Eidgenössische Technische Hochschule Zürich
Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Swiss Federal Institute of Technology Zurich

Diplomarbeit DA-2003.29

Optimierungsmethoden für einen personalisierten Informationsservice

Sommersemester 2003
10. Juli 2003

Jun Ma

Supervisors: Prof. Dr. Thomas Erlebach
Dr. Lukas Finschi
Thomas Meyer

Institut für Technische Informatik und Kommunikationsnetze
Computer Engineering and Networks Laboratory

Vorwort

Die vorliegende Arbeit entstand im Rahmen einer Diplomarbeit am Institut für Technische Informatik (TIK) innerhalb des Departements Elektrotechnik und Informationstechnologie der ETH Zürich in Kooperation mit der R&D-TM der Schindler Aufzüge AG. Die Arbeit geht um "Optimierungsmethoden für den personalisierten Informationsservice".

Bei der Schindler Aufzüge AG habe ich ein optimales Arbeitsumfeld vorgefunden. Zudem hat es mich sehr motiviert, dass für den Informationsservice ein echtes Bedürfnis vorhanden ist, insbesondere auch, dass für den Informationsservice sehr wahrscheinlich in meiner Heimat China eingesetzt werden kann.

Diese Diplomarbeit ist das Resultat beispielhafter Zusammenarbeit der ETH Zürich und der Schindler Aufzüge AG. Für das Zustandekommen der Zusammenarbeit innerhalb der Diplomarbeit möchte ich mich vor allem Prof. Dr. Thomas Erlebach, Stamatis Stefanakos, Danica Vukadinovic von TIK, ETH Zürich, Dr. Lukas Finschi und Thomas Meyer von Schindler Aufzüge AG herzlich bedanken. Ich möchte auch dem ganzen R&D-TM Team der Schindler Aufzüge AG für das angenehme und freundschaftliche Arbeitsklima danken.

Mein Dank geht auch an Dr. Christian Bodmer, welcher mir die Gelegenheit zum Schreiben dieser Arbeit bei Schindler Aufzüge AG gegeben hat.

Zürich, 10. Juli 2003

Jun MA < *ma@ee.ethz.ch* >

Departement für Elektrotechnik
und Informationstechnologie
ETH Zürich

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einführung

1.1 Hintergrund

Unser Leben ist heutzutage von einer riesigen Informationsflut geprägt. Täglich gilt es viele Informationen zu verarbeiten. Die Schwierigkeit besteht darin wichtige Informationen von unwichtigen zu trennen, damit zur richtigen Zeit die nötigen Informationen für bedeutende Entscheidungen zur Verfügung stehen. Es gibt verschiedene Ansätze und Ideen um diese Schwierigkeit zu überwinden.

Die neuste Aufzugsgeneration der Schindler Aufzüge AG[1] verfügt über ein System, das Passagiere zu identifizieren vermag und dementsprechend einen personalisierten Informations- und Unterhaltungsservice anbietet. Das ist eine bedeutende Revolution, da der neuer Aufzug nicht mehr nur ein Transportwerkzeug ist, sondern auch ein Informationszentrum ist. Zum Beispiel kann der Passagier Informationen über den Ort, die Zeit und Temperatur etc. erhalten. Individuell werden den Passagieren auf dem Display interessante Dokumente oder News nach ihrem Geschmack angeboten. Ihre persönliche Interessen werden aufgrund von dem Touchscreen durch einen intelligenten Algorithmus automatisch erlernt. So kann der Aufzugbesitzer gezielt Werbungen machen und eventuell davon profitieren.

Da die Zahl der Passagiere und Dokumente sehr gross sein können, benötigt es unbedingt eine Optimierungsmethode. Diese Diplomarbeit untersucht deshalb die verschiedene Methoden, um das Optimieren zu realisieren.

[1] http://www.schindler.ch

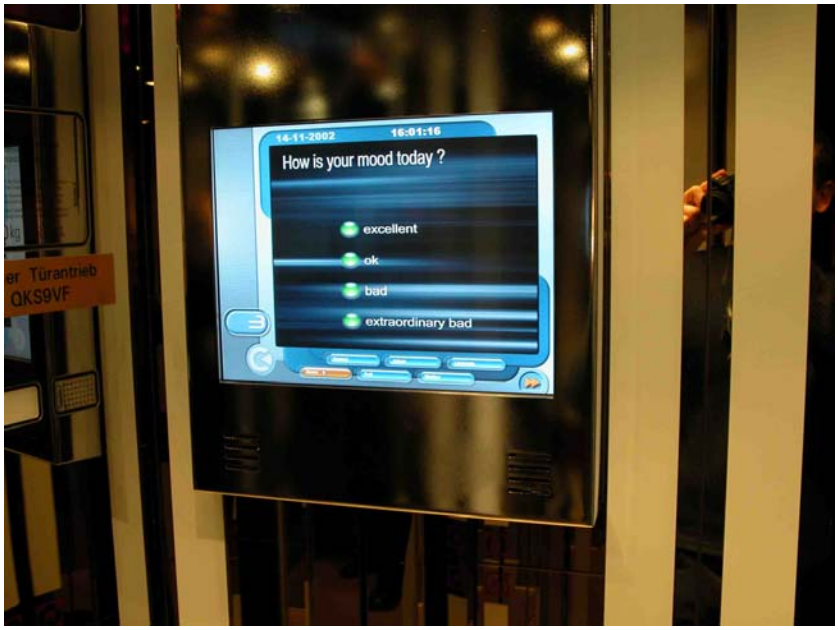

Abbildung 1.1: Servicescreen

1.2 Anwendungsbeispiele

Der Informations- und Unterhaltungsservice kann überall eingesetzt werden. Zum Beispiel bei einer grosser Firma hat jeder Mitarbeiter ein Benutzerprofil, das sowohl "demographische" als auch "psychographische" Merkmale enthält. Ein Profil kann sich auch auf eine ganze Gruppe oder Abteilung beziehen. So identifiziert der Service vor dem Aufzug jeden Passagier mittels einer Chipkarte und präsentiert die möglichst seinen Interessen entsprechende Dokumente. Der Service kann auch wichtige Informationen über die Firma beziehungsweise die Abteilung präsentieren.

In der Öffentlichkeit kann der Service aber auch ohne den Passagier zu identifizieren die nötigen Informationen anbieten. In einem Supermarkt wie Migros-city wird der Service je nach der Warentypen die Dokumente anzeigen. Zum Beispiel wenn der Passagier sich in Abteilung für Kleidung befindet, sollt der Service die Kleidungwerbungen anzeigen. In diesem Fall kann der Service auch je nach Zeit funktionieren: am Abend wäre es sicher schön, wenn man das Menu vom Abendessen im Migros Restaurant schon vor dem Aufzug wissen könnte.

Eine andere Anwendung ist, dass in Flughäfen alle Passagiere das Wetter oder Informationen des Zielorts vorzeitig erfahren werden.

Abbildung 1.2 zeigt beispielweise ein Servicescreen.

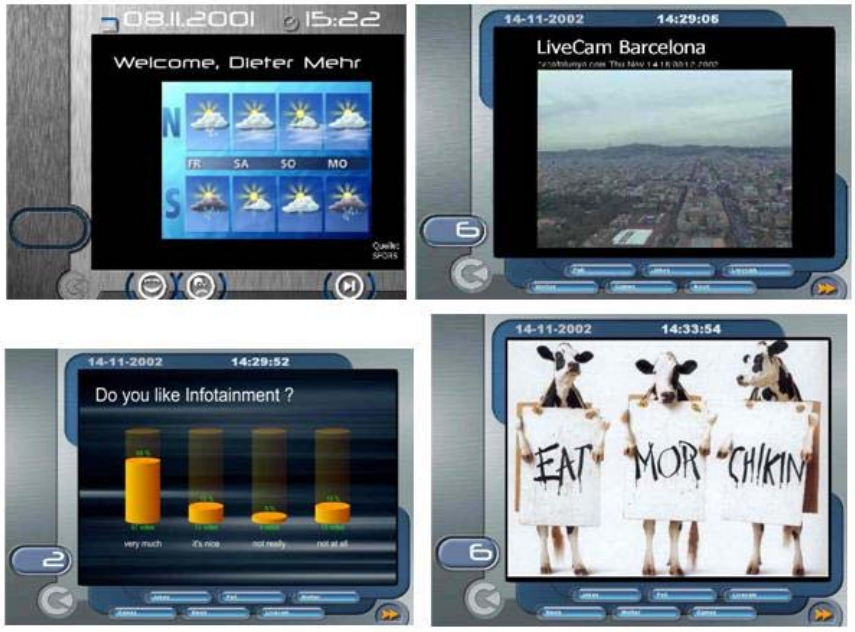

Abbildung 1.2: Dokumentbeispiele

Kapitel 2

Modellierung

Da es in dieser Arbeit um Optimierungsmethoden geht, sollen wir bevor die Arbeit überhaupt anfangen zuerst etwas über Optimierung sagen. Was ist das Optimierungsproblem bei uns? Unser Optimierungsproblem lautet:

- Der Werber will eine bestimmte Menge Werbungen nur den richtigen Personen anzeigen.
- Der Passagier will nur seinen Interessen entsprechende Dokumente anschauen.
- Der Aufzugbesitzer will möglichst grossen Gewinn durch Anzeige des Dokuments haben.

Das ist ein typisches Optimierungsproblem, müssen wir darum eine gute Optimierungsmethode oder einen Kompromiss herausfinden, die möglichst alle diese Bedürfnisse in einer kurzer Zeit erfüllen kann.

Es steht einige bekannte mathematische Methoden zum Lösen des Optimierungsproblems zur Verfügung, die sind:

- Lineare Programmierung (LP)
- Nichtlineare Programmierung (NLP)
- Integer Programmierung (IP)
- Heuristische Methode
- Kombinatorische Optimierung
- .....

Wie wir unser Problem mathematisch darstellen sollen, wird durch Modellierung individuell bestimmt.

Wir modellieren nun die Aufgabe in einer Zeitperiode. Wir betrachten:

- Ein Set von Personen P_0, P_1,...,P_{M-1} mit der relativen Häufigkeitsverteilung p_0, p_1, ..., p_{M-1}, $0 \leq p_i \leq 1$, so dass gilt:

  $$\sum_i p_i = 1$$

  Die relative Häufigkeitsverteilung p_i kann so vorgestellt werden: wenn ein Person in der betrachteten Zeit 10 Mal in den Aufzug eingestiegen ist und der Aufzug total 100 Passagiere hat, ist p_i gleich 0.1.

