

Practically solving the VRPTW

Bachelor Thesis

Verfasser:

Pascal Lüscher

Janik Meyer

Studiengang:

Informatik

Auftraggeber:

Fabian Affolter im Namen der Chlausgesellschaft Niederwil-Nesselnbach

Betreuer:

Simon Felix

Fabian Affolter

März, 2019

HELPERS

Equation

$$a^2 + b^2 = c^2 \quad (X)$$

Im Word dann (X) mit einem SEQ Feld erstellen

https://www.youtube.com/watch?v=9YGTH4WrY_8

VARIABLEN

V Menge aller Besuche, $v_i \in V$

S Menge aller Chläsue

Chläuse, welche zusätzlich hinzugefügt wurden, sind in der Menge $AS \subseteq S$ enthalten

$w_{i,j}$ Weg von Visit_i nach Visit_j

$B \subseteq V$ Menge aller Pausen

$v_0 \in V$ Startpunkt

$d_{vi} = (22, 15) \Rightarrow d_{vi0} = 22, d_{vi1} = 15$ Desired

$u_{vi} = (22, 15)$ Unavailable

P Menge aller Tage, $p_i = (0, 86000) \in P$

l_v Dauer des Besuch

$(c_{sv}$ Startzeit von Chlaus s bei Besuch v)

$b_s \in B$ ist die Pause, welche für den Chlaus s vorgesehen ist.

1-Indexiert arbeiten !

Todo:

Sicherstellen, dass überall Punkt als Dezimaltrennzeichen verwendet wird.

(<http://herrthees.de/2012/04/25/google-fusion-table-und-das-dezimale-komma/>)

Konsistente Benennung der Algorithmen prüfen

Dataset verbotenes Wort

Begriffe: Was ist Besuch? Pause/Familie?

Kosten sind in CHF, schauen ob das überall stimmt und gekennzeichnet ist

Feedback Herr Felix

- Kapitel 3: Ich hoffe, Sie werden nicht (sinngemäss) «Vorschlag von Herrn Felix» schreiben. Es gibt gute Gründe, weshalb ich (oder Herr Vogel) bei unseren Vorschlägen gelandet sind. Diese könnten für den Leser interessant sein – im Gegensatz den persönlichen Vorlieben von Herrn Vogel oder mir.
- In der Datasets-Tabelle alle Beispiele auflisten (7, 8)

- Fallfehler, Grammatikfehler, Rechtschreibfehler (z.B. «wird als phänotypischen Suchraum bezeichnet», «dass die Wegzeiten zwischen den Besuchen fix ist», «Durch das einführen»)
- Nummern für Unterkapitel 3.x
- «mindestens 1 Flow vorhanden sein muss» ist sprachlich schlecht. Die ganze Beschreibung ist für Nicht-Experten wenig hilfreich. Entweder Ausführlicher beschreiben, so dass man's verstehen könnte, oder einfach auf eine Referenz verweisen (z.B. IP5 oder ein Paper). Zwischenresultate der Teilschritte beschreiben!
- Bei der Diskussion von Zielfunktionen bitte die Probleme im klassischen Format beschreiben (z.B. wie folgt):

$$\text{minimize } \sum_i^x \sum_j^y abc$$

$$\text{s.t. } \sum_i y_{ij} = 1, i = 0, 1, \dots, n-1$$

- Visualisierungen von Systemen, Ansätzen, manchen Tabellen, etc. wären hilfreich
- ILP Grid Search Result: 0.1 liegt «gefährlich» nahe beim schlechtesten Resultat. Vermutlich wäre 0.65 robuster.
- «Laut Herrn Vogel» ist nicht zitierbar (wie sie richtig rot bemerkt haben)
- Fremde Bilder (GA) übernehmen ist heikel. Am Einfachsten ist es, wenn Sie die Grafiken kurz selbst (angepasst auf Ihren Anwendungszweck) neu aufbauen.
- Erster Absatz zu LocalSolver ist mässig gut formuliert («LocalSolver ist ein Produkt der Firma LocalSolver.» Ach? «Man kann ihn für verschiedene Probleme einsetzen» Ach?)
- «Die Aufteilung der Phasen Zwei und Drei hat den Grund, dass verlängerte Wege eine grosse Performance Auswirkung auf das Modell habe». Können Sie das zeigen?
- «gibt es nun drei Hyperparameter» - Welche? (Dass es sich um Zeitlimiten handelt, kann man als Leser erst zwei Sätze später zwischen den Zeilen rauslesen)
- LocalSolver Grid Search: P3-Diagonalen labellen wäre nützlich. In der Diskussion explizit schreiben, wozu welche Zahlen stehen. (0.8, 0.1) ist m.E. zu unklar. Weniger Nachkommastellen reichen, oder?
- Was der Unterschied von LocalSolver zu ILP ist, wird nicht klar. Vorteile/Nachteile/...?
- Evaluation: Sind HD oder Gehäuse relevant?
- «Verbrauchte Zeit» -> «Rechenzeit»
- IP5 -> IP5 SCIP
- Aus Label wird nicht klar, ob ILP2 mit GUROBI oder SCIP ist?
- Viele Barcharts (z.B. Visualisierung Skalierbarkeit ist unintuitiv
- «Schwankungen» -> «Zuverlässigkeit» oder «Lösungsstabilität»
- Schwankungen könnten als Whiskers dargestellt werden
- Aufteilung in Datensätze und «Neue Datensätze» (Pg32) ist ungünstig
- Evaluation wirkt unstrukturiert. Es ist als Leser sehr schwierig, alle Fakten, Stärken & Schwächen im Kopf zu halten. Vielleicht (!) wäre es einfacher, die Resultate nach Solver gruppiert zu diskutieren?
- Wie schneidet ILP2 SCIP ab? (via MPS sollten Modelle ausgetauscht werden können)

- Eventuell Formeln zusammenfassen, ILP Formel (33) (wobei das drei sind) sind für wegzeiten da.

Feedback Herr Affolter

- Formulieren Sie "man" Sätze besser neu.

Zusammenfassung

Diese Projektarbeit befasst sich mit Chläusen, die Familien besuchen. Familien haben gewisse Zeiten, an denen sie nicht besucht werden können und gewisse Wunschzeiten, an denen sie bevorzugt besucht werden wollen. Es kommen mehrere Chläuse zum Einsatz, die an mehreren Tagen unterwegs sind. Jeder Chlaus hat pro Tag nur eine beschränkte Kapazität. Die Frage ist, wann und in welcher Reihenfolge die Familien von den Chläusen besucht werden sollen, sodass die Kosten minimal werden. Das Problem wird in der Fachliteratur Vehicle Routing Problem with Time Windows genannt.

In einem Vorgängerprojekt wurde schon ein Algorithmus basierend auf ganzzahlig linearer Programmierung entwickelt, der das oben beschriebene Problem lösen kann. Im Rahmen dieser Arbeit werden vier neue Lösungsansätze vorgestellt, die dieses Optimierungsproblem für unseren Kunden lösen können. Diese Ansätze werden dann in einer Evaluation miteinander verglichen, dabei werden verschiedene Aspekte betrachtet.

Aus der Evaluation geht hervor, dass, gemessen an der Zielfunktion, die neu entwickelten Ansätze bessere Routenpläne, als die von Hand erstellten, erzeugen. Die Erkenntnisse aus dieser Arbeit ermöglichen es der Chlausgesellschaft, Kosten zu sparen. Genauer gesagt, **kann/können** der Arbeitsaufwand für die Planung und die Kosten für die Besuche reduziert werden. Es wird gezeigt, dass mittels optimierter Routen in den Jahren 2017 und 2018 insgesamt 230 CHF hätte gespart werden können, was 15.6% der Gesamtkosten der Besuche entspricht.

Inhaltsverzeichnis

HELPERS	1
Equation	1
Feedback Herr Felix	1
Feedback Herr Affolter	3
Zusammenfassung	3
Inhaltsverzeichnis	4
1. Einleitung	6
2. Problemdefinition	8
3. Lösungsansätze (Kapitel)	11
3.1 IP5 - Integer Linear Programming	11
3.2 ILP - Ein-Modell Formulierung	13
3.3 Genetischer Algorithmus	21
3.4 LocalSolver	32
3.5 Google OR-Tools Routing	37
4. Evaluation	42
4.1 Messmethodik	42
4.2 Übersicht	43
4.3 Laufzeit	44
4.4 Vergleich manuell geplante Route	44
4.5 Skalierbarkeit	47
4.6 Lösungsqualität	48
4.8 Lösungsstabilität	50
4.7 Auswirkung Wunschzeiten und Nichtverfügbarkeit	53
4.8 Zusätzliche Chläuse	57
4.9 Preis/Lizenzkosten	57
5. Schlussfolgerung / Empfehlung	59
6. Verzeichnisse	61
Abbildungsverzeichnis	61
Tabellenverzeichnis	61
Literaturverzeichnis	61
7. Ehrlichkeitserklärung	62
i. Anhang	63
Resultate Evaluations Durchläufe	63
IP5 Gurobi Logfile Clustering DSU	65

ILP2 DSN Logfile Beweis best	67
Test GA Parallel	69

1. Einleitung

Die Jungwacht Niederwil sorgt jedes Jahr dafür, dass der Samichlaus in Niederwil AG und Nesselrbach AG zahlreiche Familien zu Hause besucht. Die Familien können sich entweder auf der Webseite oder via schriftlichem Formular anmelden. Bei der Anmeldung können Terminwünsche geäußert werden. Nach Anmeldeschluss treffen sich die Organisatoren der Chlausgesellschaft und planen einen Abend lang den Ablauf. Dabei werden alle Anmeldungen nach bestem Wissen den Chläusen zugeteilt. Die Chläuse erstellen dann eine individuelle Routenplanung. Diese Aufteilung und Routenplanung werden aber zunehmend komplizierter. Gründe dafür sind die steigenden Anmeldungen und die zunehmend fehlende Flexibilität der Familien. Ein weiteres Problem ist, dass die Routenplanung den Chläusen viel Wissen und Erfahrung abverlangt.

In einem Vorgängerprojekt wurde bereits eine Applikation entwickelt, die solche Routen für die Chläuse erstellen kann. Bei diesem Vorgängerprojekt kam ein Algorithmus zum Einsatz, der auf ganzzahliger linearer Programmierung basiert. Im Rahmen der vorliegenden Bachelorarbeit sollen weitere Ansätze entwickelt werden. Diese Ansätze sollen dann miteinander verglichen werden, um am Schluss mindestens einen Algorithmus in die bestehende Webapplikation zu integrieren. Das Ziel ist, dass am Ende des Projekts das Optimierungsproblem für den Kunden gelöst wird.

Das beschriebene Problem ist in der Fachliteratur unter dem Namen Vehicle Routing Problem with Time Windows (VRPTW) bekannt. Das Vehicle Routing Problem with Time Windows ist eine Erweiterung des Vehicle Routing Problem (VRP). Einfach gesagt befasst sich das VRP mit einer Flotte von Fahrzeugen, welche Güter aus einem zentralen Depot zu Kunden liefern soll. Bei der Erweiterung mit Time Windows besteht die Einschränkung darin, dass Kunden nur in einem bestimmten Zeitfenster besucht werden dürfen. Bei unserer Problemstellung werden die Kunden durch Familien, welche von einem Chlaus besucht werden, repräsentiert. Dementsprechend wird die Fahrzeugflotte durch die Chläuse repräsentiert. Das VRP sowie das VRPTW gehören zur Klasse der NP-schweren Probleme.

<https://www.sciencedirect.com/science/article/pii/S1018364710000297>

Erstmals erwähnt wurde das Vehicle Routing Problem im Jahre 1959. Damals wurde das Problem von Dantzig und Ramser als “The truck dispatching problem” beschrieben. Sie haben ein lineares Optimierungsverfahren entwickelt, welches auch von Hand durchgeführt werden kann. Dieser Ansatz ist heute unter dem Namen “savings algorithm” bekannt. Dieser Algorithmus gehört zur Kategorie der Greedy-Algorithmen.

(<https://andresjaquep.files.wordpress.com/2008/10/2627477-clasico-dantzig.pdf>, G. Dantzig and J. Ramser. The truck dispatching problem. Management Science, 6:80–91, 1959.)

Christophides und Elion haben ein Verfahren entwickelt, welches aus dem VRP ein Traveling Salesman Problem (TSP) macht. Der von ihnen präsentierte Ansatz löst das VRP nicht optimal, ist aber eine gute Annäherung. Der Ansatz von Christophides und Elion wandelt das

VRP in ein TSP um, indem das Depot ein Mal pro Fahrzeug als Kunde eingetragen wird. Die Distanz zwischen den Depots wird auf unendlich gesetzt.

<https://link.springer.com/article/10.1057/jors.1969.75>

Viele betriebliche Entscheidungsprobleme können als exakte Optimierungsprobleme formuliert werden. Dies trifft auch auf unser VRPTW zu. In der Praxis kommen für betriebliche Entscheidungsprobleme jedoch häufig heuristische Verfahren zur Anwendung. Der Grund dafür ist, dass exakte Formulierungen häufig zu komplex sind um innert nützlicher Frist gelöst zu werden. Bedingt durch die heuristischen Verfahren ist es oft nicht möglich, das Problem exakt zu lösen. Nichts desto trotz hat man die hohe Leistungsfähigkeit guter Heuristiken beim Lösen des VRPTW erkannt. Als Beispiele werden Simulated Annealing, Tabu Search und genetische Algorithmen genannt.

<https://www.sciencedirect.com/science/article/pii/S1018364710000297>

Im kommerziellen Bereich gibt es verschiedene Anbieter, die Werkzeuge anbieten, um verschiedenste Routen- und Zeitplanungsprobleme zu lösen. Dabei kommen verschiedenste Technologien zum Einsatz, die oftmals miteinander kombiniert werden. Bei Quintiq kommt beispielsweise ein Hybridverfahren zum Einsatz, dabei wird unter anderem Neighborhood-Search, Constraint Programming und Mixed Integer Programming verwendet. IBM bietet mit dem CPLEX CP Optimizer ein Tool welches auf Constraint Programming basiert. Der VSR Optimizer, welcher im SAP Modul Transportation Management enthalten ist, benutzt einen erweiterten "Local Search"-Ansatz.

<http://www.quintiq.de/optimierung/weltrekorde-vrptw.html>

<https://www.ibm.com/analytics/cplex-cp-optimizer>

<https://blogs.sap.com/2014/03/21/effective-optimization-run-time/>

Das VRPTW ist auch heute noch Gegenstand der Forschung. Bei der Suche nach Vergleichswerten findet man den Solomon Benchmark. Dieser Benchmark besteht aus Probleminstanzen und den besten bekannten Lösungen. Wobei ein hierarchisches Ziel verfolgt wird. Erstens soll die Anzahl benötigter Fahrzeuge minimiert werden. Zweitens soll die totale Wegdistanz minimiert werden. Die Probleminstanzen umfassen 25, 50 oder 100 Kunden. Als Erweiterung gibt es noch den Gehring & Homberger Benchmark, welcher Problemgrößen von 200 - 1000 Kunden behandelt. An der Probleminstanz mit 1000 Kunden kann man gut erkennen, dass aktuell noch am VRPTW geforscht wird, denn die Mehrheit der Lösungen sind aus dem Jahr 2018.

<https://www.sintef.no/projectweb/top/vrptw/solomon-benchmark/>

2. Problemdefinition

Das Optimierungsproblem, welches die Chlausgesellschaft lösen will, unterscheidet sich in mehreren Punkten vom klassischen VRPTW.

Allem voran, sind die Zeitfenster komplexer. Das ursprüngliche VRPTW kennt nur ein Zeitfenster pro Kunde, welches zwingend eingehalten werden muss. Das heisst, wenn das Fahrzeug vor diesem Zeitpunkt beim Kunden eintrifft, muss bis zum Start des Zeitfensters gewartet werden. Trifft das Fahrzeug zu spät ein, so kann der Besuch nicht mehr abgehandelt werden. In unserem Fall darf der Kunde vor und nach dem Zeitfenster besucht werden, wobei dann Mehrkosten entstehen.

Ein weiterer Unterschied ist, dass die Kunden in unserer Problemstellung Zeitfenster angeben können, in welchen sie besucht werden möchten. Wenn Besuche in diesem gewünschten Zeitfenster stattfinden, wird ein Bonus zur Zielfunktion hinzugefügt. Eine zusätzliche Besonderheit ist, dass es mehrere Zeitfenster geben darf, sowohl bei der Nichtverfügbarkeit wie auch bei der Wunschzeit.

Im Gegensatz zum verbreiteten VRPTW dürfen die Chläuse auch Pausen machen. Diese Pausen müssen an einer bestimmten Adresse gemacht werden und sind fest einem Chlaus zugeordnet. Eine Pause muss vom Chlaus an jedem Tag gemacht werden, sofern der Chlaus an dem Tag mindestens eine Familie besucht. Die Pausen haben ebenfalls eine Wunschzeit, in der der Chlaus diese machen will. Auch hier gibt es einen Bonus für das Einhalten dieses Zeitfensters.

Letztlich unterscheidet sich auch die Zielfunktion. In der Literatur [REF](#) wird oftmals die Anzahl Fahrzeuge und die Wegdauer minimiert. In unserer Zielfunktion wird zwar auch die Wegzeit minimiert, jedoch sollen die Routen auch ähnlich lang sein. Dies wird erreicht, indem der längste Tag proportional zur Dauer bestraft wird.

