Probleminstanzen mit kleinen Konfigurationsmengen wurden Varianten dieses linearen Programmieransatzes in [KMMO94, LR94] für Analysen benutzt.

Für Modellinstanzen des ORSM-Problems mit konstantem Knotengrad g und begrenzter Kantenlänge d wurden bei kleinen Parameterwerten mit den oben vorgestellten Ideen, kombiniert mit effizienter Algorithmik, Leistungsanalysen von Online-Algorithmen mit Hilfe eines Computerprogrammes durchgeführt. Dabei

[33] Das in der Quelle [EK70] primär behandelte Problem ist das des minimalen Kostenflusses (Minimum-Cost-Flow-Problem).

konnten für diese konkreten Modellinstanzen und für konkrete Online-Algorithmen Werte für den Competitive Ratio per Computer nachgewiesen werden. Die Konzepte dieses Programmes und die Zusammenhänge mit dem oben beschriebenen Beweis werden am Beispiel der ORSM-Variante im nächsten Teilkapitel behandelt. Die Auswertung dieser Experimente schließt sich danach an.

8.2.2 Implementierungsdetails

Die Konfigurationsmenge

Eine Konfiguration eines Online-Algorithmus wird für die hier betrachtete ORSM-Variante durch einen Teilgraphen der Eingabe verkörpert. Der Aufbau einer Konfiguration sowie der Menge aller möglichen Konfigurationen leitet sich aus den Modellparametern und der Menge verschiedener Eingaben eines Zeitschrittes wie folgt ab:
Für einen Requestknoten r_i werden g Nachbarknoten aus der Serverknotenmenge $\{s_i, s_{i+1}, \ldots, s_{i+d}\}$ spezifiziert oder r_i bleibt isoliert. Deshalb gibt es $\binom{d+1}{g}$ verschiedene Möglichkeiten für die Struktur der Nachbarschaft von r_i (beachte $|\{s_i, s_{i+1}, \ldots, s_{i+d}\}| = d+1$) plus eine eindeutige Möglichkeit einer leeren Nachbarschaft. Somit ist die Menge und Anzahl aller möglichen Eingaben eines Zeitpunktes bestimmt.
Einem Online-Algorithmus steht zu einem Zeitpunkt i, in dem über die Nutzung von s_i entschieden wird, ein Teilgraph der Gesamteingabe und die aktuelle Eingabe von r_i zur Verfügung. Durch die Beschränkung der Kantenlänge in der betrachteten Modellvariante ist auch die Größe des Teilgraphen beschränkt. Der Knoten r_{i-d} kann noch eine Kante maximaler Länge zum Knoten s_i besitzen. Jeder Requestknoten r_j mit $j < i-d$ ist zum Zeitpunkt i entweder im Online-Matching gebunden oder irreversibel frei, er ist aber nicht mehr Teil der aktuellen Entscheidungssituation. Der Knoten r_{i-1} ist der letzte Requestknoten, der sich neben dem aktuellen Eingabeknoten r_i im Teilgraphen befindet. Alle weiteren Requestknoten r_j mit $j > i$ sind zum Zeitpunkt i unbekannt. Ebenso sind maximal $d+1$ Serverknoten für den Online-Algorithmus von Interesse. Der erste Serverknoten ist der aktuell zu bearbeitende Knoten s_i. Die Kantenlängenbegrenzung impliziert dann, daß vom letzten Requestknoten r_{i-1} höchstens der Serverknoten s_{i-1+d} bzw. vom Eingabeknoten r_i der Serverknoten s_{i+d} erreicht werden kann. Alle späteren Serverknoten s_j mit $j > i+d$ sind zum Zeitpunkt i isoliert.
Aus diesen Überlegungen ergibt sich, daß die Problemkonfiguration für einen Online-Algorithmus vor der Eingabe von r_i aus einem bipartiten Teilgraphen besteht, der auf die Knotenmenge $\{r_{i-d}, r_{i-d+1}, \ldots, r_{i-1}, s_i, s_{i+1}, \ldots, s_{i+d}\}$ beschränkt ist. Dabei handelt es sich nicht um den von dieser Knotenmenge induzierten Teilgraphen der Gesamteingabe, denn innerhalb des Zeitintervalls $[i-d, i-1]$ können einige der Requestknoten in das Online-Matching aufgenommen worden sein und stehen deshalb nicht mehr zur Verfügung.
Alle soeben beschriebenen Teilgraphen bilden die Menge von Problemkonfigu-

rationen. Um sie auf- und abzuzählen sind zwei Fakten zu berücksichtigen: Jeder Requestknoten ist eine Eingabe mit einer der $\binom{d+1}{g} + 1$ oben beschriebenen Kantenstrukturen. Weiterhin kann ein Requestknoten r_j mit $i - d \leqq j < i$ durch seine Kanten Serverknoten bis höchstens s_{j+d} erreichen und in der Menge $\{s_i, s_{i+1}, \ldots, s_{j+d}\}$ befinden sich für einen nichtisolierten Requestknoten r_j mindestens $\max\{1, g - (i - j)\}$ und höchstens $\min\{g, j + d - i + 1\}$ Nachbarknoten. Aus der Kombination aller dargelegten Möglichkeiten für jeden Requestknoten der Menge $\{r_{i-d}, \ldots, r_{i-1}\}$ ergibt sich die Anzahl aller Konfigurationen (möglicher Teilgraphen):

$$\prod_{p=1}^{d} \left[1 + \sum_{q=\max\{1,g+p-d-1\}}^{\min\{g,p\}} \binom{p}{q} \right] \quad .$$

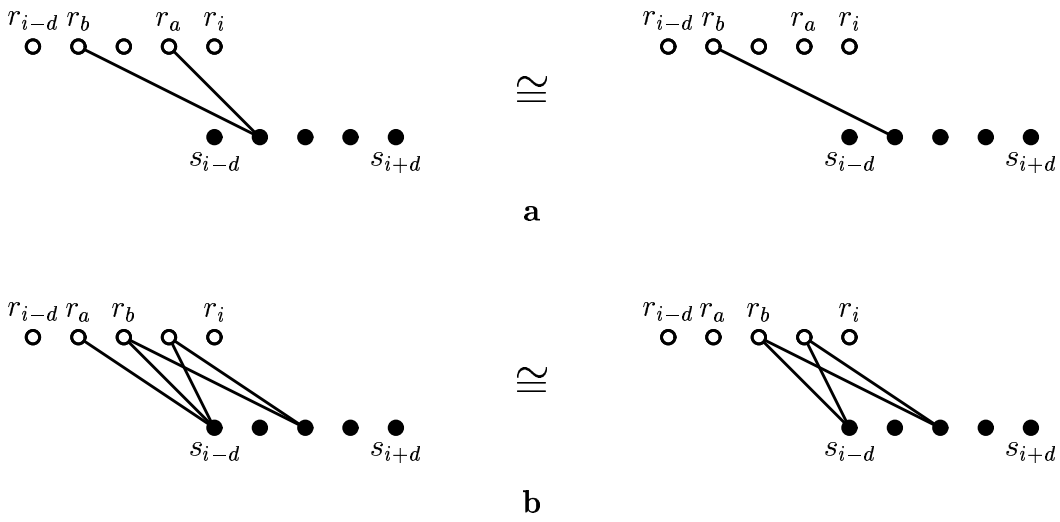
Da eine Implementierung der im Kapitel 8.2.1 beschriebenen Beweisidee auf der Konfigurationsmenge aufbaut, ist es sinnvoll und effizienzsteigernd diese zu verkleinern, falls möglich. Wenn bereits a priori erkennbar ist, daß sich verschiedene Konfigurationen in ihrer Struktur und Wirkung durch einen Algorithmus nicht unterscheiden lassen, so sollten sie zusammengefaßt werden. Derartige strukturelle Gleichheit läßt sich durch eine Äquivalenzrelation ausdrücken und für weitere Programmschritte wird nur noch die Menge von Äquivalenzklassenvertretern berücksichtigt.
Im Beispiel der Konfigurationsmenge des ORSM-Problems mit konstantem Knotengrad g und begrenzter Kantenlänge d kann die im folgenden beschriebene Äquivalenzrelation benutzt werden. Wegen der Ordnung auf den Serverknoten sind zwei Konfigurationen mit isomorphen Graphen nicht automatisch strukturäquivalent. Da aber zum Zeitpunkt i die Historie des Aufbaus des aktuellen Teilgraphen für seine strukturellen Eigenschaften und die Entscheidungsmöglichkeiten keine Rolle spielt, so sind zwei Konfigurationen äquivalent, wenn sich die Graphen durch Permutation der d Requestknoten ineinander überführen lassen.
Zweitens können Graphen mit Überlastsituationen, d. h. es sind mehr Requestknoten zu einer Serverknotenmenge adjazent, als deren Kardinalität, unter bestimmten Nebenbedingungen zu Graphen mit ausgeglichener Anfragelast äquivalent sein. Präziser ausgedrückt bedeutet das: Falls ein Graph einer Konfiguration mehr nichtisolierte Requestknoten besitzt, als die Kardinalität des Maximum-Matchings dieses Graphen groß ist, so folgt aus dem Satz von Hall über die Existenzbedingung für perfekte Matchings [Hal35], daß eine *minimale* Requestknotenmenge R' existiert, deren Nachbarschaft $\Gamma(R')$ kleiner als sie selbst ist, also

$$|R'| \quad \geqq \quad |\Gamma(R')| \quad .$$

Existiert ein Knoten $r_a \in R'$ mit minimalem Knotengrad und ein Knoten $r_b \in R'$ mit der Eigenschaft

$$\Gamma(r_a) \quad \subseteqq \quad \Gamma(r_b) \quad ,$$

d. h. die Nachbarschaft[34] von r_b beinhaltet die Nachbarschaft von r_a vollständig, so kann der Knoten r_a aus der Betrachtung für die Entscheidungen entfallen. Mindestens ein Knoten aus R' kann nicht in das Online-Matching aufgenommen werden und r_a darf dazu a priori bestimmt werden. Das letzte Argument folgt aus der höheren Flexibilität des Einsatzes von r_b und der damit verbundenen größeren kombinatorischen Möglichkeiten für die Online-Lösung. Somit ist der Graph einer Konfiguration, der die Vorbedingungen erfüllt und die Knoten r_a und r_b besitzt für das betrachtete Online-Problem strukturäquivalent zu einem Graphen ohne r_a bzw. mit einem isolierten Knoten r_a. Siehe zu dieser Idee die beiden Beispiele in Skizze 31.



Skizze 31: Zwei Beispiele für Graphen äquivalenter Konfigurationen.

Die mit diesen beiden Eigenschaften definierte Äquivalenzrelation bildet durch die Transitivität relativ große Klassen von Konfigurationen und verkleinert damit die Kardinalität der zu betrachtenden Konfigurationsmenge beträchtlich. Trotzdem ist die angegebene und im Programm implementierte Äquivalenzrelation konservativ definiert. Eine stärkere Relation, die nur noch Strukturen von perfekten bzw. optimalen Matchings auf den zu vergleichenden Graphen berücksichtigt, ist jedoch nicht ohne großen Aufwand mathematisch zu formulieren und zu begründen. Da sich durch derartige Verbesserungen bei den Computerexperimenten keine Laufzeiteinsparungen um Größenordnungen erzielen ließen, ist vorsichtshalber auf deren Einsatz verzichtet worden.

Innerhalb der Implementierung ist ein Zugriff auf die Nummer der Äquivalenzklasse mittels Graphen als Schlüssel notwendig. Dazu wurde folgendermaßen vorgegangen: Alle Graphen, die eine gültige Konfiguration darstellen, werden erzeugt und in einer Liste gespeichert. Danach wird eine Hash-Struktur über diese Liste

[34] das sind alle zu r_b adjazenten *Serverknoten*

aufgebaut, innerhalb der mit dem Graphen bzw. den Bitfolgen seiner Adjazenzmatrix als Schlüssel schnell gesucht werden kann. Die Äquivalenzrelation wird für jedes Paar von Graphen geprüft und gegebenenfalls wird von einem strukturäquivalenten Graphen ein Zeiger zu seinem Äquivalenzklassenrepräsentanten aufgebaut. Die Transitivitätseigenschaft läßt sich bei diesen Tests ausnutzen, so daß nicht wirklich alle Paare von Konfigurationen auf Äquivalenz getestet werden müssen.

Die Darstellung der Algorithmen

Nachdem für das Online-Problem eine endliche Konfigurationsmenge bestimmt wurde, können Algorithmen durch ihr Verhalten auf diesen Konfigurationen beschrieben werden, anstatt durch eine abstrakte Handlungsvorschrift, die Rechenoperationen auf den Graphen der Konfigurationen verlangt. Sollte ein Online-Algorithmus, neben den aktuell für die Bearbeitung zur Verfügung stehenden Teilgraphen — das entspricht einer Problemkonfiguration — weitere eigene Variablen (z. B. Zähler) zur Entscheidung heranziehen, so muß die Konfigurationsmenge um diese Werte erweitert werden. Alle in dieser Arbeit betrachteten Online-Algorithmen arbeiten jedoch in diesem Sinne gedächtnislos, so daß eine Vergrößerung der Konfigurationsmenge nicht notwendig ist.
Ein konkreter deterministischer Online-Algorithmus ALG läßt sich durch die eindeutigen Übergänge zwischen den Konfigurationen unter den verschiedenen Eingaben spezifizieren. Das heißt für jede Konfiguration und für jede Eingabe wird die Entscheidung von ALG festgestellt und damit die Nachfolgekonfiguration bestimmt, welche die Vorlage des nächsten Zeitschrittes darstellt. Dies kann durch eine gerichtete Verbindung zwischen den beiden beteiligten Konfigurationen geschehen. An diese Verbindung ist gleichzeitig die Eingabe als ein Attribut gekoppelt. Somit wird ein Automat als Beschreibung von ALG konstruiert. Weiterhin kann einer solchen Verbindung die erzielte Leistung zugeordnet werden. Im Beispiel des ORSM-Modells ist dies eine 1, wenn eine Kante mit dem aktuellen Serverknoten s_i für das Online-Matching M_{ALG} bestimmt wurde. Anderenfalls ist es eine 0. Mit dieser zusätzlichen Angabe stellt sich heraus, daß der deterministische Online-Algorithmus ALG als ein deterministischer Mealy-Automat[35] über der Konfigurationsmenge, angetrieben durch die Eingaben des aktuellen Zeitschrittes und mit der Ausgabe des Leistungswertes — gebunden an die Konfigurationsübergänge — dargestellt ist.
Der Online-Algorithmus sowie sein beschreibender Mealy-Automat beginnen mit der Startkonfiguration Z_0 des kantenlosen Graphen. Danach beschreibt jede Eingabesequenz σ innerhalb des Mealy-Automaten einen Weg der Konfigurationsübergänge und die dabei in jedem Schritt erzielte Leistung. Somit wird auch für ALG die Abfolge der Konfigurationen und die erreichte Leistung angegeben.

Auch das Verhalten des Gegenspielers bzw. des optimalen Algorithmus kann durch Übergänge zwischen den Konfigurationen beschrieben werden. Es ist jedoch

[35] Zur Definition siehe z. B. [HU90] oder ein anderes Standardlehrbuch.

unbekannt, welche Entscheidungen bei einer konkreten Konfiguration und Eingabe getroffen werden. Die Modellierung mittels Mealy-Automat ist zwar erneut möglich, allerdings ist dieser *nichtdeterministisch*. Ausgehend von einer Konfiguration und einer Eingabe werden alle möglichen Entscheidungen und die sich daraus ergebenden Nachfolgekonfigurationen berechnet und für jeden Übergang die entsprechende Leistung. Sämtliche Online-Algorithmen, einschließlich aller optimalen Algorithmen, sind damit modelliert.[36]

Zur Effizienzsteigerung des Programmes ist es wünschenswert, wenn die beiden Mealy-Automaten möglichst klein sind, d. h. eine kleine Anzahl von Zuständen besitzen. Für den deterministischen Mealy-Automaten ist eine Minimierung der Zustandsanzahl mit einem wohlbekannten Standardalgorithmus problemlos und effizient durchführbar. Dafür genügt eine quadratische Laufzeit in der Kardinalität der eingegebenen Zustandsmenge.

Für einen nichtdeterministischen Mealy-Automaten ist die Minimierung der Zustände ein $\mathcal{NP}$-schweres Problem. Da dessen Lösung die Laufzeit des gesamten Programmes dominieren würde, ist auf diesen Schritt verzichtet worden. Allerdings kann auch eine einfache und schnelle Heuristik helfen einige Zustände des Mealy-Automaten einzusparen. Dazu werden zwei Zustände des Mealy-Automaten miteinander verschmolzen, wenn unter den Eingaben jeweils die selben Nachfolgekonfigurationsmengen bei gleichen Leistungswerten erreicht werden. Dieser Prozeß kann iterativ so lange wiederholt werden, bis die Bedingung für kein Paar von Konfigurationen mehr zutrifft.

Beim Aufbau des nichtdeterministischen Mealy-Automaten können unter Umständen einige Übergänge eingespart werden, falls ersichtlich ist, daß mögliche Entscheidungen durch andere Entscheidungen dominiert werden. Im konkreten Online-Problem kann darauf verzichtet werden, für nichtisolierte Serverknoten s_i zum Zeitpunkt i die Entscheidung $s_i \mathrel{\mathring{\notin}} M$ zu treffen. Damit wird erzwungen, daß im aktuellen Zustand der Serverknoten für das Matching benutzt wird, wenn er einen adjazenten Requestknoten besitzt. Dies führt zwar zum Ausschluß einer Menge von optimalen Lösungen, aber eine einfache Überlegung zeigt, daß auch unter dieser Einschränkung in der Lösungsstruktur stets optimale Lösungen existieren. Diese erreichen die maximale Kardinalität des Online-Matchings mit der Nutzung möglichst früher Servicezeitpunkte.

Mit diesem Trick ist nicht nur die Struktur des nichtdeterministischen Mealy-Automaten zusätzlich verkleinert worden, sondern es wird auch sichergestellt, daß unter der selben Eingabe alle Entscheidungen aus einer Konfiguration den selben Leistungswert erzielen. Dieser Fakt vereinfacht die Implementierung der Heuristik zur Verkleinerung der Zustandsmenge und die Laufzeit dieses Programmabschnittes.

[36] In der Implementierung wird zuerst dieser nichtdeterministische Mealy-Automat aufgebaut, und danach wird der deterministische Mealy-Automat des zu untersuchenden Online-Algorithmus durch die Auswahl von Kanten abgeleitet.

Der Aufbau des Graphen

Mit den soeben getroffenen Vorarbeiten kann nun der Graph der Konfigurationspaare und ihrer Übergänge aufgebaut werden. Zur Bestimmung der Übergänge zwischen Zustandspaaren wäre es spätestens an dieser Stelle notwendig die Konfigurationsübergänge des Online-Algorithmus und die möglichen Konfigurationsübergänge in einem optimalen Algorithmus zu berechnen. Nach der Darstellung der Algorithmen als Mealy-Automaten auf der Konfigurationsmenge kann nun darauf zurückgegriffen werden. Zudem wird nur die erste starke Zusammenhangskomponente des Graphen, beginnend mit dem Startkonfigurationspaar (Z_0, Z_0) der leeren Teilgraphen in der ORSM-Modellvariante, aufgebaut. Die Kanten erhalten vorerst beide Leistungswerte, den des Online- und den des optimalen Algorithmus, als Attribute. Die Interpretation der Algorithmen als Mealy-Automaten läßt erkennen, daß der benötigte Graph der Konfigurationspaare, beschränkt auf die erste Erreichbarkeitskomponente, das Kreuzprodukt des deterministischen Mealy-Automaten des Online-Algorithmus und des nichtdeterministischen Mealy-Automaten des optimalen Algorithmus ist. Deshalb führt die Minimierung bzw. Verkleinerung der Mealy-Automaten zu einer Verkleinerung des Konfigurationsübergangsgraphen und damit zu einer Effizienzsteigerung der später darauf operierenden Algorithmen, ohne die Modellierung des Online-Problems oder der Algorithmen zu verändern.
Nach dem Aufbau dieses Konfigurationsübergangsgraphen können die Datenstrukturen gelöscht werden, welche die konkreten Eingaben und Teilgraphen der Konfigurationen beinhalten. Ebenso wird die Beschreibung der beiden Mealy-Automaten nicht mehr für den weiteren Programmablauf benötigt. Sie stellt jedoch eine kompakte Repräsentation der Problembeschreibung dar und wird in der Implementierung als Datei gespeichert, um an dieser Stelle spätere Programmläufe aufzusetzen.

