INSTITUT FÜR INFORMATIK I

Lehrstuhl für Theoretische Informatik und formale Methoden
Prof. Dr. Rolf Klein

Seminar

Randomisierte Algorithmen

Beitragsthema:

Online Algorithmen I

Bearbeiter: Sebastian Bergmann
Betreuer: Elmar Langetepe

4. Februar 2003

1 Einleitung

Bislang haben wir Algorithmen betrachtet, die ihre vollständige Eingabe zu Beginn ihrer Ausführung (*offline*) erhalten. Wir betrachten nun Algorithmen, die ihre Eingabe stückweise *online* erhalten.

1.1 Definition: Online Algorithmus

Defintion 1 (Online Algorithmus) *Ein Online Algorithmus erhält eine Sequenz von Anfragen als Eingabe. Jede dieser Anfragen muss zum Zeitpunkt ihres Eintreffens, also ohne das Wissen über die nachfolgenden Anfragen, vom Online Algorithmus beantwortet werden. Die Beantwortung einer Anfrage ist mit Kosten verbunden.*

1.2 Ein Beispiel, das jeder kennt

Ein Beispiel aus der Welt der Spiele ist Tetris. Hierbei muss eine Sequenz von Spielsteinen in einem Spielfeld den Regeln entsprechend so untergebracht werden, dass ein bestimmtes Ziel (beispielsweise Maximierung der platzierten Spielsteine) erreicht wird.

Zum Zeitpunkt des Setzens eines Spielsteins sind die zukünftigen Spielsteine nicht bekannt. Durch ungeschickte Platzierung eines Steins kann man sich "seinen Erfolg verbauen".

Dass dies nicht so einfach ist, wie es auf den ersten Blick aussieht, belegt die Tatsache, dass die Offline Variante von Tetris NP vollständig ist [2].

2 Das Online Paging Problem

2.1 Vorstellung des Problems

Wir betrachten eine Speicherverwaltung eines Rechners mit zwei Ebenen:

- Im *Cache* können k Elemente abgelegt werden, auf die schnell zugegriffen werden kann.
- Der *Hauptspeicher* kann potentiell beliebig viele Elemente, auf jeden Fall mehr als k, aufnehmen.

  Der Zugriff auf die Elemente im Hauptspeicher erfolgt langsamer als der Zugriff auf die Elemente im Cache.

Bei einem Speicherzugriff kann eine der beiden folgenden Situationen eintreten:

- *Cache Hit*
  Das Speicherelement, auf das zugegriffen wird, befindet sich im Cache. Es entstehen keine Kosten für den Zugriff.

- *Cache Miss*
  Das Speicherelement, auf das zugegriffen wird, befindet sich nicht im Cache.

  Das Speicherelement muss aus dem Hauptspeicher gelesen werden, hierfür fallen konstante Kosten an.

  Ferner muss im *Cache* Platz für das aus dem Hauptspeicher gelesene Element geschaffen werden.

2.2 Kostenmaß

Das Kostenmaß für das Online Paging Problem ist die Anzahl der *Cache Misses* für eine Sequenz von Anfragen.

Defintion 2 *Die Anzahl der Cache Misses, die ein deterministischer Online Algorithmus $\mathcal{A}$ für eine Sequenz $\rho_1, \ldots, \rho_N$ von Anfragen "erleidet", schreiben wir als $f_{\mathcal{A}}(\rho_1, \ldots, \rho_N)$.*

Defintion 3 *Die Anzahl der Cache Misses, die ein optimaler Offline Algorithmus $\mathcal{O}$ für eine Sequenz $\rho_1, \ldots, \rho_N$ von Anfragen "erleidet", schreiben wir als $f_{\mathcal{O}}(\rho_1, \ldots, \rho_N)$.*

2.3 Strategien zur Cacheverwaltung

Offensichtlich hängen die Kosten von der Wahl desjenigen Cachelementes ab, das gelöscht wird, um für ein aus dem Hauptspeicher gelesenes Platz zu schaffen.

- *First-In, First-Out (FIFO)*

  Löscht das älteste Element im Cache.

- *Least Frequently Used (LFU)*

  Löscht das am seltensten angefragte Element.

- *Least Recently Used (LRU)*

  Löscht das Element, dessen letzter Zugriff am weitesten zurück liegt.

2.4 Der MIN-Algorithmus

Bei einem *Cache Miss* wird das Element aus dem Cache gelöscht, dessen nächster Zugriff am weitesten in der Zukunft liegt.

Satz 1 (ohne Beweis) *Der MIN-Algorithmus ist optimal.*

3 Kompetitive Analyse

Ein absolutes Bewertungsmaß für einen Online Algorithmus ist in den meisten Fällen nicht sinnvoll, da die absolute Bewertung für einen Online Algorithmus beliebig schlecht gemacht werden kann, indem die Anfragesequenz entsprechend gewählt wird.

