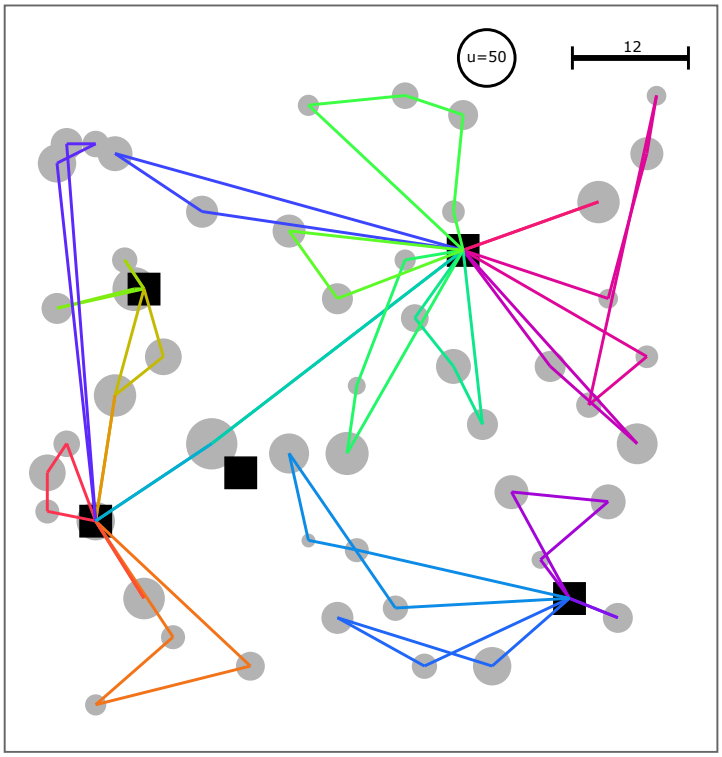
Universität Augsburg

Institut für Mathematik

Ausarbeitung

zum Programmierprojekt

...



von:
Lukas Graf

Betreut von:
Prof. Dr. Tobias Harks

„Abstract“

Zusammenfassung/Überblick der Arbeit

1 Capacitated Location Routing (CLR)

1.1 Problemdefinition

Eine Instanz des **Capacitated Location Routing Problems (CLR)** ist gegeben durch:

- einen ungerichteten, zusammenhängenden Graphen $G = (V, E)$,
- einer Partition der Knoten in Kunden $\mathscr{C}$ und Fabrikstandorte $\mathscr{F}$,
- einer metrischen Kostenfunktion auf den Kanten $c : E \to \mathbb{R}_{\geq 0}$,
- Eröffnungskosten für die Fabriken $\phi : \mathscr{F} \to \mathbb{R}_{\geq 0}$,
- Bedarfe der Kunden $d : \mathscr{C} \to \mathbb{R}_{\geq 0}$
- und einer einheitlichen Kapazität $u > 0$ für die Fahrzeuge.

Zulässige Lösungen bestehen aus

- einer Teilmenge $F \subseteq \mathscr{F}$ von eröffneten Fabriken
- und einer Menge von Touren $\mathcal{T} = \{T_1, \ldots, T_k\}$,

sodass gilt:

- Zu jeder Tour gibt es eine geöffnete Fabrik $f \in F$, an der diese startet und endet.
- Alle Touren zusammen erfüllen alle Bedarfe der Kunden.
- Keine der Touren übersteigt die Kapazität u.

Das Optimierungsziel ist es die Gesamtkosten für das Eröffnen der Fabriken und die gefahrenen Touren zu minimieren, also die Minimierung der Kostenfunktion [1]

$$\sum_{T \in \mathcal{T}} c(T) + \sum_{f \in F} \phi(f)$$

Beobachtung 1.1. CLR ist NP-schwer, denn es beinhaltet beispielsweise metrisches TSP (betrachte Instanzen mit $|\mathscr{F}| = 1$, $d \equiv 1$ und $u = |\mathscr{C}|$).

