Experimentelle Analyse von ehrlichen Graph-Scheduling-Algorithmen

Bachelor-Thesis

vorgelegt am: 28. März 2022

Institut für Informatik, Goethe-Universität Frankfurt a.M.

Name:	Boris Senatov
Matrikelnummer:	6560227
Studiengang:	B.Sc. Informatik
Betreuer:	Prof. Dr. Ulrich Meyer

Bitte dieses Formular zusammen mit der Abschlussarbeit abgeben!

Erklärung zur Abschlussarbeit

gemäß § 35, Abs. 16 der Ordnung für den Bachelorstudiengang Informatik vom 17. Juni 2019:

Hiermit erkläre ich

Senator, Boris

(Nachname, Vorname)

Die vorliegende Arbeit habe ich selbstständig und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel verfasst.

Ich bestätige außerdem, dass die vorliegende Arbeit nicht, auch nicht auszugsweise, für eine andere Prüfung oder Studienleistung verwendet wurde.

Zudem versichere ich, dass alle eingereichten schriftlichen gebundenen Versionen meiner vorliegenden Bachelorarbeit mit der digital eingereichten elektronischen Version meiner Bachelorarbeit übereinstimmen.

Frankfurt am Main, den

28.03.2022

B. Senator

Unterschrift der/des Studierenden

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

In dieser Arbeit werden wir uns mit dem GRAPH-SCHEDULING-PROBLEM befassen. Hierbei handelt es sich um einen Spezialfall des UNRELATED-SCHEDULING-PROBLEMS. Ein Job $e = \{i, j\}$ kann nur auf die Maschinen i und j zugeteilt werden. Die jeweils erlaubten Jobs für die Maschinen können durch einen Graphen modelliert werden. Die Maschinen sind die Knoten und Jobs sind die Kanten. In dieser Arbeit werden wir uns mit dem vollständigen Graphen K_n beschäftigen. Für dieses Problem wollen wir spezielle ehrliche Mechanismen, die wir λ-Mechanismen nennen, entwickeln, wo die Besitzer der Maschinen ihre eigentlichen Jobkosten als Gebot angeben. Der Mechanismus entscheidet dann anhand der Gebote, welcher Spieler welchen Job zugeteilt bekommt.
Die λ-Mechanismen werden wir für K_5 und K_6 testen. Die Ergebnisse, die wir aus den Testreihen für K_5 und K_6 erhalten, werden wir benutzen, um das Verhalten der Mechanismen für beliebige $n \geq 5$ zu analysieren.
Hierbei sind wir insbesondere am Approximationsfaktor interessiert, sofern die Jobkosten beliebige positive reelle Zahlen sind. In dieser Arbeit werden wir versuchen zu zeigen, dass ein Mechanismus existiert, der für alle $n \geq 5$ einen Approximationsfaktor von weniger als $n-1$ garantiert. Für $n \leq 3$ ist hingegen bewiesen, dass ein solcher Mechanismus stets einen Approximationsfaktor von mindestens $n-1$ besitzt. Diese Frage hat einen Bezug zu der Vermutung von Nisan und Ronen [4], welche aussagt, dass kein ehrlicher Mechanismus für das Unrelated-Scheduling-Problem existiert, s.d. ein Approximationsfaktor von weniger als n angegeben werden kann.

2 Grundlagen

In diesem Abschnitt wollen wir zunächst einige Grundlagen für unsere späteren ehrlichen GRAPH-SCHEDULING-Mechanismen schaffen. In diesem Abschnitt werden wir uns an [2] orientieren.

2.1 Unrelated Scheduling

Beim UNRELATED-SCHEDULING-Problem gibt es eine Menge $\mathcal{N}$ von n Maschinen und eine Menge $\mathcal{M}$ von m Jobs. Weiter sind für alle $i \in \mathcal{N}$ und alle $s \in \mathcal{M}$ Joblaufzeiten $\tau_{is} \in \mathbb{R}_{\geq 0}$ als Eingabe gegeben. Maschine i besitzt für den Job s also die Joblaufzeit τ_{is}.
Eine Zuteilung der Jobs ist durch $X = (X_1, \ldots, X_n)$ gegeben. Für $i \in \{1, \ldots, n\}$ ist X_i die Menge von Jobs, die Maschine i zugeteilt werden. Ein Job muss genau einer Maschine zugeteilt werden.
Die Gesamtlaufzeit einer Maschine i ergibt sich aus $\tau_i(X_i) = \sum_{j \in X_i} \tau_{is}$.
Die Eingabe lässt sich durch folgende Matrix

$$\tau = \begin{pmatrix} \tau_{11} & \dots & \tau_{1m} \\ \dots & \dots & \dots \\ \tau_{n1} & \dots & \tau_{nm} \end{pmatrix} \text{ angeben.}$$

Das Ziel ist es, durch eine geeignete Zuteilung den sogenannten Makespan $\text{SPAN} = \max_{i \in N} \tau_i(X_i)$ zu minimieren.

In dieser Arbeit wollen wir einen Spezialfall des Unrelated-Scheduling-Problems betrachten. Hierbei kann ein Job von höchstens zwei Maschinen bearbeitet werden. Dies kann wie folgt geschehen.

2.2 Graph-Scheduling

Beim GRAPH-SCHEDULING-PROBLEM haben wir als Eingabe einen Graphen $G = (\mathcal{N}, E = \{e_1, \dots, e_m\})$. Die Knoten des Graphen sind die Maschinen und eine Kante $e = \{i, j\}$ entspricht einem Job, der entweder Maschine i oder Maschine j zugeteilt werden kann. Entsprechend haben wir für jede Kante $e = \{i, j\}$ die Joblaufzeiten t_{ij} und t_{ji} gegeben. Falls eine Kante $e = \{i, j\}$ nicht existiert, so setzen wir $t_{ij} = \infty$ und $t_{ji} = \infty$.
Der Einfachheit halber erlauben wir im Gegensatz zum gewöhnlichen Unrelated-Scheduling-Problem, dass $t_{ij} = \infty$ sein darf. ∞ steht für einen hinreichend hohen reellen Wert, welcher bei einem vernünftigen Approximationsfaktor eines Mechanismus in keiner Zuteilung vorkommt. Für eine Menge X_i von zugeteilten Jobs an Spieler i erhalten wir entsprechend $\tau_i(X_i) = \sum_{j \mid \{i,j\} \in X_i} t_{ij}$.
Für den vollständigen Graphen K_4 sieht unsere τ-Matrix wie folgt aus:

$$\tau = \begin{pmatrix} t_{12} & t_{13} & t_{14} & \infty & \infty & \infty \\ t_{21} & \infty & \infty & t_{23} & t_{24} & \infty \\ \infty & t_{31} & \infty & t_{32} & \infty & t_{34} \\ \infty & \infty & t_{41} & \infty & t_{42} & t_{43} \end{pmatrix}.$$

In den nächsten Abschnitten werden wir hier jedoch auf eine andere Matrix-Darstellung zugreifen. Der Eintrag (i, j) in der Matrix soll den Kosten t_{ij} entsprechen. Die Einträge auf der Diagonalen setzen wir auf ∞. Eine mögliche t-Matrix für K_n könnte in diesem Fall wie folgt aussehen:

$$t = \begin{pmatrix} \infty & t_{12} & \dots & t_{1n} \\ t_{21} & \infty & \dots & \dots \\ \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & t_{n-1\,n} \\ t_{n1} & \dots & t_{n\,n-1} & \infty \end{pmatrix}$$

2.3 Ehrliche Mechanismen

Wir sind an Graph-Scheduling-Algorithmen interessiert, die mit geeigneten Zahlungen an die Besitzer der Maschinen in sogenannten *ehrlichen Mechanismen* benutzt werden können. Die Spieler werden dafür bezahlt, dass sie die Jobs bearbeiten. Die Laufzeit eines Jobs $e = \{i, j\}$ auf der Maschine i entspricht den Kosten t_{ij}. Ein solcher Mechanismus wird allgemein für Unrelated-Scheduling wie folgt aufgebaut:

Aufbau des Mechanismus. Wir nehmen an, dass jeder Spieler $i \in \mathcal{N}$ lediglich daran interessiert ist, seinen eigenen Gewinn zu maximieren. Jeder Spieler besitzt eine private Kostenfunktion τ_i. Diese teilt einer Menge X_i von Jobs, die wahren Kosten für Spieler i zu. Weiter meldet Spieler i eine öffentliche Kostenfunktion τ_i'. τ_i' teilt einer Menge X_i von Jobs die gemeldeten Kosten für Spieler i zu. Unser Mechanismus besteht aus dem Tupel $M = (f, P = (P_1, \ldots, P_n))$.

Zuteilungsalgorithmus: Der Zuteilungsalgorithmus f nimmt als Eingabe die Gebote aller Spieler $b' = (b_1', \ldots, b_n')$ entgegen. Es gilt $b_i' = (\tau_i'(\{e_1\}), \ldots, \tau_i'(\{e_m\})$. Die ehrlichen gemeldeten Kosten würden $b_i = (\tau_i(\{e_1\}), \ldots, \tau_i(\{e_m\})$ entsprechen. Anhand der gemeldeten Gebote bestimmt der Algorithmus anschließend die Zuteilung $X = (X_1, \ldots, X_n)$.

Zahlungsschema: Anhand der gemeldeten Gebote bezahlt der Mechanismus anschließend mit $P_i(b')$ Spieler i.

Der Gewinn für Spieler i ist $u_i(b') = P_i(b') - \tau_i(f(b'))$.
Sei b_{-i}' das Gebot b' ohne Spieler i. Wir nennen einen Mechanismus *ehrlich*, falls für alle Spieler $i \in \mathcal{N}$ und allen möglichen Gebote $b_{-i}' \in \mathbb{R}_{\geq 0}^{(n-1)\times m}, b_i' \in \mathbb{R}_{\geq 0}^m$
$u_i(b_i, b_{-i}') \geq u_i(b_i', b_{-i}')$ erfüllt ist.
Die Spieler sind in einem ehrlichen Mechanismus also nicht daran interessiert, ein unehrliches Gebot anzugeben.

Im Folgenden möchten wir uns jedoch nicht weiter mit den Zahlungen beschäftigen. Hier hilft uns der nachfolgende Satz. Für diesen brauchen wir jedoch noch die folgende Definition.

Definition 1. Ein Zuweisungsalgorithmus f ist *schwach monoton*, falls für $i \in \mathcal{N}$ und alle möglichen Gebote $b = (b_i, b_{-i})$, $b' = (b_i', b_{-i})$ $\quad \tau_i(X_i) - \tau_i(X_i') \leq \tau_i'(X_i) - \tau_i'(X_i')$ erfüllt ist.

Satz 1. *Sei ein Mechanismus M ehrlich, so besitzt dieser einen schwach monotonen Zuweisungsalgorithmus f ([1]). Gilt hingegen, dass f ein schwach monotoner Zuweisungsalgorithmus und der Strategieraum B konvex ist, dann existieren Zahlungen P, s.d. M ein ehrlicher Mechanismus ist ([5]).*

Es ist bekannt, dass $\mathbb{R}^{n\times m}$ konvex ist. Da das gerade aber unser Strategieraum ist, genügt es im Folgenden stets nachzuweisen, dass unser Zuweisungsalgorithmus schwach monoton ist.

3 λ-Mechanismen für Graph-Scheduling des K_n

In diesem Abschnitt möchten wir zeigen, wie unsere speziellen ehrlichen Mechanismen für den vollständigen Graphen K_n definiert sind.
Nachdem jeder Spieler seine Kosten b_i gemeldet hat, wird für $i, j \in \mathcal{N}$ mit $i \neq j$ $\lambda_{ij} > 0$ bestimmt. Diese Werte sollen von der Eingabe t ohne die Gebote b_i und b_j abhängen. λ_{ij} wird also durch die Funktion $\lambda_{ij}(t)$ bestimmt. Zudem gilt $\lambda_{ji} = \frac{1}{\lambda_{ij}}$. Die Zuteilung der Jobs erfolgt dann wie folgt:
Falls $t_{ij} > t_{ji} \cdot \lambda_{ij}$ gilt, dann bekommt Spieler j Job $\{i, j\}$ zugeteilt.
Falls hingegen $t_{ji} < t_{ij} \cdot \lambda_{ji}$ erfüllt ist, dann bekommt Spieler i Job $\{i, j\}$ zugeteilt.
Ansonsten wird in einem Tiebreak entschieden, welcher der beiden Spieler den Job zugeteilt bekommt. Wir möchten nun zeigen, dass die oben definierten Zuteilungen, die solche λ-Funktionen verwenden, ehrlich sind.

Satz 2. *Seien für $i, j \in \mathcal{N}$ $t_{ij} \geq 0$ beliebig, dann ist der hier beschriebene Zuteilungsalgorithmus ehrlich.*

Beweis. Nach Satz 1 reicht es zu zeigen, dass der Algorithmus schwach monoton ist. Es seien für ein beliebiges $i \in \mathcal{N}$ zwei beliebige Gebote $b = (b_i, b_{-i})$ und $b' = (b'_i, b_{-i})$ gegeben. Bei diesen beiden Geboten ändern sich höchstens die Gebote von Spieler i. Da unsere λ-Funktion nicht von den Geboten der Spieler i und Spieler j abhängen, können wir annehmen, dass $\lambda_{ij}(b) = \lambda_{ij}(b')$ für $i \neq j$ erfüllt ist. Wird also t_{ij} vom Algorithmus ausgewählt, t'_{ij} hingegen nicht, dann gilt $t'_{ij} \cdot \lambda_{ji} \geq t_{ji} \geq t_{ij} \cdot \lambda_{ji}$ s.d. $t'_{ij} \geq t_{ij}$ erfüllt ist. Weiter seien die aus den Geboten resultierenden Zuteilungen X_i und X'_i gegeben. Wir möchten nun folgende Mengen betrachten:
$A_1 = \{e | e \in X_i, e \in X'_i\}$
$A_2 = \{e | e \notin X_i, e \notin X'_i\}$
$A_3 = \{e | e \in X_i, e \notin X'_i\}$
$A_4 = \{e | e \notin X_i, e \in X'_i\}$

<u>1 Fall:</u> $e \in A_1$

Dann gilt $\tau_i(\{e\}) - \tau_i(\{e\}) = 0 = \tau'_i(\{e\}) - \tau'_i(\{e\})$

<u>2 Fall:</u> $e \in A_2$

Dann gilt $\tau_i(\emptyset) - \tau_i(\{\emptyset\}) = 0 = \tau'_i(\{\emptyset\}) - \tau'_i(\{\emptyset\})$

<u>3 Fall:</u> $e \in A_3$

Dann aber muss für $e = \{i, j\}$ $t_{ij} \leq t'_{ij}$ erfüllt sein. Daraus folgt
$\tau_i(\{e\}) - \tau_i(\{\emptyset\}) = \tau_i(\{e\}) = t_{ij} \leq t'_{ij} = \tau'_i(\{e\}) = \tau'_i(\{e\}) - \tau'_i(\{\emptyset\})$.

4 Fall: $e \in A_4$

Dann gilt für $e = \{i, j\}$ $t_{ij} \geq t'_{ij}$. Daraus folgt
$\tau_i(\{\emptyset\}) - \tau_i(\{e\}) = -\tau_i(\{e\}) = -t_{ij} \leq -t'_{ij} = -\tau'_i(\{e\}) = \tau'_i(\{\emptyset\}) - \tau'_i(\{e\})$.

Da τ_i, bzw. τ'_i additiv sind, folgt $\tau_i(X_i) - \tau_i(X') \leq \tau'_i(X_i) - \tau'_i(X')$, woraus sich direkt ergibt, das unser Algorithmus schwach monoton ist. □

In den nächsten zwei Kapiteln werden wir uns mit λ-Mechanismen beschäftigen.

4 λ-Mechanismen mit konstanter λ-Funktion

Der Algorithmus dieser λ-Funktion ist recht trivial. Wir betrachten den Fall, wenn für $i, j \in \mathcal{N}$ mit $i \neq j$ λ_{ij} ein konstanter Wert gegeben ist, welcher also unabhängig von der Eingabe t ist. Für alle $i \neq j$ gilt zudem $\lambda_{ji} = \frac{1}{\lambda_{ij}}$.
Für diese λ-Funktion lässt sich einfach beweisen, dass wenn beliebige $t_{ij} \in \mathbb{R}_{\geq 0}$ Werte in der Eingabematrix zugelassen werden, dann der Approximationsfaktor mindestens $n-1$ ist. Wir wollen zunächst untersuchen, woran das genau liegt.

4.1 Analyse

Zunächst möchten wir zeigen, dass der Approximationsfaktor mindestens $n-1$ ist. Anschließend werden wir zeigen, dass wenn wir für $i, j \in \mathcal{N}$ mit $i < j$ λ_{ij} passend wählen, wir sogar einen Approximationsfaktor von exakt $n-1$ erreichen können. Um Ersteres zu zeigen, benötigen wir jedoch noch zwei Lemmas.

Lemma 3. *Sei $\lambda > 0$ beliebig, dann gilt $\lambda + \frac{1}{\lambda} \geq 2$.*

Beweis.

$$\begin{aligned} (\lambda - 1)^2 &= \lambda^2 - 2\lambda + 1 \geq 0 \\ \lambda^2 + 1 &\geq 2\lambda \\ \lambda + \frac{1}{\lambda} &\geq 2 \end{aligned}$$

□

Lemma 4. *Sei für $i, j \in \mathcal{N}$ mit $i < j$ $\lambda_{ij} > 0$ beliebig, dann gilt*
$\sum_{i=1}^{n} \sum_{j=1, i \neq j}^{n} \lambda_{ij} \geq n \cdot (n-1)$.

Beweis. Es gilt:

$$\begin{aligned}\sum_{i=1}^{n}\sum_{j=1,i\neq j}^{n}\lambda_{ij} &= \sum_{i=1}^{n}\sum_{j|j<i}(\lambda_{ji}+\lambda_{ij})\\ &\overset{\text{Lemma 3.}}{\geq} \sum_{i=1}^{n}\sum_{j|j<i}2 = \frac{n\cdot(n-1)}{2}\cdot 2 = n\cdot(n-1)\end{aligned}$$

□

Satz 5. *Sei für $i,j \in \mathcal{N}$, mit $i \neq j$ $\lambda_{ij} > 0$, Konstanten mit $\lambda_{ij} = \frac{1}{\lambda_{ji}}$, dann ist der Approximationsfaktor des λ-Mechanismus stets $\geq n-1$.*

Beweis. Nach Lemma 4, ist $\sum_{i=1}^{n}\sum_{j|i\neq j}\lambda_{ij} \geq n\cdot(n-1)$ stets erfüllt.

Damit muss es aber einen Spieler $k \in \mathcal{N}$ geben für den $\sum_{j=1,j\neq k}^{n}\lambda_{kj} \geq n-1$ erfüllt ist. Wir betrachten nun folgende Eingabe:

$$t_{ij} = \begin{cases}\lambda_{ij}-\epsilon, \text{falls} & i=k\\ 1, \quad \text{falls} & j=k\\ 0, \quad \text{sonst}\end{cases}$$

Ein Beispiel für den K_5 mit $k=1$ sieht wie folgt aus:

Beispiel 1.

$$t = \begin{pmatrix}\infty & \lambda_{21}-\epsilon & \lambda_{31}-\epsilon & \lambda_{41}-\epsilon & \lambda_{51}-\epsilon\\ 1 & \infty & 0 & 0 & 0\\ 1 & 0 & \infty & 0 & 0\\ 1 & 0 & 0 & \infty & 0\\ 1 & 0 & 0 & 0 & \infty\end{pmatrix}$$

Spieler k bekommt den Job $\{k,j\}$ zugeteilt, sofern $t_{jk} > \lambda_{jk}\cdot t_{kj}$ erfüllt ist. In unserem Fall ist $t_{jk} = 1$, $t_{kj} = \lambda_{kj}-\epsilon$. Nun gilt $t_{jk} = 1 > \frac{1}{\lambda_{kj}}\cdot(\lambda_{kj}-\epsilon) = \lambda_{jk}\cdot t_{kj}$. Folglich bekommt Spieler k alle Jobs zugeteilt. Trivialerweise besteht die optimale Zuteilung aber darin, dass Spieler k keinen Job bekommt. In diesem Fall gilt $\text{OPT} = 1$. Da $\sum_{j|k\neq j}\lambda_{kj} \geq n-1$ gilt, erhalten wir $\frac{\text{SPAN}}{\text{OPT}} \geq \frac{(n-1)\cdot(1-\epsilon)}{1} = (n-1)\cdot(1-\epsilon)$. Da ϵ beliebig klein sein kann, folgt, dass der Approximationsfaktor mindestens $n-1$ ist. □

Nun möchten wir zeigen, dass der Approximationsfaktor exakt $n-1$ sein kann.