Ein Vergleich von verschiedenen Online Algorithmen für ein Problem ist so nicht möglich.

3.1 Kompetitivität

Kompetitivität ist ein Maß, mit dem man die Kosten für einen Online Algorithmus im Worst Case mit denen eines optimalen Offline Verfahrens vergleichen kann, das die gleiche Anfragesequenz bedient.

Defintion 4 (Kompetitivitätskoeffizient) *Ein deterministischer Online Algorithmus $\mathcal{A}$ für das Online Paging Problem heisst $\mathcal{C}$-kompetitiv, falls eine Konstante b existiert, so dass für jede Sequenz von Anfragen $\rho_1, \ldots, \rho_N$*

$$f_{\mathcal{A}}(\rho_1, \ldots, \rho_N) - \mathcal{C} \times f_{\mathcal{O}}(\rho_1, \ldots, \rho_N) \leq b$$

gilt. Die Konstante b muss hierbei unabhängig von N sein. Sie kann abhängig von k sein. Das Infimum von $\mathcal{C}$, für das $\mathcal{A}$ $\mathcal{C}$-kompetitiv ist, wird mit $\mathcal{C}_{\mathcal{A}}$ notiert und heisst Kompetitivitätskoeffizient von $\mathcal{A}$.

LRU und FIFO sind k-kompetitiv – also maximal "k-mal schlechter als ein optimales Offline Verfahren" – und optimal, für LFU lässt sich keine solche Schranke finden.

3.2 Schranke für die Kompetitivität

Satz 2 *Sei A ein deterministischer Online Algorithmus für das Online Paging Problem. Dann gilt $\mathcal{C}_{\mathcal{A}} \geq k$.*

Zum Beweis ziehen wir einen "Gegner" heran, der eine Anfragesequenz erzeugt, die das Worst Case Verhalten des Online Algorithmus provoziert.

Dieser Gegner konstruiert die Anfragesequenz wie folgt:

- Zu Beginn befinden sich die Elemente A, B, C und D im Cache.
- Der Gegner fordert Element E an, das nicht im Cache liegt. (Cache Miss)
- Ein Element, zum Beispiel A, wird aus dem Cache entfernt, um Platz für E zu schaffen.
- Setze S := {A, B, C, D, E} und frage von nun an stets nach dem Element aus S, das momentan nicht im Cache ist.

A	E	A	
B	B	B	
C	C	D	…
D	D	E	
E	A	C	

Abbildung 1: Anfragesequenz, die mit jeder Anfrage einen Cache Miss erzeugt

Für die Zwecke des Beweises teilen wir die Anfragesequenz in Runden auf, die erste Runde beginnt mit dem ersten Element der Anfragesequenz. Eine Runde ist die maximale Sequenz, in der k verschiedene Elemente angefragt werden. Diese k Elemente können beliebig oft angefragt werden. Eine Runde endet, wenn das (k+1)-te Element angefragt wird. Dieses Element ist das erste der nächsten Runde.

$\mathcal{A}$ erleidet pro Runde mindestens k Cache Misses. Ein optimaler Offline Algorithmus erleidet maximal einen Cache Miss pro Runde. Daraus folgt, dass $\mathcal{A}$ mindestends k-mal so viele Cache Misses erleidet wie ein optimaler Offline Algorithmus. Also gilt unsere Behauptung und der Satz ist bewiesen.

4 Randomisierung von Online Paging

Wir betrachten nun einen randomisierten Online Algorithmus $\mathcal{R}$ für das Online Paging Problem.

Defintion 5 *Die Anzahl der Cache Misses, die ein randomisierter Online Algorithmus $\mathcal{R}$ für eine Sequenz $\rho_1, \ldots, \rho_N$ von Anfragen "erleidet", schreiben wir als $f_{\mathcal{R}}(\rho_1, \ldots, \rho_N)$.*

$f_{\mathcal{R}}(\rho_1, \ldots, \rho_N)$ ist eine Zufallsvariable!

Wie zuvor untersuchen wir das Verhalten von $\mathcal{R}$, in dem wir $\rho_1, \ldots, \rho_N$ von einem *Gegner* erzeugen lassen.

Für diese Untersuchung ist von besonderem Interesse, welche Informationen der Gegner über den Algorithmus $\mathcal{R}$ zur Verfügung hat.

4.1 Unwissender Gegner

Ein *unwissender Gegner* kennt den Algorithmus $\mathcal{R}$, wird aber nicht über die zufälligen Entscheidungen von $\mathcal{R}$ informiert. Er kann die Anfragesequenz im Voraus erzeugen, da sie nicht aufgrund des Verhaltens von $\mathcal{R}$ anpasst werden kann.