[1] Wir verwenden hier, dass eine Funktion $\mathbb{R}$-wertige Funktion $c : M \to \mathbb{R}$ auf einer Menge M eine Funktion $\tilde{c} : \mathcal{P}(M) \to \mathbb{R} : M \supseteq N \mapsto \sum_{x \in N} c(x)$ auf der Potenzmenge $\mathcal{P}(M)$ induziert. Zur Vereinfachung der Notation bezeichnen wir diese Funktion dann ebenfalls mit c. Unter nochmaliger Anwendung dieser Konvention ließe sich die obige Kostenfunktion daher auch als $c(\mathcal{T}) + \phi(F)$ schreiben.

1.2 Ein Approximationsalgorithmus für CLR

Der in [HKM13] beschriebene 4, 38-approximative Algorithmus für CLR basiert im Wesentlichen auf den folgenden Schritten (schematisch dargestellt in Abb. 1):

Algorithmus beschreiben (auf schlechtere Approximationsgüte der Implementierung hinweisen!)

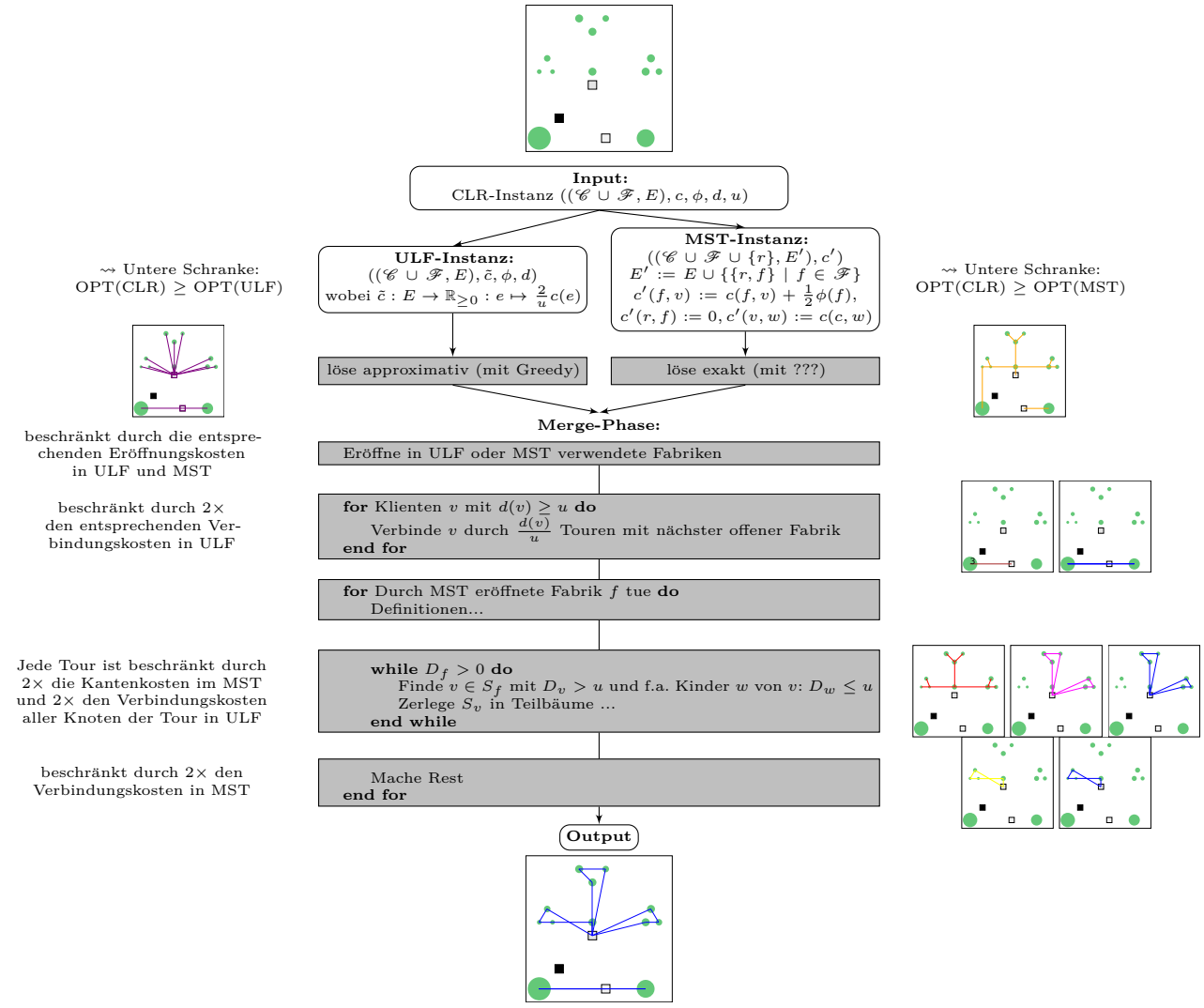


Abbildung 1: Schematische Darstellung des Algorithmus für CLR

1.3 Visualisierung

Im ersten Teil des Programmierprojektes ging es darum den Ablauf sowie das Ergebnis des oben beschriebenen Algorithmus zu visualisieren. Dazu wurde die existierende

Implementierung des Algorithmus um eine Klasse `CLR_Drawing` erweitert. Diese wird zu Beginn des Algorithmus mit der zu lösenden Instanz initialisiert und kann dann an verschiedenen Stellen des Algorithmus aufgerufen werden, um einen Schnappschuss mit dem momentanen Stand zu erstellen.

Klasse detaillierter Beschreiben + BILDER!

2 CLR with Hard Facility Capacities (CLRhFC)

2.1 Problemdefinition

Eine Instanz von **Capacitated Location Routing with Hard Facility Capacities** (CLRhFC) ist gegeben durch:

- eine Instanz $(G = (\mathscr{C} \cup \mathscr{F}, E), c, \phi, d, u)$ von CLR
- und zusätzlich Kapazitäten der Fabriken $l : \mathscr{F} \to \mathbb{R}_{\geq 0}$.

Zulässige Lösungen sind Lösungen der zugrunde liegenden CLR-Instanz, die zudem die Kapazitätsschranken der Fabriken einhalten.