- Ein Set von Dokumenten D_0, D_1,..., D_{N-1}. Wir nehmen an, dass man durch die Beobachtung der Restriktionen jedem Dokument eine Gültigkeitswahrscheinlichkeit d_0, d_1,...,d_{N-1}, $0 \leq d_j \leq 1$ zuordnen kann. Ein sehr bekanntes Beispiel der Wahrscheinlichkeit ist:

  A = {1,2,3,4,5,6 }= mögliche gewürfelte Augenzahl bei einem Würfel, ist dann die Wahrscheinlichkeit für jede einelementige Teilmenge X von A gleich 1/6.

  In dieser Arbeit wird die Gültigkeitswahrscheinlichkeit von Dokument je nach der Modellierung individuell definiert.

- Jedes Dokument kann Restriktionen bezüglich des Kontextes besitzen. Solche Restriktionen können beispielweise zeitabhängig, ortabhängig oder ganz allgemein kontextbasiert sein. So wäre denkbar, dass eine Coca-Cola Werbung nur dann angezeigt werden soll, wenn die Aussentemperatur mehr als 25 Grad beträgt, oder dass eine Kinowerbung nur gegen Abend präsentiert werden soll. Ein kontextungültiges Dokument verschwendet nicht nur die Ressource, es bringt auch keinen Gewinn. Der Einfachheit halber nehmen wir an, dass es für jeden Kontext mindestens ein gültiges Dokument gibt.

- Eine Gesamtanzeigekapazität von K, das heisst, in der betrachteten Zeitperiode können K Dokumente angezeigt werden.

- Manchmal ist es bei einer Werbekampagne wünschenswert, dass eine minimale Anzahl von Anzeigen pro Zeiteinheit garantiert werden kann. So ist beispielweise möglich, dass eine Coca-Cola Werbung pro

Woche mindestens 1000 Mal angezeigt werden muss. Darum hat jedes Dokument eine minimal vorgeschriebene Anzahl von Anzeigen pro Zeitperiode k_0, k_1,..., k_{N-1}, wobei

$$\sum_j k_j \leq K$$

Wir definieren:

$$q_j := \frac{k_j}{K}$$

damit ist

$$\sum_j q_j \leq 1$$

$$0 \leq q_j \leq 1$$

Selbstverständlich kann niemand dafür garantieren, dass die Temperatur in der betrachteten Zeitperiode jemals über 25 Grad steigen wird, und deshalb kann es keine absolute Garantie für die Erfüllung der Nebenbedingung der minimalen Anzeigen geben.

- Eine Scorefunktion, die misst, wie gut ein Dokument auf ein Profil passt:

$$f : (P, D) \rightarrow f(P, D) \in [0, \sqrt{2}]$$

In der Praxis kann man sich Scorefunktion als der Gewinn für den Aufzugbesitzer vorstellen. In der betrachteten Zeitperiode muss die Scorefuntion immer konstant sein.

Kapitel 3

Lösungsverfahren

Der nächste Schritt besteht darin, ein Lösungsverfahren nach der Modellierung herauszufinden. Wir müssen jedoch zuerst vorstellen, wie der Service funktioniert: ein Passagier P_i steht vor dem Aufzug. Der Service identifiziert den Passagier mittels einer Chipkarte und presentiert ein Dokument D_j für ihn. Dann können wir unser Ziel formulieren. Das Ziel ist mit gezieltem Dokument im Aufzug möglichst grosser Gewinn zu bekommen. In Kapitel 2 haben wir für den Gewinn eine Scorefunktion definiert, die misst, wie gut ein Dokument auf ein Passagier passt. So können wir unser Ziel mathematisch ganz allgemein so beschreiben:

$$max \sum_{l=1}^{K} f(P_i, D_j) \tag{3.1}$$

$f(P_i, D_j)$ bedeutet hier den Gewinn wenn wir für Person P_i Dokument D_j angezeigt haben. $\sum_{l=1}^{K} f(P_i, D_j)$ bedeutet denn Gewinn durch die Anzeige von Dokument für die gesamte K Passagiere in der betrachteten Zeitperiode. Da unser Ziel möglichst grosser Gewinn zu bekommen ist, ist das Ziel nichts anderes als $max \sum_{l=1}^{K} f(P_i, D_j)$.

Um Gleichung 3.1 zu realisieren müssen wir eine gute Optimierungsmethode herausfinden. Eine gute Optimierungsmethode ist mindestens effizient und einfach implementierbar. Wir messen hier die Effizienz unmittelbar an der Laufzeit, weil für riesig grosse Anzahl Dokumente und Profile die schnelle Laufzeit klar höchste Priorität besitzt. Wegen der beschränkten Zeit für das Projekt ist die einfache Implementierung der versuchter Methode auch ein wichtiger Faktor.

3.1 Linear Progamming basierte Methoden

3.1.1 Random Repeat

Als erste Versuch wollen wir nun eine sogenannte Methode "Random Repeat" untersuchen. Wir können diese Methode so erklären: wenn ein Passagier kommt, wählt der Service ein beliebiges Dokument von alle Dokumente aus und prüft ob dieses Dokument kontextgültig ist. Falls es kontextgültig ist, wird es angezeigt, sonst muss ein beliebiges Dokument nochmal ausgewählt werden. Der Prozess wiederholt sich solange, bis ein kontextgültiges Dokument gefunden wird.

Wir wissen, ein möglichst grosser Gewinn bedeutet, dass man das Dokument möglichst den Interessen des Passagiers entsprechen lassen muss. Wie bereits erwähnt, die relative Häufigkeitsverteilung von Person P_i ist p_i. Die Gültigkeitswahrscheinlichkeit von Dokument D_j ist d_j. d_j kann man mit der Hilfe von der Abbildung 3.1 nachvollziehen.

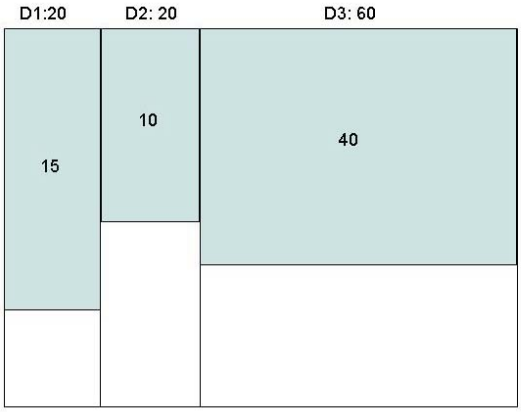

Abbildung 3.1: Gültigkeitswahrscheinlichkeit von Dokument

Es gibt in diesem Beispiel 3 Dokumente D_1, D_2, D_3, die jeweils 20, 20, 60 Mal ausgewählt aber nur 15, 10, 40 Mal angezeigt werden. Dann ist d_1, d_2, d_3 gleich 0.75, 0.5 und 0.66.

Wir definieren nun eine neue Variable X_{ij}, X_{ij} sei die Wahrscheinlichkeit, dass das Dokument D_j ausgewählt wird, wenn der Passagier P_i erscheint,

	X00	X01	X02	X03	X04	X05
P0	0.1	0.25	0.35	0.05	0.1	0.15

Tabelle 3.1: Beispiel X_{0j}

wobei

$$\forall i : \sum_j X_{ij} = 1 \quad mit \quad 0 \leq X_{ij} \leq 1 \tag{3.2}$$

Tabelle 3.1 zeigt ein kleines Beispiel mit 6 Dokumente für X_{0j} wenn Person P_0 kommt.

Wir können dann die Zielfunktion so beschreiben:

$$max \sum_{i,j} f(P_i, D_j) \cdot p_i \cdot X_{ij} \cdot d_j \tag{3.3}$$

Um ein beliebiges Dokument auszuwählen, generiert Methode "Random Repeat" eine zufällige Zahl a zwischen 0 und 1 wenn Person P_i kommt. Falls a in Intervall zwischen $\sum_{j=0}^{m} X_{ij}$ und $\sum_{j=0}^{m+1} X_{ij}$ liegt, dann wird Dokument D_{m+1} ausgewählt. Mit Hilfe von Tabelle 3.1 können wir hier auch ein Beispiel machen, z.B. a ist gleich 0.33, dann wird Dokument D_1 ausgewählt, weil $X_{00} < 0.33 < X_{00} + X_{01}$.

Wir beschreiben "Random Repeat" mit folgendem Algorithmus:

Annahme: Passagir P_i im Aufzug

repeat

generiere eine zufällige Zahl zwischen 0 und 1, und wähle gemäss X_{ij} aus Dokument D_j aus,

falls D_j kontextgültig, break

end

Für diese Methode leiten wir die Nebenbedingungen eines Optimierungs-Problems her:

Wir wissen, dass Dokument D_j mit Wahrscheinlichkeit $\sum_i p_i \cdot X_{ij} \cdot d_j$ für irgend einen Passagier P_i angezeigt wird. Falls in einer Zeitperiode total $\hat{K}$ Mal ausgewählt wird, wird D_j im Erwartungswert $\hat{K} \cdot \sum_i p_i \cdot X_{ij} \cdot d_j$ Mal angezeigt. Dies muss grösser oder gleich als die minimal vorgeschriebene Anzahl k_j sein, also,

$$\forall j : \hat{K} \cdot \sum_i p_i \cdot X_{ij} \cdot d_j \geq k_j \tag{3.4}$$

Diese Ungleichung gilt natürlich für alle Dokumente.

Nun ist $\hat{K} \cdot \sum_i p_i \cdot X_{ij} \cdot d_j$ die Anzahl von Anzeigen für ein bestimmtes Dokument, dann muss die Summe aller angezeigte Dokumente $\hat{K} \cdot \sum_{ij} p_i \cdot X_{ij} \cdot d_j$ gleich die Gesamtkapazität K sein, so gilt:

$$\hat{K} \cdot \sum_{ij} p_i \cdot X_{ij} \cdot d_j = K \tag{3.5}$$

Wir formen die Gleichung 3.5 so um,

$$\hat{K} = \frac{K}{\sum_{ij} p_i \cdot X_{ij} \cdot d_j} \tag{3.6}$$

dann setzen Gleichung 3.6 in 3.4 ein,

$$\forall j : \sum_i p_i \cdot X_{ij} \cdot d_j \geq q_j \sum_{i,j} p_i \cdot X_{ij} \cdot d_j \tag{3.7}$$

mit $q_j := \frac{k_j}{K}$.

Die obenstehende Ungleichung (3.7) wird automatisch auf gleich gesetzt, falls die Gesamtkapazität K gleich die Summe der minimal vorgeschriebene Anzahl von einzeln Anzeigen ist ($\sum_j k_j = K$).

Aber wir müssen hier aufmerksam darauf machen, dass wegen der zufälligen Auswahl diese Methode sicher nicht garantiert, dass alle Dokumente die minimal vorgeschriebene Anzahl Anzeigen erreichen.