Die zu minimierende Zielfunktion besteht aus der Summe folgender Ausdrücke.

CHF $560 \times$ Anzahl nicht besuchter Familien / Pausen

CHF $400 \times$ Anzahl zusätzliche Chläuse

CHF $40 \times$ Arbeitszeit der zusätzlichen Chläuse [h]

CHF $120 \times$ Besuchszeit ausserhalb der verfügbaren Zeit [h]

CHF $120 \times$ Wegzeit ausserhalb der verfügbaren Zeit [h]

CHF $-20 \times$ Besuchszeit innerhalb der gewünschten Zeit [h]

CHF $40 \times$ Arbeitszeit [h]

CHF $30 \times$ Dauer des längsten Arbeitstages [h]

Die Konstanten wurden zusammen mit dem Kunden definiert und repräsentieren die tatsächlichen Kosten.

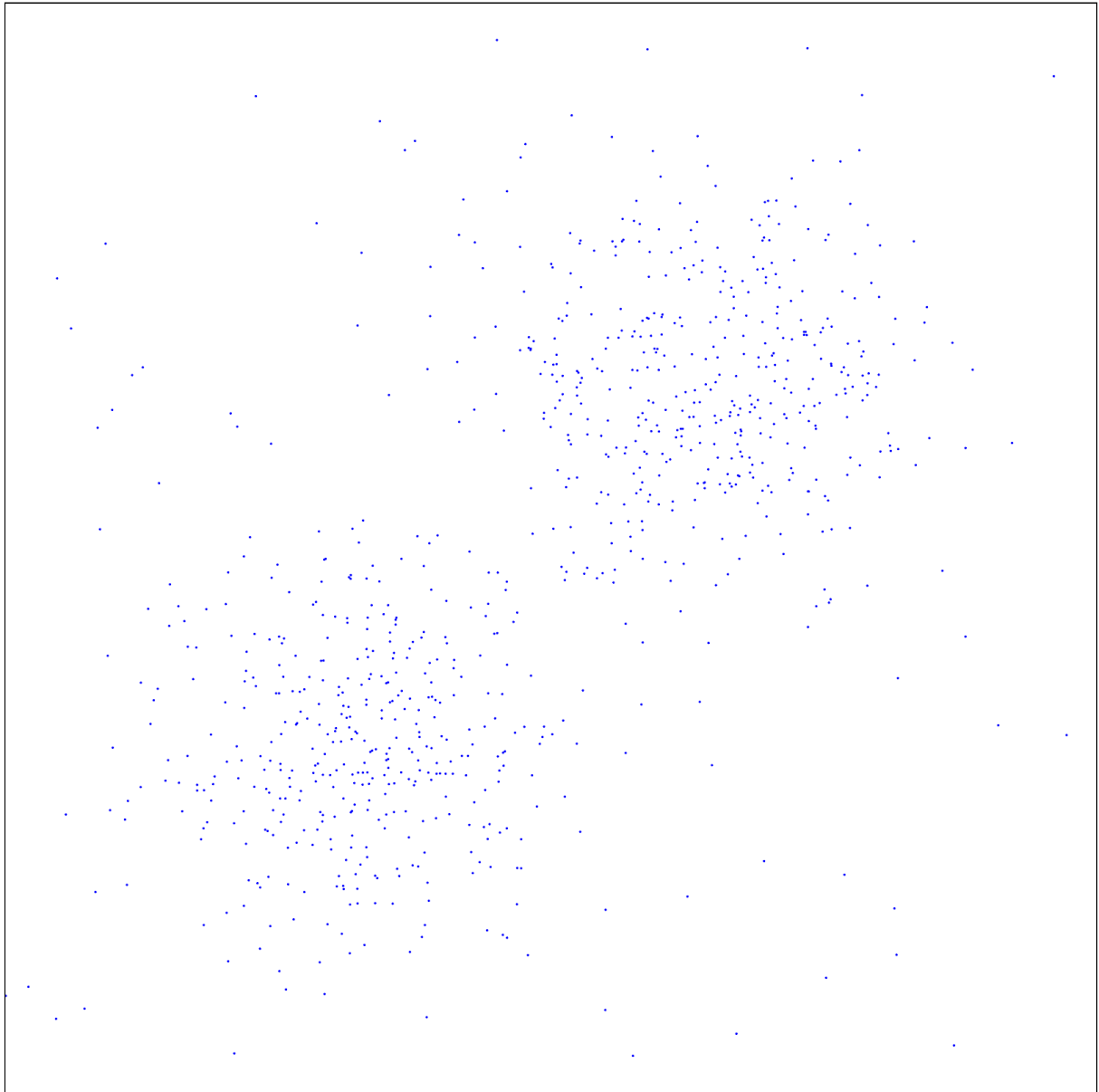
Um die Flexibilität der Lösungsansätze zu gewährleisten, benötigt man mehrere Datensätze. Es liegen nur zwei reale Beispiele vor, weshalb weitere künstliche Probleminstanzen generiert werden. Jeder Datensatz wird nach demselben Schema generiert. Die Koordinaten der Besuche werden in einem Feld der Grösse 4000×4000 verteilt. Dies stellt ein Gebiet von 16 Quadratkilometern dar. 45% der Punkte werden aus der Normalverteilung generiert, die den Mittelwert $(\frac{1}{3} \times 4000, \frac{1}{3} \times 4000)$ und die Standardabweichung 400 hat. Weitere 45% der Punkte werden aus der Normalverteilung generiert, die den Mittelwert $(\frac{2}{3} \times 4000, \frac{2}{3} \times 4000)$

und die Standardabweichung 400 hat. Die letzten 10% werden Anhand einer Gleichverteilung zufällig im Feld verteilt. Die Wegzeiten entsprechen den euklidischen Distanzen zwischen zwei Koordinaten.

Besuche werden in Familienbesuche und Pausen unterteilt. Die Wunschzeiten der Familienbesuche werden zufällig innerhalb eines einzigen Arbeitstages eingetragen. Die Dauer der Wunschzeit beträgt die Besuchsdauer plus ein zufälliger Anteil des Arbeitstages. Dasselbe gilt für die Nichtverfügbarkeit, wobei sichergestellt wird, dass sich Wunschzeit und Nichtverfügbarkeit nicht am selben Tag stattfinden.

Die Wunschzeiten der Pausen werden wie bei den Familien erstellt, mit dem Unterschied, dass an jedem Arbeitstag eine Wunschzeit generiert wird.

In Abbildung (REF) ist ein generierter Datensatz mit 1000 Besuchen ersichtlich.



BILDUNTERSCHRIFT: Beispiel generierter Datensatz mit 1000 Besuchen

In der Tabelle REF sind die Spezifikationen aufgelistet, nach welchen die Datensätze generiert werden.

Datensatz	Besuche	Pausen	Chläuse	Tage	Wunschzeiten	Nichtverfügbarkeiten
Datensatz 1	10	0	1	2	0	0
Datensatz 2	10	0	1	2	10	0
Datensatz 3	10	0	1	2	0	10
Datensatz 4	20	0	2	2	0	0
Datensatz 5	16	2	2	2	16	0
Datensatz 6	20	0	2	2	0	20
Datensatz 7	34	0	3	2	23	15
Datensatz 8	32	1	3	2	31	24
Datensatz 9	40	5	5	2	30	22
Datensatz 10	80	10	10	2	70	40
Datensatz 11	160	20	20	2	150	80
Datensatz 12	800	100	100	2	600	300

Tabelle REF

Die ersten drei Datensätze sind rein synthetische Datensätze, welche die verschiedenen Auswirkungen von Wunschzeiten und Nichtverfügbarkeiten ausweisen sollen. Die nächsten drei sind eine hochskalierte Version der ersten drei, wobei beim Datensatz 5 bereits Pausen eingebaut wurden. Als Datensatz 7 und 8 werden die realen Daten der Jahre 2017 und 2018 verwendet. Die Datensätze 9 bis 12 sind eine hochskalierte Variante der realen Daten. Der letzte Datensatz, Datensatz 12, stellt die Obergrenze mit 1000 Besuchen dar, um Limiten zu testen. Diese Obergrenze ist so gewählt, weil die grösste Probleminstanz des Gehring & Homberg Benchmarks REF ebenfalls 1000 Kunden hat. Es werden bei jedem Datensatz zwei Arbeitstage festgelegt. Dies resultiert aus der Situation der Chlausengesellschaft, welche ebenfalls zwei Arbeitstage verwendet um alle Familien zu besuchen.

3. Lösungsansätze (Kapitel)

Das in Kapitel 4 **REF** definierte Problemstellung kann auf verschiedene Arten gelöst werden. Im Rahmen dieses Projekt werden fünf mögliche Ansätze betrachtet, welche in diesem Kapitel beschrieben werden. Zuerst soll erläutert werden warum diese fünf Ansätze gewählt wurden. Der IP5-Ansatz stammt aus einem Vorgängerprojekt, welches sich mit derselben Problemstellung befasst hat. Das IP5 stellt also eine erste Implementation dar und damit einen Referenzwert. Der zweite Ansatz mit der Bezeichnung ILP basiert auf ganzzahlig linearer Programmierung und ist aus den Rückmeldungen zum IP5 entstanden. Der genetische Algorithmus ist ein weiterer Ansatz der evaluiert werden soll. Gemäss **REF** eignen sich genetische Algorithmen gut für kombinatorische Problem insbesondere dann, wenn der Suchraum viele lokale Extremalstellen aufweist. Diese beiden Eigenschaften besitzt das VRPTW. **LocalSolver**. Google Routing ist der letzte Ansatz, welcher auf den Google OR-Tools basiert. Dabei kommt die Routing Bibliothek zum Einsatz die unter anderem mit Constraint-Programmierung implementiert ist. Dieser letzte Ansatz soll die Vielfalt der verwendeten Technologien vergrössern. Es ist zu bemerken, dass, wie in Kapitel **REF** erwähnt, mehrere kommerzielle Ansätze Constraint-Programmierung benutzen.

Übersicht

Bezug nehmen auf Problemstellung im stile von “Die vorhergehende Problemstellung wird mit verschiedenen Ansätzen gelöst”

Weshalb haben wir diese Lösungsansätze ausgewählt

ILP2: Umsetzung Feedback IP5

GA: gut für kombinatorische Probleme.

Genetischen Algorithmen eignen sich insbesondere für Optimierungsprobleme, deren Suchraum nicht nur globale sondern auch viele lokale Extremalstellen aufweist. Bei konventionellen Optimierungsmethoden besteht immer die Gefahr, eine lokale Extremalstelle fälschlicherweise als Lösung des Problems aufzufassen. Bei GA's reduziert sich diese Gefahr wesentlich, weil der Suchraum immer gleichzeitig an verschiedenen Stellen untersucht wird, womit die Wahrscheinlichkeit steigt, die globale Extremalstelle auch tatsächlich zu finden.

REF

LocalSolver: Vorschlag von Herr Felix

Google Routing: Weil viele kommerzielle Produkte Constraint Programming brauchen und weil wir so eine gute Vielfalt an Verfahren haben.

3.1 IP5 - Integer Linear Programming

In dem Vorgängerprojekt wurde ein Ansatz mit ganzzahliger linearer Programmierung erstellt. Dieser Ansatz besteht im wesentlichen aus zwei Phasen, dem Clustering und dem Scheduling. Im Clustering wird das VRP mit einigen Zusätzen gelöst. Die Verteilung der Besuche auf die Chläuse geschieht mit dem Ziel, die Wegzeiten möglichst gering zu halten. Zudem werden Besuche nicht an einem Tag stattfinden, an dem sie ein Nichtverfügbarkeits-Zeitfenster haben.

Die Pausen werden fix den Routen zugewiesen. Das Subtour-Problem des ATSP wurde mit einer Flow-Formulierung gelöst, indem beim Startpunkt eine Quelle ist und jeder Besuch ein Abfluss, bei welchem mindestens 1 Flow vorhanden sein muss.

In der Zielfunktion kommen die Komponenten Arbeitszeit und längste Route vor ([REF](#)):

$$\text{minimize } 40 \times \text{totalWorkingTime} + 30 \times \text{longestRoute} \quad (\text{X})$$

Die Erzeugnisse der ersten Phasen sind vollwertige Routen. Die Routen sagen aus, welcher Chlaus an welchem Tag welche Familien besucht und welche Pausen macht. Dazu kommt, dass die Familien und Pausen eine bestimmte Reihenfolge haben, sodass die Wege und der längste Tag möglichst kurz sind.

Pro berechneter Route wird dann der zweite Teil, das Scheduling angewendet. Das Scheduling, erhält das Resultat aus dem Clustering und versucht die Besuche möglichst optimal anzuordnen, sodass die ursprüngliche Zielfunktion optimiert wird. Denn in der ersten Phase wurden die Wunschzeiten nicht beachtet. Es muss also nur noch die Wunschzeit und Wegzeit beachtet werden, da keine Zeiten ausserhalb der verfügbaren Zeit entstehen können. Die Formulierung baut darauf auf, dass die Zeit gerastert dargestellt wird. Dadurch ergibt sich ein Würfel binärer Variablen mit den Dimensionen Chlaus, Besuch und Zeit, wobei eine 1 heisst, dass der Chlaus diesen Besuch an dem Zeitpunkt besucht und 0 bedeutet, dass der Chlaus zu dem Zeitpunkt diesen Besuch nicht besucht. Durch das Rastern der Zeit hat man einen Parameter, mit welchem man die maximale Genauigkeit einstellen kann.

WEITERSCHREIBEN

Die Zielfunktion des Scheduling lautet:

$$- 20 \times \text{desiredduration} + 40 * \text{waytime} \quad \text{ZIELFUNKTION BESCHREIBEN mit richtigen Variablen}$$

Das Gesamtergebnis des Scheduling ist wieder eine komplette Routenplanung, wobei jetzt die Zeitfenster berücksichtigt werden und somit bessere Lösungen möglich sind. Sollten die Routen vom Scheduling nicht innerhalb des Zeitlimits berechenbar sein, werden die Routen vom Clustering als Schlussresultat zurückgegeben.

Das beschriebene Modell wurde ursprünglich für SCIP implementiert. SCIP ist ein Programm, um ganzzahlig lineare Probleme zu lösen. Im Rahmen dieses Projekt soll aber auch getestet werden, wie gut das Modell mit Gurobi gelöst werden kann. Gurobi ist eine Software für mathematische Optimierung, die unter anderem ganzzahlig lineare Probleme lösen kann. In den folgenden Kapiteln wird das IP5 Modell mit SCIP als “IP5 SCIP” bezeichnet. Die, in diesem Projekt entwickelte, Portierung des IP5 Modells auf Gurobi wird als “IP5 Gurobi” bezeichnet.

Kurz beschreiben, damit der Leser draus kommt.

Damit es unsere Aussagen unterstützt

3.2 ILP - Ein-Modell Formulierung

TODO: Formelnummerierung

Aufgrund der Rückmeldungen betreffend dem IP5 entschieden wir uns, ein weiteres Modell zu formulieren, welches mit ganzzahliger linearer Optimierung gelöst werden kann. Auch dieses Modell hat zwei Phasen. In der ersten Phase wird das VRP gelöst. Die Lösung dieser Phase wird dann als MIP Start in der zweiten Phase verwendet. Im Gegensatz zum IP5 beinhaltet die zweite Phase ein komplett formuliertes Modell. Dies bringt den Vorteil, dass die Problemstellung bis zum Optimum optimiert werden kann.

Durch das Einführen der ersten Phase, muss die zur Verfügung stehende Zeit aufgeteilt werden. Den optimalen Parameter haben wir mit einem Grid-Search gefunden. Für den Grid-Search wurden 15 Datensätze mit 20 Besuchen, 2 Pausen, 2 Chläusen und 2 Tagen verwendet. Dies entspricht einer skalierten Version der Evaluations-Datensätze REF. Die Resultate wurden pro Durchlauf aufsummiert. Der Grid Search wurde dreimal laufen gelassen und dann den Mittelwert pro Durchlauf angeschaut.

Zeitlimit Phase 1										
0	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
13536	12689	12698	12693	12693	12693	12692	12692	12696	12696	12977

GRID SEARCH RESULT

Aus der Tabelle REF ist zu erkennen, dass die optimale Zeiteinteilung bei dem Verhältnis 10% für die erste Phase und 90% für die zweite Phase liegt. Dieser Wert liegt sehr nahe am schlechtesten gefundenen Wert. Es wird das Resultat für zwei weitere Werte, 5% und 15% berechnet um zu zeigen, dass der Wert 10% geeignet ist.

Zeitlimit Phase 1				
0%	5%	10%	15%	20%
13536	12689	12689	12687	12698

GRID SEARCH RESULT 2

Aus Tabelle REF ist ersichtlich, dass 15% noch besser sind. Der Unterschied ist jedoch sehr klein und 20% sind schlechter als 10%. Dass 0% das schlechteste ist, lässt sich dadurch erklären, dass der MIP Start bei der zweiten Phase durch triviales Aufteilen gemacht wird. Bereits ein leicht optimierter Start hilft der zweiten Phase wesentlich bessere Resultate zu berechnen.

