Wie lange soll ich es probieren? Zeitplanung bei unbekannten Aufgabenlängen

Sven Jäger, TU Berlin

Tag der Mathematik 2026 | 25. April 2026 | Berlin

Aufgaben mit unbekannter Dauer

Klassenstufe 11–13 29. Tag der Mathematik 2026

Team-Nummer: | Team-Name: | Punkte:

Aufgabe 1: Tadellose Zahlen *(10 Punkte)*

Eine Zahl $n \in \mathbb{N} \setminus \{1, 2, 3, \ldots\}$ heißt *tadellos*, wenn sie die Gleichung $n^2 \equiv (n-1) \cdot (n-2) \cdots 2 \cdot 1 + 1$ erfüllt.

a) Welche der Zahlen $n \in 2, 3, 4, 5$ sind tadellos und welche nicht?

b) Listet alle tadellosen Zahlen $n \geq 2$ auf. Begründet eure Antwort.

c) Sei $a \geq 5$, $a \cdot a \cdot b$ mit $a, b \in \mathbb{N}$, $a \geq b$. Zeigt: $b < \frac{a}{2}$.

d) Eine Zahl n heißt *respektabel*, wenn n^2 ein Teiler von $(n-1) \cdot (n-2) \cdots 1 + 1$ ist. Zeigt: Alle respektablen Zahlen $n \geq 2$ sind prim.

Klassenstufe 11–13 29. Tag der Mathematik 2026

Team-Nummer: | Team-Name: | Punkte:

Aufgabe 2: Scheibenwischer *(10 Punkte)*

Es soll die Fläche berechnet werden, die von verschiedenen Scheibenwischermodellen eines Fahrzeugs überstrichen wird. Die Wischerscheiben werden als ebene Flächen betrachtet.

a) Ein erstes Fahrzeug ist mit einem einfachen Wischblatt ausgestattet, das von einem Motor um $\alpha = 180°$ hin und her gedreht wird. Dieses Wischblatt durchquere Strecke $\overline{AB}$ modelliert. Sei A der Punkt auf $\overline{AB}$ mit $|OA| = 15\,\text{cm}$. Das Gummiwischerblatt wird durch die Strecke $\overline{AP}$ modelliert (siehe Abbildung 1).

Abbildung 1

Berechnet den Flächeninhalt der vom Wischerblatt überstrichenen Fläche, wobei davon ausgegangen wird, dass es um den Punkt O einen Winkel von 180° überstreicht.

b) Die Wischerscheibe eines zweiten Fahrzeugs besteht zwei Scheibenwischer, die durch zwei Strecken $\overline{OP}$ und $\overline{OP'}$ gleicher Länge r modelliert werden. Der eine dreht sich um einen Punkt O, der andere um einen Punkt O', wobei $|OO'| = r$ gilt (siehe Abbildung 2). Die Gummiwischerblätter bedecken die gesamte Länge der jeweiligen Strecke. Das äußere Ende jeder Strecke beschreibt einen Halbkreis oberhalb der Geraden OO'.

Abbildung 2

Berechnet den Flächeninhalt der von den Wischerblättern überstrichenen Fläche in Abhängigkeit von r.

c) Ein drittes Fahrzeug ist mit einem Scheibenwischer ausgestattet, dessen Metallhalterung durch die Verwendung zweier Strecken modelliert wird (siehe Abbildung 3): eine Strecke $\overline{AB}$, die das Gummiwischerblatt auf ihrer gesamten Länge trägt, und eine Strecke $\overline{OC}$, die den Drehpunkt O mit einem Punkt C auf der Strecke $\overline{AB}$ verbindet, so dass $\angle OCA = 30°$, $|CB| = 8 \cdot |CA|$ und $|OC| = \sqrt{7} \cdot |CA|$. Man setzt $|CA| = a$.

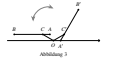


Abbildung 3

(i) Zeigt, dass das Dreieck AOC gleichschenklig ist.

Wenn der Scheibenwischer die Windschutzscheibe wischt, dreht er sich um den Punkt O. Zu Beginn des Wischvorgangs befindet sich das Gummiwischerblatt in horizontaler Position, dargestellt durch die Punkte A, B und C. Am Ende des Wischvorgangs befinden sich das Wischerblatt an den Positionen A', B' und C', so dass die Strecke $\overline{BB'}$ horizontal ist (siehe Abbildung 3).

(ii) Drückt den Flächeninhalt der von Wischerblatt überstrichene Fläche als Funktion von a aus.

Klassenstufe 11–13 29. Tag der Mathematik 2026

Team-Nummer: | Team-Name: | Punkte:

Aufgabe 3: Aufgabenverteilung *(10 Punkte)*

Bevor Elain und Sven zusammen Dinner essen gehen können, müssen sie noch n Aufgaben erledigen. Jede Aufgabe kann nur von einer Person alleine erledigt werden, und jeder kann nur an einer Aufgabe gleichzeitig arbeiten. Die Dauer jeder Aufgabe ist unabhängig davon, wer sie erledigt. Da die beiden hungrig sind und ihr Dinner nur zusammen essen wollen, ist ihr Ziel, möglichst früh alle Aufgaben erledigt zu haben. Anders ausgedrückt wollen sie die größere ihrer beiden Arbeitszeiten minimieren.

a) Elain und Sven wollen fünf Aufgaben erledigen, die jeweils 3 min, 5 min, 6 min, 7 min und 11 min dauern. Elain hat die Aufgaben unterschiedlicher Länge, um möglichst früh alle Aufgaben erledigt zu haben? Nach wie vielen Minuten sind dann alle Aufgaben fertig? Begründet, dass es nicht möglich ist, früher fertig zu werden.

b) Wir betrachten nun Aufgaben, die $1, 2, 4, \ldots, 2^{n-1}$ Minuten dauern. Was ist nun der frühestmögliche Zeitpunkt, bis zu dem alle Aufgaben abgeschlossen werden können? Begründet wieder, dass ein früherer Zeitpunkt nicht möglich ist.

Wir betrachten jetzt allgemeine Aufgabendauern (immer in ganzen Minuten). Anstatt sich eine optimale Aufgabenverteilung zu überlegen, einigen sich Elain und Sven darauf, dass jeder, wenn er gerade nichts zu tun hat und noch Aufgaben übrig sind, einfach mit irgendeiner noch nicht bearbeiteten Aufgabe weitermacht.

c) Findet ein Beispiel, in dem die beiden bei optimaler Aufgabenverteilung nach $t' \in \mathbb{N}$ Minuten fertig werden können, aber mit dem oben beschriebenen Vorgehen und einer ungünstigen Auswahl der jeweils nächsten Aufgaben erst nach $\frac{3}{2}t'$ + 10 Minuten fertig werden.

d) Sei nun t' die Dauer bis zum Abschluss der letzten Aufgabe bei einer optimalen Aufgabenverteilung. Beweist, dass Elain und Sven mit ihrer oben beschriebenen Aufteilungsstrategie für beliebige Aufgabendauern niemals länger als $\frac{3}{2}t'$ benötigen.

Klassenstufe 11–13 29. Tag der Mathematik 2026

Team-Nummer: | Team-Name: | Punkte:

Aufgabe 4: Türme *(10 Punkte)*

Als *Brett* bezeichnen wir eine quadratische Anordnung von n^2 Feldern, weißen und grauen, die n Zeilen und n Spalten der Länge n bilden. Die Abbildung zeigt vier Bretter A, B, C und D. Jeweils Brett vorgegeben, dann ist N_k die Anzahl der Möglichkeiten, k Türme auf weiße Felder des Bretts zu stellen, so dass sich die Türme paarweise nicht schlagen können, d.h., in jeder Zeile und Spalte des Bretts steht höchstens ein Turm. Für das Brett A gilt $N_1 = 6$, $N_2 = 7$ und $N_3 = 1$.

a) Bestimmt die Werte $N_1, N_2, N_3, N_4, N_5, N_6$ für das Brett B.

b) Bestimmt allgemein für $i = 1, \ldots, n$ die Werte N_i für ein Brett vom Typ B mit Seitenlänge n.

c) Bestimmt die Werte N_2 und N_3 für Brett C.

d) Bestimmt die Werte N_2, N_3, N_4, N_5 für ein Brett vom Typ D mit Seitenlänge 5. Das abgebildete Brett hat Seitenlänge 8.

e) Bestimmt allgemein für $i = 1, \ldots, n$ die Werte N_i für ein Brett vom Typ D mit Seitenlänge n.

