Implementierung eines Job shop Scheduling Problems in Xpress Mosel

Daniel Brice Kamga Nana, Eric Dietriche Sesso Domtchoueng
Technische Universität Clausthal
Institut für Wirtschaftswissenschaft

January 24, 2023

1 Beschreibung der Job Shop Scheduling Problem (JSSP)

Betrachten wir eine Werkstattfertigung mit N Aufträgen und M Maschinen. Jeder Auftrag muss eine Reihe von Operationen durchlaufen und jede Operation ist nur einer bestimmten Maschine zugeordnet. Jeder Auftrag muss eine Reihe von Maschinen in einer bestimmten Reihenfolge durchlaufen. Diese Reihenfolge ist im Voraus festgelegt und kann während der Ausführung nicht geändert werden. Zusätzlich zu den vorherigen Bedingungen hat jede Operation eine Bearbeitungszeit, die natürlich von der Maschine und dem jeweiligen Auftrag abhängt. Schließlich ist die Jobsfolge auf jeder Maschine nicht im Voraus festgelegt, sondern soll durch beispielsweise heuristische Verfahren so bestimmt werden, dass eine vordefinierte Zielfunktion minimiert wird. Eine Instanz von JSSP kann also wie folgt definiert werden:

- Eine Menge von M Maschinen, die N Aufträge bearbeiten müssen
- Eine Menge von Operationen Ω.
- Jede Operation $o_{ij} \in \Omega(\ 1 \leq i \leq M$ und $1 \leq j \leq N)$ hat eine vordefinierte Bearbeitungsdauer p_{ij}.
- Jeder Job hat eine Maschinenfolge, die sich von einem Job zum anderen unterscheiden kann.
- Die Zielfunktion ist die maximale Zykluszeit.

2 JSSP mit 2 Maschinen und 3 Jobs

Hier werden wir uns mit einem Spezialfall des JSSP mit 2 Maschinen und 3 Aufträgen beschäftigen. Zunächst erstellen wir eine Instanz und zeigen, wie man daraus eine zulässige Lösung unter Berücksichtigung der verschiedenen Nebenbedingungen der beiden Formulierungen (siehe Abschnitt 2.2) erstellt.

2.1 Formulierung einer Instanz

In der Tabelle 1 haben wir o_{ij} als binäre variable dargestellt. Wenn $o_{ij} = 1$ dann geht den Job j durch die Maschine i und wenn $o_{ij} = 0$, dann geht nicht den Job j durch die Maschine i.

Job j	Maschine 1	Maschine 2
1	1	1
2	1	1
3	1	1

Table 1: Operationen o_{ij}

Job j	Maschine 1	Maschine 2
1	10	5
2	4	8
3	6	4

Table 2: Bearbeitungszeit p_{ij}

Job j		
1	Maschine 1 $\longrightarrow$	Maschine 2
2	Maschine 1 $\longrightarrow$	Maschine 2
3	Maschine 2 $\longrightarrow$	Maschine 1

Table 3: Maschinenfolge

Aus der Tabelle 3 können wir die Matrix σ darstellen, wobei der Eintrag σ_{ij} = k wenn $o_{kj} \in \Omega$ nächste Operation ist,sonst 0 (da $o_{ij} \in \Omega$ letzte Operation von Job j ist).

$$\sigma = \begin{bmatrix} 2 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Unser formuliertes Instanz lässt sich auch als disjunktive Graph wie folgt darstellen :