Das System mit Zielfunktion 3.3 und Gleichung 3.2 und 3.7 als Nebenbedingung hat gerade die Form einer Linearen Programming. Diese Eigenschaft lässt uns selbstverständlich die Variable X_{ij} mit Linear Programming bestimmen.

Linear Programming betrifft das Problem des Findens eines Vektors x, der eine gegebene lineare Funktion $c^T x$ maximiert, wobei x die lineare Ungleichung $Ax \leq b$ erfüllt:

$$max\ c^T x$$

subject to:

$$Ax \leq b$$

So möchte man ein x mit dem grössten Wert $c^T x$ finden.

Für das LP sind einige schnelle Lösungsmethoden entwickelt worden. Die vorstehendste von diesen ist die Simplex Methode, die von Dantzig in den späten vierziger Jahren entworfen wird. 1979 bestätigt Khachiyan, dass ein LP in der polynomialer Zeit mit einer Variante der "Ellipsoidmethode" für die nichtlineare Programmierung gelöst werden kann. Die Simplex Methode ist momentan einer der beliebtesten Methoden zur Lösung des Linear Programming Problems in der Praxis.

Glücklichweise gibt es einige freie Java Programme auf Internet für die Simplex Methode. In dieser Arbeit wird eins[1] heruntergeladen und in meinem Programm eingesetzt. Dieses Programm fiel jedoch unglücklichweise aus, numerisch instabil zu sein in der Simulation, falls es riesige Anzahl Personen und Dokumente gibt.

Um das numerische Problem endgültig zu beheben, wird ein neuer Solver in meiner Arbeit eingesetzt, nämlich ILOG CPLEX[2]. ILOG CPLEX ist ein Produkt von der Firma ILOG. ILOG CPLEX liefert die leistungsstarken, robusten, flexiblen Optimierer für das Lösen Linear Programming. ILOG CPLEX kann die Probleme mit Millionen Begrenzungen und Variablen sehr gut lösen. Wir können denn hier ILOG CPLEX vertrauen.

ILOG CPLEX ist wegen License nur an der ETH Zürich zum Studium- und Forschungszweck installiert, muss dashalb alle meine Arbeit mit ILOG CPLEX an der ETH Zürich erledigt werden.

Die Zielfunktion und Nebenbedingungen von Linear Programm müssen zuerst in ein Textfile z.B. x.lp geschrieben werden, dann liest ILOG CPLEX das x.lp mit Kommando "read x.lp" und rechnet die Lösung von X_{ij} mit Kommando "primopt"[3], zum Schluss schreiben ILOG CPLEX X_{ij} mit Kommando "display solution var 1- " wieder in einem anderem Textfile Resultat.lp auf.

File x.lp beispielsweise lautet wie folgend:

Maximize

$$0.0462X_{00} + 0.0093X_{01} + 0.0383X_{10} + 0.0059X_{11}$$

[1] http://www.cs.wustl.edu/∼ javagrp/help/LinearProgramming.html
[2] http://www.ilog.com
[3] Primopt verwendet die Methode "Primal Simplex" zur Lösung des LP

Subject to

$X_{00} + X_{01} = 1$

$X_{10} + X_{11} = 1$

Bounds

$0 <= X_{00} \leq 1$

$0 <= X_{01} \leq 1$

$0 <= X_{10} \leq 1$

$0 <= X_{11} \leq 1$

end

Resultat.lp lautet beispielweise wie folgend:

Maximize = 0.031183

$X_{00} = 0.430724$

$X_{01} = 0.569276$

$X_{10} = 0.000000$

$X_{11} = 1.000000$

Um den ganzen Prozess zu vereinfachen, hat Prof. Erlebach ein C Programm geschrieben, das x.lp lesen, rechnen und Resultat.lp aufschreiben kann und meine Arbeit sehr erleichtert hat.

Wir verwenden CPLEX, um die Variable X_{ij} von dem Linear Programming zu lösen. Wie bereits erklärt, wählen wir nach dem Wert von X_{ij} ein Dokument aus und prüfen, ob das Dokument kontextgültig ist. Durch das Ergebnis von Simulation (siehe kapitel 5) haben wir die Vermutung bewiesen, dass Methode “Random Repeat” nicht garantieren kann, für alle Dokumente die minimal vorgeschriebene Anzahl zu erreichen, selbst wenn keine kontextbedingten Restriktionen vorhanden sind.

3.1.2 Random Counted

Das Ergebnis von "Random Repeat" zwingt uns, einen Zähler einbauen zu müssen oder mit anderen Worten, wir sollen besser die sogannante Methode "Random Counted" nehmen. Methode "Random Counted" ist eine Weiterentwicklung von Methode "Random Repeat". Die funktioniert so: der Service wählt für jeden Passagier ein beliebiges Dokument aus. Falls es kontextgültig ist, wird es angezeigt. Sonst wählt sie wieder zufällig ein Dokument aus, bis dieses kontextgültig ist, und zählt, wie oft es in der Vergangenheit schon angezeigt wurde. Wenn ein Dokument seine minimal vorgeschriebene Anzahl k_j schon erreicht hat, darf es nicht mehr angezeigt werden. Genau wie in Methode "Random Repeat" muss hier auch Variable X_{ij} definiert werden, um ein Dokument nach X_{ij} auswählen zu können. Die Zielfunktion 3.3, Nebenbedingungen 3.2 und 3.7 bleiben unverändert, so kann man auch mit Linear Programming die X_{ij} bestimmen.

Der Algorithmus "Random Counted" lautet:

Für alle j setze $Z_j := 0$

Annahme: Passagir P_i im Aufzug

repeat

generiere eine zufällige Zahl zwischen 0 und 1, wähle gemäss X_{ij} aus Dokument D_j aus,

falls D_j kontextgültig und k_j nicht erreicht, $Z_j := Z_j + 1$ und break

end

Manchmal ist es aber möglich, dass für einen Passagier alle kontextgültigen Dokumente schon die k_j erreicht haben, dann muss trotzdem ein kontextgültiges Dokument angezeigt werden; falls $\sum_j k_j = K$, werden in diesem Fall einige Dokumente ihre k_j nicht erreichen können.

Wenn die Gesamtkapazität K grösser als die Summe der minimal vorgeschriebene Anzahl von einzeln Anzeigen ist, erreicht irgendwann alle Dokumente die minimal vorgeschriebene Anzahl. Dann muss beispielweise für die restliche Anzeige dasjenige Dokument, das mehren Gewinn bringt nämlich die grösste Scorefunktion besitzt, ausgewählt werden.

3.2 Heuristische Methoden

Alle oben untersuchte Verfahren sind LP-basierte Verfahren, sie haben generell folgende Eigenschaft:

- Eine Vorkenntnis über LP wissen muss.
- hohe Rechenzeit, da es Vorberechnen von X_{ij} nötig ist.
- hohe Platzbebarf, da es zusätzliche Files generiert.

Dann sind alternative "Heuristische Mothode " zu LP-basierte Verfahren nötig. Heuristische Mothode hat folgende Eigenschaften:

- Griechisch heuriskein: Finden, entdecken
- Eine Technik zur Suche von guten (nahezu optimalen) Lösungen für ein Optimierungsproblem in möglichst kurzer Zeit
- Ohne Gültigkeit oder Optimalität zu garantieren
- In vielen Fällen wird nicht mal eine Aussage getroffen, wie nahe die gefundene Lösung am Optimum liegt

Wir untersuchen die folgende heuristische Verfahren.

3.2.1 Greedy Counted

Ein normaler Algorithmus ist ein schrittweises Rezept für das Lösen eines Problems. Ein Greedy[4] Algorithmus ist ein Algorithmus, der alle seine Lieblinge zuerst verschlingt. Es konnte auch "single-minded" genannt werden. Die Idee hinter einem gierigen Algorithmus ist, ein einzelnes Verfahren im Rezept immer wieder durchzuführen, bis sie nicht mehr erfolgt sein und sehen, was Art von Resultaten sie produziert. Sie kann möglicherweise nicht das Problem vollständig lösen, oder, wenn es eine Lösung produziert, ist sie nicht sehr gut, aber es ist eine Methode zum Lösen des Problems und erbringt manchmal sehr gute (oder sogar die besten) Resultate.

Wir haben ein Greedy Algorithmus gefunden, das "Greedy Counted" heisst. Methode "Greedy Counted" wählt dasjenige Dokument aus, das momentan die grösste Scorefunktion $f(P_i, D_j)$ besitzt, und zählt die bereits angezeigte Anzahl. Falls dieses Dokument k_j nicht erreicht und kontextgültig

[4] greedy: gierig

ist, wird es angezeigt, sonst wird das Verfahren mit dem Dokument wiederholt, welches das nächstkleinere $f(P_i, D_j)$ besitzt. Der Algorithmus "Greedy Counted" lautet:

Für alle j setze $Z_j := 0$

Annahme: Passagir P_i im Aufzug

Wähle D_j mit grösstem $f(P_i, D_j)$, so dass D_j kontextgültig und k_j noch nicht erreicht

$Z_j := Z_j + 1$

Diese Methode maximiert die Scorefunktion und sollte im Prinzip alle Dokumente ihre k_j erreichen lassen. Aber gibt es auch 2 Nachteile.

- Der Passagier muss immer dasselbe Dokument sehen solange die minimale vorgeschriebene Anzahl k_j nicht erreicht und er kann auch nie dasjenige Dokument sehen, das relative kleine Scorefunktion besitzt. Das ist sicher nicht ein Wunsch vom Werber.
- Manchmal ist es hier auch möglich, dass für einen Passagier alle kontextgültigen Dokumente schon die k_j erreicht, dann muss trotzdem ein kontextgültiges Dokument angezeigt werden; falls $\sum_j k_j = K$, werden in diesem Fall nicht alle k_j erreicht.

3.2.2 Greedy Discounted

Eine andere Greedy Methode heisst "Greedy Discounted". Die wählt dasjenige Dokument, das momentan die grösste Scorefunktion $f(P_i, D_j)$ besitzt, aus. Falls es kontextgültig ist, wird es angezeigt und $f(P_i, D_j)$ wird nach einem Kriterium ändern. Wir beschreiben "Greedy Discounted" mit folgendem Algorithmus:

Annahme: Passagir P_i im Aufzug

repeat

wähle nach momentan grösste $f(P_i, D_j)$

falls kontextgültig, ändere $f(P_i, D_j)$ und break

end

Scorefunktion zu ändern macht es möglich, dass kein Dokument immer angezeigt werden darf und dadurch alle Dokument ihre k_j mehr oder wenig erreichen können. Es gibt hier aber eine Schwierigkeit, nämlich wie man die Scorefunktion reduzieren muss, damit der Gewinn maximiert wird und alle k_j möglichst erreicht werden. Wir versuchen die neue Scorefunktion $F(P_i, D_j)$ so definieren:

$$F(P_i, D_j) = \frac{f(P_i, D_j) * (k_j - h_j)}{d_j * (K - \hat{K})} \tag{3.8}$$

wobei:

- Die Scorefunktion: $f(P_i, D_j)$;
- Die minimal vorgeschriebene Anzahl: k_j
- Die schon angezeigte Anzahl von Dokument D_j: h_j;
- Die Gesamtanzeigekapazität: K;
- Die schon angezeigte gesamte Anzeige: $\hat{K}$;
- Die Gültigkeitswahrscheinlichkeit von Dokument D_j: d_j.

