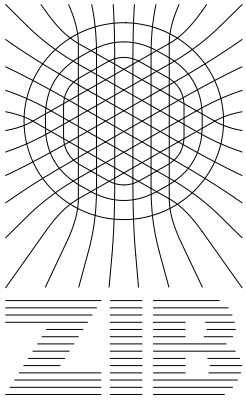

Konrad-Zuse-Zentrum
für Informationstechnik Berlin

Takustraße 7
D-14195 Berlin-Dahlem
Germany

SVEN O. KRUMKE UND JÖRG RAMBAU

Probieren geht über Studieren? Entscheidungshilfen für kombinatorische Online-Optimierungsprobleme in der innerbetrieblichen Logistik

ZIB-Report 02-05 (Januar 2002)

PROBIEREN GEHT ÜBER STUDIEREN? ENTSCHEIDUNGSHILFEN FÜR KOMBINATORISCHE ONLINE-OPTIMIERUNGSPROBLEME IN DER INNERBETRIEBLICHEN LOGISTIK

SVEN O. KRUMKE AND JÖRG RAMBAU

ZUSAMMENFASSUNG. Die Automatisierung von innerbetrieblicher Logistik erfordert – über die physikalische Steuerung von Geräten hinaus – auch eine effiziente Organisation der Transporte: ein Aufgabenfeld der kombinatorischen Optimierung. Dieser Artikel illustriert anhand von konkreten Aufgabenstellungen die Online-Problematik (unvollständiges Wissen) sowie die Echtzeit-Problematik (beschränkte Rechenzeit), auf die man in der innerbetrieblichen Logistik trifft. Der Text gibt einen Überblick über allgemeine Konstruktionsprinzipien für Online-Algorithmen und Bewertungsmethoden, die bei der Entscheidung helfen, welche Algorithmen für eine vorliegende Problemstellung geeignet sind.

1. EINLEITUNG

Fortschritte in der Automatisierungstechnik haben eine wachsende Flexibilisierung der betrieblichen Fertigung und Lagerhaltung ermöglicht. Gerätesteuerungen von Fließbändern, Aufzügen und führerlosen Transportfahrzeugen besorgen den Palettentransport in Versandzentren automatisch. Diese Errungenschaften der (meist) kontinuierlichen mathematischen Modellierung der Physik der Transportmittel sind nur eine Seite der Medaille: Flexibilität muss auch ausgenutzt werden können durch intelligente Organisation der Auftragsverarbeitung, wie z. B. Reihenfolgeplanung. Diese Probleme sind (meist) kombinatorischer Natur und können i. d. R. von den mikroskopischen physikalischen Modellen vollständig entkoppelt betrachtet werden. Sie führen auf die Methoden der Operationsplanung oder allgemeiner der kombinatorischen Optimierung. Die kombinatorische Optimierung liefert Werkzeuge, um Kosten minimieren bzw. vorhandene Ressourcen gut auszunutzen. Damit bietet sie wichtige Entscheidungshilfen.

In der *Offline-Optimierung* geht man im allgemeinen davon aus, dass die Daten jeder Probleminstanz vollständig gegeben sind. Zahlreiche Problemstellungen in der Praxis erfordern jedoch Entscheidungen ohne vollständige Informationen über die Eingabedaten. Oft ist ist unabdingbar, dass nach jedem neuen Stück der Daten, das bekannt wird, ein Teil der Gesamtlösung unmittelbar und ohne Wissen zukünftiger Ereignisse generiert wird. Ein solches Problem nennt man ein *Online-Problem*. In vielen Anwendungen bestehen zudem noch harte Zeitrestriktionen für die Rechenzeit von Algorithmen. Bei einem Online-Problem mit *Echtzeit-Anforderungen* muß ein Algorithmus innerhalb einer garantierten Antwortzeit eine gültige Lösung liefern.

Welche Hilfsmittel stehen zur Konstruktion von Online-Algorithmen zur Verfügung? Wie kann man die Qualität eines Online-Algorithmus beurteilen? Welchen Algorithmus soll man für ein vorliegendes Problem einsetzen? In diesem Artikel geben stellen wir eine Auswahl von mathematischen Entscheidungshilfen vor; Beispiele aus realen Projekten illustrieren Stärken und Schwächen der Verfahren.

Gefördert von der Deutschen Forschungsgemeinschaft durch ein Projekt im DFG-Schwerpunktprogramm SPP 462.

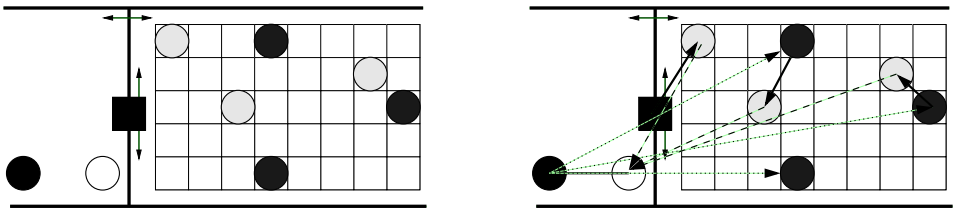

ABBILDUNG 2. Das Steuerungsproblem für ein Regalbediengerät schematisch dargestellt: helle Punkte sind Auslageraufträge und müssen auf dem hellen Punkt vor dem Lager abgesetzt werden, dunkle Punkte sind Einlageraufträge und werden vom dunklen Punkt vor dem Lager abgeholt; daneben: eine mögliche Abarbeitungsreihenfolge: man sieht, dass die Leerfahrtstrecke (durchgezogenene Pfeile) und damit die Leerfahrtzeit nur von der Abarbeitungsreihenfolge der Aufträge beeinflusst wird

2. DREI LOGISTIKPROBLEME MIT ONLINE-CHARAKTER UND ECHTZEIT-ANFORDERUNGEN

Wir starten mit drei Fallbeipielen aus der innerbetrieblichen Logistik.

