Approximations- und Online-Algorithmen

thgoebel@ethz.ch

ETH Zürich, FS 2022

This documents is a **short** summary for the course *Approximations- und Online-Algorithmen* at ETH Zurich. It is intended as a document for quick lookup, e.g. during revision, and as such does not replace attending the lecture, reading the slides or reading a proper book.

We do not guarantee correctness or completeness, nor is this document endorsed by the lecturers. Feel free to point out any errata, either by mail or on Github.

Contents

Part I.
Approximations-Algorithmen

1. Approximations-Algorithmen

TODO. Siehe das Skript von letzem Jahr.

Part II.
Online-Algorithmen

2. Das Paging-Problem

Konzepte

- Online-Problem, Online-Algorithmus, kompetitiver Faktor
- Paging-Problem

Motivation Probleme lösen ohne vollständige Informationen zu haben (die für eine optimale Lösung relevant sind). Stattdessen werden die Informationen stückweise zur Laufzeit bekannt.

Online-Problem Ein *Online-Minimierungsproblem* ist $\Pi = (I, O, cost, \min)$. Eine Eingabe $I = (x_1, ..., x_n) \in \mathcal{I}$ ist eine Folge von *Anfragen*. Eine akzeptierte Lösung $O = (y_1, ..., y_n)$ ist eine Folge von *Antworten*.

Beim analogen Maximierungsproblem spricht man statt von $cost(I, O)$ oft vom *Gewinn* $gain(I, O)$.

Online-Algorithmus Sei Π ein Online-Optimierungsproblem. Ein *Online-Algorithmus* $\mathcal{A}$ berechnet die Ausgabe $\mathcal{A}(I) = (y_1, ..., y_n)$ wobei y_i nur von $(x_1, ..., x_i)$ abhängt. $\mathcal{A}(I)$ ist eine zulässig Lösung für I.

Kompetitive Faktor (aka. competitive ratio, Wettbewerbsgüte, kompetitive Güte)
Ein Online-Algorithmus $\mathcal{A}$ ist *c-kompetitiv* falls gilt:

$$\exists \alpha \geq 0 \quad \forall I : \quad cost(\mathcal{A}(I)) \leq c \cdot cost(Opt(I)) + \alpha$$
$$\frac{cost(\mathcal{A}(I))}{cost(Opt(I))} + \alpha' \leq c$$

für ein Minimierungsproblem und α konstant. *Opt* ist ein optimaler Offline-Algorithmus, d.h. mit vollständiger Information.

Das kleinste c für das dies gilt heisst *kompetitiver Faktor*.
Für $\alpha = 0$ heisst $\mathcal{A}$ *strikt-c-kompetitiv*.
Falls $\mathcal{A}$ strikt-1-kompetitiv ist ($\alpha = 0, c = 1$) so heisst er *optimal*.

Ein Online-Algorithmus heisst *kompetitiv* wenn sein kompetititver Faktor nicht von der Länge der Eingabe abhängt. Wir sprechen dabei von *kompetitiver Analyse*. Der kompetitiver Faktor ist vergleichbar mit der Approximationsgüte von Approximationsalgorithmen.

Die Konstante α ist wichtig da sie erlaubt auf kurze Eingaben schlecht zu sein (und erst auf lange besser zu werden). [1]

[1] Warum brauchen wir bei der Approximationsgüte keine vergleichbare Konstante?

Paging

- Eingabe: $I = (x_1, ..., x_n)$ mit Speicher-Indizes $x_i \in \mathbb{N}$
- Hauptspeicher mit m Seiten: $(s_1, ..., s_m)$
- Cache-Speicher mit k Seiten: $B = (s_{j_1}, ..., s_{j_k})$, initialisiert mit $(s_1, ..., s_k)$ [2]
- Zeitschritt i:
  - Index x_i wird angefragt
  - Falls x_i im Cache (d.h. $s_{x_i} \in B$): return $y_i = 0$
  - Andernfalls: return $y_i = j$, und setze $B = B \setminus \{s_j\} \cup \{s_{x_i}\}$, d.h. lösche Seite s_j aus dem Cache. [3]
- $cost(\mathcal{A}(I)) := |\{i \mid y_i > 0\}|$
- goal := min

Strategien bei *Seitenfehlern (page faults)* zum *Verdrängen* von Seiten: First-in-First-Out (FIFO, wie eine Queue), Last-in-First-Out (LIFO, wie ein Stack), Least-Recently-Used (LRU), Longest-Forward-Distance (LFD, offline-only!).

Satz Ein Online-Algorithmus für Paging der FIFO nutzt ist strikt-k-kompetitiv.

Beweis: Gruppiere Zeitschritte in *Phasen*. Phase 1 endet nach dem ersten Seitenfehler. Phase $P \geq 2$ endet nach $1 + (P-1)k$ Seitenfehlern, d.h. alle k Fehler endet eine Phase und beginnt eine neue.

In Phase 1 machen *Opt* und $Fifo$ je genau einen Fehler (warum?).

Sei s die Seite die den letzten Seitenfehler von Phase $P-1$ verursacht (d.h. sie kommt neu in den Cache, und wird dank FIFO als letztes in Phase P verdrängt werden).
$\implies$ Zu Beginn von Phase P ist s im Cache von *Opt* und von $Fifo$.
$\implies$ Es gibt $\leq k-1$ Seiten die im Cache von *Opt* sind, aber nicht in dem von $Fifo$.
Während Phase P macht $Fifo$ genau k Fehler.
$\implies$ Während P muss *Opt* mindestens einen Seitenfehler machen.
$\implies$ $Fifo$ ist k-kompetitiv.

LRU ist in der Theorie ebenfalls k-kompetitiv, in der Praxis allerdings tendenziell besser als FIFO.

[2] Der Vorsprung eines selbstgewählten Startinhalts kann in α versteckt werden.
[3] Zusätzliches, proaktives Entfernen bringt keinen Vorteil.