Gleichung 3.8 definiert eine neue Scorefunktion, die sich dynamisch verhält. d_j im Nenner garantiert die kleine Gültigkeitswahrscheinlichkeit besitzende Dokument angeziegt werden kann. $\frac{(k_j - h_j)}{(K - \hat{K})}$ ermöglicht die Anzeige des Dokuments immer nach aktuelle Zustand durchzuführen.

3.3 Verallgemeinerung des Problems

Was wir bisher untersuchte sind eigentlich nur ein spezieller Fall, da in der Praxis es immer mehre Personen im Aufzug gibt. In dieser Fall ist vielleicht eine heuristische Methode günstig, das Problem zu lösen. Wir verwenden weiterhin “Greedy Counted”, aber mit leichter Änderung:

Wir suchen zuerst diejenige Dokumente, die alle Passagiere gerne sehen wollen und prüfen ob diese ihre k_j erreicht haben. Diejenige Dokumente, die k_j schon erreicht haben, werden aussortiert. Wir rechnen dann die durchschnittliche Scorefunktion für die bleibenden Dokumente und zeigt die grösste Scorefunktion besitzende Dokument an.

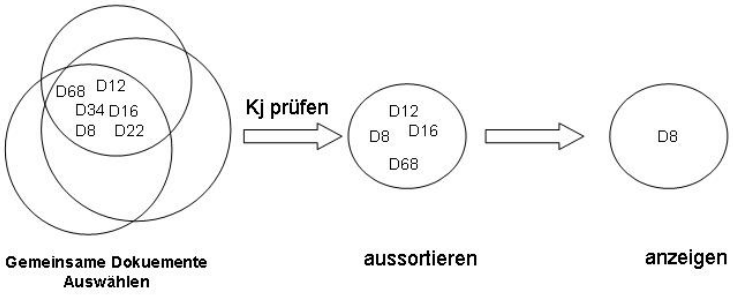

Abbildung 3.2: Verallgemeinerung des Problems

Abbilgung 3.2 zeigt uns graphisch ein Beispiel von dieser Methode: 3 Passagiere sind nun im Aufzug und Dokumente D_8, D_{12}, D_{16}, D_{22}, D_{34}, D_{68} sind die Interessen von aller Passagiere. Wir prüfen jede k_j und sortieren diejenige Dokumente aus, die k_j schon erreicht haben. Es bleibt nur noch D_8, D_{12}, D_{16}, D_{68}. Wir rechnen dann durchschnittliche $f(P_i, D_j)$ für jeden Passagir wenn ein bleibendes Dokument ausgewählt wird. Zum Schluss zeigen wir D_8 in diesem Beispiel an, weil $f(P_i, D_8)$ grösste Scorefunktion hat.

Wir müssen hier aufmerksam darauf machen, dass die Anzahl angezeigte Dokumente gleich die Anzahl Personen im Aufzug ist. Z.B. wenn 3 Personen im Aufzug ein Dokument D_j gesehen hätten, muss man so vorstellen, dass D_j schon 3 Mal angezeigt werden.

Kapitel 4

Programmaufbau

Der wichtiger Teil in dieser Arbeit ist Implementieren des untersuchte Verfahrens in Java (jdk1.3.1). Es steht eine Java Interface als Schnittstelle von Schindler Aufzüge AG zur Verfügung. Ausserdem werden noch einige Java Class für verschiedenen Zweck implementiert.

4.1 TestFileGenerator

Um das Problem weitgehend zu berücksichtigen, sind zahlreiche Versuche erwünscht. Class TestFileGenerator dient zur Erzeugung der Instanzen und wird nur aufgerufen solange eine neue Instanz verlangt. Die alle Parameters wie "xcoord","nofDisplays", "restrictions" oder "place" von der 3 Instanzen sind zufällig generiert. Die relative Häufigkeit von Personen sind auch zufällig verteilt. Die Instanz erfüllt $\sum_j k_j = K$, aber garantiert nicht alle Nebenbedingungen zu erfüllen.

Die Instanzen für die Personen lautet:

personID;Xcoord;Ycoord

person0; 0.27; 0.17

person1; 0.93; 0.37

person2; 0.36; 0.78

Die Instanzen für die Dokumente lautet:

documentID;xcoord;ycoord;nofDisplays;restrictions*

document0;	0.87;	0.66;	305;	place=[b,f,e,i,j,k]; time=[6,13]
document1;	0.76;	0.04;	103;	temperature=[-9,-2]
document2;	0.75;	0.66;	330;	place=[m,d,l,g,f,e]

Die Instanzen für Simulation lautet:

person;	place;	time;	temperature
person87;	m;	11;	-8
person1;	c;	6;	21
person68;	e;	10;	1

Als Restriktionen für die Simulation sind Ort, Zeit, Temperatur definiert, z.B. darf Dokument D_0 nur in Ort b, f, e, i, j, k und zwischen 6 und 13 Uhr angezeigt werden darf.

4.2 InstanceGenerator

Da verschiedene Methoden zur Lösung des Problems untersucht und implementiert werden, muss das Resultat der Simulation mit der gleicher Instanz verglichen werden. Class InstanceGenerator speichert die erzeugte Instanz in einem File und wird für jede Methode aufgerufen.

Um die Methode besser zu bewerten, benötigt es hier die gerade passende Instanzen. Die Instanzen müssen bei $\sum_j k_j = K$ die Nebenbedingung gerade erfüllen können und werden mit folgender Schritte erzeugt:

1. Class InstanceGenerator generiert die Instanzen
2. Ein beliebiges Dokument auswählen für die kommende Person bei Simulation und rechnen die Anzahl Anzeige t_j für alle Dokumente
3. setzen $k_j = t_j$, wobei k_j minimal vorgeschriebene Anzahl von Anzeige ist

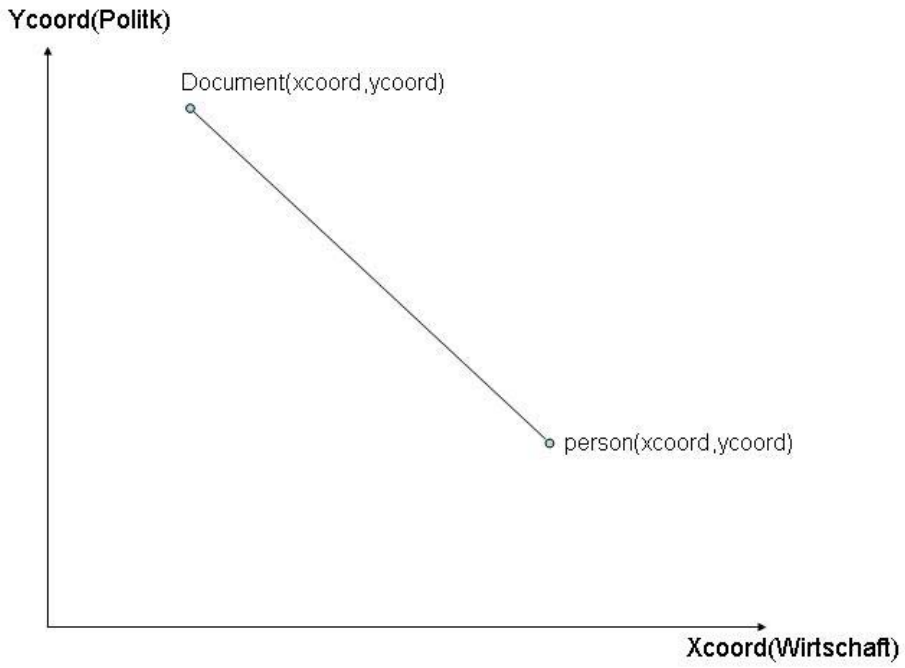

Abbildung 4.1: Scorefunktion

4.3 ScoreFunction

Die beide Koordinaten in Instanzen für Personen und Dokumente (siehe Kapitel 4.1) sind zur Rechnen der Scorefunktion gekommen. Die misst, wie stimmen das Dokument und der Person überein. Class ScoreFunctionImpl rechnet die Score.

Wie machen hier ein Beispiel (siehe Abbildung 4.1) und nehmen an, dass X-Achse auf Wirtschaft und Y-Achse auf Politik hinzielen. In dieser Abbildung lokalisiert Person eher näher auf X-Achse und das Dokument hingegen näher auf Y-Achse, das heisst, der Person interessiert sich mehr auf Wirtschaft und das Dokument geht jedoch mehr um Politik. Ein gutes angepasstes Dokument sollte so näher wie möglich auf dem Person liegen.

Wir rechnen die Scorefunktion mit der Gleichung 4.1, diese Gleichung bedeutet je grösser Abstand $\sqrt{(Xcoord - xcoord)^2 + (Ycoord - ycoord)^2}$ beide Objekte haben desto kleine Scorefunktion besitzt.

$$Scorefunktion = \sqrt{2} - \sqrt{(Xcoord - xcoord)^2 + (Ycoord - ycoord)^2} \quad (4.1)$$

wobei, $\sqrt{2}$ ist der maximale Abstand in der Abbildung 4.1.

4.4 ContextValidator

Class ContextValidator prüft ob das ausgewählt Dokument kontextgültig ist, also Temperatur, Zeit und Ort werden geprüft.

4.5 Schnittstelle

Die Schnittstelle von Schindler ist eine Java Interface "ObjectSelector" mit folgenden Methode:

Eine Methode "public Object choose(Object person, Object context)", die Dokument nach verschiedener Verfahren auswählt und als Rückgabe zurückgibt. Für CPLEX muss zusätzlich noch ein Class "CPLEXWrite" verwenden, um das x.lp zu generieren.

Eine Methode "public void getShowNumber()", die Erfüllung der Nebenbedingung und die Dokumente Wiederholung zeigt. Für Linear Programming wird die Nebenbedingung $\forall j : \sum_i p_i \cdot X_{ij} \cdot d_j \geq q_j \sum_{i,j} p_i \cdot X_{ij} \cdot d_j$ und $\forall i : \sum_j X_{ij} = 1$ mit $0 \leq X_{ij} \leq 1$ noch geprüft.