Da der zu überprüfende Wert c für den Competitive Ratio Teil der Eingabe eines Programmlaufes ist, kann für jede Kante e des Graphen aus den beiden Leistungswerten $Perf_{\mathsf{ALG}}(e)$ und $Perf_{\mathsf{OPT}}(e)$ sowie dem Wert c das Gewicht gemäß der Formel $w(e) = \mathsf{c} \cdot Perf_{\mathsf{ALG}}(e) - Perf_{\mathsf{OPT}}(e)$ berechnet werden. Nach dieser Operation ist der Graph vollständig aufgebaut und der eigentliche Test kann beginnen.
Wie schon auf Seite 141 ausgeführt, ist der exakte Wert für c ein gemeiner Bruch mit

$$\mathsf{c} \;=\; \frac{a}{b} \qquad , \quad a, b \in \{1, 2, \ldots, n\} \;\; .$$

Um numerische Instabilitäten zu vermeiden, werden die Werte für c und alle Kantengewichte konzeptionell als gemeine Brüche dargestellt. Für die Implementierung stellt ich heraus, daß *alle* Berechnungen bezüglich des selben Nenners b ablaufen können, so daß nur die Zähler gespeichert werden. Bei der Berechnung der Kantengewichte wird entsprechend mit b erweitert und alle Gewichte sind als Ganzzahlvariablen implementiert. Somit werden sämtliche Berechnungen ohne Rundungsfehler ausgeführt.

Der Test auf negative Kreise

In dem gewichteten Konfigurationsübergangsgraphen, der alle möglichen Eingabesequenzen und Reaktionen der beiden Algorithmen beschreibt, ist nun ein Kreis mit minimaler Gewichtssumme zu bestimmen. Falls diese Gewichtssumme exakt Null ist, so ist c der gesuchte Wert für den Competitive Ratio. Die mit dem Kreis implizit verbundene Eingabesequenz ist der Zeuge für eine untere Schranke. Die Tatsache, daß dieser Kreis die *minimale* Gewichtssumme besitzt, zeigt eine Grenze für den Gegenspieler auf. Deshalb ist der Competitive Ratio des Online-Algorithmus nicht schlechter als c.
Der Fall eines negativen Kreises impliziert eine untere Schranke von c für den exakten Competitive Ratio. Falls die Gewichtssumme des minimalen Kreises strikt positiv ist, so stellt c eine obere Schranke für den Competitive Ratio des Online-Algorithmus dar.
Wie schon in Kapitel 8.2.1 dargestellt, kann zur Bestimmung des Kreises mit minimaler Gewichtssumme der Floyd-Warshall-Algorithmus herangezogen werden. Bei den derzeit verbreiteten Computersystemen mit 32-Bit-Betriebssystemen, und den Computern, welche für die Untersuchungen zur Verfügung standen, können für einen Rechenprozeß maximal 2 GByte Hauptspeicher angefordert werden. Der Floyd-Warshall-Algorithmus benötigt den Graphen in der Darstellung einer vollständigen Adjazenzmatrix, so daß aus der Speicherbegrenzung eine Grenze für die Graphgröße von wenigen Zehntausend Knoten folgt. Zudem ist eine Laufzeit von $\Theta(n^3)$ bei heutiger Rechenleistung unproduktiv. Deshalb ist über den Einsatz effizienterer Algorithmen nachzudenken, die zudem auf einer Adjazenzlistendarstellung des Graphen operieren können ($O(m)$ Speicherbedarf für einen zusammenhängenden Graphen, anstatt $O(n^2)$).

Ist es für die Aufgabenstellung den exakten Competitive Ratio durch Tests zu bestimmen wirklich notwendig $W_{\min}$ zu berechnen, danach mit Null zu vergleichen und einen der drei Fälle $W_{\min} < 0$, $W_{\min} = 0$ oder $W_{\min} > 0$ festzustellen? Für die reine Detektion negativer Kreise sind effizientere Verfahren bekannt. Damit kann das Testergebnis nur noch zwischen $W_{\min} < 0$ und $W_{\min} \geqq 0$ unterscheiden, woraus sich die Aussage „c ist eine *echte* untere Schranke“ oder „c ist eine obere Schranke“ ergibt. Dieser Verlust in der Präzision des Testergebnisses läßt sich jedoch ausgleichen. Nachdem für einen Wert c der Test $W_{\min} \geqq 0$ ergeben hat, wird der Wert $\mathsf{c} - \varepsilon$ getestet. Sollte der zweite Test bei einem

$$\varepsilon = \frac{1}{n^2}$$

zu dem Ergebnis $W_{\min} < 0$ gelangen, so ist sichergestellt, daß der Wert c exakt bestimmt wurde. Diese Tatsache folgt aus der Überlegung, daß der exakte Wert von c ein Bruch a/b mit $a, b \in \{1, 2, \ldots, n\}$ und das oben definierte ε kleiner als die kleinste Differenz zweier möglicher Werte für c ist.
Sollten in dem zu untersuchenden Online-Problem Leistungswerte größer als 1 auftreten, diese aber ganzzahlig und durch $Perf_{\max}$ beschränkt sein, so ist c von

der Form a/b mit $a, b \in \{1, 2, \ldots, n \cdot Perf_{\max}\}$. Für diesen allgemeineren Fall wählt man:

$$\varepsilon \;=\; \frac{1}{(n \cdot Perf_{\max})^2} \;\cdot$$

Ein effizientes Verfahren zum Testen, ob ein gewichteter und gerichteter Graph $G = (V, E, w)$ einen negativen Kreis aufweist, ist der Bellman-Ford-Algorithmus. Er berechnet den Baum der kürzesten Wege von einem Startknoten $v_0 \in V$ zu allen anderen Knoten des Graphen. Falls a priori bekannt ist, daß alle Kantengewichte nichtnegativ sind, so ist zwar der Einsatz des Dijkstra-Algorithmus effizienter, aber diese Voraussetzung gilt für die Konfigurationsübergangsgraphen niemals. Bei Existenz von negativen Gewichten an den Kanten muß jedoch getestet werden, ob es Kreise mit negativem Gesamtgewicht gibt, weil unter dieser Bedingung kürzeste Wege im Graphen nicht mehr definiert sind. Der Bellman-Ford-Algorithmus leistet diese Aufgabe, indem er den Baum kürzester Wege berechnet oder mit der Feststellung eines negativen Kreises terminiert. Dazu wird jedem Knoten $v \in V$ ein Abstandswert $dist(v)$ zugeordnet, der mit $+\infty$ initialisiert ist. Lediglich der Startknoten besitzt den Abstand $dist(v_0) = 0$. Danach wird für jede Kante (u, v) der sogenannte Relaxationsschritt durchgeführt:

1: **if** $dist(v) > dist(u) + w((u, v))$ **then**
2: $\quad dist(v) := dist(u) + w((u, v))$
3: **end if**

Algorithmus 6: Die zentrale Funktion `relax(`u, v`)` des Bellman-Ford-Algorithmus.

Diese Prozedur, jede Kante zu relaxieren, wird für den Graphen $G = (V, E, w)$ mit $n = |V|$, $m = |E|$ n-mal iteriert. Zum Schluß wird überprüft, ob es noch eine Kante (u, v) gibt, für die $dist(v) > dist(u) + w((u, v))$ gilt. Wird eine Kante mit dieser Eigenschaft gefunden, so besitzt G einen negativen Kreis, anderenfalls stehen in den Variablen $dist(v)$ die kürzesten Abstände zum Startknoten v_0. In obiger Darstellung ist der Bellman-Ford-Algorithmus, einschließlich seines Korrektheitsbeweises, in jedem Lehrbuch für Algorithmen zu finden (z. B. [CLR90]). Er besitzt in dieser einfachen Implementierung eine Laufzeit von $\Theta(n \cdot m)$, arbeitet aber auf der speicherplatzsparenden Graphdarstellung einer Adjazenzliste. Die folgenden Überlegungen können, je nach Struktur der Eingabegraphen, zu großen Laufzeiteinsparungen führen.

Jeder Knoten $u \in V$ erhält eine Markierung, die auf „aktiv“ gesetzt wird, sobald sich sein Abstandswert $dist(u)$ verändert. In einer Iteration wird auf die Ausgangskanten des Knotens u nur dann die `relax(`u, v`)`-Funktion ausgeführt, wenn die Markierung von u auf „aktiv“ steht, d. h. falls sich $dist(u)$ vorher verringert hat. Anderenfalls müssen alle Tests $dist(v) > dist(u) + w((u, v))$ fehlschlagen und sollten nicht durchgeführt werden. Nach Ausführung der `relax(`u, v`)`-Funktion für alle Ausgangskanten des Knotens u wird seine Markierung auf „inaktiv“ zurückgesetzt.

Gleichzeitig wird in jedem Iterationsdurchlauf des Bellman-Ford-Algorithmus festgestellt, ob es überhaupt eine Kante (u, v) gab, über die eine Verringerung des Abstandes $dist(v)$ bewirkt wurde. Ist dies nicht der Fall, so sind für alle Knoten v die Werte $dist(v)$ minimal, d. h. der Baum kürzester Wege wurde erfolgreich bestimmt. Der Algorithmus kann an dieser Stelle vorzeitig terminieren, denn jede weitere Iteration hat keinerlei Veränderung mehr zur Folge.

Auch negative Kreise, vor allem wenn sie erheblich weniger als n Knoten besitzen, können vorzeitig erkannt werden. Dazu wird jedem Knoten des Graphen eine weitere Variable zugeordnet, in der sein *Vorgänger* gespeichert wird. Der Vorgänger eines Knotens v ist derjenige Knoten u, durch den per Aufruf von `relax(`u, v`)` zum letzten Mal $dist(v)$ verringert wurde. Jeder Knoten v mit $dist(v) < +\infty$ besitzt zu jedem Zeitpunkt der Ausführung des Algorithmus einen eindeutigen Vorgänger. Die Kette der Vorgänger eines Knotens endet entweder im Startknoten v_0 oder erreicht den Knoten v selbst. Im letzten Fall ist ein Kreis gefunden, der ein negatives Gesamtgewicht aufweist. Dieser Test benötigt bei effizienter Implementierung nicht mehr als $O(n)$ Laufzeit und kann zwischen zwei Iterationsschritten[37] des Bellman-Ford-Algorithmus durchgeführt werden.
Falls im untersuchten Graphen G kurze, negative Kreise existieren, so werden sie mit diesem Sondertest nach wesentlich weniger als n Iterationen gefunden und das Verfahren wird abgebrochen.

Der Bellman-Ford-Algorithmus kann durch die Verbesserung nach Yen weiter beschleunigt werden (siehe dazu die Aufgabe 25-1 in [CLR90, S. 545 f.]). Dafür wird die Kantenmenge E in zwei disjunkte Teilmengen E_f und E_b zerlegt, wozu eine beliebige lineare Ordnung $(v_0, v_1, \ldots, v_{n-1})$ auf der Knotenmenge V benötigt wird.[38] Damit wird definiert:

$$\begin{aligned} E_f &:= \{(v_i, v_j) \in E \mid i < j\} \\ E_b &:= E \setminus E_f \ . \end{aligned}$$

Der Teilgraph $G_f = (V, E_f, w)$ stellt einen azyklischen Graphen dar und die Knotenordnung $(v_0, v_1, \ldots, v_{n-1})$ eine topologische Sortierung dafür. Der Teilgraph $G_b = (V, E_b, w)$ ist ebenfalls azyklisch und die umgekehrte Ordnung auf den Knoten $(v_{n-1}, v_{n-2}, \ldots, v_1, v_0)$ ist für G_b eine topologische Sortierung. Innerhalb eines Iterationsschrittes des Bellman-Ford-Algorithmus werden zunächst die Kanten aus E_f in der Reihenfolge der topologischen Sortierung ihrer Quellknoten relaxiert und danach die Kanten aus E_b bezüglich der topologischen Sortierung von G_b. Durch diese Maßnahme wird eine Abstandsverkürzung pro Iterationsschritt nicht nur bis zu den Nachbarknoten, sondern mindestens auch bis zu deren Nachbarn propagiert. Deshalb muß der Bellman-Ford-Algorithmus nur noch $\lceil n/2 \rceil$ Iterationen durchführen. Diese Verbesserung bedarf nur eines sehr geringen Mehraufwandes bei der Implementierung und sollte darum immer benutzt werden.

[37] Sind in G große Kreise mit negativem Gewicht zu erwarten, so kann dieser Test auch nur alle 10 oder 100 Iterationsschritte durchgeführt werden.
[38] Durch die Anordnung der Knoten im Speicher ist implizit eine derartige Ordnung gegeben.

Keine der vorgestellten oder bisher bekannten Verbesserungen kann die Laufzeitschranke des Bellman-Ford-Algorithmus im O-Kalkül verringern. Die Verbesserung nach Yen spart jedoch in der realen Laufzeit einen Faktor von ca. 2.[39] Die Berücksichtigung der Aktivität der Knoten führt ebenfalls zu meßbaren Laufzeitverringerungen. Die Techniken zum vorzeitigen Abbruch des Verfahrens adaptieren die Gesamtlaufzeit auf die vorhandenen Struktureigenschaften des Eingabegraphen.

Bei den Graphen, die in den Untersuchungen der ORSM-Modelle auftraten, haben die bisher beschriebenen Beschleunigungstechniken zu extrem kurzen Laufzeiten geführt. Selbst für Graphen, die über 1.6 GByte im Hauptspeicher belegten, was ca. 2.6 Millionen Konfigurationsknoten mit durchschnittlich knapp 80 Ausgangskanten entsprach, sind nur Laufzeiten von wenigen Minuten bei dieser Implementierung des Bellman-Ford-Algorithmus aufgetreten. Diese Laufzeit nimmt jedoch nur einen geringen Anteil der Gesamtlaufzeit des Programmes in Anspruch. Deshalb ist auf den Einsatz neuerer Überlegungen zur Berechnung des Baumes kürzester Wege nach [GR93] mit recht komplizierten Implementierungsdetails bzw. von vereinfachten Lösungsansätzen gleicher Grundidee[40] verzichtet worden.

Aus dem selben Grund wurde die Idee, die Existenz eines negativen Kreises durch das Lösen des Minimum-Mean-Cycle-Problems festzustellen, nicht weiter verfolgt. Für dieses Problem, den Kreis mit minimalem Durchschnittsgewicht der Kanten zu bestimmen, wurde in [Kar78] ein Verfahren mit gleichem asymptotischem Aufwand wie beim Bellman-Ford-Algorithmus von $O(n \cdot m)$ angegeben. Bessere Resultate sind bisher für den allgemeinsten Fall der Eingabe nicht bekannt. Es gibt jedoch neuere Entwicklungen [OA92], die für ganzzahlige Kantengewichte, welche durch $w_{\max}$ beschränkt sind, eine Lösung in $O(\sqrt{n} \cdot m \cdot \log(n \cdot w_{\max}))$ berechnen. Sie sind für $w_{\max} \in O(n^k)$ bei konstantem k asymptotisch schneller und basieren auf der Skalierungstechnik. Für die zu untersuchenden Konfigurationsübergangsgraphen gilt sogar, daß $w_{\max}$ konstant ist.

Die Bestimmung der Potentialfunktion

Wie in der Vorstellung des Beweises für die Existenz von Potentialfunktionen für alle Online-Probleme und -Algorithmen zur Analyse nach der Potentialfunktionsmethode auf Seite 142 ausgesagt, wurde in [EK70] auch ein Verfahren angegeben, um derartige Potentialfunktionen zu bestimmen. Dazu muß nur einen Knoten v_0 fixiert werden, und für alle anderen Knoten des Graphen ist die Länge der kürzesten Wege von v_0 zu diesen Knoten zu bestimmen. Werden diese Abstandswerte der Knoten als deren Potential aufgefaßt, so erfüllt die so definierte und diskret dargestellte Potentialfunktion die gewünschten Eigenschaften.

[39] Je nach Ablage der Daten im Speicher und der Verwaltung der Speicherhierarchie des jeweiligen Computers kann es zu leichten Verschlechterungen der Cache-Effizienz kommen.

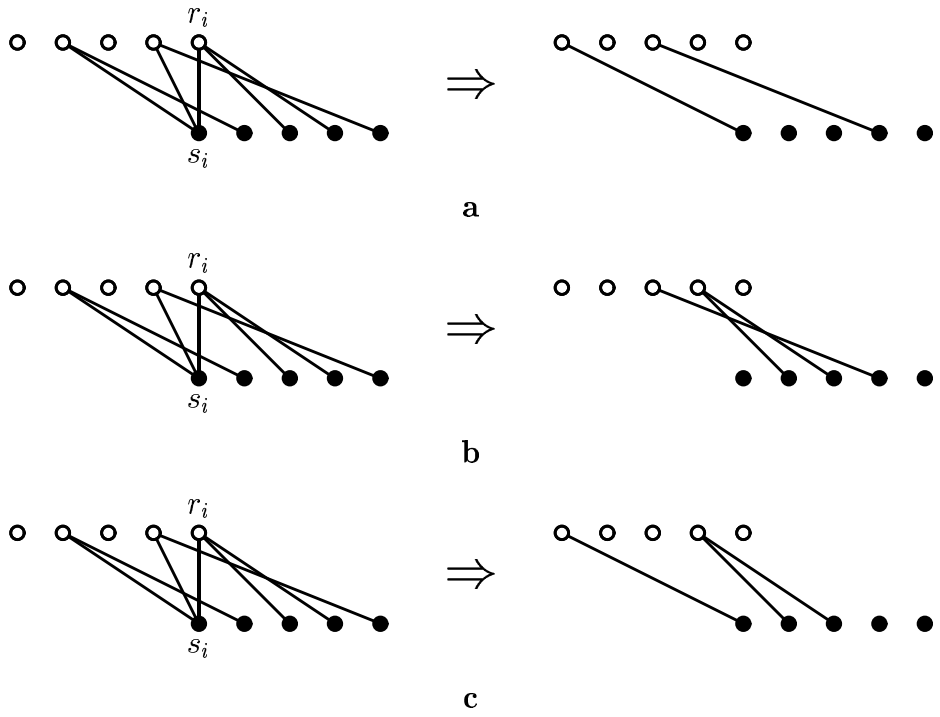
[40] Diese Entwicklungen wurden in persönlichen Gesprächen mit Meinolf Sellmann in der Mitte des Jahres 1999 diskutiert.

Durch Anwendung des Bellman-Ford-Algorithmus auf dem Konfigurationsübergangsgraphen wurde genau diese Berechnung durchgeführt. Mit dem Feststellen der Aussage $W_{\min} \geqq 0$ wurde gleichzeitig eine Potentialfunktion $\Phi : V \to \mathbb{R}$ mit $\Phi(v) = dist(v), \quad \forall v \in V$ bestimmt und die Gleichung (4) ist für jede Kante (u, v), d. h. für jede Situation des Online-Algorithmus im Spiel gegen einen optimalen Algorithmus, durch die `relax(`u, v`)`-Funktion überprüft worden. Es läßt sich erkennen, daß die vorgestellte Methode einen Computerbeweis für die Aussage „ALG ist c-competitive“ durchführt, der auf den Problemkonfigurationen und einer diskreten Darstellung der Potentialfunktion arbeitet.

8.3 Algorithmen und Ergebnisse

Da der Online-Algorithmus LMM für die Lösung des ORSM-Problems und der im Kapitel 7 untersuchten Modellvarianten optimal war, lag es nahe die Computeranalysen für die Variante des ORSM-Problems mit konstantem Knotengrad g und begrenzter Kantenlänge d ebenfalls mit dem Algorithmus LMM zu beginnen. Bei seiner Implementierung stellt sich jedoch als erstes heraus, daß die Regeln, welche LMM spezifizieren, nicht ausreichend sind, um in jeder Situation *eindeutige* Entscheidungen und damit einhergehende Nachfolgekonfigurationen zu bestimmen. Verschiedene getestete Implementierungen führen auch zu verschiedenen Werten für den Competitive Ratio bei einigen untersuchten Modellinstanzen. Weiterhin konnten aus dem Studium der Gegenspielersequenzen wertvolle Hinweise zur Verbesserungen des Online-Algorithmus gewonnen werden.
Der Algorithmus LMM stellt an seine Entscheidung, neben der bevorzugten Nutzung von s_i, nur eine Forderung. Der Graph der Nachfolgekonfiguration muß ein Matching besitzen, welches unter allen möglichen Nachfolgekonfigurationen eine maximale Kardinalität aufweist. Wie in Skizze 32 an einem Beispiel zu sehen, erfüllen alle drei möglichen Entscheidungen für die Nutzung von s_i mit den sich ergebenden Folgegraphen diese Bedingung.
Daraus ergibt sich die Einsicht, daß der Online-Algorithmus LMM in Wahrheit eine ganze Klasse von Algorithmen darstellt. Mit dem Computerexperiment wird nur ein konkreter Vertreter dieser Algorithmenklasse analysiert. Er wird durch die aktuellen Datenstrukturen im Speicher definiert. Die Aussage des Experimentes ist damit sehr eingeschränkt. Es wird lediglich festgestellt, daß es innerhalb der beschriebenen Klasse von Online-Algorithmen einen Vertreter mit exakt dem nachgewiesenen Competitive Ratio gibt. Dieser Zahlenwert sagt jedoch weder etwas über obere noch über untere Schranken für den Competitive Ratio der gesamten Klasse aus.