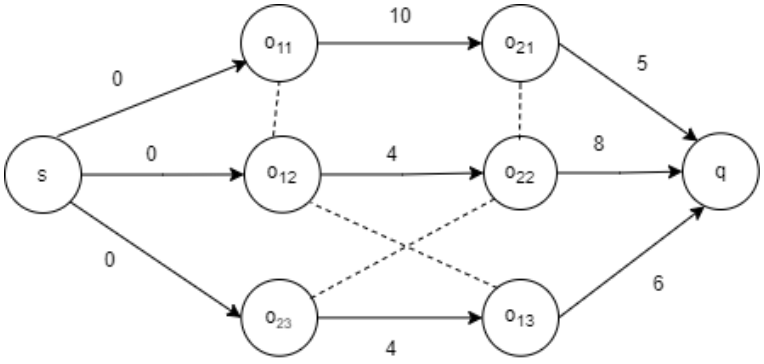

Abbildung 1: Disjunktive Graphendarstellung für ein JSSP mit 2 Maschinen

Die gestrichelten Linien zeigen die konkurrierenden Operationen an, die von einer Maschine ausgeführt werden müssen. Ziel ist es, eine Folge von Operationen so aufeinander abzustimmen, dass jede Maschine ihre komplette Bearbeitungszeit minimiert. d.h. in unserem disjunktiven Diagramm wird für jeden gestrichelten Pfeil eine Richtung festgelegt, so dass jede Maschine ihre komplette Bearbeitungszeit minimiert.

2.2 Beweis der Zulässigkeit der generierten Lösung (Schedule)

In diesem Abschnitt stellen wir zunächst 2 mathematische Formulierungen für die Lösung des JSSP vor. Dann zeigen wir anhand unserer generierten JSSP-Instanz mit 2 Maschinen und 3 Aufträgen, wie die Nebenbedingungen einen zulässigen Schedule fördern.

1. Mathematische Formulierung vom JSSP

S_{ij}= Startzeitpunkt der Operation $o_{ij} \in \Omega$
w_{ijk}= 1 wenn Operation o_{ik} vor o_{ij} ausgeführt wird ,sonst 0.

$$
\begin{aligned}
&min \quad && \max_{o_{ij}\in\Omega}(S_{ij}+p_{ij}) && && (1.1)\\
&u.d.N \quad && S_{ij}+p_{ij} \le S_{\sigma(i),j} && \forall o_{ij}\in\Omega \wedge \sigma(i)\neq 0 && (1.2)\\
& && S_{ij}+p_{ij} \le S_{i,k}+Mw_{ijk} && \forall\{o_{ij},o_{ik}\}\subseteq\Omega \wedge j\neq k && (1.3)\\
& && S_{ik}+p_{ik} \le S_{i,j}+M(1-w_{ijk}) && \forall\{o_{ij},o_{ik}\}\subseteq\Omega \wedge j\neq k && (1.4)\\
& && S_{ij}+p_{ij} \le C_{max} && \forall j \quad und \quad i \quad mit \quad \sigma(i)=0 && (1.5)\\
& && S_{ij} \ge 0 && \forall o_{ij}\in\Omega && (1.6)\\
& && w_{ijk}\in\{0,1\} && \forall\{o_{ij},o_{ik}\}\subseteq\Omega \wedge j\neq k && (1.7)
\end{aligned}
$$

- (1.1) steht für die maximale Zykluszeit der Maschinen, die minimiert werden soll.
- (1.2) zwei Maschinen, die einen Job nacheinander ausführen müssen, dürfen nicht gleichzeitig starten.
- (1.3) M sorgt dafür, dass wenn Job k vor Job j auf der Maschine i eingeplant wird, die Operation o_{ik} so früh wie möglich gestartet werden soll.
- (1.4) Wenn Job k vor Job j auf Maschine i eingeplant wird, muss die Operation o_{ij} gestartet werden, nachdem die Operation o_{ik} abgeschlossen ist.
- (1.5) wenn i die letzte Maschine ist, die den Job j bearbeitet, dann darf die Abschlusszeit der Operation o_{ij} die maximale Zykluszeit nicht überschreiten.
- (1.6) Jede Operation muss mindestens zum Zeitpunkt 0 starten

2. Mathematische Formulierung mit TSP-Variablen vom JSSP

d_{ijk}= 1 wenn Job j direkt nach Job k auf der Maschine i bearbeitet wird, sonst 0.
Sei $\Omega^{'} = \Omega \cup \{o_{i0} | 1 \leq i \leq M\}$, wobei 0 ein fiktiver Job ist.
S_{ij}= Startzeitpunkt der Operation $o_{ij} \in \Omega$

$$min \qquad \max_{o_{ij} \in \Omega}(S_{ij} + p_{ij}) \tag{2.1}$$

$$u.d.N \qquad S_{ij} + p_{ij} \leq S_{\sigma(i),j} \qquad \forall o_{ij} \in \Omega \wedge \sigma(i) \neq 0 \tag{2.2}$$