Satz 6. *Sei für $i,j \in \mathcal{N}$ mit $i \neq j$ $\lambda_{ij} = 1$. Dann erhalten wir einen Approximationsfaktor von exakt $n-1$.*

Beweis. Nach Satz 5. wissen wir, dass wir mit dieser Wahl von λ-Werten einen Approximationsfaktor von mindestens $n-1$ erhalten. Es muss also noch geklärt werden, dass es höchstens $n-1$ sein kann.
Ein Spieler $i \in \mathcal{N}$ kann höchstens $n-1$ Jobs zugeteilt bekommen. Falls für $i, j \in \mathcal{N}$ mit $i \neq j$ $\lambda_{ij} = 1$ und $\lambda_{ji} = 1$ gilt, bekommt dadurch stets der Spieler den Job, welcher die kleinere Joblaufzeit hat. Bekommt Spieler i Job $\{i, j\}$ zugeteilt, somit gilt $t_{ij} \leq t_{ji}$. Da nun entweder t_{ij} oder t_{ji} in einer optimalen Zuteilung sind, erhalten wir $\text{OPT} \geq \min\{t_{ij}, t_{ji}\} = t_{ij}$.
Ein Spieler kann also nur Jobs zugeteilt bekommen, welche $\leq \text{OPT}$ sind.
Insgesamt kann dieser also höchstens $n-1$ solcher Jobs erhalten, s.d. $\text{SPAN} \leq (n-1) \cdot \text{OPT}$ erfüllt ist. Daraus folgt $\frac{\text{SPAN}}{\text{OPT}} \leq n-1$. □

Beispiel 2. Nun wollen wir ein konkretes Beispiel für Instanzen, bei welchen ein Approximationsfaktor von $n-1$ erreicht wird, betrachten. Sei für $i, j \in \mathcal{N}$ mit $i \neq j$ $\lambda_{ij} = 1$. Zudem sei $\epsilon > 0$ hinreichend klein gegeben. Dann können wir folgende Eingabe betrachten:

$$t_{ij} = \begin{cases} 1-\epsilon, \text{falls} & i = 1, j > 1 \\ 1, \quad \text{falls} & j = 1, i > 1 \\ 0 \text{ sonst} & \end{cases} \qquad t = \begin{pmatrix} \infty & 1-\epsilon & \dots & \dots & \dots & 1-\epsilon \\ 1 & \infty & 0 & \dots & \dots & 0 \\ \dots & 0 & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & 0 \\ 1 & 0 & \dots & \dots & 0 & \infty \end{pmatrix}$$

Die optimale Zuteilung besteht darin, dass Spieler 1 höchstens einen Job zugeteilt bekommt. Folglich erhalten wir $\text{OPT} = 1$. Der Algorithmus wird jedoch Spieler 1 alle Jobs zuteilen. Insgesamt gilt also $\text{SPAN} = (n-1) \cdot (1-\epsilon)$.
Dadurch erhalten wir $\lim\limits_{\epsilon \to 0} \frac{\text{SPAN}}{\text{OPT}} = n-1$, was der gewünschten unteren Schranke für den Approximationsfaktor entspricht.

4.2 Tests

Wir haben den λ-Mechanismus mit verschiedenen λ-Matrizen implementiert und für verschiedene Eingaben getestet. Die Tests wollen wir für K_5 und K_6 angeben. Für K_5 und K_6 haben wir für jeden Testfall je zehn bzw. eine Millionen zufällige Instanzen t generiert und den entsprechenden Makespan und Optimum bestimmt. Anschließend haben wir für jede Instanz den Approximationsfaktor berechnet.
Nun werden wir mit Hilfe eines Histogramms (Vorlage für Code [3]) grafisch die Verteilung der errechneten Approximationsfaktoren darstellen. Hier werden wir zusätzlich den Durchschnitt wie auch den größtmöglichen Approximationsfaktor angeben, welcher errechnet wurde. Wir werden sehen, dass größere Approximationsfaktoren tendenziell sehr viel seltener als kleinere vorkommen. Damit diese Ergebnisse auf dem Histogramm sichtbar bleiben, werden wir für die y-Achse des Histogramms, welche die Anzahl der errechneten Approximationsfaktoren angibt, eine logarithmische Skalierung verwenden. Anschließend werden

wir Worst-Case-Fälle für den K_5 angeben und unsere Beobachtungen dazu aufschreiben. Zum Worst-Case-Fall werden wir jeweils unsere Eingabematrix t und die Matrizen λ und x angeben, wobei x die Zuteilungsmatrix mit $x_{ij} = 1$, falls $\{i, j\} \in X_i$, und ansonsten 0 ist. Für die λ-Matrix wie auch für die x-Matrix setzen wir die Werte auf der Diagonalen auf ∞. Die Werte auf der Diagonalen werden wir jedoch wie auch schon bei der t-Matrix nicht berücksichtigen.
Beim Testen werden wir nur endlich viele Jobkosten zulassen. Für $i, j \in \mathcal{N}$ mit $i \neq j$ ist t_{ij} eine Zufallsvariable, welche gemäß einer angegebenen Verteilung unabhängig und zufällig verteilt ist. Gilt $t_{ij} = \lambda_{ij} \cdot t_{ji}$, so kommt es zu einem Tiebreak. Der Job $\{i, j\}$ wird mit einer Wahrscheinlichkeit von 0.5 entweder an Spieler i oder Spieler j vergeben.

Testreihe 1. Folgend setzen wir für $i, j \in \mathcal{N}$ mit $i \neq j$ $\lambda_{ij} = 1$. Weiterhin sei t_{ij} unabhängig gemäß folgender Verteilung: $\mathbb{P}(t_{ij} = 1) = 0.7$ und $\mathbb{P}(t_{ij} = 0) = 0.3$. Wir wissen aus der Analyse, dass der Approximationsfaktor über alle Instanzen, falls $t_{ij} \geq 0$ beliebig, exakt $n - 1$ sein muss. Wir wollen nun überprüfen, ob wir auch durch Experimente entsprechende Worst-Case-Fälle ermitteln können. Erst dann können wir hoffen, dass analoge Testverfahren für allgemeine λ_{ij}-Funktionen aussagekräftig sind. Die Histogramme zur Verteilung der Approximationsfaktoren von K_5 und K_6 sehen wie folgt aus:

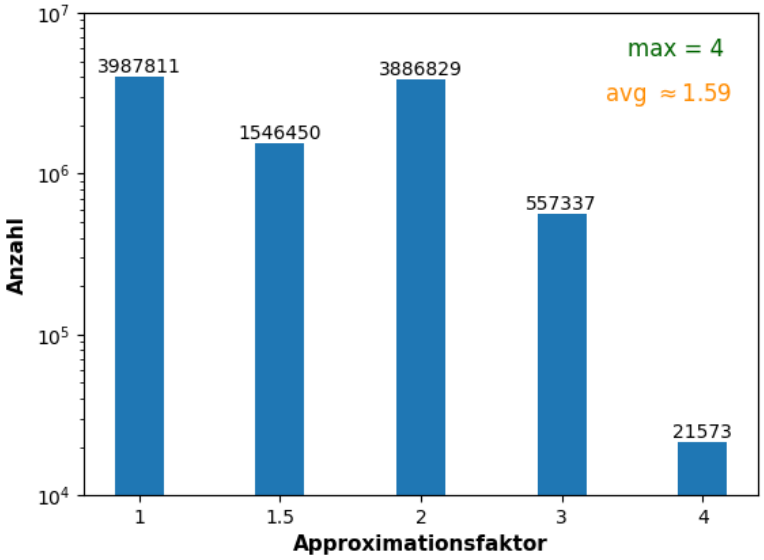

Abbildung 1: Testreihe 1 für K_5

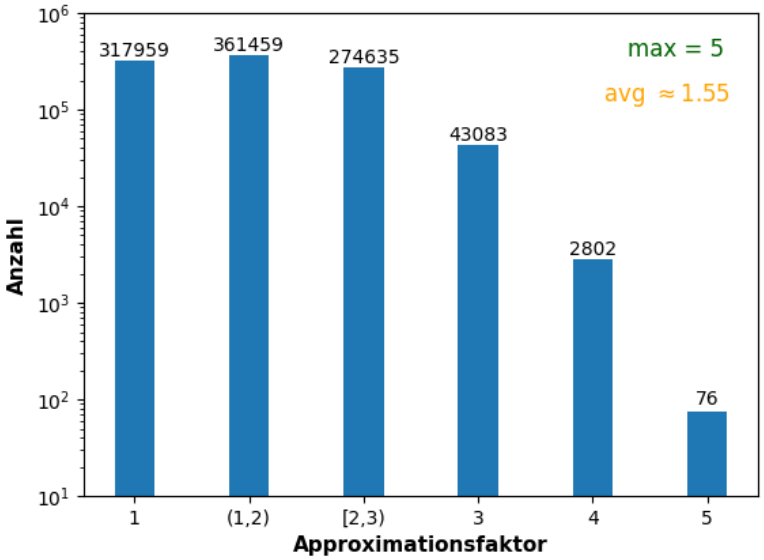

Abbildung 2: Testreihe 1 für K_6

Die Testreihe 1 zeigt uns (siehe Abbildung 1 und Abbildung 2), dass wenn wir für $i, j \in \mathcal{N}$, $i \neq j$ $\lambda_{ij} = 1$ setzen, unser Algorithmus immer wieder Zuteilungen liefern wird, bei welchen das Optimum $\text{OPT} = 1$ und der Makespan $\text{SPAN} = n - 1$ ist. Während die Anzahl der durchgeführten Testfälle für den K_6 um das Zehnfache geringer ist, sinkt die Anzahl der Worst-Case-Fälle um mehr als das 200-fache. Die relative Häufigkeit für solche Worst-Case-Fälle sinkt also mit größerem n. Das liegt daran, dass für ein solches Worst-Case-Szenario drei Bedingungen erforderlich sind:

- Für mindestens einen Spieler $i \in \mathcal{N}$ muss für alle $j \neq i$ $t_{ij} = 1$ und $t_{ji} = 1$ erfüllt sein.
- Spieler i muss $n - 1$ Jobs zugeteilt bekommen, was zudem in diesem Test nur unter Tiebreak Bedingungen möglich ist.
- Es muss gelten, dass $\text{OPT} = 1$ entspricht, ansonsten ist es nicht möglich, dass wir einen Approximationsfaktor von $n - 1$ erreichen.

Ein mögliches Worst-Case-Beispiel für den K_5 sieht wie folgt aus:

Beispiel 3.

$$t = \begin{pmatrix} \infty & 1 & 1 & 1 & 1 \\ 1 & \infty & 0 & 0 & 0 \\ 1 & 0 & \infty & 0 & 0 \\ 1 & 0 & 0 & \infty & 0 \\ 1 & 0 & 0 & 0 & \infty \end{pmatrix} \lambda = \begin{pmatrix} \infty & 1 & 1 & 1 & 1 \\ 1 & \infty & 1 & 1 & 1 \\ 1 & 1 & \infty & 1 & 1 \\ 1 & 1 & 1 & \infty & 1 \\ 1 & 1 & 1 & 1 & \infty \end{pmatrix} x = \begin{pmatrix} \infty & 1 & 1 & 1 & 1 \\ 0 & \infty & 1 & 1 & 1 \\ 0 & 0 & \infty & 1 & 1 \\ 0 & 0 & 0 & \infty & 1 \\ 0 & 0 & 0 & 0 & \infty \end{pmatrix}$$

Wie man hier sehen kann, gewinnt Spieler 1 in allen vier Tiebreaks. Somit bekommt dieser alle vier Jobs zugeteilt. Optimal ist jedoch, wenn Spieler 1 höchstens einen Job zugeteilt kriegt.

Testreihe 2. In der folgenden Testreihe wollen wir uns das Verhalten des Mechanismus anschauen, wenn auch verschiedene λ-Werte zugelassen sind. Hierbei setzen wir für $i, j \in \mathcal{N}, i \neq j$ die λ_{ij} mit einem Beispiel für K_5 wie folgt:

$$\lambda_{ij} = \begin{cases} 2, \text{falls} \quad i < j, i+j \text{ gerade} \\ 2, \text{falls} \quad i > j, i+j \text{ ungerade} \\ \frac{1}{2}, \text{ falls } i < j, i+j \text{ ungerade} \\ \frac{1}{2}, \text{ falls } i > j, i+j \text{ gerade} \end{cases} \qquad \lambda = \begin{pmatrix} \infty & 0.5 & 2 & 0.5 & 2 \\ 2 & \infty & 0.5 & 2 & 0.5 \\ 0.5 & 2 & \infty & 0.5 & 2 \\ 2 & 0.5 & 2 & \infty & 0.5 \\ 0.5 & 2 & 0.5 & 2 & \infty \end{pmatrix}$$

Sei für $i, j \in \mathcal{N}$ mit $i \neq j$ t_{ij} unabhängig und randomisiert gemäß folgender Verteilung: $\mathbb{P}(t_{ij} = 0) = 0.3$ und $\mathbb{P}(t_{ij} = 1) = 0.7$. Die Histogramme zur Verteilung der Approximationsfaktoren von K_5 und K_6 sehen wie folgt aus:

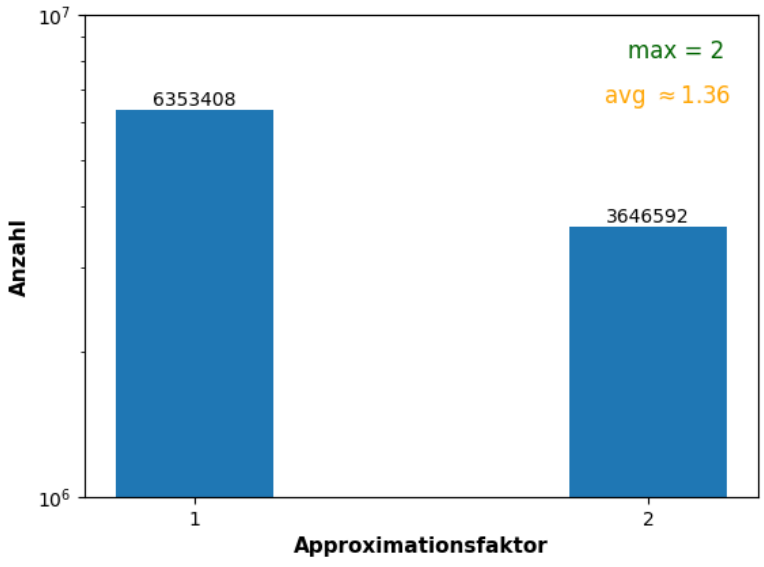

Abbildung 3: Testreihe 2 für K_5

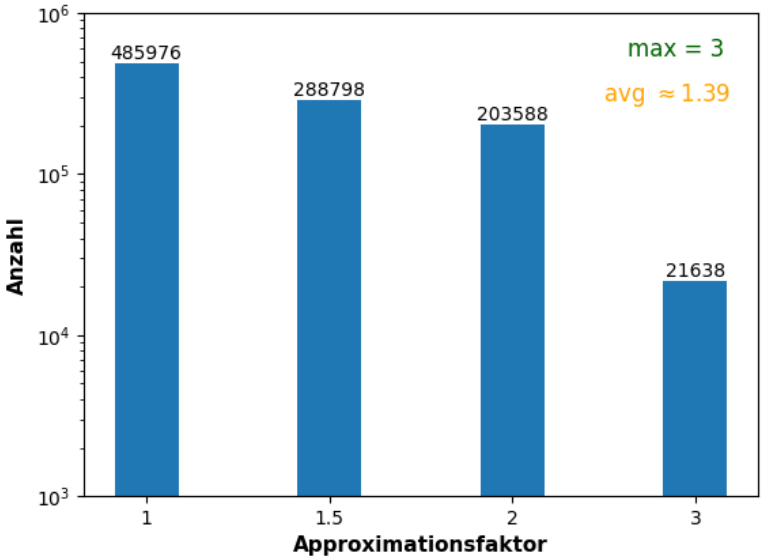

Abbildung 4: Testreihe 2 für K_6

Wie man hier sehen kann (siehe Abbildung 3 und Abbildung 4), ist es möglich für $i, j \in \mathcal{N}$ λ_{ij} Werte zu verwenden, s.d. für $t_{ij} \in \{0, 1\}$ der Approximationsfaktor echt kleiner als $n-1$ ist. Dies ist jedoch natürlich nur möglich, da die t_{ij}-Werte in der Eingabe eingeschränkt sind. Hier haben wir den kleineren Approximationsfaktor dadurch erreicht, da für jedes $i \in \mathcal{N}$ nach Konstruktion für $j \in \mathcal{N}$ mindestens $\lfloor \frac{n}{2} \rfloor$ λ-Werte mit $\lambda_{ij} = 0.5$ existieren. Dies sorgt dafür, dass wenn $t_{ij} = 1$ und $t_{ji} = 1$, dann t_{ji} bevorzugt wird.
Ein mögliches Worst-Case-Beispiel sieht für K_5 wie folgt aus:

Beispiel 5.

$$t = \begin{pmatrix} \infty & 1 & 1 & 1 & 1 \\ 1 & \infty & 0 & 0 & 0 \\ 1 & 0 & \infty & 0 & 0 \\ 1 & 0 & 0 & \infty & 0 \\ 1 & 0 & 0 & 0 & \infty \end{pmatrix} \lambda = \begin{pmatrix} \infty & \frac{1}{2} & 2 & \frac{1}{2} & 2 \\ 2 & \infty & \frac{1}{2} & 2 & \frac{1}{2} \\ \frac{1}{2} & 2 & \infty & \frac{1}{2} & 2 \\ 2 & \frac{1}{2} & 2 & \infty & \frac{1}{2} \\ \frac{1}{2} & 2 & \frac{1}{2} & 2 & \infty \end{pmatrix} x = \begin{pmatrix} \infty & 0 & 1 & 0 & 1 \\ 1 & \infty & 1 & 1 & 1 \\ 0 & 0 & \infty & 1 & 1 \\ 1 & 0 & 0 & \infty & 1 \\ 0 & 0 & 0 & 0 & \infty \end{pmatrix}$$

In diesem Beispiel ist zu sehen, dass Spieler 2 insgesamt vier Jobs gewinnt. Das ist aber nur möglich, weil bei drei seiner Jobs die Kosten 0 betragen. Den Makespan bestimmt in diesem Beispiel jedoch Spieler 1, der insgesamt zwei Jobs mit Kosten 1 zugeteilt bekommt.

Testreihe 3. In dieser Testreihe übernehmen wir die λ-Werte aus der Testreihe 2.
Seien die t_{ij} Zufallsvariablen, welche nach folgender Verteilung unabhängig verteilt sind: Es gilt $\mathbb{P}(t_{ij} = 0) = 0.4, \mathbb{P}(t_{ij} = 1) = 0.3$ und $\mathbb{P}(t_{ij} = 2) = 0.3$. Die Histogramme zur Verteilung der Approximationsfaktoren von K_5 und K_6 sehen wie folgt aus:

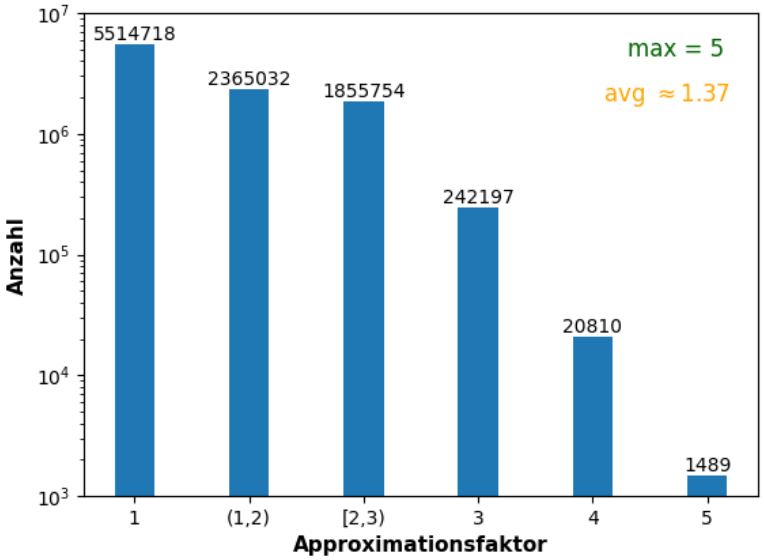

Abbildung 5: Testreihe 3 für K_5

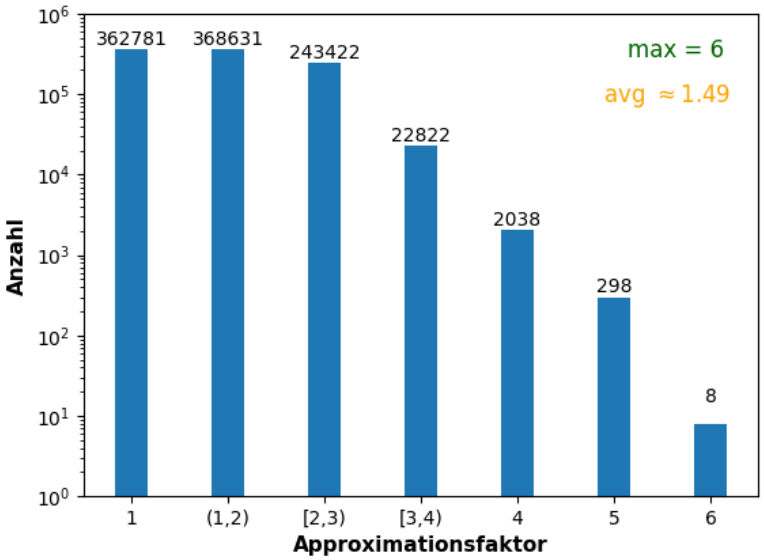

Abbildung 6: Testreihe 3 für K_6

Hier können wir sehen (siehe Abbildung 5 und Abbildung 6), dass es mit den entsprechenden λ-Werten, wenn für $i, j \in \mathcal{N}$ mit $i \neq j$ $t_{ij} \in \{0, 1, 2\}$ erfüllt ist, möglich ist, einen Approximationsfaktor von deutlich mehr als $n-1$ zu erreichen. Ein errechnetes Worst-Case-Beispiel sieht für den Fall K_5 wie folgt aus:

Beispiel 6.

$$t = \begin{pmatrix} \infty & 1 & 2 & 1 & 2 \\ 2 & \infty & 0 & 0 & 0 \\ 1 & 0 & \infty & 0 & 0 \\ 1 & 0 & 0 & \infty & 0 \\ 1 & 0 & 0 & 0 & \infty \end{pmatrix} \lambda = \begin{pmatrix} \infty & \frac{1}{2} & 2 & \frac{1}{2} & 2 \\ 2 & \infty & \frac{1}{2} & 2 & \frac{1}{2} \\ \frac{1}{2} & 2 & \infty & \frac{1}{2} & 2 \\ 2 & \frac{1}{2} & 2 & \infty & \frac{1}{2} \\ \frac{1}{2} & 2 & \frac{1}{2} & 2 & \infty \end{pmatrix} x = \begin{pmatrix} \infty & 1 & 1 & 0 & 1 \\ 0 & \infty & 1 & 1 & 1 \\ 0 & 0 & \infty & 1 & 1 \\ 1 & 0 & 0 & \infty & 1 \\ 0 & 0 & 0 & 0 & \infty \end{pmatrix}$$

In diesem Beispiel bekommt Spieler 1 zwei Jobs mit den Kosten 2 und einen Job mit den Kosten 1 zugeteilt. Eine optimale Zuteilung besteht jedoch darin, das Spieler 1 nur einen Job mit Kosten 1 zugeteilt bekommt. Daraus resultiert letztlich ein Approximationsfaktor von 5 für diese Instanz. Eine Modifizierung von t ergibt:

$$t' = \begin{pmatrix} \infty & 1 & 2 & 1 & 2 \\ 2 & \infty & 0 & 0 & 0 \\ 1 & 0 & \infty & 0 & 0 \\ 2 & 0 & 0 & \infty & 0 \\ 1 & 0 & 0 & 0 & \infty \end{pmatrix}$$

In diesem Fall würde ein Makespan von 6 erreicht werden. Eine optimale Zuteilung besteht jedoch darin, wenn keiner der Jobs, welche die Jobkosten 2 haben, verwendet werden. Folglich wären dann $\text{OPT} = 2$ und der Approximationsfaktor gleich 3. Dies zeigt uns, dass der maximale Makespan nicht immer auch ein möglicher Approximationsfaktor sein muss.