Um die Ergibnisse graphisch darzustellen, wird ein MATLAB File auch in dieser Methode generiert.

Kapitel 5

Experimentelle Ergebnisse

Nach der Implementierung aller Methoden ist eine Diskussion über die Leistung Pflicht. In dieser Kapitel werden darum das Resultat von aller Verfahren miteinander verglichen.

5.1 Ergebnis von der kleiner Instanzen

Wir haben zuerst einige kleine Instanzen simuliert. Das interessante Ergebnis sagt uns einige Eigenschaften von aller Methoden. Als erste wird eine Instanz mit 4 Dokumente und 5 Personen (siehe Tabelle 5.1 - Tabelle 5.4) simuliert. Wir finden, Methode "Greedy Discounted" hat die Nebenbedingung am besten erfüllt und dazu die beste Scorefunktion bekommen. Wie erwartet ist "Random Counted" bessere Methode gemäss der Erfüllung der Nebenbedingung gegen "Random Repeat", weil die einen Zähler hat.

Wir ändern leicht die Instanz indem wir die gültige Anzeigezeit von Dokumente 1, 2 und 3 sich miteinander überlappen lassen (siehe Tabelle 5.5 - Tabelle 5.8). In diesem Fall muss Dokument 0 als einziger Kandidat für die restliche Anzeigezeit präsentieren und deshalb immer wieder angezeigt

Dokument	Sollwert	Realwert	in %
Doc.0	4000	3950	-1.25
Doc.1 time=[6,16]	3000	3253	8.43
Doc.2 time=[16,18]	1000	1185	18.50
Doc.3 time=[18,23]	2000	1612	-19.40
avgScore		0.7672	

Tabelle 5.1: Random Repeat mit 4 Dok. und 5 Per. Variante1

Dokument	Sollwert	Realwert	in %
Doc.0	4000	4129	3.23
Doc.1 time=[6,16]	3000	3100	3.33
Doc.2 time=[16,18]	1000	1009	0.90
Doc.3 time=[18,23]	2000	1762	-11.90
avgScore		0.7668	

Tabelle 5.2: Random Counted mit 4 Dok. und 5 Per. Variante1

Dokument	Sollwert	Realwert	in %
Doc.0	4000	4382	9.55
Doc.1 time=[6,16]	3000	3969	32.30
Doc.2 time=[16,18]	1000	555	-44.50
Doc.3 time=[18,23]	2000	1094	-45.30
avgScore		0.7321	

Tabelle 5.3: Greedy Counted mit 4 Dok. und 5 Per. Variante1

Dokument	Sollwert	Realwert	in %
Doc.0	4000	4049	1.23
Doc.1 time=[6,16]	3000	3012	0.40
Doc.2 time=[16,18]	1000	916	-8.40
Doc.3 time=[18,23]	2000	2023	1.15
avgScore		0.7996	

Tabelle 5.4: Greedy Disounted mit 4 Dok. und 5 Per. Variante1

Dokument	Sollwert	Realwert	in %
Doc.0	4000	5696	42.40
Doc.1 time=[6,16]	3000	2288	-23.73
Doc.2 time=[11,13]	1000	602	-39.8
Doc.3 time=[12,17]	2000	1414	-29.30
avgScore		0.6561	

Tabelle 5.5: Random Repeat mit 4 Dok. und 5 Per. Variante2

Dokument	Sollwert	Realwert	in %
Doc.0	4000	5176	29.40
Doc.1 time=[6,16]	3000	2597	-13.43
Doc.2 time=[11,13]	1000	624	-37.60
Doc.3 time=[12,17]	2000	1603	-19.85
avgScore		0.6561	

Tabelle 5.6: Random Counted mit 4 Dok. und 5 Per. Variante2

wird und darum haben alle Methoden ein schlechtes Ergebnis gegenüber die originalle Instanz.

Nun vergrössern wir die Instanz indem wir die Anzahl Personen auf 16 erhöhen (siehe Tabelle 5.9 - Tabelle 5.12). wir können hier behaupten, dass beide Greedy Methoden bei grösserer Instanz ein besseres Ergebnis haben während das Ergebnis von beider Random Methoden fast unverändert bleiben.

Zum Schluss simulieren wir auch eine völlig andere Instanz (siehe Anhang C) mit 5 Dokumente und 5 Personen und das Ergebnis zeigt in der Tabelle 5.13 - Tabelle 5.16. Diese Ergebnisse haben unsere Vermutung bewiesen, dass die beide Greedy Methoden besseres Resultat als beide Random Methoden geben können und je grösser die Instanz ist desto besseres Resultat besitzten die beide heuristische Methoden.

Dokument	Sollwert	Realwert	in %
Doc.0	4000	5200	30.00
Doc.1 time=[6,16]	3000	2060	-31.33
Doc.2 time=[11,13]	1000	1000	0.00
Doc.3 time=[12,17]	2000	1740	-13.00
avgScore		0.7230	

Tabelle 5.7: Greedy Counted mit 4 Dok. und 5 Per. Variante2

Dokument	Sollwert	Realwert	in %
Doc.0	4000	4677	19.93
Doc.1 time=[6,16]	3000	2586	-13.80
Doc.2 time=[11,13]	1000	882	-11.80
Doc.3 time=[12,17]	2000	1855	-7.24
avgScore		0.7278	

Tabelle 5.8: Greedy Disounted mit 4 Dok. und 5 Per. Variante2

Dokument	Sollwert	Realwert	in %
Doc.0	4000	4118	2.95
Doc.1 time=[6,16]	3000	3230	7.67
Doc.2 time=[16,18]	1000	1149	14.90
Doc.3 time=[18,23]	2000	1503	-24.85
avgScore		0.6561	

Tabelle 5.9: Random Repeat mit 4 Dok. und 16 Per.

Dokument	Sollwert	Realwert	in %
Doc.0	4000	4381	9.53
Doc.1 time=[6,16]	3000	3000	0.00
Doc.2 time=[16,18]	1000	1000	0.00
Doc.3 time=[18,23]	2000	1619	-19.05
avgScore		0.6996	

Tabelle 5.10: Random Counted mit 4 Dok. und 16 Per.

Dokument	Sollwert	Realwert	in %
Doc.0	4000	4347	8.67
Doc.1 time=[6,16]	3000	2320	-22.67
Doc.2 time=[16,18]	1000	1218	21.80
Doc.3 time=[18,23]	2000	1640	-18.00
avgScore		0.6773	

Tabelle 5.11: Greedy Counted mit 4 Dok. und 16 Per.

Dokument	Sollwert	Realwert	in %
Doc.0	4000	4042	1.05
Doc.1 time=[6,16]	3000	2990	-0.33
Doc.2 time=[16,18]	1000	1020	2.00
Doc.3 time=[18,23]	2000	1948	-2.60
avgScore		0.7935	

Tabelle 5.12: Greedy Discounted mit 4 Dok. und 16 Per.

Dokument	Sollwert	Realwert	in %
Doc.0	298	261	-12.42
Doc.1	81	45	-44.44
Doc.2	388	674	73.71
Doc.3	363	234	-35.54
Doc.4	210	126	-40.00
avgScore		0.6191	

Tabelle 5.13: Random Repeat mit 5 Dok. und 5 Per.

Dokument	Sollwert	Realwert	in %
Doc.0	298	299	0.33
Doc.1	81	56	-30.86
Doc.2	388	569	46.64
Doc.3	363	263	-27.55
Doc.4	210	153	-27.14
avgScore		0.6956	

Tabelle 5.14: Random Counted mit 5 Dok. und 5 Per.

Dokument	Sollwert	Realwert	in %
Doc.0	298	336	12.75
Doc.1	81	88	8.64
Doc.2	388	442	13.92
Doc.3	363	318	-12.40
Doc.4	210	156	-25.71
avgScore		0.7446	

Tabelle 5.15: Greedy Counted mit 5 Dok. und 5 Per.

Dokument	Sollwert	Realwert	in %
Doc.0	298	298	0.00
Doc.1	81	81	0.00
Doc.2	388	388	0.00
Doc.3	363	363	0.00
Doc.4	210	210	0.00
avgScore		0.8037	

Tabelle 5.16: Greedy Discounted mit 5 Dok. und 5 Per.

Die Ergebnisse von der kleiner Instanzen geben uns zwar einige interessante Eigenschaften, stecken hier aber sehr wahnscheinlich viele Zufälle. In der folgender Simulation werden darum immer 10 verschiedene Versuche mit Instanzen von 100 Personen und 50 Dokumente verwendet (Instanz von 5.Versuch siehe Anhang D). Diese 10 Instanzen können die Nebenbedingung gerade gut erfüllen und ausserdem gelten für alle $\sum_j k_j = K$.

Bei der Vorstellung von Verfahren interessieren, eben dem Zweck und dem eigentlichen Ablauf, auch die Eigenschaften (z.B. Laufzeit, Platzbedarf, Vollständigkeit, Korrektheit, Endlichkeit) und die daraus resultierende Vor-/Nachteile des Verfahrens. Es ist aber sicher schwierig, nur ein bestimmtes Kriterium zu definieren. Wir versuchen hier folgende Kriterien zu berücksichtigen.

5.2 Scorefunktion

Da unser primäres Ziel möglich grosser Gewinn zu bekommen ist, besitzt die Scorefunkion klar höchste Priorität. Abbildung5.1 zeigt die simulierte durchschnittliche Scorefunktion von aller 4 Methoden.

Aus Abbildung 5.1 ist es ganz klar, dass Methode “Greedy Discounted” die beste durchschnittliche Scorefunktion gewonnen hat während “Greedy Counted” die 2. beste Scorefunktion hat. “Random Counted” folgt die beide heuristische Methode auf Rang 3. “Random Repeat” hat schliesslich die schlechteste durchschnittliche Scorefunktion.

Da der Werber sehr wahrscheinlich für die überschüssige Dokumente nicht zahlen will oder als Busse von der mangelte Anzeige, rechnen wir für die überschüssige Dokumente nicht die Scorefunktion, das heisst, wenn ein Dokument k_j schon erreicht hat, werden dann die weitere Anzeige nicht mehr zum Rechnen der Scorefunktion berücksichtigt.