In der ersten Phase wird das Vehicle Routing Problem als ganzzahliges lineares Programm formuliert. Die verwendete Formulierung basiert auf einer Travelling Salesman Formulierung von Gurobi. [<http://examples.gurobi.com/traveling-salesman-problem/>]

Angepasst an die Zielfunktion ist das zu minimierende Ziel bei diesem Modell die totale Wegzeit und die längste Route (1). In der folgenden Formulierung wird die Variable V als Menge aller Besuche ohne den Startpunkt verwendet. Die Anzahl der Besuche wird mit $n_v = |V|$ bezeichnet. Analog zu den Besuchen ist die Menge aller Chläuse als S und die Anzahl

aller Chläuse als n_s bezeichnet. Die Anzahl der Chläuse n_s entspricht den anzahl Tagen multipliziert mit den anzahl Chläusen. Die Pause b_s für Chlaus s in der Menge $B_s \subseteq V$ vorhanden. Um die Wege zwischen den Besuchen abzubilden wird die Variable $x(s, i, j) \in \{0, 1\} \forall s \in S, \forall i, j \in V$ verwendet. Sie nimmt den Wert 1 an, falls der Chlaus s den Weg von Besuch i nach j benutzt. Ob ein Chlaus eine Familie besucht wird mit der Variable $v(s, i) \in \{0, 1\} \forall s \in S, \forall i \in V$ angegeben. Sie nimmt den Wert 1 an falls der Chlaus s die Familie i besucht. Zudem sind die Wege von und zum Startpunkt als $w(s, 0, i) \in \{0, 1\} \forall s \in S, \forall i \in V$ respektive $w(s, i, 0) \in \{0, 1\} \forall s \in S, \forall i \in V$ definiert. Analog dazu ist $v(s, 0) \in \{0, 1\} \forall s \in S$ die Variabel ob ein Chlaus den Startpunkt besucht. Die Funktion $distance(i, j)$ gibt die Distanz in Sekunden zwischen Besuch i und j zurück. Die Funktion $duration(i)$ gibt die Besuchsdauer für Besuch i zurück.

$$\text{minimize } 4 \times \left(\sum_{s=1}^{n_s} \sum_{i=1}^{n_v} \sum_{j=1}^{n_v} w(s, i, j) \times distance(i, j) + \sum_{s=1}^{n_s} \sum_{i=1}^{n_v} v(s, i) \times duration(i) \right) + 3 \times longestRoute \quad (1)$$

$$\text{s.t. } longestRoute \geq \sum_{i=1}^{n_v} \sum_{j=1}^{n_v} w(s, i, j) \times distance(i, j) + \sum_{i=1}^{n_v} v(s, i) \times duration(i) \quad \forall s \in S \quad (2)$$

$$w(s, i, i) = 0 \quad \forall s \in S, \forall i \in V \quad (3)$$

$$\sum_{s=1}^{n_s} v(s, i) = 1 \quad \forall i \in V \setminus B \quad (4)$$

$$\sum_{s=1}^{n_s} \sum_{j=1}^{n_v} w(s, i, j) = \sum_{s=1}^{n_s} \sum_{j=1}^{n_v} w(s, j, i) = 1 \quad \forall i \in V \quad \text{3Fach gleichung erlaubt?} \quad (5)$$

$$\sum_{j=1}^{n_v} w(s, i, j) = \sum_{j=1}^{n_v} w(s, j, i) = v(s, i) \quad \forall s \in S, \forall i \in V \quad (6)$$

$$\sum_{j=1}^{n_v} w(s, 0, j) = \sum_{j=1}^{n_v} w(s, j, 0) \quad \forall s \in S \quad (7)$$

$$\sum_{j=1}^{n_v} w(s, 0, j) \geq v(s, i) \quad \forall i \in S \quad (8)$$

$$\sum_{j=1}^{n_v} w(s, 0, j) \leq \sum_{i=1}^{n_v} v(s, i) \quad \forall s \in S \quad (9)$$

$$\sum_{i=0}^{n_v} \sum_{j=0}^{n_v} w(s, i, j) = \sum_{i=0}^{n_v} v(s, i) \quad \forall s \in S \quad (10)$$

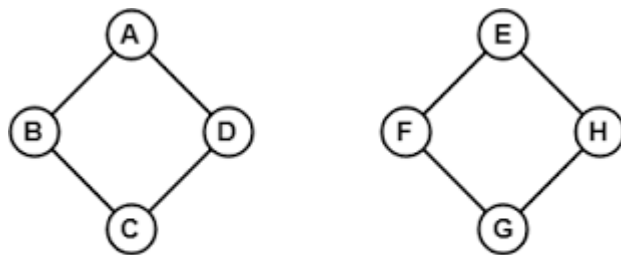
$$v(s, b_s) \leq \sum_{j=0}^{n_v} v(s, j) \quad \forall s \in S \quad (11)$$

$$v(s, b_s) \geq v(s, i) \quad \forall s \in S, \forall i \in V \quad (12)$$

$$v(s_2, b_{s_1}) = 0 \forall s_1, s_2 \in S, s_1 \neq s_2 \quad (13)$$

$$\sum_{i,j \in T} w(s, i, j) \leq |T| - 1 \quad \forall s \in S, \forall T \subset \{x | x \in V \wedge v(s, x) = 1\}, T \neq \emptyset \quad \text{so nicht} \quad (14)$$

Damit eine valide Lösung entsteht müssen diverse Einschränkungen gelten. So darf ein Chlaus nicht von der Familie i zu der Familie i gehen (3). Jede Familie muss von genau einmal besucht werden (4). Zudem muss er bei jeder Familie die er besucht einmal zu der Familie kommen und einmal von der Familie weggehen (5). Um die Formulierung einzuengen wird die Einschränkung, dass alle eingehenden Wege für einen Besuch über alle Chläuse gesehen auch den ausgehenden Wegen entsprechen muss eingeführt (6) {{Mit Gurobi zeigen dass das etwas hilft}}. Die eingehenden und ausgehenden Wege müssen auch für den Startpunkt gleich sein (7). Jeder Chlaus muss vom Startpunkt aus gehen sofern er mindestens einen Besuch besucht (8) (9). Zudem muss er gleich viele Wege gehen wie er Familien besucht (10). Besucht ein Chlaus eine Familie, so muss er seine Pause ebenfalls machen (11) (12). Kein Chlaus darf die Pause eines anderen Chlaus machen (13). Mit nur diesen Einschränkungen gibt es Lösungen, welche Subtouren enthalten. Eine Subtour ist eine Route, welche für einen Chlaus berechnet wurde, in der nicht alle Besuche zusammenhängen. In Abbildung {REF} ist ein Beispiel einer solchen Subtour dargestellt.



Disconnected Graph

BILD SUBTOUR SELBST MACHEN

Dieses Subtouren Problem wird dadurch gelöst, indem all diese Subtouren verboten werden (14). Weil das berechnen aller möglichen Subtouren ein zu grosser Aufwand ist, wird nach jeder Lösung überprüft, ob eine Subtour vorkommt. Ist dies der Fall, so wird diese Subtour für alle Chläuse ausgeschlossen. Sobald die optimale Lösung gefunden wurde, welche keine Subtouren beinhaltet, ist die Phase 1 beendet.

Phase 2

Die zweite Phase löst das Optimierungsproblem vollständig. Bei dieser Formulierung werden deutlich mehr Variablen und Einschränkungen benötigt. Analog zur ersten Phase wird $V \subseteq \mathbb{N}$ als Menge aller Besuche ohne Startpunkt, $n_v = |V|$ als Anzahl der Besuche, $B_s \subseteq V$ als Menge aller Pausen für Chlaus s , $S \subseteq \mathbb{N}$ als Menge aller Chläuse, $n_s = |S|$ als Anzahl aller Chläuse, $v(s, i) \in \{0, 1\}$ als Indikator, ob Chlaus s die Familie i besucht und $w(s, i, j)$ als Indikator, ob Chlaus s den Weg von Besuch i nach Besuch j benutzt, verwendet. Die Anzahl der Chläuse entspricht erneut den anzahl Tagen multipliziert mit den zur verfügung stehenden Chläusen. Hinzu kommt die Variable $c(s, i) \in \mathbb{R}$ welche angibt, zu welchem Zeitpunkt der Chlaus s den Besuch i besucht. Dieser Zeitpunkt wird zum Tagesstart der Route dazugerechnet um den effektiven Zeitpunkt zu erhalten. Die Startzeit in Sekunden des Arbeitstages von Chlaus s wird über die Funktion $p(s, 0)$ zurückgegeben. Das Ende des Arbeitstages wird von der

Funktion $p(s, 1)$ bestimmt. Bei diesem Modell sind die Nichtverfügbarkeiten in der Menge U enthalten, die i -te Nichtverfügbarkeit für Besuch v wird als $u_v(i) \in U$ behandelt, wobei $u(v, i, 0) \in \mathbb{R}$ den Startzeitpunkt der Nichtverfügbarkeit und $u(v, i, 1) \in \mathbb{R}$ das Ende ist. Zur vereinfachten Schreibweise wird $n_u(v)$ als Anzahl aller Nichtverfügbarkeiten von Besuch v verwendet. Nach demselben Prinzip werden Wunschzeiten mit D , $d(v, i) \in D$ sowie $n_d(v)$ verwendet. Die Überschneidung zwischen der u -ten Nichtverfügbarkeit des Besuches i und der tatsächlichen Besuchszeit von Chlaus s wird mit der Variable $unavailableDuration(s, i, u) \in \mathbb{R}$ dargestellt. Analog dazu stellt die Variable $desiredDuration(s, i, d)$ die Überschneidung der d -ten Wunschzeit von Besuch i und der tatsächlichen Besuchszeit von Chlaus s dar. Weiter werden die Hilfsvariablen $minRoute(s) \in \mathbb{R} \forall s \in S$ für den Startzeitpunkt der Route sowie $maxRoute(s) \in \mathbb{R} \forall s \in S$ für den Rückkehr Zeitpunkt des Chlauses s eingeführt. Weitere Hilfsvariablen sind $desiredStart(s, i, d) \in \mathbb{R}$, $desiredEnd(s, i, d) \in \mathbb{R}$, $desiredOverlap(s, i, d) \in \{0, 1\}$, $unavailableStart(s, i, d) \in \mathbb{R}$, $unavailableEnd(s, i, d) \in \mathbb{R}$, $bUnavailableStart(s, i, d) \in \{0, 1\}$, $bUnavailableEnd(s, i, d) \in \{0, 1\}$

minimize

$$\begin{aligned}
& 12 \times \sum_{s=1}^{n_s} \sum_{i=1}^{n_v} \sum_{u=1}^{n_u(v)} unavailableDuration(s, i, u) + \\
& 4 \times \sum_{s=1}^{n_s} (maxRoute(s) - minRoute(s)) + \\
& - 2 \times \sum_{s=1}^{n_s} \sum_{i=1}^{n_v} \sum_{d=1}^{n_d(v)} desiredDuration(s, i, d) + \\
& 3 \times longestRoute
\end{aligned} \tag{1}$$

$$s.t. \text{ } longestRoute \geq maxRoute(s) - minRoute(s) \forall s \in S \tag{2}$$

$$w(s, i, i) = 0 \forall s \in S, \forall i \in V \tag{3}$$

$$\sum_{s=1}^{n_s} v(s, i) = 1 \forall i \in V \setminus B \tag{4}$$

$$\sum_{s=1}^{n_s} \sum_{j=1}^{n_v} w(s, i, j) = \sum_{s=1}^{n_s} \sum_{j=1}^{n_v} w(s, j, i) = 1 \forall i \in V \text{ incoming} = \text{outgoing glob} \tag{5}$$

$$\sum_{j=1}^{n_v} w(s, i, j) = \sum_{j=1}^{n_v} w(s, j, i) = v(s, i) \forall s \in S, \forall i \in V \text{ incoming} = \text{outgoing per santa} \tag{6}$$

$$\sum_{j=1}^{n_v} w(s, 0, j) = \sum_{j=1}^{n_v} w(s, j, 0) \forall s \in S \text{ incoming} = \text{outgoing start visit} \tag{7}$$

$$\sum_{j=1}^{n_v} w(s, 0, j) \geq v(s, i) \forall i \in S \text{ santa has to use home OR} \tag{8}$$

$$\sum_{j=1}^{n_v} w(s, 0, j) \leq \sum_{i=1}^{n_v} v(s, i) \forall s \in S \text{ santa has to use home OR} \tag{9}$$

$$\sum_{i=0}^{n_v} \sum_{j=0}^{n_v} w(s, i, j) = \sum_{i=0}^{n_v} \sum_{j=0}^{n_v} w(s, j, i) = \sum_{i=0}^{n_v} v(s, i) \quad \forall s \in S \text{ ways} = \text{visits} \quad (10)$$

$$v(s, b_s) \leq \sum_{j=0}^{n_v} v(s, j) \quad \forall s \in S \text{ break has to be made OR} \quad (11)$$

$$v(s, b_s) \geq v(s, i), \forall s \in S, \forall i \in V \text{ break has to be made OR} \quad (12)$$

$$v(s_2, b_{s_1}) = 0 \forall s_1, s_2 \in S, s_1 \neq s_2 \text{ break not by other santa} \quad (13)$$

$$c(s, i) \leq p(s, 1) - p(s, 0) \quad \forall s \in S, \forall i \in V \quad (40)$$

$$v(s, i) = 0 \rightarrow c(s, i) = 0 \quad \forall s \in S, \forall i \in V \quad (14)$$

$$w(s, j, i) = 1 \rightarrow c(s, i) \geq c(s, j) + \text{distance}(j, i) + \text{duration}(j) \quad \forall s \in S, \forall i \in V \forall j \in V \cup \{0\} \quad (15)$$

$$\text{maxRoute}(s) \geq c(s, i) + (\text{duration}(i) + \text{distance}(i, 0)) \times v(s, i) \quad \forall s \in S, \forall i \in V \quad (36)$$

$$\text{minRoute}(s) \leq c(s, i) - \text{distance}(0, i) \times v(s, i) + (1 - v(s, i)) \times M \quad \forall s \in S, \forall i \in V \quad (37)$$

$$\text{minRoute}(s) \leq \text{maxRoute}(s) \quad \forall s \in S \quad (38)$$

$$\text{desiredDuration}(s, i, k) \leq \text{Min}(\text{duration}(i), d(i, k, 1) - d(i, k, 0)) \quad \forall s \in S, \forall i \in V \quad \forall k \in L \quad (39)$$

$$\text{für Gleichung 16 - 24 gilt } \forall s \in S, \forall i \in V, \forall k \in D_i \quad (16)$$

$$\text{desiredStart}(s, i, k) \geq \max(d(i, k, 0) - p(s, 0), 0)$$

$$\text{desiredStart}(s, i, k) \leq p(s, 1) - p(s, 0) \quad (17)$$

$$\text{desiredStart}(s, i, k) \geq c(s, i) \quad (18)$$

$$\text{desiredEnd}(s, i, k) \leq d(i, k, 1) - d(i, k, 0) \quad (19)$$

$$\text{desiredEnd}(s, i, k) \leq c(s, i) + \text{duration}(i) \times v(s, i) \quad (20)$$

$$\text{desiredOverlap}(s, i, k) = 1 \rightarrow \text{desiredEnd}(s, i, k) - \text{desiredStart}(s, i, k) \geq 0 \quad (21)$$

$$\text{desiredOverlap}(s, i, k) = 1 \rightarrow \text{desiredDuration}(s, i, k) = \text{desiredEnd}(s, i, k) - \text{desiredStart}(s, i, k) \quad (22)$$

$$\text{desiredOverlap}(s, i, k) = 0 \rightarrow \text{desiredEnd}(s, i, k) - \text{desiredStart}(s, i, k) \leq 0 \quad (23)$$

$$\text{desiredOverlap}(s, i, k) = 0 \rightarrow \text{desiredDuration} = 0 \quad (24)$$

$$\text{für Gleichung 25 - 35 gilt } \forall s \in S, \forall i \in V, \forall k \in U_i \quad (25)$$

$$\text{unavailableDuration}(s, i, k) \leq \text{Min}(\text{duration}(i), u(i, k, 1) - u(i, k, 0))$$

$$\text{unavailableStart}(s, i, k) \geq u(i, k, 0) - p(s, 0) \quad (26)$$

$$\text{unavailableStart}(s, i, k) \leq p(s, 1) - p(s, 0) \quad (27)$$

$$unavailableStart(s, i, k) \geq c(s, i) \quad (28)$$

$$bUnavailableStart(s, i, k) = 0 \rightarrow unavailableStart(s, i, k) \leq u(i, k, 0) - p(s, 0) \quad (29)$$

$$bUnavailableStart(s, i, k) = 1 \rightarrow unavailableStart(s, i, k) \leq c(s, i) \quad (30)$$

$$unavailableEnd(s, i, k) \leq u(s, i, k) - p(s, 0) \quad (31)$$

$$unavailableEnd(s, i, k) \leq c(s, i) + duration(i) \quad (32)$$

$$bUnavailableEnd(s, i, k) = 0 \rightarrow unavailableEnd(s, i, k) \geq u(i, k, 1) - p(s, 0) \quad (33)$$

$$bUnavailableStart(s, i, k) = 1 \rightarrow unavailableEnd(s, i, k) \geq c(s, i) + duration(i) \quad (34)$$

$$unavailableDuration(s, i, k) \geq unavailableEnd(s, i, k) - unavailableStart(s, i, k) \quad (35)$$

Die Zielfunktion kann direkt aus der Problemstellung übernommen werden (1). Um Numerische Ungenauigkeiten zu vermeiden, werden die Konstanten mit 360 multipliziert. Die längste Route wird durch die maximale Differenz der Start- und der Endzeit aller Routen berechnet (2). Es gelten sehr Ähnliche weitere Einschränkungen wie in der Phase 1. So darf ein Chlaus den Weg von Besuch i zu Besuch i nicht beschreiten (3). Weiter muss jede Familie genau einmal besucht werden (4). Dabei muss jeder Besuch über einen Hinweg gestartet und über einen Rückweg wieder verlassen werden (5). Um eine bessere LP-Relaxation zu erreichen, wird diese Einschränkung auch pro Chlaus angewendet (6). Dasselbe gilt ebenfalls für den Startpunkt, falls ein Chlaus den Startpunkt verlässt, so muss er auch zurückkehren (7). Falls ein Chlaus eine Familie besucht, so muss er zwingend auch den Startpunkt besuchen (8)(9). Um die LP-Relaxation weiter einzuengen müssen pro Chlaus die Anzahl begangener Wege und die Anzahl der Besuche gleich sein (10). Wenn ein Chlaus eine Familie besucht und eine Pause eingetragen hat, so muss er diese zwingend machen (11)(12). Zudem darf diese Pause nur von dem eingetragenen Chlaus gemacht werden (13).