2.1. **Regalbediengeräte.** Das Bediengerät eines Hochregallagers hat die Aufgabe, Transportaufträge abzuarbeiten, deren Start- und Zielpunkte sich auf den Seiten des von ihm bedienten Regalganges befinden.

Normalerweise liegen mehrere noch nicht abgearbeitete Transportaufträge vor, deren Anzahl von den momentanen Produktionsverhältnissen abhängt. Im zeitlichen Produktionsablauf muss entschieden werden, welcher der Transportaufträge als nächster abzuarbeiten ist (Abbildung 2). Ziel ist die Minimierung der Gesamt-Leerfahrzeit des Bediengerätes, wobei u. U. zusätzlich zu berücksichtigen ist, dass Transportaufträge nicht beliebig lange bis zu ihrer Abarbeitung warten dürfen und Reihenfolgebedingungen eingehalten werden müssen. Außerdem soll das Bediengerät nicht stillstehen, wenn Aufträge verfügbar sind.

ABBILDUNG 1. Ein Hochregallagerbediengerät

Der *Online*-Aspekt dieses Problems besteht darin, dass zum Zeitpunkt einer Steuerungsentscheidung (welcher Auftrag soll als nächstes ausgeführt werden?) noch nicht alle zukünftigen Aufträge bekannt sind. Die *Echtzeit-Anforderung* besteht in der Auflage, nicht zu lange mit der Berechnung einer Steuerung zuzubringen (etwa eine Sekunde maximal).

In einer PC-Fertigungsanlage der Firma Siemens-Nixdorf-Informationssysteme GmbH (SNI) Augsburg wird ein solches Hochregallagersystem eingesetzt. Die vorhandene Prioritäten-Steuerung zeigt ein auffallend ineffizientes Verhalten nach Störungen – also dann, wenn besonders viele unbearbeitete Aufträge vorliegen. Wie kann man im Voraus abschätzen, wie sich alternative Steuerungen verhalten werden? Kann man durch eine optimierende Steuerung die Leerfahrtzeit reduzieren?

2.2. **Kommissioniermobile.** Das Versandzentrum der Herlitz PBS AG, Berlin, besteht aus automatischen Hochregallagern, die mittels moderner Fördertechnik mit verschiedenen Kommissionierbereichen verbunden sind.

In diesem Versandzentrum gehen permanent vorher nicht bekannte Bestellungen ein, die in einem vorgegebenen (kurzen) Zeitrahmen kommissioniert werden müssen.

ABBILDUNG 3. Ein halbautomatisches Kommissioniermobil

In der Glückwunschkartenkommissionierungsanlage fahren auf einem festen Rundkurs acht Kommissioniermobile an Regallagern entlang, in denen Gebinde von rund 4000 verschiedenen Glückwunschkarten lagern. Die Mobile können bis zu 19 Aufträge gleichzeitig bearbeiten. Das System unterliegt einer Vielzahl von technischen Rahmenbedingungen. Beispielsweise können sich die Mobile einander nicht überholen, was häufig Rückstaus verursacht. Das Problem der Steuerung der Kommissioniermobile besteht darin, die Aufträge so auf die Fahrzeuge zu verteilen, dass eine schnellstmögliche und termingerechte Bearbeitung der Kundenaufträge gewährleistet ist.

Jeder Kundenauftrag erfordert eine Anzahl von „Picks“ (Entnahme von Karten aus den Regalen). Da diese unabhängig von der Zuweisung der Aufträge auf die Mobile ausgeführt werden müssen, besteht ein möglicher Optimierungsansatz darin, pro Stopp möglichst viele Picks durchzuführen. Äquivalent formuliert heißt dies, die Gesamtanzahl aller Stopps zu minimieren (Abbildung 4).

Auch hier sind zukünftige Aufträge noch nicht vollständig bekannt, wenn eine (irreversible) Zuordnung von bekannten Aufträgen auf Mobile vorgenommen werden muss (*Online-Aspekt*). Die *Echtzeit-Anforderung* bedeutet eine Antwortzeit im Sekundenbereich.

Welche Algorithmen sind aber nun geeignet und wie kann man eine Antwort auf diese Frage begründen?

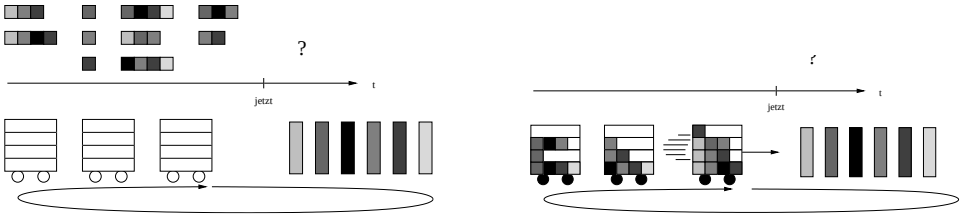

ABBILDUNG 4. Das Kommissionierproblem schematisch dargestellt: Kästchenreihen symbolisieren Aufträge, die über die Zeit verteilt bekannt werden; für jeden Auftrag muss an der Regalposition entsprechend der Graustufen der Kästchen gestoppt werden; daneben: eine mögliche Zuordnung: die Anzahl der Stopps eines Fahrzeugs entspricht der Anzahl verschieden heller Kästchen; ist ein Fahrzeug voll beladen, so fährt es los. Nachdem ein Mobil seine kommissionierten Aufträge abgegeben hat, steht es wieder leer zur Verfügung

2.3. **Aufzüge.** Im selben Versandlager der Firma Herlitz befinden sich die Produktions- und Kommissionierbereiche in unterschiedlichen Etagen, die durch zwei Palettenfördertürme miteinander verbunden sind.

