Einleitung

STOCHASTIC SCHEDULING beschäftigt sich mit Problemen, bei denen die Laufzeiten eines Jobs durch Zufallsvariablen dargestellt sind. Dadurch ist die tatsächliche Zeit vor Ende des Jobs unbekannt. Das 'Scheduling' ist je nach Problem auf ein oder mehrere Prozessoren und mit verschiedenartigen Nebenbedingungen bestückt.

Fall: Ein Prozessor

- n Jobs auszuführen, ein Job ist eine beliebige Aufgabe die ein Prozessor berechnen kann, wobei die Jobs nicht gleich sein müssen
- Ein einzelnen Prozessor, zu jeder Zeit maximal ein Job
- Fragestellung: Welche Reihenfolge minimiert Zeit um alle Jobs auszuführen?
- Was ist Gesamtzeit? In der Realität sehr komplex $\Rightarrow$ Vereinfachung: alle Jobs unabhängig voneinander
- M: benötigte Gesamtzeit zur Abarbeitung aller Jobs
- Politik $\pi = (i_1, i_2, \ldots, i_n), \qquad i_j \neq i_k$ für $j \neq k, 0 < i_j \leq n$ Jobnummer
- X_j Zufallsvariable mit *Exponentialverteilung* mit Parameter λ_j
- $E(M) = E(X_{i_1}) + E(X_{i_2}) + \cdots + E(X_{i_n}) = \sum_{j=1}^{n} E(X_j)$

Fall: Zwei Prozessoren parallel

- Fall wie oben, Unterschied: Zwei identische Prozessoren
- Erster Ansatz: Weise jedem Job einen Startzeitpunkt und Prozessor zu
- Problem: kann nicht optimal sein wegen Idle-Zeiten
- Benutzung von Expertenwissen: Entscheidung welcher Job einzufügen ist kann während der Abarbeitung getroffen werden
- $\Rightarrow$ Regel: Füge Job ein sobald ein Prozessor unbeschäftigt, $(n+1)!$ verschiedene Politiken
- Minimierung der Zielfunktion $E_\pi(M) = \frac{E_\pi(D)+c}{2}$ durch Minimierung von D
- Weglassen der Zuweisung zu einem Prozessor durch Verfeinerung der Regel führt zu $n!$ Möglichkeiten
- Aufgabe aber noch nicht gelöst, da z.B. absteigende Reihenfolge von λ_i zu Maximierung von D führt
- Vergleich zweier ähnlicher Politiken (Job *1* und *2* vertauscht), Job *0* erster Job
- Längerer Beweis über Induktion durch Betrachtung von p_1 und den Fall mit $(n-1)$ Jobs, siehe Ausarbeitung
- Beweis, dass Lemma 1 auch bei Vertauschung von Job *1* und Job i_2 stimmt

- Beweis, dass Lemma 1 auch an beliebigen Stellen und beliebigen Jobs stimmt, sofern Job kleinstes λ besitzt
- Wiederholtes Anwenden führt zu Satz 1

Lemma 1: Seien zwei Politiken π und $\bar{\pi}$ gegeben:

$$\pi = (0, 2, 1, i_3, i_4, \ldots, i_n) \qquad \bar{\pi} = (0, 1, 2, i_3, i_4, \ldots, i_n)$$

Gelte außerdem, dass $\lambda_1 = min_{k \geq 1}(\lambda_k)$, dann gilt $E_{\bar{\pi}}(D) \leq E_{\pi}(D)$

Korollar 1: *Lemma 1* gilt auch für beliebige Politiken der Form $\pi = (0, 1, i_2, i_3, \ldots, i_n)$ und $\bar{\pi} = (0, i_2, 1, i_3 \ldots, i_n)$ mit $\lambda_{i_2} \geq \lambda_1$.

Lemma 2: Seien zwei Politiken π und $\bar{\pi}$ gegeben:

$$\pi = (0, i_1, i_2, \ldots, i_{k-2}, i_k, i_{k-1} = 1, i_{k+1}, \ldots, i_n) \qquad \bar{\pi} = (0, i_1, i_2, \ldots, i_{k-2}, i_{k-1} = 1, i_k, i_{k+1}, \ldots, i_n)$$

Gelte außerdem, dass $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$, dann gilt:

$$E_{\bar{\pi}}(D) \leq E_{\pi}(D)$$

Korollar 2: *Lemma 2* gilt auch für beliebige Jobs j sofern alle $\lambda_{i_k} \geq \lambda_j$ mit $i_k > j$, der Job j also genau der Job mit der niedrigsten Nummer aller Jobs $\geq j$ ist.

Satz 1: Eine aufsteigend sortierte Politik $\pi = (0, 1, 2, \ldots, n)$ ist die optimale Politik um den Erwartungswert für D zu minimieren, falls $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$ gilt.

Ein Prozessor mit begrenzter Zeit

- Wie oben mit Unterschied 1 Prozessor, jeder Job erziehlt einen Gewinn und Gesamtzeit ist begrenzt
- Ziel: Maximiere E_{π}(Gesamtgewinn) $= \sum_{j=0}^{n} \lambda_j R_j E_{\pi}$(Zeit die Job j bearbeitet wurde)

Satz 2: Der Gesamtgewinn bei gegebener Zeit T wird maximiert, wenn eine Politik gewählt wird, die die Jobs in einer absteigenden Folge von $\lambda_j R_j$ für $j = 1, \ldots, n$ sortiert.

Zwei Prozessoren mit begrenzter Zeit

- Fragestellung: Kombination aus Teil 2 und 3 (2 Prozessoren, Gesamtgewinn wie in 3), aber welche Reihenfolge optimal?
- Ergebnis: Nicht wie in Teil 3, Gegenbeispiel $\lambda_j R_j \equiv 1$für alle j und$\lambda_1 < \lambda_2 < \cdots < \lambda_n \Rightarrow$ absteigende Folge von $\lambda_j R_j$ minimiert nicht
- Aber: Unter etwas anderen Voraussetzung minimiert aufsteigende Folge von Jobs $1, 2, \ldots, n$ die Zeit

Satz 3:

$$\text{Sei } \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$$
$$\text{und } \lambda_1 R_1 \geq \lambda_2 R_2 \geq \cdots \geq \lambda_n R_n \geq 0$$

dann ist die Reihenfolge $(1, 2, \ldots, n)$ optimal und maximiert den zu erwartenden Gesamtgewinn für Zeit $t > 0$.

Ausblick und Zusammenfassung

Insgesamt: Aufstellen einer guten Form von Politik wichtig, Finden einer optimalen Lösung ohne starke Voraussetzungen sehr schwierig Bei realen Problemen liegt das Hauptaugenmerk auf das Aufstellen einer Politik, gelöst wird es meist mit probabalistischen Verfahren, die den durch die Politiken gegebenen Raum durchsuchen.