Wie in Skizze 32 zu sehen, ist die Wahl der Entscheidungsvariante in Teil a ungünstig. Besser ist die Wahl eines Nachfolgegraphen mit größerer *Expansionseigenschaft*. In diesem Fall ist es die Auswahl eines Nachfolgegraphen mit möglichst vielen, nichtisolierten Serverknoten. Die Anzahl der nichtisolierten Requestknoten ist in allen infrage kommenden Nachfolgegraphen identisch. In Teilskizze b ist eine solche Auswahl für das Beispiel zu sehen. Durch diese Bedingung an eine



Skizze 32: Die Teilskizzen a, b und c zeigen eine Beispielsituation und drei mögliche Entscheidungen zum Zeitpunkt i mit den daraus folgenden, strukturell verschiedenen Nachfolgesituationen (mit den Parametern $g = 3$ und $d = 4$).

Entscheidung wird die Flexibilität bzw. der Freiheitsgrad für die nachfolgenden Situationen, deren Eingaben noch nicht bekannt sind, erhöht.

Eine weitere Verbesserung für das Widerstehen gegen den Gegenspieler ist für das Beispiel in Teilskizze c gezeigt. Die nichtisolierten Serverknoten sollten möglichst kleine Zeitindizes besitzen. Diese Forderung behindert den Gegenspieler beim Aufbau einer größeren Blockstruktur, die sich gegen die noch zur Verfügung stehenden Servicezeitpunkte richtet, da sie selbst aus mehreren Requestknoten besteht und deren Aufbau eine entsprechende Anzahl von Zeitschritten benötigt.

Diese Beobachtungen führen zu der Definition von Attributen bzw. Bewertungen für die einzelnen Konfigurationen. Jedem Graphen einer Konfiguration wird die Größe seines Maximum-Matchings, seine Expansionszahl — es genügt die Anzahl der nichtisolierten Serverknoten — und eine numerische Bewertung der Zeitindizes nichtisolierter Serverknoten zugeordnet. Die letzte Aufgabe kann z. B. dadurch gelöst werden, daß die Folge $(s_i, \ldots, s_{i+d})$ von Serverknoten als Binärdarstellung einer Zahl aufgefaßt wird, wobei ein nichtisolierter Serverknoten als 1 und ein isolierter Serverknoten als 0 interpretiert wird. Mit diesen drei Werten lassen sich Online-Algorithmen beschreiben, in denen die Entscheidung durch die Maximierung einer Zielfunktion in diesen charakteristischen Werten ausgedrückt wird.

Die Computerexperimente ergaben mit derartigen Online-Algorithmen für einige

Modellinstanzen verbesserte Competitive Ratios, als mit einer einfachen Implementierung von LMM erzielt wurden. Bei Modellinstanzen mit etwas umfangreicheren Parametern, wie z. B. $g = 4$ und $d = 5$, erreicht die soeben beschriebene Menge von Zielfunktionen immer noch keine eindeutigen Definitionen für die Entscheidungen eines Online-Algorithmus. Außerdem ist es notwendig die Freiheitsgrade in der Struktur des Nachfolgegraphen noch exakter zu spezifizieren. Das führt zu einer weiteren Beschreibung der Flexibilität, die eine Konfiguration aufweist. Dabei wird die Expansionszahl und der Wert für die Zeitindize-Bewertung nichtisolierter Serverknoten ersetzt. Statt dessen werden für den Graphen einer jeden Konfiguration alle Maximum-Matchings bestimmt und deren Struktur bezüglich der Nutzung der Serverknoten gespeichert. Die so entstehende Menge von „Mustern“ kann gemäß der Bewertung von Zeitindizes genutzter Serverknoten in eine kanonische Ordnung gebracht werden. Die Expansionszahl geht indirekt in die Größe dieser Menge von Mustern ein, ebenso wird der dritte charakteristische Wert durch die Ordnung in der Mustermenge berücksichtigt.

Nun können die Entscheidungen eines Online-Algorithmus durch Vergleich der Mustermengen der möglichen Nachfolgezustände getroffen werden, nachdem Folgekonfigurationen, die nicht dem Kriterium der Klasse der LMM-Algorithmen genügen, aus der Betrachtung entfernt wurden. Der Vergleich zweier Mustermengen basiert dann auf klaren Dominanzrelationen, wie echte Teilmengenbeziehung, Differenzen in der Kardinalität der Mengen und der Zeitindize-Bewertung einzelner Muster.

Für den Fall, daß die Mustermengen zweier Nachfolgekonfigurationen identisch sind, werden deren Nachfolger unter der leeren Eingabe samt ihrer Charakteristika zum Vergleich herangezogen. Damit werden Substrukturen der Graphen, die beim Voranschreiten der Zeit zur Geltung gelangen, in der Entscheidungsfindung berücksichtigt.

Da zu Beginn eine Äquivalenzrelation über die Konfigurationen eingeführt wurde, und die Entscheidungen des Online-Algorithmus nur noch für die einzelnen Äquivalenzklassenrepräsentanten durchgeführt werden, ergibt sich nun eine *eindeutige* Definition für den Online-Algorithmus. Die Rücktransformation einer solchen Entscheidung für jede Konfiguration einer Äquivalenzklasse ist mittels Betrachtung der vollständigen Äquivalenzklasse der Nachfolgekonfiguration problemlos möglich.

Für eine kleine Menge von Parameterkombinationen des ORSM-Problems mit konstantem Knotengrad g und begrenzter Kantenlänge d konnte für den soeben beschriebenen Online-Algorithmus der exakte Competitive Ratio mittels Computeranalyse bestimmt werden. Die Resultate sind in Tabelle 6 dargestellt und stellen die besten bekannten Ergebnisse für diese Variante des ORSM-Problems dar.

$d-g=$		0	1	2	3
$g=$	2	$\mathbf{\frac{5}{4} = 1.25}$	$\mathbf{\frac{4}{3} = 1.\overline{3}}$	$\mathbf{\frac{10}{7} \approx 1.429}$	$\mathbf{\frac{3}{2} = 1.5}$
	3	$\mathbf{\frac{7}{6} = 1.1\overline{6}}$	$\mathbf{\frac{9}{7} \approx 1.286}$	$\frac{15}{11} = 1.\overline{36}$	$\frac{10}{7} \approx 1.429$
	4	$\mathbf{\frac{9}{8} = 1.125}$	$\mathbf{\frac{11}{9} = 1.\overline{2}}$		
	5	$\mathbf{\frac{11}{10} = 1.1}$	$\mathbf{\frac{13}{11} = 1.\overline{18}}$		
	6	$\mathbf{\frac{13}{12} = 1.08\overline{3}}$			
	7	$\mathbf{\frac{15}{14} \approx 1.071}$			
	8	$\mathbf{\frac{17}{16} = 1.0625}$			

Tabelle 6: Competitive Ratios, die durch Computeranalyse des besten bekannten Online-Algorithmus bewiesen wurden. Zahlen im Fettdruck sind identisch zu den Werten der unteren Schranke.

8.4 Zusammenfassung

Welche Erkenntnisse ergeben sich durch die Auswertung der in diesem Kapitel aufgezeigten Resultate?

Für das konkret untersuchte ORSM-Problem mit konstantem Knotengrad g und begrenzter Kantenlänge d läßt sich eine Abhängigkeit der erzielbaren Competitive Ratios vom Verhältnis der beiden Modellparameter d und g erkennen. Ab $d \geqq 4g-4$ (für $g \geqq 3$, bzw. bei $g=2$ und $d \geqq 5$) zeigt die Konstruktion der unteren Schranke in Satz 32 (bzw. Korollar 33 und Satz 35) einen Competitive Ratio von 1.5 auf. Dieser Wert wird vom Online-Algorithmus LMM auch als maximaler Leistungsverlust garantiert. Für Modellinstanzen mit großen Kantenlängen und damit einhergehender großer Variationsvielfalt in den Strukturen einer Einzeleingabe ist deshalb die garantierbare Leistungsfähigkeit von Online-Algorithmen nicht besser als für das Standard-ORSM-Problem, und schon der einfache LMM-Algorithmus ist in der Lage diese Garantie zu übernehmen.

Für ein kleineres Verhältnis der Modellparameter d und g existieren Online-Algorithmen mit besseren Compeititve Ratios. Diese Tatsache folgt aus den Ergebnissen der Computeranalyse. Die dazu passenden bzw. nur leicht abweichenden unteren Schranken zeigen, daß diese Resultate — siehe Tabelle 5 und 6 — die korrekten Tendenzen in den Werten der Competitive Ratios reflektieren.

Um jedoch die guten bzw. optimalen Lösungsqualitäten garantieren zu können, müssen sich die Online-Algorithmen sehr viel geschickter verhalten, als dies der

Algorithmus LMM fordert. In Kapitel 8.3 wurden derartige Techniken zur Entscheidungsfindung, die präzise auf die in diesem Kapitel behandelte Problemklasse zugeschnitten sind, schrittweise hergeleitet und beschrieben. Aus ihrer Charakteristik lassen sich allgemeinere Designregeln für Online-Algorithmen ableiten, die Online-Echtzeit-Planungsprobleme lösen bzw. allgemeine Online-Probleme bearbeiten.

Für Planungsprobleme mit Aufträgen von Einheitsgröße und individuellen Fristen ist es für einen guten Online-Algorithmus wichtig

1. im aktuellen Zeitschritt — also lokal — eine optimale Lösung zu realisieren, so daß
2. für die folgenden Zeitschritte — also global — auch optimale Lösungen für die bisher bekannten Eingabeteile existieren, und
3. diese optimalen Lösungen in der Zukunft einen maximalen Grad an Freiheiten bzw. Flexibilität aufweisen.

Die erste Forderung stellt eine Art Greedy-Verhalten dar, welches jedoch für die zugrundeliegenden Matching-Probleme keine Einschränkung für zukünftige Lösungsteile mit sich bringt. Auch die zweite Bedingung sucht in ähnlicher Manier auf der Grundlage der bisher bekannt gewordenen Teile der Eingabesequenz und den in der Vergangenheit festgelegten Entscheidungen nach einer optimalen Lösung. Die entscheidende dritte Forderung verlangt unter den möglichen Entscheidungen, die ein lokales und globales Optimum besitzen, diejenige auszuwählen, deren Menge von optimalen zukünftigen Lösungen viele verschiedene Strukturen aufweist. Damit ermöglicht sich ein Online-Algorithmus auf die verschiedensten Eingabestrukturen flexibel zu reagieren und möglichst viele, zum Entscheidungszeitpunkt noch nicht bearbeitete Aufträge gleichzeitig mit den noch unbekannten Aufträgen zu bedienen. Aus der Sichtweise des Gegenspielers läßt sich diese dritte Forderung wie folgt interpretieren. Der hohe Freiheitsgrad in den zukünftigen optimalen Lösungen erschwert es dem Gegenspieler nur wenige Ressourcen zu blockieren und schon damit die Entscheidungen des Online-Algorithmus als ungünstig herauszustellen. Konnte sich der Online-Algorithmus die Flexibilität für die noch nicht bearbeiteten Aufträge bewahren, so kann er einem Blockierungsversuch des Gegenspielers entweder mit gutem Erfolg ausweichen, oder der Gegenspieler muß so viele neue Aufträge hinzufügen, daß er selbst nicht alle bedienen kann. Dann hat der Online-Algorithmus, ebenso wie der Gegenspieler, sehr viele der Ressourcen benutzt und der Competive Ratio wird klein gehalten.
Wie schon bei der Entwicklung und Beschreibung des besten bekannten Online-Algorithmus für die ORSM-Problemvariante mit konstantem Knotengrad g und begrenzter Kantenlänge d gesehen, ist die Formalisierung des dritten Optimierungskriteriums des Freiheitsgrades zukünftiger Lösungen schwierig und offensichtlich hochgradig von der konkreten Struktur eines Online-Problems abhängig. Aber es ist genau dieses Detail, welches unabdingbar für gute bzw. optimale

Online-Algorithmen ist, solange die Problemstruktur nicht von vornherein die Entwicklung von geschickten Algorithmen negiert, weil schon einfache Online-Algorithmen die bestmöglichen Competitive Ratios erreichen. Da die Competitive Analysis eine Form der Analyse des schlechtesten Falles darstellt, kann es für praktische Anwendungen trotzdem interessant sein bei gleichen Competitive Ratios einen aufwendigeren Online-Algorithmus einzusetzen.

Für allgemeine Online-Probleme sind die oben aufgestellten drei Regeln zur Konstruktion von Online-Algorithmen zu stringent formuliert. Es kann sich durchaus positiv auf den Competitive Ratio auswirken, wenn ein Online-Algorithmus die ersten beiden Forderungen nicht optimal erfüllt, sondern dort nur Lösungen mit einer definierten Mindestqualität auswählt, falls sich durch diese Maßnahme eine dramatische Erhöhung im Freiheitsgrad der geforderten dritten Bedingung einstellt. Ein einfaches Beispiel stellt der Algorithmus zur Lösung des Online-Lastbalancierungsproblems in [Alb00] dar. Während Graham's List-Scheduling-Algorithmus — der einer Greedy-Strategie entspricht — nur einen Competitive Ratio von 2 (bzw. $2 - 1/m$ bei m Maschinen) erzielt, kann mit einem definierten Verletzen der lokalen Optimalität der Entscheidung ein Competitive Ratio von substantiell unter 2 erreicht werden. Auch der WFA-Algorithmus für das eingehend untersuchte k-Server-Problem kann in dieser Weise interpretiert werden. Dies erfordert jedoch ein weitreichendes Verständnis der Struktur des Online-Problems und des Online-Algorithmus WFA. Deshalb wird in dieser Arbeit auf eine solche Betrachtung verzichtet.
Das wORSM-Problems aus Kapitel 6 stellt ein weiteres Beispiel dar. Der gierige und lokal optimale wLMM-Algorithmus kann einen Competitive Ratio von 2 nicht unterbieten. Dagegen verletzt der vorgeschlagene Online-Algorithmus PHI bewußt die lokale Optimalität der Entscheidung um einen Faktor von maximal ϕ.

Aus den oben beschriebenen Erkenntnissen und Beobachtungen lassen sich auch stark verallgemeinerte Designprinzipien ableiten. So sollte ein Online-Algorithmus eine Entscheidung mit guter Lösungsqualität — d. h. in ihrer Güte nur durch einen Faktor von einer optimalen Lösung abweichend — treffen, die sowohl gute Lösungen für noch nicht bearbeitete Teile der Eingabe ermöglicht (falls so etwas nach Definition des Online-Problems existiert) als auch einen hohen Freiheitsgrad für die Zukunft offen läßt. Eine exakte Formulierung von „gut“ und „Freiheitsgrad“ ist hochgradig problemspezifisch und stellt eine erhebliche Herausforderung beim Algorithmenentwurf dar.
Auch wenn diese sehr allgemeinen Regeln für die Konstruktion von Online-Algorithmen jedem als natürlich und offensichtlich erscheinen, wird diese intuitive Vorgehensweise durch die Untersuchungen innerhalb dieser Arbeit formal bestätigt und konkret ausformuliert.

9 Das Data-Access-Problem (DAP)

Übersicht: Das Data-Access-Problem (DAP) stellt ein abstraktes Zugriffsproblem innerhalb eines Datenservers mit parallelen Ressourcen und Realzeitanforderungen dar. Da auch dieses Modell als Spezialfall des ORSM-Problems betrachtet werden kann, überträgt sich die obere Schranke des Competitive Ratios von 1.5. Im ersten Teil dieses Kapitels werden allgemeingültige, konstante untere Schranken für den Competitive Ratio des DAP bewiesen, die substantiell von der Trivialschranke von 1 entfernt sind. Darüber hinaus sind einige weitere Untersuchungsresultate zu diesem Modell bekannt und in [BRS99] veröffentlicht. Die beste obere Schranke für den Competitive Ratio bei $c = 2$ sowie eine Diskussion zum Design von Online-Algorithmen für das DAP sind im zweiten Teilkapitel dargestellt.

9.1 Untere Schranken

Dieses Teilkapitel umfaßt die Darstellung mehrerer Gegenspielerstrategien, um untere Schranken für den Competitive Ratio des DAP aufzuzeigen. Bei diesem Modell gilt es zu beachten, daß der Parameter d — abweichend von der Interpretation in den ORSM-Modellvarianten — die Anzahl von Zeitschritten bezeichnet, in denen eine Anfrage bedient werden darf. Eine Anfrage zum Zeitpunkt i kann somit im Intervall $[i, i+d-1]$ bearbeitet werden. Mit dieser Definition vereinfacht sich die Darstellung der Untersuchungen und Resultate.
Die grundlegende Idee der Gegenspielerstrategie wird im Beweis des folgenden Satzes besonders deutlich, da ein Modell mit minimalen Parametern betrachtet wird.

Satz 36:

Kein deterministischer Online-Algorithmus für das DAP mit den Modellparametern $c = 2$, $d = 2$ und $m \geqq 3$ kann einen Competitive Ratio unter 1.2 erreichen.

Beweis: Die Gegenspielerstrategie arbeitet in Runden von je zwei Zeitschritten und benutzt drei Speicherressourcen S_I, S_{II} und S_{III}. In einem Initialschritt werden S_I und S_{II} blockiert, indem ein Block von vier Anfragen $r_{1,1}$, $r_{2,1}$, $r_{3,1}$ und $r_{4,1}$ mit $\mathcal{S}_{r_{i,1}} = \{S_I, S_{II}\}\ \ \forall\, i \in \{1,2,3,4\}$ im ersten Zeitschritt gestellt werden. Diese Anfragen können genau von den beiden Ressourcen in den zwei zur Verfügung stehenden Zeitschritten bedient werden.
Im zweiten Zeitschritt beginnt die erste Runde. Der Gegenspieler stellt zwei Anfragen $r_{1,2}$ und $r_{2,2}$ mit $\mathcal{S}_{r_{1,2}} = \{S_I, S_{III}\}$ und $\mathcal{S}_{r_{2,2}} = \{S_{II}, S_{III}\}$. Ein Online-Algorithmus muß die Entscheidung treffen, welche der beiden Anfragen mit der Ressource $s_{3,2}$ bearbeitet wird. Im nächsten Zeitschritt wird die noch nicht bearbeitete Anfrage blockiert, indem an beide zugehörigen Ressourcen ein Block aus vier Anfragen gestellt wird. Der Online-Algorithmus kann in diesem Zeitschritt eine der drei Ressourcen nicht benutzen, obwohl dies in einer optimalen Lösung der Fall ist. Zusätzlich sind durch den Block die Voraussetzungen für die nächste Runde gegeben, in der diese Strategie mit wechselnden Rollen der Speicherressourcen wiederholt wird.
Pro Runde werden sechs Anfragen gestellt, und es sind drei Ressourcen à zwei Zeitschritte involviert. Von diesen kann der Online-Algorithmus eine Ressource einmal nicht benutzen, da er die einzige dort bedienbare Anfrage schon einen Zeitschritt zuvor bearbeitet hat.[41] In einer optimalen Lösung werden hingegen alle Aufträge bearbeitet. Durch permanente Wiederholung dieser Runden werden die Anfragen des initialen Blockes irrelevant und es ergibt sich:

$$\mathcal{R} \ \geqq\ \frac{6}{5} \ =\ 1.2$$

als untere Schranke. ■ Satz 36

[41] Als Alternative könnte der Online-Algorithmus im ersten Zeitschritt der Runde eine Ressource nicht genutzt haben.

Bei Vergrößerung der Modellparameter des DAP kann diese Gegenspielerstrategie auf verschiedene Weise verallgemeinert werden. In jedem Fall werden die Blöcke größer, d. h. sie umfassen mindestens c Speicherressourcen und cd Anfragen.