Jeder dieser Türme enthält ein System aus fünf Palettenaufzügen, die jeweils für eine Palette Platz bieten. Die Aufzüge werden unabhängig voneinander gesteuert. Warten nun mehrere Paletten auf denselben Aufzug, so muss die zugehörige Steuerung *online* und in *Echtzeit* entscheiden, welche als nächstes bedient werden soll. Ziel ist es, die durchschnittliche bzw. die maximale Flusszeit (d. h. Wartezeit plus Transportzeit) der Paletten zu minimieren (Abbildung 6).

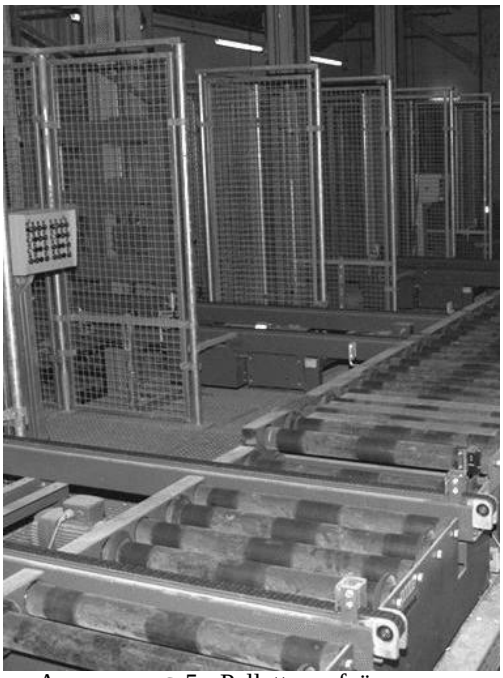

ABBILDUNG 5. Pallettenaufzüge

Vorinstalliert durch den Hersteller sind eine FIFO-Steuerung (bearbeite Aufträge in der Reihenfolge ihres Auftretens) und eine „Nächste-Nachbar-Steuerung“ (fahre zuerst zum nächstgelegenen Auftrag). Erstere zeigt sich als durchsatzschwach, letztere als „ungerecht“ bzgl. der Flusszeiten individueller Paletten: Es kann passieren, dass einzelne Paletten ewig auf den Aufzug warten.

Wie kann man nun das Verhalten von Online-Algorithmen vorhersagen und beurteilen?

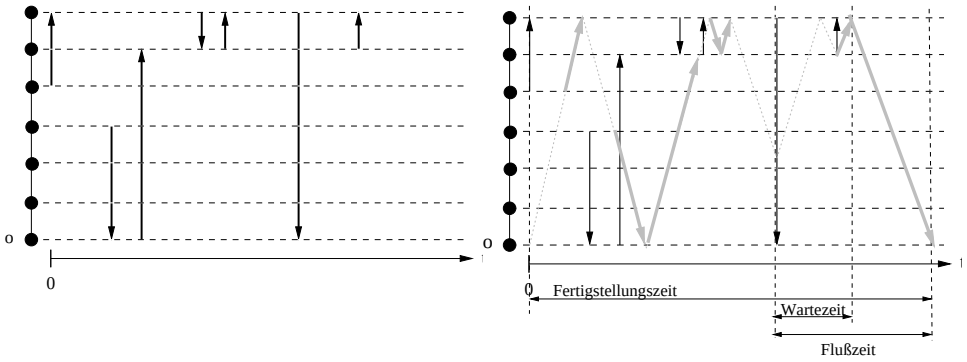

ABBILDUNG 6. Das Aufzugproblem schematisch dargestellt: Pfeile über der Zeitachse symbolisieren Transportaufträge, die zu den entsprechenden Zeitpunkten bekannt werden; daneben: ein möglicher Transportplan: aufgetragen ist die Aufzugbewegung über der Zeit; ein beladener Transport ist dicker gezeichnet

3. MATHEMATISCHE MODELLIERUNG VON ONLINE-PROBLEMEN

Informell kann man ein Online-Problem als ein Problem definieren, in dem die Problemdaten „schrittweise“ einem Online-Algorithmus bekannt werden. Formaler modelliert man in der Online-Optimierung die Eingabe als eine (endliche) *Anfragefolge* $\sigma = r_1, r_2, \ldots$. Nach dem Bekanntwerden der Anfrage r_i muss ein Online-Algorithmus eine Entscheidung über seine weitere Arbeitsweise treffen. Die Anfragen $r_{i+1}, r_{i+2}, \ldots$ sind ihm dabei unbekannt.

Je nach dem spezifischen Online-Modell sind die Entscheidungen des Online-Algorithmus zusätzlichen Einschränkungen unterworfen. Im *Sequenzmodell* etwa muß die Anfrage r_i bei Bekanntwerden sofort und unwiderruflich verarbeitet werden, erst danach wird r_{i+1} bekanntgegeben. Im *Zeitstempelmodell* hat jede Anfrage r_i eine Freigabezeit $t_i \geq 0$, eine nichtnegative reelle Zahl, welche den Zeitpunkt spezifiziert, zu dem r_i bekannt wird. Der Online-Algorithmus muß zu jeder Zeit t sein Verhalten nur aufgrund der bisher bekannten Anfragen, d. h. der Anfragen mit Freigabezeit höchstens t, bestimmen. Der Unterschied zum Sequenzmodell liegt darin, dass der Online-Algorithmus mit der Bearbeitung von Anfragen warten und so Entscheidungen verzögern kann. Allerdings impliziert Warten zusätzliche Kosten, die normalerweise von der verstrichenen Zeit abhängen. Auch im Zeitstempelmodell kann der Online-Algorithmus nur solche Entscheidungen revidieren, die noch nicht ausgeführt worden sind.