Zielfunktion

- Beim Tag der Mathematik war das einzige Ziel, so viele Aufgaben wie möglich innerhalb der Bearbeitungszeit fertigzustellen.

Zielfunktion

- Beim Tag der Mathematik war das einzige Ziel, so viele Aufgaben wie möglich innerhalb der Bearbeitungszeit fertigzustellen. $\rightsquigarrow$ maximiere *Durchsatz*

Zielfunktion

- Beim Tag der Mathematik war das einzige Ziel, so viele Aufgaben wie möglich innerhalb der Bearbeitungszeit fertigzustellen. $\rightsquigarrow$ maximiere *Durchsatz*
- Andere Wettbewerbe (z. B. ICPC) berücksichtigen bei Gleichstand die Geschwindigkeit für die einzelnen Aufgaben.

Zielfunktion

- Beim Tag der Mathematik war das einzige Ziel, so viele Aufgaben wie möglich innerhalb der Bearbeitungszeit fertigzustellen. $\rightsquigarrow$ maximiere *Durchsatz*
- Andere Wettbewerbe (z. B. ICPC) berücksichtigen bei Gleichstand die Geschwindigkeit für die einzelnen Aufgaben. $\rightsquigarrow$ minimiere Summe der Fertigstellungszeiten

Zielfunktion

- Beim Tag der Mathematik war das einzige Ziel, so viele Aufgaben wie möglich innerhalb der Bearbeitungszeit fertigzustellen. $\rightsquigarrow$ maximiere *Durchsatz*
- Andere Wettbewerbe (z. B. ICPC) berücksichtigen bei Gleichstand die Geschwindigkeit für die einzelnen Aufgaben. $\rightsquigarrow$ minimiere Summe der Fertigstellungszeiten

Unser Ziel: Minimiere die Summe der Fertigstellungszeiten

Zielfunktion

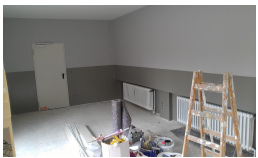
- Beim Tag der Mathematik war das einzige Ziel, so viele Aufgaben wie möglich innerhalb der Bearbeitungszeit fertigzustellen. $\rightsquigarrow$ maximiere *Durchsatz*
- Andere Wettbewerbe (z. B. ICPC) berücksichtigen bei Gleichstand die Geschwindigkeit für die einzelnen Aufgaben. $\rightsquigarrow$ minimiere Summe der Fertigstellungszeiten

Unser Ziel: Minimiere die Summe der Fertigstellungszeiten

Weitere Anwendungen

Renovierungsarbeiten

Jede fertiggestellte Wohnung bringt Mieteinnahmen.



Zielfunktion

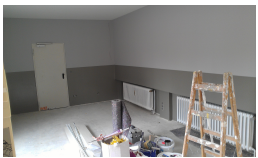
- Beim Tag der Mathematik war das einzige Ziel, so viele Aufgaben wie möglich innerhalb der Bearbeitungszeit fertigzustellen. $\rightsquigarrow$ maximiere *Durchsatz*
- Andere Wettbewerbe (z. B. ICPC) berücksichtigen bei Gleichstand die Geschwindigkeit für die einzelnen Aufgaben. $\rightsquigarrow$ minimiere Summe der Fertigstellungszeiten

Unser Ziel: Minimiere die Summe der Fertigstellungszeiten

Weitere Anwendungen

Renovierungsarbeiten

Jede fertiggestellte Wohnung bringt Mieteinnahmen.



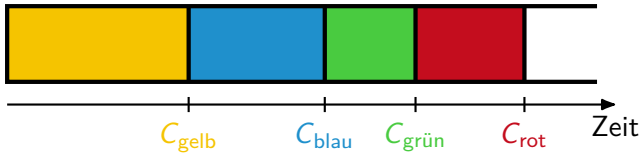
Cloud Computing

Jeder Kunde möchte sein Ergebnis so früh wie möglich.

Vorbereitung: Bekannte Aufgabendauern

Gegeben: Aufgaben j mit Bearbeitungszeiten $p_j \geq 0$.

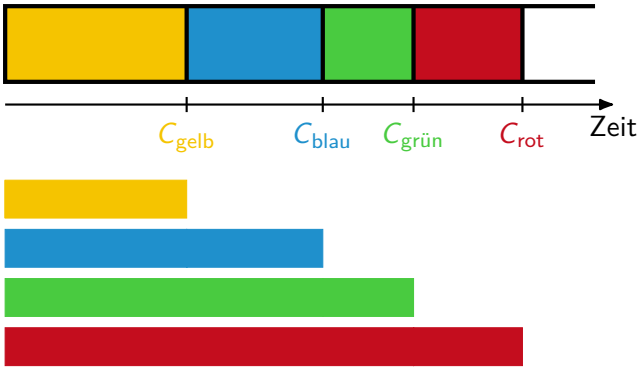
Gesucht: Bearbeitungsreihenfolge, die die Summe der Fertigstellungszeiten C_j minimiert.



Vorbereitung: Bekannte Aufgabendauern

Gegeben: Aufgaben j mit Bearbeitungszeiten $p_j \geq 0$.

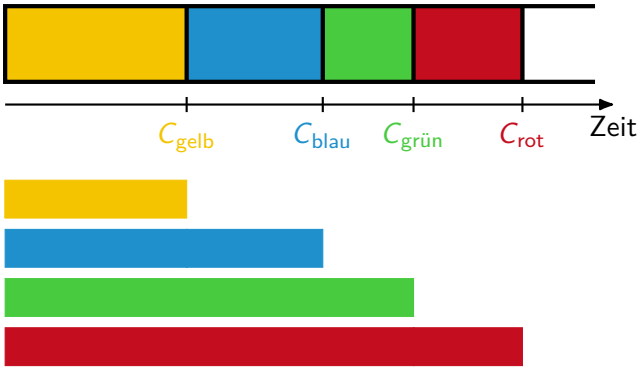
Gesucht: Bearbeitungsreihenfolge, die die Summe der Fertigstellungszeiten C_j minimiert.



Vorbereitung: Bekannte Aufgabendauern

Gegeben: Aufgaben j mit Bearbeitungszeiten $p_j \geq 0$.

Gesucht: Bearbeitungsreihenfolge, die die Summe der Fertigstellungszeiten C_j minimiert.



Frage: Was ist die beste Bearbeitungsreihenfolge?

Kürzeste Aufgabe Zuerst

Satz 1.

Die Aufgaben in aufsteigender Länge zu bearbeiten minimiert die Summe der Fertigstellungszeiten.

Kürzeste Aufgabe Zuerst

Satz 1.

Die Aufgaben in aufsteigender Länge zu bearbeiten minimiert die Summe der Fertigstellungszeiten.

Beweis.

Austauschargument:

Kürzeste Aufgabe Zuerst

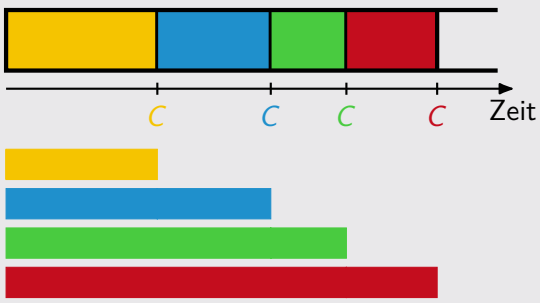
Satz 1.

Die Aufgaben in aufsteigender Länge zu bearbeiten minimiert die Summe der Fertigstellungszeiten.

Beweis.

Austauschargument:

Angenommen, eine längere Aufgabe wird direkt vor einer kürzeren bearbeitet.



□

Kürzeste Aufgabe Zuerst

Satz 1.