Ein DAP mit dem Parameter $c = m$ ist trivial lösbar, denn jede Anfrage kann von jeder Speicherressource bedient werden. Deshalb kann der EDF-Algorithmus aus Kapitel 7 benutzt werden um optimale Lösungen auch im Online-Fall zu erzeugen. Schon ab $m \geqq c+1$ sind keine 1-competitiven Online-Algorithmen möglich. Dazu verwendet der Gegenspieler initial ein Block von cd Anfragen an die ersten c Speicherressourcen. Zum Zeitpunkt d, also im letzten Zeitschritt des Blockes werden zwei Anfragen $r_{1,d}$ und $r_{2,d}$ gestellt. Anfrage $r_{1,d}$ kann die erste Ressource S_I nicht benutzen und Anfrage $r_{2,d}$ kann von Speicherressource S_{II} nicht bedient werden. Der Online-Algorithmus muß also entscheiden, wie die Ressource $s_{c+1,d}$ einzusetzen ist. Im nächsten Zeitschritt werden alle Speicherressourcen der nicht beantworteten Anfrage blockiert. Mit dieser Verallgemeinerung des Beweises zu Satz 36 ergibt sich das Korollar:

Korollar 37:

Jeder deterministische Online-Algorithmus für das DAP mit $c, d \in \mathbb{N}$, $c \geqq 2$, $d \geqq 2$ *und* $m \geqq c+1$ *besitzt einen Competitive Ratio* $\mathcal{R}$ *von:*

$$\mathcal{R} \geqq \frac{cd+2}{cd+1} .$$

Diese untere Schranke konvergiert für steigende Parameter gegen 1 . Ein Modell für das DAP, in dem die Anzahl der Speicherressourcen wächst, jeder Auftrag aber nur von einer Speicherressource nicht bedient werden kann, nähert sich in seiner Struktur dem Fall $m = c$, für das der EDF-Algorithmus 1-competitive ist. Deshalb ist die Konvergenz dieser unteren Schranke auch intuitiv einleuchtend.

Können die Anfragen nur von einem konstanten Bruchteil der vorhandenen Speicherressourcen bedient werden, so ergeben sich konstante untere Schranken für den Competitive Ratio. Dies ist in allen folgenden Gegenspielerstrategien zu sehen.

Bei größeren Modellparametern wird nicht nur die Anzahl der benötigten Anfragen für einen Block größer. Anstelle der beiden Anfragen $r_{1,d}$ und $r_{2,d}$ werden dann zwei Gruppen von gleichartigen Anfragen benutzt. Um diese Gruppen zu unterscheiden wird im folgenden von *roten* und *blauen* bzw. farbigen Anfragen gesprochen. Diese Anfragen werden pro Gruppe so auf die Speicherressourcen verteilt, daß sie alle konfliktfrei in der Zeit von d Schritten bedient werden können.

Satz 38:

Kein deterministischer Online-Algorithmus für das DAP mit $c, d \in \mathbb{N}$, $c \geqq 2$, $d \geqq 2$ *und* $d \equiv 0 \bmod 2$ *kann den folgenden Competitive Ratio* $\mathcal{R}$ *unterschreiten:*

$$\mathcal{R} \geqq \begin{cases} \dfrac{3cd}{3cd - 2\left\lceil \frac{1}{8}cd \right\rceil} & \text{für } c \equiv 0 \bmod 2 \text{ und } m \geqq \frac{3}{2}c \\ \dfrac{3cd}{3cd - \left\lceil \frac{1}{4}cd \right\rceil} & \text{für } m \geqq 3c \end{cases}$$

Beweis: Die Gegenspielerstrategie verallgemeinert das Verfahren aus dem Beweis zu Satz 36. Für den Fall, daß c gerade ist, werden drei Gruppen von Speicherressourcen S_I, S_{II} und S_{III} mit je $c/2$ einzelnen Speicherressourcen benutzt. Initial werden S_I und S_{II} durch einen Block aus cd Anfragen blockiert. Nach $d/2$ Zeitschritten beginnt die erste Runde. Die rote Gruppe von $cd/4$ Anfragen kann durch S_I und S_{III} bearbeitet werden. Die blaue Anfragegruppe der gleichen Mächtigkeit von $cd/4$ ist durch die Speicherressourcen aus S_{II} und S_{III} bedienbar. Nach weiteren $d/2$ Zeitschritten sind von den Ressourcen aus S_{III} maximal $c/2 \cdot d/2 = cd/4$ der farbigen Anfragen bearbeitet worden. Deshalb muß eine der farbigen Gruppen mindestens $\lceil cd/8 \rceil$ noch nicht bediente Anfragen besitzen[42]. Diese Anfragegruppe wird dann mit einem neuen Block der Größe cd blockiert. Die Runde endet nach insgesamt d Schritten, und durch den Block sind die Voraussetzungen für die nächste Runde geschaffen. Darin wechseln höchstens die Speicherressourcegruppen untereinander ihre Rollen.
Der initiale Block hat bei beliebig vielen Runden keinen Einfluß auf das Resultat und es genügt der Vergleich der Leistungen einer Runde. Es werden pro Runde ein Block und zwei farbige Anfragegruppen, d. h. $cd + 2(cd/4) = 3/2 \cdot cd$ Anfragen gestellt, die von einer optimalen Lösung vollständig bedient werden. Der Gegenspieler garantiert durch die obige Strategie, daß ein Online-Algorithmus mindestens $\lceil cd/8 \rceil$ Anfragen einer Runde nicht bearbeitet, woraus

$$\mathcal{R} \geqq \frac{\frac{3}{2}cd}{\frac{3}{2}cd - \left\lceil \frac{1}{8}cd \right\rceil} = \frac{3cd}{3cd - 2\left\lceil \frac{1}{8}cd \right\rceil}$$

für $c \equiv 0 \bmod 2$, $d \equiv 0 \bmod 2$ und $m \geqq 3/2 \cdot c$ folgt.

Für den Fall, daß c ungerade ist, wird die selbe Gegenspielerstrategie angewandt, die jedoch auf doppelt so großen Speicherressource- und Anfragegruppen basiert. Das heißt, jede Speicherressourcegruppe S_I, S_{II} und S_{III} besitzt c Einzelressourcen, der Block besteht aus $2cd$ und jede farbige Anfragegruppe aus $cd/2$ Anfragen. Pro Runde werden somit $3cd$ Anfragen gestellt, von denen ein Online-Algorithmus $\lceil cd/4 \rceil$ nicht bearbeiten kann. Es folgt:

[42] Oder es wurden statt dessen Anfragen aus den Blöcken nicht bearbeitet, was auf die Leistungsfähigkeit des Online-Algorithmus und die Analyse keinen verbessernden Einfluß hat.

$$\mathcal{R} \geqq \frac{3cd}{3cd - \left\lceil \frac{1}{4}cd \right\rceil}$$

für $c \equiv 1 \bmod 2$, $d \equiv 0 \bmod 2$ und $m \geqq 3c$. ■ Satz 38

Beide Terme der unteren Schranken sind immer mindestens $12/11$, weshalb für gerade Werte von d: $\mathcal{R} \geqq 12/11 = 1.\overline{09}$ gilt.

Falls d ungerade ist, lassen sich die beiden Phasen einer Runde nicht mehr symmetrisch in $d/2$ Zeitschritte für Entscheidungen und $d/2$ Zeitschritte, in denen der Online-Algorithmus eine verminderte Leistung erzielt, aufteilen. Der Gegenspieler verliert etwas von seiner Leistungsfähigkeit, da die Entscheidungsphase einer Runde mit $\lfloor d/2 \rfloor$ Schritten einen Zeitschritt kürzer ist als die Verlustphase mit $\lceil d/2 \rceil$ Zeitschritten. Bei $m \geqq 3c$ und einer Größe der Speicherressourcegruppen von c besteht der Block aus $2cd$ und jede Gruppe farbiger Anfragen aus $\lfloor d/2 \rfloor c = (d-1)c/2$ Anfragen. Deshalb kann nach $\lfloor d/2 \rfloor$ Zeitschritten einer Runde vom Gegenspieler garantiert werden, daß eine farbige Anfragegruppe mit mindestens $\lceil (d-1)c/4 \rceil$ offenen Anfragen existiert, die wegen der folgenden Blockierung des Gegenspielers vom Online-Algorithmus nicht bedient werden können. Für den Fall, daß der Parameter c gerade ist, können die Kardinalitäten der Gruppen halbiert werden, und die Rundungseffekte der Gauß-Klammern erhalten einen vergrößerten Einfluß.
Mit der weiteren Argumentation aus dem Beweis des Satzes 38 folgt das Korollar:

Korollar 39:

Kein deterministischer Online-Algorithmus für das DAP mit $c, d \in \mathbb{N}$, $c \geqq 2$, $d \geqq 2$ und $d \equiv 1 \bmod 2$ kann den folgenden Competitive Ratio $\mathcal{R}$ unterschreiten:

$$\mathcal{R} \geqq \begin{cases} \dfrac{(3d-1)c}{(3d-1)c - 2\left\lceil \frac{1}{8}(d-1)c \right\rceil} & \text{für } c \equiv 0 \bmod 2 \text{ und } m \geqq \frac{3}{2}c \\ \dfrac{(3d-1)c}{(3d-1)c - \left\lceil \frac{1}{4}(d-1)c \right\rceil} & \text{für } m \geqq 3c \end{cases}$$

Diese Terme sind nach unten durch $16/15 = 1.0\overline{6}$ beschränkt, und sie konvergieren bei Wachstum beider Modellparameter zügig gegen $12/11$.

Ist m ausreichend groß, so kann eine größere Anzahl von Speicherressourcegruppen benutzt werden, und die Gegenspielerstrategien können anstatt der roten und blauen Anfragegruppe mit mehr als zwei farbigen Anfragegruppen arbeiten. Der Grundaufbau einer Runde bleibt dabei gleich:
Sei f die Anzahl farbiger Anfragegruppen, $d \equiv 0 \bmod f$ und $m \geqq (2f-1)c$. Dann werden $(2f-1)$ Speicherressourcegruppen à c Speicherressourcen benutzt. Die ersten f Gruppen werden mit fcd Anfragen blockiert. Jede farbige Anfragegruppe besteht aus $(f-1) \cdot c \cdot d/f$ Anfragen. Diese werden nach $d - d/f$ Zeitschritten vom Gegenspieler so gestellt, daß jede Anfragegruppe entweder im Zeit-

intervall bis zum Blockende (das sind d/f Zeitschritte) durch die $f-1$ noch freien Speicherressourcegruppen, oder in den nachfolgenden $d-d/f$ Zeitschritten durch eine bisher blockierte Speicherressourcegruppe vollständig bearbeitet werden kann. Dabei hat jede farbige Anfragegruppe ihre eigene, eindeutige Gruppe mit c Speicherressourcen.
Nach d/f Zeitschritten der Runde — also mit Beendigung des vorhergehenden Blockes — bleibt eine farbige Anfragegruppe mit mindestens

$$\left\lceil \frac{(f-1)^2}{f^2} cd \right\rceil$$

unbedienten Anfragen übrig.

Bemerkung: Es werden f Gruppen à $cd - cd/f$ Anfragen gestellt, von denen in den ersten d/f Zeitschritten der Runde neben dem Block $(f-1)\cdot c\cdot d/f$ bearbeitet werden. Damit verbleiben $f(cd-cd/f)-(f-1)\cdot c\cdot d/f = (f-2+1/f)cd = (f-1)^2/f\cdot cd$ farbige Anfragen offen. Ergo muß mindestens eine Anfragegruppe existieren, in der wenigstens der Durchschnittswert, nach oben aufgerundet, von $\lceil (f-1)^2/f^2 \cdot cd \rceil$ Anfragen nicht bearbeitet wurde.

Diese Anfragegruppe wird im folgenden vom Gegenspieler blockiert und nach insgesamt d Zeitschritten endet die Runde. Durch den Block ab Zeitschritt $d/f+1$ der Runde ist die Vorbedingung der nächsten Runde geschaffen. Falls zusätzlich $c \equiv 0 \bmod f$ gilt, können die Kardinalitäten der Speicherressource- und Anfragegruppen sowie des Blockes jeweils um den Faktor f vermindert werden. Aus diesen Überlegungen ergibt sich:

Korollar 40:

Kein deterministischer Online-Algorithmus für das DAP mit $c, d, f \in \mathbb{N}$, $c \geqq 2$, $f \geqq 2$, $d \geqq f$ und $d \equiv 0 \bmod f$ kann den folgenden Competitive Ratio $\mathcal{R}$ unterschreiten:

$$\mathcal{R} \geqq \begin{cases} \dfrac{(2f-1)cd}{(2f-1)cd - f\left\lceil \frac{(f-1)^2}{f^3} cd \right\rceil} & \text{für } c \equiv 0 \bmod f \text{ und } m \geqq \frac{2f-1}{f} c \\ \dfrac{(2f-1)cd}{(2f-1)cd - \left\lceil \frac{(f-1)^2}{f^2} cd \right\rceil} & \text{für } m \geqq (2f-1)c \end{cases}$$

Diese Terme können durch Weglassen der Gauß-Klammern nach unten abgeschätzt werden:

$$\mathcal{R} \geqq \frac{(2f-1)cd}{(2f-1)cd - \frac{(f-1)^2}{f^2} cd} = \frac{2f^3 - f^2}{2f^3 - 2f^2 + 2f - 1}\ , \qquad (19)$$

und Funktion (19) besitzt ein Maximum bei:

$$f = \frac{3+\sqrt{5}}{2} = 1 + \phi \approx 2.618\ .$$

Mit der oben beschriebenen Gegenspielerstrategie ließen sich deshalb die besten unteren Schranken für den Competitive Ratio des DAP zeigen, wenn 2.618 verschiedene farbige Anfragegruppen verwendet würden. Es ist jedoch nicht möglich, eine gebrochene Anzahl solcher Gruppen zu benutzen, aber der nächstliegende Wert von drei verschiedenen Anfragegruppen ist von Interesse. Zur besseren Verständlichkeit wird dieser Fall nochmals als Satz mit einem in den Werten angepaßten Beweis vorgestellt. Danach folgen Betrachtungen zu den Parameterkombinationen, bei denen $d \equiv 0 \bmod 3$ nicht gilt.

Satz 41:

Kein deterministischer Online-Algorithmus für das DAP mit $c, d \in \mathbb{N}$, $c \geqq 2$, $d \geqq 3$ *und* $d \equiv 0 \bmod 3$ *kann den folgenden Competitive Ratio* $\mathcal{R}$ *unterschreiten:*

$$\mathcal{R} \geqq \begin{cases} \dfrac{5cd}{5cd - 3\left\lceil \frac{4}{27}cd \right\rceil} & \text{für } c \equiv 0 \bmod 3 \text{ und } m \geqq \frac{5}{3}c \\ \dfrac{5cd}{5cd - \left\lceil \frac{4}{9}cd \right\rceil} & \text{für } m \geqq 5c \end{cases}$$

Beweis: Für den Fall $c \equiv 0 \bmod 3$ wird die Gegenspielerstrategie gezeigt. Es werden fünf Gruppen von Speicherressourcen S_I, S_{II}, S_{III}, S_{IV} und S_V mit jeweils $c/3$ einzelnen Speicherressourcen benutzt. Initial werden die Gruppen S_I, S_{II} und S_{III} durch einen Block von cd Anfragen blockiert. Die erste Runde beginnt im Zeitschritt $2/3 \cdot d + 1$ und jede Runde ist d Schritte lang.
Zu Beginn einer Runde werden drei farbige Anfragegruppen „rot“, „blau“ und „grün“ à $2/9 \cdot cd$ Anfragen gestellt. Die rote Gruppe kann von den Ressourcegruppen S_I, S_{IV} und S_V, die blaue von S_{II}, S_{IV} und S_V und die grüne von S_{III}, S_{IV} und S_V bedient werden. Innerhalb der ersten $d/3$ Zeitschritte werden von Ressourcen aus S_{IV} und S_V höchstens $2/9 \cdot cd$ Anfragen bearbeitet. Aus den verbleibenden $3 \cdot 2/9 \cdot cd - 2/9 \cdot cd = 4/9 \cdot cd$ farbigen Anfragen folgt die Existenz einer Anfragegruppe mit mindestens $\lceil 4/27 \cdot cd \rceil$ nicht bearbeiteten Anfragen. Die drei Ressourcegruppen, die diese Anfragegruppe bedienen können, werden ab dem Zeitschritt $d/3 + 1$ der Runde durch cd neue Anfragen blockiert.
Eine optimale Lösung kann die Anfragen dieser Gruppe im Zeitintervall $[1, d/3]$ der Runde von den Ressourcen der Gruppe S_{IV} und S_V bedienen lassen und die Anfragen der anderen farbigen Gruppen werden von den nichtblockierten Speicherressourcegruppen im Zeitintervall $[d/3+1, d]$ der Runde bearbeitet. Durch den Block wird die Eingangsbedingung für die nächste Runde geschaffen, in der die Speicherressourcegruppen höchstens untereinander ihre Rollen vertauschen.
Pro Runde werden $cd + 3 \cdot 2/9 \cdot cd = 5/3 \cdot cd$ Anfragen gestellt, von denen ein Online-Algorithmus $\lceil 4/27 \cdot cd \rceil$ nicht bedienen kann. Daraus folgt die gesuchte untere Schranke für den Competitive Ratio.
Gilt $c \equiv 0 \bmod 3$ nicht, so kann obige Konstruktion ebenfalls durchgeführt werden, falls alle Gruppengrößen verdreifacht werden. ■ Satz 41

Die Terme dieser unteren Schranke für $d \equiv 0 \bmod 3$ sind immer größer als

$$\mathcal{R} \geqq \frac{45}{41} = 1.\overline{09756} \ .$$

Ähnlich wie im Korollar 39 gesehen, büßt auch diese Gegenspielerkonstruktion etwas von ihrer Leistungsfähigkeit ein, wenn $d \equiv 0 \bmod 3$ nicht gilt. An dieser Stelle soll auf eine ausführliche Beweisführung zur Gegenspielerkonstruktion verzichtet werden, da keine neuen Beweisideen Anwendung finden. Es folgen lediglich die Resultate der unteren Schranken:

Korollar 42:

Kein deterministischer Online-Algorithmus für das DAP mit $c, d \in \mathbb{N}$, $c \geqq 2$, $d \geqq 2$ *kann im Fall* $d \equiv 1 \bmod 3$ *den Competitive Ratio*

$$\mathcal{R} \geqq \begin{cases} \dfrac{(5cd-2)}{(5cd-2)-3\left\lceil \frac{4}{27}(d-1)c \right\rceil} & \text{für } c \equiv 0 \bmod 3 \text{ und } m \geqq \frac{5}{3}c \\ \dfrac{(5cd-2)}{(5cd-2)-\left\lceil \frac{4}{9}(d-1)c \right\rceil} & \text{für } m \geqq 5c \end{cases}$$

unterschreiten, und im Fall $d \equiv 2 \bmod 3$ *gilt als untere Schranke für den Competitive Ratio:*

$$\mathcal{R} \geqq \begin{cases} \dfrac{(5d-1)c+9\lfloor \frac{c}{9} \rfloor}{(5d-1)c+6\lfloor \frac{c}{9} \rfloor-3\left\lceil \frac{1}{27}(4d-5)c \right\rceil} & \text{für } c \equiv 0 \bmod 3 \text{ und } m \geqq \frac{5}{3}c \\ \dfrac{(5d-1)c+3\lfloor \frac{c}{3} \rfloor}{(5d-1)c+2\lfloor \frac{c}{3} \rfloor-\left\lceil \frac{1}{9}(4d-5)c \right\rceil} & \text{für } m \geqq 5c \end{cases}$$

Die Konstruktionen für die unteren Schranken im Fall $d \equiv 1 \bmod 3$ folgen kanonisch den für das Korollar 39 vorgestellten Modifikationen. Die Entscheidungsphase einer jeden Runde ist $\lfloor d/3 \rfloor = (d-1)/3$ Zeitschritte lang und pro Anfragegruppe werden $\lfloor d/3 \rfloor \cdot 2c = 2/3 \cdot (d-1)c$ Anfragen gestellt (bzw. falls zusätzlich $c \equiv 0 \bmod 3$ gilt, sind es $\lfloor d/3 \rfloor \cdot 2/3 \cdot c = 2/9 \cdot (d-1)c$ Anfragen).

