CHRISTIAN-ALBRECHTS-UNIVERSITÄT ZU KIEL
Institut für Informatik, Arbeitsgruppe Theorie der Parallelität
Prof. Dr. K. Jansen, K. Klein

17. Dezember 2015

Übungen zur Vorlesung »WInf III / Einf. OR«

Übungsblatt 3a

Hausaufgabe 3a.1 (Wettbewerbsaufgabe)
Entwerfen Sie eine Heuristik, die den Wert der Lösung des Savings-Algorithmus abschätzt.

Die Eingabe besteht aus den Daten zu Kundenbesuchen. Zu jedem Kundenbesuch sind die geographischen Koordinaten und die Besuchsdauer bekannt. Zusätzlich sind die Koordinaten des Depots und Fahrzeiten vom Depot zu jedem Kunden, aber nicht die Fahrzeiten zwischen den Kunden, gegeben, siehe auch die Datei problems/fls/lib/java/Node.java.

Gesucht ist die Gesamtdauer aller Touren der Lösung, die der Savings-Algorithmus auf den vollständigen Daten (d.h. mit bekannter Fahrtdauer zwischen den Kunden) liefert, d. h. der Wert dieser Lösung soll möglichst gut angenähert werden. Dazu ist ein knappes Zeitlimit von 200 ms gegeben. Der Savings-Algorithmus hat als zusätzliche Einschränkung, dass die Gesamtdauer (Fahrtdauer und Besuchsdauer) jeder Tour nicht 8 Stunden überschreiten darf. Der Algorithmus, mit dem die zu erstellende Heuristik verglichen wird, ist die auf Seite 2 gegebene Variante des Savings-Algorithmus.

Bewertet werden nur Abgaben, die mit dem Auswertungsprogramm funktionieren: python test.py -t 0.2 fls <program>

Hinweis: Im Gegensatz zu den bisherigen Aufgaben findet hier die Kommunikation mit dem Auswertungsprogramm auf dem Standardausgabestrom (stdout) statt. Um die Kommunikation nicht zu stören, sollten Ausgaben (z.B. zu Debug-Zwecken) auf der Standardfehlerausgabe ausgegeben werden (System.err.print(...) bzw. System.err.println(...) in Java).

Abgabe: Donnerstag, den 14. Januar, bis spätestens 10 Uhr im Schrein

input : Number n of visits, travel time matrix $(c_{ij})_{0\le i,j\le n}$, visit durations $(d_i)_{1\le i\le n}$
output: A list of tours

Initialize list with trivial tours:
tours $= \emptyset$
for $i = 1$ **to** n **do**
    tours $=$ tours $\cup \{(0,i,0)\}$

Merge tours while possible:
merge $=$ **True**
while merge **do**
    maxsaving $= 0$
    merge $=$ **False**
    foreach $t, t' \in$ tours **do**
        let $t = (0, v_1, \ldots, v_k, 0), t' = ((0, v'_1, \ldots, v'_\ell, 0))$
        $t_{\text{new}} = (0, v_1, \ldots, v_k, v'_1, \ldots, v'_\ell, 0)$
        if t_{new} *is shorter than 8 hours* **then**
            saving $= c_{v_k,0} + c_{0,v'_1} - c_{v_k,v'_1}$
            if saving $>$ maxsaving **then**
                maxsaving $=$ saving
                newtours $= T \setminus \{t, t'\} \cup \{t_{\text{new}}\}$
    if maxsaving > 0 **then**
        merge $=$ **True**
        tours $=$ newtours
return tours