5 λ Mechanismen mit nicht-konstanter $\lambda(t)$ Funktion

Des Weiteren möchten wir uns mit nicht-konstanten λ-Funktionen beschäftigen. Bei konstanten λ-Funktionen sind die λ-Werte fest und es ist nicht möglich, die λ-Werte an die Eingabe t anzupassen. Im Folgenden versuchen wir also λ-Funktionen zu definieren, die von der Eingabe t abhängen, und dadurch versuchen einen Approximationsfaktor von $< n - 1$ zu erreichen. Unsere nicht konstanten λ-Funktionen dürfen bei der Berechnung von λ_{ij} nicht von den Jobkosten der Spieler i und j abhängen. Alle anderen Jobkosten dürfen verwendet werden.
Bevor wir uns mit konkreten λ-Funktion beschäftigen, möchten wir uns zunächst anschauen, weshalb einerseits die errechneten λ_{ij}-Werte beschränkt sein sollen. Anschließend wollen wir eine Intuition dafür geben, auf welcher Grundlage die λ_{ij}-Funktionen die λ_{ij}-Werte bestimmen sollen.

Größe der λ-Werte. Im Folgenden möchten wir uns näher betrachten, welchen Einfluss die λ-Werte auf den Approximationsfaktor nehmen. Hierfür definieren wir ein $\lambda_{\max} \geq 1$. Dies soll der größte λ-Wert sein, denn jede λ_{ij}-Funktion erreichen kann. Analog setzen wir $\lambda_{\min} = \frac{1}{\lambda_{\max}}$. Dann gilt folgendes Lemma:

Lemma 7. *Sei t_{ij} zugeteilt, also $t_{ij} \leq t_{ji} \cdot \lambda_{ij}$, dann gilt $t_{ij} \leq \text{OPT} \cdot \lambda_{\max}$.*

Beweis. Falls $t_{ji} > \text{OPT}$, dann muss t_{ij} in einer optimalen Zuteilung sein. Somit gilt $t_{ij} \leq \text{OPT} \leq \text{OPT} \cdot \lambda_{\max}$. Ansonsten gilt $t_{ji} \leq \text{OPT}$. Nun aber gilt $t_{ij} \leq t_{ji} \cdot \lambda_{\max} \leq \lambda_{\max} \cdot \text{OPT}$. □

Daraus folgt jedoch unmittelbar auch folgendes Lemma:

Lemma 8. *Sei t_{ij} zugeteilt, $\lambda_{ij} \leq \lambda_{ji}$, dann gilt $t_{ij} \leq \text{OPT}$.*

Nun wollen wir noch überprüfen, wie groß der Approximationsfaktor mit einem gegebenen $\lambda_{\max}$ höchstens werden kann. Hier hilft uns folgendes Lemma:

Lemma 9. *Sei $\lambda_{\max} \geq 1$ beliebig, dann erhalten wir mit Hilfe einer beliebigen λ-Funktion einen Approximationsfaktor von höchstens $\lambda_{\max} \cdot (n-1)$.*

Beweis. Im Folgenden nehmen wir an, dass ein Spieler $k \in \mathcal{N}$ den Makespan bestimmt. Zudem nutzen wir Lemma 7 aus. Für einen Job $\{k, j\}$ muss also $t_{kj} \cdot x_{kj} \leq \lambda_{\max} \cdot \text{OPT}$ erfüllt sein. Insgesamt gilt:

$\text{SPAN} = \sum_{j=1, j\neq k}^{n} t_{kj} \cdot x_{kj} \leq \sum_{j=1, j\neq k}^{n} \lambda_{\max} \cdot \text{OPT} = (n-1) \cdot \lambda_{\max} \cdot \text{OPT}$, woraus unser Approximationsfaktor von $(n-1) \cdot \lambda_{\max}$ resultiert. □

Bei der Bestimmung von λ_{ij} haben wir keine Informationen über die Jobkosten t_{ij} oder t_{ji}. Wählen wir also beispielsweise $\lambda_{ij} > n-1$, so erhalten wir mit $t_{ji} = \text{OPT}$ und $t_{ij} = (n-1) \cdot \text{OPT}$, dass Spieler i einen Job $\{i, j\}$, mit Kosten $(n-1) \cdot \text{OPT}$, zugeteilt bekommt. Damit würden wir jedoch direkt einen Approximationsfaktor $\geq n-1$ erhalten. Das wollen wir im Folgenden verhindern. Aus diesem Grund sollen die Werte, welche die hier betrachteten λ-Funktionen berechnen, durch ein $\lambda_{\max}$, welches hinreichend klein ist, beschränkt werden.

Intuition. Nun wollen wir zwei Eingaben für K_5 untersuchen, um zu verdeutlichen, wie unsere λ-Funktion λ_{ij} wählen sollte, damit möglichst ein Approximationsfaktor von unter $n-1$ erreicht werden kann.

Beispiel 7. Wir befassen uns mit folgendem Beispiel. Sei

$$t = \begin{pmatrix} \infty & 1 & 1 & 1 & 1 \\ 1 & \infty & 0 & 0 & 0 \\ 1 & 0 & \infty & 0 & 0 \\ 1 & 0 & 0 & \infty & 0 \\ 1 & 0 & 0 & 0 & \infty \end{pmatrix}.$$

In diesem Fall ist Spieler 1 der Einzige, der bis zu vier Jobs mit Jobkosten > 0 erhalten kann. Aus diesem Grund werden wir uns auf die Bestimmung der λ-Werte fokussieren, die Spieler 1 zugeordnet werden. Um λ_{12} zu bestimmen, darf unsere $\lambda_{12}(t)$-Funktionen keine

Informationen über die Jobkosten von Spieler 1 und 2 verwenden. Folglich seien die Jobkosten, welche sich in den ersten beiden Zeilen befinden unbekannt. Aus diesem Grund werden wir bei der Bestimmung von λ_{12} auf folgende Darstellung zugreifen:

$$t' = \begin{pmatrix} \infty & t_{12} & t_{13} & t_{14} & t_{15} \\ t_{21} & \infty & t_{23} & t_{24} & t_{25} \\ 1 & 0 & \infty & 0 & 0 \\ 1 & 0 & 0 & \infty & 0 \\ 1 & 0 & 0 & 0 & \infty \end{pmatrix}$$

Wir wollen nun überprüfen, ob sich anhand der Jobkosten der Spieler 3,4 und 5 feststellen lässt, wie λ_{12}, bzw. λ_{21} zu setzen sind, um den potenziellen Approximationsfaktor möglichst zu reduzieren. Um das zu beurteilen, wollen wir uns zunächst anhand der gegebenen Kosten anschauen, wie groß der Makespan werden kann, wenn Spieler 1 bzw. Spieler 2 alle Jobs zugeteilt bekommen.

Fall 1: Spieler 1 bekommt all seine Jobs zugeteilt.

Für $i \in \{3,4,5\}$ gilt $t_{i1} > 0$. Somit kann Spieler 1 auf jeden Fall drei Jobs mit Jobkosten > 0 zugeteilt bekommen. Wird $\{i,1\}$ an Spieler 1 zugeteilt, so gilt nach Lemma 7 $t_{1i} \leq \lambda_{\max} \cdot \text{OPT}$. In diesem Fall gilt $\tau_1(X_1) = t_{12} + t_{13} + t_{14} + t_{15} \leq 3 \cdot \lambda_{\max} + t_{12} \leq 3 \cdot \lambda_{\max} \cdot \text{OPT} + t_{12}$.
Wählen wir $\lambda_{12} > \lambda_{\max}$, so erhalten wir eine obere Schranke von $\tau_1(X_1) \leq 4 \cdot \lambda_{\max} \cdot \text{OPT}$. Für $\lambda_{12} < 1$ erhalten wir nach Lemma 8 eine obere Schranke von $\tau_1(X_1) \leq (3 \cdot \lambda_{\max} + 1) \cdot \text{OPT}$. Die obere Schranke ist hier für $n = 5$ unabhängig von $\lambda_{\max}$ stets $\geq (n-1) \cdot \text{OPT}$.

Fall 2: Spieler 2 bekommt all seine Jobs zugeteilt.

In diesem Fall können wir sehen, dass für $i \in \{3,4,5\}$ $t_{i2} = 0$ gilt. Spieler 2 kann also höchstens einen Job mit Jobkosten > 0 zugeteilt bekommen. Es gilt also mit Lemma 7 $\tau_2(X_2) = t_{21} \leq \lambda_{\max} \cdot \text{OPT}$. Wählen wir $\lambda_{21} < 1$, so erhalten wir hingegen eine obere Schranke von $\tau_2(X_2) \leq \text{OPT}$. Für $\lambda_{\max} < 4$ erhalten wir hier für $n = 5$ also eine obere Schranke $< (n-1) \cdot \text{OPT}$.

Es zeigt sich also, dass wenn wir $\lambda_{21} > \lambda_{12}$ wählen und $\lambda_{\max} < 4$ die Gesamtkosten von Spieler 2 nicht größer als $(n-1)\text{OPT}$ werden können. Auf der anderen Seite reduzieren wir auf diese Weise die potenziellen Gesamtkosten von Spieler 1. Daher wollen wir in diesem Fall $\lambda_{21} > \lambda_{12}$ setzen.
Dieses Beispiel ist nun so ausgelegt, dass wir für $i \in \{2,3,4,5\}$ bei der Bestimmung von λ_{1i} in Spalte 1 stets 3 Jobkosten mit Kosten 1 und in Spalte i nur Jobkosten mit Kosten 0 sehen werden. Aus diesem Grund würde die λ_{1i}-Funktion stets $\lambda_{1i} < 1$ setzen.
Auf diese Weise bekommt für die Eingabe t Spieler 1 keinen Job zugeteilt und wir erhalten $\text{SPAN} = 1 = \text{OPT}$.

Beispiel 8. In Beispiel 7 konnten wir mittels Jobkosten 0 argumentieren, weshalb es für $i \in \{2,3,4,5\}$ sinnvoll ist $\lambda_{1i} < 1$ zu setzen. Im folgenden Beispiel wollen wir die Jobkosten 0 durch einen beliebig kleinen Wert $1 > \epsilon > 0$ ersetzen. Zudem setzen wir $\lambda_{\max} = 1.5$. Es sei

$$t = \begin{pmatrix} \infty & 1 & 1 & 1 & 1 \\ 1 & \infty & \epsilon & \epsilon & \epsilon \\ 1 & \epsilon & \infty & \epsilon & \epsilon \\ 1 & \epsilon & \epsilon & \infty & \epsilon \\ 1 & \epsilon & \epsilon & \epsilon & \infty \end{pmatrix}.$$

Um λ_{12} zu bestimmen wollen wir ähnlich wie in Beispiel 7 auf folgende Darstellung zugreifen:

$$t' = \begin{pmatrix} \infty & t_{12} & t_{13} & t_{14} & t_{15} \\ t_{21} & \infty & t_{23} & t_{24} & t_{25} \\ 1 & \epsilon & \infty & \epsilon & \epsilon \\ 1 & \epsilon & \epsilon & \infty & \epsilon \\ 1 & \epsilon & \epsilon & \epsilon & \infty \end{pmatrix}.$$

Wir wollen uns nun darauf fokussieren, wie groß der Approximationsfaktor höchstens werden kann, wenn Spieler 1 bzw. Spieler 2 alle Jobs zugeteilt bekommen. Anhand der ϵ-Werte in Spalte 3, 4 und 5 gilt zunächst in jedem Fall $\text{OPT} \geq \epsilon$. Nun gibt es zwei Fälle die wir betrachten möchten.

Fall 1: Spieler 1 bekommt alle Jobs zugeteilt.

Wir nehmen hier an, dass mindestens ein Job außer $\{1,2\}$ nicht in einer optimalen Zuteilung ist. Ansonsten gilt mit Lemma 7 $\tau_1(X_1) = t_{12} + t_{13} + t_{14} + t_{15} \leq 1.5\text{OPT} + \text{OPT} = 2.5\text{OPT}$. Es sei also $\text{OPT} \geq 1 + \epsilon$. In diesem Fall gilt mit Lemma 7 $\tau_1(X_1) = t_{12} + \leq 3 \cdot 1.5 \leq 4.5\frac{\text{OPT}}{1+\epsilon} + 1.5\text{OPT} < 6\text{OPT}$. Mit $\epsilon \to \infty$ kann so eine obere Schranke von $6 \cdot \text{OPT}$ erreicht werden. Mit $\lambda_{12} < 1$ erhalten wir hingegen eine obere Schranke von $5.5 \cdot \text{OPT}$.

Fall 2: Spieler 2 bekommt alle Jobs zugeteilt.

Analog zu **Fall 1** sei dann $\text{OPT} \geq 2\epsilon$. Falls Spieler 2 alle Jobs zugeteilt bekommen, so erhalten wir $\tau_2(X_2) \leq 3 \cdot 1.5 \cdot \epsilon + 1.5\text{OPT} \leq 2.25\text{OPT} + 1.5\text{OPT} = 3.75\text{OPT}$. Mit $\lambda_{21} < 1$ würden wir hingegen eine obere Schranke von 3.25 erhalten.

Analog zu Beispiel 7 können wir nun folgern, dass eine Setzung von $\lambda_{12} < \lambda_{21}$ den potenziellen Approximationsfaktor reduziert. Wie auch in Beispiel 7 ist die Eingabe t so gewählt, dass wir für $i \in \{2,3,4,5\}$ bei der Bestimmung von λ_{1i} stets einen analogen Fall 1 und Fall 2 vorliegen haben und folglich stets $\lambda_{1i} < \lambda_{i1}$ wählen würde.
Für die Eingabe t gilt $\text{OPT} = 1 + 2\epsilon$. Unser Algorithmus würde hier Spieler 1 keinen Job

zuteilen und folglich würden wir SPAN $\leq 1 + 3\epsilon$ erhalten. Damit würde der Approximationsfaktor für $n = 5$ kleiner als $n - 1$ sein.

In den Beispielen 7 und 8 sind wir so vorgegangen, dass wir bei der Bestimmung von λ_{12} zunächst einmal die Zeilen 1 und 2 nicht berücksichtigt haben. Wir haben diese also gestrichen. Anschließend haben wir uns stets zwei Fälle betrachtet. Im Fall 1 haben wir zusätzlich die Spalte 2 und für den Fall 2 Spalte 1 jeweils nicht berücksichtigt. In beiden Fällen haben wir den potenziellen Makespan bestimmt, sofern Spieler 1 bzw. Spieler 2 alle Jobs zugeteilt bekommt. Im Folgenden möchten wir λ_{ij} konkret berechnen können. Wir möchten hierfür zwei unabhängige Bewertungen s_{ij} und s_{ji} einführen. s_{ij} ist die Bewertung, welche gebildet wird, wenn Spalte i gestrichen wird, während s_{ji} die Bewertung ist, wenn Spalte j gestrichen wird. Zudem wird in beiden Fällen die Zeile i und j gestrichen. Je größer die Bewertung s_{ij} ist, desto größer schätzen wir also den potenziellen Approximationsfaktor ein, wenn Spieler j den Makespan bestimmt. Wenn $s_{ij} > s_{ji}$ entspricht, so wollen wir also, dass $\lambda_{ij} > \lambda_{ji}$ erfüllt ist.

Nun wollen wir, dass obige Bedingungen bei der Bestimmung von λ_{ij} eingehalten werden. Dies lässt sich wie folgt umsetzen:

Falls $s_{ij} > s_{ji}$ dann setze $\lambda_{ij} = \min\left\{\lambda_{\max}, \frac{s_{ij}}{s_{ji}}\right\}$. Ansonsten setze $\lambda_{ij} = \max\left\{\lambda_{\min}, \frac{s_{ij}}{s_{ji}}\right\}$.
Damit ist garantiert, dass obige Bedingungen eingehalten werden.
Hierbei haben wir Divisionen durchgeführt. Aus diesem Grund wollen wir die Annahmen treffen, dass $\lambda_{ij} = 1$, falls $s_{ij} = s_{ji} = 0$ und $\lambda_{ij} = \lambda_{\max}$ falls $s_{ij} > 0$ und $s_{ji} = 0$.

Im Folgenden möchten wir nun λ-Funktionen einführen, welche wie auch in der Intuition zwei Bewertungen verwenden um λ_{ij} zu bestimmen. Für diese werden wir jedoch andere Variablennamen einführen. Anschließend werden wir diese Funktionen experimentell oder analytisch untersuchen. Bei der experimentellen Untersuchung werden wir ähnlich wie im Abschnitt 4.2 beschrieben vorgehen. Wir werden jedoch später sehen, dass wir mit der experimentellen Untersuchung an Grenzen stoßen, weshalb wir ab der λ_3-Funktion rein analytisch weiter arbeiten werden.

5.1 λ_1-Funktion

Im folgenden Algorithmus berechnen wir zwei optimale Makespans OPT_{ij} bzw. OPT_{ji}. Anders als bei OPT, berechnen sich die beiden optimalen Zuteilungen wie folgt: Wird t_{lm} gestrichen t_{ml} aber nicht, so teilen wir Spieler m den Job $\{m, l\}$ zu. Wird hingegen t_{lm} und t_{ml} gestrichen, so wird der Job $\{l, m\}$ nicht zugeteilt. Alle anderen Zuteilungen erfolgen so, dass OPT_{ij} bzw. OPT_{ji} minimiert wird. Die Idee besteht nun darin, dass falls $\text{OPT}_{ij} > \text{OPT}_{ji}$, wir davon ausgehen, dass der potenzielle Makespan von Spieler j größer als der von Spieler i ist. Daher haben wir ein Interesse daran $\lambda_{ij} > \lambda_{ji}$ zu setzen.

Der Algorithmus sieht wie folgt aus:

Algorithmus 1:
1. Streiche Zeile i und j aus der t-Matrix.

Bestimmung von OPT_{ij}:
2.1 Streiche Spalte i aus der t-Matrix.
2.2 Anschließend bestimme die optimale Zuteilung dieser neuen t-Matrix.
2.3 Gib OPT_{ij} als den Makespan dieser Zuteilung aus.

Bestimmung von OPT_{ji}:
3.1. Streiche Spalte j aus der t-Matrix.
3.2. Anschließend bestimme die optimale Zuteilung dieser neuen t-Matrix.
3.3. Gib OPT_{ji} als den Makespan dieser Zuteilung aus.

4.1 Falls $\text{OPT}_{ij} > \text{OPT}_{ji}$, setze $\lambda_{ij} = \min\left\{\lambda_{\max}, \dfrac{\text{OPT}_{ij}}{\text{OPT}_{ji}}\right\}$.
4.2 Ansonsten setze $\lambda_{ij} = \max\left\{\lambda_{\min}, \dfrac{\text{OPT}_{ij}}{\text{OPT}_{ji}}\right\}$.