Der Gegner beantwortet die Anfragesequenz seinerseits mit dem MIN Algorithmus bei fixen Kosten $f_{\mathcal{O}}(\rho_1, \ldots, \rho_N)$.

Defintion 6 *Ein randomisierter Online Algorithmus $\mathcal{R}$ für das Online Paging Problem heisst $\mathcal{C}$-kompetitiv gegen einen unwissenden Gegner, falls eine Konstante b existiert, so dass für jede Sequenz von Anfragen $\rho_1, \ldots, \rho_N$*

$$E[f_{\mathcal{R}}(\rho_1, \ldots, \rho_N)] - \mathcal{C} \times f_{\mathcal{O}}(\rho_1, \ldots, \rho_N) \leq b$$

gilt. Die Konstante b muss hierbei unabhängig von N sein. Das Infimum von $\mathcal{C}$, für das $\mathcal{R}$ $\mathcal{C}$-kompetitiv ist, wird mit $\mathcal{C}_{\mathcal{R}}^{obl}$ notiert und heisst Kompetitivitätskoeffizient von $\mathcal{R}$ gegen einen unwissenden Gegner.

4.2 Adaptive Gegner

- Ein *adaptiver Offline Gegner* wählt das Element ρ_{i+1} der Anfragesequenz aufgrund des Verhaltens von $\mathcal{R}$ für $(\rho_1, \ldots, \rho_i)$. Nach Erzeugung der Anfragesequenz $(\rho_1, \ldots, \rho_N)$ beantwortet der adaptive Offline Gegener diese unter Verwendung des MIN Algorithmus.

- Ein *adaptiver Online Gegner* wählt das Element ρ_{i+1} aufgrund des Verhaltens von $\mathcal{R}$ für $(\rho_1, \ldots, \rho_i)$, verwaltet aber selber einen Cache und beantwortet die Anfrage ρ_{i+1} zum Zeitpunkt ihrer Erzeugung ebenfalls "online".

$f_{\mathcal{R}}(\rho_1, \ldots, \rho_N)$ und $f_{\mathcal{O}}(\rho_1, \ldots, \rho_N)$ sind in beiden Fällen Zufallsvariablen.

Defintion 7 *Ein randomisierter Online Algorithmus $\mathcal{R}$ für das Online Paging Problem heisst $\mathcal{C}$-kompetitiv gegen einen adaptiven Offline Gegner, falls eine Konstante b existiert, so dass für jede Sequenz von Anfragen $\rho_1, \ldots, \rho_N$*

$$E[f_{\mathcal{R}}(\rho_1, \ldots, \rho_N)] - \mathcal{C} \times E[f_{\mathcal{O}}(\rho_1, \ldots, \rho_N)] \leq b$$

gilt. Die Konstante b muss hierbei unabhängig von N sein. Das Infimum von $\mathcal{C}$, für das $\mathcal{R}$ $\mathcal{C}$-kompetitiv ist, wird mit $\mathcal{C}_{\mathcal{R}}^{aof}$ notiert und heisst Kompetitivitätskoeffizient von $\mathcal{R}$ gegen einen adaptiven Offline Gegner.

Analog: $\mathcal{C}_{\mathcal{R}}^{aon}$ Kompetitivitätskoeffizient von $\mathcal{R}$ gegen einen adaptiven Online Gegner.

Es gilt

$$\mathcal{C}_{\mathcal{R}}^{obl} \leq \mathcal{C}_{\mathcal{R}}^{aon} \leq \mathcal{C}_{\mathcal{R}}^{aof}$$

Defintion 8 *Wir notieren die jeweils kleinsten Kompetitivitätskoeffizienten gegen unwissende Gegner, adaptive Offline Gegner bzw. adaptive Online Gegner für beliebige, randomisierte Algorithmen für das Online Paging Problem mit $\mathcal{C}^{obl}$, $\mathcal{C}^{aof}$ bzw. $\mathcal{C}^{aon}$.*

Den kleinsten Kompetitivitätskoeffizienten eines beliebigen, deterministischen Algorithmus notieren wir mit $\mathcal{C}^{det}$.

Es gilt

$$\mathcal{C}^{obl} \leq \mathcal{C}^{aon} \leq \mathcal{C}^{aof} \leq \mathcal{C}^{det}$$

Literatur

[1] *Randomized Algorithms*, Rajeev Motwani, Prabhakar Raghavan; Cambridge University Press (1995)

[2] *Tetris is Hard, Even to Approximate*, Erik D. Demaine, Susan Hohenberger, David Liben–Nowell; Laboratory for Computer Science, Massachusetts Institute of Technology (2002)