$$\sum_{\forall o_{ij} \in \Omega} d_{ijk} + d_{i0k} = 1 \qquad \forall o_{ik} \in \Omega \wedge j \neq k \tag{2.3}$$

$$\sum_{\forall o_{ik} \in \Omega} d_{ijk} + d_{ij0} = 1 \qquad \forall o_{ij} \in \Omega \wedge j \neq k \tag{2.4}$$

$$\sum_{\forall o_{ij} \in \Omega} d_{ij0} = 1 \tag{2.5}$$

$$\sum_{\forall o_{ik} \in \Omega} d_{i0k} = 1 \tag{2.6}$$

$$S_{ij} + p_{ij} \leq S_{i,k} + M(1 - d_{ijk}) \qquad \forall \{o_{ij}, o_{ik}\} \subseteq \Omega \wedge j \neq k \tag{2.7}$$

$$S_{ij} + p_{ij} \leq C_{max} \qquad \forall j \quad und \quad i \quad mit \quad \sigma(i) = 0 \tag{2.8}$$

$$S_{ij} \geq 0 \qquad \forall o_{ij} \in \Omega \tag{2.9}$$

$$d_{ijk} \in \{0, 1\} \qquad \forall \{o_{ij}, o_{ik}\} \subseteq \Omega^{'} \tag{2.10}$$

Nebenbedingungen (2.2) , (2.7) , (2.8) und (2.9) entsprechen die Nenbebingungen (1.3)-(1.7). Zudem haben Die zwei Formulierungen die gleiche Zielfunktion.

- (2.3) Jede Operation o_{ik} ist genau der Nachfolger einer Operation o_{ij} für jede Maschinen-Rundreise i. Außerdem ist einer der Operationen o_{ik} auf Maschine i genau der Nachfolger von dem fiktiven Job 0.
- (2.4) Jede Operation o_{ij} ist genau der Vorgänger einer Operation o_{ik} für jede Maschinen-Rundreise i.Außerdem ist einer der Operationen o_{ij} auf Maschine i genau der Vorgänger von dem fiktiven Job 0.
- (2.5) der Job 0 hat genau einen Vorgänger in jeder Maschinen-Rundreise i.
- (2.6) der Job 0 hat genau einen Nachfolger in jeder Maschinen-Rundreise i.

Aus der Formulierung 1 des JSSP lassen sich die folgenden Einschränkungen beispielweise auf unsere Instanz ableiten:

(1.2):

$$\begin{aligned}
&\mathrm{S}_{11} + 10 \leq S_{21} \\
&S_{12} + 4 \leq S_{22} \\
&S_{23} + 4 \leq S_{13}
\end{aligned}$$

(1.3) :

$$\begin{aligned}
&S_{12} + 4 \leq S_{11} + M * w_{121} \\
&S_{13} + 6 \leq S_{12} + M * w_{132} \\
&S_{13} + 6 \leq S_{11} + M * w_{131} \\
&S_{21} + 5 \leq S_{23} + M * w_{213} \\
&S_{22} + 8 \leq S_{23} + M * w_{223} \\
&S_{22} + 8 \leq S_{21} + M * w_{221}
\end{aligned}$$

(1.4):

$$\begin{aligned}
&\mathrm{S}_{11} + 10 \leq S_{12} + M * (1 - w_{121}) \\
&\mathrm{S}_{12} + 4 \leq S_{13} + M * (1 - w_{132}) \\
&\mathrm{S}_{11} + 10 \leq S_{13} + M * (1 - w_{131}) \\
&\mathrm{S}_{23} + 4 \leq S_{21} + M * (1 - w_{213}) \\
&\mathrm{S}_{23} + 4 \leq S_{22} + M * (1 - w_{223}) \\
&\mathrm{S}_{21} + 5 \leq S_{22} + M * (1 - w_{221})
\end{aligned}$$

(1.5):

$$\begin{aligned}
&\mathrm{S}_{21} + 10 \leq \mathrm{C}_{max} \\
&\mathrm{S}_{22} + 8 \leq \mathrm{C}_{max} \\
&\mathrm{S}_{13} + 6 \leq \mathrm{C}_{max}
\end{aligned}$$