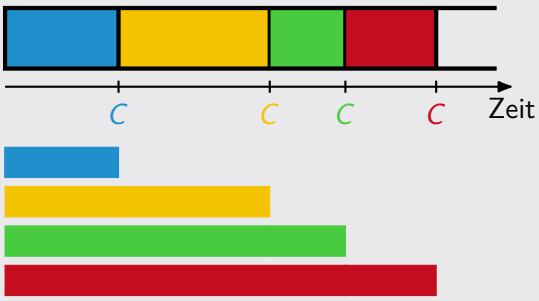
Die Aufgaben in aufsteigender Länge zu bearbeiten minimiert die Summe der Fertigstellungszeiten.

Beweis.

Austauschargument:

Angenommen, eine längere Aufgabe wird direkt vor einer kürzeren bearbeitet.

Wenn man die beiden Aufgaben vertauscht, bleibt die zweite Fertigstellungszeit gleich, aber die erste wird kleiner.



□

Kürzeste Aufgabe Zuerst

Satz 1.

Die Aufgaben in aufsteigender Länge zu bearbeiten minimiert die Summe der Fertigstellungszeiten.

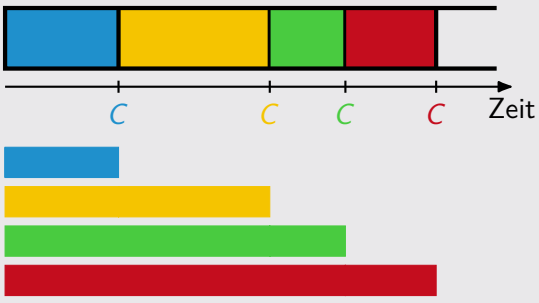
Beweis.

Austauschargument:

Angenommen, eine längere Aufgabe wird direkt vor einer kürzeren bearbeitet.

Wenn man die beiden Aufgaben vertauscht, bleibt die zweite Fertigstellungszeit gleich, aber die erste wird kleiner.

Also kann dies in der optimalen Reihenfolge nicht vorkommen.



□

Unbekannte Aufgabenlängen

In der Realität wissen wir oft nicht, wie lange eine Aufgabe dauert.

Unbekannte Aufgabenlängen

In der Realität wissen wir oft nicht, wie lange eine Aufgabe dauert.

Idee: Bearbeite Aufgaben in irgendeiner Reihenfolge.

Unbekannte Aufgabenlängen

In der Realität wissen wir oft nicht, wie lange eine Aufgabe dauert.

Idee: Bearbeite Aufgaben in irgendeiner Reihenfolge.

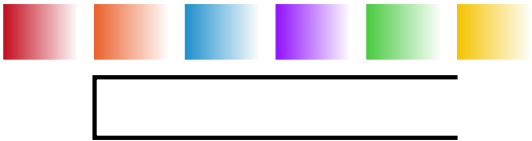
Dies kann beliebig schlecht sein.

Unbekannte Aufgabenlängen

In der Realität wissen wir oft nicht, wie lange eine Aufgabe dauert.

Idee: Bearbeite Aufgaben in irgendeiner Reihenfolge.

Dies kann beliebig schlecht sein.

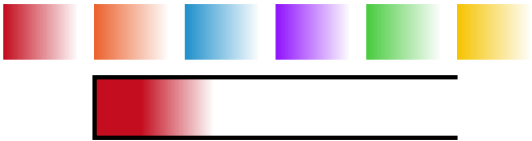


Unbekannte Aufgabenlängen

In der Realität wissen wir oft nicht, wie lange eine Aufgabe dauert.

Idee: Bearbeite Aufgaben in irgendeiner Reihenfolge.

Dies kann beliebig schlecht sein.

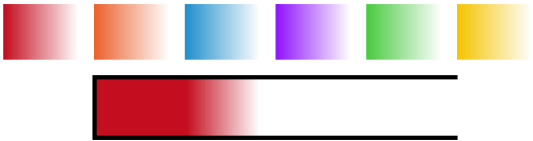


Unbekannte Aufgabenlängen

In der Realität wissen wir oft nicht, wie lange eine Aufgabe dauert.

Idee: Bearbeite Aufgaben in irgendeiner Reihenfolge.

Dies kann beliebig schlecht sein.

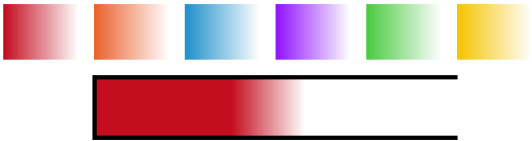


Unbekannte Aufgabenlängen

In der Realität wissen wir oft nicht, wie lange eine Aufgabe dauert.

Idee: Bearbeite Aufgaben in irgendeiner Reihenfolge.

Dies kann beliebig schlecht sein.

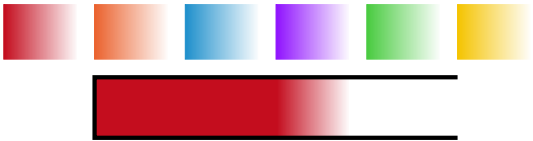


Unbekannte Aufgabenlängen

In der Realität wissen wir oft nicht, wie lange eine Aufgabe dauert.

Idee: Bearbeite Aufgaben in irgendeiner Reihenfolge.

Dies kann beliebig schlecht sein.



Unbekannte Aufgabenlängen

In der Realität wissen wir oft nicht, wie lange eine Aufgabe dauert.

Idee: Bearbeite Aufgaben in irgendeiner Reihenfolge.

Dies kann beliebig schlecht sein.

Unbekannte Aufgabenlängen

In der Realität wissen wir oft nicht, wie lange eine Aufgabe dauert.

Idee: Bearbeite Aufgaben in irgendeiner Reihenfolge.

Dies kann beliebig schlecht sein.

Unbekannte Aufgabenlängen

In der Realität wissen wir oft nicht, wie lange eine Aufgabe dauert.

Idee: Bearbeite Aufgaben in irgendeiner Reihenfolge.

Dies kann beliebig schlecht sein.

Unbekannte Aufgabenlängen

In der Realität wissen wir oft nicht, wie lange eine Aufgabe dauert.

Idee: Bearbeite Aufgaben in irgendeiner Reihenfolge.

Dies kann beliebig schlecht sein.

Frage: Wie würdet ihr vorgehen?

Unbekannte Aufgabenlängen

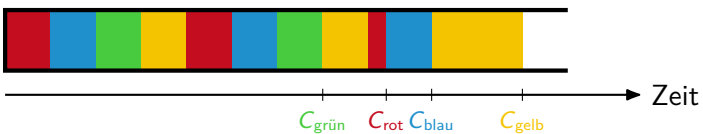
In der Realität wissen wir oft nicht, wie lange eine Aufgabe dauert.

Idee: Bearbeite Aufgaben in irgendeiner Reihenfolge.

Dies kann beliebig schlecht sein.

Frage: Wie würdet ihr vorgehen?

Idee: Wechsle zwischen verschiedenen Aufgaben hin und her.



Unbekannte Aufgabenlängen

In der Realität wissen wir oft nicht, wie lange eine Aufgabe dauert.

Idee: Bearbeite Aufgaben in irgendeiner Reihenfolge.

Dies kann beliebig schlecht sein.

Frage: Wie würdet ihr vorgehen?

Idee: Wechsle zwischen verschiedenen Aufgaben hin und her.

Unbekannte Aufgabenlängen

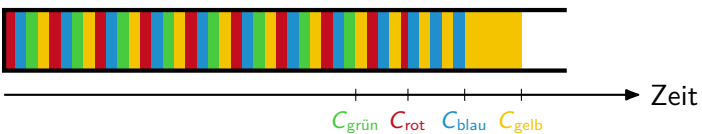
In der Realität wissen wir oft nicht, wie lange eine Aufgabe dauert.

Idee: Bearbeite Aufgaben in irgendeiner Reihenfolge.

Dies kann beliebig schlecht sein.

Frage: Wie würdet ihr vorgehen?

Idee: Wechsle zwischen verschiedenen Aufgaben hin und her.