Anders als bei der Phase 1 wird das Verhindern von Subtouren über die Zeit Variabel c sichergestellt. Die Variabel c wird in Ihrem Maximum auf die Dauer des Arbeitstages beschränkt (40). Falls eine Familie nicht von dem Chlaus besucht wird, so wird auch keine Zeit eingetragen (14). Wird der Weg von Familie i zur Familie j begangen, so muss die Familie j später als die Familie i besucht werden (15). Der Mindestabstand zwischen dem Besuchsstart von Familie i und dem Besuchsstart von Familie j ergeben sich durch die Besuchszeit von Familie i sowie der Wegzeit von i nach j . Durch diese Einschränkung erhöht sich die Zeit mit jedem begangenen Weg und verhindert somit Subtouren. Abbildung REF ist eine bildliche Darstellung, dieser Einschränkung.

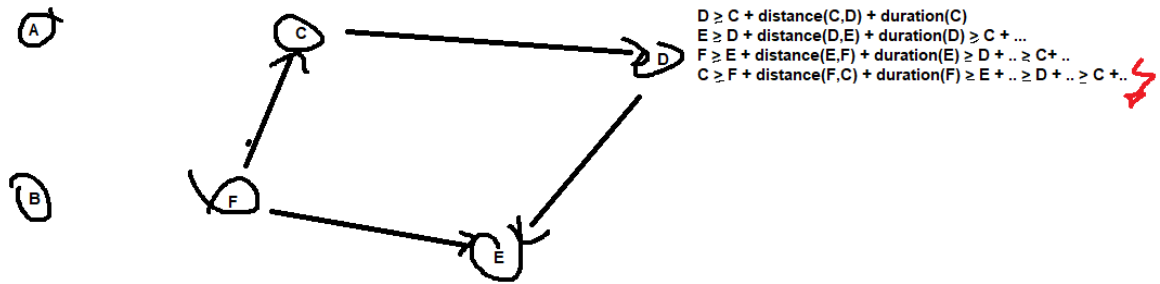


BILD Subtour increasing c impossible in schön machen

Den Rückkehr Zeitpunkt kann durch eine Maximum-Einschränkung der Zeitvariable fixiert werden. Der Rückkehr Zeitpunkt ist später als jedes Besuchsende plus den Rückweg zum Startpunkt (36). Den Startpunkt der Route kann durch eine Minimal-Einschränkung der Zeitvariable minus dem Weg vom Startpunkt zu dem Besuch beschränkt werden. Weil die Zeitvariable für eine Familie, welche der Chlaus nicht besucht den Wert 0 annimmt, muss diese Einschränkung mit einer Big-M Formulierung gelöst werden (37). Das M wird hier auf den Upper-Bound der Variable $\minRoute(s)$ gesetzt, was der Tagesdauer entspricht.

Damit die erfüllte Wunschzeit berechnet werden kann, muss die Überlappung zwischen der Wunschzeit des besuches und der tatsächlichen Besuchszeit berechnet werden. Die erfüllte Wunschzeit wird maximal so lange wie der kleiner Wert zwischen der Besuchsdauer und der Wunschzeit dauert. Der Start dieser Überlappung liegt frühestens bei dem Startpunkt der Wunschzeit(16). Er spätestens beim Tagesende (17) und frühestens beim Startpunkt der Besuchszeit (18).

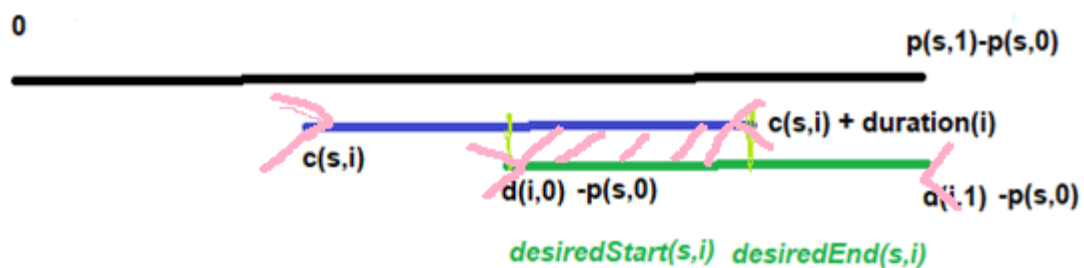


Bild in schön erstellen

Das Ende der Überlappung liegt spätestens am Tagesende (19) und spätestens Ende des Besuches. Wird die Familie nicht besucht, so wird das Ende der Überlappung auf null gesetzt(20). Ist das Ende der Überlappung später als der Anfang, so gibt es eine Überlappung (21) und die erfüllte Wunschzeit wird durch die Differenz der beiden berechnet(22). Ist das Ende vor dem Start(23), so wird die Wunschzeit nicht erfüllt (24). Die Einschränkungen beschränken die Überlappung nur in den maximal Punkten. Dadurch dass die Zielfunktion die Kosten minimiert und die Wunschzeit mit einer negativen Konstante multipliziert wird, ist die maximale mögliche Wunschzeit das Optimum, welches verwendet wird.

Für die Zeit in der Nichtverfügbarkeit kann nicht dasselbe gemacht werden, da diese Zeit mit einem positiver positiven Konstante in der Zielfunktion multipliziert wird. Die maximale Nichtverfügbarkeit pro Besuch ist durch die Besuchsdauer sowie durch die gesamt-dauer der Nichtverfügbarkeit beschränkt (25).



Bild in schön machen

Der Start der Überlappung ist frühestens beim späteren Zeitpunkt zwischen Start der Nichtverfügbarkeit und Start des Besuches. Zudem ist der Start der Überlappung früher als das Tagesende (26)(27)(28). Der Start der Überlappung muss auf den späteren Zeitpunkt fixiert werden. Hierfür wird eine binäre Entscheidungsvariable $bUnavailableStart(s,i,k)$ verwendet. Hat diese Entscheidungsvariable den Wert 0, so gilt die Einschränkung, dass der Start der Überlappung spätestens beim Start der Nichtverfügbarkeit ist (29). Dadurch wird der Wert fixiert. Hat $bUnavailableStart(s,i,k)$ den Wert 1, so gilt dass der Start der Überlappung auf den Start des Besuches fixiert wird (30). Eine der beiden Einschränkungen führt dazu, dass das Problem nicht mehr lösbar ist, wodurch leicht bestimmt wird, welchen Wert $bUnavailableStart(s,i,k)$ annehmen muss.

Ähnlich wird das Ende der Überlappung bestimmt, dieses ist spätestens am früheren Zeitpunkt zwischen Ende der Nichtverfügbarkeit und Ende des Besuches (31)(32). Mithilfe der binären Entscheidungsvariable $bUnavailableEnd(s,i,k)$ wird das Ende der Überlappung fixiert. Nimmt die $bUnavailableEnd(s,i,k)$ den Wert 0 an, so ist das Ende frühestens beim Ende der Nichtverfügbarkeit (33), bei 1 ist es frühestens am Ende des Besuches (34). Auch hier wird eine der beiden Einschränkungen dazu führen dass das Problem nicht mehr lösbar ist und dadurch die Entscheidungsvariable bestimmen. Die Dauer der Überlappung wird aus der Differenz der End und Startzeit berechnet (35).

END

3.3 Genetischer Algorithmus

erste Kapitel zitieren

Einleitung, was ist GA? (abstrakt)

Bei genetischen Algorithmen (GA) handelt es sich um Suchverfahren, welche nach dem Prinzip der biologischen Evolution auf Basis des Darwinismus funktionieren. Genetische Algorithmen gehören zu den Metaheuristiken. Metaheuristiken sind Optimierungsverfahren, die für theoretisch beliebige Probleme eingesetzt werden können. Metaheuristiken definieren dabei lediglich abstrakte Vorgehensweisen, welche dann problemspezifisch implementiert werden müssen. Laut **REF** eignen sich genetische Algorithmen vor allem für kombinatorische Probleme, also Probleme, bei denen die Anordnung von Elementen entscheiden ist.

Aufbau GA, Suchräume (abstrakt)

Um den Aufbau von genetischen Algorithmen verstehen zu können, müssen zuerst einige Begriffe eingeführt werden. Zuerst muss der Suchraum unterteilt werden. Der eigentliche Suchraum, wie er für die Chlausgesellschaft verständlich ist, wird als phänotypischer Suchraum bezeichnet. Eine einzelne Lösung ist demnach ein Phänotyp. Ein Phänotyp enthält bei unserer Problemstellung die Information, welcher Chlaus zu welchem Zeitpunkt welche Familie besucht. Demnach wird ein Phänotyp gebraucht, um die Kosten einer Planung zu berechnen. Diese Kosten geben an, wie gut die Qualität einer Lösung ist. Die Lösungsqualität wird auch Fitness genannt. Evolutionsoperatoren, welche später beschrieben werden, können in der Regel nur auf die genetischen Repräsentationen der Phänotypen angewandt werden. Diese Repräsentation wird Genotyp genannt. Mithilfe einer Dekodierungsfunktion kann ein Genotyp in einen Phänotyp umgewandelt werden.

Ablauf / Vorgehen GA (abstrakt)

Der grundsätzliche Ablauf von genetischen Algorithmen sieht vor, dass als Erstes eine Startpopulation kreiert wird. Die Individuen dieser Population sind Genotypen. Diese Startpopulation enthält meist zufällig generierte Lösungen. Danach beginnt der eigentliche Evolutionsprozess bei dem die Evolutionsoperatoren zum Einsatz kommen. Bei der Selektion werden zuerst - anhand der Fitness - Individuen bestimmt, welche sich Fortpflanzen sollen. Danach werden die Nachkommen mittels geeigneter Rekombination erzeugt. Als nächstes können die Individuen für die neue Generation noch mutiert werden. Der letzte Schritt der Evolution ist es dann, die alte Generation mit der neuen zu ersetzen. Das Ziel ist es dann, dass man in der letzten Generation eine genügend gute Lösung erhält.

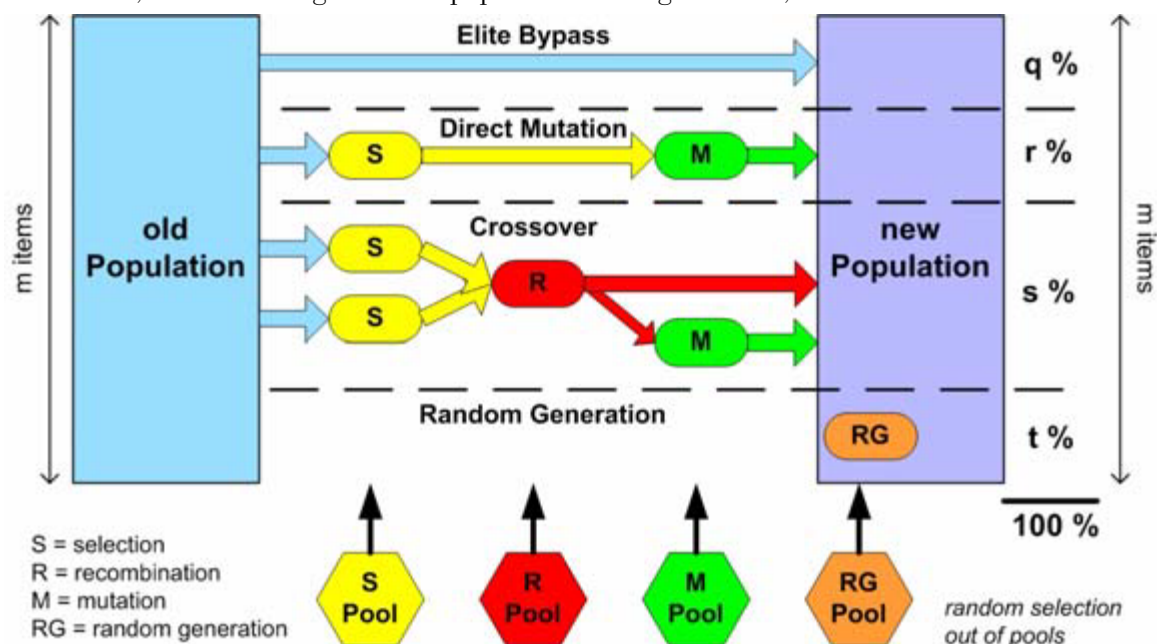
Start-Population	
	Selektion
	Rekombination
	Mutation

Abbruch-Bedingung
End-Population

bildunterschrift

Aufbau GA-Modell (halbabstrakt)

Im Rahmen dieser Arbeit wird ein erweitertes Modell eingesetzt, welches am Institut für 4D-Technologien an der FHNW zur Optimierung unterschiedlicher Probleme eingesetzt wird. Der erste Schritt des genetischen Algorithmus ist das Erzeugen der Startpopulation. Dabei wird eine bestimmte Anzahl Genotypen zufällig erzeugt. Als nächstes beginnt die eigentliche Evolution. Wie in **Abbildung X** dargestellt, können die Individuen der nächsten Population aus verschiedenen Quellen stammen. Der erste Anteil kommt über den Elitismus. Das bedeutet, dass die besten $q\%$ direkt in die nächste Generation übernommen werden. In dieser Implementation ist es wichtig, dass mindestens das beste Individuum übernommen wird, um die beste Lösung zu erhalten. Die nächsten $r\%$ werden zuerst Selektiert und dann direkt mutiert. Die nächsten $s\%$ werden durch Rekombination erzeugt und anschliessend mit einer bestimmten Wahrscheinlichkeit noch mutiert. Zum Schluss werden noch $t\%$ an zufälligen Individuen, welche analog zur Startpopulation erzeugt werden, übernommen.



Bildunterschrift, indirekt zitieren, also nachbauen

Codierung (konkret)

Um die genaue Funktionsweise des genetischen Algorithmus, welcher für dieses Projekt entwickelt wurde, zu beschreiben, muss zuerst die Codierung erklärt werden. Eine Codierung beschreibt, wie ein Genotyp aufgebaut ist. In diesem Fall ist ein Genotyp eine Liste aus ganzzahligen Werten. Diese Werte werden auch Allele genannt. In der Genetik gibt es beispielsweise ein Allel für blonde Haare. Ein Genotyp könnte wie folgt aussehen.

0	8	5	4	3	-1	6	1	2	7
---	---	---	---	---	----	---	---	---	---

beschriften

Es gibt in diesem Modell zwei Arten von Allelen. Allele mit einem Wert grösser gleich Null repräsentieren Besuche oder Pausen. Allele mit negativen Wert sind Separatoren. Diese Separatoren trennen zwei Routen voneinander. Demnach ist die Reihenfolge entscheidend. Im obigen Beispiel werden bei der ersten Route die Familien 0, 8, 5, 4 und 3 besucht. In der zweiten Route werden dann die Familien 6, 1, 2 und 7 besucht. Bei den Besuchen gilt es noch zu beachten, dass die Pausen für jeden Tag dupliziert werden. Das heisst, wenn ein Chlaus eine Pause machen möchte und es zwei Arbeitstage gibt, dann gibt es die Pause danach zwei Mal. Zwei Mal heisst, dass es zwei Besuche mit unterschiedlicher Nummer gibt, welche dann am Ende auf die originale Pause abgebildet werden. Das ist bedingt durch die Struktur des Genotyps. Routen, die von Separatoren getrennt sind, gehören zu einem Chlaus und zu einem Tag. Demnach kann man den obigen Genotyp auf zwei Arten interpretieren. Die erste Route gehört jedoch in beiden Fällen zum ersten Chlaus und zum ersten Tag. Bei der ersten Interpretation kann man annehmen, dass es einen Chlaus und zwei Tage gibt. Somit gehört die zweite Route zum ersten Chlaus und zum zweiten Tag. Alternativ kann man annehmen, dass es zwei Chläuse und einen Tag gibt. In diesem Fall gehört die zweite Route zum zweiten Chlaus und zum ersten Tag.