(1.6):

$$\mathrm{S}_{11}, S_{12}.S_{21}, S_{22}, S_{13}, S_{23} \geq 0$$

(1.7):

$$w_{112}, w_{132}, w_{131}, w_{213}, w_{223}, w_{221} = 1$$

Nebendingung (1.2) fordert , dass die Operationen o_{11}, o_{12}, o_{23} jeweils vor der Operationen o_{21}, o_{22}, o_{13} gestartet werden ,damit können die Jobs 1, 2 und 3 nicht parallel auf die zwei Maschinen ausgeführt werden.Zudem sicherstellt die Nebenbedingungen (1.3) , (1.4) und (1.7) , dass

- die Operationen o_{11}, o_{12}, o_{13} auf Maschine 1 in der Reihenfolge $o_{11} - o_{12} - o_{13}$ bearbeitet werden sollen,wobei jede Operation erst gestartet werden soll , wenn eine vorläufige Operation abgeschlossen ist
- die Operationen o_{21}, o_{22}, o_{23} auf Maschine 2 in der Reihenfolge $o_{23} - o_{21} - o_{22}$ bearbeitet werden sollen,wobei jede Operation erst gestartet werden soll , wenn eine vorläufige Operation abgeschlossen ist

Das oben beschriebene Schedule ist zulässig.

Aus der Formulierung 2 des JSSP lassen sich die folgenden Einschränkungen beispielweise auf unsere Instanz ableiten:

(2.2):

$\mathrm{S}_{11} + 10 \leq S_{21}$
$S_{12} + 4 \leq S_{22}$
$S_{23} + 4 \leq S_{13}$

(2.3) :

$d_{121} + d_{131} + d_{101} = 1$
$d_{112} + d_{132} + d_{102} = 1$
$d_{113} + d_{123} + d_{103} = 1$
$d_{221} + d_{231} + d_{201} = 1$
$d_{212} + d_{232} + d_{202} = 1$
$d_{213} + d_{223} + d_{203} = 1$

(2.4) :

$d_{112} + d_{113} + d_{110} = 1$
$d_{121} + d_{123} + d_{120} = 1$
$d_{131} + d_{132} + d_{130} = 1$
$d_{212} + d_{213} + d_{210} = 1$
$d_{221} + d_{223} + d_{220} = 1$
$d_{231} + d_{232} + d_{230} = 1$

(2.5) :

$d_{110} + d_{120} + d_{130} = 1$
$d_{210} + d_{220} + d_{230} = 1$

(2.6):

$\mathrm{d}_{201} + d_{202} + d_{203} = 1$
$\mathrm{d}_{101} + d_{102} + d_{103} = 1$

(2.7):

$\mathrm{S}_{12} + 4 \leq S_{11} + M * (1 - d_{121})$
$\mathrm{S}_{13} + 6 \leq S_{12} + M * (1 - d_{132})$
$\mathrm{S}_{23} + 4 \leq S_{21} + M * 1 - (d_{231})$
$\mathrm{S}_{21} + 5 \leq S_{22} + M * (1 - d_{212})$

(2.8):

$\mathrm{S}_{21} + 10 \leq \mathrm{C}_{max}$
$\mathrm{S}_{22} + 8 \leq \mathrm{C}_{max}$
$\mathrm{S}_{13} + 6 \leq \mathrm{C}_{max}$

(2.9):

$\mathrm{S}_{11}, S_{12}.S_{21}, S_{22}, S_{13}, S_{23} \geq 0$

(2.10):

$d_{121}, d_{132}, d_{103}, d_{110} = 1$
$\mathrm{d}_{203}, d_{231}, d_{212}, d_{220} = 1$

Nebendingung (2.1) fordert , dass die Operationen o_{11}, o_{12}, o_{23} jeweils vor der Operationen o_{21}, o_{22}, o_{13} gestartet werden ,damit können die Jobs 1, 2 und 3 nicht parallel auf die zwei Maschinen ausgeführt werden.Zudem sicherstellt die Nebenbedingungen (2.3) , (2.4) , (2.5) , (2.6) und (2.10) dass

- die Operationen o_{11}, o_{12}, o_{13} auf Maschine 1 in der Reihenfolge $o_{13} - o_{12} - o_{11}$ bearbeitet werden sollen,wobei jede Operation erst gestartet werden soll , wenn eine vorläufige Operation abgeschlossen ist
- die Operationen o_{21}, o_{22}, o_{23} auf Maschine 2 in der Reihenfolge $o_{23} - o_{21} - o_{22}$ bearbeitet werden sollen,wobei jede Operation erst gestartet werden soll , wenn eine vorläufige Operation abgeschlossen ist

Das oben beschriebene Schedule ist zulässig.

