Online-Algorithmen – Jun. Prof. Dr. Christian Sohler, Zusammenfassung von Florian Schoppmann

Zusammenfassung Online-Algorithmen

1. Einleitung

1.1 Grundbegriffe

Definition: Optimierungsproblem $\mathcal{P} = (\mathcal{I}, \mathcal{C})$, wobei $\mathcal{I}$ Menge von Eingaben und $\mathcal{C}$ Kostenfunktion. Jede Eingabe $I \in \mathcal{I}$ hat Menge von zulässigen Ausgaben $F(I)$, wobei mit jeder Ausgabe O die Kosten $C(I, O)$ assoziiert sind.
Ein Optimaler Algorithmus OPT ist durch $\text{OPT}[I] = \min_{O \in F(I)} C(I, O)$ definiert.

Definition: Ein Algorithmus heißt [stark] c-competitive, wenn für jede endl. Eingabesequenz I gilt:
$\text{ALG}[I] \leq c \cdot \text{OPT}[I] + \alpha$, wobei α konst.[, $\alpha \leq 0$]

Definition: Das Competitive Ratio eines Algorithmus ALG ist das Infimum über alle c mit der Eigenschaft, dass ALG c-competitive ist:
$\inf\{c \mid \text{ALG}(I) \leq c \cdot \text{OPT}(I) + \alpha,\ I \in \mathcal{I}\}$

1.2 Amortisierte Analyse

2. Das Listen-Zugriff-Problem

Zugriff auf Elemente einer verketteten Liste; sequentielles Durchsuchen; Zugriff auf Datensatz an Position x verursacht Kosten x; Einfügen $l + 1$ (l Länge der Liste); Löschen wie Zugriff; Kosten für Umorganisation: Transpositionen; Datensatz, auf den gerade zugegriffen wurde, kann ohne Kosten näher zum Listenanfang bewegt werden

2.1 Move-to-Front-Strategie (MTF)

Algorithmus: Nach jedem Zugriff oder Einfügen: Bewegen des Datensatzes an den Anfang.

Satz: MTF ist 2-competitive.

2.2 Untere Schranke für Listen-Zugriff

Satz: Jeder deterministische Online-Algorithmus für das statische Listen-Zugriff-Problem hat ein Competitive Ratio von mindestens $2 - \frac{2}{l+1}$. (l Länge der Liste)

2.3 Randomisierte Online-Algorithmen

Definition: Ein Algorithmus ALG ist c-competitive gegen einen blinden Gegenspieler, wenn für jede Eingabesequenz σ gilt:
$\text{E}[\mathbb{A}(\sigma)] \leq c \cdot \text{OPT}(\sigma) + \alpha$, α konst. Competitive Ratio analog zum deterministischen Fall.

Algorithmus BIT: Bit $b(x) \in \{0, 1\}$ für jeden Datensatz x. Zufällige gleichverteilte Initialisierung am Anfang. Umdrehen von $b(x)$ bei Zugriff auf x, falls anschließend $b(x) = 1$, Bewegen von x an den Anfang.

Satz: BIT ist $1,75$-competitive.

3. Paging

Definition: Seitenfehler-Modell: Lesen aus schnellem Speicher verursacht keine Kosten, ein Seitenfehler jedoch Kosten von 1.

3.1 Deterministische Online-Algorithmen

LRU: Least Recently Used
FIFO: First In First Out
LIFO: Last In First Out
LFU: Least Frequently Used: Seite, die am wenigsten angefragt wurde, seit sie im schnellen Speicher ist

Satz: LFD (Longest Forward Distance) ist ein optimaler Offline Algorithmus für das Paging Problem.

Lemma: Sei σ eine Anfragesequenz der Länge n. Sei OPT_i ein optimaler Paging Algorithmus für σ, der die ersten i Seitenfehler wie LFD bearbeitet. Dann gibt es einen optimalen Paging Algorithmus OPT_{i+1} für σ, der die erstn $i + 1$ Seitenfehler wie LFD behandelt.

Definition: Sei σ eine Eingabesequenz, die in folgende Phasen aufgeteilt werde: Phase 0 ist die leere Sequenz, alle weiteren Phasen $i, i > 0$ sind jeweils die maximal Sequenz, die auf Phase $i - 1$ folgt und Anfragen nach höchstens k verschiedenen Seiten enthält. Eine solche Unterteilung heißt k-Phasen Partition

Definition: Jede Seite im Speicher erhalte ein Markierungs-Bit. Zu Beginn einer Phase werden alle Markierungen gelöscht. Während einer Phase wird eine Seite markiert, wenn auf sie zugegriffen wird. Ein Markierungsalgorithmus entfernt niemals markierte Seiten aus dem schnellen Speicher.