5.3 Erfüllung der Nebenbedingung

Für den Werber ist die Erfüllung der Nebenbedingung sehr wichtig. Sie wollen sehen, wieviele Mal wird das Dokument in der betrachteten Zeitperiode angezeigt. Abbildung 5.2 zeigt die Erfüllung in Prozent von 10 Versuche für alle 4 Methode

Aus dieser Abbildung ist es klar, dass “Greedy Discounted” immer die Nebenbedingung 100% erfüllt hat während “Random Repeat” nur ungefähr

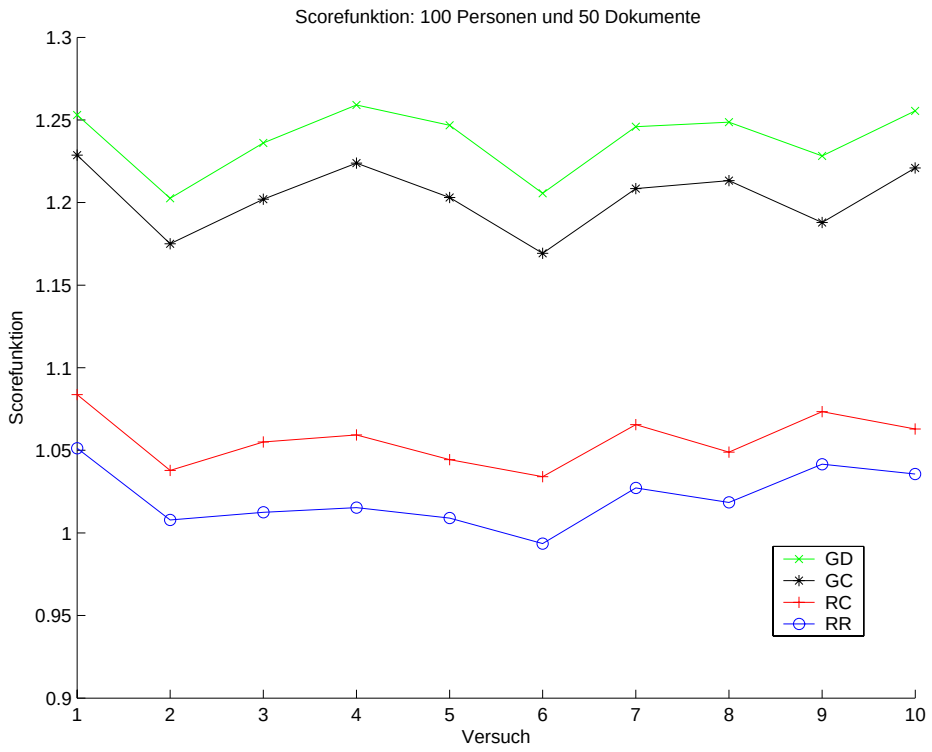

Abbildung 5.1: Ergebnis von Scorefunktion

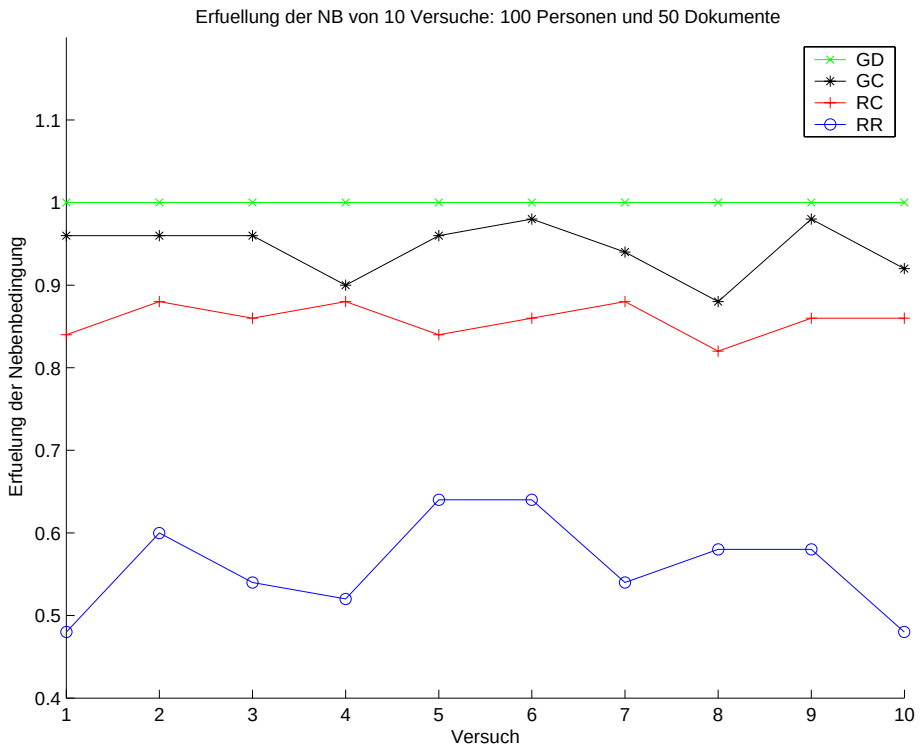

Abbildung 5.2: Erfüllung der Nebenbedingung

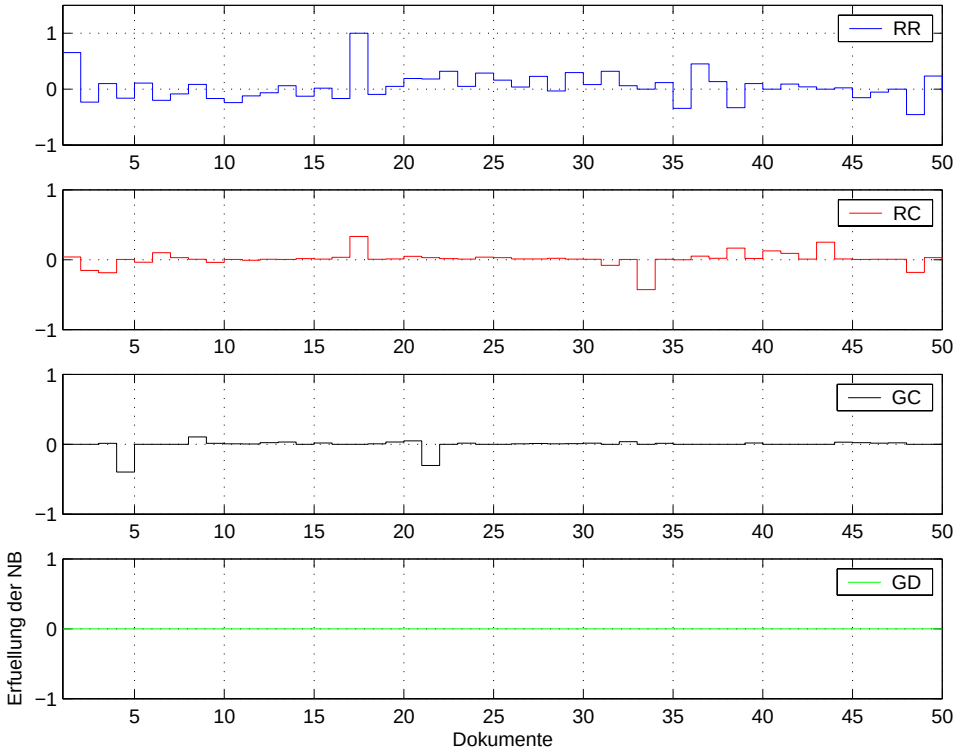

Abbildung 5.3: Erfüllung der Nebenbedingung von 5.Versuch

60% Nebenbedingung erfüllt hat. “Greedy Counted” und “Random Counted” haben ein besseres Ergebnis als “Random Repeat” mit jeweils ungefähr 85% und 95% die Nebenbedingung erfüllt.

Aber Abbildung 5.2 kann nur sagen, dass “Greedy Discounted” die beste Methode zur Erfüllung der Nebenbedingung ist. Man muss hier noch genau untersuchen, wie weit die Nebenbedingung nicht erfüllt werden. Wir betrachten nun beispielweise das Ergebnis von 5.Versuch (Instanz siehe Anhang D). Abbildung 5.3 zeigt diese Erfüllung für alle 4 Methoden.

Man kann aus Abbildung 5.3 leicht erkennen, dass bei Methode “Random Repeat” 36% Dokument die Nebenbedingung so weit (max. 50% weniger Dokumente angezeigt werden) nicht erfüllt haben während bei “Greedy Counted” und “Random Counted” jeweils 96% (max. 39% weniger Dokumente angezeigt werden) und 84% (max. 42% weniger Dokumente angezeigt werden) Dokumente erfüllt haben. Bei “Greedy Discounted” haben alle Dokumente die vorgeschriebene Anzahl erfüllt.

	Random Repeat	Random Counted	Greedy Counted	Greedy Discounted
Laufzeit	13ms	29ms	6ms	29ms

Tabelle 5.17: maximale Laufzeit einer Dokumentauswahl von 5.Versuch

5.4 Anzahl max. wiederholter Dokumente

Aus der Sicht dem Passagier her ist das langweilig, dasselbe Dokumente mehre Mal hintereinander zu sehen, obwohl dieses Dokument genau sein Interesse entspricht. Aus diesem Grund ist maximale Anzahl der wiederholte Dokumente[1] ein sinnvolles Kriterium, wobei maximale Anzahl bedeutet wenn jemand ein Dokument hintereinander 10 Mal gesehen und dasselbe Dokument oder ein anderes Dokument später 16 Mal hintereinander gesehen hat, ist der maximale Anzahl gleich 16. Im Prinzip ist je mehr Anzahl Dokumente desto tritt wenig möglich wiederholte Dokumente auf.

Wir verwenden die Instanz von 5.Versuch (siehe Anhang D) und zeigen die Ergebnisse in Abbildung 5.4: es gibt ein besseres Resultat im Gegenteil zu obiger Kriterien bei Methode "Random Counted" und "Random Repeat". Das ist doch einfach nachzuvollziehen, weil diese beide "Random" Methoden immer eine zufällige Auswahl haben und die andere 2 "Greedy" Methode immer nach Score ein Dokument ausgewählt haben, denn ist es nicht vermeidbar, dass einige grössere Score besitzende Dokumente hintereinander immer ausgewählt werden.

5.5 Laufzeit

Wir hoffen, dass das Service das richtige Dokument so schnell wie möglich auswählen kann. Tabelle 5.17 zeigt darum die maximale Laufzeit einer Dokumentauswahl von 5.Versuch.

"Greedy Discounted" und "Random Counted" besitzen eine längere Laufzeit gegenüber die andere Methode. Der Grund ist: bei "Greedy Discounted" muss für jede Auswahl die Scorefunktion neu berechnet werden und bei "Random Counted" muss nach der Auswahl gemäss X_{ij} aus immer prüfen, ob alle kontextgültige Dokumente ihre k_j bereits erreicht haben. "Greedy Counted" benötigt zwar auch die Zeit, k_j zu prüfen, aber die Auswahl ist einfach, da die Score bei Initial des Class schon berechnet werden.