3 2-Maschinen-JSSP mit Jobs, die mehrere Operationen pro Maschine umfassen können, ist NP-vollständig

Für den Beweis der NP-Vollständigkeit von JSSP mit 2 Maschinen $J_2||C_{max}$ werden wir die "Reduktionsmethode" verwenden, die darin besteht zu zeigen, dass ein bekanntes NP-komplettes Problem X reduziert oder in unser JSSP transformiert werden kann. Das bekannte NP-komplette Problem, das wir verwenden werden, ist das 3-Teilungs-Problem: Das Problem besteht darin, zu entscheiden, ob eine gegebene Menge von ganzen Zahlen in Triplets geteilt werden kann, die alle die gleiche Summe haben. d.h. Wir wollen für eine gegebene ganze Zahl B bestimmen, ob es möglich ist, eine Menge A von n ganzen Zahlen in drei disjunkte Teilmengen A_1, A_2, A_3 zu unterteilen, so dass $\sum_{a\in A_1} a = \sum_{b\in A_2} b = \sum_{c\in A_3} c = B$.
Darüber hinaus soll für eine gegebene Instanz x des 3-Teilungs-problems mit polynomieller Zeit eine Instanz y von $J_2||C_{max}$ so konstruiert werden, dass y genau dann mit "ja" antwortet (alle Aufträge werden bis zur Zeit C_{max} bearbeitet), wenn x mit "ja" antwortet.

Beweis: Seien n und B zwei positive ganzzahlige Zahlen(n,B $\in \mathbf{N}$). Die Menge A $= \{a_1, a_2, \ldots, a_{3n}\}$, wobei $\frac{B}{4} < a_i < \frac{B}{2}$ und $\sum_{i=1}^{3n} a_i = nB$ für $1 \leq i \leq 3n$.Diese formulierte 3-Teilungs-problem Instanz kann eine Job-Shop Instanz wie folgt konstruiert werden. Die Menge J=$\{J_0, ..., J_{3n}\}$ von Jobs ist so definiert, dass J_0 hat 2n Operationen und die Jobs J_i für $1 \leq i \leq 3n$ haben jeweils 2 Operationen. Anschließend müssen wir die Maschinenreihenfolge für jeden Job und die entsprechende Bearbeitungszeit für jede Operation festlegen. Wir können den folgenden Regel verwenden :
Sei m ein Funktion ,die jede Operation $J_i[j]$, $1 \leq i \leq 3n$ und $1 \leq j \leq 2n$ an Maschinen M $\in \{M_1, M_2\}$ einplant.
Sei τ ein Funktion ,die jede Operation $J_i[M_j]$, $1 \leq i \leq 3n$ und $1 \leq j \leq 2$ einer positiven ganzzahligen Zahl zuweist.

$$m(J_0[j]) = \begin{cases} M_1 : 0 \leq j \leq 2n & \text{und j gerade} \\ M_2 : 1 \leq j \leq 2n-1 & \text{und j ungerade} \end{cases} \tag{1}$$

Die Maschinenfolge von J_0 sieht so aus :$M_1 - M_2 - M_1 - ... - M_1$

$$m(J_i[1]) = M_1, \quad 1 \leq i \leq 3n$$
$$m(J_i[2]) = M_2, \quad 1 \leq i \leq 3n$$

Die Maschinenfolge von J_i für $1 \leq i \leq 3n$ sehen so aus :$M_1 - M_2$

$$\tau(J_0[j]) = B, \quad 1 \leq j \leq 2n$$

Jede Operation von Job J_0 dauert B Zeiteinheit

$$\tau(J_i[1]) = 0, \quad 1 \leq i \leq 3n$$

Die Bearbeitungszeit jeder Job J_i auf Maschine M_1 beträgt 0 Zeiteinheit.

$$\tau(J_i[2]) = a_i, \quad 1 \leq i \leq 3n$$

Die Bearbeitungszeit jeder Job J_i auf Maschine M_2 beträgt a_i Zeiteinheit.