4. KONSTRUKTIONSPRINZIPIEN FÜR ONLINE-ALGORITHMEN

Es zeigt sich, dass viele Online-Algorithmen nach einem der folgenden Schemata funktionieren. Dabei kann man zwei prinzipiell verschiedene Herangehensweisen unterscheiden:

- Die Steuerung entscheidet stets „ad hoc“, welchen Auftrag sie als nächstes bearbeitet.
- Die Steuerung arbeitet nach einem im Voraus berechneten „Plan“, der ggf. nach neuen Ereignissen (neuer Auftrag, Auftrag fertig, Störung, Störung beseitigt etc.) modifiziert wird.

4.1. **Ohne Vorausplanung.** Die folgenden Verfahren benötigen keinen Plan, nach dem sie arbeiten:

FIFO. Hier werden die Aufträge in der Reihenfolge ihres Auftretens erledigt. Wenn diese Reihenfolge aus technischen Gründen notwendig ist, hat man keine Wahl. Ansonsten ist dieser Ansatz nicht effizienzorientiert: Wie die Aufträge für den jeweiligen „Server“ zusammenpassen, beeinflusst die Steuerung nicht.

Es ist interessant zu entdecken, dass in vielen Steuerungen das FIFO-Konzept als „tiebreaker“ benutzt wird: innerhalb einer Prioritätsklasse wird die Abarbeitungsreihenfolge nach FIFO bestimmt.

Nächster-Nachbar. „Der nächste Auftrag ist der nächste.“ Man bedient den Auftrag als nächstes, mit dessen Bearbeitung man unter dem geringsten Ressourcenaufwand aus der augenblicklichen Konfiguration beginnen kann. Hier versucht man, unter geringstmöglichen *lokalen* Kosten zum nächsten Auftrag zu gelangen. Man beachtet jedoch nicht andere noch abzuarbeitende Aufträge.

Dies kann zu insgesamt ungünstigen Abarbeitungsreihenfolgen führen. Ein wichtigeres Problem ist aber die schlechte Zuverlässigkeit: Befindet sich der Server nach der Abarbeitung eines Auftrags nie in der Nähe einer bestimmten Palette, so wird diese auch nie transportiert – ein Effekt, der im dritten Fallbeispiel durchaus auftritt und dort ärgerliche Konsequenzen hat.

4.2. **Mit Vorausplanung.** Bei den folgenden Verfahren wird der Server stets „nach Plan“ gesteuert.

Greedy. Eine Vorausplanung findet zwar statt, man revidiert aber niemals die Position eines Auftrags im Plan. Neue Aufträge werden bestmöglich eingefügt. Aufträge werden aber durch das Eintragen in einen Plan wenigstens nicht beliebig lange stehengelassen. Man kann sagen, dass das *zeitlich lokale* Optimierungsproblem, aus den bekannten Aufträgen einen optimalen Abarbeitungsplan zu berechnen, nur näherungsweise gelöst wird.

Replan. Nach jedem neuen Ereignis wird für alle bekannten Aufträge ein Plan berechnet, der zum Planungszeitpunkt optimal ist, d. h. der Plan ist *zeitlich lokal* gesehen optimal. Zwischen den Ereignissen werden die Aufträge „nach Plan“ abgearbeitet.

Es hat viele unserer Projektpartner erstaunt, dass selbst dieses Vorgehen – über einen längeren Zeithorizont betrachtet – nicht unbedingt die besten Ergebnisse liefert. Mögliche Schwächen dieser Strategie liegen daran, dass die zum Planungszeitpunkt noch unbekannten Aufträge sich im Nachhinein als sehr ungünstig für den gerade noch optimalen Plan herausstellen können. Außerdem kann es auch hier passieren, dass individuelle Aufträge in jedem zeitliche optimalen Plan ganz hinten landen und somit nie bearbeitet werden.

Ignore. Die Strategie verbindet Optimierung zur Effizienzsteigerung mit Fixierung von Entscheidungen zur Stabilisierung eines Plans: Ein einmal berechneter Plan wird stets zu Ende bearbeitet; dann erst wird aus allen bis dahin aufgelaufenen Aufträgen ein neuer, zeitlich lokal optimaler Plan berechnet.

Hier wird Optimierungspotential des Gesamtsystems geopfert, um eine höhere Stabilität für individuelle Aufträge zu erzielen.

5. Bewertungsmethoden für kombinatorische Online-Algorithmen

Für die Konzeption des jeweiligen logistischen Systems stellt sich die Frage: Welchen Algorithmus soll man wählen, und wie kann man eine Entscheidung begründen? Es hat sich als nützlich herausgestellt, nach *Zertifikaten* für die Leistung eines Online-Algorithmus zu suchen. Unter einem Zertifikat für einen Algorithmus verstehen wir eine (beweisbare oder empirisch erhaltene) Aussage über die Qualität des Algorithmus. Dieses Zertifikat kann sowohl eine absolute Kostenschranke beinhalten als auch einen Vergleich zu den Kosten, die in einer jeden Lösung unvermeidbar sind (*untere Schranken*).