Unbekannte Aufgabenlängen

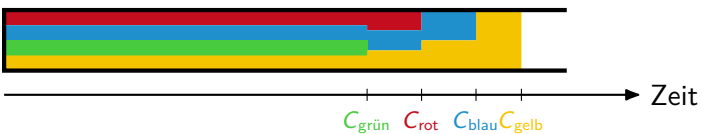
In der Realität wissen wir oft nicht, wie lange eine Aufgabe dauert.

Idee: Bearbeite Aufgaben in irgendeiner Reihenfolge.

Dies kann beliebig schlecht sein.

Frage: Wie würdet ihr vorgehen?

Idee: Wechsle zwischen verschiedenen Aufgaben hin und her.



Parallele Bearbeitung

Immer schnellere Wechsel $\longrightarrow$ Aufgaben werden gleichzeitig bearbeitet.

Parallele Bearbeitung

Immer schnellere Wechsel $\longrightarrow$ Aufgaben werden gleichzeitig bearbeitet.

Das formalisieren wir jetzt:

Parallele Bearbeitung

Immer schnellere Wechsel $\longrightarrow$ Aufgaben werden gleichzeitig bearbeitet.

Das formalisieren wir jetzt:

- Zu jedem Zeitpunkt t kann jede Aufgabe j mit einer Rate $y_j(t) \in [0, 1]$ bearbeitet werden.

Parallele Bearbeitung

Immer schnellere Wechsel $\longrightarrow$ Aufgaben werden gleichzeitig bearbeitet.

Das formalisieren wir jetzt:

- Zu jedem Zeitpunkt t kann jede Aufgabe j mit einer Rate $y_j(t) \in [0, 1]$ bearbeitet werden.
- Zu jedem Zeitpunkt t muss $y_1(t) + y_2(t) + \cdots + y_n(t) \leq 1$ gelten.

Parallele Bearbeitung

Immer schnellere Wechsel $\longrightarrow$ Aufgaben werden gleichzeitig bearbeitet.

Das formalisieren wir jetzt:

- Zu jedem Zeitpunkt t kann jede Aufgabe j mit einer Rate $y_j(t) \in [0, 1]$ bearbeitet werden.
- Zu jedem Zeitpunkt t muss $y_1(t) + y_2(t) + \cdots + y_n(t) \leq 1$ gelten.
- Der bis zum Zeitpunkt t gemachte Fortschritt an Aufgabe j ist $Y_j(t) = \int_0^t y_j(s)\,\mathrm{d}s$.

Parallele Bearbeitung

Immer schnellere Wechsel $\longrightarrow$ Aufgaben werden gleichzeitig bearbeitet.

Das formalisieren wir jetzt:

- Zu jedem Zeitpunkt t kann jede Aufgabe j mit einer Rate $y_j(t) \in [0, 1]$ bearbeitet werden.
- Zu jedem Zeitpunkt t muss $y_1(t) + y_2(t) + \cdots + y_n(t) \leq 1$ gelten.
- Der bis zum Zeitpunkt t gemachte Fortschritt an Aufgabe j ist $Y_j(t) = \int_0^t y_j(s)\,\mathrm{d}s$.
- Eine Aufgabe ist fertig, wenn $Y_j(t) = p_j$ ist.

Parallele Bearbeitung

Immer schnellere Wechsel $\longrightarrow$ Aufgaben werden gleichzeitig bearbeitet.

Das formalisieren wir jetzt:

- Zu jedem Zeitpunkt t kann jede Aufgabe j mit einer Rate $y_j(t) \in [0, 1]$ bearbeitet werden.
- Zu jedem Zeitpunkt t muss $y_1(t) + y_2(t) + \cdots + y_n(t) \leq 1$ gelten.
- Der bis zum Zeitpunkt t gemachte Fortschritt an Aufgabe j ist $Y_j(t) = \int_0^t y_j(s)\,\mathrm{d}s$.
- Eine Aufgabe ist fertig, wenn $Y_j(t) = p_j$ ist.

Zu jedem Zeitpunkt t sei $J(t)$ die Menge der verbleibenden Aufgaben.

Parallele Bearbeitung

Immer schnellere Wechsel $\longrightarrow$ Aufgaben werden gleichzeitig bearbeitet.

Das formalisieren wir jetzt:

- Zu jedem Zeitpunkt t kann jede Aufgabe j mit einer Rate $y_j(t) \in [0,1]$ bearbeitet werden.
- Zu jedem Zeitpunkt t muss $y_1(t) + y_2(t) + \cdots + y_n(t) \leq 1$ gelten.
- Der bis zum Zeitpunkt t gemachte Fortschritt an Aufgabe j ist $Y_j(t) = \int_0^t y_j(s)\,\mathrm{d}s$.
- Eine Aufgabe ist fertig, wenn $Y_j(t) = p_j$ ist.

Zu jedem Zeitpunkt t sei $J(t)$ die Menge der verbleibenden Aufgaben.

Algorithm (Round-Robin).
Zu jedem Zeitpunkt t bearbeite alle Aufgaben aus $J(t)$ mit derselben Rate $y_j(t) := \frac{1}{|J(t)|}$.

Wie gut ist diese Strategie?

Natürlich ist die Summe der Fertigstellungszeiten im Allgemeinen größer als wenn wir die Aufgabenlängen kennen.

Wie gut ist diese Strategie?

Natürlich ist die Summe der Fertigstellungszeiten im Allgemeinen größer als wenn wir die Aufgabenlängen kennen.

Wie können wir die Qualität eines Verfahrens ohne vollständige Information messen?

Wie gut ist diese Strategie?

Natürlich ist die Summe der Fertigstellungszeiten im Allgemeinen größer als wenn wir die Aufgabenlängen kennen.

Wie können wir die Qualität eines Verfahrens ohne vollständige Information messen?

Idee: Vergleiche unsere Lösung mit der optimalen Lösung und betrachten das Verhältnis im schlimmsten Fall.

Wie gut ist diese Strategie?

Natürlich ist die Summe der Fertigstellungszeiten im Allgemeinen größer als wenn wir die Aufgabenlängen kennen.

Wie können wir die Qualität eines Verfahrens ohne vollständige Information messen?

Idee: Vergleiche unsere Lösung mit der optimalen Lösung und betrachten das Verhältnis im schlimmsten Fall.

Dieses Maß heißt der Competitive Ratio.

Competitive Ratio von Round-Robin

Satz 2.

Unter Verwendung des Round-Robin-Verfahrens ist die Summe der Fertigstellungszeiten höchstens doppelt so hoch wie in dem bestmöglichen Schedule.

Competitive Ratio von Round-Robin

Satz 2.

Unter Verwendung des Round-Robin-Verfahrens ist die Summe der Fertigstellungszeiten höchstens doppelt so hoch wie in dem bestmöglichen Schedule.

- Völlige Unwissenheit über die Aufgabendauer verschlechtert unsere Zielfunktion ‘nur’ um einen Faktor 2.

Competitive Ratio von Round-Robin

Satz 2.

Unter Verwendung des Round-Robin-Verfahrens ist die Summe der Fertigstellungszeiten höchstens doppelt so hoch wie in dem bestmöglichen Schedule.

- Völlige Unwissenheit über die Aufgabendauer verschlechtert unsere Zielfunktion 'nur' um einen Faktor 2.
- In der Praxis wissen wir oft ein bisschen über die Dauern. Dadurch können wir nur besser werden.

Competitive Ratio von Round-Robin

Satz 2.

Unter Verwendung des Round-Robin-Verfahrens ist die Summe der Fertigstellungszeiten höchstens doppelt so hoch wie in dem bestmöglichen Schedule.

- Völlige Unwissenheit über die Aufgabendauer verschlechtert unsere Zielfunktion 'nur' um einen Faktor 2.
- In der Praxis wissen wir oft ein bisschen über die Dauern. Dadurch können wir nur besser werden.
- Solche weniger extremen Modelle sind Gegenstand aktueller Forschung:

Competitive Ratio von Round-Robin

Satz 2.

Unter Verwendung des Round-Robin-Verfahrens ist die Summe der Fertigstellungszeiten höchstens doppelt so hoch wie in dem bestmöglichen Schedule.