Dabei müssen alle Operation spätesten zu C_{max} = 2nB abgeschlossen werden. Wir wollen zeigen, dass die gewünschte Partition von A existiert nur dann, wenn es einen Schedule S für Jobs J mit Länge (S) $\leq C_{max}$ gibt.
Angenommen, wir wollen nur die Operationen $J_0[1], ..., J_o[2n]$ einplanen. Setzen wir sie

in der Reihenfolge $J_0[1], ..., J_o[2n]$ ein, so ergibt sich nach jeder Operation an Maschine 2 und an Maschine 1 eine Zeitlücke der Länge B. Die Zeitlücke für diesen Zeitplan ist durch die Fertigstellungszeit von Maschine 2 gegeben: nB+ nB = 2nB =C_{max}. Fragestellung: Können wir die verbleibenden 3n Aufträge einplanen, ohne die Zeitlücke zu vergrößern? Da die Bearbeitungszeit dieser 3n Aufträge auf Maschine 1 gleich 0 ist, können wir diese Aufträge ganz am Anfang des Schedules einplanen, ohne die Zeitspanne zu vergrößern. Ebenso können wir diese 3n Aufträge am Ende des Zeitplans auf der zweiten Maschine einplanen, ohne die Zeitspanne zu verändern, weil es Auf Maschine 2 noch Leerlaufzeiten der Länge B gibt. Wenn es also eine Möglichkeit gibt, diese Aufträge in diese Leerlaufzeiten aufzuteilen, dann gibt es eine n-Partition, so dass jede Menge von 3 Aufträgen eine Bearbeitungszeit B hat. Da die Summe dieser 3n Bearbeitungszeiten nB ist, muss jede Menge eine Gesamtbearbeitungszeit B haben. Wenn es eine Lösung für das 3-Teilungs-Problem gibt, dann gibt es eine n-Teilung der Zahlen in 3-Element-Untermengen, so dass die Summe jeder Menge gleich B ist. Diese 3-Element-Untermengen passen in die Leerstellen in unserem Schedule.

4 Beschreibung der Implementierung der mathematischen Formulierungen in Xpress

Die beiden mathematischen Formulierungen des JSSPs sind abhängig von dem gewählten Wert von Big-M. Darüber hinaus müssten wir auch die Matrix Sigma bestimmen, da diese in Probleminstanz nicht vorgegeben wurde. In diesem Teil der Arbeit werden wir eine grobe Beschreibung geben , wie wir die Big-M , die Matrix Sigma und die Nebenbedingungen der mathematischen Formulierungen in Xpress umgesetzt haben.

4.1 Aufbau von Sigma

Sigma ist eine Matrix, wo jede Spalte bzw. Zeile einen Job bzw. eine Spalte darstellt. Die Einträge geben die Nachfolge Maschine von einem Job j ($1 \leq j \leq N$) auf einer Maschine i ($1 \leq j \leq M$). In unserer Implementierung haben wir diese Matrix als mehrdimensionales dynamisches Array deklariert. So am Anfang wurde jeder Eintrag zu 0 initialisiert. Dann haben wir anhand der Matrixreihenfolge (diese Matrix wurde im Probleminstanz gegeben) die Einträge mit Nummern der Maschinen geändert.Dabei haben wir geachtet , dass $\sigma(i, j) =$ k , wenn $o_{ik} \in \Omega$, anderfalls $\sigma(i, j)$=-1.Somit bleiben 0 alle nicht geänderte Einträge und diese entsprechen die letzte Maschinen , die die Jobs ausführen.