Andere genetische Algorithmen, die für das klassische VRPTW entwickelt wurden, haben oftmals keine Separatoren. Die Routen werden dort so erstellt, dass der Genotyp von links nach rechts durchlaufen wird und so viele Besuche wie möglich in die Route eingefügt werden. Erst, wenn eine Route voll ist, wird eine neue erstellt. Voll bedeutet in diesem Fall, dass beim Einfügen eines weiteren Besuchs die maximale Dauer einer Route überschritten würde. Dieses Vorgehen ist jedoch für diese Problemstellung ungeeignet, da die Routen gemäss der Zielfunktion möglichst gleich lange dauern sollen.

<http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=8B47E4720C0B9D1B705F726E1935F199?doi=10.1.1.35.1316&rep=rep1&type=pdf>

Dekodierung (konkret)

Da für das Berechnen der Fitness ein Phänotyp benötigt wird, muss auch eine Dekodierungsfunktion definiert werden. Die Aufteilung der Routen und die Besuchsreihenfolge sind bereits im Genotyp direkt ersichtlich. Dazu gehört auch, dass jeder Route ein Chlaus und ein Tag zugewiesen ist. Was noch fehlt sind die Startzeitpunkte der Besuche. Der Startzeitpunkt vom ersten Besuch jeder Route ist gleich dem Start vom dazugehörigen Tag plus der Wegzeit vom Startort zum ersten Besuch. Der Startzeitpunkt von allen weiteren Besuchen ist dann gleich dem Startzeitpunkt vom vorherigen Besuch plus die Dauer des vorherigen Besuchs plus Wegzeit zwischen den beiden Besuchen. Diese Methodik hat die folgenden beiden Einschränkungen. Erstens, fängt eine Route immer am Anfang des Tages an. Zweitens, können so keine Wartezeiten eingeplant werden, um beispielsweise eine Familie nicht zu früh zu besuchen. Der Grund für diese Entscheidung war vor allem der **Datensatz aus dem Jahr 2017**, wo problemlos alle Besuche in den Wunschzeiten besucht werden konnten, ohne dass Chläuse später hätten starten müssen. Zudem wäre der Genotyp und damit der genetische Algorithmus wesentlich aufwändiger geworden, wenn die Start- und Wegzeit noch hätte variabel sein sollen.

Selektion

Als Selektionsmechanismus wurde wie in REF die binäre Wettkampfsituation implementiert. Um dabei ein Elternpaar für die Rekombination zu finden, werden im ersten Schritt zwei zufällige Individuen aus der Population ausgewählt. Dann wird von diesen beiden das Individuum mit der besseren Fitness als erstes Elternteil akzeptiert. Das zweite Elternteil wird inwiefern das erste Elternteil aus zwei weiteren, zufälligen Individuen bestimmt.

<http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=8B47E4720C0B9D1B705F726E1935F199?doi=10.1.1.35.1316&rep=rep1&type=pdf>

Vogel

Rekombination

Rekombination, häufig auch “crossover” genannt, bedeutet Informationsaustausch zwischen zwei gepaarten Individuen. Das bedeutet in diesem Fall, dass aus zwei Elternteilen ein Nachkomme kreiert werden muss. Es wurden zwei Varianten umgesetzt, welche sich gemäß REF für das Traveling Salesman Problem (TSP) eignen sollen. Das VRPTW ist eine Erweiterung des TSP.

OX2

Die erste Variante ist der sogenannte “Order-based Crossover” (OX2). Dieses Verfahren wurde in Zusammenhang mit Zeitplanungsproblemen vorgeschlagen. Der OX2 Operator selektiert zuerst ein paar zufällige Positionen in einem Elternteil. Danach wird die Reihenfolge der selektierten Allele auf das andere Elternteil aufgezwungen. Als Beispiel seien die beiden Elternteile (1,2,3,4,5,6,7,8) und (2,4,6,8,7,5,3,1). Dann würden zufällig die Positionen zwei, drei und sechs ausgewählt. Die Allele an diesen Positionen sind 4, 6 und 5. Diese Allele befinden sich im ersten Elternteil an der vierten, fünften und sechsten Position. Der Nachkomme entspricht jetzt dem ersten Elternteil, aber ohne die vierte, fünfte und sechste Position: (1,2,3,*,*,*,7,8). Die fehlenden Allele werden dann in der Reihenfolge hinzugefügt, wie sie im zweiten Elternteil vorkommen. Das Resultat ist dann (1,2,3,4,6,5,7,8). Würde man die Elternteile vertauschen, entstünde der Nachkomme (2,4,3,8,7,5,6,1).

ER

Die zweite Variante ist der sogenannte “Edge Recombination crossover” (ER). Der ER eignet sich gut, wenn die verwendeten Kanten der Elternpaare beibehalten werden sollen und ist so der Ausgleich zu OX2. Bei ER wird für jedes Allel die Adjazenzmenge, d.h. eine Liste mit allen benachbarten Allelen erstellt. Die Strategie ist dann, dass man zuerst aus einem Elternteil das erste Allel nimmt und dann mithilfe der Adjazenzmenge das nächste Allel so wählt, dass das nächste Allel möglichst wenig verbleibende Nachbarn hat. Dafür muss ein Allel, welches zum Nachkomme hinzugefügt wird, aus allen Adjazenzmengen entfernt werden. Als Beispiel seien die beiden Elternteile (1,3,5,6,4,2,8,7) und (1,4,2,3,6,5,7,8). Die dazugehörigen Adjazenzmengen sind wie folgt:

Allel 1 hat Nachbarn: 3, 4, 7, 8

Allel 2 hat Nachbarn: 3, 4, 8

Allel 3 hat Nachbarn: 1, 2, 5, 6

Allel 4 hat Nachbarn: 1, 2, 6

Allel 5 hat Nachbarn: 3, 6, 7

Allel 6 hat Nachbarn: 3, 4, 5

Allel 7 hat Nachbarn: 1, 5, 8

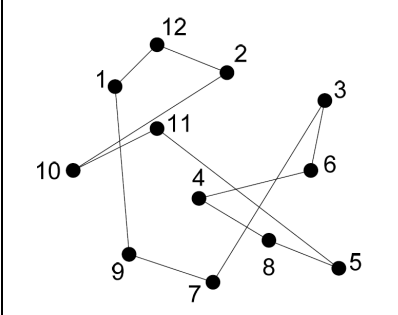
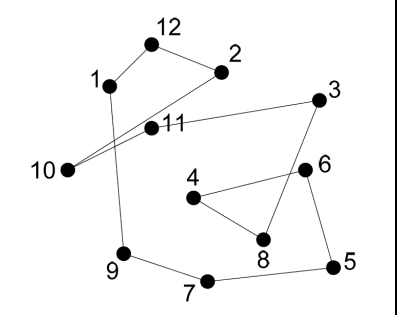
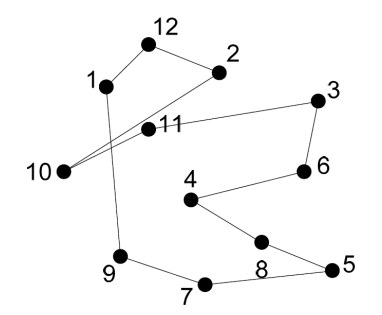
Allel 8 hat Nachbarn: 1, 2, 7

In diesem Beispiel würde der Nachkomme mit dem Allel 1 starten. Demnach muss das Allel 1 aus allen Adjazenzmengen entfernt werden. Allel 1 hat die Nachbarn 3, 4, 7 und 9. Das Allel 3 hat drei Nachbarn, da das Allel 1 bereits entfernt wurde. Die Allele 4, 7 und 8 haben alle zwei Nachbarn, somit kann aus diesen dreien zufällig ein Allel gewählt werden. In diesem Fall wird das Allel 8 gewählt und dem Nachkommen hinzugefügt. Allel 8 hat die Nachbarn 2 und 7. Als nächstes wird das Allel 7 gewählt, da es einen Nachbarn weniger hat als das Allel 2. Allel 7 hat dann nur noch den Nachbarn 5. Vom Allel 5 kann man wieder zufällig zwischen Allel 3 und 6 wählen, da beide zwei Nachbarn haben. Hier wird das Allel 6 zufällig gewählt. Von Allel 6 gibt es dann die Nachbarn 3 und 4, mit jeweils einem Nachbarn. Es wird das Allel 4 gewählt. Das Allel 4 hat nur den Nachbarn 2, welcher einzig den Nachbarn 3 hat. Der Nachkomme ist also am Ende (1,8,7,5,6,4,2,3).

http://ictactjournals.in/paper/IJSC_V6_I1_paper_4_pp_1083_1092.pdf

Mutation

Mutation bedeutet bei genetischen Algorithmen, dass ein Individuum beim Übergang in eine neue Generation zufällig abgeändert wird. Oftmals werden Mutationen von der Selektion wieder ausgemerzt, weil sie für die Fitness keinen Vorteil bringen. Teilweise können Mutationen jedoch überleben und zu besseren Lösungen führen. Generell sollen die Mutationen aber auch die Wahrscheinlichkeit reduzieren, dass der Algorithmus zu früh an einer lokalen Extremalstelle konvergiert und so die besten Lösungen nicht gefunden werden. Für dieses Modell wurde die Positionsmutation und die Inversion gewählt. Bei der Positionsmutation werden zwei zufällig gewählte Allele miteinander vertauscht. Bei der Inversion wird ein zusammenhängender Bereich gewählt und die Reihenfolge der Allele wird invertiert. Um lokalen Extremalstellen besser entkommen zu können, wird bei einer Mutation auch das Ausmass berechnet. Das Ausmass, oder die Schwere einer Mutation sagt aus, wie stark der Genotyp verändert wird. Um die Evolution nicht zu stark zu beeinträchtigen, sollte eine Mutation meistens ein kleines Ausmass haben. Nur mit einer geringen Wahrscheinlichkeit soll die Mutation den Genotypen stark verändern. Um diese Eigenschaft abzubilden, wird die Mutationsgrösse mittels einer Normalverteilung berechnet, wobei die Standardabweichung auf $\frac{1}{4}$ der Anzahl Allele festgelegt wird. Für den Erwartungswert wird bei der Positionsmutation eins und bei der Inversion zwei genommen. Da eine Mutation nicht ein negatives Ausmass haben kann, wird der Betrag der Abweichung zum Erwartungswert addiert. Ein Beispiel zu den beiden Mutationen findet sich in REF.

Original	Positionsmutation 3 ↔ 5	Inversion 5,8,4,6,3
(1,12,2,10,11,5,8,4,6, 3,7,9)	(1,12,2,10,11,3,8,4,6,5,7,9)	(1,12,2,10,11,3,6,4,8,5,7,9)
		

Beschriften REF Ausmass=1 und Ausmass=5

Konfiguration Einleitung

Nachdem nun der Aufbau sowie die Evolutionsoperatoren des genetischen Algorithmus bekannt sind, stellt sich die Frage nach der Konfiguration. Beispielsweise müssen die Populationsgrösse m und der prozentuale Anteil an zufällig generierten Individuen t bestimmt werden. Eine Möglichkeit diese Parameter zu bestimmen ist mittels sogenannter “Meta-Optimization”. Das bedeutet, dass mithilfe eines Optimierungsverfahren ein anderes Optimierungsverfahren verbessert wird. Diese Methodik wurde bereits im Jahr 1970 von Mercer and Sampson REF verwendet, um die optimalen Parameter für einen genetischen Algorithmus zu finden.

<https://www.emeraldinsight.com/doi/abs/10.1108/eb005486>

Konfiguration Umsetzung

In diesem Projekt wird eine Partikel-Schwarm-Optimierung (PSO) verwendet um die Parameter des genetischen Algorithmus zu optimieren. PSO ist selber eine Metaheuristik und eignet sich gut für numerische Optimierungsprobleme. Für eine genauere Beschreibung des Verfahrens sei auf REF oder REF verwiesen.

Wichtig ist zu wissen, dass beim PSO eine Zielfunktion definiert werden muss. In diesem Fall sieht das so aus, dass in der Zielfunktion der GA aufgerufen wird. Aufgrund dessen, dass der GA ein stochastische Suchverfahren ist, sind die Resultate nicht immer gleich gut. Um die Zielfunktion stabiler zu machen, wird der GA zehn Mal aufgerufen. Aus diesen zehn Durchläufen wird dann der Durchschnitt berechnet. Der Grund für die Wahl der Durchschnittsfunktion ist, dass so ein möglichst genauer Wert entsteht und so ein Optimum besser gefunden werden sollte. Würde beispielsweise immer den maximalen Wert genommen, wäre das Resultat stärker vom Zufall abhängig, als beim Durchschnitt.

Bei der Zielfunktion sollten bestenfalls alle 12 Datensätze aus Kapitel REF berechnet werden. Jedoch würde das zu lange dauern. Aus diesem Grund wird ein neu generierter Datensatz verwendet, der die 12 Datensätze repräsentieren soll. Der Datensatz wird ebenfalls nach dem Schema aus Kapitel REF erzeugt und hat die folgenden Eigenschaften. Der Datensatz besteht aus 16 Familienbesuchen und zwei Chläusen, welche zwei Arbeitstage haben. Beide Chläuse haben eine eingetragene Pause. Die Besuche haben zehn Wunschzeiten und zehn Nichtverfügbarkeiten, welche gleichmässig auf die beiden Tage aufgeteilt sind. Als Zeitlimit wird eine halbe Sekunde genommen.

Die Konfiguration des GA besteht aus sieben Parametern, welche definiert werden sollen. Zuerst sind das die Prozentwerte aus Abbildung REF. Dabei müssen nur drei Werte optimiert werden, da s so gewählt wird, dass die Summe von q, r, s und t gleich eins sind. Der nächste Parameter ist u dieser steht für die Wahrscheinlichkeit, dass ein Individuum, welches durch Rekombinations entstanden ist, zusätzlich noch mutiert wird. Dann muss noch definiert werden, welcher Operator für die Rekombination verwendet wird. Der Parameter o gibt an, mit welcher Wahrscheinlichkeit OX2 verwendet wird. ER wird demnach mit einer Wahrscheinlichkeit von $1 - o$ verwendet. Ähnlich ist bei den beiden möglichen Mutationen. Mit einer Wahrscheinlichkeit von p wird die Positionsmutation verwendet, andernfalls wird eine Inversion durchgeführt. Als letztes muss noch die Populationsgrösse m berechnet werden.

Konfiguration PSO selber

Dieser Abschnitt befasst sich mit der genauen Konfiguration des PSO und ist für das Verständnis nicht zwingend notwendig. Wie schon erwähnt, ist PSO eine Metaheuristik. Dementsprechend gibt es beim PSO auch Parameter die festgelegt werden müssen. Da diese Parameter nicht einfach zu bestimmen sind, werden gute Parameter aus einem Bericht [REF](#) von Herrn Pedersen verwendet. Diese guten Parameter wurden mithilfe von Meta-Optimization von verschiedenen Problemstellungen gefunden. Um die Konfiguration des GA zu optimieren, kommen gemäss Bericht [REF](#) zwei Problemgrößen für die PSO-Konfiguration in Frage. Erstens, eine Problemgröße die fünf Parameter und 10'000 Evaluationen der Zielfunktion hat. Zweitens, eine Problemgröße die zehn Parameter und 20'000 Evaluationen der Zielfunktion hat. Wie oben beschrieben, müssen für den GA sieben Parameter optimiert werden. Die Anzahl der Auswertungen der Zielfunktion wird auf 17280 gesetzt. Das entspricht einer Berechnungszeit von drei Stunden alleine für das Auswerten der Zielfunktion. Die Zielfunktionen, mit einer Berechnungszeit von $10 \times 500ms$ pro Auswertung, sollen achtfach parallel ausgeführt werden. Wie im Bericht [REF](#) ersichtlich, gibt es für die erste Problemgröße zwei mögliche PSO-Konfigurationen. Damit die beiden PSO-Konfigurationen möglichst unterschiedlich sind, wird die Konfiguration mit $S = 203$ genommen. Die beiden PSO-Konfigurationen werden nachfolgend mit C1 ($S = 203, \omega = 0.5069, \Phi_p = 2.5524, \Phi_g = 1.0056$) und C2 ($S = 53, \omega = -0.3488, \Phi_p = -0.2699, \Phi_g = 3.3950$) bezeichnet. Beim Ausführen des PSO könnte noch der Fall auftreten, dass eine Parameter ungültig wird. In diesem Falls wird Parameter auf den nächsten, gültigen Wert gesetzt. Ein Beispiel für einen ungültigen Parameter ist eine negative Populationsgröße, diese würde dann auf zwei korrigiert werden.