Wir stellen im Folgenden einige Evaluierungsmethoden für Online-Algorithmen vor. Diese Evaluierungsmethoden unterscheiden sich in der mathematischen Strenge ihrer Zertifikate. Hier ist jedoch zu beachten, dass nicht immer das mathematisch schärfste Zertifikat

auch am überzeugendsten ist: zum Beispiel ist ein Zertifikat der Art „Kosten sind mathematisch beweisbar nie höher als k“ womöglich nur für ein sehr großes k beweisbar; die Aussage ist dann zwar verlässlich aber wertlos. Auf der anderen Seite ist ein Zertifikat wie „In Experimenten sind nie Kosten größer als k“ aufgetreten evt. für kleinere k erhältlich; die Aussagekraft kann hier aber dadurch eingeschränkt sein, dass man nicht genau weiß, ob nicht doch einmal in einer unglücklichen Situation höhere Kosten anfallen.

Im Folgenden präsentieren wir vier Evaluierungsmethoden für Online-Algorithmen in der Reihenfolge abnehmender mathematischer Strenge. Wir beschränken uns dabei auf Probleme, bei denen eine Zielfunktion („Kosten“) minimiert werden soll.

5.1. **Kompetitive Analyse.** Das (mathematisch zu beweisende) Zertifikat dieser Evaluierungsmethode für einen Algorithmus A ist von folgender Gestalt: „Auch ein optimaler Algorithmus mit vollständigem Wissen über die zukünftigen Aufträge (ein *optimaler Offline-Algorithmus*) produziert stets mindestens $1/c$ der Kosten, die A produziert.“ In diesem Falle heißt A *c-kompetitiv*.

Die *kompetitive Analyse*, die von Sleator und Tarjan [18] vorgeschlagen wurde, ist eine *worst case* Analyse. Hat man einen c-kompetitiven Algorithmus für ein befriedigend kleines c gefunden, so ist dieses Zertifikat sehr aussagekräftig: auch im schlimmsten denkbaren Fall kann man nicht wesentlich schlechter abschneiden als ein fiktiver optimaler hellsehender Algorithmus.

Ist der Wert c zu groß oder findet man gar überhaupt keinen c-kompetitiven Algorithmus, so kann man i. d. R. mit diesem Zertifikat nicht mehr überzeugen.

Eine Unzulänglichkeit der kompetitiven Analyse besteht darin, dass der zugrundeliegende theoretisch schlimmste Fall in der Praxis häufig nur sehr unwahrscheinlich oder sogar ausgeschlossen ist. Die kompetitive Analyse ist sehr pessimistisch. Oft ist ein optimaler Offline-Algorithmus derart durch seine vollständige Information bevorteilt, dass nur nahezu triviale Kompetitivitätsaussagen beweisbar sind (vgl. *triviality barrier* in [8]).

Durch Einschränkungen für die möglichen Eingaben oder zusätzliche Restriktionen auf die Arbeitsweise des Offline-Algorithmus kann man versuchen, die kompetitive Analyse weniger pessimistisch zu machen [13, 16, 6, 4]. Allerdings helfen auch diese Modifikationen nicht immer, um eine gute Entscheidungshilfe bei der Auswahl eines für die Praxis geeigneten Algorithmus zu erhalten.

5.2. **Vertretbare Belastung.** Bei der recht jungen Methode der *vertretbaren Belastung* [11] wird der Vergleich der Güte des Online-Algorithmus nicht mit dem Ergebnis irgendeines anderen Algorithmus durchgeführt sondern mit einem Parameter, der die Systemlast beschreibt. Die Idee ist, dass der Online-Algorithmus es schwerer hat, wenn das System stärker belastet wird.

Wir messen die Systembelastung durch eine Auftragsmenge R in der folgenden Weise: R heißt Δ-vertretbar, wenn alle endlichen Auftragsmengen $R' \subseteq R$, die während einer Zeitspanne $\Delta' \geq \Delta$ auftreten, in einer Zeitspanne von höchstens Δ' von einem optimalen Server (offline) bearbeitet werden könnten.

Hinter dieser Definition steckt die Erkenntnis, dass die für eine Auftragsmenge R benötigte Abarbeitungszeit umso kürzer ist, je mehr Aufträge man nach Effizienzgesichtspunkten in einem Planungsschritt zusammenfassen kann.

Für eine Δ-vertretbare Auftragsmenge reichen Planungsintervalle von Δ aus, um genausoviel Aufträge abzuarbeiten wie hereinkommen. Ist Δ also klein, so kann der Planungshorizont für einen Planungsschritt recht kurz sein, und trotzdem „läuft das System nicht voll“. Ist Δ groß, so muss man – um überhaupt den nötigen Durchsatz zu erzielen – lange Planungsintervalle erlauben.

Das (mathematisch zu beweisende) Zertifikat dieser Methode für einen Online-Algorithmus A ist von folgender Gestalt: „Die Kosten von A betragen für eine Δ-vertretbare Auftragsmenge höchstens (oder: asymptotisch) $f(\Delta)$ (für eine geeignete reelle Funktion f).“

Auch hier bestehen immer noch die Probleme einer Evaluierung des „schlimmsten Falls“, der möglicherweise nie auftritt. Die Verzerrung durch den Vergleich mit einem fiktiven Algorithmus hat man jedoch eliminiert.