Beispiel 9. Nun wollen wir uns ein einfaches Beispiel anschauen, um zu zeigen, dass diese λ-Funktion für gewisse Instanzen tatsächlich ganz gut funktioniert. Als Eingabe sei folgende Matrix gegeben:

$$t = \begin{pmatrix} \infty & 1 & 1 & 1 & 1 \\ 1 & \infty & 0 & 0 & 0 \\ 1 & 0 & \infty & 0 & 0 \\ 1 & 0 & 0 & \infty & 0 \\ 1 & 0 & 0 & 0 & \infty \end{pmatrix}.$$

Wählen wir $\lambda_{\max} = 2$, dann erhalten wir

$$\lambda = \begin{pmatrix} \infty & 0.5 & 0.5 & 0.5 & 0.5 \\ 2 & \infty & 1 & 1 & 1 \\ 2 & 1 & \infty & 1 & 1 \\ 2 & 1 & 1 & \infty & 1 \\ 2 & 1 & 1 & 1 & \infty \end{pmatrix} x = \begin{pmatrix} \infty & 0 & 0 & 0 & 0 \\ 1 & \infty & 1 & 1 & 1 \\ 1 & 0 & \infty & 1 & 1 \\ 1 & 0 & 0 & \infty & 1 \\ 1 & 0 & 0 & 0 & \infty \end{pmatrix}.$$

Für $j \in \mathcal{N} \setminus \{1\}$ gilt stets $\text{OPT}_{j1} = 1$ und $\text{OPT}_{1j} = 0$. Folglich wird $\lambda_{j1} = 2$ gesetzt. Das führt letztlich dazu, dass Spieler 1 keinen Job zugeteilt bekommt und unser Algorithmus für diese Instanz sogar einen optimalen Makespan ausgibt.

5.1.1 Tests

Testreihe 4: Für unsere konstante λ-Funktion haben wir gesehen, dass wenn wir nur die Kosten 0 und 1 zulassen, es bereits möglich ist, einen Approximationsfaktor von mindestens $n-1$ zu erreichen. Aus diesem Grund wollen wir zunächst überprüfen, ob die λ_1-Funktion hier über alle Instanzen, in denen Jobkosten 0 und 1 zugelassen sind, einen besseren Approximationsfaktor erreichen kann. Hierbei seien die t_{ij}-Werte gemäß folgender Verteilung gewählt: Es gilt $\mathbb{P}(t_{ij} = 1) = 0.7$ und $\mathbb{P}(t_{ij} = 0) = 0.3$. Weiter setzen wir $\lambda_{\max} = 2$. Die Histogramme zur Verteilung der Approximationsfaktoren für K_5 und K_6 sehen dann wie folgt aus:

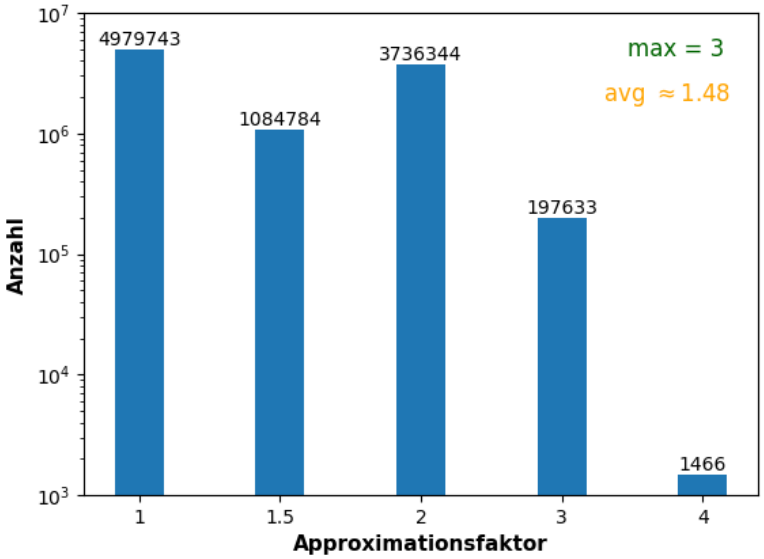

Abbildung 7: Testreihe 4 für K_5

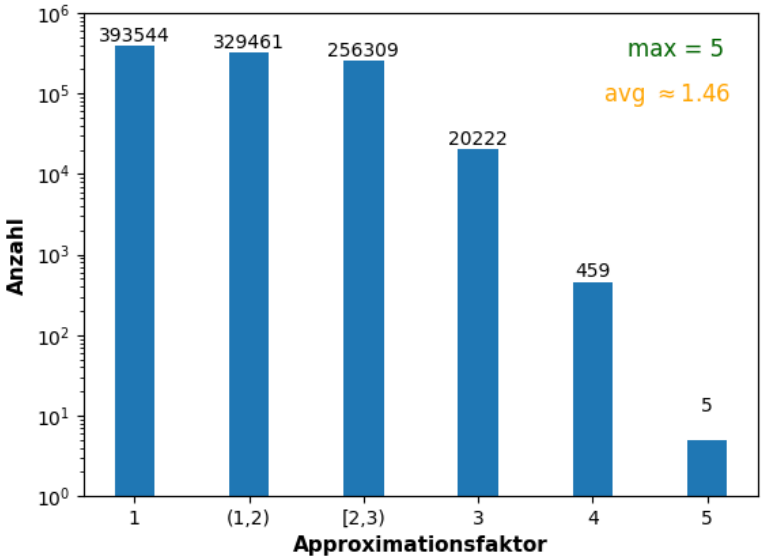

Abbildung 8: Testreihe 4 für K_6

Im Vergleich zur konstanten λ-Funktion, bei welcher wir für $i, j \in \mathcal{N}$ mit $i \neq j$ $\lambda_{ij} = 1$ gesetzt haben, hat sich die Anzahl der zufälligen Instanzen, bei welchen ein Approximationsfaktor von mindestens $n-1$ erreicht wurde, drastisch verringert. Auf der anderen Seite sieht es so aus, dass wenn wir die Werte λ_{ij} mit Hilfe der λ_1-Funktion bestimmen, wir stets einen Approximationsfaktor von mindestens $n-1$ erhalten werden. Für den K_5 erhalten wir folgendes Worst-Case-Beispiel.

Beispiel 10.

$$t = \begin{pmatrix} \infty & 1 & 1 & 1 & 1 \\ 1 & \infty & 0 & 0 & 1 \\ 1 & 1 & \infty & 0 & 0 \\ 1 & 0 & 1 & \infty & 0 \\ 1 & 0 & 0 & 1 & \infty \end{pmatrix} \lambda = \begin{pmatrix} \infty & 1 & 1 & 1 & 1 \\ 1 & \infty & 1 & 1 & 1 \\ 1 & 1 & \infty & 1 & 1 \\ 1 & 1 & 1 & \infty & 1 \\ 1 & 1 & 1 & 1 & \infty \end{pmatrix} x = \begin{pmatrix} \infty & 1 & 1 & 1 & 1 \\ 0 & \infty & 1 & 1 & 1 \\ 0 & 0 & \infty & 0 & 0 \\ 0 & 0 & 1 & \infty & 0 \\ 0 & 0 & 1 & 1 & \infty \end{pmatrix}$$

Eine optimale Zuteilung besteht darin, dass Spieler 1 keinen Job zugeteilt bekommt. In diesem Fall erhalten wir $\text{OPT} = 1$. Andererseits teilt der Algorithmus in vier Tiebreaks Spieler 1 alle seine Jobs zu. Obwohl sich nur eine Eins in Spalte 2,3,4 und 5 befindet, kann der Algorithmus nicht erkennen, dass es hier sinnvoller ist, Spieler 1 nicht zu bevorzugen und somit für $i \in \{2, 3, 4, 5\}$ $\lambda_{1i} < 1$ zu setzen.
Nun wollen wir Algorithmus 1 nochmals genauer anschauen.

5.1.2 Analyse

Anhand der λ-Werte von Beispiel 10 ist zu sehen, dass bei der Bestimmung von OPT_{ji} und OPT_{ij} lediglich der größte Eintrag $\max_{ij} = \max_{k \in \mathcal{N}, k \neq i} t_{ki}$ oder $\max_{ji} = \max_{k \in \mathcal{N}, k \neq j} t_{kj}$ relevant sind. Dies führte letztlich dazu, dass wir beim Testen einen Approximationsfaktor von $n-1$ erreichen konnten. Im Folgenden möchten wir einen Approximationsfaktor für unseren Mechanismus über alle Instanzen mit beliebigen $t_{ij} \geq 0$ angeben.

Satz 10. *Für alle $\lambda_{\max} \geq 1$ erhält man mit Hilfe der λ_1-Funktion einen Approximationsfaktor von exakt $\lambda_{\max} \cdot (n-1)$.*

Beweis. Setze

$$t_{ij} = \begin{cases} \lambda_{\max} - \epsilon, \text{falls} \quad i = 1 \\ 1, \quad \text{falls} \quad j = 1 \\ \lambda_{\max}, \text{falls} \quad i = j+1 \text{ oder } i = 2, j = n \\ 0, \textit{sonst} \end{cases}$$

Ein Beispiel für den K_5 sieht wie folgt aus:

Beispiel 11.

$$t = \begin{pmatrix} \infty & \lambda_{\max} - \epsilon & \lambda_{\max} - \epsilon & \lambda_{\max} - \epsilon & \lambda_{\max} - \epsilon \\ 1 & \infty & 0 & 0 & \lambda_{\max} \\ 1 & \lambda_{\max} & \infty & 0 & 0 \\ 1 & 0 & \lambda_{\max} & \infty & 0 \\ 1 & 0 & 0 & \lambda_{\max} & \infty \end{pmatrix}$$

Für $k \in \mathcal{N}$ gilt bei der Berechnung λ_{k1}, dass $\text{OPT}_{k1} = 1$ und $\text{OPT}_{1k} = \lambda_{\max}$ gilt. Dies führt dazu, dass $\lambda_{1k} = \lambda_{\max}$ gesetzt wird. In diesem Fall bekommt Spieler 1 alle Jobs zugeteilt, obwohl, die optimale Zuteilung darin besteht, dass Spieler 1 keinen Job bekommt. In diesem Fall erhalten wir $\text{OPT} = 1$.
Seine Gesamtkosten sind $\text{SPAN} = (n-1) \cdot (\lambda_{\max} - \epsilon)$. Mit $\lim_{\epsilon \to 0}(n-1) \cdot (\lambda_{\max} - \epsilon) = (n-1) \cdot \lambda_{\max}$ folgt so unser gewünschter Approximationsfaktor. Nach Lemma 9. kann dieser auch nicht höher ausfallen. □

5.2 λ_2-Funktion

Für die folgende λ-Funktion gibt es die Bewertungen SUM_{ij} und SUM_{ji}. Bei der λ_1-Funktion haben wir gesehen, dass bei der Bestimmung von OPT_{ij} und OPT_{ji} jeweils der größte Wert der Spalte j, bzw. i ohne Zeile i und j relevant sind, um zu entscheiden, ob $\lambda_{ij} > \lambda_{ji}$ gesetzt werden soll. Nun wollen wir jeden einzelnen Wert der Spalte j berücksichtigen, um auf diese Weise für eine hoffentlich größere Teilmenge an Instanzen einen Approximationsfaktor unter $n-1$ zu erreichen. Der Algorithmus für diese Funktion zur Bestimmung von λ_{ij} ist wie folgt

definiert:

Algorithmus 2:

1. Setze $\text{SUM}_{ij} = \sum_{k=1,k\notin\{i,j\}}^{n} t_{kj}$

2. Setze $\text{SUM}_{ji} = \sum_{k=1,k\notin\{i,j\}}^{n} t_{ki}$

3.1 Falls $\text{SUM}_{ij} > \text{SUM}_{ji}$, setze $\lambda_{ij} = \min\left\{\lambda_{\max}, \frac{\text{SUM}_{ij}}{\text{SUM}_{ji}}\right\}$

3.2 Ansonsten setze $\lambda_{ij} = \max\left\{\lambda_{\min}, \frac{\text{SUM}_{ij}}{\text{SUM}_{ji}}\right\}$

Beispiel 12. Hier möchten wir eine Intuition dafür geben, weshalb wir hier die λ_2-Funktion so definiert haben. Hierfür betrachten wir die Eingabe t aus Beispiel 10 und zeigen, dass unsere λ_2-Funktion hier sogar eine optimale Zuteilung angeben kann. Setze $\lambda_{\max} = 2$.

$$t = \begin{pmatrix} \infty & 1 & 1 & 1 & 1 \\ 1 & \infty & 0 & 0 & 1 \\ 1 & 1 & \infty & 0 & 0 \\ 1 & 0 & 1 & \infty & 0 \\ 1 & 0 & 0 & 1 & \infty \end{pmatrix} \lambda = \begin{pmatrix} \infty & 0.5 & 0.5 & 0.5 & 0.5 \\ 2 & \infty & 2 & 1 & 0.5 \\ 2 & 0.5 & \infty & 2 & 1 \\ 2 & 1 & 0.5 & \infty & 2 \\ 2 & 2 & 1 & 0.5 & \infty \end{pmatrix} x = \begin{pmatrix} \infty & 0 & 0 & 0 & 0 \\ 1 & \infty & 1 & 1 & 0 \\ 1 & 0 & \infty & 1 & 1 \\ 1 & 0 & 0 & \infty & 1 \\ 1 & 1 & 0 & 0 & \infty \end{pmatrix}$$

Für $j \in \{2,3,4,5\}$ gilt stets $\text{SUM}_{j1} = 3$ und $\text{SUM}_{1j} = 1$. Folglich wird $\lambda_{j1} = 2$ gesetzt und Spieler 1 bekommt so keinen Job zugeteilt. Dies führt letztlich dazu, dass der Algorithmus hier eine Zuteilung mit einem Makespan von 1 berechnet und somit auch eine optimale Zuteilung angibt.
Im Folgenden möchten wir nun unsere λ_2-Funktion testen. Hierfür wollen wir überprüfen, ob wir experimentell einen Approximationsfaktor unter $n-1$ erreichen können, sofern nur die Jobkosten 0 oder 1 zugelassen werden.

5.2.1 Tests

Testreihe 5. Wir haben bei der λ_1-Funktion gesehen, dass selbst wenn für $i,j \in \mathcal{N}$ mit $i \neq j$ $t_{ij} \in \{0,1\}$ erfüllt ist, es nicht möglich ist, einen Approximationsfaktor unter $n-1$ anzugeben. Mit der folgenden Testreihe wollen wir überprüfen, wie robust unsere λ_2-Funktion für diesen einfachen Fall ist. Hierfür setzen wir $\lambda_{\max} = 2$. Zusätzlich seien t_{ij} Zufallsvariablen, die gemäß folgender Verteilung unabhängig gewählt sind: $\mathbb{P}(t_{ij} = 1) = 0.7$ und $\mathbb{P}(t_{ij} = 0) = 0.3$. Die Histogramme zur Verteilung der Approximationsfaktoren für K_5 und K_6 sehen dann wie folgt aus:

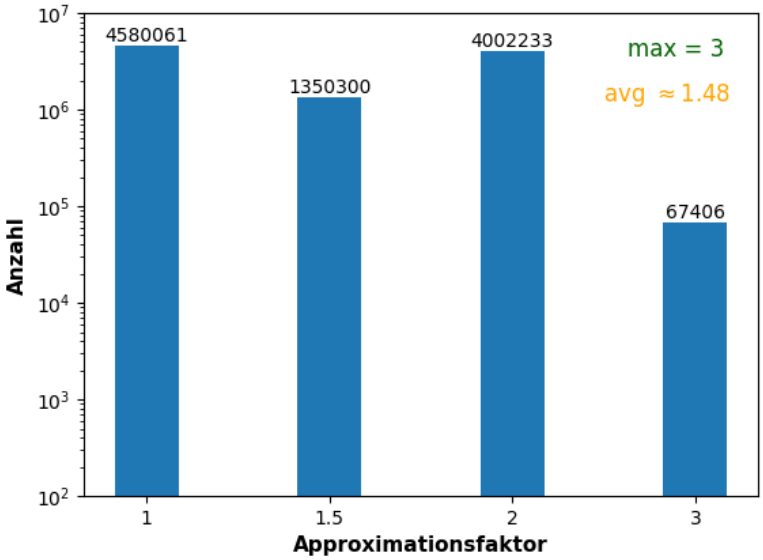

Abbildung 9: Testreihe 5 für den K_5

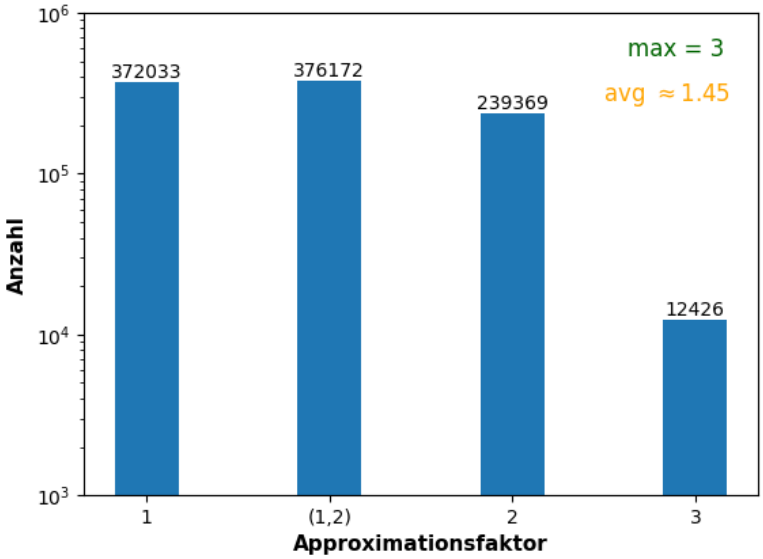

Abbildung 10: Testreihe 5 für den K_6

Es zeigt sich (siehe Abbildung 9 und Abbildung 10), dass wir für K_5 und K_6 einen Approximationsfaktor von mindestens 3 erreichen, sofern wir nur die Jobkosten 0 oder 1 zulassen. Die Ergebnisse lassen vermuten, dass wir für $t_{ij} \in \{0, 1\}$ und sehr große n einen wesentlich besseren Approximationsfaktor als $n - 1$ erreichen können. Dies werden wir uns später in der Analyse genauer ansehen. Ein mögliches Worst-Case-Beispiel aus den betrachteten zufälligen Instanzen des K_5 sieht wie folgt aus:

Beispiel 13.

$$t = \begin{pmatrix} \infty & 0 & 1 & 1 & 1 \\ 1 & \infty & 1 & 1 & 1 \\ 1 & 1 & \infty & 1 & 0 \\ 0 & 1 & 1 & \infty & 1 \\ 0 & 1 & 1 & 0 & \infty \end{pmatrix} \lambda = \begin{pmatrix} \infty & 2 & 2 & 1 & 1 \\ 0.5 & \infty & 1.5 & 1 & 1 \\ 0.5 & \frac{2}{3} & \infty & \frac{2}{3} & 1 \\ 1 & 1 & 1.5 & \infty & \frac{2}{3} \\ 1 & 1 & 1 & 1.5 & \infty \end{pmatrix} x = \begin{pmatrix} \infty & 1 & 1 & 0 & 0 \\ 0 & \infty & 1 & 1 & 1 \\ 0 & 0 & \infty & 0 & 1 \\ 1 & 0 & 1 & \infty & 0 \\ 1 & 0 & 0 & 1 & \infty \end{pmatrix}$$

Hier ist zu sehen, dass Spieler 2 einen Job direkt zugeteilt bekommt und zwei weitere in je einem Teilbreak kriegt. Alle anderen Spieler bekommen höchstens einen Job, welcher die Jobkosten 1 besitzt. Eine optimale Zuteilung mit OPT=1 besteht hier jedoch darin, wenn Spieler 2 nur einen Job zugeteilt bekommt. Daraus resultiert für diese Instanz ein Approximationsfaktor von 3.