Satz: Jeder Markierungsalgorithmus mit schnellem Speicher der Größe k ist k-competitive.

Lemma: LRU ist ein Markierungsalgorithmus (und damit k-competitive).

Satz: (über eine untere Schranke für deterministische Paging Algorithmen) Jeder deterministische Paging Algorithmus mit Speicher der Größe k hat ein Competitive Ratio von mindestens k.

Lemma: LIFO und LFU sind nicht competitive, wie folgende Sequenzen zeigen:
$\sigma = p_1, \ldots, p_k, p_{k+1}, \ldots, p_k, p_{k+1}$ bzw.
$\sigma = p_1^2, p_2^2, \ldots, p_{k-1}^2, p_k, p_{k+1}, \ldots, p_k, p_{k+1}$.

Satz: Ein Markierungsalgorithmus mit schnellem Speicher der Größe k ist gegenüber offline Algorithmen mit $h \leq k$ Speicher $\frac{k}{k-h+1}$ competitive.

3.2 Randomisierte Paging Algorithmen

MARK ist ein Markierungsalgorithmus, der bei Seitenfehlern aus der Menge der unmarkierten Seiten zufällig eine zur Ersetzung auswählt.

Satz: MARK mit schnellem Speicher der Größe k ist $2H_k$-competitive gegen einen blinden Gegenspieler. ($H_k := \sum_{i=1}^{k} \frac{1}{i}$ heißt „k-te hamonische Zahl“ und es gilt: $\ln k \leq H_k \leq \ln k + 1$.)

Satz: Jeder randomisierte Paging Algorithmus mit schnellem Speicher der Größe k und langsamem Speicher $\geq k+1$ hat ein Competitive Ratio von mindestens H_k.

4. Last Balancierung

Motivation: Es sollen Aufgaben an Maschinen so verteilt werden, dass die maximale Auslastung der Maschinen minimiert wird. Jede Aufgabe erzeugt auf den Maschinen eine Last (ggf. abhängig von der Maschine).

4.1 Identische Maschinen

Der Algorithmus GREEDY weist jedem eingehenden Job der Maschine (mit der kleinsten Nummer) zu, die zu diesem Zeitpunkt die niedrigste Last hat.

Satz: Für N Maschinen ist das Competitive Ratio von GREEDY genau $2 - \frac{1}{N}$.

4.2 Eingeschränkte Maschinen

Im Modell für eingeschränkte Maschinen existieren N identische Maschinen, wobei allerdings nicht jede Aufgabe jeder Maschine zugewiesen werden kann. Jeder Aufgabe ist daher eine Menge von zugelassenen Maschinen zugeordnet.

Satz: Für N eingeschränkte Maschinen ist GREEDY $(\lceil \log N \rceil + 1)$-competitive.

4.3 Verwandte Maschinen

Jede Maschine hat eine bestimmte Geschwindigkeit, so dass ein Job der Größe v_k auf verschiedenen Maschinen verschiedene Last erzeugen kann. Die auf Maschine i erzeugte Last ist dann $\frac{v_k}{\alpha_i}$, wobei α_i die Geschwindigkeit von Maschine i bezeichnet.

Unter der Annahme, dass $\text{OPT}(\sigma) \leq \Lambda$ ist, ordnet der Algorithmus SLOWFIT_Λ einen Job jeweils der langsamsten Maschine zu, die nach der Zuweisung noch eine Last $\leq 2\Lambda$ hat. Wir sagen, dass SLOWFIT_Λ einen Fehler macht, wenn es keine solche Maschine gibt.

Satz: Sei σ eine beliebige Eingabesequenz. Gilt $\text{OPT} \leq \Lambda$, so macht SLOWFIT_Λ keinen Fehler und es gilt $\text{SLOWFIT}_\Lambda \leq 2\Lambda$.

Aus SLOWFIT_Λ lässt sich der Algorithmus SLOWFIT konstruieren, der in Etappen abläuft: In Etappe $i, i = 0, 1, \ldots$ führen wir $\text{SLOWFIT}_{\Lambda_i}$ solange durch, bis er bei einer Anfrage r einen Fehler macht. Dann beginnt mit Anfrage r die nächste Etappe und es wird $\Lambda_{i+1} = 2\Lambda_i$ gesetzt. Wird in einer Etappe i kein Fehler gemacht, so wird die Zuweisung von $\text{SLOWFIT}_{\Lambda_i}(\sigma_i)$ benutzt. σ_i bezeichnet hierbei die Untersequenz der Eingabe, die in Etappe i abgearbeitet wird.

Satz: SLOWFIT ist 8-competitive.

5. Selbstanpassende Suchbäume

Rest fehlt noch