Programmierpraktikum Diskrete Optimierung
Das Vehicle-Routing-Problem

Prof. Dr. Stephan Held
held@dm.uni-bonn.de

Sommersemester 2017

1 Problemformulierung

Wir betrachten in diesem Praktikum TOURENPLANUNGSPROBLEME (engl.: CAPACITATED VEHICLE-ROUTING-PROBLEM (CVRP)). Hierbei ist eine endliche Menge von Kunden gegeben, die von einem Standort (z.B. einer Fabrik) aus mit Gütern beliefert werden müssen. Jeder Kunde hat eine Güternachfrage und die Auslieferung erfolgt über Lastwagen mit einer festen Kapazität.

Das Vehicle-Routing-Problem hat auch eine Anwendung im Chip-Design, wo die Kunden den Registern (1-Bit-Speicherelementen) auf einem Chip entsprechen. Um einen Chip nach der Produktion testen zu können, müssen wir in der Lage sein Bit-Pattern in die Speicherelemente zu laden. Dies wird über sogenannte Scan-Ketten druchgeführt, bei denen bis zu Q Register durch einem Weg aneinandergekettet werden. Über diese Wege können dann in Q Rechenzyklen Bit-Pattern eingelesen oder auch ausgelesen werden. Um ein schnelles Ein-/Auslesen zu bewerkstelligen, ist Q i.d.R. eine kleine Konstante, z.B. $Q = 50$. Die Nachfrage eines jeden Registers ist eins. Die Ketten sollen möglichst resource-schonend realisert werden, d.h. wir suchen eine Partitionierung aller Register sowie Touren in jeder Partitionsmege/Scan-Kette, so dass die Gesamtlänge aller Scan-Ketten minimal ist.

Mathematisch kann das kapazitätsbeschränkte Vehicle-Routing-Problem wie folgt spezifiziert werden:

Instanz:

- Ein vollständiger gewichteter Graph (K_{n+1}, c)
- Eine **Kundenmenge** $N = V(K_n) \subset V(K_{n+1})$
  mit einer **Nachfragefunktion** $q : N \to \mathbb{N}$.
- Ein **Depot** $r \in V(K_{n+1}) \setminus N$ (also $\{r\} = V(K_{n+1}) \setminus N$).
- Eine **Ladekapazität** $Q \in \mathbb{N}$.

Lösungen: Eine **zulässige Fahrzeugtour/Cluster** ist ein Kreis C in K_{n+1} mit $r \in V(C)$ (das Fahrzeug startet und ended im Depot r) und

$$q(C) := \sum_{v \in V(C) \setminus \{r\}} q(v) \leq Q.$$

Eine **zulässige Lösung des VRP** besteht aus einer Menge von Clustern $\{C_1, \ldots, C_k\}$, mit $V(C_i) \cap V(C_j) = \{r\}$ für alle $1 \leq i \neq j \leq k$ und

$$V(K_{n+1}) = \bigcup_{i=1}^{k} V(C_i).$$

Aufgabe: Finde eine zulässige Lösung $\{C_1, \ldots, C_k\}$ des VRP, mit minimaler Länge

$$\sum_{i=1}^{k} c(E(C_i)).$$

Hierbei treten zwei in sich schon schwere Teilproblem auf:

- **Partitionierung** der Kunden in zulässige Cluster $C_1, \ldots, C_k$.
- **TSP-Tour** innerhalb jedes Clusters.

2 Eingabe-Daten

Die Problem-Instanzen liegen im TSPLIB-Format für geometrische Instanzen vor:
`http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/DOC.PS`.

Dieses Format war urpsrünglich für das Traveling-Salesman-Problem (TSP) gedacht, enthält aber Erweiterungen für Vehicle-Routing-Probleme. Wir verwenden nur die Kostentypen `EUC_2D`, `CEIL_2D` oder `MAN_2D`. In diesem Format gibt es insbesondere einen Abschnitt `NODE_COORD_SECTION`, in dem die Kundenkoordinaten in Zeilen mit drei Integer-Zahlen der Form:

`<Kunden-Nr.> <x> <y>`

aufgelisted sind. Die einheitliche Fahrzeugkapazität wird in einer Zeile

`CAPACITY : <Q>`

spezifizert. Ausserdem gibt es einen Abschnitt `DEMAND_SECTION`, in dem für eine Teilmenge der Knoten der Demand $q(v)$ spezifiziert wird:

`<Kunden-Nr.> <q>`,

sowie einen Abschnitt `DEPOT_SECTION` in dem die (moeglicherweise mehreren) Depots spezifiziert sind, wobei wir ggf. nur das erste Depot verwenden.