Testreihe 6: Nun wollen wir testen, wie unsere λ_2-Funktion reagiert, sofern wir neben den Jobkosten 0 und 1 weitere Jobkosten zulassen, die nicht all zu weit von der 1 entfernt liegen. Hierfür setzen wir $\lambda_{\max} = 1.25$. t_{ij} seien unabhängige Zufallsvariablen mit folgender Verteilung: $\mathbb{P}(t_{ij} = 0) = 0.3$ und $\mathbb{P}(t_{ij} = 1) = 0.3$, $\mathbb{P}(t_{ij} = 1.25) = 0.2$, $\mathbb{P}(t_{ij} = 0.8) = 0.2$.
Die Histogramme zur Verteilung der Approximationsfaktoren für K_5 und K_6 sehen dann wie folgt aus:

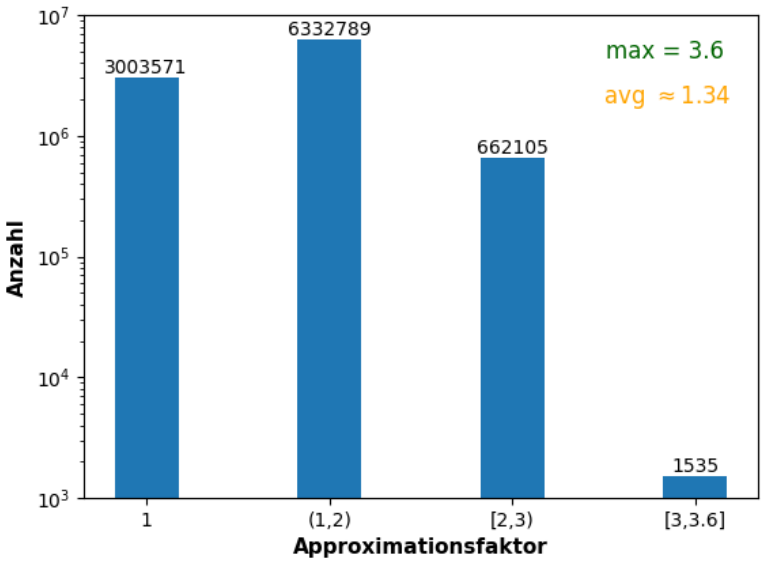

Abbildung 11: Testreihe 6 für den K_5

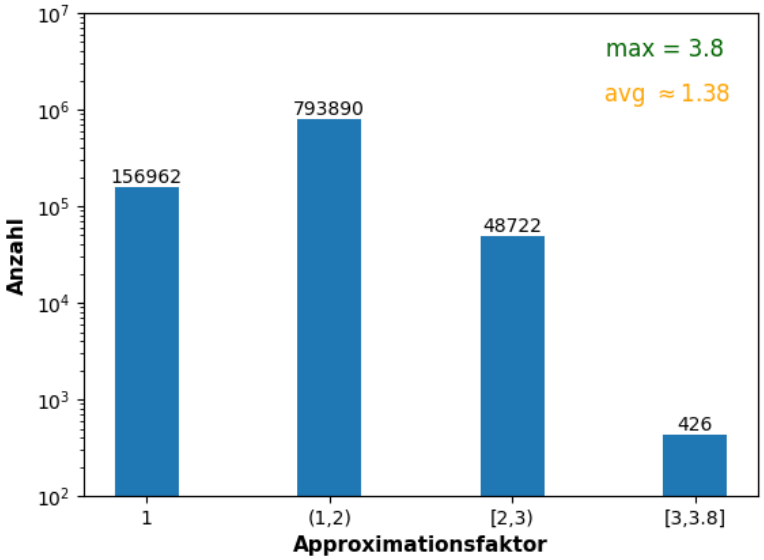

Abbildung 12: Testreihe 6 für den K_6

Die berechneten Testfälle (siehe Abbildung 11 und Abbildung 12) weisen darauf hin, dass wenn die Jobkosten bis auf die 0 nicht all zu weit auseinander liegen, es vermutlich auch möglich ist, für solche Instanzen einen Approximationsfaktor unter $n-1$ zu beweisen. Das getestete Worst-Case-Beispiel für den K_5 sieht wie folgt aus:

Beispiel 14.

$$t = \begin{pmatrix} \infty & 0.8 & 1 & 0.8 & 1 \\ 1 & \infty & 1.25 & 0 & 0 \\ 1 & 0 & \infty & 0 & 0.8 \\ 1 & 0 & 1.25 & \infty & 0 \\ 1.25 & 0 & 0.8 & 1.25 & \infty \end{pmatrix} \lambda = \begin{pmatrix} \infty & 0.8 & \frac{66}{65} & 0.8 & 0.8 \\ 1.25 & \infty & 1.25 & 1.25 & 1.25 \\ \frac{65}{66} & 0.8 & \infty & 0.8 & 0.8 \\ 1.25 & 0.8 & 1.25 & \infty & 1.25 \\ 1.25 & 0.8 & 1.25 & 0.8 & \infty \end{pmatrix} x = \begin{pmatrix} \infty & 1 & 1 & 1 & 1 \\ 0 & \infty & 0 & 1 & 0 \\ 0 & 1 & \infty & 1 & 0 \\ 0 & 0 & 0 & \infty & 1 \\ 0 & 1 & 1 & 0 & \infty \end{pmatrix}$$

Hier ist zu sehen, dass Spieler 1 den Job $\{1,3\}$ direkt zugeteilt bekommt und alle anderen Jobs im Tiebreak erhält. Der nächste Testfall wird jedoch zeigen, dass wenn die Jobkosten außer der 0 nicht mehr in einem hinreichend kleinen Intervall um die 1 liegen, es nicht möglich sein wird, einen Approximationsfaktor echt kleiner als $n-1$ zu erreichen.

Testreihe 7: Im Folgenden möchten wir testen, wie sich unsere λ-Funktion verhält, falls neben den 0 und 1 sehr große Werte als Jobkosten zugelassen werden. Hierfür setzen wir $\lambda_{\max} = 1.25$. t_{ij} sind unabhängige Zufallsvariablen mit folgender Verteilung: $\mathbb{P}(t_{ij} = 0) = 0.4$ und $\mathbb{P}(t_{ij} = 1) = 0.3$ und $\mathbb{P}(t_{ij} = 100) = 0.3$. Die Histogramme, welche die Verteilung der Approximationsfaktoren für den K_5 und K_6 angeben, sehen dann wie folgt aus:

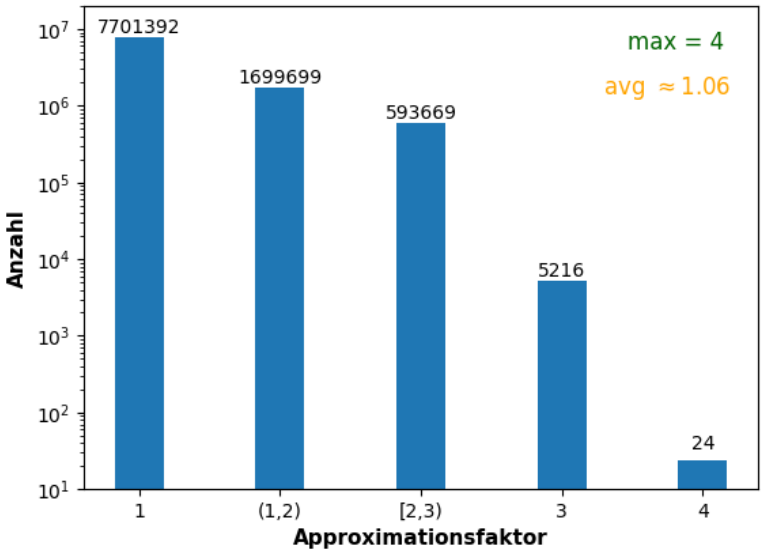

Abbildung 13: Testreihe 7 für den K_5

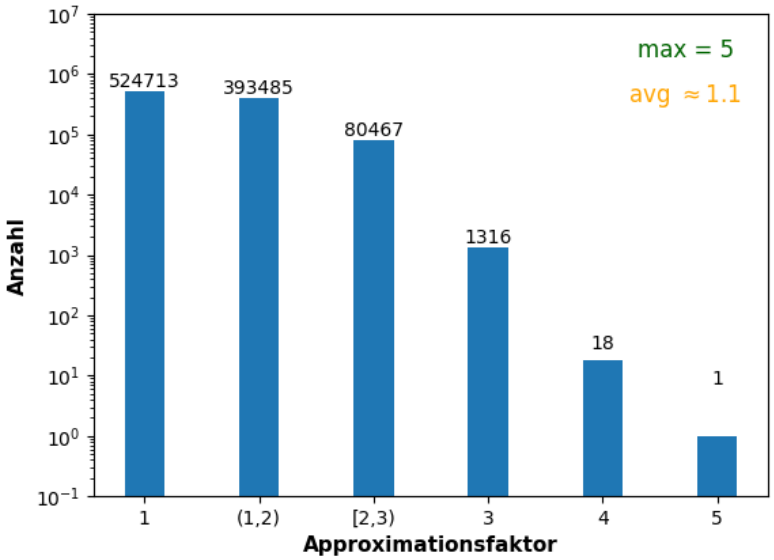

Abbildung 14: Testreihe 7 für den K_6

Es zeigt sich (siehe Abbildung 13 und Abbildung 14), dass wir für $5 \leq n \leq 6$ einen Approximationsfaktor von mindestens $n-1$ erreichen können. Ein mögliches Worst-Case-Beispiel für K_5 sieht wie folgt aus:

Beispiel 15.

$$t = \begin{pmatrix} \infty & 100 & 1 & 0 & 0 \\ 0 & \infty & 1 & 100 & 0 \\ 1 & 1 & \infty & 1 & 1 \\ 100 & 0 & 1 & \infty & 100 \\ 100 & 1 & 1 & 0 & \infty \end{pmatrix} \lambda = \begin{pmatrix} \infty & 0.8 & 0.8 & 1 & 1 \\ 1.25 & \infty & 0.8 & 0.8 & 1 \\ 1.25 & 1.25 & \infty & 1.25 & 1.25 \\ 1 & 1.25 & 0.8 & \infty & 0.8 \\ 1 & 1 & 0.8 & 1.25 & \infty \end{pmatrix} x = \begin{pmatrix} \infty & 0 & 0 & 1 & 1 \\ 1 & \infty & 0 & 0 & 1 \\ 1 & 1 & \infty & 1 & 1 \\ 0 & 1 & 0 & \infty & 0 \\ 0 & 0 & 0 & 1 & \infty \end{pmatrix}$$

Es ist zu erkennen, dass in jeder Spalte der Eingabe t mindestens eine 100 vorhanden ist. Die drei Einsen in Spalte 3 unterliegen der 100, s.d. für alle $i \in \{1, 2, 4, 5\} \quad \lambda_{i3} = \lambda_{\min} = 0.8$ gesetzt wird. Das führt dazu, dass Spieler 3 alle 4 möglichen Jobs zugeteilt bekommt. In einer optimalen Verteilung, dürfte Spieler 3 aber höchstens einen Job zugeteilt kriegen. Dies führt letztlich dazu, dass wir hier einen Approximationsfaktor von 4 erreichen. In der Analyse werden wir später klären, wie groß der Approximationsfaktor für beliebige $t_{ij} \in \mathbb{R}_{\geq 0}$, werden kann.

5.2.2 Analyse

Die Testreihe 5 (siehe Abbildung 9 und Abbildung 10) hat uns ein recht interessantes Verhalten gezeigt, wenn nur Nullen und Einsen als mögliche Jobkosten zugelassen werden. Dieses Verhalten möchten wir hier ein wenig weiter analysieren.

Definition 2. Im Folgenden sei $\text{ONES}_i = \{j \in \mathcal{N} \setminus \{i\} | t_{ji} = 1\}$

Satz 11. *Für alle $i, j \in \mathcal{N}$ mit $i \neq j$ sei $t_{i,j} \in \{0, 1\}$. Weiter werden die λ_{ij}-Werte durch die λ_2-Funktion bestimmt. Dann erhalten wir für $n \geq 3$ einen Approximationsfaktor von höchstens $\left\lceil \frac{n-1}{\sqrt{2}} \right\rceil$*

Beweis. Für den Beweis treffen wir folgende Annahmen:

- wir nehmen o.B.d.A. an, dass Spieler 1 den Makespan bestimmt. Es gilt also $\text{SPAN} = \sum_{j=2}^{n} t_{1j} \cdot x_{1j}$. Weiter nehmen wir o.B.d.A. an, dass $t_{21} \geq \ldots \geq t_{n1}$ erfüllt ist.
- Weiter nehmen wir an, dass $\text{ONES}_1 \geq 1$ erfüllt ist und somit $\text{OPT} \geq 1$ gilt. Ansonsten kann Spieler 1 keinen Job der Größe 1 zugeteilt bekommen und wir erhalten $\text{OPT}=0$.
- Zudem nehmen wir an, dass für $j \in \mathcal{N}$ $t_{1j} = 1$ gilt. Falls $t_{1j} = 0$ entspricht, kann Spieler 1 zwar den Job $\{1, j\}$ erhalten, jedoch wird der Makespan sich dadurch nicht erhöhen. Zwar kann durch eine solche Setzung sich OPT erhöhen, s.d. $\text{OPT} \geq 2$ erfüllt ist, aber in diesem Fall wäre der Approximationsfaktor $\leq \frac{n-1}{2}$. Da wir jedoch einen größeren Approximationsfaktor mit $\text{OPT} = 1$ zeigen wollen, müssen wir uns um diesen Fall nicht kümmern.
- weiter nehmen wir o.B.d.A. an, dass Spieler 1 jeden Tiebreak gewinnt.

Behauptung 1. *Sei* $\text{ONES}_1 = k$ *dann erhalten wir einen Approximationsfaktor von höchstens* k*.*

Beweis. Nach unserer Annahme gilt für $i \in \mathcal{N} \setminus \{1\}$ $t_{1i} = 1$. Falls $t_{i1} = 0$, so kann Spieler 1 den Job $\{i, 1\}$ wegen $t_{1i} = 1 > 0 = t_{i1} \cdot \lambda_{1i}$ nicht zugeteilt bekommen. Somit verbleiben k Fälle in denen Spieler 1 einen Job zugeteilt bekommen kann. Da wir $\text{OPT} \geq 1$ annehmen, erhalten wir einen Approximationsfaktor von höchstens k. □

Behauptung 2. *Sei* $\text{ONES}_1 = k$*,* OPT*=1, so können für* $i, j \in \mathcal{N} \setminus \{1\}$ *mit* $i \neq j$ *höchstens* $n - k + \frac{(n-1)(n-2)}{2}$ t_{ij}*-Einträge auf 1 gesetzt werden.*

Beweis. Für $i, j \in \mathcal{N} \setminus \{1\}$ mit $i \neq j$ werden die Jobkosten auf dem K_{n-1} gesetzt. Hier gibt es aber lediglich $\frac{(n-1)(n-2)}{2}$ Jobs. Gilt $t_{ij} = 1$ und $t_{ji} = 1$, so erhöhen sich die Gesamtkosten von Spieler i oder Spieler j um 1. Somit können höchstens $\frac{(n-1)(n-2)}{2}$ Jobkosten auf 1 gesetzt werden, s.d. die Gesamtkosten jedes Spielers sich nicht verändern.
Nun haben wir $n - 1 - k$ Einträge mit $t_{i1} = 0$. Zusätzlich kann für einen Eintrag $t_{i1} = 1$ gelten, dass $\{i, 1\}$ in einer optimalen Zuteilung Spieler 1 zugewiesen wird. Es können also bis zu $n - k$ zusätzliche Einsen platziert werden, s.d $\text{OPT} = 1$ erfüllt bleibt. □

Behauptung 3. *Sei* $ones_1 = k$*,* $t_{i1} = 1$*,* $\text{OPT} = 1$ *so darf* SUM_{1i} *höchstens den Wert 0 oder* $k - 1$ *annehmen, um den potenziellen Approximationsfaktor zu maximieren.*

Beweis. Es gilt nach Konstruktion $\text{SUM}_{i1} = k - 1$. Nach Behauptung 2 ist die Anzahl der zu setzenden Einsen beschränkt. Zudem können wir annehmen, dass Spieler 1 in einem Tiebreak stets einen Job zugeteilt bekommt. Damit Spieler 1 Job $\{1, i\}$ zugeteilt bekommen muss $\text{SUM}_{1i} \geq k - 1$ erfüllt sein. Kann Spieler 1 also Job $\{i, j\}$ zugeteilt bekommt, so muss SUM_{1i} minimal gewählt werden. Das ist für $\text{SUM}_{i1} = k - 1$ der Fall. Kann dieser hingegen Job $\{i, 1\}$ nur dann zugeteilt bekommen, wenn OPT erhöht wird, so muss $\text{SUM}_{i1} = 0$ gesetzt werden. □

Nach Behauptung 2 wissen wir also, dass höchstens $n - k + \frac{(n-1)(n-2)}{2}$ zusätzliche Einsen platziert werden dürfen und nach Behauptung 3, dass eine höhere Setzung als $\text{SUM}_{1i} = k-1$ den Approximationsfaktor höchstens verringert. Zudem wissen wir nach Behauptung 1, dass für $\text{ONES}_1 = k$ der Approximationsfaktor höchstens k ist. Um Letzteres zu erreichen, werden also $k \cdot (k-1) \leq \frac{(n-1)(n-2)}{2} + n - k$ Einsen benötigt, wobei k hier maximal groß sein soll. Wir nehmen $k \geq \frac{n+1}{2}$ an. Wegen $\left\lceil \frac{n-1}{\sqrt{2}} \right\rceil \geq \frac{n+1}{2}$ für $n \geq 3$ können wir das auch annehmen. Dann gilt $k \cdot (k-1) \leq \frac{(n-1)^2}{2}$ mit $k - 1 < \sqrt{(k-1) \cdot k} \leq \frac{n-1}{\sqrt{2}}$. Damit aber muss $k \leq \left\lceil \frac{n-1}{\sqrt{2}} \right\rceil$ erfüllt sein. □

Wie der Testfall 7 (siehe Abbildung 13 und Abbildung 14) zeigt, konnten wir für die dort getesteten Instanzen, einen Approximationsfaktor von mindestens $n - 1$ erreichen. Im Folgenden möchten wir nun zeigen, dass für beliebige Instanzen der Approximationsfaktor sogar noch größer ist.

Satz 12. *Sei $\lambda_{\max} > 1$ beliebig und für alle $i, j \in \mathcal{N}$ mit $i \neq j$ sei $t_{ij} \in \mathbb{R}_{\geq 0}$. Weiter werden die λ_{ij}-Werte durch die λ_2-Funktion bestimmt. Dann ist der Approximationsfaktor exakt $\lambda_{\max} \cdot (n-1)$.*

Beweis. Für $i, j \in \mathcal{N}$ mit $i \neq j$ mit $\epsilon > 0$ hinreichend klein setzen wir:

$$t_{ij} = \begin{cases} \lambda_{\max} - \epsilon, \text{falls} \quad i = 1 \\ 1, \quad \text{falls} \quad j = 1 \\ n\lambda_{\max}, \text{falls} \quad i = j + 1 \quad \text{oder } i = 2, j = n \\ 0, \textit{sonst} \end{cases}$$

Die folgende Matrix zeigt diese Instanz für K_5:

Beispiel 16.

$$t = \begin{pmatrix} \infty & \lambda_{\max} - \epsilon & \lambda_{\max} - \epsilon & \lambda_{\max} - \epsilon & \lambda_{\max} - \epsilon \\ 1 & \infty & 0 & 0 & n\lambda_{\max} \\ 1 & n\lambda_{\max} & \infty & 0 & 0 \\ 1 & 0 & n\lambda_{\max} & \infty & 0 \\ 1 & 0 & 0 & n\lambda_{\max} & \infty \end{pmatrix}$$

Nach Konstruktion gilt für $i \in \mathcal{N} \setminus \{1\}$ $\text{SUM}_{i1} = n - 2$ und $\text{SUM}_{1i} = n\lambda_{\max}$. Somit gilt $\lambda_{1i} = \min\left\{\lambda_{\max}, \frac{n\lambda_{\max}}{n-2}\right\} = \lambda_{\max}$. Damit erhalten wir $t_{i1} = 1 > \frac{1}{\lambda_{\max}} \cdot (\lambda_{\max} - \epsilon)$, sodass Spieler 1 alle Jobs zugeteilt bekommt.

Insgesamt gilt hier $\text{SPAN} = (n-1) \cdot (\lambda_{\max} - \epsilon)$. Die optimale Zuteilung besteht zudem darin, dass Spieler 1 keinen Job bekommt. In allen anderen Fällen wird der Spieler mit den Jobkosten 0 bevorzugt. Insgesamt erhalten wir also $\text{OPT} = 1$.

Somit gilt $\lim_{\epsilon \to 0} \frac{\text{SPAN}}{\text{OPT}} = (n-1) \cdot \lambda_{\max}$ □

5.3 λ_3-Funktion

Bei der Analyse der λ_2-Funktion haben wir gesehen, dass bei der Bestimmung von λ_{ij} einzelne große Werte, welche keinen Einfluss auf den Makespan haben, dazu führen, dass SUM_{ij} oder SUM_{ji} viel zu groß werden. Dies führt für bei gewissen Instanzen dazu, dass stets der falsche Spieler bevorzugt wird. Das Ziel dieser λ-Funktion ist es also, dass wir eine noch größere Teilmenge an Instanzen finden können, bei welchen ein Approximationsfaktor echt kleiner als $n-1$ vorliegt. Insbesondere wollen wir für den Worst-Case-Fall aus der Analyse für die λ_2-Funktion einen Approximationsfaktor echt kleiner als $n-1$ erreichen. Der Algorithmus, um das durchzusetzen, ist wie folgt definiert:

Algorithmus 3:

Bestimmung von $\overline{\text{SUM}}_{ij}$:
1.1 Erstelle den Median m_{ij} aller Werte t_{kj} für $k \in \mathcal{N} \setminus \{i,j\}$.
1.2 Setze für $k \in \mathcal{N} \setminus \{i,j\}$, $\overline{t_{kj}} = \min\{t_{kj}, m_{ij}\}$.
1.3 Setze $\overline{\text{SUM}}_{ij} = \sum_{k|k\neq i,j} \overline{t_{kj}}$.