Für den letzten Fall $d \equiv 2 \bmod 3$ benutzt der Gegenspieler für die im Korollar 42 genannten Terme der unteren Schranken des DAP leicht abgewandelte Konstruktionsparameter. Die Entscheidungsphase einer Runde besteht aus $\lfloor d/3 \rfloor + 1 = (d+1)/3$ Zeitschritten und jede der drei Anfragegruppen besitzt $\lfloor d/3 \rfloor \cdot 2c + \lfloor 4/3 \cdot c \rfloor = (2d-1) \cdot c/3 + \lfloor c/3 \rfloor$ Anfragen. (Falls $c \equiv 0 \bmod 3$ gilt und die Speicherressourcegruppen nur eine Kardinalität von $c/3$ aufweisen, werden entsprechend $(2d-1) \cdot c/9 + \lfloor c/9 \rfloor$ Anfragen pro Gruppe gestellt.) In einer optimalen Lösung werden alle Ressourcen der fünf Speicherressourcegruppen permanent ausgelastet.
Auch im Fall $d \equiv 2 \bmod 3$ kann die Entscheidungsphase einer Runde auf $\lfloor d/3 \rfloor = (d-2)/3$ Zeitschritte verkürzt und pro Gruppe nur $\lfloor d/3 \rfloor \cdot 2c = 2/3 \cdot (d-2)c$

Anfragen gestellt werden; bzw. für $c \equiv 0 \bmod 3$ und Speicherressourcegruppen der Größe $c/3$ besitzen die Anfragegruppen entsprechend $2/9 \cdot (d-2)c$ Einzelanfragen. Damit ergeben sich die unteren Schranken:

$$\mathcal{R} \geqq \begin{cases} \dfrac{(5cd-4)}{(5cd-4) - 3\left\lceil \frac{4}{27}(d-2)c \right\rceil} & \text{für } c \equiv 0 \bmod 3 \text{ und } m \geqq \frac{5}{3}c \\ \dfrac{(5cd-4)}{(5cd-4) - \left\lceil \frac{4}{9}(d-2)c \right\rceil} & \text{für } m \geqq 5c \end{cases}$$

Die zuletzt genannten Terme sind für einige Parameterkombinationen des DAP größer als die für $d \equiv 2 \bmod 3$ im Korollar 42 genannten. Allerdings sind die Werte der Terme aus Korollar 42 in der Mehrzahl aller möglichen Parameterbelegungen dominierend.

Es bleibt zu beachten, daß alle Terme für die unteren Schranken im Competitive Ratio des DAP mit $d \not\equiv 0 \bmod 3$ bei Wachstum beider Modellparameter c und d gegen den Wert $45/41 = 1.\overline{09756}$ konvergieren.

Zusammenfassend läßt sich zu den unteren Schranke für den Competitive Ratio des DAP folgendes feststellen:
Nach Satz 38 und Korollar 39 sind die Gegenspielerstrategien für alle Kombinationen der Modellparameter in der Lage, konstante untere Schranken von $\mathcal{R} \geqq 16/15 = 1.0\overline{6}$ bzw. in der Hälfte der Fälle auch $\mathcal{R} \geqq 12/11 = 1.\overline{09}$ zu zeigen, falls eine ausreichende Anzahl m von Speicherressourcen vorhanden sind. Von diesen werden maximal $5c$ Stück benötigt. Für viele Parameterkombinationen sowie asymptotisch bei wachsenden Parameterwerten gilt auch eine untere Schranke von $\mathcal{R} \geqq 45/41 = 1.\overline{09756}$. In einigen Fällen von Modellparameterbelegungen wird diese untere Schranke überschritten und ist maximal 1.2 (Satz 36) für die in diesem Kapitel dargestellten Gegenspielerstrategien.
Durch die komplizierten Terme und Nebenbedingungen der unteren Schranken und die darin enthaltenen Gauß-Klammern wird deutlich, daß die exakten Analysewerte der gezeigten Konstruktionen von zahlentheoretischen Eigenschaften der Modellparameter abhängen.

9.2 Algorithmen und obere Schranken

Für das DAP mit $c = 2$ wurden in [BRS99] mehrere einfache Klassen von Online-Algorithmen untersucht. Da der Autor die Urheberschaft dieser Forschungsergebnisse nicht eindeutig von denen der Koautoren des Artikels [BRS99] trennen kann, soll in diesem Kapitel auf Beweise verzichtet werden. Es wird lediglich das beste Resultat der genannten Publikation vorgestellt, und der interessierte Leser sei auf die Originalquelle verwiesen.
Neben diesem Ergebnis lassen sich aus dem Aufbau der anderen Online-Algorithmen und den erzielten Analyseresultaten ein weiteres Mal Rückschlüsse auf das benötigte Algorithmendesign ziehen. Diese Erkenntnisse werden ebenfalls im folgenden dargestellt.

Die beste Online-Strategie in [BRS99] ist nicht eindeutig in ihren Entscheidungen spezifiziert und definiert deshalb eine ganze Klasse von Algorithmen. Diese wird am angegebenen Ort $\mathsf{A}_{\text{balance}}$ genannt und führt Berechnungen ähnlich dem Algorithmus LMM mit den in Kapitel 8.3 beschriebenen Modifikationen aus. Von $\mathsf{A}_{\text{balance}}$ wird verlangt, daß bei einer Entscheidung über die Nutzung der m aktuellen Ressourcen $\{s_{j,i} \mid j \in \{1, \ldots, m\}\}$ die folgenden Bedingungen erfüllt werden:

1. Im knoteninduzierten Teilgraphen B_i, welcher aus den aktuellen und zukünftigen Serverknoten $\{s_{j,k} \mid k \geqq i, j \in \{1, \ldots, m\}\}$ und den schon bekannten und noch nicht abschließend behandelten Requestkonten besteht, ist ein Maximum-Kardinalitäts-Matching $\mathcal{M}(B_i)$ zu bestimmen, so daß
2. die Funktion

$$\sum_{j=0}^{d-1} n_{i+j} \cdot (m+1)^{d-j}$$

   maximiert wird. Dabei ist i der aktuelle Zeitschritt und n_k ist die Anzahl gebundener Serverknoten in $\mathcal{M}(B_i)$ mit dem Zeitindex k.
   Aus beweistechnischen Gründen wird weiterhin gefordert:
3. Alle in den vorigen Zeitschritten eingeplanten Anfragen werden bearbeitet, d. h. die in $\mathcal{M}(B_j)$ mit $i - d \leqq j < i$ gebundenen Requestknoten bleiben gebunden und werden in die Online-Lösung aufgenommen.

Die dritte Forderung wird automatisch erfüllt, wenn die Matchings $\mathcal{M}(B_i)$ durch Erweiterungen aus den vorherigen Matchings $\mathcal{M}(B_{i-1})$ erzeugt werden. Dabei bleiben alle eingeplanten Anfragen im zukünftigen Plan, sie können jedoch anderen Ressourcen zugeordnet werden.
Die Funktion in Forderung 2 erfüllt zwei Eigenschaften: Erstens werden die Anfragen zu möglichst frühen Zeitpunkten eingeplant. Dieses impliziert zweitens, daß die Last der Anfragen zwischen den betroffenen Speicherressourcen möglichst gleichmäßig verteilt wird. Es erfolgt eine Art *Balancierung*.

Für $\mathsf{A}_{\text{balance}}$ konnten in [BRS99] die folgenden Schranken für den Competitive Ratio im Modell des DAP mit $c = 2$ gezeigt werden:

- für $d = 2$ die exakte Analyse $\mathcal{R}(\mathsf{A}_{\text{balance}}) = 4/3 = 1.\overline{3}$
- für $d \equiv 2 \bmod 3$ eine untere Schranke von

$$\mathcal{R} \geqq \frac{5d+2}{4d+1}$$

- für $d > 2$ eine obere Schranke von

$$\mathcal{R} \leqq \frac{6(d-1)}{4d-3}$$

Es wurden weiterhin Klassen von Online-Algorithmen untersucht, die auf einfacheren Strategien beruhen. Von einem Online-Algorithmus wird nur der aktuelle Zeitschritt betrachtet und darin eine maximale Anzahl von Anfragen bedient. Somit wird keine Vorausplanung betrieben. Zwei weitere untersuchte Klassen von Online-Algorithmen fixieren die Matchings $\mathcal{M}(B_i)$, also den Plan für die Zukunft. Neue Eingaben in den nächsten Zeitschritten werden ausschließlich den noch nicht verplanten Ressourcen zugeordnet.
Es sind obere Schranken für den Competitive Ratio dieser Online-Algorithmen bestimmt worden. Bei wachsendem Parameter d konvergieren die Werte der Schranken gegen 2. Die nachgewiesenen unteren Schranken konvergieren gegen 1.5, $e/(e-1) \approx 1.582$ bzw. 2.
Daraus läßt sich erkennen, daß diese einfachen Strategien für gute Online-Algorithmen zum Lösen des DAP nicht ausreichen. Das Aufstellen von optimalen Plänen für die Zukunft auf Basis des bisherigen Wissens und das Anpassen und Umstellen dieser Pläne beim Eintreffen neuer Eingaben ist für die Entscheidungsfindung eines Online-Algorithmus notwendig, wenn dieser einen Competitive Ratio unterhalb von 1.5 erreichen soll. Damit bestätigt sich die Notwendigkeit der Berücksichtigung von Konstruktionsregeln für Online-Algorithmen, wie sie in Kapitel 8.4 aufgestellt wurden.

10 Randomisierte Algorithmen — untere Schranken

Übersicht: Da in Auseinandersetzung mit anderen Wissenschaftlern immer wieder die Frage nach dem Vergleich zu randomisierten Algorithmen für die untersuchte Problemklasse auftrat, sollen zu diesem Thema einige prinzipielle Anmerkungen folgen. Es wird dabei festgestellt, daß die Konstruktionen der unteren Schranken für deterministische Online-Algorithmen mittels einer einfachen Transformation in *konstante* untere Schranken für den Competitive Ratio gegen den blinden Gegenspieler — dem schwächsten der drei Gegenspielermodelle — umgeformt werden können.

Die Definition von randomisierten Online-Algorithmen sowie die Abwandlungen in der Technik der Competitive Analysis sind dem Kapitel 2.5.1 zu entnehmen. Dort ist ebenfalls eine Stellungnahme zum Vergleich der Analyseresultate von deterministischen und randomisierten Online-Algorithmen zu finden.
Die folgenden Aussagen beschränken sich auf das schwächste Gegenspielermodell des blinden Gegenspielers. Die Beweisidee zu Satz 3 kann gleichfalls zur Bestimmung einer unteren Schranke für den Competitive Ratio eines randomisierten Online-Algorithmus für das ORSM-Problem genutzt werden. Statt die Eingabe je nach Reaktion des Online-Algorithmus gemäß Fall 1 (siehe Skizze 3, Teil a) oder Fall 2 (siehe Skizze 3, Teil b) fortzusetzen, werden die beiden Eingabesequenzen aller vier Zeitschritte unabhängig gleichverteilt zufällig gewählt. Die Entscheidungen des Online-Algorithmus bleiben dabei unberücksichtigt. Es ist dann zu erwarten, daß der Online-Algorithmus in der Hälfte der Fälle ein Matching der Größe 3 und in der anderen Hälfte der Fälle ein Matching der Größe 2 erzielt. Die optimale Lösung hat jeweils aller vier Zeitschritte drei Matchingkanten im Eingabesequenzteil erreicht, woraus sich

$$\overline{\mathcal{R}}_{\mathsf{BL}} \geqq \frac{3+3}{2+3} = \frac{6}{5} = 1.2$$

ergibt.

In gleicher Weise lassen sich die Beweise der anderen in dieser Arbeit gezeigten, unteren Schranke modifizieren, um erste Ergebnisse zu unteren Schranken im Competitive Ratio randomisierter Online-Algorithmen gegen den blinden Gegenspieler zu erhalten. Sind die in der Gegenspielerstrategie verlangten Entscheidungen binär, d. h. ein Online-Algorithmus hat bei der Entscheidung über die Nutzung eines Serverknotens s zwischen zwei Möglichkeiten zu wählen, und ist der Aufbau der Entscheidungssituation unabhängig von den vorher vom Online-Algorithmus festgelegten Lösungsteile, so folgt aus einer unteren Schranke im Competitive Ratio für deterministische Online-Algorithmen der Form

$$\mathcal{R} \geqq \frac{a}{a-b} \quad , \quad a, b \in \mathbb{N},\ a > b$$

eine untere Schranke im Competitive Ratio für randomisierte Online-Algorithmen gegen den blinden Gegenspieler von

$$\overline{\mathcal{R}}_{\mathsf{BL}} \geqq \frac{2a}{2a-b} \ .$$

Dieser Term ist zwar kleiner, er stellt jedoch immer noch eine Konstante dar.
Auch aus Beweisen unterer Schranken dieser Abhandlung, welche die oben geforderten Voraussetzungen nicht erfüllen, lassen sich mit derselben Technik konstante untere Schranken für den Competitive Ratio randomisierter Online-Algorithmen der entsprechenden Modellvarianten ableiten.

Obwohl ohne die Entwicklung und Analyse randomisierter Online-Algorithmen keine Aussage über die Güte dieser soeben vorgestellten unteren Schranken möglich ist, bleibt festzuhalten, daß sich aus den obigen Überlegungen selbst für das

schwächste Gegenspielermodell *konstante* untere Schranken für den Competitive Ratio ergeben. Damit sind die unter Zuhilfenahme von Zufallsentscheidungen bestimmten Lösungen in ungünstigen Fällen immer noch substantiell schlechter als die entsprechenden optimalen Lösungen. Dieses Faktum stellt zusammen mit den Aussagen auf Seite 29 den Grund für den Verzicht auf das Studium randomisierter Online-Algorithmen in dieser Arbeit dar.

11 Weiterführende Forschungsaufgaben

Innerhalb der vorliegenden Arbeit konnten für einige der betrachteten Modelle unter dem Gesichtspunkt der Competitive Analysis exakte Resultate vorgelegt werden. In anderen Fällen weichen die präsentierten Ergebnisse zu den oberen und unteren Schranken der Competitive Ratios voneinander ab.
Für das wORSM-Modell ist der Online-Algorithmus wLMM genau analysiert worden. Es ist jedoch keine Gegenspielerstrategie bekannt, die einen Competitive Ratio von 2 erzwingt. Das Schließen der Lücke zur besten bekannten unteren Schranke von $\phi \approx 1.618$ bleibt als eine Aufgabe offen. Dabei werden im Kapitel 6.5 Hinweise gegeben, wie ein Online-Algorithmus mit besserer Leistungsgarantie unter Umsetzung der Designregeln aus Kapitel 8.4 aussehen könnte, und es werden die Grenzen der bisherigen Beweismethode aufgezeigt.
Es wäre auch wünschenswert, für das in Kapitel 8 behandelte ORSM-Problem mit konstantem Knotengrad g und begrenzter Kantenlänge d exakte analytische Beweise für obere Schranken des Competitive Ratios zu besitzen. Einerseits kann eine Auseinandersetzung mit diesem Problem zu einer Verbesserung des Verständnisses zum speziellen Algorithmendesign für diese Problemklasse beitragen. Gleiches gilt für das DAP, für das nur einfachste Online-Algorithmen untersucht und erste obere Schranken des Competitive Ratios bestimmt wurden. Andererseits zeigt die Struktur der unteren Schranken dieser beiden Modellklassen eine Abhängigkeit der exakten Werte des Competitive Ratios von zahlentheoretischen Eigenschaften der Modellparameter auf. Deshalb verbleiben Zweifel, ob umfangreiche Untersuchungen zur Bestimmung der exakten Competitive Ratios für sämtliche Modellparameter geeignet sind die Erkenntnisse zu den Strukturen der Modelle und den algorithmischen Aspekten zu erhöhen.

Es ist ebenfalls anzustreben, für das DAP gute analytische Beweise für obere Schranken zu kennen. Dies würde die Konstruktion von stark verbesserten Online-Algorithmen einschließen, die sich gemäß den Designregeln aus Kapitel 8.4 verhalten. Desweiteren ist dazu die Entwicklung neuer Beweistechniken nötig. Bevor jedoch die oberen und unteren Schranken für alle Modellparameterkombinationen eng aneinander gebracht werden, hält der Autor die Beschäftigung mit weiteren Modellvarianten für sinnvoll.
Solche Erweiterungen könnten das schrittweise Entfernen von Einschränkungen wie die Einheitsgröße der Aufträge und die Aufweichung der starren Modellpa-

rameter c und d für jeden Auftrag sein. Stattdessen könnten die Beschreibungen der Aufträge individuelle Festlegungen dieser Parameter beinhalten. Da sich die Struktur der Modelle unter solchen Modifikationen radikal ändert, ist vor einer Untersuchung derartiger Modelle genau zu überprüfen, ob diese Probleme oder sehr ähnliche Formulierungen schon in der Literatur abgehandelt wurden.

Andere interessante Modellvarianten werden durch das Einfügen zusätzlicher Abhängigkeiten gebildet. So kann das Zugriffsproblem auf Speicherressourcen um ein reales Verbindungsnetzwerk ergänzt werden, durch welches die Daten nach dem Auslesen geschickt werden müssen. Die dabei auftretenden Konflikte können das DAP ergänzen. Auch auf diese Weise läßt sich das abstrakte Modell des DAP näher an reale Fragestellungen angleichen.

Resümee

Die Competitive Analysis ermöglicht Untersuchungen von Online-Problemen und -Algorithmen. Dabei müssen die Algorithmen endgültige Entscheidungen über Teile der Lösung treffen, ohne die vollständige Eingabe des zu bearbeitenden Problems zu kennen. Die Competitive Analysis bestimmt dann den schlimmsten Fall des Verhältnisses der optimalen Lösungsqualität zur Lösungsqualität des Online-Algorithmus über alle möglichen Eingaben.

Vor dem Aufstellen und den Untersuchungen der Modelle wird eine Abhandlung und kritische Auseinandersetzung zur Methode der Competitive Analysis gegeben.
Diesen Ausführungen schließt sich eine kurze Abhandlung vorangegangener Forschungen, ihrer Resultate und eine Abgrenzung zu den in dieser Arbeit behandelten Modellen an.

Die neu eingeführten Modelle entstanden beim Studium von Online-Planungsproblemen mit individuellen Auftragsfristen. Es wurde eine Beschränkung auf Einheitsaufträge eingeführt, woraus sich Zuordnungsprobleme von Aufträgen zu Ressourcen ergeben. Diese können mit Hilfe von bipartiten Graphen modelliert werden, und die Lösung des Zuordnungsproblems entspricht der Bestimmung eines Matchings. Der Online-Charakter der Planungsprobleme impliziert eine neuartige Variante des Online-Matching-Problems in bipartiten Graphen.
Bei der Aufgabenstellung wird eine bisher selten beachtete Zielfunktion benutzt: Es soll die Anzahl der erfolgreich bearbeiteten Aufträge maximiert werden. Daraus folgt sofort, daß Aufträge, deren Fristen nicht eingehalten werden können, später gar nicht mehr bearbeitet werden. Eine Motivation dieses Zielfunktionstyps ist durch die Bereitstellung und Wiedergabe von kontinuierlichen Medien-Datenströmen (Video, Audio) gegeben.

Die ersten Untersuchungen beschränken sich auf ein sehr abstraktes Basismodell namens Online-Request-Server-Matching-Problem (ORSM-Problem). Dieses Modell wurde exakt analysiert und der Online-Algorithmus LMM (Local Maximum Matching) als 1.5-competitive nachgewiesen.
Eine Erweiterung dieses Online-Matching-Problems mit Gewichten an den Kanten des bipartiten Graphen (wORSM-Problem) wurde ebenfalls betrachtet. Das Ziel ist hierbei die Maximierung des Gesamtgewichtes der Kanten des Online-Matchings. Für dieses Problem gibt es eine allgemeingültige untere Schranke des

Competitive Ratios vom Wert des Goldenen Schnittes $\phi = (\sqrt{5}+1)/2 \approx 1.618$. Mit einigem Aufwand konnte der Online-Algorithmus wLMM für dieses Problem als 2-competitive bestimmt werden. Dieser Beweis einer oberen Schranke für den Competitive Ratio des wORSM-Problems benötigt eine Charakterisierung von erweiternden Wegen in bipartiten Graphen, die sich unter einer festgelegten Menge von Operationen dynamisch verändern. Es wurde ebenfalls gezeigt, auf welchem Wege eine Verbesserung des Online-Algorithmus möglich ist und warum die entwickelte Beweistechnik für derartige Modifikationen nicht mehr wirksam benutzt werden kann.