Instanzen werden hier veröffentlicht:
`http://www.dm.uni-bonn.de/lectures/ss17/praktikum_ss17.html`

3 Programm

Das Programm soll den Dateinamen mit der Instanz sowie die Parameter u und f aus der Kommandozeile lesen. Der Aufruf erfolgt dann wie folgt

`programm <Instanzdateiname>`

Das Programm sollte die Lösung in folgendem Format in die Standardausgabe schreiben:

1. Eine Zeile, welche die Gesamtlänge aller Touren angibt:
   `OBJECTIVE <Gesamtlänge>`
2. Für jede erzeugte Route eine Zeile mit der Nummer $(1, \ldots, |I|)$ und den Indizes der Kunden:
   `ROUTE <Route-Nr.> <Kunde-1> <Kunde-2> ... <Kunde-<=k>`

Programmiersprache

Die Implementierungen müssen in C/C++ ausgeführt werden und unter Linux mit den Compilern clang/g++/gcc kompilierbar sein, d.h. sie müssen dem ANSI C11/C++11-Standard genügen. Empfohlen wird C++!!!

Außer den Standard-Bibliotheken, insbesondere der STL, dürfen keine Hilfsbibliotheken verwendet werden, es sei denn es wird in der Aufgabenstellung ausdrücklich erwähnt. Ausnahmen sprechen Sie bitte mit ihrem Betreuer ab. Der Code muss verständlich programmiert und kommentiert werden. Selbstverständlich müssen die Programme fehlerfrei funktionieren. Insbesondere werden sie mit dem Programm valgrind (`http://valgrind.org`) sowie dem Clang-Analyzer (`http://clang-analyzer.llvm.org/`) auf Fehler überprüft, welche zu Abzügen führen.

Eine mathematische Einführung in C++ findet sich insbesondere in dem Buch von Stefan Hougardy und Jens Vygen: "Algorithmische Mathematik", Springer, 2015.

4 Einführungsaufgabe

Implementieren Sie folgende Heuristik für das CVRP.

1. Berechnen sie eine kurze Tour T durch alle Kunden mittels der 2-OPT Heuristik, die mit einer beliebigen Tour startet und diese iterative durch den Austausch von 2 Kanten durch zwei neue Kanten verbessert (siehe B. Korte und J. Vygen: "Combinatorial optimization", Springer, 2011, Kapitel 21.3).

   Diese Funktion werden Sie auch später nutzen, um am Ende die Fahrzeugtouren zu verbessern.

2. Spalten Sie T wie folgt in Unterwege auf: Folge der Tour T startend im Depot. Immer wenn die Hinzunahme des nächsten Knotens auf T die Kapazität überschreiten würde, beginne eine neue Subtour. Bei Einheitskapazitäten von 1 entstehen so genau $\lceil |N|/Q \rceil$ Fahrzeugtouren.

Spätester Abgabetermin für die Einführungsaufgabe ist der 30.04.2017, 24 Uhr.

Wer die Einführungsaufgabe nicht meistert, ist durchgefallen!

Wir empfehlen zu diesem Zeitpunkt auch schon die Hauptaufgabe weitestgehend fertigzustellen, da im Semester neben Seminaren und Übungen häufig wenig Zeit bleibt.

5 Hauptaufgaben

Als Hauptaufgabe hat jeder Teilnehmer eine der folgenden ihm zugewiesenen Algorithmen zu implementieren. Nicht alle Algorithmen sind für große Instanzen brauchbar. Dennoch sollte versucht werden, auch auf großen Instanzen gute Lösungen zu berechnen. Beim Branch-&-Bound kann z.B. die Anzahl der Branching-Knoten beschränkt werden. Hierdurch kann dann ggf. keine Optimalität gewährleistet werden, aber durch Verleich von Relaxierung und zulässiger Lösung kann deren Güte abgeschätzt werden.

Alle Teilnehmer sollten nach Ihrem Hauptalgorithmus, einfache Verbesserungs-Heuristiken, wie von Clarke und Wright [1964]:

(`http://neo.lcc.uma.es/vrp/solution-methods/heuristics/savings-algorithms/`),

oder die 2-OPT Heuristik innerhalb der Cluster, benutzen, um suboptimale Lösungen zu verbessern.

5.1 Matching-basierte Branch-&-Bound-Algorithmen

Aufgabe 1: Implementieren Sie den Branch-&-Bound-Algorithmus von Laporte et al. auf Basis des Assignment-Problems (angepasst auf symmetrische Instanzen).