- Völlige Unwissenheit über die Aufgabendauer verschlechtert unsere Zielfunktion 'nur' um einen Faktor 2.
- In der Praxis wissen wir oft ein bisschen über die Dauern. Dadurch können wir nur besser werden.
- Solche weniger extremen Modelle sind Gegenstand aktueller Forschung:
  - Stochastisches Scheduling

Competitive Ratio von Round-Robin

Satz 2.

Unter Verwendung des Round-Robin-Verfahrens ist die Summe der Fertigstellungszeiten höchstens doppelt so hoch wie in dem bestmöglichen Schedule.

- Völlige Unwissenheit über die Aufgabendauer verschlechtert unsere Zielfunktion 'nur' um einen Faktor 2.
- In der Praxis wissen wir oft ein bisschen über die Dauern. Dadurch können wir nur besser werden.
- Solche weniger extremen Modelle sind Gegenstand aktueller Forschung:
  - Stochastisches Scheduling
  - Scheduling mit Vorhersagen

Competitive Ratio von Round-Robin

Satz 2.

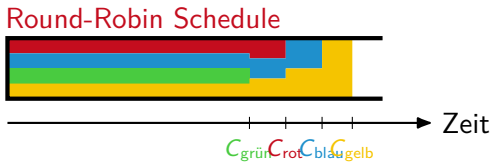
Unter Verwendung des Round-Robin-Verfahrens ist die Summe der Fertigstellungszeiten höchstens doppelt so hoch wie in dem bestmöglichen Schedule.

- Völlige Unwissenheit über die Aufgabendauer verschlechtert unsere Zielfunktion 'nur' um einen Faktor 2.
- In der Praxis wissen wir oft ein bisschen über die Dauern. Dadurch können wir nur besser werden.
- Solche weniger extremen Modelle sind Gegenstand aktueller Forschung:
  - Stochastisches Scheduling
  - Scheduling mit Vorhersagen
  - Scheduling mit Signal, wenn man fast fertig ist

Beweisskizze

Hauptbeobachtung: Wenn eine Aufgabe j im Round-Robin-Schedule fertig wird, wurden

- alle kürzeren Aufgaben k vollständig bearbeitet,
- alle längeren Aufgaben k genauso viel wie j bearbeitet: $Y_k(C_j) = p_j$.



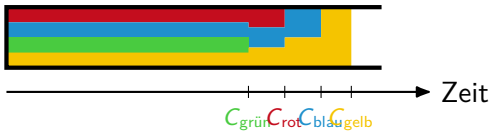
Beweisskizze

Hauptbeobachtung: Wenn eine Aufgabe j im Round-Robin-Schedule fertig wird, wurden

- alle kürzeren Aufgaben k vollständig bearbeitet,
- alle längeren Aufgaben k genauso viel wie j bearbeitet: $Y_k(C_j) = p_j$.

Wir nummerieren die Aufgaben so, dass $p_1 \leq p_2 \leq \cdots \leq p_n$.

Round-Robin Schedule



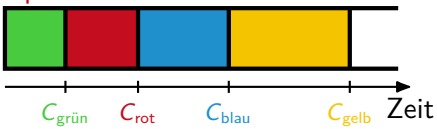
Beweisskizze

Hauptbeobachtung: Wenn eine Aufgabe j im Round-Robin-Schedule fertig wird, wurden

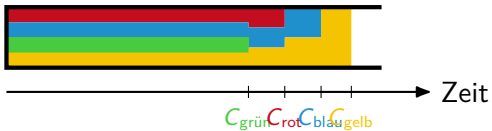
- alle kürzeren Aufgaben k vollständig bearbeitet,
- alle längeren Aufgaben k genauso viel wie j bearbeitet: $Y_k(C_j) = p_j$.

Wir nummerieren die Aufgaben so, dass $p_1 \leq p_2 \leq \cdots \leq p_n$.

Optimaler Schedule



Round-Robin Schedule



Für jedes Paar $j \leq k$ verzögert Aufgabe j die Aufgabe k um p_j.

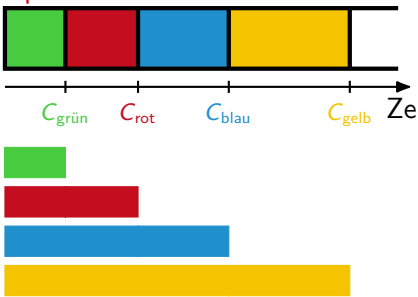
Beweisskizze

Hauptbeobachtung: Wenn eine Aufgabe j im Round-Robin-Schedule fertig wird, wurden

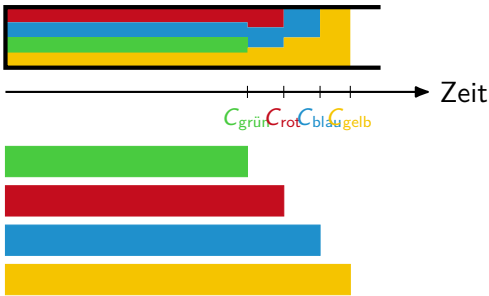
- alle kürzeren Aufgaben k vollständig bearbeitet,
- alle längeren Aufgaben k genauso viel wie j bearbeitet: $Y_k(C_j) = p_j$.

Wir nummerieren die Aufgaben so, dass $p_1 \leq p_2 \leq \cdots \leq p_n$.

Optimaler Schedule



Round-Robin Schedule



Für jedes Paar $j \leq k$ verzögert Aufgabe j die Aufgabe k um p_j.

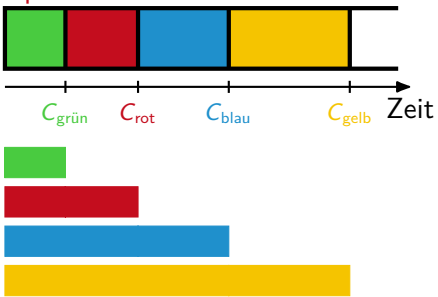
Beweisskizze

Hauptbeobachtung: Wenn eine Aufgabe j im Round-Robin-Schedule fertig wird, wurden

- alle kürzeren Aufgaben k vollständig bearbeitet,
- alle längeren Aufgaben k genauso viel wie j bearbeitet: $Y_k(C_j) = p_j$.

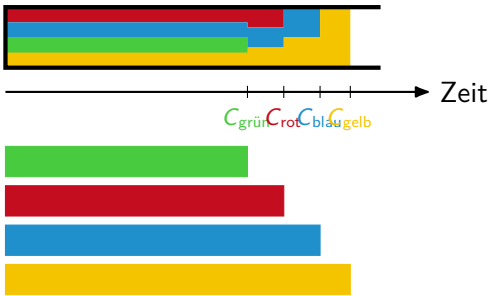
Wir nummerieren die Aufgaben so, dass $p_1 \leq p_2 \leq \cdots \leq p_n$.

Optimaler Schedule



Für jedes Paar $j \leq k$ verzögert Aufgabe j die Aufgabe k um p_j.

Round-Robin Schedule



Für $j \leq k$ verzögert Aufgabe j die Aufgabe k um p_j und Aufgabe k die Aufgabe j um p_j.

Multitasking-Overhead

Frage: Würdet ihr das Round-Robin-Verfahren für einen Wettbewerb anwenden?

Multitasking-Overhead

Frage: Würdet ihr das Round-Robin-Verfahren für einen Wettbewerb anwenden?

Problem:

- Man kann nicht unendlich schnell zwischen Aufgaben wechseln.

Multitasking-Overhead

Frage: Würdet ihr das Round-Robin-Verfahren für einen Wettbewerb anwenden?

Problem:

- Man kann nicht unendlich schnell zwischen Aufgaben wechseln.
- Bei jedem Wechsel muss man sich wieder in die Aufgabe einarbeiten und verliert Zeit.

Multitasking-Overhead

Frage: Würdet ihr das Round-Robin-Verfahren für einen Wettbewerb anwenden?

Problem:

- Man kann nicht unendlich schnell zwischen Aufgaben wechseln.
- Bei jedem Wechsel muss man sich wieder in die Aufgabe einarbeiten und verliert Zeit.

Im Folgenden betrachten wir das Extremmodell, das wir nach jedem Wechsel wieder komplett neu beginnen müssen.