Das Optimierungsziel ist weiterhin die Minimierung der unveränderten Kostenfunktion der CLR-Instanz.

Bemerkung 2.1. Es gibt auch Varianten von CLR mit *weichen Fabrikkapazitäten.*

Näher erklären, Quellen (z.B. 10.1007/s00453-007-9032-7 ?)

2.2 Lösungsansätze

Um überhaupt zulässige Lösungen für CLRhFC zu finden, muss der bestehende Algorithmus an wenigstens zwei Stellen angepasst werden: Beim Erstellen der Touren muss nun zusätzlich darauf geachtet werden, dass die Kapazität der ausgewählten Fabrik nicht überschritten wird. Und zudem muss sichergestellt werden, dass überhaupt immer offene Fabriken mit freier Kapazität verfügbar sind.

Ersteres lässt sich sicherstellen, indem Touren gegebenenfalls noch weiter in Teiltouren aufgespalten werden, die jeweils klein genug sind, um von einer einzelnen offenen Fabrik vollständig beliefert zu werden.

Für letzteres gibt es zwei naheliegende Ansätze: Entweder man passt die ULF- und/oder MST-Phase so an, dass bereits hier die Fabrikkapazitäten berücksichtigt und dementsprechend viele Fabriken geöffnet werden. Oder man erlaubt dem Algorithmus noch in der Large-Demand- bzw. Merge-Phase bei Bedarf zusätzliche Fabriken zu eröffnen.

Der erste Ansatz ließe sich beispielsweise dadurch verwirklichen, dass statt einer ULF-Instanz eine Instanz des *nonuniform Capacitated Facility Location Problems* erstellt und

(approximativ) gelöst wird. Ein entsprechender auf lokaler Suche basierender Approximationsalgorithmus wird in [PTW01] beschrieben. Alternativ (oder auch zusätzlich) könnte man versuchen beim Erstellen des Spannbaumes die Fabrikkapazitäten zu berücksichtigen. Dies hätte möglicherweise den Vorteil . Allerdings ist nicht klar, wie eine entsprechende Anpassung dieser Phase aussehen könnte.