5.3. **A-Posteriori-Analyse.** Dieser Ansatz ist dadurch motiviert, dass die schlimmsten Fälle in den beiden bisher beschriebenen Methoden oft gar nicht auftreten. Daher werden bei der A-posteriori-Analyse nur bereits aufgetretene Auftragsmengen berücksichtigt. Im Nachhinein werden dann für diese – wie bei der kompetitiven Analyse – die Kosten des Online-Algorithmus mit den Kosten eines optimalen Offline-Algorithmus verglichen. Das Verhältnis der Kosten wird manchmal auch als *experimentelle Kompetitivität* bezeichnet.

Häufig kann man auch mit fortgeschrittenen Methoden eine optimale Offline-Lösung nicht finden. Man versucht dann, untere Schranken für die Kosten eines optimalen Offline-Algorithmus (unvermeidbare Kosten) durch eine Relaxierung des Originalproblems zu beweisen.

Das (mathematisch zu beweisende) Zertifikat für einen Online-Algorithmus A ist von folgender Art: „Auf einer speziellen Instanz des Problems (auf einer Menge von Instanzen) hat A (im Durchschnitt) höchstens c mal soviel Kosten produziert wie ein optimaler Offline-Algorithmus (auf einem relaxierten Problem).“ Gelegentlich gibt man auch die *Optimalitätslücke* an: „Der Abstand zwischen den Kosten von A und denen einer optimalen Offline-Lösung beträgt höchstens k%.“ Der Abstand k ist dann $c-1$, i. d. R. ausgedrückt in Prozent.

Hier laufen Theorie und Experiment Hand in Hand: die Menge der zu betrachtenden Instanzen kommt aus der Beobachtung; für jede Instanz ist das Zertifikat jedoch mathematisch rigoros.

5.4. **Parametrisierte Simulation.** Diese Evaluierungsmethode verlässt den theoretischen Rahmen: das Verhalten verschiedener Algorithmen wird unter verschiedenen Systemparametern im Computer-Experiment verglichen. Zwar kann man auch die Werte von Kostenfunktionen verfolgen, die große Stärke liegt aber in der Beobachtbarkeit des Gesamtverhaltens, das sich nicht immer auf einen eindimensionale Kostenfunktion projizieren lässt. Allerdings erfordert die experimentelle Analyse eine sorgfältige Auswahl der verwendeten Daten und Systemparameter (vgl. hierzu auch[12]).

Das Zertifikat kommt dementsprechend nur aus der Beobachtung und aus einer statistischen Auswertung; es ist experimentell. Durch eine ausführliche Betrachtung des Verhaltens in verschiedensten Systemkonfigurationen kann man aber Risiken besser abschätzen, als wenn man ausschließlich den Original-Systemzustand simuliert.

6. Die Verfahren in den Anwendungen

Hier beschreiben wir nun unsere Erfahrungen mit den oben vorgestellten Evaluierungsmethoden in den eingangs vorgestellten Fallbeispielen.

6.1. **Regalbediengerät.** Durch Weiterentwicklung der Offline-Optimierungsverfahren für asymmetrische Handlungsreisenden-Probleme konnte der Replan-Ansatz echtzeittauglich gemacht werden [2]. Das heißt, eine Neuplanung aller bekannten Ein- und Auslageraufträge zu einem Zeitpunkt mit minimaler Gesamt-Leerfahrzeit ist i. d. R. in einer Sekunde möglich.

Was hat das Steuerungsproblem für das Regalbediengerät mit dem Handlungsreisenden-Problem zu tun? Wir können uns alle Ein- und Auslageraufträge als Städte vorstellen. Die Fahrzeit von Stadt A nach Stadt B ist die Zeit, die das Regalbediengerät nach Fertigstellung von Auftrag A benötigt, um vom Endpunkt von Auftrag A zum Startpunkt von Auftrag B zu gelangen. Also ist es von A nach B i. d. R. nicht genau so weit wie von B nach A. Eine kürzeste Tour durch alle Städte ist nun genauso lang wie die Gesamt-Leerfahrtzeit bei der entsprechenden Abarbeitungsreihenfolge. Und praxistaugliche Verfahren für die

Bestimmung einer kürzesten Tour im sogenannten *asymmetrischen Handlungsreisenden-Problem (ATSP)* werden laufend weiterentwickelt.

Aber wie gut ist die Qualität der gelieferten Online-Lösung? Betrachten wir ein verwandtes Problem: Steuere so, dass die Fertigstellungszeit der Aufträge minimal ist. Der neue Zielfunktionswert ergibt sich aus dem alten durch Addition der Dauern der beladenen Bewegungen und der Standzeiten des Bediengeräts. Hier kann man folgendes beweisen:

Satz 1 ([3, 7])**.** *Der Online-Algorithmus Replan ist bzgl. der Minimierung der Fertigstellungszeit* 2.5*-kompetitiv.*

Leider überträgt sich dieses Ergebnis nicht auf die originale Zielfunktion, da das Addieren auch eines nicht vom Algorithmus abhängenden Summanden zur Zielfunktion das Kosten*verhältnis* sehr wohl beeinflusst.

Also reicht uns diese Aussage nicht für die Entscheidung, die Replan-Steuerung zu wählen. Eine A-posteriori-Analyse ergab folgendes Bild:

Beobachtung 1. *Auf Original-Instanzen von SNI beträgt die Optimalitätslücke zwischen Replan-Lösung und A-posteriori-Optimallösung zwischen* 10 *und* 70 *Prozent.*

Wir haben also eine bewiesene Gütegarantie nur für bereits beobachtete Instanzen; diese ist jedoch schärfer als die in der allgemeingültigen kompetitiven Analyse.