Bestimmung von $\overline{\text{SUM}}_{ji}$:
2.1 Erstelle den Median m_{ji} aller Werte t_{ki} für $k \in \mathcal{N} \setminus \{i,j\}$.
2.2 Setze für $k \in \mathcal{N} \setminus \{i,j\}$, $\overline{t_{ki}} = \min\{t_{ki}, m_{ji}\}$.
2.3 Setze $\overline{\text{SUM}}_{ji} = \sum_{k|k\neq i,j}^{n} \overline{t_{ki}}$.

3.1 Falls $\overline{\text{SUM}}_{ij} > \overline{\text{SUM}}_{ji}$, setze $\lambda_{ij} = \min\left\{\lambda_{\max}, \frac{\overline{\text{SUM}}_{ij}}{\overline{\text{SUM}}_{ji}}\right\}$.
3.2 Ansonsten setze $\lambda_{ij} = \max\left\{\lambda_{\min}, \frac{\overline{\text{SUM}}_{ij}}{\overline{\text{SUM}}_{ji}}\right\}$.

Beispiel 17. Wir wollen nun ein modifiziertes Beispiel 16 mit $\lambda_{\max} = 2$ und ohne ϵ-Werte betrachten. Wir möchten zeigen, dass wir mit Hilfe der λ_2-Funktion hier sogar eine optimale Zuteilung erreichen können. Sei

$$t = \begin{pmatrix} \infty & 2 & 2 & 2 & 2 \\ 1 & \infty & 0 & 0 & 10 \\ 1 & 10 & \infty & 0 & 0 \\ 1 & 0 & 10 & \infty & 0 \\ 1 & 0 & 0 & 10 & \infty \end{pmatrix}.$$

Unsere vorherige λ_2-Funktion hätte hier unter Umständen in vier Tiebreaks alle vier Jobs an Spieler 1 übergeben. Unsere jetzige λ-Funktion bestimmt λ und x wie folgt:

$$\lambda = \begin{pmatrix} \infty & 0.5 & 0.5 & 0.5 & 0.5 \\ 2 & \infty & 2 & 1 & 0.5 \\ 2 & 0.5 & \infty & 2 & 1 \\ 2 & 1 & 0.5 & \infty & 2 \\ 2 & 2 & 1 & 0.5 & \infty \end{pmatrix} \quad x = \begin{pmatrix} \infty & 0 & 0 & 0 & 0 \\ 1 & \infty & 1 & 1 & 0 \\ 1 & 0 & \infty & 1 & 1 \\ 1 & 0 & 0 & \infty & 1 \\ 1 & 1 & 0 & 0 & \infty \end{pmatrix}$$

Für $i \in \{2,3,4,5\}$ ergibt sich $\lambda_{i1} = 2$, da $\text{SUM}_{i1} = 3$ und $\text{SUM}_{1i} = 0$ stets erfüllt sind.

Für diese wie auch für die nachfolgenden λ-Funktionen wurden keine Testfälle mehr durchgeführt. Testfall 7 der λ_2-Funktion lieferte für den K_6 lediglich eine Instanz, bei welcher ein Approximationsfaktor von $n-1$ erreicht wurde. Mit besseren λ-Funktionen würden vermutlich wesentlich mehr Testfälle benötigt werden, um entsprechende Worst-Case-Instanzen

finden zu können. Zusätzlich steigt die Laufzeit, um ein einziges OPT zu bestimmen, exponentiell an. Wenn wir λ-Funktionen testen wollen, die nur für $n \geq 7$ funktionieren, so können wir nur eine geringe Ausbeute an berechneten Instanzen erwarten. Es zeigt uns daher, dass wir mit dem Testen hier an eine gewisse Grenze stoßen. Es gibt also keine Garantie dafür, dass man bei hinreichend vielen Tests tatsächlich auch die gewünschten Worst-Case-Fälle findet. Folglich haben wir uns an diesem Punkt dafür entschieden, die nachfolgenden λ-Funktion direkt zu analysieren.
Andererseits ist es uns durch theoretische Analyse gelungen Schranken für den Approximationsfaktor herzuleiten.

5.3.1 Analyse

Für die λ_3-Funktion wollen wir zeigen, dass auch hier ein Approximationsfaktor von mindestens $n-1$ gilt, sofern für $i, j \in \mathcal{N}$ mit $i \neq j$ $t_{ij} \in \mathbb{R}_{\geq 0}$.

Satz 13. *Sei $\lambda_{\max} > 1$ beliebig. Weiter seien für $i, j \in \mathcal{N}$ mit $i \neq j$ $t_{ij} \in \mathbb{R}_{\geq 0}$ beliebig. Werden alle λ_{ij}-Werte durch die λ_3-Funktion bestimmt, dann erhalten wir einen Approximationsfaktor von mindestens $n-1$.*

Beweis. Sei $n \geq 5$ beliebig, $\epsilon > 0$ hinreichend klein, dann definiere für $i, j \in \mathcal{N}$ mit $i \neq j$

$$t_{ij} = \begin{cases} \lambda_{\max}, \text{ falls } i = 2, j = 1 \\ 1, \text{ falls } i > 2, j = 1 \\ 1 - \epsilon, \text{ falls } i = 1, j = 2 \\ \lambda_{\min} - \epsilon, \text{ falls } i = 1, 2 < j \leq \lceil \frac{n}{2} \rceil \\ \lambda_{\max} - \epsilon, \text{ falls } i = 1, \lceil \frac{n}{2} \rceil < j \\ 2\lambda_{max}, \text{ falls } i < \lceil \frac{n}{2} \rceil, j > \lceil \frac{n}{2} \rceil \\ 0, \text{ sonst} \end{cases}$$

Die folgende Matrix zeigt diese Instanz für K_6

Beispiel 18.

$$t = \begin{pmatrix} \infty & 1-\epsilon & \lambda_{\min}-\epsilon & \lambda_{\min}-\epsilon & \lambda_{\max}-\epsilon & \lambda_{\max}-\epsilon \\ \lambda_{\max} & \infty & 0 & 0 & 2 \cdot \lambda_{\max} & 2 \cdot \lambda_{\max} \\ 1 & 0 & \infty & 0 & 2 \cdot \lambda_{\max} & 2 \cdot \lambda_{\max} \\ 1 & 0 & 0 & \infty & 0 & 0 \\ 1 & 0 & 0 & 0 & \infty & 0 \\ 1 & 0 & 0 & 0 & 0 & \infty \end{pmatrix}$$

Behauptung 4. *Spieler 1 erhält alle seine Jobs in der λ_3-Zuteilung.*

Beweis. Für $j \in \{2, \dots, \lceil \frac{n}{2} \rceil\}$ gehen die Jobs $\{1, j\}$ an Spieler 1, denn $t_{j1} > \lambda_{\max} \cdot t_{1j}$ ist hier nach Konstruktion stets erfüllt.
Für $j > \lceil \frac{n}{2} \rceil$ gilt einerseits $\overline{\text{SUM}}_{j1} = n - 2$. Das liegt daran, dass alle Werte in Spalte 1 als 1 gezählt werden.
Falls n gerade ist, so haben wir $\frac{n-2}{2}$ Joblaufzeiten, welche den Wert $2 \cdot \lambda_{\max}$ annehmen. Die andere Hälfte nimmt den Wert 0 an.
In diesem Fall ist $\overline{\text{SUM}}_{1j} = (n-2) \cdot \lambda_{\max}$.
Falls n ungerade ist, so nehmen $\frac{n-1}{2}$ Joblaufzeiten den Wert $2 \cdot \lambda_{\max}$ an. In jedem Fall gilt $\overline{\text{SUM}}_{1j} \geq (n-2) \cdot \lambda_{\max}$.
Dies führt dazu, dass $\lambda_{1j} = \min\left\{\lambda_{\max}, \frac{\lambda_{\max} \cdot (n-2)}{n-2}\right\} = \lambda_{\max}$ gesetzt wird. Nun aber ist auch hier $t_{j1} = 1 > \lambda_{\min} \cdot (\lambda_{\max} - \epsilon) = \lambda_{j1} \cdot t_{1j}$ erfüllt und Spieler 1 bekommt somit Job j zugeteilt. Insgesamt bekommt dieser also alle $n - 1$ Jobs zugeteilt. □

Nun möchten wir zeigen, dass $\text{OPT} = 1$ erfüllt ist. Wie man direkt sieht, besteht die optimale Zuteilung darin, dass Spieler 1 lediglich den Job $\{1, 2\}$ zugeteilt bekommt. Für $i, j \in \mathcal{N} \setminus \{1\}$ mit $i \neq j$ gibt es nun folgende Fälle:

Falls $t_{ij} = 2 \cdot \lambda_{\max}$, dann gilt nach Konstruktion $t_{ji} = 0$ und Spieler j bekommt den Job zugeteilt.
Ansonsten gilt nach Konstruktion $t_{ij} = 0$ und $t_{ji} = 0$ und hier ist es egal ob Spieler i oder j den Job $\{i, j\}$ zugeteilt bekommen. Insgesamt erhalten wir also $\text{OPT} = 1$.

Zusammengefasst gilt also $\text{SPAN} \geq (1 - \epsilon) + \frac{n-2}{2}(\lambda_{\min} - \epsilon) + \frac{n-2}{2}(\lambda_{\max} - \epsilon)$.
Daraus folgt $\frac{\text{SPAN}}{\text{OPT}} \geq 1 + \frac{n-2}{2}(\lambda_{\min}) + \frac{n-2}{2}(\lambda_{\max}) - (n-1) \cdot \epsilon \overset{\text{Lemma 3.}}{\geq} n - 1 - (n-1) \cdot \epsilon$. Wegen $\lim_{\epsilon \to 0}(n - 1 - (n-1) \cdot \epsilon) = n - 1$ erhalten wir einen Approximationsfaktor von mindestens $n - 1$. □

5.4 λ_4-Funktion

In den vorherigen zwei λ-Funktionen haben wir gesehen, dass sehr große Jobkosten, die keinen Einfluss auf den Makespan haben, dazu beigetragen haben, dass der unerwünschte Spieler den entsprechenden Job zugeteilt bekommen hat. Die λ_3-Funktion hatte das Ziel möglichst auszuschließen, dass solche Jobkosten Einfluss auf den Makespan nehmen. Jedoch haben wir auch hier gesehen, dass es immer noch Instanzen gibt, bei welchen die λ_3 Funktion eben solche Jobkosten nicht ignoriert. Nun wollen wir einen anderen Ansatz verwenden, beim welchem solche Fälle prinzipiell nicht auftreten können. Das liegt daran, dass wir nur die minimalen Joblaufzeiten über die Nachbarn einer Maschine betrachten. Der Algorithmus zu Bestimmung von λ_{ij} ist hierfür wie folgt:

Algorithmus 4:
1.1 Setze $\text{MIN}_{ij} = \min_{k|k\notin\{i,j\}}\{t_{kj}\}$.
1.2 Setze $\text{MIN}_{ji} = \min_{k|k\notin\{i,j\}}\{t_{ki}\}$.

2.1 Falls $\text{MIN}_{ij} > \text{MIN}_{ji}$, setze $\lambda_{ij} = \min\left\{\lambda_{\max}, \frac{\text{MIN}_{ij}}{\text{MIN}_{ji}}\right\}$.
2.2 Ansonsten setze $\lambda_{ij} = \max\left\{\lambda_{\min}, \frac{\text{MIN}_{ij}}{\text{MIN}_{ji}}\right\}$.

Beispiel 19. Nun wollen wir für den K_6 mit $\lambda_{\max} = 2$ untersuchen, wie die λ_4-Funktion mit Beispiel 18 umgeht. In diesem Fall sei ϵ hinreichend klein. Dann sei die Eingabe t zunächst:

$$t = \begin{pmatrix} \infty & 1-\epsilon & 0.5-\epsilon & 0.5-\epsilon & 2-\epsilon & 2-\epsilon \\ 2 & \infty & 0 & 0 & 4 & 4 \\ 1 & 0 & \infty & 0 & 4 & 4 \\ 1 & 0 & 0 & \infty & 0 & 0 \\ 1 & 0 & 0 & 0 & \infty & 0 \\ 1 & 0 & 0 & 0 & 0 & \infty \end{pmatrix}$$

Unsere λ_4-Funktion bestimmt λ und x wie folgt:

$$\lambda = \begin{pmatrix} \infty & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 \\ 2 & \infty & 1 & 1 & 1 & 1 \\ 2 & 1 & \infty & 1 & 1 & 1 \\ 2 & 1 & 1 & \infty & 1 & 1 \\ 2 & 1 & 1 & 1 & \infty & 1 \\ 2 & 1 & 1 & 1 & 1 & \infty \end{pmatrix} \quad x = \begin{pmatrix} \infty & 1 & 1 & 1 & 0 & 0 \\ 0 & \infty & 1 & 1 & 0 & 0 \\ 0 & 0 & \infty & 1 & 0 & 0 \\ 0 & 0 & 0 & \infty & 1 & 1 \\ 1 & 1 & 1 & 0 & \infty & 1 \\ 1 & 1 & 1 & 0 & 0 & \infty \end{pmatrix}$$

Für $j \in \{2,3,4,5,6\}$ gilt stets $\text{MIN}_{j1} = 1$ und $\text{MIN}_{1j} = 0$. Folglich gilt stets $\lambda_{1j} = 0.5$. Somit kann Spieler 1 nur drei Jobs zugeteilt bekommen. s.d. SPAN $<$ 2OPT erfüllt ist.

5.4.1 Analyse

Im Folgenden möchten wir zeigen, dass für die λ_4-Funktion, für $n \geq 5$, ein Approximationsfaktor $\alpha < n-1$ gilt, sofern die Jobkosten beliebige positive reelle Zahlen sind. Für den Beweis benötigen wir noch ein Lemma.

Definition 3. Für $j \in \mathcal{N}$ sei $count_j = |\{i \in \mathcal{N} | \text{MIN}_{ij} \leq \text{MIN}_{ji}, i \neq j\}|$.

Lemma 14. *Für $j \in \mathcal{N}$ sei $min\_val_j = \min\{t_{ij} | i \in \mathcal{N}, i \neq j\}$. Falls $count_j \geq 3$ und Spieler j in jeder optimalen Zuteilung höchstens einen Job zugeteilt bekommt, dann gilt $\text{OPT} \geq 2 \cdot min\_val_j$.*

Beweis. Es erhalte Spieler $j \in \mathcal{N}$ höchstens einen Job in jeder optimalen Zuteilung. Wegen $count_j \geq 3$ existieren $k_1, k_2, k_3 \in \mathcal{N} \setminus \{j\}$ paarweise verschieden, so dass $\text{MIN}_{kj} \leq \text{MIN}_{jk}$ gilt. Zudem muss $min\_val_j \leq \text{MIN}_{kj}$ nach Konstruktion erfüllt sein. Sei $A = \{t_{k_1k_2}, t_{k_1k_3}, t_{k_2k_1}, t_{k_3k_1}\}$. Das bedeutet insbesondere, dass $min\_val_j \leq \min(A)$ erfüllt sein muss.
Weiter wissen wir, dass folgende 2 Jobkosten in einer optimalen Zuteilung sein müssen. Entweder $t_{k_1k_2}$ oder $t_{k_2k_1}$ und entweder $t_{k_1k_3}$ oder $t_{k_3k_1}$. Dies führt zu folgenden zwei Fällen.

<u>**Fall 1**</u>: Spieler k_1 bekommt in einer optimalen Zuteilung den Job $\{k_1k_3\}$ und den Job $\{k_1k_2\}$ zugeteilt.

In diesem Fall wissen wir, dass $\text{OPT} \geq t_{k_1k_3} + t_{k_1k_2} \geq min\_val_j + min\_val_j \geq 2 \cdot min\_val_j$ erfüllt sein muss.

<u>**Fall 2**</u>: Spieler k_1 bekommt höchstens einen Job zugeteilt.

In diesem Fall bekommen genau zwei Spieler $i, l \in \{k_1, k_2, k_3\}$ mit $i \neq l$ einen Job mit Kosten mindestens so groß wie min_val_j zugeteilt. Nach Annahme bekommt Spieler j höchstens einen Job in einer optimalen Zuteilung. Somit muss Spieler i oder l bereits einen Job der Größe min_val_j besitzen. Damit bekommt mindestens ein Spieler zwei Jobs zugeteilt, welche mindestens so groß wie min_val_j sind. Damit gilt aber $\text{OPT} \geq 2 \cdot min\_val_j$. □

Satz 15. *Sei für $i, j \in \mathcal{N}$ mit $i \neq j$ $t_{ij} \in \mathbb{R}_{\geq 0}$ beliebig. Weiter sei $\lambda_{\max} = 1 + \frac{1}{4n}$. Werden die λ_{ij}-Werte mit Hilfe der λ_4- Funktion berechnet, so erhalten wir für $n \geq 5$ einen Approximationsfaktor echt kleiner als $n - 1$. Für $n \geq 11$ liegt dieser bei höchstens $n - 1.15$.*

Beweis. Für den Beweis wollen wir zunächst folgende Annahmen treffen.

- Wir nehmen o.B.d.A. an, dass der Makespan vom Spieler 1 bestimmt wird und der Algorithmus Spieler 1 alle Jobs zuteilt. Konkret gilt $\text{SPAN} = \sum_{i=2}^{n} t_{1i}$. Letzteres können wir annehmen, da wir im Zweifel für einen Job $\{1, j\}$ $t_{1j} = 0$ setzen können und den Job Spieler 1 zuteilen, ohne dabei den Makespan zu verändern. Insbesondere gilt dann für $i \in \mathcal{N} \setminus \{1\}$ mit Lemma 7 $t_{1i} \leq (1 + \frac{1}{4n}) \cdot \text{OPT}$.
- Weiter nehmen wir o.B.d.A. an, dass $t_{21} \geq t_{31} \geq .... \geq t_{n1}$ erfüllt ist.
- Zu guter Letzt nehmen wir an, dass $\text{OPT} > 0$ erfüllt ist. Ansonsten gilt zwangsweise $\text{SPAN} = 0$.

Für den Beweis gibt es folgende Fälle zu unterscheiden.

Fall 1: Spieler 1 bekommt in mindestens einer optimalen Zuteilung mindestens zwei Jobs zugewiesen.

Aufgrund unserer ersten Annahme wissen wir, dass für $i \in \mathcal{N} \setminus \{1\} \quad t_{1i} \leq \text{OPT} \cdot \left(1 + \frac{1}{4n}\right)$ erfüllt sein muss. Mindestens zwei verschiedene Jobs $\{1, l\}$ und $\{1, k\}$ sind zudem in einer optimalen Zuteilung. Somit gilt $t_{1l} + t_{1k} \leq \text{OPT}$.

Zusammengefasst gilt:

$$\begin{aligned} \text{SPAN} &= \sum_{i=2}^{n} t_{1i} \leq (n-3) \cdot \text{OPT} \cdot \left(1 + \frac{1}{4n}\right) + \text{OPT} \\ &= \left(n + \frac{1}{4} - 3 - \frac{3}{4n}\right) \cdot \text{OPT} + \text{OPT} \\ &\leq (n - 2.75) \cdot \text{OPT} + \text{OPT} \\ &= (n - 1.75) \cdot \text{OPT} \\ \Rightarrow \frac{\text{SPAN}}{\text{OPT}} &\leq n - 1.75 \end{aligned}$$

Falls ein Spieler also mindestens zwei Jobs in einer optimalen Zuteilung zugewiesen bekommt, so erhalten wir einen Approximationsfaktor von höchstens $n - 1.75$.
Es bleibt also noch folgender Fall:

Fall 2: Spieler 1 bekommt in jeder optimalen Zuteilung höchstens einen Job zugeteilt.

Hier werden wir folgende zwei Fälle unterscheiden:

Fall 2.1: Es gilt $t_{n1} \leq 0.6 \cdot \text{OPT}$.

Aufgrund unserer ersten Annahme wissen wir, dass für $i \in \mathcal{N} \setminus \{1\}$ $t_{1i} \leq \left(1 + \frac{1}{4n}\right) \cdot \text{OPT}$ gilt. Weiter wissen wir, dass $t_{1n} \leq 0.6\text{OPT} \cdot \left(1 + \frac{1}{4n}\right)$. Zusammengefasst gilt:

$$
\begin{aligned}
\text{SPAN} &= \sum_{i=2}^{n} t_{1i} \\
&\leq \left(((n-2)+0.6)\cdot\left(1+\frac{1}{4n}\right)\right)\cdot \text{OPT} \\
&= \left((n-1.4)\cdot\left(1+\frac{1}{4n}\right)\cdot \text{OPT}\right) \\
&= \left(n-1.15-\frac{0.35}{n}\right)\cdot \text{OPT} \\
&\leq (n-1.15)\cdot \text{OPT} \\
\Rightarrow \frac{\text{SPAN}}{\text{OPT}} &\leq n-1.15
\end{aligned}
$$

Fall 2.2: $t_{n1} > 0.6 \cdot \text{OPT}$
In diesem Fall gilt $count_1 \leq 2$. Ansonsten würde nach Lemma 14
$\text{OPT} > 1.2 \cdot \text{OPT}$ gelten. Das aber wäre für $\text{OPT} > 0$ ein Widerspruch.

Wir wollen noch untersuchen, wie groß die Jobkosten von Spieler 1 unter diesen Bedingungen werden können. Dafür betrachten wir zwei Fälle.