Add Vorteil

Der zweite Ansatz - neue Fabriken bei Bedarf eröffnen - ist dagegen deutlich leichter zu implementieren und ist daher auch der, der in diesem Programmierprojekt weiterverfolgt wurde. Er hat allerdings den Nachteil, dass dadurch der Zusammenhang zwischen den Kosten der in ULF- und MST-Phase gefundenen Lösungen und denen der letztendlich bestimmten Lösung der CLRhFC-Instanz verloren gehen.

siehe Abschnitt 2.5

2.3 Algorithmus

Beschreibung des angepassten Algorithmus

2.3.1 Algorithmus 1

2.3.2 Algorithmus 2

2.4 Implementierungen

2.4.1 Zulässigkeitsprüfung

Beschreibe Edmonds-Karp-Implementierung

2.5 Analyse der Algorithmen

2.5.1 Theoretische Betrachtungen

Untere Schranken

Schlechte Beispiele

2.5.2 Heuristische Beurteilung

$$
\begin{aligned}
&\text{minimiere } \sum_{f\in\mathscr{F}} o_f\phi(f) + \sum_{T\in\mathscr{T}} y_T c(T) \\
&\text{unter d. N.: } \sum_{v\in T\setminus\mathscr{F}} x_{vT} && \leq u y_T && ,T\in\mathscr{T} \\
&\qquad \sum_{T\in\mathscr{T}_f}\sum_{v\in T\setminus\mathscr{F}} x_{vT} && \leq l(f) && ,f\in\mathscr{F} \\
&\qquad y_T && \leq (l(f)/u+1)\, o_f && ,f\in\mathscr{F}, T\in\mathscr{T}_f \\
&\qquad \sum_{T\in\mathscr{T},v\in T} x_{vT} && \geq d(v) && ,v\in\mathscr{C} \\
&\qquad o_f\in\{0,1\},\quad y_T\in\mathbb{N}_0,\quad x_{vT}\geq 0
\end{aligned}
$$

2.6 Ausblick

Was könnte man verbessern? Welche Probleme gibt es dabei?

Liste der noch zu erledigenden Punkte

Zusammenfassung/Überblick der Arbeit 2

Algorithmus beschreiben (auf schlechtere Approximationsgüte der Implementierung hinweisen!) 3

Klasse detaillierter Beschreiben + BILDER! 4

Näher erklären, Quellen (z.B. 10.1007/s00453-007-9032-7 ?) 4

Add Vorteil 5

siehe Abschnitt 2.5 5

Beschreibung des angepassten Algorithmus 5

Beschreibe Edmonds-Karp-Implementierung 5

Untere Schranken 5

Schlechte Beispiele 5

Was könnte man verbessern? Welche Probleme gibt es dabei? 6

Literatur

[HKM13] Tobias Harks, Felix G. König und Jannik Matuschke. „Approximation Algorithms for Capacitated Location Routing“. In: *Transportation Science* 47.1 (2013), S. 3–22. DOI: http://dx.doi.org/10.1287/trsc.1120.0423. URL: http://researchers-sbe.unimaas.nl/tobiasharks/wp-content/uploads/sites/29/2014/02/HKM-TS-2013.pdf.

[PTW01] Martin Pal, Eva Tardos und Tom Wexler. „Facility Location with Nonuniform Hard Capacities“. In: *Proceedings of the 42nd IEEE Symposium on the Foundations of Computer Science*. 2001, S. 329–338. URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.16.1003.

[Tur10] Mark Turney. *simple-svg*. Google Code Archive. simple-svg ist eine header-only C++ Library, mit deren Hilfe einfache svg-Graphicen erstellt werden können. 2010. URL: https://code.google.com/archive/p/simple-svg/.