G. Laporte, H. Mercure, Y. Norbert: "A Branch and Bound Algorithm for a Class of Assymmetrical Vehicle Routing Problems", J. Opl. Res. Soc. 43(5), 1992, 469–481.

Aufgabe 2: Implementieren Sie den Branch-&-Bound-Algorithmus von Fischetti et al. basierend auf der ADD_DISJ Procedure für untere Schranken (Kapitel 4).

Fischetti, Toth, Vigo: "A Branch-And-Bound Algorithm for the Capacitated Vehicle Routing Problem on Directed Graphs", OR 42(5), 1994, 846–859.

Aufgabe 3: Implementieren Sie den Branch-&-Bound-Algorithmus von Fischetti et al. basierend auf der Min-Cost-Flow Relaxierung für untere Schranken (Kapitel 5).

Fischetti, Toth, Vigo: "A Branch-And-Bound Algorithm for the Capacitated Vehicle Routing Problem on Directed Graphs", OR 42(5), 1994, 846–859.

5.2 Spannbaum-basierte Branch-&-Bound-Algorithmen

Aufgabe 4: Implementieren Sie den Branch-&-Bound-Algorithmus von Christofides et al.. Dabei sollen Sie die k-degree center trees als Schranke benutzen (nicht die q-routes). *Christofides, Mingozzi, Toth: "Exact Algorithms for the Vehicle Routing Problem, Based on Spanning Tree and Shortest Path Relaxations [1981]", Mathematical Programming 20, 1981, 255–282.*

Aufgabe 5: Implementieren Sie den Branch-&-Bound-Algorithmus von Christofides et al. Dabei sollen Sie die q-routes center trees als Schranke benutzen (nicht die k-trees).

Christofides, Mingozzi, Toth: "Exact Algorithms for the Vehicle Routing Problem, Based on Spanning Tree and Shortest Path Relaxations [1981]", Mathematical Programming 20, 1981, 255–282.

Aufgabe 6: Implementieren Sie den Branch-&-Bound-Algorithmus von Fisher mit verbesserten k-Bäumen.

M.L.Fisher: "Optimal Solution of Vehicle Routing Problems Using Minimum k-Trees", OR 42(4), 1994, 626–642. Mathematical Programming 20, 1981, 255–282.

M.L.Fisher: "A Polynomial Algorithm for the Degree-Constrained Minimum k-tree Problem", OR 42(4), 1994, 775–779.

5.3 Partitionierungs-Heuristiken

Aufgabe 7: Implementieren sie die Partitionierungs-Heuristiken und lokale Suche nach Taillard.

E. Taillard: "Parallel Iterative Search Methods for Vehicle Routing Problems", Networks 23, 1993: 661–673.

Aufgabe 8: Implementieren Sie den Multiple-Partition Algorithm von Golden et al., startend mit einer 2-OPT TSP Lösung.

B. Golden, A. Assad, L. Levy, and F. Gheysens: "The fleet size and mix vehicle routing problem", Computers & Operations Research 11(1), 1984, 49–66.

5.4 Approximations Algorithmen

Aufgabe 9: Implementieren Sie die einfache Approximationsheuristik und das Approximationsschema von Haimovich und Kan.

M. Haimovich and A.H.G.R.Kan: " Bounds and Heuristics for Capacitated Routing Problems", Mathematics of Operations Research 10(4), 1985, 525–542.

Aufgabe 10: Implementieren Sie den Approximationsalgorithmus von Asano et al.

T. Asano, N. Katoh, H. Tamaki, and T. Tokuyama: "Covering points in the plane by k-tours: towards a polynomial time approximation scheme for general k, STOC'97, 1997, 275–283.

Aufgabe 11: Implementieren Sie den Approximationsalgorithmus von Berman und Das.

P. Berman, S.K.Das:"On the Vehicle Routing Problem" WADS 2005, 360–371.

6 Abgabe und Abschlussvortrag

Spätester Abgabetermin für die Einführungsaufgabe ist der 30.04.2017, 24 Uhr.
Spätester Abgabetermin für das Programmierpraktikum ist der 23.07.2017, 24 Uhr.

Am Ende des Semesters muss jeder Teilnehmer seine Implementierung im Rahmen eines Blockseminars vorgestellen. Hierbei sollen in 15 Minuten (12 Minuten Vortrag + 3 Minuten Diskussion) der Algorithmus, die interessantesten Code-Fragmente, sowie experimentelle Ergebnisse vorgestellt werden. Voraussichtlicher Termin für das Blockseminar ist der 28.07.2017.