Eine zusätzliche parametrisierte Simulation zeigte, dass Replan im Wettbewerb mit anderen Algorithmen knapp die Nase vorn hat. Gegenüber der installierten Prioritätensteuerung sind die Verbesserungen sogar erheblich; diese war nur etwa so effizient wie die zufällige Auswahl des jeweils nächsten Auftrags.

Nach Störungen konnte Replan die anderen Verfahren ausstechen:

Beobachtung 2. *Nach Störungen hat Replan in Simulationsstudien auf realen Daten die deutlich kürzeste Gesamtleerfahrtstrecke.*

Wir sehen, dass die Anwendung mehrerer Evaluierungsverfahren ein schärferes Bild von der Qualität eines Online-Algorithmus zeichnet als nur eines allein. (Eine Binsenweisheit vielleicht, aber dennoch zu selten angewendet.)

SNI hat jedenfalls die Steuerung installiert und erzielte die vorausgesagten Verbesserungen.

6.2. **Kommissioniermobile.** In diesem Fall benötigten die Online-Algorithmen, die zu bestimmten Zeitpunkten Offline-Optimallösungen zur Planung verwenden, zu viel Rechenzeit: wegen der Echtzeit-Anforderungen kamen sie also nicht in Frage.

Gerade wenn man aus einer großen Menge von heuristisch arbeitenden Algorithmen einen aussuchen muss, benötigt man Entscheidungshilfe. In diesem Fall kann man mit der kompetitiven Analyse keine verwertbare Aussage erhalten. Schlimmer noch: für ein vereinfachtes Problem (siehe [14] für Details) kann man sogar *beweisen*, dass ein offensichtlich dummer Algorithmus einen besseren Kompetitivitätsfaktor hat als ein auf den ersten Blick vernünftiger Algorithmus.

Satz 2 ([14])**.** *Für das vereinfachte Kommissionierproblem mit einem Stopp pro Auftrag und dem Ziel der Minimierung der maximalen Anzahl von Stopps eines (individuellen) Mobils ist der dumme Algorithmus, der nur eines von q Mobilen benutzt,* $(2q-1)$*-kompetitiv, und das ist im Wesentlichen bestmöglich (*$q > 1$*):* jeder *Online-Algorithmus ist nicht besser als q-kompetitiv.*

Außerdem ist ein Greedy-Algorithmus, der bei jeder Zuordnung die Anzahl der Stopps eines Mobils am wenigsten erhöht, nicht besser als 2*q-kompetitiv.*

Experimentell ist der Greedy-Algorithmus in fast allen Fällen dem dummen Algorithmus weit überlegen.

Auch in der A-posteriori-Analyse kann kein brauchbares Zertifikat für einen Online-Algorithmus gefunden werden. Daher bleibt in diesem Fall nur die parametrisierte Simulation übrig. Und diese zeigt etwas Überraschendes:

Beobachtung 3. *Ändert man den Parameter „Anzahl der Mobile“ von acht auf sechs, so werden die Ergebnisse für einen greedyartigen Algorithmus nicht schlechter, eher besser.*

Diese Beobachtung hätten wir bei der ausschließlichen Untersuchung der Originalzielfunktion gar nicht machen können. Sie gilt aber wieder nur für alle in der Simulation verwendeten Eingabedaten.

Die Simulation und einige der Online-Heuristiken sind von Herlitz als Decision Support Tool installiert worden.

6.3. **Aufzüge.** Die vorausplanenden Online-Algorithmen Replan und Ignore können echtzeittauglich implementiert werden. Nun stellt sich noch die Frage nach der Güte im Vergleich zu „Nächster Nachbar“ oder FIFO. Eine kompetitive Analyse ergibt folgendes ernüchternde Resultat:

Satz 3 ([11])**.** *Für das Aufzugsteuerungsproblem mit dem Ziel „Minimiere die durchschnittliche/maximale“ Flusszeit existiert kein kompetitiver Algorithmus. Salopp ausgedrückt: alle Online-Algorithmen sind ∞-kompetitiv.*

Auch hier hängt das Ergebnis stark von der Zielfunktion ab. Will man statt der Flusszeiten die Fertigstellungszeiten minimieren, so kann man zeigen:

Satz 4 ([3])**.** *Replan und Ignore sind für das Aufzugsteuerungsproblem mit dem Ziel „Minimiere die Fertigstellungszeit“ 2.5-kompetitiv.*

Es gibt sogar einen von Ignore abgeleiteten 2-kompetitiven Algorithmus, und das ist bestmöglich.

Hier sind Replan und Ignore so zu verstehen, dass eine Neuplanung immer einen Plan mit minimaler Fertigstellungszeit erzeugt. Aber was bedeutet schon „Fertigstellungszeit“ in einem kontinuierlich arbeitenden System? Ferner zeigen Experimente, dass das Verhalten von Replan sich qualitativ stark von dem von Ignore unterscheidet. Das kommt in den Kompetitivitätsresultaten nicht zum Ausdruck.

Also müssen noch andere Evaluierungsmethoden geprüft werden. Und tatsächlich kann man rigoros beweisen, dass unter Δ-vertretbarer Belastung (nur Δ-vertretbare Auftragsmengen werden berücksichtigt) folgendes gilt:

Satz 5 ([11])**.** *Für das Aufzugsteuerungsproblem mit dem Ziel „Minimiere die durchschnittliche/maximale“ Flusszeit sind unter Δ-vertretbarer Belastung die Flusszeiten von Ignore (optimiert für jeden neuen Plan die Fertigstellungszeit) beschränkt durch 2Δ.*

Die Flusszeiten von Replan können trotz Δ-vertretbarer Belastung unbeschränkt sein.