Das Data-Access-Problem (DAP) besitzt weitere Restriktionen und Abhängigkeiten zwischen den Aufträgen. Damit ist es in der Lage, ein Zugriffsproblem auf verteilte Speicherressourcen in einem Server für kontinuierliche Datenströme abzubilden. Es abstrahiert ebenfalls stark, da ausschließlich die Aufträge nach Lieferung von Datenpaketen und die Speicherressourcen berücksichtigt werden.
Es wurden mehrere Modellzwischenstufen studiert. Diese Modellvarianten entstehen aus dem ORSM-Problem durch Hinzufügen von Konzepten, die ebenfalls im DAP zu finden sind. All diese Modellvarianten und das DAP sind bei genauer Betrachtung Spezialfälle des ORSM-Problems. Deshalb überträgt sich die obere Schranke des Competitive Ratios von 1.5, die vom Algorithmus LMM erreicht wird.
Für einige der Modellvarianten konnten untere Schranken von 1.5 nachgewiesen werden, womit die Analysen vollständig und exakt sind. Bei zwei Modellen, die beide eine Vorschau in der Eingabe sowie eine Restriktion in den individuellen Auftragsfristen besitzen, sind exakte Ergebnisse von unter 1.5 für den Competitive Ratio gezeigt worden.

Es wurde ein System verschiedener Konstruktionen benutzt, um untere Schranken für den Competitive Ratio des ORSM-Problems mit konstantem Knotengrad g und begrenzter Kantenlänge d nachzuweisen. Die in dieser Modellvariante auftretenden Abhängigkeiten entzogen sich analytischen Beweisen für obere Schranken. Deshalb ist für kleine Modellparameter der Competitive Ratio unter Zuhilfenahme eines Computerprogrammes getestet worden. Auf trickreiche Art wurden alle Eingaben überprüft und so entstanden Computerbeweise. Diese, auf Vorarbeiten in der Literatur beruhenden Ideen sowie die Überlegungen zur Steigerung der algorithmischen Effizienz der Implementierung nehmen einen großen Teil des entsprechenden Kapitels ein. Für einige Parameterkombinationen konnten so, z. T. exakte, Competitive Ratios mit Werten deutlich unter 1.5 aufgezeigt werden. Diese Resultate erfordern eine starke Weiterentwicklung der eingesetzten Online-Algorithmen, woraus sich Erkenntnisse über die Algorithmenkonstruktion für diese Problemklasse gewinnen ließen.
Ähnliche Erfahrungen wurden bei der Untersuchung des DAP gemacht, für welches ebenfalls konstante untere Schranken bestimmt wurden. Deren Werte zeigen leichte Abhängigkeiten von zahlentheoretischen Eigenschaften der Modellparameter auf. Bei ausreichender Anzahl von Ressourcen im Modell liegen die allermeisten dieser unteren Schranken bei einem Wert von mindestens $12/11 = 1.\overline{09}$ und

häufig wird ein Wert von 1.1 nur knapp verfehlt. Es war ein anfangs überraschendes Resultat in diesem Modell für alle Kombinationen der Modellparameter c und d *konstante* untere Schranken für den Competitive Ratio zu finden, die sich deutlich von 1 abheben.

Aus der Beschäftigung mit all diesen Modellen ergeben sich Anforderungen für die Konstruktion guter Online-Algorithmen für die behandelte Klasse von Planungsproblemen[43]. Diese Designregeln sind das Bestimmen von Entscheidungen mit:

1. lokaler Optimalität — im aktuellen Zeitschritt wird die Zielfunktion maximal verbessert;
2. globaler Optimalität — es werden Lösungen nach Regel 1 bestimmt, die in den folgenden Zeitschritten die Zielfunktion maximal verbessern, wobei nur die bisher bekannten Eingabeteile berücksichtigt werden können;
3. maximalem Freiheitsgrad — eine Lösung für die aktuelle Entscheidung soll eine maximale Anzahl verschiedener Strukturen von Lösungsvarianten für die nächsten Zeitschritte aufweisen.

Mit der Regel 3 wird eine hohe Flexibilität bei der Reaktion auf die noch nicht bekannten Teile der Eingabesequenz erzielt. Unter Umständen kann bei Modelländerungen (z. B. mit dem Einführen von Gewichten/Prioritäten) ein definiertes und begrenztes Verletzen der Regeln 1 und 2 sinnvoll und notwendig sein, wenn dadurch die in Regel 3 geforderte Flexibilität stark erhöht wird.
Viele erfolgreiche Online-Algorithmen in der Literatur berücksichtigen implizit die oben aufgeführten Konstruktionsregeln. Deshalb plädiert der Autor dafür, diese drei Regeln zum Ausgangspunkt beim Design von Online-Algorithmen zu machen. Für ein konkretes Online-Problem müssen dabei die exakten Formulierungen für Optimalität und Freiheitsgrad problemspezifisch angepaßt werden.

[43]Sie sind charakterisiert durch Einheitsaufträge mit Fristen und der Zielfunktion, die Anzahl erfolgreicher Aufträge zu maximieren.

Literaturverzeichnis

[AADW94] Ajtai, Miklos ; Aspnes, James ; Dwork, Cynthia ; Waarts, Orli: A Theory of Competitive Analysis for Distributed Algorithms. In: *Proceedings of the 35th Annual Symposium on Foundations of Computer Science —FOCS '94 (Santa Fe, New Mexico, November 20–22, 1994).* Los Alamitos-Washington-Brussels-Tokyo : IEEE Computer Society Press, 1994, S. 401–411

[AAF+93] Aspnes, James ; Azar, Yossi ; Fiat, Amos ; Plotkin, Serge ; Waarts, Orli: On-Line Load Balancing with Application to Machine Scheduling and Virtual Circuit Routing. In: *Proceedings of the 25th Annual ACM Symposium on Theory of Computing —STOC '93 (San Diego, California, May 16–18, 1993).* New York : ACM Press, 1993, S. 623–631. – (Überarbeitete Journalveröffentlichung ist [AAF+97].)

[AAF+96] Awerbuch, Baruch ; Azar, Yossi ; Fiat, Amos ; Leonardi, Stefano ; Rosén, Adi: Online Competitive Algorithms for Call Admission in Optical Networks. In: Diaz, J. (Hrsg.) ; Serna, M. (Hrsg.): *Proceedings of the Fourth European Symposium on Algorithms —ESA '96 (Barcelona, Spain, September 25–27, 1996).* Berlin-Heidelberg-New York : Springer-Verlag, 1996 (Lecture Notes in Computer Science 1136), S. 431–444

[AAF+97] Aspnes, James ; Azar, Yossi ; Fiat, Amos ; Plotkin, Serge ; Waarts, Orli: On-Line Routing of Virtual Circuits with Application to Load Balancing and Machine Scheduling. In: *Journal of the ACM* 44 (1997), Mai, Nr. 3, S. 486–504. – (Vorabveröffentlichung ist [AAF+93].)

[AAPW94] Awerbuch, Baruch ; Azar, Yossi ; Plotkin, Serge ; Waarts, Orli: Competitive Routing of Virtual Circuits with Unknown Duration. In: *Proceedings of the Fifth Annual ACM-SIAM Symposium on Discrete Algorithms —SODA '94 (Arlington, Va., January 23–25, 1994).* New York : ACM Press, 1994, S. 321–327

[ABFR94] Awerbuch, Baruch ; Bartal, Yair ; Fiat, Amos ; Rosén, Adi: Competitive Non-Preemtive Call Control. In: *Proceedings of the Fifth Annual ACM-SIAM Symposium on Discrete Algorithms —SODA '94 (Arlington, Va., January 23–25, 1994).* New York : ACM Press, 1994, S. 312–320

[ABK94] Azar, Yossi ; Broder, Andrei Z. ; Karlin, Anna R.: On-Line Load Balancing. In: *Theoretical Computer Science* 130 (1994), Nr. 1, S. 73–84

[ABKU99] Azar, Yossi ; Broder, Andrei Z. ; Karlin, Anna R. ; Upfal, Eli: Balanced Allocations. In: *SIAM Journal on Computing* 29 (1999), Nr. 1, S. 180–200

[ABMP91] Alt, H. ; Blum, N. ; Mehlhorn, K. ; Paul, M.: Computing a Maximum Cardinality Matching in a Bipartite Graph in Time $O(n^{1.5}\sqrt{m/\log n})$. In: *Information Processing Letters* 37 (1991), Februar, S. 237–240

[ACN96] Achlioptas, Dimitris ; Chrobak, Marek ; Noga, John: Competitive Analysis of Randomized Paging Algorithms. In: Diaz, J. (Hrsg.) ; Serna, M. (Hrsg.): *Proceedings of the Fourth European Symposium on Algorithms —ESA '96 (Barcelona, Spain, September 25–27, 1996).* Berlin-Heidelberg-New York : Springer-Verlag, 1996 (Lecture Notes in Computer Science 1136), S. 419–430

[AE97] Azar, Yossi ; Epstein, Leah: On-Line Load Balancing of Temporary Tasks on Identical Machines. In: *Proceedings of the Fifth Annual Israel Symposium on Theory of Computing and Systems —ISTCS '97 (Ramat Gan, Israel, June 17–19, 1997)*, 1997, S. 119–125

[AGH95] Aggarwal, Sudhanshu ; Garay, Juan A. ; Herzberg, Amir: Adaptive Video on Demand. In: Spirakis, Paul G. (Hrsg.): *Proceedings of the Third Annual European Symposium on Algorithms —ESA '95 (Corfu, Greece, September 25–27, 1995).* Berlin-Heidelberg-New York : Springer-Verlag, 1995 (Lecture Notes in Computer Science 979), S. 538–553

[AKP+97] Azar, Yossi ; Kalyanasundaram, Bala ; Plotkin, Serge ; Pruhs, Kirk R. ; Waarts, Orli: Online Load Balancing of Temporary Tasks. In: *Journal of Algorithms* 22 (1997), Januar, Nr. 1, S. 93–110

[Alb93a] Albers, Susanne: A Competitive Analysis of the List Update Problem with Lookahead. In: *Proceedings of the 19th International Symposium on Mathematical Foundations of Computer Science —MFCS '93 (Košice, Slovakia, August 22–26, 1993).* Berlin-Heidelberg-New York : Springer-Verlag, 1993 (Lecture Notes in Computer Science 841), S. 201–210. – (Journalveröffentlichung ist [Alb98a].)

[Alb93b] Albers, Susanne: The Influence of Lookahead in Competitive Paging Algorithms (Extended Abstract). In: Lengauer, T. (Hrsg.): *Proceedings of the First Annual European Symposium on Algorithms —ESA '93 (Bad Honnef, Germany, September 30 – October 2, 1993).* Berlin-Heidelberg-New York : Springer-Verlag, 1993 (Lecture Notes in Computer Science 726), S. 1–12. – (Journalveröffentlichung ist [Alb97].)

[Alb95] Albers, Susanne: Improved Randomized On-Line Algorithms for the List Update Problem. In: *Proceedings of the Sixth Annual ACM-SIAM Symposium on Discrete Algorithms —SODA '95 (San Francisco, California, January 22–24, 1995).* New York : ACM Press, 1995, S. 412–419. – (Journalveröffentlichung ist [Alb98b].)

[Alb97] Albers, Susanne: On the Influence of Lookahead in Competitive Paging Algorithms. In: *Algorithmica* 18 (1997), Juli, Nr. 3, S. 283–305. – (Konferenzveröffentlichung ist [Alb93b].)

[Alb98a] ALBERS, Susanne: A Competitive Analysis of the List Update Problem with Lookahead. In: *Theoretical Computer Science* 197 (1998), Nr. 1-2, S. 95–109. – (Konferenzveröffentlichung ist [Alb93a].)

[Alb98b] ALBERS, Susanne: Improved Randomized On-Line Algorithms for the List Update Problem. In: *SIAM Journal on Computing* 27 (1998), Juni, Nr. 3, S. 682–693. – (Konferenzveröffentlichung ist [Alb95].)

[Alb00] ALBERS, Susanne: Better Bounds for Online Scheduling. In: *SIAM Journal on Computing* 29 (2000), April, Nr. 2, S. 459–473

[Amb00] AMBÜHL, Christoph: Offline List Update is NP-hard. In: PATERSON, M. (Hrsg.): *Proceedings of the Eighth European Symposium on Algorithms —ESA '00 (Saarbrücken, Germany, September 5–8, 2000)*. Berlin-Heidelberg-New York : Springer-Verlag, 2000 (Lecture Notes in Computer Science 1879), S. 42–51

[AMO93] AHUJA, Ravindra K. ; MAGNANTI, Thomas L. ; ORLIN, James B.: *Network Flows: Theory, Algorithms, and Applications*. New Jersey : Prentice Hall, 1993

[ANR95] AZAR, Yossi ; NAOR, Joseph ; ROM, Raphael: The Competitiveness of On-Line Assignments. In: *Journal of Algorithms* 18 (1995), S. 221–237

[ASW95] ALBERS, Susanne ; VON STENGEL, Bernhard ; WERCHNER, Ralph: A Combined BIT and TIMESTAMP Algorithm for the List Update Problem. In: *Information Processing Letters* 56 (1995), S. 135–139

[BBS99] BERENBRINK, Petra ; BRINKMANN, Andre ; SCHEIDELER, Christian: Design of the PRESTO Multimedia Data Storage Network (Extended Abstract). In: *Proceedings of the International Workshop on Communication and Data Management in Large Networks —CDMLarge '99*, 1999

[BCK00] BERMAN, Piotr ; CHARIKAR, Moses ; KARPINSKI, Marek: On-Line Load Balancing for Related Machines. In: *Journal of Algorithms* 35 (2000), April, Nr. 1, S. 108–121

[BDB94] BEN-DAVID, Shai ; BORODIN, Allan: A New Measure for the Study of On-Line Algorithms. In: *Algorithmica* 11 (1994), Nr. 1, S. 73–91

[BDBK+94] BEN-DAVID, Shai ; BORODIN, Allan ; KARP, Richard M. ; TARDOS, Gábor ; WIGDERSON, Avi: On the Power of Randomization in On-Line Algorithms. In: *Algorithmica* 11 (1994), Nr. 1, S. 2–14

[Bei66] Kapitel XVI In: BEILER, Albert H.: *Recreations in the Theory of Numbers – The Queen of Mathematics Entertains*. 2. New York : Dover Publications, INC., 1966, S. 168–172

[Bel66] BELADY, L. A.: A Study of Replacement Algorithms for Virtual Storage Computers. In: *IBM Systems Journal* 5 (1966), S. 78–101

[Ber57] Berge, Claude: Two Theorems in Graph Theory. In: *Proceedings of the National Academy of Science of the United States of America* 43 (1957), S. 842–844

[BESW94] Błażewicz, Jacek ; Ecker, Klaus ; Schmidt, Günter ; Węglarz, Jan: *Scheduling in Computer ans Manufacturing Systems.* 2. Berlin-Heidelberg-New York : Springer-Verlag, 1994

[BEY98] Borodin, Allan ; El-Yaniv, Ran: *Online Computation and Competitive Analysis.* Cambridge University Press, 1998

[BFKV95] Bartal, Yair ; Fiat, Amos ; Karloff, Howard ; Vohra, Rakesh: New Algorithms for an Ancient Scheduling Problem. In: *Journal of Computer and System Sciences* 51 (1995), Nr. 3, S. 359–366

[BIRS91] Borodin, Allan ; Irani, Sandy ; Raghavan, Prabhakar ; Schieber, Baruch: Competitive Paging with Locality of Reference (Preliminary Version). In: *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing —STOC '91 (New Orleans, Louisiana, May 6–8, 1991).* New York : ACM Press, 1991, S. 249–259. – (Journalveröffentlichung ist [BIRS95].)

[BIRS95] Borodin, Allan ; Irani, Sandy ; Raghavan, Prabhakar ; Schieber, Baruch: Competitive Paging with Locality of Reference. In: *Journal of Computer and System Sciences* 50 (1995), Nr. 2, S. 244–258. – (Konferenzveröffentlichung ist [BIRS91].)

[BJK97] Brucker, Peter ; Jurisch, Bernd ; Krämer, Andreas: Complexity of Scheduling Problems with Multi-Purpose Machines. In: *Annals of Operations Research* 70 (1997), S. 57–73

[BKM+91a] Baruah, Sanjoy. ; Koren, Gilad ; Mishra, B. ; Raghunathan, A. ; Rosier, Louis ; Shasha, Dennis: On-Line Scheduling in the Presence of Overload. In: *Proceedings of the 32nd Annual Symposium on Foundations of Computer Science —FOCS '91 (San Juan, Puerto Rico, October 1–4, 1991).* Los Alamitos-Washington-Brussels-Tokyo : IEEE Computer Society Press, 1991, S. 100–110

[BKM+91b] Baruah, Sanjoy ; Koren, Gilad ; Mao, Decao ; Mishra, B. ; Raghunathan, A. ; Rosier, Louis ; Shasha, Dennis ; Wang, Fuxing: On the Competitiveness of On-Line Real-Time Task Scheduling. In: *Proceedings of the IEEE Real-Time Systems Symposium, 1991* IEEE, 1991, S. 106–115. – (Journalveröffentlichung ist [BKM+92].)

[BKM+92] Baruah, Sanjoy ; Koren, Gilad ; Mao, Decao ; Mishra, B. ; Raghunathan, A. ; Rosier, Louis ; Shasha, Dennis ; Wang, Fuxing: On the Competitiveness of On-Line Real-Time Task Scheduling. In: *Journal of Real-Time Systems* 4 (1992), S. 124–144. – (Konferenzveröffentlichung ist [BKM+91b].)

[BKPS96] Bouras, Christos ; Kapoulas, Vaggelis ; Pantziou, Grammati E. ; Spirakis, Paul G.: Competitive Scheduling Schemes for Video On Demand / Computer Technology Institute. Patras, Greece, 1996 (CTI-TR 96.5.12). – Forschungsbericht

[BL97] Bartal, Yair ; Leonardi, Stefano: On-Line Routing in All-Optical Networks. In: Degano, P. (Hrsg.) ; Gorrieri, R. (Hrsg.) ; Marchetti-Spaccamela, A. (Hrsg.): *Proceedings of the 24th International Colloquium on Automata, Languages and Programming —ICALP '97 (Bologna, Italy, July 7–11, 1997).* Berlin-Heidelberg-New York : Springer-Verlag, 1997 (Lecture Notes in Computer Science 1256), S. 516–526

[BL99] Boyar, Joan ; Larsen, Kim S.: The Seat Reservation Problem. In: *Algorithmica* 25 (1999), Nr. 4, S. 403–417

[BLMS+96] Bartal, Yair ; Leonardi, Stefano ; Marchetti-Spaccamela, Alberto ; Sgall, Jiří ; Stougie, Leen: Multiprocessor Scheduling with Rejection. In: *Proceedings of the Seventh Annual ACM-SIAM Symposium on Discrete Algorithms —SODA '96 (Atlanta, Georgia, January 28–30, 1996).* New York : ACM Press, 1996, S. 95–103

[BLMS+00] Bartal, Yair ; Leonardi, Stefano ; Marchetti-Spaccamela, Alberto ; Sgall, Jiří ; Stougie, Leen: Multiprocessor Scheduling with Rejection. In: *SIAM Journal on Discrete Mathematics* 13 (2000), Februar, Nr. 1, S. 64–78. – (Konferenzveröffentlichung ist [BLMS+96].)

[BLN98] Boyar, Joan ; Larsen, Kim S. ; Nielsen, Morton N.: The Accommodating Function – A Generalization of the Competitive Ratio / IMADA - the Department of Mathematics and Computer Science at University of Southern Denmark, Odense, Denmark. 1998 (PP-1998-24). – IMADA preprint

[BLN99a] Boyar, Joan ; Larsen, Kim S. ; Nielsen, Morton N.: The Accommodating Function – A Generalization of the Competitive Ratio. In: Dehne, F. (Hrsg.) ; Gupta, A. (Hrsg.) ; Sack, J.-R. (Hrsg.) ; Tamassia, R. (Hrsg.): *Proceedings of the Sixth International Workshop on Algorithms and Data Structures —WADS '99 (Vancouver, Canada, August 11–14, 1999).* Berlin-Heidelberg-New York : Springer-Verlag, 1999 (Lecture Notes in Computer Science 1663), S. 74–79. – (Ausführliche Veröffentlichung ist [BLN98].)