4.2 Aufbau von Big-M

M ist insofern eine wichtige Variable unseres Programms, als sie korrekt gewählt werden sollte. Andernfalls wird die Implementierung nicht funktionieren. M ist in unserem Programm definiert durch das Produkt aus der Summe der maximalen Bearbeitungszeit pro Maschine und der Anzahl der Aufträge plus 1.

$$M = \#Jobs \cdot \sum_{1 \leq i \leq M} max\{p_{ij} | 1 \leq j \leq N\} \quad + 1$$

4.3 Umsetzung der zwei Formulierungen in Xpress

Dieser Abschnitt stellt den entscheidenden Teil unseres Programms dar. Wie in der Aufgabenstellung gesagt, sollen in diesem Programm zwei Formulierungen umgesetzt werden.
Die Zielfunktion könnte nicht direkt umgesetzt werden. Dafür haben wir eine zusätzliche Nebenbedingung mit der Entscheidungsvariable C_{max} eingeführt, nämlich C_{max} muss die größte Durchlaufzeit aller Operationen darstellen.

4.3.1 Umsetzung der JSSP-Nebenbedingungen aus der Skript

Die Schwierigkeit bestand darin, die For-Schleifen für jede der Bedingungen korrekt zu erstellen. Insbesondere in dem Fall, dass wir prüfen sollten, ob zwei Aufträge auf einer Maschine zu verarbeiten sind. Wir haben dieses Problem gelöst, indem wir auf den Maschinen iterierten und für jeden zwei Aufträge überprüft haben, ob die Operationen in der Menge der Operationen vorhanden sind.

4.3.2 Umsetzung der zusätzliche TSP-Nebenbedingungen für das JSSP

Diese Implementierung war für uns am schwierigsten. Die Idee hier ist ein Dummy Job am Anfang und am Ende jeder Maschine zu Berücksichtigen. Dieser Job wurde eingestellt, indem wir durch eine zusätzliche Variable der Anzahl von Jobs erhöht haben. Danach haben wir alle unsere vorherige Nebenbedingungen, die wir in der letzten Formulierung Benutzen haben, angepasst.

4.4 Der Aufbau der Abgabe Funktion

Die Ausgabe sollte den Endzustand der Lösung, den Wert der Zielfunktion, die Startzeit jedes Auftrags und die Reihenfolge, in der jeder Auftrag an den Maschinen bearbeitet wird, enthalten.Um die Reihenfolge zu erstellen, werden zunächst die Startzeiten der einzelnen Operationen je Maschine in aufsteigender Reihenfolge sortiert. Dann werden die Aufträge auf den einzelnen Maschinen entsprechend dieser neu gebildeten Menge an Maschinen eingeplant.

5 Performanceanalyse der beiden mathematischen Formulierungen

In unseren theoretischen Arbeit wurden zehn Job Shop Instanzen für die Analyse der Performance unsere implementierte JSSP-Formulierungen. Dabei müsste jede Instanz mit einer maximale Laufzeit von 60 Sekunden gelöst werden. Das Ziel ist es, die maximale Durchlauftzeit der Maschinen zu minimieren. Die folgende Tabelle zeigt die Problemparameters der verschiedenen Instanzen und die daraus resultierenden Ergebnisse.

Probleminstanz	#Jobs	#Maschinen	Dauer(s)	Status	Zfw
1	7	4	0,346	Optimal	82
2	10	4	9,693	Optimal	82
3	18	4	59,956	Zulässig	223
4	7	4	0,37	Optimal	88
5	7	4	0,288	Optimal	83
6	12	4	4,977	Optimal	135
7	17	4	59,21	zulässig	211
8	9	4	0,395	Optimal	102
9	8	4	0,429	Optimal	86
10	13	4	4,049	Optimal	132

Tabelle 4: Maximale Zykluszeit in Abhängigkeit der Probleminstanzen mit der ersten Formulierung (Zfw: Zielfunktiowert, #: Anzahl)

Probleminstanz	#Jobs	#Maschinen	Dauer(s)	Status	Zfw
1	7	4	0,611	Optimal	82
2	10	4	59,45	Zulässig	118
3	18	4	60	Zulässig	233
4	7	4	47,303	Optimal	88
5	7	4	3,512	Optimal	83
6	12	4	60	zulässig	152
7	17	4	60	zulässig	232
8	9	4	60	Zulässig	102
9	8	4	31,643	Optimal	86
10	13	4	59,642	Optimal	142

Tabelle 5: Maximale Zykluszeit in Abhängigkeit der Probleminstanzen mit der zweiten Formulierung (Zfw: Zielfunktiowert, #: Anzahl)