Interessant ist, dass dieses Ergebnis sich auf allgemeinere Probleme überträgt: es gilt sowohl für Aufzüge, die für mehr als eine Palette Platz bieten als auch für Systeme aus mehreren Aufzügen.

Da auch diese Analyse auch den schlimmsten Fall abdeckt, ist das Ergebnis besonders für die maximale Flusszeit von Interesse. Und dort zeigte sich auch in Simulationsexperimenten ein deutlicher Vorteil von Ignore über Replan.

Beobachtung 4. *In Simulationsexperimenten ist die maximale Flusszeit von Ignore stabiler und im Durchschnitt signifikant niedriger als die von Replan.*

Die durchschnittliche Flusszeit von Ignore ist dafür etwas höher als die von Replan.

Man muss sich also vor der Entscheidung für einen der Algorithmen Gedanken machen, welchem Ziel man den Vorrang einräumt.

7. FAZIT

Online-Algorithmen spielen eine wichtige Rolle bei der effizienten Steuerung der innerbetrieblichen Logistik. Die Evaluierung solcher Algorithmen soll einerseits die begründete Auswahl einer Steuerung ermöglichen; andererseits sind Vorhersagen über die Leistungsdaten eines Steuer-Algorithmus auch für die Kapazitätsplanung von Belang. Die dargestellten Evaluierungsmethoden beleuchten diese Problematik von verschiedenen Seiten; jede Methode liefert in geeigneten Fällen hilfreiche Aussagen. Wichtige Entscheidungen werden sich jedoch auf ganze Bündel von Evaluierungsaussagen stützen müssen. Dies wird leider gelegentlich versäumt, da die einzelnen Verfahren mathematisch aus verschiedenen Disziplinen stammen. Wichtig ist es, das Zertifikat einer Evaluierungsmethode genau zu verstehen, um aus den Ergebnissen die richtigen Schlüsse zu ziehen.

LITERATUR

[1] N. Ascheuer. AMSEL—a modelling and simulation environment library. Online-Dokumentation `http://www.zib.de/ascheuer/AMSEL.html`.

[2] N. Ascheuer. *Hamiltonian Path Problems in the On-line Optimization of Flexible Manufacturing Systems*. Dissertation, Technische Universität Berlin, 1995.

[3] N. Ascheuer, S. O. Krumke, and J. Rambau. Online dial-a-ride problems: Minimizing the completion time. In *Proc. 17th Int. Symposium on Theoretical Aspects of Computer Science*, vol. 1770 of *Lecture Notes in Computer Science*, pages 639–650. Springer, 2000.

[4] M. Blom, S. O. Krumke, W. E. de Paepe, and L. Stougie. The online-TSP against fair adversaries. *Informs Journal on Computing*, 13(2):138–148, 2001.

[5] A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge Univ. Press, 1998.

[6] A. Borodin, S. Irani, P. Raghavan, and B. Schieber. Competitive paging with locality of reference. *Journal of Computer and System Sciences*, 50:244–258, 1995.

[7] E. Feuerstein and L. Stougie. On-line single server dial-a-ride problems. *Theoretical Computer Science*, 2002.

[8] A. Fiat and G. J. Woeginger, Hrsg. *Online Algorithms: The State of the Art*, vol. 1442 of *Lecture Notes in Computer Science*. Springer, 1998.

[9] M. Grötschel, S. O. Krumke, and J. Rambau. Wo bleibt der Aufzug? *OR News*, (5):11–13, March 1999.

[10] M. Grötschel, S. O. Krumke, and J. Rambau, Hrsg. *Online Optimization of Large Scale Systems*. Springer, Berlin Heidelberg New York, 2001.

[11] D. Hauptmeier, S. O. Krumke, and J. Rambau. The online dial-a-ride problem under reasonable load. *Theoretical Computer Science*, 2002.

[12] D. S. Johnson. A theoretician's guide to the experimental analysis of algorithms. Preliminary partial draft available at URL: `http://www.research.att.com/~dsj/papers.html`, 2001.

[13] E. Koutsoupias and C. Papadimitriou. Beyond competitive analysis. In *Proc. 35th Annual IEEE Symp. on the Foundations of Computer Science*, pages 394–400, 1994.

[14] S. O. Krumke, W. E. de Paepe, L. Stougie, and J. Rambau. Online bincoloring. In *Proc. 9th Annual European Symposium on Algorithms*, vol. 2161 of *Lecture Notes in Computer Science*, pages 74–, 2001.

[15] A. Kuhn, A. Reinhardt, and H.-P. Wiendahl, Hrsg. *Handbuch Simulationsanwendungen in Produktion und Logistik*. Fortschritte in der Simulationstechnik. Friedrich Vieweg & Sohn, Braunschweig/Wiesbaden, 1993.

[16] C. Phillips, C. Stein, E. Torng, and J. Wein. Optimal time-critical scheduling via resource augmentation. In *Proceedings of the 29th Annual ACM Symposium on the Theory of Computing*, pages 140–149, 1997.

[17] H.-J. Siegert. *Simulation zeitdiskreter Systeme*. Oldenbourg, München, Wien, 1991.

[18] D. D. Sleator and R. E. Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28(2):202–208, 1985.

SVEN O. KRUMKE, KONRAD-ZUSE-ZENTRUM FÜR INFORMATIONSTECHNIK, TAKUSTR. 7, 14195 BERLIN, GERMANY

E-mail address: `krumke@zib.de`

JÖRG RAMBAU, KONRAD-ZUSE-ZENTRUM FÜR INFORMATIONSTECHNIK, TAKUSTR. 7, 14195 BERLIN, GERMANY

E-mail address: `rambau@zib.de`