Konfiguration Resultate

Da der PSO ebenfalls ein stochastisches Suchverfahren ist, werden pro PSO-Konfiguration drei Durchläufe gemacht. Die Abbildung [REF](#) zeigt die Resultate der insgesamt sechs Durchgänge. Daraus lassen sich folgende Beobachtungen ableiten. Der Elitismus q macht über einen Drittel der Population, was relativ hoch ist. Der hohe Elitismus führt dazu, dass die Population schnell konvergiert, dafür wird der GA anfälliger für lokale Minima [REF](#). Der Operator OX2 ist gegenüber ER klar zu bevorzugen. Ein Grund könnte sein, dass OX2 weniger Rechenzeit braucht als ER. Bis auf eine Ausnahme, schneidet die Positionsmutation besser ab, als die Inversion. Bei den Populationsgrößen wird mehrfach zwei genommen, was für praktische Anwendungen ungeeignet ist. Denn mit zwei Individuen fehlt die Diversität der Population, was die Rekombination nutzlos macht. Dementsprechend kommen gemäss [REF](#) bei den meisten Implementationen von genetischen Algorithmen Populationsgrößen von mindestens 30 zum Einsatz.

Aufgrund des besten Werts für die Kosten und der geeignete Populationsgröße, werden die Parameter in der dritten Spalte für den GA verwendet.

https://www.researchgate.net/profile/Colin_Reeves/publication/2264198_Using_Genetic_Algorithms_With_Small_Populations/links/00b495205750f212f9000000.pdf

Konfiguration	C1	C1	C1	C2	C2	C2
q	0,5149	0,3567	0,6608	0,9982	0,5908	0,6018
r	0,0447	0,378	0,0108	0,0017	0,4091	0
s	0,2176	0,2653	0,1836	0,0001	0,0001	0,3982

t	0,2228	0	0,1448	0	0	0
u	0,3486	0	0,1101	0,0832	0	0
o	0,9037	0,8837	1	0,9039	1	1
p	0,3984	0,8856	0,7694	0,7556	0,6897	0,5944
m	2	102	2	2	2	2
Kosten	783,8	779,7	781,3	783,1	782,3	783,9

Beschriftung Lenovo W541

Um zu zeigen, dass die angenommenen 0.5 Sekunden für eine Berechnung des GA nicht zu viel sind, wurde der obige Test nochmal mit 0.25 Sekunden pro Berechnung wiederholt. Die Resultate vom Test mit 0.25 Sekunden sind in Abbildung REF ersichtlich. Bei genauerer Betrachtung fällt auf, dass die numerischen Werte zwischen den Durchgängen stark voneinander abweichen oder gar zufällig aussehen. Die grossen Unterschiede deuten darauf hin, dass die Qualität der Resultate stark vom Zufall abhängt und die 0.25 Sekunden nicht ausreichen.

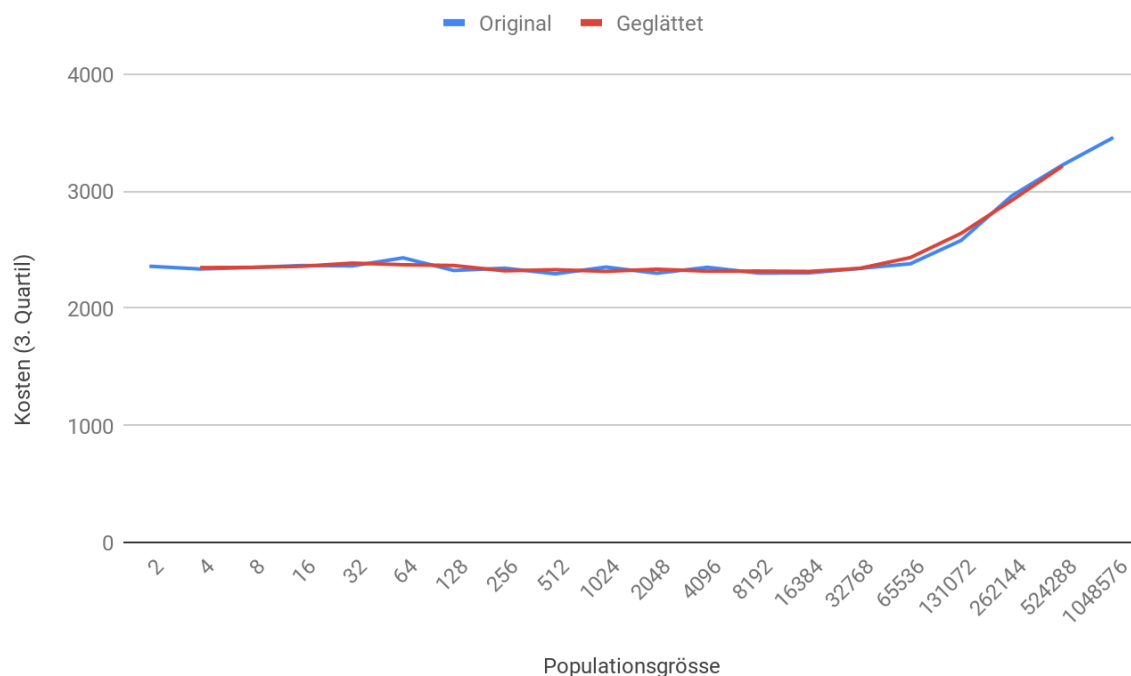
Konfiguration	C1	C1	C1	C2	C2	C2
q	0,6162	0,6651	0,5523	0,2633	0,4098	0,2145
r	0,3508	0,1703	0	0,0034	0,0724	0,195
s	0,0001	0,062	0,0001	0,7322	0,5178	0,5905
t	0,0329	0,1026	0,4476	0,0011	0	0
u	0,9995	0,7129	0,8981	0,9473	1	0,9987
o	0,0314	0,1053	0,1189	0,5218	0,4363	0
p	0,3548	0,4668	0,4823	0,0056	1	0
m	2	2	2	62	51	18
Kosten	785,3	790,5	786,8	791,8	787,3	793,3

Beschriftung Lenovo W541

Die Berechnungen vom PSO zeigen, dass die Population für den GA aus 102 Individuen bestehen sein soll, wenn das Zeitlimit für den GA 0.5 Sekunden beträgt. Bei den Datensätzen, die für die Evaluation verwendet werden, ist das Zeitlimit jedoch wesentlich höher. Zudem kommt eine Arbeit von Gotshall und Rylander REF kommt zum Schluss, dass die Populationsgrösse abhängig von der Problemgrösse sein soll. Um dem Rechnung zu tragen, werden die 12 Datensätze aus Kapitel REF innerhalb der Zeitlimiten mit verschiedenen Populationsgrössen berechnet. Die Populationsgrössen werden mit der Exponentialfunktion mit Basis zwei erstellt. Die kleinste getestete Populationsgrösse ist zwei. Die grösste getestete Populationsgrösse ist 1048576 was 2^{20} entspricht. Pro Populationsgrösse werden sechs Durchläufe gemacht, die parallel auf eine Lenovo W541 berechnet wurden. Dass genau sechs Instanzen parallel laufen hat den Grund, dass dies den Bedingungen der Evaluation aus Kapitel REF am nächsten kommen, was auch im Anhang REF ersichtlich ist. Als Mass für die

Performance wird die Anzahl der erzeugten Generationen verwendet. Die insgesamt 1440 Messwerte sind hier **REF** zu finden. Der Anhang zeigt auch, dass bei der Evaluation am besten zwei GA parallel ausgeführt werden sollen, um die Rechenkapazität optimal auszunutzen.

Um pro Populationsgrösse auf eine einzige Zahl zu kommen, wird entsprechend der Evaluation das dritte Quartil genommen. Bei der Betrachtung der Messwerte pro Datensatz fällt dann auf, dass sechs Messwerte nicht genug sind, um allfällige Schwankungen des GA zu kompensieren. Als Beispiel werden in Abbildung **REF** die Werte von Datensatz 10 gezeigt. Um diese Schwankungen auszugleichen, wurden diese Werte geglättet. Beim Glätten wurde statt dem originalen Messwert der Durchschnitt aus dem originalen Messwert und den beiden benachbarten Messwerten genommen. Wenn man dann pro Instanzgrösse den besten, geglätteten Wert nimmt, erhält man die Zuweisung aus Abbildung **REF**. Mit Instanzgrösse sind die Anzahl Besuche gemäss Definition in Kapitel **REF** gemeint. Bei Instanzgrössen, die mehrere zugehörige Datensätze haben, wird der Durchschnitt genommen.



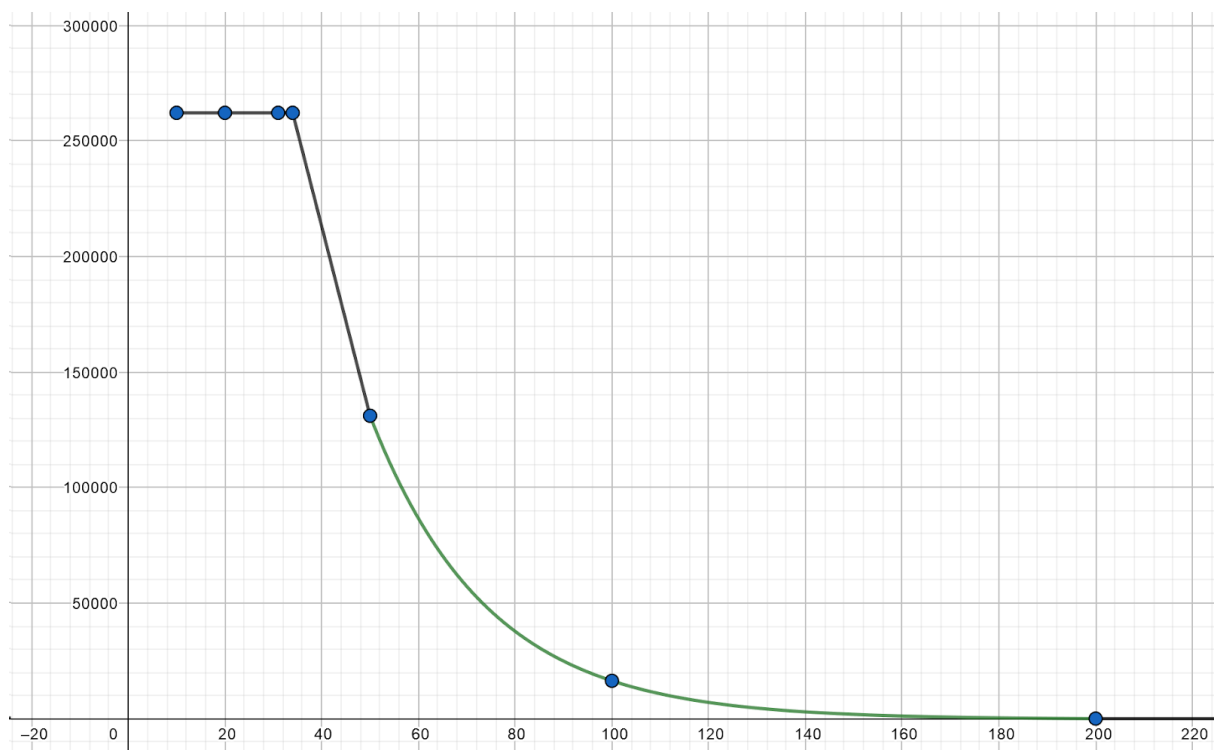
Beschriftung DS

Instanzgrösse	Beste Populationsgrösse
10	262144
20	262144
34	262144
50	131072
100	16384
200	16

Beschriftung

Die Werte aus Abbildung REF mögen für dieses Projekt genügen, jedoch soll für jede natürliche Zahl eine Populationsgrösse berechnet werden können. Formel REF zeigt die Funktion f , welche die Populationsgrösse m für eine beliebige Problemgrösse n berechnet. Zusätzlich zeigt Abbildung REF den grafischen Verlauf der Funktion. Der Bereich zwischen 34 und 50 wird linear interpoliert. Die Punkte bei 50, 100 und 200 werden mit einer Exponentialfunktion verbunden, damit die Populationsgrösse monoton fällt.

$$f(n) = \begin{cases} 262144 & \text{für } n \leq 34 \\ -8192 * n + 540672 & \text{für } 34 < n \leq 50 \\ -250.92 + 1036717 * e^{(-0.0413231 * x)} & \text{für } 50 < n < 200 \\ 16 & \text{für } n \geq 200 \end{cases} \quad (X)$$



Beschriften, Die Punkte entsprechen den Werten aus Abbildung REF

End

<https://www.geogebra.org/graphing/hpaj3bdz>

3.4 LocalSolver

LocalSolver ist ein Produkt um mathematische Optimierungsprobleme zu lösen. Im Hintergrund wird ein local search Algorithmus verwendet. Die Hauptprinzipien sind multithreaded adaptive simulated annealing, autonomous moves, welche die Lösbarkeit beibehalten und hoch optimierte inkrementelle Evaluation, welche es ermöglicht Millionen von Lösungen pro Minute zu analysieren. [<http://yetanothermathprogrammingconsultant.blogspot.com/2012/03/localsolver.html>] [https://www.msi-jp.com/localsolver/techinfo/the_first_paper.pdf]

Obwohl bei LocalSolver Einschränkungen gemacht werden können, sollen harte Einschränkungen so oft es geht vermieden werden. LocalSolver hat seine Stärken in kombinatorischen Problemen. Harte Einschränkungen werden im Idealfall als soft constraint eingebaut werden.

[<https://www.localsolver.com/docs/last/quickstart/mathematicalmodelingfeatures.html#constraints>]

Die Probleme werden für LocalSolver in einer eigenen Modellierungssprache beschrieben. Es gibt auch eine objektorientierte Schnittstelle zu diversen Programmiersprachen wie Python, C++, .NET und Java.

[<https://www.localsolver.com/product.html>]

Wir haben die Formulierung des Modells in drei Phasen aufgeteilt. Zuerst wird nur das VRP gemäss unserer Zielfunktion gelöst. In der zweiten Phase wird das VRPTW gelöst, allerdings gilt, dass die Wegzeiten zwischen den Besuchen fix sind und nicht verlängert werden dürfen. In der dritten Phase ist dies dann erlaubt. Dadurch werden alle optimalen Lösungen möglich.

Das Resultat der vorherigen Phase wird jeweils als Startpunkt der nächsten Phase verwendet. Die erste Phase wird durch gleichmässiges Aufteilen der Besuche auf die Chläuse initialisiert. Wir teilten das Problem in die verschiedenen Phasen auf, da ein grossteil der Kostenfunktion vom VRP abhängig ist. Das VRP ist vergleichsmässig einfach zu lösen und bietet einen guten Startpunkt für das VRPTW. Die Aufteilung der Phasen Zwei und Drei hat den Grund, dass verlängerte Wege eine grosse Performance Auswirkung auf das Modell haben, jedoch nur in seltenen Fällen einen wirklichen Nutzen bringt.

Die Gesamte Zeitlimite muss auf die drei Phasen verteilt werden. Die zur Verfügung stehende Zeit wird prozentual verteilt, dadurch gibt es nun drei Hyperparameter, welche zu optimieren sind. Es müssen jedoch nur zwei der optimiert werden, da sich der dritte Parameter trivial aus $1 - (P_1 + P_2)$ ergibt.

Um die Hyperparameter zu optimieren, haben wir ein Grid-Search gemacht. Wir haben zehn Datensätze mit 15 Besuchen, zwei Chläusen und zwei Tagen generiert. Vier Besuche haben am ersten Tag Wunschzeit und vier am zweiten Tag. Zudem haben zwei Besuche am ersten Tag nicht verfügbare Zeit und vier Besuche am zweiten Tag. Dies entspricht skaliert dem, was in den Evaluations-Datensätzen REF geprüft wird. Das Zeitlimit pro Datensatz beträgt 60 Sekunden.

Der Grid-Search wurde zweimal laufen gelassen. Die Kosten wurden pro Datensatz durch das jeweils beste Resultat des Datensatzes dividiert. Pro Datensatz wurde der Durchschnitt über

die zwei Läufe genommen. Danach wurde der Durchschnitt über alles genommen. Zur besseren Darstellung wurden die Werte noch mit der Funktion $f(x) = 100 \times (x - 1)$ transformiert.

		Zeitanteil Phase 1										
		0.0%	10.0%	20.0%	30.0%	40.0%	50.0%	60.0%	70.0%	80.0%	90.0%	100.0%
Zeitanteil Phase 2	0%	2.110	1.265	0.698	0.698	0.698	0.698	0.698	0.409	0.409	0.409	6.437
	10%	3.218	1.397	0.698	0.698	0.698	0.698	0.698	0.409	0.409	0.447	
	20%	2.897	0.716	0.698	0.698	0.698	0.698	0.698	0.409	0.447		
	30%	2.731	0.716	0.698	0.698	0.698	0.698	0.698	0.447			
	40%	2.470	0.716	0.698	0.698	0.698	0.698	0.717				
	50%	2.092	0.716	0.698	0.698	0.698	0.717					
	60%	1.226	0.716	0.698	0.698	0.717						
	70%	1.158	0.427	0.698	0.632							
	80%	1.158	0.378	0.379								
	90%	1.158	0.378									
	100%	1.158										

Grid-Search Resultat BILDBESCHRIFTUNG

Das Minimum des Grid-Search liegt bei den Konfigurationen (10%, 80%, 10%) sowie (10%, 90%, 0%).

Das Resultat zeigt, dass es zwei Hotspots für gute Konfigurationen gibt. Dies ist so zu erklären, da einige Probleme ihr Optimum nahe beim VRP haben und andere weniger. Wir entschieden uns die Konfiguration (10%, 80%, 10%) zu nehmen, da wir mindestens 10% für die dritte Phase verwenden wollen um keine optimale Lösung auszuschließen.

