Sitzplatzreservierungsproblem

Bei vielen Zugsystemen in Europa müssen Passagiere mit ihrem Zugticket eine Sitzplatzreservierung kaufen. Da das Ticketsystem Kunden einen einzelnen Platz zuweisen muss, wenn diese ein Ticket kaufen, ohne zu wissen, welche zukünftigen Bestellungen für die Sitzplätze eingehen werden, ist dies ein online Algorithmus.

Bei den folgenden Untersuchungen nehmen wir an , dass ein Zug mit n Sitzplätzen von einem Startbahnhof zu einer Endstation fährt und an $k \geq 2$ Stellen inklusive Start- und Endstation hält. Der Startbahnhof ist 1, der Zielbahnhof ist k. Die Sitzplätze sind von 1 bis n durchnumeriert.

Reservierungen können vor jeder Reise von einer Startstation s bis zu einer Endstation t mit $1 \leq s < t \leq k$ vorgenommen werden. Der Reisende bekommt eine einzelne Sitzplatznummer zugewiesen, wenn er das Ticket kauft bzw. bestellt. Dies kann zu jeder Zeit vor der Abfahrt von seinem Startbahnhof geschehen.

Die Algorithmen versuchen, den Erlös zu maximieren, d.h. die Summe der Preise der verkauften Tickets. Dabei hängt die performance von der Preispolitik ab. Die Kosten sollen bei unserer Betrachtung keine Rollespielen. Es gibt zum einen das unit-price-problem, bei dem alle Tickets unabhängig von der Länge der Reise den gleichen Preis haben und zum anderen das proportional price problem, bei dem der Preis des Tickets proportional zur Reisestrecke ist. Im Rahmen dieser Arbeit beschränken wir uns auf das unit-price-problem.

Zusätzlich definieren wir die accommodating ratio, die ähnlich zur competitive ratio ist, nur dass sich diese auf den optimalen offline Algorithmus bezieht, der die gesamte Nachfrage nach Sitzplätzen bedienen kann. C-accommodating heißt also, dass ein online Algorithmus einen Bruchteil c vom möglichen Gesamteinkommen des offline Algorithmus erwirtschaftet. Die Annahme, dass es genügend Sitzplätze für den optimalen annehmbaren offline Algorithmus gibt, ist nur dann gültig, wenn das Management eine vernünftige Vorhersage über die Ticketnachfrage gemacht hat und eine entsprechende Anzahl an Waggons bereitstellt. Die Dichte zwischen einer Station s und (s+1) beschreibt die Anzahl der Personen, die zwischen diesen mit dem Zug reisen wollen. Im Folgenden wird die Dichte kleiner gleich n gesetzt.

Wenn ein Kunde versucht, ein Ticket zu bestellen, darf dieser aus politischen Gründen nicht vom Ticketagent zurückgewiesen werden, falls es möglich ist, ihn unterzubringen. Falls ein Sitz die ganze Fahrt über frei ist, muss dem Kunden ein Sitzplatz zugewiesen werden. Diese

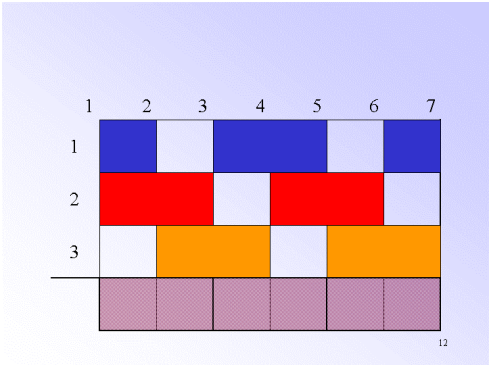
Eigenschaft des Algorithmus wird als „fair“ bezeichnet. Im folgenden werden nur „faire“ Algorithmen betrachtet.

Das unit price problem offline entspricht dem Intervallgraphfärbeproblem offline, da jedes Ticket den selben Preis unabhängig von der Weglänge hat. Die Wegstrecke von der Startstation 1 bis zur Endstation k entspricht dem Gesamtintervall. Die Reisestrecke eines Passagiers entspricht einem Teilintervall. Zum Färben braucht man dann so viele Farben wie Teilintervalle übereinander liegen. Im Beispiel (Abb. 1) sind dies drei Teilintervalle.

Die Intervallgraphfärbung online ist durch die Anwendungen in der dynamischen Speicherplatzzuweisung gut erforscht. Kierstead und Trotter haben herausgefunden, dass es einen online Algorithmus gibt, so dass $A(G) \leq 3 \cdot X(G) - 2$ gilt, wobei X(G) die Anzahl der Farben sind, die ein offline Algorithmus benötigt, und A(G) die Anzahl der Farben, die ein online Algorithmus benötigt, um G zu färben. Auf das online Ticketproblem bezogen bedeutet dies, dass der Zug dreimal mehr Sitzplätze haben müsste als bei einer Bearbeitung mit einem optimalen offline Algorithmus. Diese sind jedoch fest vorgegeben. Außerdem geht man beim online Graphfärben von einer Vielzahl von unterschiedlichen Endpunkten aus, beim Sitzplatzreservieren sind die Anzahl der Bahnhöfe aber vorgegeben.

