Code for BWI Lösung

Kenneth Schröder

20. Dezember 2020

Dokumentation

Diese Lösung wurde mit python3 erstellt und nutzt die Bibliotheken tqdm für die Fortschrittsanzeige in der Konsole, sowie pandas zum Einlesen und Verarbeiten des Datensatzes. Auf Linux kann python3 über `sudo apt install python3.8` installiert werden, in einigen Linux Versionen ist es aber schon standardmäßig enthalten. pip3 ermöglicht die einfache Installation von tqdm und pandas.

Anschließend kann die `main.py` mit `python3 main.py` gestartet werden. Wichtig ist dabei nur, dass sich `bwiTable.txt` im gleichen Verzeichnis befindet, da diese Datei die Daten im csv Format enthält.

Auf einem 2,9Ghz Intel Core i9 Prozessor beträgt die Laufzeit ca. 7 Minuten.

Algorithmus

Bei der vorliegenden Aufgabe handelt es sich um eine spezielle Form des Rucksackproblems, bei der zwei Rucksäcke in Form der Transporter zur Verfügung stehen und einzelne Gegenstände mit einer vorgegebenen Häufigkeit genutzt werden dürfen.

Ein korrekter Algorithmus zur Lösung dieses Problems wäre ein dreidimensionaler DP-Algorithmus (dynamische Programmierung), bei dem über die Variablen *Kapazität des ersten Transporters*, *Kapazität des zweiten Transporters* und *Teilmenge der Gegenstände* iteriert wird. Da der größte gemeinsame Teiler aller gegebenen Gewichte ein Gramm beträgt und die Kapazitäten der Transporter in gramm sehr groß sind (1 Mio. Gramm), würde dieser Algorithmus ohne starke Optimierungen eine enorme Laufzeit und großen Speicherverbrauch aufweisen (1Mio. * 1Mio. * 10 Items = 10 Billionen Iterationen mit dynamischen Speicherzugriffen, unoptimiert auch über 1TB Hauptspeicherauslastung).

Einzelne Instanzen dieses zwei-Rucksack-Problems können auch effizienter gelöst werden. Dazu wird angenommen, dass ein großer Transporter, anstelle der zwei kleinen zur Verfügung steht. Dieser bekommt die exakte Summe der Kapazitäten der kleinen Transporter. Dann wird ein regulärer Ein-Rucksack-Algorithmus mit diesem Transporter gestartet. Danach muss

noch die Frage geklärt werden, ob die berechneten Gegenstände auf die zwei kleinen Transporter aufgeteilt werden können. Ist das der Fall, handelt es sich um eine optimale Lösung[1].

Die Frage, ob die Gegenstände auf die zwei kleinen Transporter aufgeteilt werden können, kann mit einem weiteren Durchlauf des Rucksack-Algorithmus geklärt werden. Im zweiten Durchlauf werden nur die Gegenstände und deren Anzahl beachtet, wie sie vom ersten Algorithmus ausgewählt wurden. Der erste Transporter soll damit so voll wie möglich (bzgl. Gewicht) gepackt werden. Also Nutzwert dient dises mal also das Gewicht der Gegenstände. Wenn der Algorithmus eine Lösung findet, die die Kapazität des ersten Transporters so gut ausnutzt, dass die restlichen Elemente in den zweiten Transporter passen, ist eine optimale Lösung gefunden. Falls es eine solche Lösung nicht gibt, müsste der zeitaufwändige Zwei-Rucksack-Algorithmus genutzt werden.

Die vom BWI gegebene Instanz des Problems kann allerdings auf diese Weise gelöst werden! Der entsprechende Algorithmus wurde in `main.py` umgesetzt

Ergebnisse

Hardware	**Anzahl T1**	**Anzahl T2**	**Gesamt**	**Bedarf**
Notebook Büro 13”	0	0	0	205
Notebook Büro 14”	0	0	0	420
Notebook outdoor	0	0	0	450
Mobiltelefon Büro	60	0	60	60
Mobiltelefon Outdoor	155	2	157	157
Mobiltelefon Heavy Duty	219	1	220	220
Tablet Büro klein	398	197	595	620
Tablet Büro groß	0	0	0	250
Tablet outdoor klein	3	1	4	540
Tablet outdoor groß	0	370	370	370
Gewicht inkl. Fahrer	1100,000kg	1099,971kg	2199,971kg	6889,151kg
Nutzwert	41.390	33.270	74.660	168.680

Die Tabelle zeigt eine optimale Verteilung der Hardware auf Transporter 1 (T1) und Transporter 2 (T2). Insgesamt bleiben 29 Gramm ungenutzt. Der transportierte Gesamtnutzwert liegt bei 74.660.

[1]Beweis per Widerspruch: Sei v_{opt} der berechnete, optimale Wert der mit dem großen Transporter transportiert werden kann. Angenommen mit den zwei kleinen Transportern kann ein Wert $v_{klein} > v_{opt}$ erreicht werden, dann könnte man die gleichen Gegenstände einfach in den großen Transporter laden (da ihr Gewicht nach Definition des großen Transporters auch von diesem getragen werden kann) und v_{opt} wäre nicht optimal. Widerspruch