- Wir denken über eine Aufgabe nach, ohne uns Notizen zu machen.
- Wenn wir über eine andere Aufgabe nachdenken, vergessen wir sofort alles von der ersten Aufgabe.

Neustart-Strategien

- Wenn wir immer wieder wechseln, müssen wir jedes Mal von vorne anfangen.

Neustart-Strategien

- Wenn wir immer wieder wechseln, müssen wir jedes Mal von vorne anfangen.
- Wenn wir nicht wechseln, können wir an der einzigen langen Aufgabe sitzen und beliebig schlecht im Vergleich zum Optimum sein.

Neustart-Strategien

- Wenn wir immer wieder wechseln, müssen wir jedes Mal von vorne anfangen.
- Wenn wir nicht wechseln, können wir an der einzigen langen Aufgabe sitzen und beliebig schlecht im Vergleich zum Optimum sein.

Frage: Wie schlimm ist Vergesslichkeit?

- Gibt es eine Strategie, mit der wir nur einen konstanten Faktor schlechter als das Optimum sind,
- oder ist das grundsätzlich unmöglich?

Neustart-Strategien

- Wenn wir immer wieder wechseln, müssen wir jedes Mal von vorne anfangen.
- Wenn wir nicht wechseln, können wir an der einzigen langen Aufgabe sitzen und beliebig schlecht im Vergleich zum Optimum sein.

Frage: Wie schlimm ist Vergesslichkeit?

- Gibt es eine Strategie, mit der wir nur einen konstanten Faktor schlechter als das Optimum sind,
- oder ist das grundsätzlich unmöglich?

Antwort: Es geht – mit exponentiellem Wachstum!

Verdopplungs-Strategie

- Bearbeite erst jede Aufgabe bis zu eine Zeiteinheit lang.

Verdopplungs-Strategie

- Bearbeite erst jede Aufgabe bis zu eine Zeiteinheit lang.
- Bearbeite dann jede nicht fertig gewordene Aufgabe bis zu zwei Zeiteinheiten.

Verdopplungs-Strategie

- Bearbeite erst jede Aufgabe bis zu eine Zeiteinheit lang.
- Bearbeite dann jede nicht fertig gewordene Aufgabe bis zu zwei Zeiteinheiten.
- Dann 4, 8, 16, ...

Verdopplungs-Strategie

- Bearbeite erst jede Aufgabe bis zu eine Zeiteinheit lang.
- Bearbeite dann jede nicht fertig gewordene Aufgabe bis zu zwei Zeiteinheiten.
- Dann 4, 8, 16, ...

Beobachtung: Jede Aufgabe wird irgendwann fertig.

Analyse der Verdopplungs-Strategie

Satz 3.

Unter Verwendung der Verdopplungs-Strategie ist die Summe der Fertigstellungszeiten höchstens 8 mal so hoch wie im bestmöglichen Schedule.

Analyse der Verdopplungs-Strategie

Satz 3.

Unter Verwendung der Verdopplungs-Strategie ist die Summe der Fertigstellungszeiten höchstens 8 mal so hoch wie im bestmöglichen Schedule.

Frage: Wie sieht der bestmögliche Schedule aus, wenn man alle Längen kennt?

Analyse der Verdopplungs-Strategie

Satz 3.

Unter Verwendung der Verdopplungs-Strategie ist die Summe der Fertigstellungszeiten höchstens 8 mal so hoch wie im bestmöglichen Schedule.

Frage: Wie sieht der bestmögliche Schedule aus, wenn man alle Längen kennt?

- Man wird niemals eine Aufgabe neu starten.

Analyse der Verdopplungs-Strategie

Satz 3.

Unter Verwendung der Verdopplungs-Strategie ist die Summe der Fertigstellungszeiten höchstens 8 mal so hoch wie im bestmöglichen Schedule.

Frage: Wie sieht der bestmögliche Schedule aus, wenn man alle Längen kennt?

- Man wird niemals eine Aufgabe neu starten.
- Man bearbeitet die Aufgaben von der kürzesten zur längsten.

Analyse der Verdopplungs-Strategie

Satz 3.

Unter Verwendung der Verdopplungs-Strategie ist die Summe der Fertigstellungszeiten höchstens 8 mal so hoch wie im bestmöglichen Schedule.

Frage: Wie sieht der bestmögliche Schedule aus, wenn man alle Längen kennt?

- Man wird niemals eine Aufgabe neu starten.
- Man bearbeitet die Aufgaben von der kürzesten zur längsten.

Beweis

Wir wollen zeigen, dass

$$\sum_{j=1}^{n} C_j^{\text{Verdopplung}} \leq 8 \cdot \sum_{j=1}^{n} C_j^{\text{optimal}}.$$

Analyse der Verdopplungs-Strategie

Beweis (Fortsetzung)

Wir nehmen erstmal an, dass alle Bearbeitungszeiten Zweierpotenzen sind.

Analyse der Verdopplungs-Strategie

Beweis (Fortsetzung)

Wir nehmen erstmal an, dass alle Bearbeitungszeiten Zweierpotenzen sind.

$\Longrightarrow$ In jeder Runde arbeiten wir gleich lang an jeder noch nicht fertigen Aufgabe, egal, ob sie fertig wird oder nicht.

Analyse der Verdopplungs-Strategie

Beweis (Fortsetzung)

Wir nehmen erstmal an, dass alle Bearbeitungszeiten Zweierpotenzen sind.

$\Longrightarrow$ In jeder Runde arbeiten wir gleich lang an jeder noch nicht fertigen Aufgabe, egal, ob sie fertig wird oder nicht.

Die Gesamtzeit, die wir an einer Aufgabe j der Länge $p_j = 2^{q_j}$ arbeiten, ist

$$1 + 2 + 4 + \cdots + 2^{q_j} = \sum_{k=0}^{q_j} 2^k = \ ?$$

Analyse der Verdopplungs-Strategie

Beweis (Fortsetzung)

Wir nehmen erstmal an, dass alle Bearbeitungszeiten Zweierpotenzen sind.

$\Longrightarrow$ In jeder Runde arbeiten wir gleich lang an jeder noch nicht fertigen Aufgabe, egal, ob sie fertig wird oder nicht.

Die Gesamtzeit, die wir an einer Aufgabe j der Länge $p_j = 2^{q_j}$ arbeiten, ist

$$1 + 2 + 4 + \cdots + 2^{q_j} = \sum_{k=0}^{q_j} 2^k = 2^{q_j+1} - 1$$

Analyse der Verdopplungs-Strategie

Beweis (Fortsetzung)

Wir nehmen erstmal an, dass alle Bearbeitungszeiten Zweierpotenzen sind.

$\Longrightarrow$ In jeder Runde arbeiten wir gleich lang an jeder noch nicht fertigen Aufgabe, egal, ob sie fertig wird oder nicht.

Die Gesamtzeit, die wir an einer Aufgabe j der Länge $p_j = 2^{q_j}$ arbeiten, ist

$$1 + 2 + 4 + \cdots + 2^{q_j} = \sum_{k=0}^{q_j} 2^k = 2^{q_j+1} - 1 \leq 2^{q_j+1} = 2 \cdot p_j$$

Analyse der Verdopplungs-Strategie

Beweis (Fortsetzung)

Fazit der letzten Folie: Mit der Verdopplungs-Strategie arbeiten wir an jeder Aufgabe höchstens doppelt so lang wie wir bräuchten, um sie am Stück abzuarbeiten.

Analyse der Verdopplungs-Strategie

Beweis (Fortsetzung)

Fazit der letzten Folie: Mit der Verdopplungs-Strategie arbeiten wir an jeder Aufgabe höchstens doppelt so lang wie wir bräuchten, um sie am Stück abzuarbeiten.