Fall 2.2.1 Spieler 1 bekommt in jeder optimalen Zuteilung keinen Job zugeteilt.

Dann wissen wir, dass alle Spieler $l \in \{2, \ldots, n\}$ in jeder optimalen Zuteilung den Job t_{l1} zugeteilt bekommen mit $t_{l1} > 0.6\text{OPT}$. Andererseits gilt auch $\text{OPT} \geq t_{l1}$, da diese Jobs sich in einer optimalen Zuteilung befinden.
Angenommen für $i, k \in \mathcal{N} \setminus \{1\}$ mit $i \neq k$ gilt $\text{MIN}_{1i} \geq 0.4\text{OPT}$ und $\text{MIN}_{1k} \geq 0.4\text{OPT}$. Dann gilt nach Definition von MIN_{1i} und MIN_{1k}, dass $t_{ik} \geq \text{MIN}_{1k} \geq 0.4\text{OPT}$ und dass $t_{ki} \geq \text{MIN}_{1i} \geq 0.4\text{OPT}$ sein muss. In diesem Fall muss einer der beiden Jobs in einer optimalen Zuteilung sein. Beide Spieler besitzen jedoch einen Job, welcher größer als 0.6OPT ist.
Dies würde dazu führen dass $\text{OPT} > 0.6\text{OPT} + 0.4\text{OPT} = \text{OPT}$, was jedoch zu einem Widerspruch führt. Folglich kann höchstens für ein $i \in \mathcal{N} \setminus \{1\}$ $\text{MIN}_{1i} \geq 0.4\text{OPT}$ gelten. Da $\lambda_{1i} \leq 1 + \frac{1}{4n}$, gilt $t_{1i} \leq \left(1+\frac{1}{4n}\right) \cdot \text{OPT}$. Für alle anderen $j \in \mathcal{N} \setminus \{1, i\}$ gilt somit $\text{MIN}_{1j} < 0.4\text{OPT}$.
Für $i \neq j$ gilt $\lambda_{1j} = \max\left(\frac{0.4\text{OPT}}{0.6\text{OPT}}, \frac{1}{1+\frac{1}{4n}}\right) = \frac{1}{1+\frac{1}{4n}}$. Entsprechend gilt $t_{1j} \leq \text{OPT} \cdot \frac{1}{1+\frac{1}{4n}}$.
Insgesamt nehmen hier also $n-2$ λ-Werte den Wert $\frac{1}{1+\frac{1}{4n}}$ an und höchstens ein Eintrag nimmt einen Wert bis zu $1+\frac{1}{4n}$ an.

Fall 2.2.2 Es gibt optimale Zuteilungen, in denen Spieler 1 genau einen Job zugeteilt bekommt.

Sei für $i \in \mathcal{N} \setminus \{1\}$ t_{1i} in einer optimalen Zuteilung. Dann gilt $\text{OPT} \geq t_{1i}$. Wir wissen also, dass alle anderen Spieler $j \in \mathcal{N} \setminus \{i, 1\}$ in einer optimalen Zuteilung Job t_{j1} mit $\text{OPT} \geq t_{j1} > 0.6\text{OPT}$ zugeteilt bekommen. Angenommen es gibt $j, k \in \mathcal{N} \setminus \{i, 1\}$ mit $j \neq k$, s.d. $\text{MIN}_{1j} \geq 0.4\text{OPT}$ und $\text{MIN}_{1k} \geq 0.4\text{OPT}$ erfüllt ist, dann gilt $t_{jk} \geq \text{MIN}_{1k} \geq 0.4\text{OPT}$ und $t_{kj} \geq \text{MIN}_{1j} \geq 0.4\text{OPT}$. Wie auch im **Fall 2.2.1** können wir nun folgern, dass höchstens einer der Werte $t_{jk}, t_{kj} \geq 0.4\text{OPT}$ sein kann. Der entsprechende λ-Wert ist dann höchstens $1 + \frac{1}{4n}$. Entsprechend gilt analog zum **Fall 2.2.1** für $l \in \mathcal{N} \setminus \{1, i, j, k\}$ $\lambda_{1l} = \frac{1}{1 + \frac{1}{4n}}$. Zusätzlich gilt $\lambda_{1j} = \frac{1}{1 + \frac{1}{4n}}$ oder $\lambda_{1k} = \frac{1}{1 + \frac{1}{4n}}$.
Insgesamt erhalten wir für Spieler 1 mindestens $n - 3$ λ-Werte, welche den Wert $\frac{1}{1 + \frac{1}{4n}}$ annehmen. Höchstens zwei Werte können einen Wert $\geq \text{OPT}$ annehmen, wobei einer dieser Werte sich in einer optimalen Zuteilung befindet und höchstens den Wert OPT annehmen kann.

Für **Fall 2.2.1** und **Fall 2.2.2** gilt nun

$$\text{SPAN} = \sum_{i=2}^{n} t_{1i} \leq \text{OPT} \cdot \left(1 + \frac{1}{4n}\right) + \text{OPT} + \frac{n-3}{\left(1 + \frac{1}{4n}\right)} \cdot \text{OPT}$$
$$\Rightarrow \frac{\text{SPAN}}{\text{OPT}} \leq 2 + \frac{1}{4n} + \frac{n-3}{1 + \frac{1}{4n}}$$

Mittels Einsetzen von Werten erhalten wir:
Für $n = 5$ gilt $2 + \frac{1}{4n} + \frac{n-3}{1 + \frac{1}{4n}} \approx 3.955$
Für $n = 11$ gilt $2 + \frac{1}{4n} + \frac{n-3}{1 + \frac{1}{4n}} \approx 9.84$
Sei $f(x) = 2 + \frac{1}{4x} + \frac{x}{1 + \frac{1}{4x}} : x \in \mathbb{R}_{\geq 5}$
Dann gilt $f'(x) = \frac{16x^2 + 8x - 12}{16x^2 + 8x + 1} - \frac{1}{4x^2} \leq \frac{16x^2 + 8x - 12}{16x^2 + 8x + 1}$ für $x \geq 5$
Nun aber ist $n \in \mathbb{N}_{\geq 5}$ $1 > \frac{16n^2 + 8n - 12}{16n^2 + 8n + 1}$. Somit gilt $f(n+1) < f(n) + 1$, da $f(5) < 4$ entspricht, also $f(n) < n - 1$ für alle $n \geq 5$. Insbesondere erhalten wir für $n \geq 11$ $f(n) < n - 1.15$.

Aus dem **Fall 1** erhalten wir einen Approximationsfaktor von von $n - 1.75$. Während aus dem **Fall 2.1** sich eine Approximationsfaktor von $n - 1.15$ ergibt. Zusammengefasst ist unser Approximationsfaktor über alle Instanzen
$\alpha = \max\left\{n - 1.15, 2 + \frac{1}{4n} + \frac{n-3}{1 + \frac{1}{4n}}\right\}$, wobei dieser für $n \geq 11$ stets bei $n - 1.15$ ist. □

Satz 16. *Sei $\lambda_{\max} > 1$ beliebig. Seien für alle $i, j \in \mathcal{N}$ mit $i \neq j$ $t_{ij} \in \mathbb{R}_{\geq 0}$ beliebig. Weiter werden die Werte λ_{ij} mit Hilfe der λ_4-Funktion bestimmt. Dann ist der Approximationsfaktor für alle $n > 4$ über alle möglichen Instanzen stets größer als $n - 2$.*

Beweis. Hierfür betrachten wir folgende Eingabe:

$$t_{ij} = \begin{cases} \lambda_{\max} - \epsilon, \text{falls} \quad i = 1 \\ 1, \text{falls} \quad j = 1, i \neq n \\ 0, \text{falls} \quad j = 1, i = n \\ \epsilon, \text{sonst} \end{cases} \qquad t = \begin{pmatrix} \infty & \lambda_{\max} - \epsilon & \dots & \dots & \dots & \lambda_{\max} - \epsilon \\ 1 & \infty & \epsilon & \dots & \dots & \epsilon \\ \dots & \epsilon & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 1 & \dots & \dots & \dots & \dots & \epsilon \\ 0 & \epsilon & \dots & \dots & \epsilon & \infty \end{pmatrix}$$

Die optimale Zuteilung, mit $\epsilon > 0$ hinreichend klein, besteht darin, dass Spieler 1 keinen Job zugeteilt bekommt. Mit allen anderen Jobs kommt man so auf $\text{OPT} = 1 + (n-2) \cdot \epsilon$. Für $i \in \mathcal{N} \setminus \{1\}$ gilt $\text{MIN}_{1i} = \epsilon$ und $\text{MIN}_{i1} = 0$. Dies führt dazu, dass $\lambda_{1i} = \lambda_{\max}$ gesetzt wird. So erhält Spieler 1 nach Konstruktion $n-2$ Jobs zugeteilt. Zudem bestimmt Spieler 1 auch den Makespan SPAN.

Insgesamt gilt $\text{SPAN} = (n-2) \cdot (\lambda_{\max} - \epsilon)$.

$$\Rightarrow \lim_{\epsilon \to 0} \quad \frac{\text{SPAN}}{\text{OPT}} = (n-2) \cdot \lambda_{\max} > n - 2. \qquad \square$$

Insbesondere erhalten wir, dass für $\lambda_{\max} \geq 1 + \frac{1}{n-2}$ der Approximationsfaktor $\geq n - 1$ ist, denn dann gilt $\frac{\text{SPAN}}{\text{OPT}} = (n-2) \cdot \lambda_{\max} = (n-2) \cdot \left(1 + \frac{1}{n-2}\right) = (n-2) + 1 = n - 1$. Daraus folgt:

Lemma 17. *Sei $\lambda_{\max} > 1 + \frac{1}{n-2}$ beliebig. Seien für alle $i, j \in \mathcal{N}$ t_{ij} mit $i \neq j \in \mathbb{R}_{\geq 0}$ beliebig. Weiter werden die Werte λ_{ij} mit Hilfe der λ_4-Funktion bestimmt. Dann ist der Approximationsfaktor für alle $n > 4$ mindestens $n - 1$.*

Dies zeigt uns nun, dass wir nun zwar eine λ-Funktion gefunden haben, welche uns einen Approximationsfaktor liefert, welches echt kleiner als $n - 1$ ist. Dennoch wollen wir nun noch weitere λ-Funktionen untersuchen und überprüfen, ob es unter Umständen möglich ist, zumindest für sehr große $n \in \mathbb{N}$ einen Approximationsfaktor $\leq n - 2$ zu erreichen.

5.5 λ_{5_k}-Funktion

Wir haben bei der λ_2-Funktion zur Bestimmung von SUM_{ij} jeweils die Summe aus $n-2$ Elementen der Spalte j ohne Zeile i und j einfließen lassen. Bei der λ_3-Funktion sind wir bei

der Bestimmung von $\overline{\text{SUM}}_{ij}$ ähnlich vorgegangen. Hierbei haben wir jedoch alle Werte welche größer als der Median sind auf den Median abgerundet. Bei der λ_4-Funktion haben wir bei der Bestimmung von MIN_{ij} ohne es zu erwähnen auch die entsprechende Summe betrachtet, in der jeder Wert der Summe auf das Minimum der entsprechenden Spalte abgerundet wurde. Bei der λ_2-Funktion flossen also in der entsprechenden Summe die $\lceil \frac{n-2}{1} \rceil = n-2$ kleinsten Werte der entsprechenden Spalte ein, während bei der λ_3-Funktion die $\lfloor \frac{n-2}{2} \rfloor$ kleinsten Werte miteinbezogen wurden. Bei der λ_4-Funktion wurden hingegen die $\lceil \frac{n-2}{n-2} \rceil = 1$ kleinsten Werte berücksichtigt. Wir haben uns hier gefragt, ob sich dieses Konzept für ein $k \geq 1$ mit $\lceil \frac{n-2}{k} \rceil$ verallgemeinern lässt und ob wir unter Umständen auf diese Weise für mindestens ein $k \geq 3$ einen Approximationsfaktor unter $n-2$ angeben können. Entsprechend wollen wir im Folgenden eine Familie von λ-Funktionen mit folgendem Algorithmus betrachten:

Algorithmus 5:

Bestimmung von $\widehat{\text{SUM}}_{ij}$:
1.1 Erstelle ein Array A aller Werte t_{lj} für $l \in \mathcal{N} \setminus \{i,j\}$.
1.2 Sortiere A aufsteigend.
1.3 Setze $\text{VAL}_{ij} = A[\lfloor \frac{n-2}{k} \rfloor]$.
1.4 Setze für $l \in \mathcal{N} \setminus \{i,j\} \quad \widehat{t}_{lj} = \min\{\text{VAL}_{ij}, t_{lj}\}$.
1.5 Setze $\widehat{\text{SUM}}_{ij} = \sum_{l \mid l \notin \{i,j\}} \widehat{t}_{lj}$.

Bestimmung von $\widehat{\text{SUM}}_{ji}$:
2.1 Erstelle ein Array A aller Werte t_{li} für $l \in \mathcal{N} \setminus \{i,j\}$.
2.2 Sortiere A aufsteigend.
2.3 Setze $\text{VAL}_{ji} = A[\lfloor \frac{n-2}{k} \rfloor]$.
2.4 Setze für $l \in \mathcal{N} \setminus \{i,j\} \quad \widehat{t}_{li} = \min\{\text{VAL}_{ji}, t_{li}\}$.
2.5 Setze $\widehat{\text{SUM}}_{ij} = \sum_{l \mid l \notin \{i,j\}} \widehat{t}_{li}$.
3.1 Falls $\widehat{\text{SUM}}_{ij} > \widehat{\text{SUM}}_{ji}$, setze $\lambda_{ij} = \min\left\{\lambda_{\max}, \frac{\widehat{\text{SUM}}_{ij}}{\widehat{\text{SUM}}_{ji}}\right\}$.
3.2 Ansonsten setze $\lambda_{ij} = \max\left\{\lambda_{\min}, \frac{\widehat{\text{SUM}}_{ij}}{\widehat{\text{SUM}}_{ji}}\right\}$.

5.5.1 Analyse

Wir möchten nun zeigen, dass es für alle $k \geq 5$ und alle $r \in \mathbb{R}_{\geq 0}$ ein n_0 gibt, sodass wir für alle $n \geq n_0$ mit Hilfe der λ_{5_k}-Funktion einen Approximationsfaktor von höchstens $n-r$ über alle möglichen Instanzen erhalten. Um das zu zeigen, brauchen wir noch etwas Vorbereitung.

Definition 4. Sei $\text{ZEROS}_j = |\{i \in \mathcal{N} \setminus \{j\} | t_{ij} = 0\}|$.

Lemma 18. *Angenommen es seien nur die Jobkosten 0 oder a mit $a > 0$ erlaubt. Weiter sei für ein $k_0 \in \mathbb{N}$, und alle $j \in \mathcal{N}$ $\text{ZEROS}_j = k_0$. Falls das Graph-Scheduling-Problem auf dem Graphen K_n $\text{OPT} = 0$ ergibt, dann ist $n \leq 2k_0 + 1$.*

Beweis. Für jeden Job $\{i, j\}$ muss mindestens einer der beiden Jobkosten t_{ij} oder t_{ji} die Kosten 0 tragen. Das bedeutet wiederum, dass mindestens die Hälfte aller Jobkosten 0 betragen soll. Damit haben wir insgesamt bis zu $2 \cdot k_0$ Jobkosten t_{ij} mit $i \neq j$ in jeder Spalte. Somit gilt $n \leq 2k_0 + 1$

□

Unter gegebenen Bedingungen haben hier lediglich gezeigt, dass K_{2k_0+1} der größtmögliche Graph ist auf dem so was unter Umständen gelten könnte. Wie genau eine solche Konstruktion aussieht, sofern diese existiert, darum werden wir uns hier nicht kümmern. Wir werden dies jedoch später als obere Schranke verwenden, um unseren Approximationsfaktor zu maximieren.
Weiter möchten wir eine weitere Eigenschaft bezüglich unserer λ-Werte zeigen.

Lemma 19. *Sei $\lambda \geq 1$, $k_0 \in \mathbb{N}_{\geq 3}$ beliebig, dann gilt $2 \cdot \lambda + k_0 \cdot \frac{1}{\lambda} < 2 + k_0$ genau dann wenn $1 < \lambda < \frac{k_0}{2}$ erfüllt ist.*

Beweis. Die Bedingung

$$2 \cdot \lambda + k_0 \cdot \frac{1}{\lambda} \quad < \quad k_0 + 2$$

ist äquivalent zur Folgenden:

$$\begin{aligned}
2 \cdot \lambda + k_0 \cdot \frac{1}{\lambda} - (k_0 + 2) &< 0 \\
2 \cdot \lambda^2 + k_0 - (k_0 + 2) \cdot \lambda &< 0 \\
\lambda^2 + \frac{k_0}{2} - \frac{(k_0 + 2)}{2} \cdot \lambda &< 0 \quad //\text{pq-Formel} \\
\lambda_1 &> \frac{2 + k_0}{4} - \sqrt{\left(\frac{2 + k_0}{4}\right)^2 - \frac{k_0}{2}} \\
&= \frac{2 + k_0}{4} - \frac{k_0 - 2}{4} = 1 \\
\lambda_2 &< \frac{2 + k_0}{4} + \frac{k_0 - 2}{4} = \frac{k_0}{2} \\
&\Rightarrow \quad 1 < \lambda < \frac{k_0}{2}.
\end{aligned}$$

□

Satz 20. *Seien die Jobkosten beliebige positive reelle Zahlen, dann gilt, dass es für alle* $k \geq 5$, *mit* $k = \mathcal{O}(1)$, $\lambda_{\max} = 1 + \frac{1}{4k}$ *und alle* $r \in \mathbb{R}_{\geq 0}$ *ein* $n_0 \in \mathbb{N}$ *gibt, s.d. wir für alle* $n \geq n_0$ *einen Approximationsfaktor von höchstens* $n - r$ *erhalten.*

Beweis. Für den Beweis treffen wir, wie auch in der Analyse der λ_4-Funktion, folgende Annahmen:

- Wir nehmen o.B.d.A. an, dass der Makespan vom Spieler 1 bestimmt wird und der Algorithmus Spieler 1 alle Jobs zuteilt. Konkret gilt $\textsc{span} = \sum_{i=2}^{n} t_{1n}$. Letzteres können wir annehmen, da für Jobs $\{1, j\}$ die nicht Maschine 1 zugeteilt werden wir o.B.d.A. $t_{1j} = 0$ setzen und annehmen, dass $\{1, j\}$ Spieler 1 zugeteilt wird.
- Weiter nehmen wir o.B.d.A. an, dass $t_{21} \geq t_{31} \geq \ldots \geq t_{n1}$ erfüllt ist.
- Zu guter Letzt nehmen wir an, dass $\textsc{opt} > 0$ erfüllt ist. Ansonsten gilt zwangsweise $\textsc{span} = 0$.

Für den ersten Teil des Beweises wollen wir zunächst einfachheitshalber annehmen, dass k ein Teiler von $n - 2$ ist. Konkret sei $n - 2 = (k \cdot l)$.

Wir unterscheiden folgende Fälle:

Fall 1.1: Sei $z = (k-1) \cdot l + 2$, $t_{z1} < 0.6\text{OPT}$. Dann gilt:

$$\begin{aligned}
\text{SPAN} &= \sum_{i=2}^{n} t_{1i} = \sum_{i=2}^{(k-1)\cdot l+2} t_{1i} + \sum_{(k-1)\cdot l+3}^{k\cdot l+2} t_{1i} \\
&\leq \left(1 + \frac{1}{4k}\right) \cdot (((k-1) \cdot l + 1) + 0.6 \cdot l) \cdot \text{OPT} \\
&= \left(1 + \frac{1}{4k}\right) \cdot ((k - 0.4) \cdot l + 1) \cdot \text{OPT} \\
&= \left(1 + \frac{1}{4k}\right) \cdot (kl - 0.4l + 1) \cdot \text{OPT} \\
&= \left(kl - 0.4l + 1 + \frac{kl - 0.4l + 1}{4k}\right) \cdot \text{OPT} \\
&= \left(kl - 0.4l + 1 + \frac{l}{4} - \frac{l}{10k} + \frac{1}{4k}\right) \cdot \text{OPT} \\
&= \left(n - 1 - 0.15l - \frac{l}{10k} + \frac{1}{4k}\right) \cdot \text{OPT} \\
&\leq \left(n - 1 - 0.15l + \frac{1}{4k}\right) \cdot \text{OPT} \\
&\leq \left(n - 1 - 0.15l + \frac{1}{20}\right) \cdot \text{OPT} \\
&= \left(n - \frac{19}{20} - 0.15l\right) \cdot \text{OPT} \\
\Rightarrow \frac{\text{SPAN}}{\text{OPT}} &\leq n - \frac{19}{20} - 0.15l \\
&= \mathcal{O}(1) + n - \Theta(n).
\end{aligned}$$

Für den nächsten Fall sei $n - 2 = k \cdot l + m$ mit $1 \leq m \leq k - 1$ und k ist kein Teiler von $n - 2$ und $z = ((k-1) \cdot l + 2 + m)$.

Fall 1.2: $t_{z1} < 0.6\text{OPT}$ und k ist kein Teiler von $n - 2$.