[1] Dokumentswiederholung 1 bedeutet der Passagier hat dasselbe Dokument 2 mal hintereinander gesehen

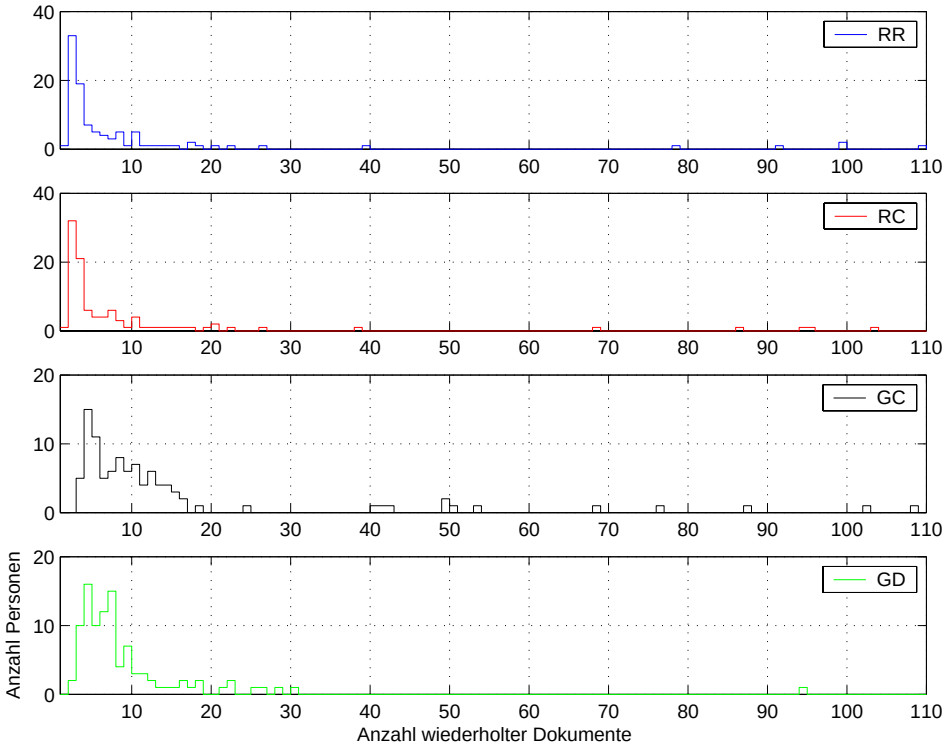

Abbildung 5.4: Anzahl max. wiederholter Dokumente

Die ganze Simulation ist auf ein SunBladen100 mit CPU 500MHz an der ETH Zürich gelaufen. Die gemessene Lautzeit sind aber stark abhängig von dem Netzwerk und der Arbeit, die gerade auf der Maschinen läuft. Die Laufzeit aus Tabelle 5.17 sind aber auf jeden Fall so gering dass man die zur Bewertung der Methode nicht ernst nehmen muss.

5.6 Platzbedarf

Da CPLEX zusätzlich File x.lp generieren muss, benötigt uns zu wissen, wieviele freie Speichern braucht man wenn man Methode "Random Counted" oder "Random Repeat" nimmt. Bei grösser Instanzen z.B. mehr als 300 Personen und 100 Dokumente wird einige Giga Byte Speicher benötigt und kommt immer ein Gefahr "Out of Memeory" an.

Es spielt hier eben eine Rolle, wie präzis werden die Daten gespeichert. Ein File mit 4-stellige Zahl benötigt 4 Mal weniger Speicher als eine double dargestellte Zahl.

Kapitel 6

Schlussfolgerung und Ausblick

6.1 Schlussfolgerung

Nun wir fassen alle Tabellen aus Kapitel 5 in einer neuer Tabelle 6.1[1] zusammen und bewerten aufgrund der Kriterien aus Kapitel 5 die 4 untersuchte Methoden.

Aus der Tabelle 6.1 ist die Methode "Greedy Discounted" ohne Zweifel der Favorit dieser Arbeit. Methode "Greedy Discounted" hat die Nebebbedingung immer 100% erfüllt und deshalb die grösste Scorefunktion gewonnen. Das einzige Problem, dass Anzahl wiederholter Dokumente relative viel ist, kann man mit einer Änderung lösen indem wir z.B. das letzte oder die letzte 3 angezeigt Dokumente nicht mehr auswählen beschranken. Abbildung 6.1 zeigt die neue Scorefunktion: die 2. oben stehende Linie mit "+" ist das Ergebnis von der veränderte Methode "Greedy Discounted", die ist zwar tiefer als "Greedy Discounted", aber immer besser als die alle andere Methoden.

Diese Änderung geht auch auf Kosten von Nebenbedingung, statt alle 10 Versuche 100% Nebenbedingung zu erfüllen, erüllt die neue Methode "Greedy Discounted" 2 Versuche mit 98% Nebenbedingung. Die beide nicht angepasste Versuche haben jeweils ein Dokument ein Mal weniger als vorgeschrieben angezeigt und darum ein Passagier ein Dokument 2 Mal hintereinander gesehen haben muss. Diese Abweichung ist aber so klein dass diese Methode trotzdem die beste Methode bleibt.

Nun sieht die Tabelle 6.1 so aus (siehe Tabelle 6.2) :

[1] Beurteilungen gemäss: + + + best, + + sehr gut, + gut, - schlechte, - - sehr schlecht - - - ganz schlecht

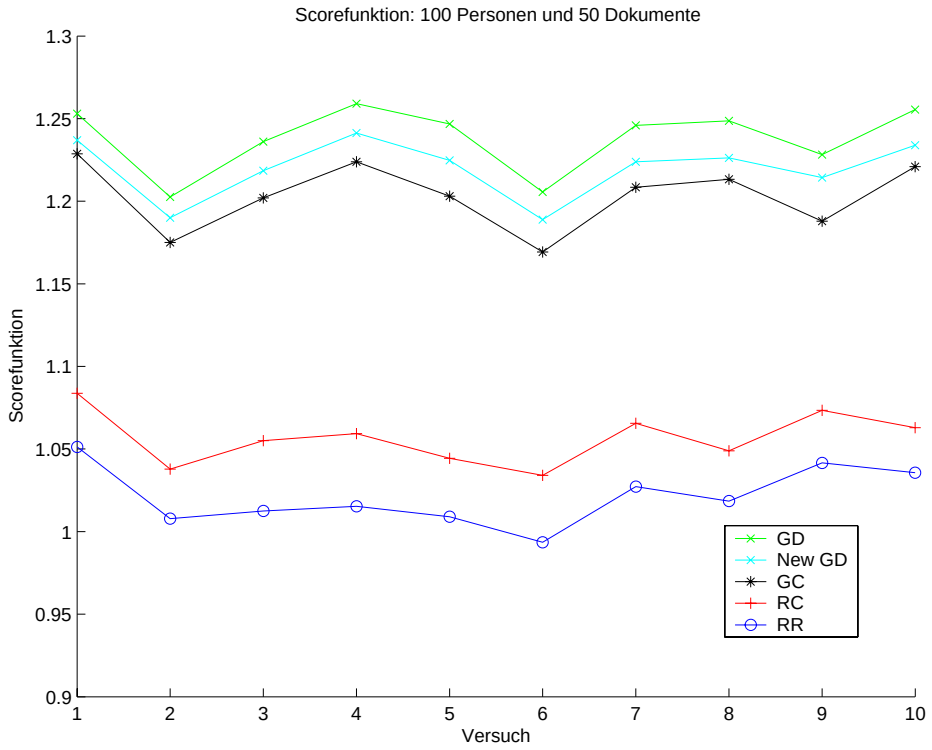

Abbildung 6.1: Ergebnis von Scorefunktion mit veränderte “Greedy Discounted”

	RR	RC	GC	GD
Scorefunktion	- -	+	++	+++
Nebenbedingung	- -	-	+	+++
Anzahl wiederholter Dok.	+	+	- - -	- -
Laufzeit	++	-	+++	-
Platzbedarf	- -	- -	+++	+++

Tabelle 6.1: Bewertung

	RR	RC	GC	GD
Scorefunktion	- -	+	++	+++
Nebenbedingung	- -	-	+	+++
Anzahl wiederholter Dok.	+	+	- - -	+++
Laufzeit	++	-	+++	-
Platzbedarf	- -	- -	+++	+++

Tabelle 6.2: Neue Bewertung

Es scheint erstaunlich, als die beide "Random" Methoden keine gute Lösung geben würden. Ein möglicher Grund dafür ist, dass wir die Zielfunktion und Nebenbedingungen (siehe Kapitel 3) nicht optimal geschrieben haben. Aber der grosse Vorteil von Linear Programming ist, dass es uns sagen kann, ob das betreffende Problem lösbar oder nicht ist. Wir können darum Linear Programming als Checker verwenden und versuchen nur dasjenige Probleme, das Linear Programming erlaubt, zu lösen. Oder können wir dadurch vermeiden, den unlösbaren Vertrag überhaupt zu schliessen.

6.2 Ausblick

In dem Verlauf der Arbeit treten viele Probleme auf. Meistens von der werden bereits gelöst. Es gibt jedoch noch einiges zu behandeln:

- Die Zielfunktion und Nebenbedingungen von Linear Programming besser definieren.
- Ein Programm zu schreiben, dass gerade gute Intanzen generieren kann. Bisher werden alle Instanzen von Hand gemacht, es ist sehr zeitaufwendig und manchmal fehlerhaft.
- grösse Instanzen (z.B. 1000 Personen und 100 Dokumente) simulieren.

- eine Heuristische Methode für das verallgemeinte Problem implementieren.
- Java-Programme zu optimieren am Schluss.

Anhang A

Linear Programming

Um sich mit Linear Programming vertraut zu machen, muss man vor allem folgende Theorem kennenlernen:

Theorem A.1 (Farkas' Lemma für Ungleichung) Das System $Ax \leq b$ hat eine Lösung x, dann und nur dann es wenn keinen Vektor y gibt, der $y \geq 0$, $y^T A = 0$ und $y^T b < 0$ erfüllt.

Corollary A.2 (Farkas' Lemma) Die System $Ax = b$ hat eine nichtnegative Lösung, dann und nur dann wenn es keinen Vektor y gibt, der $y^T A \geq 0$ und $y^T b < 0$ erfüllt.

Corollary A.3 Das System $Ax \leq b$ habe mindestens eine Lösung. Dann erfüllt jede Lösung x $Ax \leq b$, $c^T \leq \delta$, dann und nur dann wenn es einen Vektor $y \geq 0$ gilt, so dass $y^T A = c^T$ und $y^T b \leq \delta$ gilt

Theorem A.4 (Weak Duality Theorem) A sei eine m x n Matrix. Annahme, dass x eine durchführbare Lösung zu $Ax \leq$ b ist und y eine durchführbare Lösung zu $y \geq 0$, $y^T A = c^T$ ist, dann $c^T \tilde{x} \leq \tilde{y}^T b$

Theorem A.5 (Duality Theorem) A sei eine m x n Matirx, $b \in R^m$, $c \in R^n$ dann

$$max\{c^T x : Ax \leq b\} = min\{y^T b : y \geq 0, y^T A = c^T\}$$

vorausgesetzt dass beide Sätze nicht leer sind.