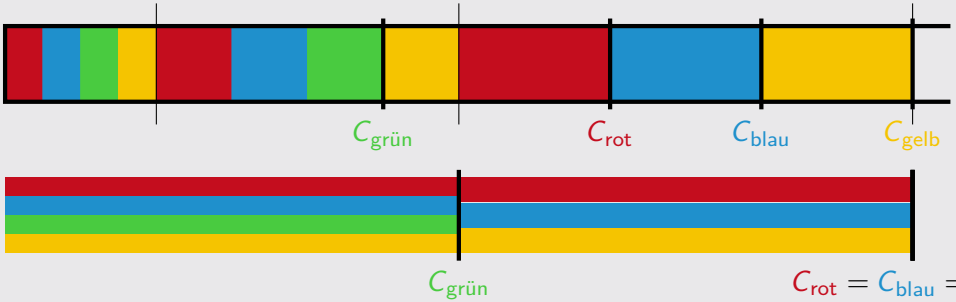
Beobachtung: Am Ende einer Runde sind alle Aufgaben der Runde gleich lang bearbeitet.

Analyse der Verdopplungs-Strategie

Beweis (Fortsetzung)

Fazit der letzten Folie: Mit der Verdopplungs-Strategie arbeiten wir an jeder Aufgabe höchstens doppelt so lang wie wir bräuchten, um sie am Stück abzuarbeiten.

Beobachtung: Am Ende einer Runde sind alle Aufgaben der Runde gleich lang bearbeitet.



Analyse der Verdopplungs-Strategie

Beweis (Fortsetzung)

Fazit der letzten Folie: Die Originalaufgaben werden mit der Verdopplungs-Strategie früher fertig als (fast) doppelt so lange Aufgaben mit Round-Robin.

$$\sum_{j=1}^{n} C_j^{\text{Verdopplung}}(\vec{p}) \leq \sum_{j=1}^{n} C_j^{\text{Round-Robin}}(2\vec{p})$$

,

Analyse der Verdopplungs-Strategie

Beweis (Fortsetzung)

Fazit der letzten Folie: Die Originalaufgaben werden mit der Verdopplungs-Strategie früher fertig als (fast) doppelt so lange Aufgaben mit Round-Robin.

$$\sum_{j=1}^{n} C_j^{\text{Verdopplung}}(\vec{p}) \leq \sum_{j=1}^{n} C_j^{\text{Round-Robin}}(2\vec{p})$$

$$\overset{\text{Satz 2}}{\leq} 2 \cdot \sum_{j=1}^{n} C_j^{\text{optimal}}(2\vec{p})$$

,

Analyse der Verdopplungs-Strategie

Beweis (Fortsetzung)

Fazit der letzten Folie: Die Originalaufgaben werden mit der Verdopplungs-Strategie früher fertig als (fast) doppelt so lange Aufgaben mit Round-Robin.

$$\begin{aligned}\sum_{j=1}^{n} C_j^{\text{Verdopplung}}(\vec{p}) &\leq \sum_{j=1}^{n} C_j^{\text{Round-Robin}}(2\vec{p}) \\ &\overset{\text{Satz 2}}{\leq} 2 \cdot \sum_{j=1}^{n} C_j^{\text{optimal}}(2\vec{p}) \\ &\leq 4 \cdot \sum_{j=1}^{n} C_j^{\text{optimal}}(\vec{p}),\end{aligned}$$

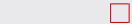
denn wenn alle Aufgaben doppelt so lang sind, ist die bestmögliche Summe der Fertigstellungszeiten auch doppelt so groß.

Beweis (Abschluss).

- Für Zweierpotenzen sind wir höchstens 4 mal so schlecht wie der beste Schedule.

Beweis (Abschluss).

- Für Zweierpotenzen sind wir höchstens 4 mal so schlecht wie der beste Schedule.
- Man kann zeigen, dass man dann für allgemeine Bearbeitungszeiten höchstens 8 mal so schlecht ist.



Beweis (Abschluss).

- Für Zweierpotenzen sind wir höchstens 4 mal so schlecht wie der beste Schedule.
- Man kann zeigen, dass man dann für allgemeine Bearbeitungszeiten höchstens 8 mal so schlecht ist.

□

Fazit

- Wenn wir nicht wissen, wie lange wir brauchen, verlieren wir höchstens einen Faktor von 2.

Beweis (Abschluss).

- Für Zweierpotenzen sind wir höchstens 4 mal so schlecht wie der beste Schedule.
- Man kann zeigen, dass man dann für allgemeine Bearbeitungszeiten höchstens 8 mal so schlecht ist.

□

Fazit

- Wenn wir nicht wissen, wie lange wir brauchen, verlieren wir höchstens einen Faktor von 2.
- Wenn wir uns zusätzlich nichts merken können, verlieren wir höchstens einen Faktor von 8.

Beweis (Abschluss).

- Für Zweierpotenzen sind wir höchstens 4 mal so schlecht wie der beste Schedule.
- Man kann zeigen, dass man dann für allgemeine Bearbeitungszeiten höchstens 8 mal so schlecht ist.

□

Fazit

- Wenn wir nicht wissen, wie lange wir brauchen, verlieren wir höchstens einen Faktor von 2.
- Wenn wir uns zusätzlich nichts merken können, verlieren wir höchstens einen Faktor von 8.

Bezug zu unserer Forschung:

Beweis (Abschluss).

- Für Zweierpotenzen sind wir höchstens 4 mal so schlecht wie der beste Schedule.
- Man kann zeigen, dass man dann für allgemeine Bearbeitungszeiten höchstens 8 mal so schlecht ist.

□

Fazit

- Wenn wir nicht wissen, wie lange wir brauchen, verlieren wir höchstens einen Faktor von 2.
- Wenn wir uns zusätzlich nichts merken können, verlieren wir höchstens einen Faktor von 8.

Bezug zu unserer Forschung:

- Mit komplizierten mathematischen Methoden (ca. 7 Seiten Beweis) konnten wir zeigen, dass man mit einer Ver$b$fachungs-Strategie höchstens einen Faktor von $1 + \frac{2b^{\frac{3}{2}}}{b-1}$ verlieren kann.

Beweis (Abschluss).

- Für Zweierpotenzen sind wir höchstens 4 mal so schlecht wie der beste Schedule.
- Man kann zeigen, dass man dann für allgemeine Bearbeitungszeiten höchstens 8 mal so schlecht ist.

□

Fazit

- Wenn wir nicht wissen, wie lange wir brauchen, verlieren wir höchstens einen Faktor von 2.
- Wenn wir uns zusätzlich nichts merken können, verlieren wir höchstens einen Faktor von 8.

Bezug zu unserer Forschung:

- Mit komplizierten mathematischen Methoden (ca. 7 Seiten Beweis) konnten wir zeigen, dass man mit einer Ver$b$fachungs-Strategie höchstens einen Faktor von $1 + \frac{2b^{\frac{3}{2}}}{b-1}$ verlieren kann.
- Am besten ist dies für $b = 3$, mit dem wir einen Faktor von $1 + 3\sqrt{3} \approx 6.1962$ verlieren.

Beweis (Abschluss).

- Für Zweierpotenzen sind wir höchstens 4 mal so schlecht wie der beste Schedule.
- Man kann zeigen, dass man dann für allgemeine Bearbeitungszeiten höchstens 8 mal so schlecht ist.

□

Fazit

- Wenn wir nicht wissen, wie lange wir brauchen, verlieren wir höchstens einen Faktor von 2.
- Wenn wir uns zusätzlich nichts merken können, verlieren wir höchstens einen Faktor von 8.

