Das Online Rucksack-Problem

Seminararbeit - Theoretische Informatik

Marco Wettstein, ZHAW

2015-06-17

1 Begriffe

1.1 Online-Problem

Ein Online-Optimierungs-Problem (Maximierung, Minimierung) ist ein Optimierungs-Problem, bei welchem die Elemente des Inputs nicht von Anfang an bekannt sind, sondern nach und nach eintreffen.

Online-Optimierungs -Problem:	Eingabe als Sequenz von Anfragen $I = \{x_1, ..., x_n\}$ mögliche Antworten $O = \{y_1, \ldots, y_n\}$ für jede Eingabe. Die Ertragsfunktion $gain$, weisst jeder Eingabe I und zugehöriger Antwort O einen positiven Wert (Ertrag) zu. Wir nennen $OPT(I)$ eine optimale Lösung für I, wenn der zugehörige Ertrag maximal ist.

1.2 Online-Problem mit Advice

Hätten wir ein allwissendes Orakel, welches alle möglichen Antworten kennen würde, wieviele Hinweisen müssten wir von diesem Orakel erhalten, um eine optimale Lösung zu finden?

Wir nennen diese Hinweise advice-bits und definieren

Advice-Bits:	ϕ : (unbegrenztes) Advice-Band als Sequenz von bits

Wie gut schlägt sich ein solcher Algorithmus mit gegebener Anzahl Advice bits im Vergleich zu einem optimalen Offline-Algorithmus, der alle Eingaben kennt?

Advice-Komplexität, **c-kompetitiv**:	Eingabe I, *Online-Algorithmus mit Advice* A. $A^{\phi}(I)$ berechnet unter Verwendung von $\phi, x_1, \ldots, x_n$ eine Ausgabe $O = \{y1, \ldots, yn\}$ mit dem Ertrag $gain(A^{\phi}(I))$. A ist c-kompetitiv mit Advice-Komplexität $s(n)$, wenn $\exists\alpha$ Konstante, s.d. für jedes n und für jede Eingabe I mit Maximallänge n ein ϕ existiert, sodass $gain(A^{\phi}(I)) >= \frac{1}{c} * gain((OPT(I)) - \alpha$ und höchstens $s(n)$ bits von ϕ gelesen wurden während der Berechnung. Falls $\alpha = 0$, nennen wir A *strikt c-kompetitiv*.

1.3 Online-Rucksack Problem (Online KNAPSACK)

Wir erhalten nach und nach verschiedene Gegenstände mit jeweils einem Gewicht und einem Wert. Welche Gegenstände müssen wir in den Rucksack packen ohne ihn zu überfüllen um den maximal möglichen Gesamtwert zu erhalten?

Online-KNAPSACK:	Das Online-KNAPSACK-Problem ist das folgende Online Maximierungs-Problem: Eingabesequenz aus Tupeln $I = \{(w_1, v_1), (w_2, v_2), \ldots, (w_n, v_n)\}$, wobei $0 < w_i < 1$ und $v_i > 0$. Diese Eingabeelemente erscheinen nach und nach und ein zugehöriger Online-Algorithmus entscheidet für jedes Element, ob es Teil der Lösung ist. Eine mögliche Lösung ist eine Menge von Indizes $S' \subseteq \{1, ..n\}$, sodass $\sum_{i \in S'} w_i \leq$. Der zugehörige Ertrag ist $\sum_{i \in S'} v_i$, welcher maximiert werden soll.

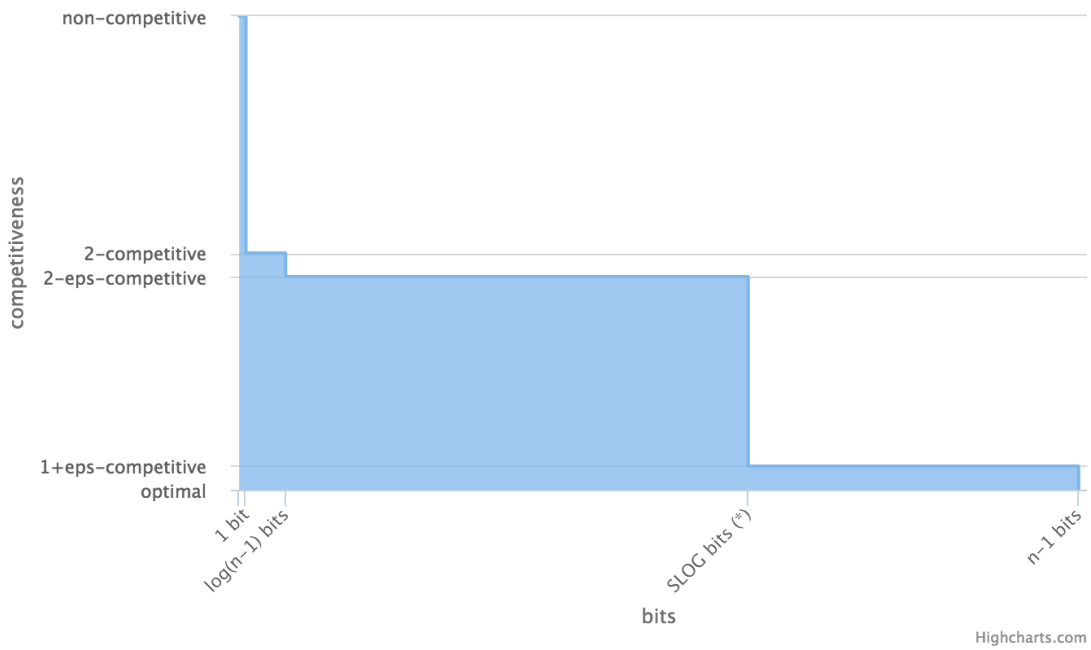
1.4 Das einfache Online-Rucksack-Problem (Simple Knapsack)

Falls der Wert eines Elements gleich ist wie dessen Gewicht, so sprechen wir vom einfachen Rucksackproblem.

SIMPLE-KNAPSACK:	Für jedes Element gilt $v_i = w_i$. Somit auch $\sum_{i \in S'} w_i = \sum_{i \in S'} v_i \leq 1$

In diesem Vortrag wird nur das einfache Online-Rucksack-Problem betrachtet.

2 Grenzen



(*) SLOG = $\left\lceil \frac{2\varepsilon+2}{\varepsilon} \right\rceil \cdot \lceil \log n \rceil + 2 \cdot \left\lceil \log \frac{2\varepsilon+2}{\varepsilon} \right\rceil + 2 \cdot \lceil \log \lceil \log n \rceil \rceil + 1$

Figure 1: Anzahl bits VS competitiveness.

3 Code & Live-Version

- Code: github.com/macrozone/seminar_np
- Live-Demo of the experiments: online-knapsack.macrozone.ch