Corollary A.6 A sei eine m x n Matrix, b c , dann

$$max\{c^T x : x \geq 0, Ax = b\} = min\{y^T b : y^T A \geq c^T\}$$

vorausgesetzt dass beide Sätze nicht leer sind.

Theorem A.7 (Complementary Slackness Theorem) x sei eine durchführbare Lösung von $max\{c^T x : Ax \leq b\}$, und y sei eine durchführbare Lösung von $min\{y^T b : y \geq 0, y^T A = c^T\}$, dann x und y sind optimale Lösungen für das maximum und minimum, wenn und nur wenn die Complementary Slackness Bedingung erfüllt.

Anhang B

Simplex Methode

Simplex Methode kann als folgende Algorithmus zusammengefasst werden:

Loop

Find the unique solution to $y^T A_B = c_B$

If $y^T a_i \geq c_i$ for all $i \in T \setminus B$

Stop, the current solutin is optimal.

Else

Choose i such that $y^T a_i < c_i$;

Find the unique solution to $A_B z = a_i$;

Find the largest ε such that $x_B - \varepsilon z \geq 0$;

If ε does not exist

Stop, the LP problem is unbounded.

Else

Choose $j \in B$ such that $z_j > 0$ and the

jth component of $x_B - \varepsilon z$ is 0;

Replace B by $(B \bigcup \{i\}) \setminus \{j\} and)$

x_B by $x_B - \varepsilon z$ and $x_i = \varepsilon$

Anhang C

Kleine Instanz

Die Instanz von Tabelle 5.13, Tabelle 5.14, Tabelle 5.15, Tabelle 5.16 sind:

Dokument:

documentID;xcoord;ycoord;nofDisplays;restrictions*
document0;0.35;0.19;298;place=[f,b,m,d,l,i,j,g,e,h,c,a,k];time=[6,16]
document1;0.43;0.11;81;place=[m,a,c,k];time=[0,9]
document2;0.6;0.99;388;place=[e,l,f,m,k,h,b,c,g,a,i,j]
document3;0.69;0.72;363;place=[h,d,b,m]
document4;0.59;0.77;210;temperature=[22,50];time=[0,19]

Person:

personID;xcoord;ycoord
person0;0.31;0.35
person1;0.81;0.06
person2;0.65;0.0
person3;0.98;0.29
person4;0.6;0.14

Ein Teil von Simulation:

person;place;time;temperature
person4;h;3;13
person3;a;3;4
person0;m;10;4
person1;k;15;23
person2;k;12;0
person0;e;12;-8

person2;k;18;-8
person2;d;15;16
person4;c;4;-7
person1;i;17;16
person0;m;19;-3
person4;i;2;22
person3;a;3;3
person3;i;6;0
person2;f;2;23
person1;g;10;32
person2;i;19;6
person4;b;23;13
person2;e;9;5
person3;l;7;19
person4;f;16;14
person4;m;13;-7
person3;m;17;-4
person4;j;23;24
person2;f;17;0
person3;d;14;25
person0;i;7;29
person2;k;23;29
person2;e;15;4
person0;e;17;18
person0;a;4;20
person3;l;15;-4
person4;h;5;5
person2;h;18;31
person3;b;13;4
person2;e;16;17

Anhang D

Grosse Instanz

Die Instanz von Abbildung 5.3, Abbildung 5.4 und Tabelle 5.17 sind:

Dokument:

documentID;xcoord;ycoord;nofDisplays;restrictions*
document0;0.12;0.99;26;time=[19,23];place=[a,g,j,i]
document1;0.13;0.91;280;time=[1,17];place=[l,a,m,g,e,h,i,d]
document2;0.49;0.91;80;time=[0,3]
document3;0.09;1.0;379;place=[l,g,f,c,h,b,i]
document4;0.82;0.49;28;time=[2,10];place=[l,g]
document5;0.27;0.93;10;time=[9,15];place=[h]
document6;0.3;0.27;35;temperature=[-7,-2];place=[f,b,h,k,i,m]
document7;0.59;0.3;193;place=[k,b,j,d]
document8;0.72;0.87;77;time=[14,17];place=[i,m,c,e,g,d,h,f,k,l,b]
document9;0.67;0.65;376;time=[0,19];place=[f,i,c,h,l,d,g,j,e]
document10;0.43;0.56;231;time=[0,11];place=[d,c,m,g,h,i,a,l,e,f]
document11;0.34;0.81;585;place=[a,i,g,h,j,b,d,k,l,c,m]
document12;0.14;0.34;577;place=[k,f,g,a,i,l,d,m,e,j,c]
document13;0.93;0.34;63;time=[13,16];place=[g,b,j,h,i,c,e,m,d,l,f]
document14;0.45;0.32;543;place=[k,g,j,h,a,m,e,b,i,c,d]
document15;0.52;0.64;30;time=[9,10];place=[m,g,j,b,c,k,f,l,e,a,h,i]
document16;0.81;0.21;3;time=[11,12];place=[f,l]
document17;0.67;0.1;149;place=[a,b,f]
document18;0.82;0.15;581;place=[d,j,a,i,g,l,e,m,k,h,b,f,c]
document19;0.57;0.37;21;time=[17,22];place=[j,f]
document20;0.21;0.97;33;time=[4,21];place=[c]
document21;0.12;0.64;69;time=[5,13];place=[d,c,g,m,b,e]
document22;0.97;0.73;558;place=[m,i,j,a,c,l,d,b,f,k,g,h,e]

document23;0.75;0.65;28;time=[1,3];place=[e,k,h,d,i,a,j,l,g,c]
document24;0.8;0.77;100;time=[3,14];place=[h,m,k,i,a,d]
document25;0.23;0.05;183;time=[16,23]
document26;0.58;0.82;96;time=[1,7];temperature=[-9,22]
document27;0.49;0.42;145;time=[1,8]
document28;0.55;0.31;119;place=[j,g,b,c]
document29;0.55;0.84;334;place=[l,b,d,i,f,c,h,m]
document30;0.5;0.82;25;time=[2,9];place=[a,d,e]
document31;0.49;0.77;388;place=[l,a,b,j,d,i,m,e,f]
document32;0.7;0.81;7;time=[8,11];place=[d]
document33;0.45;0.88;594
document34;0.44;0.19;32;time=[0,2];place=[l,k,m,a,g,j]
document35;0.78;0.32;20;time=[13,21];place=[h]
document36;0.88;0.9;52;time=[4,19];temperature=[16,25];place=[g,h,f,e,b,a,k,i,m,c,l,j]
document37;0.92;0.05;6;time=[21,22];place=[b,l]
document38;0.56;0.5;112;time=[0,22];place=[l,a,g]
document39;0.37;0.64;8;time=[1,3];place=[g,m]
document40;0.41;0.43;11;time=[2,7];place=[g,f]
document41;0.56;0.05;120;place=[l,g,h]
document42;0.6;0.95;4;time=[16,18];place=[l]
document43;0.94;0.66;405;place=[b,f,e,k,l,d,h,g,c]
document44;0.06;0.22;308;place=[f,j,d,h,k,i]
document45;0.08;0.71;439;place=[i,j,c,e,l,b,f,m,h]
document46;0.68;0.9;145;place=[l,j,a]
document47;0.31;0.62;11;time=[9,11];place=[c,e]
document48;0.28;0.65;34;time=[5,13];temperature=[2,18];place=[j,a,i,h,b,f,g,e]
document49;0.5;0.01;451;place=[c,a,f,d,i,g,k,e,h]

Ein Teil von Person:

personID;xcoord;ycoord
person0;0.29;0.33
person1;0.69;0.75
person2;0.91;0.6
person3;0.48;0.21
person4;0.98;0.61
person5;0.69;0.88
person6;0.25;0.09
person7;0.28;0.27
person8;0.72;0.38
person9;0.71;0.41

person10;0.51;0.63
person11;0.42;0.94
person12;0.77;0.78
person13;0.98;0.49
person14;0.26;0.34
person15;0.1;0.69
person16;0.65;0.89
person17;0.82;0.04
person18;0.38;0.14
person19;0.79;0.05
person20;0.27;0.98
person21;0.05;0.43
person22;0.98;0.6
person23;0.38;0.84
person24;0.0;0.63
person25;0.46;0.69
person26;0.47;0.67
person27;0.33;0.76
person28;0.01;0.9
person29;0.59;0.91
person30;0.45;0.55

Ein Teil von Simulation:

person;place;time;temperature
person2;i;2;13
person93;f;8;-2
person48;d;4;12
person75;d;19;16
person34;j;9;12
person35;g;6;0
person3;k;3;-8
person94;e;7;15
person90;a;18;23
person17;j;8;0
person0;g;5;18
person2;j;10;15
person48;b;22;8
person97;g;23;-8
person60;j;22;4
person60;e;17;21

person48;g;23;4
person96;f;10;13
person27;h;4;24
person38;d;16;27
person31;d;9;23
person56;c;11;-4
person46;c;16;22
person13;l;6;1
person67;l;7;15
person96;d;6;12
person54;j;1;3
person1;f;14;0
person32;l;8;25
person38;m;0;-6
person59;m;10;5
person86;j;17;26
person83;d;15;-1

Literaturverzeichnis

[1] Cook, Cunningham, Pulleyblank, Schrijver. 1998. *Combinatorial Optimization*, JOHN WILEY & SONS, INC

[2] Colin R Reeves. 1993 *Modern Heuristic Techniques For Combinatorial Problems*, Blacktwell Scientific Publications

[3] Christos H. Papadimitriou, Kenneth Steiglitz. 1998 *Combinatorial Optimization: Algorithms and Complexity*, University of California-Berkeley, Princeton University

[4] Optimization Technology Center of Northwestern University and Argonne National Laboratory. 2003 *Linear Programming Frequently Asked Questions*, http://www-unix.mcs.anl.gov/otc/Guide/faq/linear-programming-faq.html

[5] Robert J. Vanderbei. 1996 *Linear Programming*, Operations Research and Financial Engineering, Princeton Univercity

[6] Peter Merz. 2003 *Moderne Heuristische Optimierungsverfahren*, Wilhelm-Sckickard-Institut für Informatik

[7] Alper Atamtürk, Martin W. P. Savelsbergh. 2003 *Commercial Interger Programming Software Systems*

[8] ILOG, Inc. July 2002. *Getting Started*, ILOG CPLEX 8.0.