Bezug zu unserer Forschung:

- Mit komplizierten mathematischen Methoden (ca. 7 Seiten Beweis) konnten wir zeigen, dass man mit einer Ver$b$fachungs-Strategie höchstens einen Faktor von $1 + \frac{2b^{\frac{3}{2}}}{b-1}$ verlieren kann.
- Am besten ist dies für $b = 3$, mit dem wir einen Faktor von $1 + 3\sqrt{3} \approx 6.1962$ verlieren.
- Es gibt Beispiele, bei denen man beliebig nah an diese Schranke herankommt.

$$\boldsymbol{Z}_L = \begin{pmatrix} 0 & \frac{1}{\sqrt{b}-1} & & & & & \\ \frac{1}{\sqrt{b}-1} & -\frac{2}{b-1} & \frac{\sqrt{b}}{b-1} & & & & \\ & \frac{\sqrt{b}}{b-1} & -\frac{2}{b-1} & \frac{\sqrt{b}}{b-1} & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & \frac{\sqrt{b}}{b-1} & -\frac{2}{b-1} & \frac{\sqrt{b}}{b-1} & \\ & & & & \frac{\sqrt{b}}{b-1} & -\frac{2}{b-1} & \end{pmatrix} = \begin{pmatrix} 0 & \frac{1}{\sqrt{b}-1}\boldsymbol{e}_1^\top \\ \frac{1}{\sqrt{b}-1}\boldsymbol{e}_1 & \boldsymbol{T}_L\big(-\frac{2}{b-1}, \frac{\sqrt{b}}{b-1}\big) \end{pmatrix}.$$

We now want to show that $\lambda_{\max}(\boldsymbol{Z}_L) \leq \frac{2(\sqrt{b}-1)}{b-1}$. This is equivalent to the matrix $\boldsymbol{H} \coloneqq \frac{2(\sqrt{b}-1)}{b-1}\boldsymbol{I}_{L+1} - \boldsymbol{Z}_L$ being positive semidefinite, where $\boldsymbol{I}_{L+1}$ denotes the identity matrix with indices $\{0, \dots, L\}$. We compute

$$\boldsymbol{H} = \begin{pmatrix} \frac{2(\sqrt{b}-1)}{b-1} & -\frac{1}{\sqrt{b}-1}\boldsymbol{e}_1^\top \\ -\frac{1}{\sqrt{b}-1}\boldsymbol{e}_1 & \boldsymbol{T}_L\big(\frac{2\sqrt{b}}{b-1}, -\frac{\sqrt{b}}{b-1}\big) \end{pmatrix} = \frac{\sqrt{b}}{b-1}\begin{pmatrix} 2-\frac{2}{\sqrt{b}} & -\sqrt{1-\frac{1}{b}}\boldsymbol{e}_1^\top \\ -\sqrt{1-\frac{1}{b}}\boldsymbol{e}_1 & \boldsymbol{T}_L(2,-1) \end{pmatrix}.$$

This matrix has the form required in Lemma A.4 with $k = 1$, $\alpha = 2 - \frac{2}{\sqrt{b}}$, and $\boldsymbol{v} = -\sqrt{1-\frac{1}{b}} \in \mathbb{R}^1$. Since $\alpha - \|\boldsymbol{v}\|^2 = 2 - \frac{2}{\sqrt{b}} - \big(1 - \frac{1}{b}\big) = \frac{(\sqrt{b}-1)^2}{b} \geq 0$, the lemma implies that $\boldsymbol{H}$ is positive semidefinite, so that $\lambda_{\max}(\boldsymbol{Z}_L) \leq \frac{2(\sqrt{b}-1)}{b-1}$. Since this holds for every $L \in \mathbb{N}$, we obtain with inequality (5)

$$\frac{\mathfrak{D}_b}{\mathrm{OPT}} \leq 1 + \frac{2b}{b-1} + \frac{2b(\sqrt{b}-1)}{b-1} = 1 + \frac{2b^{\frac{3}{2}}}{b-1}.$$

The latter is minimized for $b = 3$ yielding the performance guarantee of $1 + 3\sqrt{3}$. □

Next, we show that our analysis of $\mathfrak{D}_b$ is asymptotically tight.

Theorem 4.5. *For every $b > 1$, there exists a sequence of instances $(\boldsymbol{p}_L)_{L\in\mathbb{N}}$ for $1 \,||\, \sum C_j$ such that*

$$\lim_{L\to\infty} \frac{\mathfrak{D}_b(\boldsymbol{p}_L)}{\mathrm{OPT}(\boldsymbol{p}_L)} = 1 + \frac{2b^{\frac{3}{2}}}{b-1}.$$

Proof. For $L \geq 1$ let $\boldsymbol{Y}_L$, and $\boldsymbol{Z}_L$ be the matrices defined above, and let $\boldsymbol{T}_L \coloneqq \boldsymbol{T}_L(-\frac{2}{b-1}, \frac{\sqrt{b}}{b-1})$ be the principal submatrix of $\boldsymbol{Z}_L$ and $\boldsymbol{z}_L = (z_\ell^{(L)})_{0\leq\ell\leq L}$ with $z_\ell^{(L)} \coloneqq \sqrt{\frac{2}{L+1}} \cdot \sin\big(\frac{\ell\pi}{L+1}\big)$. By [9, Theorem 2.4], $\tilde{\boldsymbol{z}}_L \coloneqq (z_\ell^{(L)})_{1\leq\ell\leq L}$

Was haben wir gelernt?

- Wir haben ein Beispiel für ein Online-Problem gesehen, bei dem wir zu Beginn nicht alles wissen.

Was haben wir gelernt?

- Wir haben ein Beispiel für ein Online-Problem gesehen, bei dem wir zu Beginn nicht alles wissen.
- Algorithmen für solche Probleme werden anhand des Competitive Ratio bewertet. Dies ist das Verhältnis des erreichten Werts zum bestmöglichen Wert im schlechtesten Fall.

Was haben wir gelernt?

- Wir haben ein Beispiel für ein Online-Problem gesehen, bei dem wir zu Beginn nicht alles wissen.
- Algorithmen für solche Probleme werden anhand des Competitive Ratio bewertet. Dies ist das Verhältnis des erreichten Werts zum bestmöglichen Wert im schlechtesten Fall.

Allgemeiner:

Was haben wir gelernt?

- Wir haben ein Beispiel für ein Online-Problem gesehen, bei dem wir zu Beginn nicht alles wissen.
- Algorithmen für solche Probleme werden anhand des Competitive Ratio bewertet. Dies ist das Verhältnis des erreichten Werts zum bestmöglichen Wert im schlechtesten Fall.

Allgemeiner:

- Unwissenheit und Vergesslichkeit sind nicht so schlimm (nur ein konstanter Faktor) :)

Was haben wir gelernt?

- Wir haben ein Beispiel für ein Online-Problem gesehen, bei dem wir zu Beginn nicht alles wissen.
- Algorithmen für solche Probleme werden anhand des Competitive Ratio bewertet. Dies ist das Verhältnis des erreichten Werts zum bestmöglichen Wert im schlechtesten Fall.

Allgemeiner:

- Unwissenheit und Vergesslichkeit sind nicht so schlimm (nur ein konstanter Faktor) :)
- Wenn etwas nicht fertig wird, sollte man es ruhig abbrechen und später mit mehr Zeit erneut angehen.

Was haben wir gelernt?

- Wir haben ein Beispiel für ein Online-Problem gesehen, bei dem wir zu Beginn nicht alles wissen.
- Algorithmen für solche Probleme werden anhand des Competitive Ratio bewertet. Dies ist das Verhältnis des erreichten Werts zum bestmöglichen Wert im schlechtesten Fall.

Allgemeiner:

- Unwissenheit und Vergesslichkeit sind nicht so schlimm (nur ein konstanter Faktor) :)
- Wenn etwas nicht fertig wird, sollte man es ruhig abbrechen und später mit mehr Zeit erneut angehen.

- Einfache Fragen (Wie gut sind wir mit einer Verdopplungsstrategie im schlechtesten Fall im Vergleich zum Optimum?)
  können überraschende Antworten haben (genau $1 + 4\sqrt{2}$ mal so schlecht).

Was haben wir gelernt?

- Wir haben ein Beispiel für ein Online-Problem gesehen, bei dem wir zu Beginn nicht alles wissen.
- Algorithmen für solche Probleme werden anhand des Competitive Ratio bewertet. Dies ist das Verhältnis des erreichten Werts zum bestmöglichen Wert im schlechtesten Fall.

Allgemeiner:

- Unwissenheit und Vergesslichkeit sind nicht so schlimm (nur ein konstanter Faktor) :)
- Wenn etwas nicht fertig wird, sollte man es ruhig abbrechen und später mit mehr Zeit erneut angehen.

- Einfache Fragen (Wie gut sind wir mit einer Verdopplungsstrategie im schlechtesten Fall im Vergleich zum Optimum?)
  können überraschende Antworten haben (genau $1 + 4\sqrt{2}$ mal so schlecht).
- Hoffentlich: Mathematik macht Spaß!