[BLN99b] Boyar, Joan ; Larsen, Kim S. ; Nielsen, Morton N.: Separating the Accommodating Ratio from the Competitive Ratio / IMADA - the Department of Mathematics and Computer Science at University of Southern Denmark, Odense, Denmark. 1999 (PP-1999-03). – IMADA preprint

[BLRK83] Błażewicz, Jacek ; Lenstra, J. K. ; Rinnooy Kan, A. H. G.: Scheduling Subject to Resource Constraints: Classification and Complexity. In: *Discrete Applied Mathematics* 5 (1983), S. 11–24

[BLS87] Borodin, Allan ; Linial, Nathan ; Saks, Michael E.: An Optimal On-Line Algorithm for Metrical Task System. In: *Proceedings of the 19th Annual ACM Symposium on Theory of Computing —STOC '87 (New York City, May 25–27, 1987)*. New York : ACM Press, 1987, S. 373–382. – (Journalveröffentlichung ist [BLS92].)

[BLS92] Borodin, Allan ; Linial, Nathan ; Saks, Michael E.: An Optimal On-Line Algorithm for Metrical Task System. In: *Journal of the ACM* 39 (1992), Oktober, Nr. 4, S. 745–763. – (Konferenzveröffentlichung ist [BLS87].)

[BM85] Bentley, Jon L. ; McGeoch, Catherine C.: Amortized Analyses of Self-Organizing Sequential Search Heuristics. In: *Communications of the ACM* 28 (1985), April, Nr. 4, S. 404–411

[BMRC] verschiedene Aufsätze und Berichte seit ca. 1991, sowie aktuelle Informationen der von Prof. Larry Rowe geleiteten Berkeley Plateau Multimedia Research Group, Computer Science Division – EECS, University of California, Berkeley; Teil des Berkeley Multimedia Research Center. `http://www-plateau.cs.berkeley.edu/`

[BN99] Boyar, Joan ; Nielsen, Morton N.: An Improved Lower Bound on First-Fit's Accommodating Ratio for the Unit Price Bin Packing Problem / IMADA - the Department of Mathematics and Computer Science at University of Southern Denmark, Odense, Denmark. 1999 (PP-1999-11). – IMADA preprint

[BNGNS99] Bar-Noy, Amotz ; Guha, Sudipto ; Naor, Joseph ; Schieber, Baruch: Approximating the Throughput of Multiple Machines under Real-time Scheduling (Extended Abstract). In: *Proceedings of the 31st Annual ACM Symposium on Theory of Computing —STOC '99 (Atlanta, Georgia, May 1–4, 1999)*. New York : ACM Press, 1999

[BR93] Bernstein, Ethan ; Rajagopalan, Sridhar: The Roommates Problem: Online Matching on General Graphs / University of California, Berkeley. CS Department, 1993 (CSD-93-757). – Forschungsbericht

[Bre98] Breslauer, D.: On Competitive On-Line Paging with Lookahead. In: *Theoretical Computer Science* 209 (1998), Dezember, Nr. 1–2, S. 365–375

[Bro60] Brocot, Achille: Calcul des rouages par approximation, nouvelle méthode. In: *Revue Chronométrique* 6 (1860), S. 186–194

[BRS99] Berenbrink, Petra ; Riedel, Marco ; Scheideler, Christian: Simple Competitive Request Scheduling Strategies. In: *Proceedings of the Eleventh ACM Symposium on Parallel Algorithms and Architectures, (Saint-Malo, France, June 27-30, 1999)*. New York : ACM Press, 1999, S. 33–42

[Bru95] Brucker, Peter: *Scheduling Algorithms*. Berlin-Heidelberg-New York : Springer Verlag, 1995

[Buck94] BUCK, Ron: The Oracle Media Server for nCUBE Massively Parallel Systems. In: *Proceedings of the 8th International Parallel Processing Symposium (Cancún, Mexico, April 1994).* Los Alamitos-Washington-Brussels-Tokyo : IEEE Computer Society Press, 1994, S. 670–673

[CG96] Kapitel 6 In: CONWAY, John H. ; GUY, Richard K.: *The Book of Numbers.* New York : Copernicus, Springer-Verlag, 1996, S. 152–156

[CKPV91] CHROBAK, Marek ; KARLOFF, Howard ; PAYNE, T. ; VISHWANATHAN, Sundar: New Results on Server Problems. In: *SIAM Journal on Discrete Mathematics* 4 (1991), Mai, Nr. 2, S. 172–181

[CL92] CHROBAK, Marek ; LARMORE, Lawrence L.: The Server Problem and On-Line Games. In: MCGEOCH, L. A. (Hrsg.) ; SLEATOR, D. D. (Hrsg.): *Proceedings of a DIMACS Workshop on On-Line Algorithms (Rutgers University, February 11–13, 1991)* Bd. 7 American Mathematical Society, 1992, S. 11–64

[CLR90] CORMEN, Thomas H. ; LEISERSON, Charles E. ; RIVEST, Ronald L.: *Introduction to Algorithms.* New York [u.a.] : McGraw Hill, 1990 (The MIT electrical engineering and computer science series). – 21. Nachdruck 1998

[CMM67] CONWAY, Richard W. ; MAXWELL, William L. ; MILLER, Louis W.: *Theory of Scheduling.* Reading, Mass. : Addison-Wesley, 1967

[CN99] CHROBAK, Marek ; NOGA, John: LRU is Better than FIFO. In: *Algorithmica* 23 (1999), Nr. 2, S. 180–185

[Cof76] COFFMAN, E. G. (Hrsg.): *Computer and Job-Shop Scheduling Theory.* New York-London-Sidney-Toronto : John Wiley & Sons, 1976

[CPS+96] CHAKRABARTI, Soumen ; PHILLIPS, Cynthia A. ; SCHULZ, Andreas S. ; SHMOYS, David B. ; STEIN, Clifford ; WEIN, Joel: Improved Scheduling Algorithms for Minsum Criteria. In: MEYER AUF DER HEIDE, F. (Hrsg.) ; MONIEN, B. (Hrsg.): *Proceedings of the 23th International Colloquium on Automata, Languages and Programming —ICALP '96 (Paderborn, Germany, July 8–12, 1996).* Berlin-Heidelberg-New York : Springer-Verlag, 1996 (Lecture Notes in Computer Science 1099), S. 646–657

[CSZ95] COLWELL, Rege ; SUCHOZA, Richard ; ZORINE, Dmitr: On-Line Real-Time Scheduling with Laxities / Deptartment of Computer Science, University of Pittsburgh. 1995. – Manuscript

[CV97] CHEN, Bo ; VESTJENS, Arjen P.: Scheduling on Identical Machines: How Good is LPT in an On-line Setting? In: *Operations Research Letters* 21 (1997), Nr. 4, S. 165–169

[CVW94] CHEN, Bo ; VAN VLIET, André ; WOEGINGER, Gerhard J.: New Lower and Upper Bounds for On-Line Scheduling. In: *Operations Research Letters* 16 (1994), S. 221–230

[CZ96] Chang, E. ; Zakhor, A.: Cost Analyses for VBR Video Servers. In: *Proceedings of the IS&T/SPIE International Symposium on Electronic Imaging: Science and Technology (San Jose, California, January 29–31, 1996)* Bd. 2667, 1996

[DDM+95] Dan, Asit ; Dias, Daniel M. ; Mukherjee, Rajat ; Sitaram, Dinkar ; Tewari, Renu: Buffering and Caching in Large-Scale Video Servers. In: *Technologies for the Information Superhighway, Proceedings of COMPCON 1995.* Los Alamitos, CA : IEEE Computer Society Press, 1995, S. 217–224

[Dic71] Kapitel V In: Dickson, Leonard E.: *History of the Theory of Numbers – Volume I (of 3): Divisibility and Primality.* Bd. 1. New York, NY : Chelsea Publishing Company, 1971, S. 155–158. – textual unaltered reprint of a work first published in 1919, (1920, and 1923) by the Carnegie Institute of Washington as publication number 256 of the Carnegie Institute of Washington

[DM91] Deng, Xiaotie ; Mahajan, Sanjeev: Infinite Games: Randomization, Computability, and Applications to Online Problems (Preliminary Version). In: *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing —STOC '91 (New Orleans, Louisiana, May 6–8, 1991).* New York : ACM Press, 1991, S. 289–298. – (Überarbeitete Journalveröffentlichung ist [DM97].)

[DM97] Deng, Xiaotie ; Mahajan, Sanjeev: The Cost of Derandomization: Computability or Competitiveness. In: *SIAM Journal on Computing* 26 (1997), Juni, Nr. 3, S. 786–802. – (Konferenzveröffentlichung ist [DM91].)

[Edm65a] Edmonds, Jack: Maximum Matchings and a Polyhedron with 0,1-Vertices. In: *Journal of Research of the National Bureau of Standards (Section B)* 69B (1965), Nr. 1–2, S. 125–130

[Edm65b] Edmonds, Jack: Paths, Trees, and Flowers. In: *Canadian Journal on Mathematics* 7 (1965), S. 449–467

[EK70] Edmonds, Jack ; Karp, Richard M.: Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems. In: Guy, R. K. (Hrsg.) ; Hanani, H. (Hrsg.) ; Sauer, N. (Hrsg.) ; Schonheim, J. (Hrsg.): *Proceedings of the Calgary International Conference on Combinatorial Structures and Their Applications, (Calgary, Alberta, Canada, June, 1969).* New York-London-Paris : Gordon and Breach, 1970, S. 93–96. – (Journalveröffentlichung ist [EK72].)

[EK72] Edmonds, Jack ; Karp, Richard M.: Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems. In: *Journal of the ACM* 19 (1972), April, Nr. 2, S. 248–264. – (Konferenzveröffentlichung ist [EK70].)

[Far16] Farey, John: On a curious Property of vulgar Fractions. In: *The Philosophical magazine and journal* 47 (1816), Mai, Nr. 217, S. 385–386

[FK95] FIAT, Amos ; KARLIN, Anna R.: Randomized and Multipointer Paging with Locality of Reference. In: *Proceedings of the 27th Annual ACM Symposium on Theory of Computing —STOC '95 (Las Vegas, Nevada, May 29 – June 1, 1995).* New York : ACM Press, 1995, S. 626–634

[FKL+91] FIAT, Amos ; KARP, Richard M. ; LUBY, Michael ; MCGEOCH, Lyle A. ; SLEATOR, Daniel D. ; YOUNG, Neal E.: Competitive Paging Algorithms. In: *Journal of Algorithms* 12 (1991), S. 685–699

[FM97] FIAT, Amos ; MENDEL, Manor: Truly Online Paging with Locality of Reference (Extended Abstract). In: *Proceedings of the 38th Annual Symposium on Foundations of Computer Science —FOCS '97 (Miami Beach, Florida, October 20–22, 1997).* Los Alamitos-Washington-Brussels-Tokyo : IEEE Computer Society Press, 1997, S. 326–335

[FN95] FAIGLE, Ulrich ; NAWIJN, Willem M.: Note on Scheduling Intervals On-Line. In: *Discrete Applied Mathematics* 58 (1995), S. 13–17

[FR97] FIAT, Amos ; ROSEN, Ziv: Experimental Studies of Access Graph Based Heuristics: Beating the LRU Standard? In: *Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms —SODA '97 (New Orleans, Louisiana, January 5–7, 1997).* New York : ACM Press, 1997, S. 63–72

[FT84] FREDMAN, Michael L. ; TARJAN, Robert E.: Fibonacci Heaps and Their Uses in Improved Network Optimization Algorithm. In: *Proceedings of the 25th Annual Symposium on Foundations of Computer Science — FOCS '84 (Singer Island, Florida, October 24–26, 1984).* Los Alamitos-Washington-Brussels-Tokyo : IEEE Computer Society Press, 1984, S. 338–346. – (Journalveröffentlichung ist [FT87].)

[FT87] FREDMAN, Michael L. ; TARJAN, Robert E.: Fibonacci Heaps and Their Uses in Improved Network Optimization Algorithm. In: *Journal of the ACM* 34 (1987), Juli, Nr. 3, S. 596–615. – (Konferenzveröffentlichung ist [FT84].)

[FW98] FIAT, A. (Hrsg.) ; WOEGINGER, G. J. (Hrsg.): *Online Algorithms: The State of the Art.* Berlin-Heidelberg-New York : Springer-Verlag, 1998 (Lecture Notes in Computer Science 1442)

[Gab90] GABOW, Harold N.: Data Structures for Weighted Matching and Nearest Common Ancestors with Linking. In: *Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms —SODA '90 (San Francisco, Ca., January 22–24, 1990).* New York : ACM Press, 1990, S. 434–443

[GGK+97] GARAY, Juan A. ; GOPAL, Inder S. ; KUTTEN, Shay ; MANSOUR, Yishay ; YUNG, Moti: Efficient On-Line Call-Control Algorithms. In: *Journal of Algorithms* 23 (1997), S. 180–194

[GI89] Gusfield, Dan ; Irving, Robert W.: *The Stable Marriage Problem: Structure and Algorithms.* Cambridge, MA and London : The MIT Press, 1989 (Foundations of Computing)

[GJ79] Garey, Michael R. ; Johnson, David S.: *Computers and Intractability: A Guide to the Theory of NP-Completeness.* San Francisco : W. H. Freemann and Company, 1979

[GKKV95] Grove, Edward F. ; Kao, Ming-Yang ; Krishnan, P. ; Vitter, Jeffrey S.: Online Perfect Matching and Mobile Computing. In: Akl, S. G. (Hrsg.) ; Dehne, F. (Hrsg.) ; Sack, J.-R. (Hrsg.) ; Santoro, N. (Hrsg.): *Proceedings of the Fourth International Workshop on Algorithms and Data Structures —WADS '95 (Kingston, Canada, August 16–18, 1995).* Berlin-Heidelberg-New York : Springer-Verlag, 1995 (Lecture Notes in Computer Science 955), S. 194–205

[GKP94] Graham, Ronald L. ; Knuth, Donald E. ; Patashnik, Oren: *Concrete Mathematics.* 2. Reading, Mass. : Addison-Wesley, 1994

[Gla79] Glaisher, J. W. L.: On a Property of Vulgar Fractions. In: *London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science, Series 5* 7 (1879), Mai, Nr. 44, S. 321–336

[GLLRK79] Graham, R. L. ; Lawler, E. L. ; Lenstra, J. K. ; Rinnooy Kan, A. H. G.: Optimization and Approximation in Deterministic Sequencing and Scheduling: A Survey. In: *Annals of Discrete Mathematics* 5 (1979), S. 287–326

[Goe97] Goemans, Michel X.: Improved Approximation Algorithms for Scheduling with Release Dates. In: *Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms —SODA '97 (New Orleans, Louisiana, January 5–7, 1997).* New York : ACM Press, 1997, S. 591–598

[GR93] Goldberg, Andrew V. ; Radzik, Tomasz: A Heuristic Improvement of the Bellman-Ford Algorithm. In: *Applied Mathematics Letters* 6 (1993), Mai, Nr. 3, S. 3–6

[Gra66] Graham, R. L.: Bounds for Certain Multiprocessor Anomalies. In: *Bell System Technical Journal* 45 (1966), S. 1563–1581

[Gra69] Graham, R. L.: Bounds on Multiprocessing Timing Anomalies. In: *SIAM Journal of Applied Mathematics* 17 (1969), S. 416–429

[GS62] Gale, David ; Shapley, Lloyd S.: College Admissions and the Stability of Marriage. In: *The American Mathematical Monthly* 69 (1962), Januar, S. 9–15

[GT89] Gabow, Harold N. ; Tarjan, Robert E.: Faster Scaling Algorithms for Network Problems. In: *SIAM Journal on Computing* 18 (1989), Oktober, Nr. 5, S. 1013–1036

[GT91] GABOW, Harold N. ; TARJAN, Robert E.: Faster Scaling Algorithms for General Graph-Matching Problems. In: *Journal of the ACM* 38 (1991), Oktober, Nr. 4, S. 815–853

[GW93] GALAMBOS, Gábor ; WOEGINGER, Gerhard J.: An On-Line Scheduling Heuristic with Better Worst Case Ratio than Graham's List Scheduling. In: *SIAM Journal on Computing* 22 (1993), Nr. 2, S. 349–355

[Hal35] HALL, P.: On Representatives of Subsets. In: *The Journal of The London Mathematical Society* 10 (1935), S. 26–30

[Hay88] HAYES, John P.: *Computer Architecture and Organization*. 2. New York : McGraw-Hill, 1988 (McGraw-Hill Series in Computer Organization and Architecture)

[HK73] HOPCROFT, John E. ; KARP, Richard M.: An $n^{5/2}$ Algorithm for Maximum Matchings in Bipartite Graphs. In: *SIAM Journal on Computing* 2 (1973), Dezember, Nr. 4, S. 225–231

[HL92] HONG, Kwang S. ; LEUNG, Joseph Y-T.: On-Line Scheduling of Real-Time Tasks. In: *IEEE Transactions on Computers* 41 (1992), Oktober, Nr. 10, S. 1326–1331

[Hoc96] HOCHBAUM, D. S. (Hrsg.): *Approximation Algorithms for NP-hard Problems*. Boston, Mass. : PWS Publishing Company, 1996

[HS87] HOCHBAUM, Dorit S. ; SHMOYS, David B.: Using Dual Approximation Algorithms for Scheduling Problems: Theoretical and Practical Results. In: *Journal of the ACM* 34 (1987), Januar, Nr. 1, S. 144–162

[HS88] HOCHBAUM, Dorit S. ; SHMOYS, David B.: A Polynomial Approximation Scheme for Scheduling on Uniform Procesors: Using the Dual Approximation Approach. In: *SIAM Journal on Computing* 17 (1988), Juni, Nr. 3, S. 539–551

[HS94] HALLDÓRSSON, Magnús M. ; SZEGEDY, Márió: Lower Bounds for On-Line Graph Coloring. In: *Theoretical Computer Science* 130 (1994), Nr. 1, S. 163–174

[HSSW97] HALL, Leslie A. ; SCHULZ, Andreas S. ; SHMOYS, David B. ; WEIN, Joel: Scheduling to Minimize Average Completion Time: Off-line and On-Line Approximation Algorithms. In: *Mathematics of Operations Research* 22 (1997), August, Nr. 3, S. 513–544

[HSW96] HALL, Leslie A. ; SHMOYS, David B. ; WEIN, Joel: Scheduling to Minimize Average Completion Time: Off-line and On-Line Algorithms. In: *Proceedings of the Seventh Annual ACM-SIAM Symposium on Discrete Algorithms —SODA '96 (Atlanta, Georgia, January 28–30, 1996)*. New York : ACM Press, 1996, S. 142–151

[HU90] HOPCROFT, John E. ; ULLMAN, Jeffrey D.: *Einführung in die Automatentheorie, formale Sprachen und Komplexitätstheorie*. 2. Bonn–Reading, Mass. [u.a.] : Addison-Wesley, 1990

[HW38] Kapitel III & XI In: HARDY, G. H. ; WRIGHT, E. M.: *An Introduction to the Theory of Numbers*. Oxford University Press, 1938

[IK96] IRANI, Sandy ; KARLIN, Anna R.: Online Computation. In: HOCHBAUM, D. S. (Hrsg.): *Approximation Algorithms for NP-hard Problems*. Boston, Mass. : PWS Publishing Company, 1996, Kapitel 13, S. 521–564

[IKP96] IRANI, Sandy ; KARLIN, Anna R. ; PHILLIPS, Steven: Strongly Competitive Algorithms for Paging with Locality of Reference. In: *SIAM Journal on Computing* 25 (1996), Juni, Nr. 3, S. 477–497

[Ira91] IRANI, Sandy: Two Results on the List Update Problem. In: *Information Processing Letters* 38 (1991), S. 301–306

[Ira94] IRANI, Sandy: Coloring Inductive Graphs On-Line. In: *Algorithmica* 11 (1994), Nr. 1, S. 53–72

[Jac55] JACKSON, J. R.: Scheduling a Production Line to Minimize Maximum Tardiness / Management Sciences Research Project, University of California at Los Angeles. 1955. – Research Report No. 43

[Kar78] KARP, Richard M.: A Characterization of the Minimum Cycle Mean in a Digraph. In: *Discrete Mathematics* 23 (1978), S. 309–311

[KMMO90] KARLIN, Anna R. ; MANASSE, Mark S. ; MCGEOCH, Lyle A. ; OWICKI, Susan: Competitive Randomized Algorithms for Non-Uniform Problems. In: *Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms —SODA '90 (San Francisco, Ca., January 22–24, 1990)*. New York : ACM Press, 1990, S. 301–309. – (Journalveröffentlichung ist [KMMO94].)