Dann wissen wir, dass die zusätzlichen m Elemente $\leq \left(1 + \frac{1}{4k}\right) \cdot \text{OPT}$ sind. Die l Elemente der zweiten Summe aus dem **Fall 1.1** bleiben unverändert. Dann gilt:

$$
\begin{aligned}
\text{SPAN} &\leq \left((n-m) - \frac{19}{20} - 0.15l + m \cdot \left(1 + \frac{1}{4k}\right)\right) \cdot \text{OPT} \\
&\leq \left(n - \frac{19}{20} - 0.15l + \frac{m}{4k}\right) \cdot \text{OPT} \\
&\leq \left(n - \frac{19}{20} - 0.15l + \frac{k-1}{4k}\right) \cdot \text{OPT} \\
&\leq \left(n - \frac{19}{20} - 0.15l + \frac{5}{20}\right) \cdot \text{OPT} \\
&\leq \left(n - \frac{14}{20} - 0.15l\right) \cdot \text{OPT} \\
\Rightarrow \frac{\text{SPAN}}{\text{OPT}} &\leq n - \frac{14}{20} - 0.15l \\
&= \mathcal{O}(1) + n - \Theta(n).
\end{aligned}
$$

Für den nächsten Fall wollen wir nicht mehr zu unterscheiden, ob $n-2$ ungerade oder gerade ist, daher sei von nun an $n-2 = k \cdot l + m$ mit $0 \leq m \leq k-1$ mit $z = (k-1) \cdot l + 2 + m$.

Fall 2: $t_{z1} \geq 0.6\text{OPT}$
Wir wollen hier nochmals $\widehat{\text{SUM}}_{j1}$ aus dem Algorithmus betrachten. Es gilt einerseits $l = \frac{n-2-m}{k} \leq \frac{n-2-m}{5} \leq \frac{n-1}{5}$ für $k \geq 5$. Insgesamt erhalten wir:

$$
\begin{aligned}
\widehat{\text{SUM}}_{j1} &= \sum_{i|i \notin \{1,j\}} \widehat{t}_{i1} \\
&= \sum_{i=2, l \neq j}^{(k-1)\cdot l+2+m} \widehat{t}_{i1} + \sum_{i=(k-1)\cdot l+3+m, i \neq j}^{k \cdot l+2+m} \widehat{t}_{i1} \\
&\geq \sum_{i=2, i \neq j}^{(k-1)\cdot l+2+m} \widehat{t}_{i1} \\
&\geq 0.6 \cdot \text{OPT} \cdot (n-1-l) \\
&\overset{k \geq 5}{\geq} 0.6 \cdot \text{OPT} \cdot (n-1-\frac{n-1}{5}) \\
&= 0.6 \cdot \text{OPT} \cdot (0.8 \cdot (n-1)) \\
&= 0.48 \cdot \text{OPT} \cdot (n-1).
\end{aligned}
$$

Definition 5. Sei $\text{VALS}_j = |\{i \in \mathcal{N} \setminus \{1,j\} | t_{ij} > 0.4\text{OPT}\}|$.

Es lassen sich Folgendes beobachten:

- falls $\widehat{\text{SUM}}_{1j} \leq 0.4 \cdot (n-1) \cdot \text{OPT}$, dann gilt wegen $\widehat{\text{SUM}}_{j1} \geq 0.48 \cdot \text{OPT} \cdot (n-1)$ und $\frac{0.4}{0.48} < \frac{1}{1+\frac{1}{4k}}$ für $k \geq 5$ $\lambda_{1j} = \lambda_{\min}$.

- falls hingegen $\widehat{\text{SUM}}_{1j} \geq 0.4(n-1)\text{OPT}$, so gibt es Fälle, bei welchen $\lambda_{1j} = \lambda_{\max}$ möglich ist.
- $\widehat{\text{SUM}}_{1j} \geq 0.4(n-1)\text{OPT}$ ist nur dann möglich, falls $\text{VALS}_j = (k-1) \cdot l + 1 + m$ erfüllt ist. Letzteres folgt, da aus den $n-2$ betrachteten Werten stets der $l-1$ kleinste Wert $\geq 0.4\text{OPT}$ sein muss.

Im Folgenden möchten wir uns einige Behauptungen anschauen.

Behauptung 5. *Falls* $\widehat{\text{SUM}}_{1j} \leq 0.4\text{OPT}$, $\widehat{\text{SUM}}_{j1} \geq 0.48$, *so wird der Approximationsfaktor über alle Instanzen dadurch maximiert, wenn für* $i \in \mathcal{N} \setminus \{1, j\} \quad t_{ij} = 0$ *erfüllt ist.*

Beweis. Nach den obigen Beobachtungen gilt in diesem Fall $\lambda_{1j} = \lambda_{\min}$. Damit kann Spieler 1 lediglich den Job $\{1, j\}$ zugeteilt bekommen, falls $t_{j1} \geq \lambda_{\max} \cdot t_{1j}$ erfüllt ist. Falls $t_{ij} \neq 0$ ist, kann sich der mögliche Approximationsfaktor nur verringern, denn falls $t_{ji} \neq 0$ entspricht, so erhöhen sich in einer optimalen Zuteilung die Gesamtkosten von Spieler i oder Spieler j. Das würde dann zur Folge haben, dass OPT größer wird. □

Wir wollen zunächst hier einige Annahmen treffen falls $\widehat{\text{SUM}}_{j1} \geq 0.48$ erfüllt ist:

- Falls $\text{VALS}_j = (k-1) \cdot l + 1 + m$, dann wissen wir, dass es Fälle gibt wo $\widehat{\text{SUM}}_{1j} \geq \widehat{\text{SUM}}_{j1}$ möglich ist. Dann schätzen wir λ_{1j} auf $\lambda_{\max}$ ab. Zudem schätzen wir t_{1j} auf $\lambda_{\max} \cdot \text{OPT}$ ab. Alle anderen t_{ij} Werte schätzen wir auf 0 ab, s.d. $\text{VALS}_j = (k-1) \cdot l + 1 + m$ erfüllt bleibt und OPT minimal gehalten wird.
- Falls $\text{VALS}_j < (k-1) \cdot l + 1 + m$, dann setzen wir für $i \in \mathcal{N} \setminus \{j, 1\}$ nach Behauptung 5 $t_{ij} = 0$. Falls $t_{j1} \leq \text{OPT}$, dann wollen wir entsprechend t_{1j} auf $\frac{\text{OPT}}{\lambda_{\max}}$ abschätzen.
- Falls $t_{1i} < \frac{\text{OPT}}{\lambda_{\max}}$, dann setzen wir $t_{i1} = \text{OPT}$ und $t_{1i} = \frac{\text{OPT}}{\lambda_{\max}}$ und $t_{ij} = 0$ für $i \in \mathcal{N} \setminus \{j, 1\}$. Letzteres führt dazu, dass $\widehat{\text{SUM}}_{i1}$ für $i \in \mathcal{N} \setminus \{1, j\}$ größer werden kann. Wir werden jedoch später sehen, dass wir statt VALS_j $\widehat{\text{VALS}}_j = |\{i \in \mathcal{N} \setminus \{1, j\} | t_{ij} > 2\text{OPT}\}|$ betrachten können um den Approximationsfaktor zu erhöhen. Letzteres garantiert uns wegen $\widehat{\text{SUM}}_{1j} \leq (n-1) \cdot \text{OPT}$, dass für $\widehat{\text{VALS}}_j = (k-1) \cdot l + 1 + m \ \frac{\widehat{\text{SUM}}_{1j}}{\widehat{\text{SUM}}_{j1}} > 1 + \frac{1}{4k}$ erfüllt ist. Damit würde λ_{1j} direkt auf $\lambda_{\max}$ gesetzt werden.

Behauptung 6. *Der potenzielle Approximationsfaktor, mit* $\widehat{\text{SUM}}_{j1} \geq 0.48$, *über alle Instanzen wird maximiert, wenn für jeden Spieler* $i \in \mathcal{N} \setminus \{1\} \quad t_{j1} \geq 0.6\text{OPT}$ *erfüllt ist.*

Beweis. Falls es ein $t_{j1} < 0.6\text{OPT}$ gibt, dann gilt
$t_{1j} \leq 0.6 \cdot \lambda_{\max} \cdot \text{OPT} = 0.6 \cdot (1 + \frac{1}{4k}) \cdot \text{OPT} \overset{k \geq 5}{\leq} 0.6 \cdot (1 + \frac{1}{20}) \cdot \text{OPT} = 0.63 \cdot \text{OPT} < \frac{\text{OPT}}{\lambda_{\max}}$.
Wenn hingegen $t_{j1} = \text{OPT}, t_{1j} = \frac{\text{OPT}}{\lambda_{\max}}$, dann können wir für $l \in \mathcal{N} \setminus \{1, j\}$ $t_{lj} = 0$ setzen und so mit Behauptung 5 und unseren Annahmen einen größeren Approximationsfaktor erreichen. □

Im Folgenden werden wir daher annehmen, dass $t_{j1} \geq 0.6 \cdot \text{OPT}$ erfüllt ist.

Behauptung 7. *Der potenzielle Approximationsfaktor, mit* $\widehat{\text{SUM}}_{j1} \geq 0.48$*, über alle Instanzen wird maximiert, wenn Spieler 1 höchstens einen Job aus der optimalen Zuteilung zugeteilt bekommt.*

Beweis. Sei $\text{OPT}_1 = \{i \in \mathcal{N} \setminus \{1\} | t_{1i} \text{ ist in einer optimalen Zuteilung}\}$ mit $|\text{OPT}_1| \geq 2$, dann gilt $\sum_{i \in \text{OPT}_1} t_{1i} \leq \text{OPT} < 2 \cdot \frac{\text{OPT}}{\lambda_{\max}}$. Falls also $i \in \text{OPT}_1$, dann erhalten wir durch Behauptung 5 und unseren Annahmen einen höheren Approximationsfaktor, indem wir $t_{i1} = \text{OPT}$ und $t_{1i} = \frac{\text{OPT}}{\lambda_{\max}}$ und zusätzlich für $s \in \mathcal{N} \setminus \{1, j\}$ $t_{sj} = 0$ setzen. □

Wir können also für den weiteren Verlauf annehmen, dass Spieler 1 höchstens einen Job in einer optimalen Zuteilung erhält. Wir möchten nun zwei Fälle unterscheiden.

Fall 2.1: Spieler 1 bekommt in keiner optimalen Zuteilung einen Job zugewiesen.

Wir möchten nun unsere Annahmen anwenden. Falls $\text{VALS}_j = (k-1) \cdot l + 1 + m$, so schätzen wir λ_{1j} auf $\lambda_{\max}$ ab. Für $i \in \mathcal{N} \setminus \{1, j\}$ muss also für $l - 1$ Einträge $t_{ij} = 0$ erfüllt sein. Wir möchten die Anzahl der Spalten, für die $\text{VALS}_j = (k-1) \cdot l + 1 + m$ erfüllt ist, maximieren. Falls für $i, j \in \mathcal{N} \setminus \{1\}$ mit $i \neq j$ $t_{ij} > 0.4\text{OPT}$ und $t_{ji} > 0.4\text{OPT}$, so bekommt einer der beiden Spieler einen zusätzlichen Job $> 0.4\text{OPT}$ zugeordnet s.d. $\text{OPT} > 0.6\text{OPT} + 0.4\text{OPT} = \text{OPT}$ gilt, was jedoch ein Widerspruch ist.
Sei $\text{COLS} = \{j \in \mathcal{N} \setminus \{1\} | \text{VALS}_j = (k-1) \cdot l + 1 + m\}$ beliebig. Dann wissen wir, dass für Spieler $i \in \text{COLS}, j \in \text{COLS} \setminus \{i\}$ Jobkosten t_{ij} existieren. Es findet also für ein $n' = |\text{COLS}|$ ein Graph-Scheduling auf dem $K_{n'}$ statt. Nach Lemma 18 wissen wir, dass $n' \leq 2 \cdot (l-1) + 1$ erfüllt sein muss, sodass $\text{OPT}' = 0.$ für den $K_{n'}$ erfüllt wird. Wir nehmen an, dass $n' = 2 \cdot (l-1) + 1$ möglich ist. Diese Wahl kann nach Konstruktion den Approximationsfaktor nur erhöhen. Dann können wir wie folgt ein solches $K_{n'}$ konstruieren:

Konstruktion: Für $j \in \{2, \ldots, 2 \cdot (l-1) + 2\}$ setze $\text{ZEROS}_j = l - 1$. Unser $K_{n'}$ erstreckt sich o.B.d.A. über alle Jobs $J = \{\{i, j\} \in E | n' + 1 \geq i > 1, n' + 1 \geq j > 1, i \neq j\}$. Das können wir annehmen, da wir für $1 < j < n' + 1$ t_{1j} auf $\lambda_{\max} \cdot \text{OPT}$ abschätzen wollen. Zudem setzen wir für $1 < j_0 < 2 \cdot (l-1) + 2$ und $i_0 > 2 \cdot (l-1) + 2$ $t_{i_0 j_0}$ auf einen Eintrag $> 0.4\text{OPT}$. s.d. letztlich $\text{VALS}_j = (k-1) \cdot l + 1 + m$ erfüllt ist.
Nun wissen wir, dass falls $t_{i_0 j_0} > 0.4\text{OPT}$, dann nach unserer Annahme $t_{j_0 i_0} = 0$ gelten muss. Damit aber muss $zeros_{j_0} \geq 2 \cdot (l-1) + 1$ erfüllt sein. Zusammengefasst gibt es also bis zu $2 \cdot (l-1) + 1$ Jobgrößen, welche für Spieler 1 den Wert $\lambda_{\max} \cdot \text{OPT}$ erreichen können. In allen anderen Fällen ist $\lambda_{1j} = \lambda_{\min}$. Somit gibt es $n - 1 - (2 \cdot (l-1) + 1)$ Jobs mit Jobgrößen, welche für Spieler 1 ein Wert von $\frac{\text{OPT}}{\lambda_{\max}}$ haben.

Fall 2.2: Es gibt optimale Zuteilungen, in denen Spieler 1 einen Job zugewiesen bekommt.

Analog wie im **Fall 2.1** versuchen wir COLS zu maximieren. Sei n' entsprechend analog. Wir wissen, dass falls $j \notin \text{COLS}$ dann $\text{ZEROS}_j \geq 2 \cdot (l-1) + 1$. Wenn also $\{1, j\}$ in einer optimalen Zuteilung ist, so kann Spieler j bis zu zwei Jobs $> 0.4\text{OPT}$ zugeteilt bekommen. In diesem Fall würde für höchstens $j_1, j_2 \notin \text{COLS}$ $\text{ZEROS}_{j_1} \geq 2 \cdot (l-1)$ und $\text{ZEROS}_{j_2} \geq 2 \cdot (l-1)$ gelten. Wir könnten also zwei Jobkosten 0 durch einen Wert $> 0.4\text{OPT}$ ersetzen. Wegen $2 \cdot (l-1) > l-1$ für $l > 1$, bleibt λ_{1j_1} und λ_{1j_2} bei $\lambda_{\min}$. Zusammengefasst erhalten wir also für Spieler 1 einen Job mit Jobkosten bis zu OPT, $2 \cdot (l-1) + 1$ Jobs mit Jobkosten bis zu $\lambda_{\max} \cdot \text{OPT}$ und $n - 1 - (2 \cdot (l-1) + 2)$ Jobs mit Jobgrößen, welche für Spieler 1 einen Wert bis zu $\frac{\text{OPT}}{\lambda_{\max}}$ haben.
Es zeigt sich auch hier, dass wir für $j \in \mathcal{N} \setminus \{1\}$ statt VALS_j $\widehat{\text{VALS}}_j$ betrachten können. Der Grund liegt letztlich darin, dass wir durch die 2 Jobkosten $> 0.4\text{OPT}$ nicht erreichen können, dass COLS erhöht wird. Folglich können wir alternativ beliebig hohe Jobkosten wie $> 2\text{OPT}$ betrachten. Insgesamt gilt also für Fall 2.1 und Fall 2.2:

$$
\begin{aligned}
\frac{\text{SPAN}}{\text{OPT}} &\leq 1 + (2 \cdot (l-1) + 1) \cdot \left(1 + \frac{1}{4k}\right) + (n - 1 - (2 \cdot (l-1) + 2)) \cdot \frac{1}{1 + \frac{1}{4k}} \\
&= 1 + (2 \cdot l - 1) \cdot \left(1 + \frac{1}{4k}\right) + (kl + m + 1 - 2 \cdot l) \cdot \frac{1}{1 + \frac{1}{4k}} \\
&= 1 + (2 \cdot l - 1) \cdot \left(1 + \frac{1}{4k}\right) + ((k-2)l + m + 1) \cdot \frac{1}{1 + \frac{1}{4k}} \\
&\leq 2 + \frac{1}{4k} + m + 1 + 2l \cdot \left(1 + \frac{1}{4k}\right) + (k-2)l \cdot \frac{1}{1 + \frac{1}{4k}} \\
&= 2 + \frac{1}{4k} + m + 1 + l \cdot \left(2 \cdot \left(1 + \frac{1}{4k}\right) + (k-2) \cdot \frac{1}{1 + \frac{1}{4k}}\right) \\
&\overset{\text{Lemma 19}}{=} 2 + \frac{1}{4k} + m + 1 + l \cdot (k - \epsilon) \quad \text{für ein } \epsilon > 0 \\
&\leq 2 + \frac{1}{4k} + m + 1 + (n-2) - l \cdot (k - \epsilon) \\
&= \mathcal{O}(1) + n - l\epsilon \\
&= \mathcal{O}(1) + n - \Theta(n).
\end{aligned}
$$

In allen hier betrachteten Fällen ist der Approximationsfaktor der Form $\mathcal{O}(1) + n - \Theta(n)$. Dann aber gilt für alle $r \in \mathbb{R}_{\geq 0}$, dass es ein n_0 geben muss, s.d. wir für alle $n \geq n_0$ einen Approximationsfaktor von höchstens $n - r$ erhalten. Damit ist die Behauptung bestätigt.
□

6 Zusammenfassung

In dieser Arbeit haben wir ehrliche λ-Mechanismen für das Graph-Scheduling-Problem auf dem K_n mit dem Ziel entwickelt, über alle möglichen Instanzen einen Approximationsfaktor

unter $n-1$ für alle $n \geq n_0$ angeben zu können. Wir haben die λ_1- und λ_2-Funktionen getestet und analysiert, wodurch wir feststellen konnten, dass wir mit λ-Mechanismen, welche die λ_1- oder die λ_2-Funktionen zur Bestimmung der λ_{ij}-Werte verwenden, über alle Instanzen bei beliebigen Jobkosten einen Approximationsfaktor von exakt $(n-1) \cdot \lambda_{\max}$ erhalten.
Anschließend haben wir noch drei weitere λ-Funktionen analysiert. Für die λ_3-Funktion konnten wir feststellen, dass über alle möglichen Instanzen ein Approximationsfaktor von mindestens $n-1$ vorliegen muss.
Mit der λ_4-Funktion haben wir es letztlich geschafft einen Widerspruch zur Behauptung zur Behauptung aus der Einleitung anzugeben. Wir haben es hier geschafft für $n \geq 5$ einen Approximationsfaktor unter $n-1$ anzugeben und konnten hier auch zeigen, dass der Approximationsfaktor über alle Instanzen mindestens $n-2$ ist.
Zum Schluss haben wir uns gefragt, ob ein Approximationsfaktor von weniger als $n-2$ möglich ist. Hier haben wir mit λ_{5_k}, eine Familie von λ-Funktionen, betrachtet. Mit dieser konnten wir für $k \geq 5$ zeigen, dass für alle $r \in \mathbb{R}_{\geq 0}$ es ein $n \geq n_0$ gibt, s.d. wir für alle $n \geq n_0$ einen Approximationsfaktor von höchstens $n-r$ erhalten.

References

[1] Sushil Bikhchandani et al. "Weak Monotonicity Characterizes Deterministic Dominant-Strategy Implementation". In: *Econometrica* 74.4 (2006), pp. 1109–1132. ISSN: 00129682, 14680262. URL: http://www.jstor.org/stable/3805917.

[2] George Christodoulou, Elias Koutsoupias, and Annamária Kovács. "Truthful allocation in graphs and hypergraphs". In: *CoRR* abs/2106.03724 (2021), pp. 1–5. arXiv: 2106.03724. URL: https://arxiv.org/abs/2106.03724.

[3] John Hunter et al. *Grouped bar chart with labels*. https://matplotlib.org/stable/gallery/lines_bars_and_markers/barchart.html. [Online; accessed 18-February-2022].

[4] Noam Nisan and Amir Ronen. "Algorithmic Mechanism Design". In: *Games and Economic Behavior* 35.1 (2001), pp. 166–196. ISSN: 0899-8256. DOI: https://doi.org/10.1006/game.1999.0790. URL: https://www.sciencedirect.com/science/article/pii/S089982569990790X.

[5] Michael Saks and Lan Yu. "Weak Monotonicity Suffices for Truthfulness on Convex Domains". In: EC '05 (2005), pp. 286–293. DOI: 10.1145/1064009.1064040. URL: https://doi.org/10.1145/1064009.1064040.

7 Anhang

Unter folgenden Link befindet sich einerseits der Quellcode zu den Histogrammen als py-Code und der Code für die hier eingesetzten Algorithmen als C++-Code

Link: https://github.com/boris602/Bachelor-Graph-Scheduling