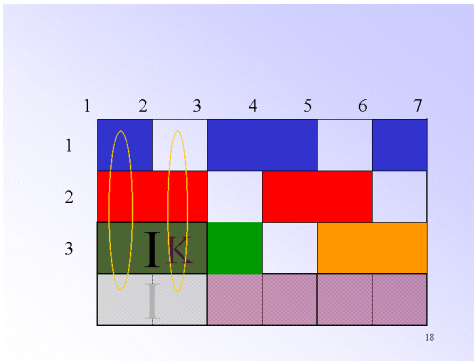
Satz 1: Jeder (noch so „dumme“) faire, deterministische online Algorithmus für das unit-price-problem ist mindestens ½-accommodating.

Für den Beweis kann man jede Sitzplatzkonstellation in Betracht ziehen, die der optimale faire offline Algorithmus bedienen kann. Dabei sei S die Menge aller Sitzplatzzuweisungen, die durch einen fairen online Algorithmus bedient werden konnten, und U die Menge aller unbesetzten Sitzplätze. Die Induktionsbeweisidee ist nun die, dass man zeigen muss, dass es mindestens genau so viele Intervalle in S wie in U gibt, indem man jedem Intervall in U ein bestimmtes Intervall in S zuweist. Dazu wird die Menge S’ definiert, die zu Beginn gleich S ist, aber durch den folgenden Prozess abgeändert wird. Die Intervalle aus U werden den Startpunkten nach sortiert und bearbeitet. Wurde ein Intervall bearbeitet, so wird es aus U entfernt, einem Intervall aus S zugewiesen und ändere damit ein Intervall in S’ um. Die



folgende Invariante wird nun fortgeführt. Da der Algorithmus fair ist, kann keines der Intervalle aus U in S' eingefügt werden, aber alle Intervalle aus U und S' könnten durch einen optimalen offline Algorithmus nach Definition von U und S' untergebracht werden.

Der Induktionsschritt sind nun wie folgt aus. Für ein gegebenes Intervall $I \in U$ muss ein freier Sitzplatz in S' von der Startstation bis zur zumindest nächsten Station gefunden werden können, denn der optimale offline Algorithmus könnte alle Sitzplatzanfragen bedienen. Durch die Invariante kann I kein Platz zugewiesen werden. Demnach muss ein Intervall J dem Sitzplatz in S' zugewiesen werden, das I überschneidet. I wird nun J zugewiesen und aus U entfernt. J wird in S' auf diesem Platz durchein Intervall K ersetzt, das so groß ist, wie momentan von $I \cup J$ auf den Sitzplatz passen.



Alle Intervalle, die jetzt in S' enthalten sind und alle unbesetzten Intervalle in U könnten durch den optimalen offline Algorithmus untergebracht werden, da dieser Vorgang nicht die Dichte erhöhen kann. Außerdem wurde S' expandiert, so dass keines der verbleibenden Intervalle aus U passen wird.

Dieser Vorgang kann wiederholt werden. Die geordnete Reihenfolge stellt sicher, dass jedem Intervall $I \in U$ ein bestimmtes Intervall in S zugewiesen wird. Daher ist jeder online Algorithmus mindestens ½-accommodating.

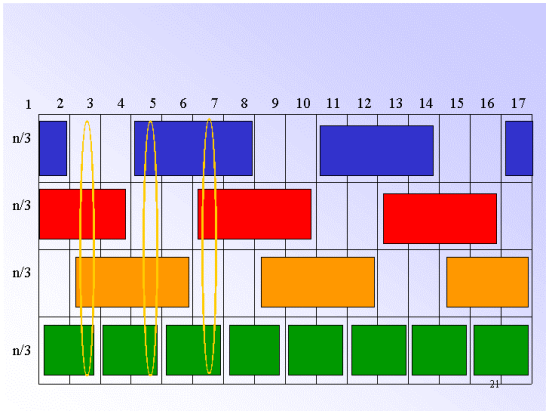
Satz 2: Die Accommodating Ratio für First-Fit und Best-Fit sind für das unit-price-problem nicht besser als $\frac{k}{2k-6}$ unter der Annahme, dass k ≥ 4 und k ≡ 4 mod

Wir nehmen an, dass n durch 3 teilbar ist und teilen die Anzahl der Sitzplätze in drei gleich große Teile ein. Es kommen nacheinander vier verschiedene Anfragesequenzen an.

1. Zunächst kommen n/3 Anfragen für das Intervall [1, 2], dann für s = 0, 1, ..., (k-10)/6 gibt es n/3 Anfragen für die Intervalle [6s + 4, 6s + 8].
2. Am Anfang kommen n/3 Anfragen für das Intervall [1, 4], danach für s = 1, 2, ..., (k-4)/6 werden die Intervalle [6s, 6s + 4] nachgefragt.
3. Auf die letzten n/3 Plätze kommen die Anfragen für s = 0, 1, ..., (k-10)/6 auf die Intervalle [6s + 2, 6s + 6], da First-Fit diese nicht auf die ersten beiden n/3 Sitzplätze unterbringen kann.