[KMMO94] KARLIN, Anna R. ; MANASSE, Mark S. ; MCGEOCH, Lyle A. ; OWICKI, Susan: Competitive Randomized Algorithms for Nonuniform Problems. In: *Algorithmica* 11 (1994), S. 542–571. – (Konferenzveröffentlichung ist [KMMO90].)

[KMV91] KHULLER, Samir ; MITCHELL, Stephen G. ; VAZIRANI, Vijay V.: On-Line Algorithms for Weighted Bipartite Matching and Stable Marriages. In: ALBERT, J. L. (Hrsg.) ; MONIEN, B. (Hrsg.) ; RODRÍGUEZ-ARTALEJO, M. (Hrsg.): *Proceedings of the 18th International Colloquium on Automata, Languages and Programming —ICALP '91 (Madrid, Spain, July 8–12, 1991)*. Berlin-Heidelberg-New York : Springer-Verlag, 1991 (Lecture Notes in Computer Science 510), S. 728–738. – (Journalveröffentlichung ist [KMV94].)

[KMV94] KHULLER, Samir ; MITCHELL, Stephen G. ; VAZIRANI, Vijay V.: On-Line Algorithms for Weighted Bipartite Matching and Stable Marriages. In: *Theoretical Computer Science* 127 (1994), Mai, Nr. 2, S. 255–267. – (Konferenzveröffentlichung ist [KMV91].)

[Kou99] KOUTSOUPIAS, Elias: Weak Adversaries for the k-Server Problem (Extended Abstract). In: *Proceedings of the 40th Annual Symposium on Foundations of Computer Science —FOCS '99 (New York City, NY, October 17–19, 1999)*. Los Alamitos-Washington-Brussels-Tokyo : IEEE Computer Society Press, 1999, S. 444–449

[KP91] KALYANASUNDARAM, Bala ; PRUHS, Kirk: On-Line Weighted Matching. In: *Proceedings of the Second Annual ACM-SIAM Symposium on Discrete Algorithms —SODA '91 (San Francisco, Ca., January 28–30, 1991)*. New York : ACM Press, 1991, S. 234–240. – (Journalveröffentlichung ist [KP93].)

[KP93] KALYANASUNDARAM, Bala ; PRUHS, Kirk: Online Weighted Matching. In: *Journal of Algorithms* 14 (1993), Nr. 3, S. 478–488. – (Konferenzveröffentlichung ist [KP91].)

[KP94a] KOUTSOUPIAS, Elias ; PAPADIMITRIOU, Christos H.: Beyond Competitive Analysis. In: *Proceedings of the 35th Annual Symposium on Foundations of Computer Science —FOCS '94 (Santa Fe, New Mexico, November 20–22, 1994)*. Los Alamitos-Washington-Brussels-Tokyo : IEEE Computer Society Press, 1994, S. 394–400

[KP94b] KOUTSOUPIAS, Elias ; PAPADIMITRIOU, Christos H.: On the k-Server Conjecture. In: *Proceedings of the 26th Annual ACM Symposium on Theory of Computing —STOC '94 (Montréal, Québec, Canada, May 23–25, 1994)*. New York : ACM Press, 1994, S. 507–511. – (Journalveröffentlichung ist [KP95c].)

[KP95a] KALYANASUNDARAM, Bala ; PRUHS, Kirk: The Online Transportation Problem (Preliminary Version). In: SPIRAKIS, P. G. (Hrsg.): *Proceedings of the Third Annual European Symposium on Algorithms —ESA '95 (Corfu, Greece, September 25–27, 1995)*. Berlin-Heidelberg-New York : Springer-Verlag, 1995 (Lecture Notes in Computer Science 979), S. 484–493

[KP95b] KALYANASUNDARAM, Bala ; PRUHS, Kirk: Speed is as Powerful as Clairvoyance (Preliminary Version). In: *Proceedings of the 36th Annual Symposium on Foundations of Computer Science —FOCS '95 (Milwaukee, Wisconsin, October 23–25, 1995)*. Los Alamitos-Washington-Brussels-Tokyo : IEEE Computer Society Press, 1995, S. 214–221

[KP95c] KOUTSOUPIAS, Elias ; PAPADIMITRIOU, Christos H.: On the k-Server Conjecture. In: *Journal of the ACM* 42 (1995), Nr. 5, S. 971–983. – (Konferenzveröffentlichung ist [KP94b].)

[KP96] KALYANASUNDARAM, Bala ; PRUHS, Kirk: An Optimal Deterministic Algorithm for Online b-Matching. In: CHANDRU, V. (Hrsg.): *Proceedings of the 16th Conference of Foundations of Software Technology and Theoretical Computer Science, (Hyderabad, India, December 18–20, 1996)*. Berlin-Heidelberg-New York : Springer-Verlag, 1996 (Lecture Notes in Computer Science 1180), S. 193–199

[KP00] KALYANASUNDARAM, Bala ; PRUHS, Kirk: An Optimal Deterministic Algorithm for Online b-Matching. In: *Theoretical Computer Science* 233 (2000), S. 319–325

[KPR92] KARLIN, Anna R. ; PHILLIPS, Steven J. ; RAGHAVAN, Prabhakar: Markov Paging (Extended Abstract). In: *Proceedings of the 33rd Annual Symposium on Foundations of Computer Science —FOCS '92 (Pittsburgh, Pennsylvania, October 24–27, 1992).* Los Alamitos-Washington-Brussels-Tokyo : IEEE Computer Society Press, 1992, S. 208–217

[KS92] KOREN, Gilad ; SHASHA, Dennis: D^{over}: An Optimal Online Scheduling Algorithm for Overloaded Real-Time Systems. In: *Proceedings of the Real-Time Systems Symposium (Phoenix, Arizona, December 2 –4, 1992).* Los Alamitos, Ca : IEEE Computer Society Press, 1992, S. 290–299

[KS94] KOREN, Gilad ; SHASHA, Dennis: MOCA: a Multiprocessor On-Line Competitive Algorithm for Real-Time System Scheduling. In: *Theoretical Computer Science* 128 (1994), S. 75–97

[KT91] KAO, Ming-Yang ; TATE, Stephen R.: Online Matching with Blocked Input. In: *Information Processing Letters* 38 (1991), Mai, Nr. 3, S. 113–116

[KT92] KIERSTEAD, H. A. ; TROTTER, W. T.: On-Line Graph Coloring. In: MCGEOCH, L. A. (Hrsg.) ; SLEATOR, D. D. (Hrsg.): *Proceedings of a DIMACS Workshop on On-Line Algorithms (Rutgers University, February 11–13, 1991)* Bd. 7 American Mathematical Society, 1992, S. 85–92

[KTW99] KELLERER, Hans ; TAUTENHAHN, Thomas ; WOEGINGER, Gerhard J.: Approximability and Nonapproximability Results for Minimizing Total Flow Time on a Single Machine. In: *SIAM Journal on Computing* 28 (1999), Nr. 4, S. 1155–1166

[Kuh55] KUHN, H. W.: The Hungarian Method for the Assignment Problem. In: *Naval Research Logistics Quarterly* 2 (1955), S. 83–97

[KVV90] KARP, Richard M. ; VAZIRANI, Umesh V. ; VAZIRANI, Vijay V.: An Optimal Algorithm for On-Line Bipartite Matching. In: *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing —STOC '90 (Baltimore, Maryland, May 14–16, 1990).* New York : ACM Press, 1990, S. 352–358

[Leo98] LEONARDI, Stefano: On-Line Network Routing. In: FIAT, A. (Hrsg.) ; WOEGINGER, G. J. (Hrsg.): *Online Algorithms: The State of the Art.* Berlin-Heidelberg-New York : Springer-Verlag, 1998 (Lecture Notes in Computer Science 1442), S. 242–267

[LMS99] LEONARDI, Stefano ; MARCHETTI-SPACCAMELA, Alberto: On-Line Resource Management with Applications to Routing and Scheduling. In: *Algorithmica* 24 (1999), Nr. 1, S. 29–49

[LP86] Lovász, L. ; Plummer, M. D.: *Matching Theory.* Amsterdam-New York-Oxford-Tokyo : North-Holland, 1986 (North-Holland Mathematics Studies 121, Annals of Discrete Mathematics 29)

[LR94] Lund, Carsten ; Reingold, Nick: Linear Programs for Randomized On-Line Algorithms. In: *Proceedings of the Fifth Annual ACM-SIAM Symposium on Discrete Algorithms —SODA '94 (Arlington, Va., January 23–25, 1994).* New York : ACM Press, 1994, S. 382–391

[LR97] Leonardi, Stefano ; Raz, Danny: Approximating Total Flow Time on Parallel Machines. In: *Proceedings of the 29th Annual ACM Symposium on Theory of Computing —STOC '97 (El Paso, Texas, May 4–6, 1997).* New York : ACM Press, 1997, S. 110–119

[LST90] Lenstra, Jan K. ; Shmoys, David B. ; Tardos, Éva: Approximation Algorithms for Scheduling Unrelated Parallel Machines. In: *Mathematical Programming* 46 (1990), Nr. 3, S. 259–271

[LT94] Lipton, Richard J. ; Tomkins, Andrew: Online Interval Scheduling. In: *Proceedings of the Fifth Annual ACM-SIAM Symposium on Discrete Algorithms —SODA '94 (Arlington, Va., January 23–25, 1994).* New York : ACM Press, 1994, S. 302–311

[MBLR97] Monien, Burkhard ; Berenbrink, Petra ; Lüling, Reinhard ; Riedel, Marco: Online Scheduling of Continuous Media Streams. In: Freksa, C. (Hrsg.) ; Jantzen, M. (Hrsg.) ; Valk, R. (Hrsg.): *Foundations of Computer Science: Potential-Theory-Cognition.* Berlin-Heidelberg-New York : Springer-Verlag, 1997 (Lecture Notes in Computer Science 1337), S. 313–320

[McN59] McNaughton, Robert: Scheduling with Deadlines and Loss Functions. In: *Management Science* 6 (1959), Oktober, Nr. 1, S. 1–12

[MMS88] Manasse, Mark S. ; McGeoch, Lyle A. ; Sleator, Daniel D.: Competitive Algorithms for On-Line Problems. In: *Proceedings of the 20th Annual ACM Symposium on Theory of Computing —STOC '88 (Chicago, IL, May 2–4, 1988).* New York : ACM Press, 1988, S. 322–333. – (Journalveröffentlichung ist [MMS90].)

[MMS90] Manasse, Mark S. ; McGeoch, Lyle A. ; Sleator, Daniel D.: Competitive Algorithms for Server Problems. In: *Journal of Algorithms* 11 (1990), Nr. 2, S. 208–230. – (Konferenzveröffentlichung ist [MMS88].)

[MPT94] Motwani, Rajeev ; Phillips, Steven ; Torng, Eric: Nonclairvoyant scheduling. In: *Theoretical Computer Science* 130 (1994), Nr. 1, S. 17–47

[MS91] McGeoch, Lyle A. ; Sleator, Daniel D.: A Strongly Competitive Randomized Paging Algorithm. In: *Algorithmica* 6 (1991), Nr. 6, S. 816–825

[MV80] Micali, Silvio ; Vazirani, Vijay V.: An $O(\sqrt{|v|} \cdot |E|)$ Algorithm for Finding Maximum Matching in General Graphs. In: *Proceedings of the 21th*

Annual Symposium on Foundations of Computer Science —FOCS '80 (Syracuse, New York, October 13–15, 1980). Los Alamitos-Washington-Brussels-Tokyo : IEEE Computer Society Press, 1980, S. 17–27

[OA92] Orlin, James B. ; Ahuja, Ravindra K.: New Scaling Algorithms for the Assignment and Minimum Mean Cycle Problems. In: *Mathematical Programming* 54 (1992), S. 41–56

[PL88] Peterson, Paul A. ; Loui, Michael C.: The General Maximum Matching Algorithm of Micali and Vazirani. In: *Algorithmica* 3 (1988), S. 511–533

[Pre99] Preis, Robert: Linear Time 1/2-Approximation Algorithm for Maximum Weighted Matching in General Graphs. In: Meinel, Ch. (Hrsg.) ; Tison, S. (Hrsg.): *Proceedings of the 16th Annual Symposium on Theoretical Aspects in Computer Science —STACS '99, (Trier, Germany, March 4–6, 1999).* Berlin-Heidelberg-New York : Springer-Verlag, 1999 (Lecture Notes in Computer Science 1563), S. 259–269

[PSTW97] Phillips, Cynthia A. ; Stein, Cliff ; Torng, Eric ; Wein, Joel: Optimal Time-Critical Scheduling via Resource Augmentation (Extended Abstract). In: *Proceedings of the 29th Annual ACM Symposium on Theory of Computing —STOC '97 (El Paso, Texas, May 4–6, 1997).* New York : ACM Press, 1997, S. 140–149

[PSW95] Phillips, Cynthia A. ; Stein, Clifford ; Wein, Joel: Scheduling Jobs that Arrive Over Time (Extended Abstract). In: Akl, S. G. (Hrsg.) ; Dehne, F. (Hrsg.) ; Sack, J.-R. (Hrsg.) ; Santoro, N. (Hrsg.): *Proceedings of the Fourth International Workshop on Algorithms and Data Structures — WADS '95 (Kingston, Canada, August 16–18, 1995).* Berlin-Heidelberg-New York : Springer-Verlag, 1995 (Lecture Notes in Computer Science 955), S. 86–97

[PW98] Phillips, Steven J. ; Westbrook, Jeffery: Online Load Balancing and Network Flow. In: *Algorithmica* 21 (1998), Juli, Nr. 3, S. 245–261

[Rag92] Raghavan, Prabhakar: A Statistical Adversary for On-Line Algorithms. In: McGeoch, L. A. (Hrsg.) ; Sleator, D. D. (Hrsg.): *Proceedings of a DIMACS Workshop on On-Line Algorithms (Rutgers University, February 11–13, 1991)* Bd. 7 American Mathematical Society, 1992, S. 79–83

[RS89] Raghavan, Prabhakar ; Snir, Marc: Memory Versus Randomization in On-Line Algorithms (Extended Abstract). In: Ausiello, G. (Hrsg.) ; Dezani-Ciancaglini, M. (Hrsg.) ; Rocca, S. R. D. (Hrsg.): *Proceedings of the 16th International Colloquium on Automata, Languages and Programming —ICALP '89 (Stresa, Italy, July 11–15, 1989).* Berlin-Heidelberg-New York : Springer-Verlag, 1989 (Lecture Notes in Computer Science 372), S. 687–703. – (Überarbeitete Journalveröffentlichung ist [RS94].)

[RS94] Raghavan, Prabhakar ; Snir, Marc: Memory Versus Randomization in On-Line Algorithms. In: *IBM Journal of Research and Development* 38

(1994), November, Nr. 6, S. 683–707. – (Konferenzveröffentlichung ist [RS89].)

[RWS94] Reingold, Nick ; Westbrook, Jeffery ; Sleator, Daniel D.: Randomized Competitive Algorithms for the List Update Problem. In: *Algorithmica* 11 (1994), Nr. 1, S. 15–32

[Sga98] Sgall, Jiří: On-Line Scheduling. In: Fiat, A. (Hrsg.) ; Woeginger, G. J. (Hrsg.): *Online Algorithms: The State of the Art*. Berlin-Heidelberg-New York : Springer-Verlag, 1998 (Lecture Notes in Computer Science 1442), S. 196–231

[ST85] Sleator, Daniel D. ; Tarjan, Robert E.: Amortized Efficiency of List Update and Paging Rules. In: *Communications of the ACM* 28 (1985), Februar, Nr. 2, S. 202–208

[Ste58] Stern, M. A.: Ueber eine Zahlentheoretische Funktion. In: *Journal für die reine und angewandte Mathematik* 55 (1858), S. 193–220

[SWW95] Shmoys, David B. ; Wein, Joel ; Williamson, David P.: Scheduling Parallel Machines On-Line. In: *SIAM Journal on Computing* 24 (1995), Dezember, Nr. 6, S. 1313–1331

[Tar85] Tarjan, Robert E.: Amortized Computational Complexity. In: *SIAM Journal on Algebraic and Discrete Methods* 6 (1985), April, Nr. 2, S. 306–318

[Tei93] Teia, Boris: A Lower Bound for Randomized List Update Algorithms. In: *Information Processing Letters* 47 (1993), Nr. 1, S. 5–9

[TGS94] Tanaev, V. S. ; Gordon, V. S. ; Shafransky, Y. M.: *Mathematics and Its Applications*. Bd. 284: *Scheduling Theory. Single-Stage Systems*. Dordrecht-Boston-London : Kluwer Academic Publishers, 1994

[Tim93] Timkovsky, Vadim G.: The Complexity of Unit-Time Job Shop Scheduling / McMaster University, Department of Computer Science and Systems. Hamilton, Ontario, Canada, Dezember 1993 (No. 93-09). – Forschungsbericht

[Tor98] Torng, Eric: A Unified Analysis of Paging and Caching. In: *Algorithmica* 20 (1998), S. 175–200

[TSS94] Tanaev, V. S. ; Sotskov, Y. N. ; Strusevich, V. A.: *Mathematics and Its Applications*. Bd. 285: *Scheduling Theory. Multi-Stage Systems*. Dordrecht-Boston-London : Kluwer Academic Publishers, 1994

[Vaz94] Vazirani, Vijay V.: A Theory of Alternating Paths and Blossoms for Proving Correctness of the $O(\sqrt{V}E)$ General Graph Matching Algorithm. In: *Combinatorica* 14 (1994), Nr. 1, S. 71–109

[Ves97] Vestjens, Arjen P. A.: *On-line Machine Scheduling*, Eindhoven University of Technology, Dissertation, November 1997

[Wes00] WESTBROOK, Jeffery: Load Balancing for Response Time. In: *Journal of Algorithms* 35 (2000), April, Nr. 1, S. 1–16

[WM91] WANG, Fuxing ; MAO, Decao: Worst Case Analysis for On-Line Scheduling in Real-Time Systems / Department of Computer and Information Science (Computer Science Department), University of Massachusetts at Amherst. 1991 (UM-CS-1991-054). – Forschungsbericht

[Woe94] WOEGINGER, Gerhard J.: On-Line Scheduling of Jobs with Fixed Start and End Times. In: *Theoretical Computer Science* 130 (1994), Nr. 1, S. 5–16

[Yao77] YAO, Andrew Chi-Chih: Probabilistic Computations: Toward a Unified Measure of Complexity (extended abstract). In: *Proceedings of the 18th Annual Symposium on Foundations of Computer Science (Providence, Rhode Island, October 31–November 2, 1977)*. Los Alamitos-Washington-Brussels-Tokyo : IEEE Computer Society Press, 1977, S. 222–227

[You91a] YOUNG, Neal: *Competitive Paging and Dual-Guided On-Line Weighted Caching and Matching Algorithms*, Princeton University, Computer Science Departement, Dissertation, 1991. – Verfügbar als Technischer Bericht CS-TR-348-91, Ergebnisse veröffentlicht in [You94].

[You91b] YOUNG, Neal: On-Line Caching as Cache Size Varies. In: *Proceedings of the Second Annual ACM-SIAM Symposium on Discrete Algorithms — SODA '91 (San Francisco, Ca., January 28–30, 1991)*. New York : ACM Press, 1991, S. 241–250

[You94] YOUNG, Neal: The k-Server Dual and Loose Competitiveness for Paging. In: *Algorithmica* 11 (1994), S. 525–541. – (Konferenzveröffentlichung ist [You91b], detailierte Ausgabe ist [You91a].)

In dieser Referenzliste werden einige Arbeiten mehrfach, d. h. in verschiedenen Versionen zitiert. Häufig handelt es sich dabei um eine frühe Veröffentlichung im Rahmen einer Konferenz und eine spätere, überarbeitete Version, die in einer Zeitschrift erschienen ist. Die frühere Vorstellung einer Arbeit ist in diese Referenzliste mit aufgenommen, wenn es für die historische Einordnung und die korrekte Darstellung der Abfolge von Aufsätzen mit den darin enthaltenen Ideen notwendig erschien. Schwieriger zu beschaffende Quellen wie Dissertationen, Technische Berichte oder Vorabdrucke wurden nur zitiert, wenn darin Analysen und Beweise umfangreicher dargestellt werden, als es in den späteren Zeitschriftenartikeln der Fall ist, oder wenn bisher keine weitere Version der Arbeit publiziert wurde.