Durch den Vergleich der Lösung mit den Parametern (0%, 0%, 100%) und (0%, 100%, 0%) wird ersichtlich, dass die dritte Phase viel rechenaufwändiger ist als die zweite Phase. Die Kosten, falls nur die dritte Phase verwendet wird, sind fast doppelt so hoch.

TODO: Übergang zu Formulierung

In der folgenden Formulierung wird die Variable V als Menge aller Besuche verwendet. Die Anzahl der Besuche wird mit $n_v = |V|$ bezeichnet und ein einzelner Besuch mit $v_i \in \{1, \dots, n_v\}$. Nach dem selben Prinzip wird die Variable S als Menge aller Chläuse verwendet, n_s ist die Anzahl der Chläuse und $s_i \in \{1, \dots, n_s\}$ als einzelner Chlaus i . Um mehrere Tage abbilden zu können, wurden die ursprünglichen Chläuse mit den Anzahl Tagen n_p multipliziert. So ist bei einem Beispiel mit fünf Chläusen und zwei Arbeitstagen der Chlaus s_6 der erste Chlaus am Arbeitstag zwei. Damit jedes Problem optimal gelöst werden kann, können zusätzliche Chläuse erstellt werden. Es können pro Tag die Differenz der Anzahl Chläuse zu der Anzahl Besuche an Chläusen erstellt werden. Chläuse, welche zusätzlich hinzugefügt wurden, sind in der Menge $AS \subseteq S$ enthalten. Weiter wird die Variabel $n_{as} = |AS|$ verwendet.

Pausen werden sehr ähnlich wie Besuche behandelt, weshalb sie ebenfalls in der Menge V enthalten sind. Die Pausen sind in der Menge $B \subseteq V$ separat zusammengefasst. Ist eine Pause für einen Chlaus vorgesehen, so ist diese n_p mal in der Menge B enthalten. Die Pause $b_s \in B$ ist die Pause, welche für den Chlaus s vorgesehen ist. Die distanzen zwischen den Besuchen sind in einer Matrix *distance* eingetragen, wobei gilt *distance*(0, i) ist die Distanz von dem Startpunkt zum Besuch i . Analog dazu gilt *distance*(i , 0) ist der Rückweg von Besuch i zum Startpunkt.

Das Fundament der Formulierung ist die Menge $visitSequence_s$, welche für den Chlaus s alle besuchte Besuche enthält.

Es gilt die Einschränkung, dass alle $visitSequence$ eine Partition aller Besuche ist({REF})

$$\bigcup_{s=1}^{n_s+1} visitSequence_s = V \quad (X)$$

Nicht Benötigte Pausen werden in die Menge $visitSequences_{n_s+1}$ verschoben. Dadurch ist die Partitions-Bedingung erfüllt. Damit keine echten Besuche in dieser Menge sind, gilt die Einschränkung ({REF})

$$v \notin visitSequence_{n_s+1} \quad \forall v : v \notin B \quad (X)$$

Pausen dürfen nur von dem entsprechenden Chlaus besucht werden. Hierfür führen wir zuerst die Hilfsvariable $santaUsed_s \in \{0, 1\}$ ein, welche den Wert 1 annimmt, falls der Chlaus s benutzt wird: $santaUsed_s = |visitSequence_s| > 0 \quad \forall s \in S$. Damit lässt sich die Bedingung, dass die Pause für Chlaus s in der Menge $visitSequence_s$ vorhanden sein muss, falls der Chlaus benötigt wird ({REF1}). Ansonsten muss die Pause in der Menge $visitSequence_{n_s+1}$ vorhanden sein({REF2}).

$$santaUsed_s = 1 \rightarrow b_s \in visitSequence_s \quad \forall s \in S \quad (X)$$

$$santaUsed_s = 0 \rightarrow b_s \in visitSequence_{n_s+1} \quad \forall s \in \{1, \dots, n_s\} \quad (X)$$

Um die benötigte Wegzeit eines Chlaues zu berechnen, definieren wir die Funktion $distSelector(s, i)$ welche die Distanz für den Chlaus s von dem Besuch $i - 1$ in seiner $visitSequence_s$ zum Besuch an Stelle i zurückgibt ({REF}).

$$distSelector(s, i) = distance(visitSequence_s(i-1), visitSequence_s(i)) \quad (X)$$

Mithilfe dieser Funktion kann nun die Summe $SantaWalkingTime_s$ gebildet werden, welche die gesamte Wegzeit eines Chlaues umfasst ({REF}).

TODO IM WORD: Formel in eine Formel packen & Umbrüche schon machen :)

$$\begin{aligned} SantaWalkingTime_s = & distance(0, visitSequence_s(1)) + distance(visitSequence_s(|visitSequence_s|), 0) \\ & + \sum_{i=2}^{|visitSequence_s|} distSelector(s, i) \quad \forall s \in S \end{aligned} \quad (X)$$

Die Arbeitszeit eines Chlaues beinhaltet auch die Besuchsdauer, welche durch die Summe aller Besuchszeiten des Chlaues gebildet werden kann ({REF}).

$$SantaVisitTime_s = \sum_{i=1}^{|visitSequence_s|} duration(visitSequence_s(i)) \quad \forall s \in S \quad (X)$$

Die gesamte Arbeitszeit eines Chlaues s wird als $SantaWorkingTime_s = SantaWalkingTime_s + SantaVisitTime_s$ definiert. Diese Arbeitszeit muss innerhalb des Arbeitstages geschehen ({REF}).

$$SantaWorkingTime_s \leq dayDuration(s) \quad \forall s \in S \quad (X)$$

Das VRP ist nun formuliert und kann nach der Zielfunktion minimiert werden ({REF}). Diese Formulierung ist die erste Phase unseres Modells.

Besseren Satz finden

$$\begin{aligned}
& \text{minimize} \\
& \frac{1}{120} \times \max_i \text{SantaWorkingTime}_i + \\
& \frac{1}{90} \times \sum_{s=1}^{n_s} \text{SantaWorkingTime}_s + \\
& \frac{1}{90} \times \sum_{a=1}^{n_{as}} \text{SantaWorkingTime}_a + \\
& 400 \times \sum_{a=1}^{n_{as}} \text{santaUsed}_a \\
& \forall s \in S, \forall a \in AS
\end{aligned} \tag{X}$$

Damit aus dem VRP das VRPTW wird, müssen die Zeitfenster beachtet werden. Dazu definieren wir eine rekursive Funktion $\text{visitStartingTime}(s, i)$ welche den Startzeitpunkt des Chlauses s an der Stelle i zurückgibt. Die Abbruchbedingung ist bei dem ersten Besuch. Der Chlaus kann entweder gleich zu Tagesbeginn starten oder noch warten, hierfür verwenden wir die Varibel waitBeforeStart_s ({REF}).

$$\text{visitStartingTime}(s, i) = \begin{cases} \text{dayStart}_s + \text{waitBeforeStart}_s + \text{distance}(0, \text{visitSequence}_s(i)) & \text{für } i = 1 \\ \text{visitStartingTime}(s, i-1) + \text{duration}(\text{visitSequence}_s(i-1)) + \text{distance}(\text{visitSequence}_s(i-1), \text{visitSequence}_s(i)) & \text{für } i \neq 1 \end{cases} \tag{X}$$

Mit dem Startzeitpunkt der Besuche und der Besuchsdauer kann die Wunschzeit berechnet werden. Zur vereinfachung der Formulierung führen wir die Hilfsvariable $\text{visitEndTime}(s, i) = \text{visitStartingTime}(s, i) + \text{duration}_{\text{visitSequence}_s(i)}$ ein, welche den Abreisezeitpunkt für den Besuch an Stelle i in visitSequence_s abbildet ({REF}).

$$\text{visitDesiredDuration}(s, v) = \sum_{i=1}^{|d_v|} \begin{cases} 0 & \text{für } d_v(i, 2) < \text{visitStartingTime}(s, i) \vee d_v(i, 1) > \text{visitEndTime}(s, i) \\ \min(d_v(s, 2), \text{visitEndTime}(s, i)) - \max(d_v(s, 1), \text{visitStartingTime}(s, i)) & \text{sonst} \end{cases} \tag{X}$$

Analog dazu kann die Besuchsdauer in der Nichtverfügbarkeits Zeit berechnet werden ({REF}).

$$\text{visitUnavailableDuration}(s, v) = \sum_{i=1}^{|u_v|} \begin{cases} 0 & \text{für } u_v(i, 2) < \text{visitStartingTime}(s, i) \vee u_v(i, 1) > \text{visitEndTime}(s, i) \\ \min(u_v(s, 2), \text{visitEndTime}(s, i)) - \max(u_v(s, 1), \text{visitStartingTime}(s, i)) & \text{sonst} \end{cases} \tag{X}$$

Dadurch dass der Chlaus jetzt mitten im Tag Starten kann, muss die Einschränkung ({REF: $\text{SantaWorkingTime}_s \leq \text{dayDuration}(s) \quad \forall s \in S$ }) erweitert werden. Zeit ausserhalb der Tageszeit wird als Überzeit, Variabel overtime_s für Chlaus s , bezeichnet ({REF}).

$$\begin{aligned}
& \text{lastVisit}_s = \text{visitSequence}_s(|\text{visitSequence}_s|) \\
& \text{visitStartingTime}_s(|\text{visitSequence}_s|) + \text{duration}_{\text{lastVisit}_s} + \text{distance}(\text{lastVisit}_s, 0) \\
& \leq \text{dayEnd}_s + \text{overtime}_s
\end{aligned} \tag{X}$$

Die Zielfunktion der zweiten Phase wird mit den Summen über die Wunschzeiten und Nicht-Verfügbarkeitszeiten erweitert ({REF}).

$$\begin{aligned}
& \text{minimize} \\
& \text{WORD FORMATIEREN DER FORMEL}
\end{aligned} \tag{X}$$

$$\begin{aligned}
& \frac{1}{30} \times \sum_{s=1}^{n_s} overtime_s + \\
& \frac{1}{30} \times \sum_{s=1}^{n_s} \sum_{i=1}^{|visitSequence_s|} visitUnavailableDuration(s, i) + \\
& \frac{1}{90} \times \sum_{s=1}^{n_s} SantaWorkingTime_s + \\
& \frac{1}{120} \times \max(\{SantaWorkingTime_1, ..., SantaWorkingTime_{n_s}\}) + \\
& - \frac{1}{180} \times \sum_{s=1}^{n_s} \sum_{i=1}^{|visitSequence_s|} visitDesiredDuration(s, i) + \\
& \frac{1}{90} \times \sum_{a=1}^{n_{as}} SantaWorkingTime_a + \\
& 400 \times \sum_{a=1}^{n_{as}} santaUsed_a \\
& \forall s \in S, \forall a \in AS
\end{aligned}$$

In der dritten Phase sind auch verlängerte Wegzeiten zugelassen. Deshalb wird die Funktion $visitStartingTime_s$ leicht modifiziert. Die Variabel $waitBetweenVisit(s, i) \in \mathbb{R}^{\geq 0}$ wird eingeführt um die zusätzliche Wegzeit des Chlauses s zwischen seinem Besuch $i - 1$ und i abzubilden (**{REF}**).

$$visitStartingTime(s, i) = \begin{cases} dayStart_s + waitBeforeStart_s + distance(0, visitSequence_s(i)) & \text{für } i = 1 \\ visitStartingTime(s, i - 1) + duration(visitSequence_s(i - 1)) + distance(visitSequence_s(i - 1), visitSequence_s(i)) + waitBetweenVisit(s, i) & \text{für } i \neq 1 \end{cases} \quad (\mathbf{X})$$

Die Zielfunktion der dritten Phase bleibt dieselbe. Nach dem optimieren dieser Phase ist das Problem optimal gelöst.

ENDE DES KAPITELS SCHREIBEN ?

3.5 Google OR-Tools Routing

Intro

Der Ansatz Google OR-Tools Routing (Google Routing) basiert auf dem gleichnamigen Open-Source Softwarepaket. Google OR-Tools kann für verschiedene Problemstellungen benutzt werden, dazu gehören beispielsweise Flow-Probleme oder ganzzahlige lineare Programmierung. Der Google Routing Ansatz basiert nur auf der Routing Bibliothek, welche wiederum auf Constraint-Programmierung und verschiedenen Heuristiken basiert. Die Arbeit beim Google Routing besteht darin, das Problem zu formulieren, welches nachher von Google OR-Tools optimiert wird.

REF

Variablen

Vor der Formulierung werden zuerst noch die verwendeten Variablen definiert. Die Anzahl der Besuche entspricht n_v . In den Besuchen $v_i \in V$ sind auch die Pausen $b_s \in B$ enthalten, das heisst $B \subseteq V$. Der Start- und Endpunkt ist v_0 . Die Wegdauer, um von Besuch x zu Besuch y zu kommen, wird als $distance(x,y)$ definiert. Der Weg vom Startpunkt zum ersten Besuch ist also $distance(0,1)$. Die Variable l_x bezeichnet die Dauer eines Besuchs x . Der Startpunkt hat eine Besuchsdauer von null, das heisst es gilt $l_0 = 0$. In der Variable d_v sind die Wunschzeiten vom Besuch v enthalten. Ein Besuch kann mehrere Wunschzeiten haben, so entspricht d_{vi} der Wunschzeit i des Besuchs v . Eine einzelne Wunschzeit besteht wiederum aus einem Start und einem Ende. Der Start der Wunschzeit i vom Besuch v entspricht d_{vi0} , das dazugehörige Ende ist d_{vi1} . Analog zu den Wunschzeiten, steht u_{vi} für die Nichtverfügbarkeit i vom Besuch v .

Dimensionen

Bei der Routing Bibliothek ist das Problem in Dimensionen aufgeteilt. Wobei jeder Besuch pro Dimension einen Wert hat. Ein Besuch kann hierbei eine besuchte Familie, eine Pause oder der Startpunkt sein. Die erste Dimension ist die Dimension Zeit. In der Dimension Zeit ist jedem Besuch ein ganzzahliger Wert zugeordnet. In dieser Formulierung entspricht der Wert in der Zeitdimension dem Startzeitpunkt des Besuchs. Die Werte einer Dimension werden dabei aufsummiert. Aufsummieren heisst, dass der Wert jeweils zum vorherigen Besuch addiert wird. Der Betrag um den sich die Summe erhöht, wird mittels Callback berechnet. Der Callback für die Zeitdimension benötigt nur die Information, welcher Besuch y gemacht wird und welches der letzte Besuch x war. Da sich die Zeitdimension lediglich mit Startzeitpunkten befasst muss der Callback auch die Besuchszeiten dazurechnen. Konkret kann der Zeit-Callback wie folgt als Funktion dargestellt werden.

$$timeCallback(x,y) = l_x + w_{xy} \quad (X)$$

Bei einer Dimension kann eingestellt werden, ob der Wert des ersten Besuchs auf null fixiert werden soll. Der erste Besuch muss nicht am frühest möglichen Zeitpunkt stattfinden, also muss der Wert nicht auf null gesetzt werden. Des Weiteren, kann ein Schlupf definiert werden. Dieser Schlupf definiert einen maximalen Betrag, um den der Wert vergrössert werden darf. Ein Schlupf von null bedeutet also, dass die Wege exakt addiert werden müssen und nicht künstlich erhöht werden dürfen. Die Wege zwischen den Besuchen sollen nicht künstlich

verlängert werden, demnach wird der Schlupf auf null gesetzt. Zum Schluss kann der Wert maximale Wert einer Dimension noch begrenzt werden, diese Begrenzung wird auf das Ende des letzten Tages gesetzt, da später keine Besuche mehr gemacht werden sollen.

Als zweites wird noch eine Dimension Dauer erstellt, diese misst die Dauer der Routen. Bei dieser Dimension wird der Startwert fest auf null gesetzt, denn die Route beginnt beim Startpunkt mit einer Dauer von null. Die Dimension Dauer benutzt ebenfalls den Zeit-Callback [REF](#) und verbietet verlängerte Wege.

Als letztes werden noch Dimensionen für die Zuordnung der Pausen eingeführt. Um Pausen fest einem Chlaus zuzuordnen zu können, müssen diese eine Ressource verbrauchen, die nur der dazugehörige Chlaus hat. Solche Ressourcen werden mithilfe von Dimensionen abgebildet. Als Beispiel hat der Chlaus 1 eine Pause eingetragen. Es wird also eine Chlaus1Pause-Dimension erstellt, die misst, wie viele Pausen des ersten Chlaus in dieser Route liegen. Der Callback [REF](#) prüft also, ob ein Besuch die Pause von Chlaus 1 ist und gibt in diesem Fall die Zahl 1 zurück.

$$santa1BreakCallback(x, y) = \begin{cases} 1 & \text{für } v_x = b_1 \\ 0 & \text{sonst} \end{cases} \quad (X)$$

Für jeden Chlaus, der eine Pause zugeordnet hat, muss eine solche Pausen-Dimension erstellt werden. Die Pausen-Dimensionen haben alle einen Startwert von null und haben einen Schlupf von null. Der maximale Wert der Pausen-Dimension entspricht eins, da jeder Chlaus maximal eine Pause machen darf. Nachdem diese Pausen-Dimensionen erstellt wurden, kann für jede einzelne Pausen-Dimension und jeden Chlaus eine maximale Kapazität gesetzt werden. Da diese maximale Kapazität pro Chlaus optional ist, kann das jedoch vereinfacht werden. Damit Chlaus 2 keine Pausen von Chlaus 1 macht, wird die maximale Kapazität von Chlaus 2 in der Pausen-Dimension 1 auf null gesetzt. Würde jetzt Chlaus 2 die Pause von Chlaus 1 machen, hätte Chlaus 2 einen Wert grösser null in der Pausen-Dimension 1, was nicht zugelassen ist. Für Chlaus 1 wird in der dazugehörigen Pausen-Dimension 1 kein Maximum gesetzt, somit darf Chlaus 1 seine eigenen Pausen machen.