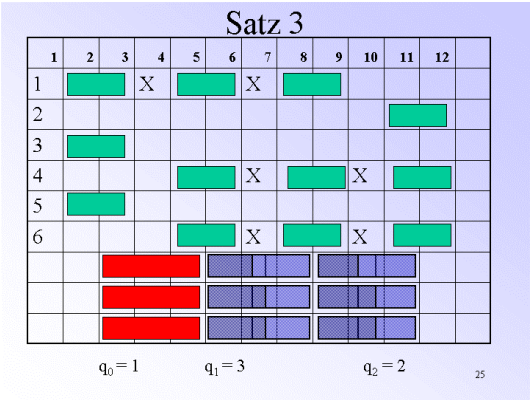
4. Für s = 0, 1, ..., (k-8)/2 kommen die Anfragen für die Intervalle [2s + 1, 2s + 3]. Diese können nun nicht mehr von First-Fit untergebracht werden, da kein Sitzplatz für mehr als zwei Stationen unbelegt ist und keine Lücke an einem Bahnhof mit einer ungeraden Nummer frei wird.

Für die oben beschriebenen Abfolgesequenzen machen First-Fit und Best-Fit den gleichen Belegungsplan.



Da First-Fit $\frac{n}{3} \cdot \frac{3(k-4)}{6} + 2$ Anfragen bedient, der optimale offline Algorithmus aber $\frac{n}{3} \cdot \frac{3(k-4)}{6} + 2 + \frac{k-6}{2}$, ist die accommodating ratio nicht besser als $\frac{k}{2k-6}$.

Satz 3: Kein deterministischer fairer online Algorithmus ist mehr als $\frac{8k-9}{10k-15}$-accommodating, wenn k durch 3 teilbar ist. Sei n durch zwei teilbar. Es kommen dann immer n/2 Anfragen für die Intervalle [3s + 1, 3s + 3] mit s = 0, 1, ..., (k-3)/3. Es wird angenommen, dass der faire online Algorithmus diese k*n/6 Intervalle so anordnet,



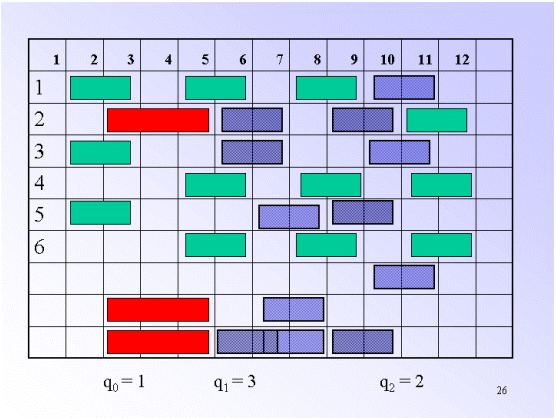
dass es nach diesen Anfragen genau q_i Sitzplätze gibt, die beide Intervalle [3i+1, 3i+3] und [3i+4, 3i+6] mit i = 0, 1, ..., (k-6)/3 enthalten. Dann gibt es genau q_i die von Station 3i+2 bis Station 3i+5 frei sind. Der Beweis unterscheidet in 2 Fälle:

1. $q_i \leq 3n/10$

   Nun werden n/2 Anfragen für das Intervall von [3i+2, 3i+5] nachgefragt. Der Algorithmus wird q_i Anfragen bedienen können, der optimale offline Algorithmus kann allerdings alle bedienen.

2. $q_i > 3n/10$

   In diesem Fall kommen n/2 Anfragen für die Intervalle von [3i+2, 3i+4] und n/2 Anfragen für das Intervall [3i+3, 3i+5]. Der online Algorithmus kann n-q_i Anfragen bedienen, der optimale offline Algorithmus könnte aber wieder alle Anfragen bedienen.



Sei S die Menge der Indizes, in denen Fall 1 zutrifft, und $\overline{S}$ die Menge aller Indizes, in denen Fall 2 zutrifft.

$$\frac{\frac{k\cdot n}{6}+\sum_{i\in S} q_i+\sum_{i\in \overline{S}}(n-q_i)}{\frac{k\cdot n}{6}+\sum_{i\in S}\frac{n}{2}+\sum_{i\in \overline{S}} n} \leq \frac{\frac{k}{6}+\sum_{i\in S}\frac{3}{10}+\sum_{i\in \overline{S}}\frac{7}{10}}{\frac{k}{6}+\sum_{i\in S}\frac{1}{2}+\sum_{i\in \overline{S}}1} = \frac{\frac{1}{2}+\sum_{i\in S}(\frac{1}{2}+\frac{3}{10})+\sum_{i\in \overline{S}}(\frac{1}{2}+\frac{7}{10})}{1+\sum_{i\in S}1+\sum_{i\in \overline{S}}\frac{3}{2}}$$

$$\leq \frac{\frac{1}{2}+\frac{k-3}{3}\left(\frac{4}{5}\right)}{\frac{1}{2}+\frac{k-3}{3}} = \frac{8k-9}{10k-15}$$