- Für die Probleminstanz 1 finden die beiden Formulierungen den gleichen Zfw. und sogar mit ähnlicher Rechenzeit
- Für die Probleminstanzen 4 und 5 finden die beiden Formulierungen die gleiche Zfw,allerdings mit einer stark differenzierten Rechenzeit
- Für die Probleminstanzen 2 und 10 findet der JSP-Formulierung aus der Skript (1. Formulierung) einen optimalen Zfw. deutlich niedriger als die JSP-Formulierung mit TSP-Variablen(2. Formulierung).
- Für die Instanz 7 findet der Formulierung 1 innerhalb von 60 Sekunden einen zulässigen Zfw. , der kleiner als mit der Formulierung 2.
- Für die Instanz 6 konnte die Formulierung 2 innerhalb von 60 Sekunden keinen optimalen Zfw., obwohl die Formulierung 1 einen optimalen Zfw geliefert hat.
- Für die Instanz 8 haben die beiden Formulierungen den gleichen Zfw. geliefert. Aber für die Formulierung 1 ist diese Lösung Optimal,während für Formulierung 2 diese Lösung nicht optimal ist.

Diskussion

Die Ausführungszeit für die beiden Formulierungen unterscheidet sich für jede Probleminstanz erheblich. Dies kann auf die Komplexität der Nebenbedingungen zurückgeführt werden. Die Formulierung mit TSP-Variablen enthält mehr zusätzliche Bedingungen und benötigt mehr Überprüfungen und damit mehr Ausführungszeit. Wenn wir die Rechenzeit begrenzen, kann eine der Formulierungen eine optimale Lösung finden, während der andere es nicht schafft, obwohl die andere eine zulässige Lösung gefunden hat. Wir glauben auch, dass die Anzahl der Operationen pro Maschinen eine Einfluss auf die Ergebnisse der beiden Formulierung hat, denn der JSSP mit TSP Variablen hat für Instanzen mit großem Anzahl an Operationen schlechtere Ergebnisse bekommen als der JSSP Formulierung aus der Skript.

Wir wollen anhand von Gantt-Diagrammen das Schedule eines Probleminstanzenb darstellen.Dabei wurde Das Problem mit Hilfe der Formulierung 1 gelöst.

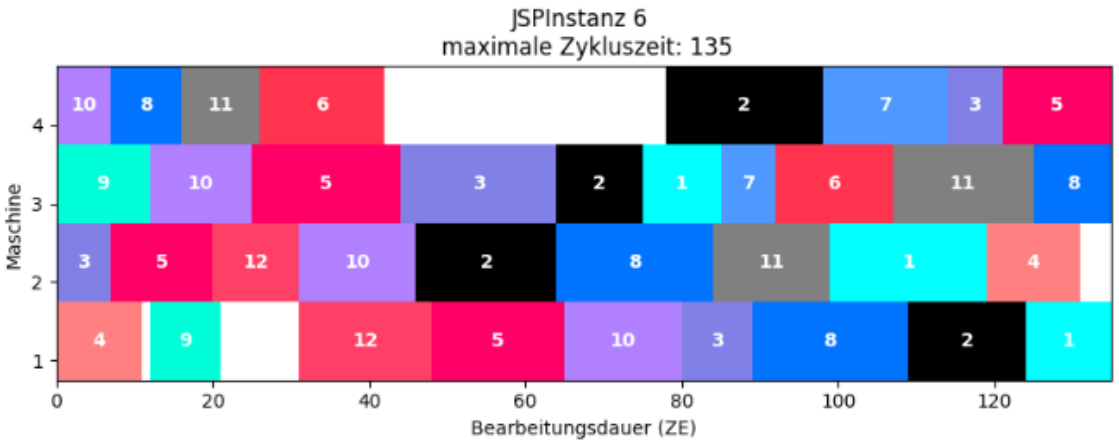

Abbildung 3: Gantt-Diagramm für das Schedule des Probleminstanzen 6

References

[1] M.R. Garey, D.S.Johnson,Ravie Sethi, May, 1976, "The Complexity of Flowshop and Jobshop Scheduling", pp. 117-129

[2] M.Tiedemann, "Algorithmic Approaches to Flexible Job Shop Scheduling", 14 June 2012, https://num.math.uni-goettingen.de/picap/pdf/E730.pdf, pp.34-36.