Technische Universität Clausthal
Dr. Andreas Tillmann
Felix Merz, M.Sc.

Optimierungsheuristiken
Sommersemester 2023
Programmierblatt 2

Freiwillige Abgabe bis einschließlich Donnerstag, den 25.05.2023. Senden Sie dazu ihren kommentierten Code an `felix.merz@tu-clausthal.de`*. Bei Fragen oder Problemen können Sie sich gerne melden.*

Beim Problem Makespan Minimierung auf identischen Maschinen müssen n Jobs auf m identische (parallel arbeitende) Maschinen verteilt werden. Jeder Job $j \in [n] \coloneqq \{1, 2, \ldots, n\}$ muss dabei $p_j > 0$ Zeiteinheiten auf einer Maschine ohne Unterbrechung bearbeitet werden. Jede Maschine kann zu jedem Zeitpunkt nur einen Job bearbeiten. Ziel ist es den letzten Fertigstellungszeitpunkt eines Jobs zu minimieren, d.h., wenn C_j den Fertigstellungszeitpunkt von Job j bezeichnet, möchte man die *Makespan* $C_{\max} = \max_{j \in [n]} C_j$ minimieren.

Aufgabe 1
Lesen Sie die Anzahl der Maschinen m und die Bearbeitungszeiten $p_1, \ldots, p_n$ der Jobs aus der im Stud.IP zur Verfügung gestellten Datei `scheduling.txt` ein. Verwalten Sie die Bearbeitungszeiten der Jobs dabei in einer Liste. Die Datei hat das folgende Format:

```
m=<Integer Anzahl Maschinen>
<Integer Bearbeitungszeit p_1>
<Integer Bearbeitungszeit p_2>
            ...
<Integer Bearbeitungszeit p_n>
```

Aufgabe 2
Bestimmen Sie mit dem folgenden *List-Scheduling-Algorithmus* eine Lösung für die in Aufgabe 1 eingelesene Instanz: Durchlaufen Sie die Jobs *in der gegebenen Reihenfolge* $(1, 2, \ldots, n)$ und ordnen den jeweils nächsten Job einer Maschine zu, die bisher am wenigstens ausgelastet ist (d.h. mit kleinster Summe der Bearbeitungszeiten bereits darauf eingeplanter Jobs). Stehen mehrere Maschinen mit gleicher Auslastung zur Auswahl, wählen Sie diejenige mit kleinstem Index.

Aufgabe 3
Implementieren Sie eine lokale Suche, bei der jeweils ein Nachbar der aktuellen Lösung mit geringerer Makespan ausgewählt wird, solange ein solcher existiert. Ein Nachbar einer Lösung unterscheidet sich von dieser genau durch einen Maschinentausch zweier Jobs (Job j_1 wechselt von Maschine i_1 zu Maschine i_2 und Job j_2 von i_2 zu i_1) oder durch eine Verschiebung von einem Job zu einer anderen Maschine (Job j wechselt von Maschine i_1 zu Maschine i_2). Wenden Sie diese lokale Suche auf die in Aufgabe 2 bestimmte Lösung an.

Aufgabe 4
Implementieren Sie ein Multi-Start Hill Climbing Verfahren, indem Sie für jede Iteration eine zufällige Reihenfolge der Jobs erzeugen, auf diesen den List Scheduling Algorithmus aus Aufgabe 2 anwenden und die erhaltene Lösung anschließend mit der lokalen Suche aus Aufgabe 3 verbessern. Wenden Sie das Verfahren auf die Instanz aus Aufgabe 1 mit 5 Iterationen an.

Allgemeiner Hinweis: Überlegen Sie sich eine kleine Instanz zum Testen.