TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Maschinenwesen Institut für Technische Logistik und Arbeitssysteme

Professur für Technische Logistik

Logistics Lab

Bericht Gruppe 6

Pascal Juppe

Matrikelnummer: 4765520

Nico

Matrikelnummer: 87654321

Ferdinand Thiessen

Matrikelnummer: 87654321

5. März 2022

Inhaltsverzeichnis

1 Aufgabe 1

Aufgabenstellung: Erstellen Sie ein Konzept zur Berechnung eines gültigen sowie möglichst optimalen Einsatzplanes der Fahrzeuge.

1.1 Ruin and recreate

Ein weiterer Ansatz zur Berechnung eines möglichst optimalen Ablaufplans ist die Verwendung der Ruin-and-Recreate Strategie beschrieben durch Schrimpf et al. [1] Dieser Algorithmus verwendet einen iterativen Ansatz, in dem zuerst Teile der Lösung entfernt werden (Ruin-Schritt) und nachfolgend wieder anders zusammengesetzt (Recreate-Schritt) werden. Bei der erneuten Zusammensetzung der Lösung wird dabei ein Ergebnis präferiert, welches eine beschriebene Zielfunktion maximiert - in unserem Anwendungsfall also eine Lösung, die die zurückgelegte Strecke aller Fahrzeuge (und damit die Gesamtausführungszeit des Ablaufplans) minimiert.

Zur Implementation dieser Meta-Heuristik verwenden wir das Java-Framework jSprit, welches auf dem Grundalgorithmus beschrieben durch Schrimpf et al. basiert. Dazu werden unsere bereitgestellten Fahrzeuge durch Objekte der Klasse *Vehicle* dargestellt. Diese werden mit einer Gesamtkapazität von 1 erstellt und initial an den korrespondierenden Maschinen positioniert. Die Transportanforderungen aus transport_demand.txt werden in jSprit als Objekte der Klasse *Shipment* abgebildet. Jedes Shipment hat eine Größe von 1, es werden also gegebenenfalls mehrere Shipments mit der gleichen Start- und Zielposition erstellt, wenn die Transportanforderungen dies vorgeben. Zusätzlich werden dem Algorithmus noch benutzerdefinierte Constraints vorgegeben, die eine Senkung der maximalen Transportzeit im Recreate-Schritt des Algorithmus bevorzugt. Der Algorithmus wird dann mit einer Obergrenze von 2000 Iterationen gestartet und terminiert in ca. 100 Sekunden. Die Lösungsgüte ist in Tabelle 1.1 abgebildet:

Anzahl Fahrzeuge	Berechneter Score
1	6,995.31
3	2,324.02
5	1,389.65

Tabelle 1.1: Ergebnisse ruin-and-recreate

2 Aufgabe 2

2.1 Initiale Probleme

Bei der Umsetzung von Aufgabe 2 wurde zur Programmierung von der hauseigenen Lego Software abgesehen.

Leider wollte uns die Benutzung des `repeat until`-Blockes, welcher eine while-Schleife abbilden soll, nicht gelingen. Sowohl die Bedingung $x < y$ als auch $x > y$ führten nicht zum gewünschten Verhalten. Nur bei $x = y$ zeigte der Roboter eine Reaktion.

Da das Programm die Motoren Rotation allerdings in zu großen Abständen abtastet, wurde nie exakt die 2-Meter-Marke gemessen und somit überfahren. Aufgrund dessen wurde nach einer Alternative gesucht. Der Hinweis der Tutoren, die alte Softwareversion zu benutzen, kam leider zu spät.

Das MicroPython Package, welches ebenfalls von Lego bereitgestellt wird, bot sich hierfür bestens an. Außerdem ermöglichte das eine einfachere Einarbeitung durch Python-Vorkenntnisse, verständlicheren Code und Kollaboration durch Git.

2.2 Experiment 1: Freie Fahrt

2.2.1 Aufgabenstellung

Das Fahrzeug soll ohne weitere Sensorik eine Strecke von 2 m geradeaus zurücklegen

2.2.2 Herangehensweise

Die Geschwindigkeit des Motors setzen wir initial auf 440, da dies scheinbar die Maximalgeschwindigkeit ist.

Um die zurückgelegte Strecke des Roboters zu ermitteln, muss die Rotation des Motors gemessen werden. Mit Hilfe der Funktion `motor.angle()` wird die Motorstellung, als Winkel, ausgelesen und kann mit Hilfe des Radumfangs in eine Strecke umgerechnet werden.

3 Aufgabe 3

Literaturverzeichnis

[1] Gerhard Schrimpf, Johannes Schneider, Hermann Stamm-Wilbrandt, and Gunter Dueck. Record breaking optimization results using the ruin and recreate principle. *Journal of Computational Physics*, 159(2):139–171, 2000.