Tage

Der Zeitraum, in welchem die Besuche beim Google Routing gemacht werden müssen, ist grundsätzlich zusammenhängend und startet beim Zeitpunkt null. Jedoch geht die Problemstellung von mehreren, disjunkten Tagen aus. Um das umzusetzen, wird die Dauer des Zeitraums auf die Differenz vom Start des ersten Tages und dem Ende des letzten Tages gesetzt. Jedoch sind so immer noch Besuche zwischen zwei Tagen möglich. Um dieses Problem zu lösen, muss jeder Chlaus pro weiterem Tag einmal dupliziert werden. So kann jeder Chlaus einem Tag zugeordnet werden. Bei einem Beispiel mit 3 Chläusen und 2 Tagen, wären es dann insgesamt 6 Chläuse. Die ersten drei Chläuse werden dann dem ersten Tag zugeordnet. Die letzten drei Chläuse werden dem zweiten Tag zugeordnet. Eine solche Zuordnung geschieht über die Zeitdimension. So kann pro Chlaus ein Bereich gesetzt werden, der die Werte der Dimension einschränkt. Dabei gilt es zu beachten, dass die Besuche einem Chlaus zugeordnet werden und die Einschränkung nur für die Besuche des jeweiligen Chlaus gilt.

Zielfunktion

Bevor es um die Wunschzeiten und die Nichtverfügbarkeiten geht, soll zuerst das Optimierungsziel betrachtet werden. Beim Routing von Google OR-Tools kann nicht direkt eine Zielfunktion definiert werden, so wie es beispielsweise beim genetischen Algorithmus der Fall ist. Stattdessen, können einzelne Kosten definiert werden. Ein erster Kostenanteil ist der längste Arbeitstag, welcher mit 30 CHF pro Stunde bestraft wird. Um diesem Kostenanteil abzubilden kommt die Dimension Dauer ins Spiel. Wie oben beschrieben misst die Dimension die Dauer einer einzelnen Route. Es wird ein Kosten-Koeffizient von 30 CHF pro Stunde zum globalen Abstand dieser Dimension gesetzt. Dieser globale Abstand ist die maximale Differenz zwischen Start und Ende einer Route. Da eine Route immer mit einer Dauer von null startet, wird einfach der Wert beim Endpunkt genommen, also nach wie viel Zeit der Chlaus wieder zurück ist. Als nächstes entstehen Kosten für die Arbeitszeit der Chläuse. Diese Kosten werden über Callbacks definiert, die einem Chlaus zugewiesen sind. Da die Kosten von der Zeit abhängig sind, wird im Kosten-Callback **REF** wiederum ein Zeit-Callback verwendet.

$$costCallback(x,y) = timeCallback(x,y) \times 40 \quad (X)$$

Mithilfe dieses Kosten-Callbacks können auch die Kosten der zusätzlichen Chläuse berechnet werden. Um die einmaligen Kosten von 400 CHF für den Einsatz eines zusätzlichen Chlaus zu berechnen, wird dieser Betrag beim Startpunkt zusätzlich addiert. Die Kosten zwischen anderen Besuchen sind dann wie bei den normalen Chläusen, nur mit einem anderen Stundensatz. Beim Kosten-Callback müssen die 400 CHF noch mit 3600 multipliziert werden, da als Einheit überall Sekunden verwendet wird.

$$costAdditionalCallback(x,y) = \begin{cases} 400 \times 3600 + timeCallback(x,y) \times (40 + 40) & \text{für } x = 0 \\ timeCallback(x,y) \times (40 + 40) & \text{für } x \neq 1 \end{cases} \quad (X)$$

Als nächstes wird noch definiert, welche Kosten entstehen, wenn ein Besuch nicht gemacht wird. Dieser Fall tritt dann ein, wenn das Google Routing keine Lösung findet, dann werden nämlich keine Besuche gemacht. Die Kosten eines fehlenden Besuchs können mit einer Disjunktion gesetzt werden. Gemäss der Zielfunktion entstehen dann Kosten von 560 CHF, dieser Betrag muss erneut mit 3600 multipliziert werden. Hier ist noch zu erwähnen, dass diese Kosten auch entstehen, wenn eine Pause nicht gemacht wird. Das heisst, wenn ein Chlaus eine Route hat, die ausschliesslich aus einer Pause besteht, wird die Pause trotzdem gemacht. Da diese Route nur unnötige Kosten verursacht, wird diese nach dem Lösungsprozess einfach gelöscht.

Timewindows

Als letzter Teil der Formulierung wird noch die Umsetzung der Wunschzeiten und Nichtverfügbarkeiten beschrieben. Für die Zeitfenster, also Wunschzeiten und Nichtverfügbarkeiten, gibt es vier mögliche Strategien.

Die erste und einfachste Strategie (SN) ist es, die Zeitfenster zu ignorieren. Bei dieser Strategie wird also lediglich ein VRP mit mehreren Tagen gelöst.

Die zweite Strategie (SU) beschränkt sich auf die Nichtverfügbarkeiten. Die Besuche werden anhand der Nichtverfügbarkeiten ähnlich wie bei den Tagen beschränkt. Genauer gesagt, wird für jede Nichtverfügbarkeit eines Besuchs ein Bereich in der Zeitdimension des Besuchs verboten. Das sorgt dafür, dass der Besuch in der Nichtverfügbarkeit nicht besucht werden darf.

Die dritte Strategie (SDH) ist eine Abwandlung der zweiten Strategie, jedoch werden Wunschzeiten berücksichtigt. Für Besuche, die keine Wunschzeit haben, wird ausschliesslich die zweite Strategie verwendet. Das heisst Besuche in der Nichtverfügbarkeit werden verboten. Für Besuche, die eine Wunschzeit haben, wird die mögliche Besuchszeit so beschränkt, dass der Besuch in der Wunschzeit stattfinden muss. Das kann jedoch nur für eine Wunschzeit gemacht werden. Deshalb wird vorgängig die beste Wunschzeit berechnet. Wobei das alleinige Mass für die Güte einer Wunschzeit deren Dauer ist. Die möglichste Besuchszeit wird also beschränkt, sodass der Besuch in der längsten Wunschzeit stattfinden muss. Diese Beschränkung hat den Effekt, dass ein Besuch, welcher nicht in der Wunschzeit stattfinden kann, nicht gemacht wird.

Die vierte Strategie (SDS) baut auf der zweiten Strategie auf und erweitert diese um die Wunschzeiten. Die Wunschzeiten werden mittels einer weichen Einschränkung gesetzt. Auch hier kann nur eine einzige Wunschzeit verwendet werden. Die Auswahl der Wunschzeit aus mehreren Wunschzeiten erfolgt gleich wie bei der dritten Strategie. Die eigentliche Erweiterung besteht darin, dass eine obere und eine untere Beschränkung der Zeitdimension bei allen Besuchen mit Wunschzeiten gesetzt wird. Sollte eine der beiden Beschränkungen verletzt werden, kommen Mehrkosten dazu. Diese Kosten entsprechen dem maximalen Bonus, der durch das Einhalten der Wunschzeiten erzielt hätte werden können. Bei einem Besuch, bei welchem maximal 30 Minuten in der Wunschzeit liegen kann, beträge dieser Bonus $0.5 * 20$ CHF. Dieser Betrag muss auch hier wieder mit 3600 multipliziert werden.

Um zu entscheiden, welche dieser vier Strategien eingesetzt werden sollen, wurden alle 12 Datensätze mit jeder Strategie optimiert. Diese Tests wurden auf dem Server gemacht, der auch für die Evaluation in Kapitel REF verwendet wird. So wird sichergestellt, dass die Rechenleistung dieselbe ist, wie bei der Evaluation. Die Resultate dieses Tests sind in der Abbildung REF ersichtlich. Dabei wurden jeweils zwei Strategien parallel berechnet, um die zwei Kerne des Servers optimal zu nutzen.

	SN	SU	SDH	SDS
1	560	560	560	560
2	527	527	480	452
3	878	532	532	532
4	912	912	912	912
5	851	851	685	872
6	1121	855	855	855
7	1148	732	645	645
8	1215	690	601	601
9	2283	1723	22400	1448
10	4149	44800	44800	44800
11	7693	89600	89600	89600
12	38254	448000	448000	448000

Beschriftung

Entsprechend der Tabelle REF werden bei einem Rechner mit zwei Prozessorkernen bis Datensatz 9 die Strategien SDH und SDS verwendet. Bei grösseren Problem instanzen kommen dann die Strategien SN und SU zum Einsatz. Generell wird diese Unterscheidung der Strategien anhand der Anzahl Besuche gemacht. So sind grössere Problem instanzen als Probleme mit über 50 Besuchen definiert. Sollten jedoch mehr als zwei Prozessorkerne verfügbar sein, werden bis zu vier Formulierungen gleichzeitig berechnet, die sich lediglich in der Strategie unterscheiden. Allgemein wird jedoch maximal eine Strategie pro verfügbarem Prozessorkern gleichzeitig berechnet.

4. Evaluation

In diesem Kapitel werden die beschriebenen Lösungsansätze auf die in Kapitel [REF](#) definierten Datensätze angewendet. Die Resultate werden ausgewertet und diskutiert. Dabei wird auf verschiedene Eigenschaften der Ansätze eingegangen.

4.1 Messmethodik

Damit die Messungen reproduzierbar und realitätsnahe sind, müssen die Rahmenbedingungen definiert werden. Alle Messungen fanden auf einem HP ProLiant MicroServer Gen8 (Hersteller Nr: 712318-421) mit einem Intel Pentium G2020T Prozessor, 2GB Arbeitsspeicher und auf einer WD Green 1TB Festplatte statt. **Betriebssystem?** Dieser Server wird vom Kunde zur Verfügung gestellt und wird später für den produktiven Betrieb genutzt.

Evaluiert werden die im Kapitel 2 [REF](#) eingeführten Datensätze. Pro Datensatz wird ein Zeitlimit definiert. Jeder Lösungsansatz wird fünfmal pro Datensatz ausgeführt. Sollte ein Ansatz nicht ausgeführt werden können oder keine Lösung berechnen können, so wird die maximale Zeit und eine Lösung ohne Besuche angenommen. Die erhaltenen Kosten werden danach durch die Anzahl Besuche geteilt um die Datensätze in einem Verhältnis darzustellen. Die Anzahl der Besuche ist die Summe aus Familienbesuchen und der Anzahl Pausen multipliziert mit der Anzahl Tage. Weil die Resultate schwanken können, wird das dritte Quartil als Messwert genommen. Diese Messmethode soll sicherstellen, dass die Werte realistisch sind. Danach wird ein gewichteter Mittelwert pro Lösungsansatz über alle Datensätze berechnet. Die Gewichte sind so gesetzt, dass die realen Datensätze und der nächste hochskalierte Datensatz 50% des Gewichtes ausmachen, die kleineren Datensätze ein Gesamtgewicht von 25% haben und die grossen Beispiele die restlichen 25% ausmachen. Eine Übersicht der Werte findet sich in der Tabelle [REF](#).

Datensatz	Zeitlimit	Anzahl Besuche	Gewichtung
Datensatz 1	10 min	10	1
Datensatz 2	10 min	10	1
Datensatz 3	10 min	10	1
Datensatz 4	20 min	20	1
Datensatz 5	20 min	20	1
Datensatz 6	20 min	20	1
Datensatz 7	90 min	34	4
Datensatz 8	90 min	34	4
Datensatz 9	90 min	50	4

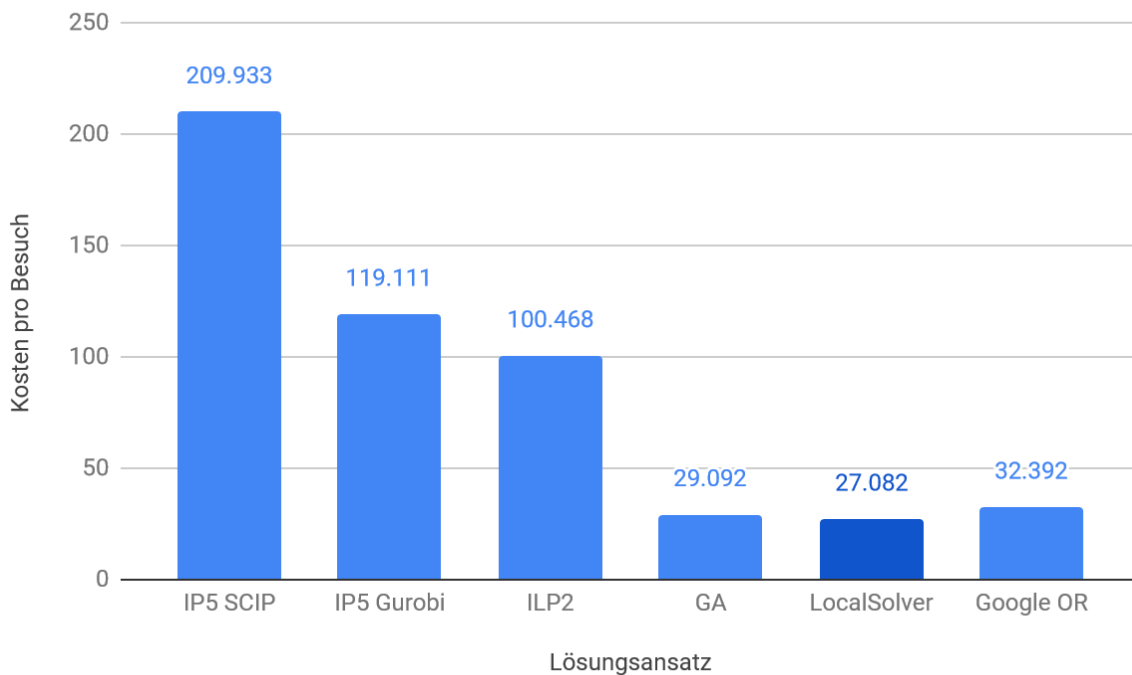
Datensatz 10	120 min	100	2
Datensatz 11	120 min	200	2
Datensatz 12	120 min	1000	2

Beschriftung

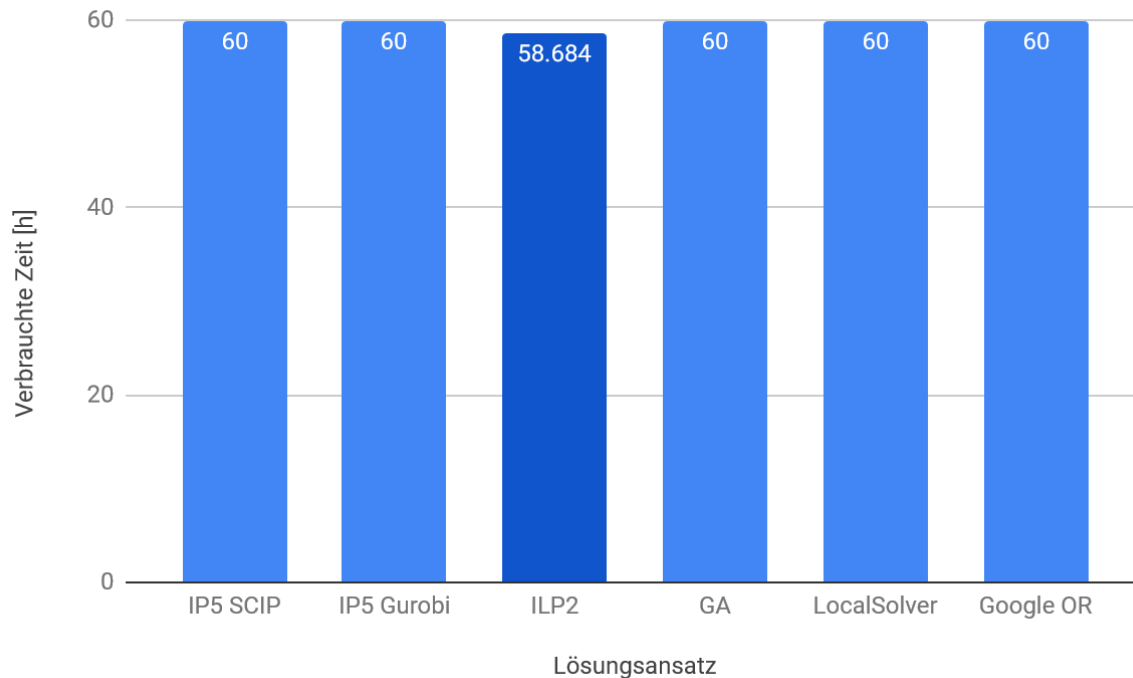
4.2 Übersicht

Aus Abbildung REF geht hervor, dass LocalSolver der beste Ansatz ist, um unsere Probleminstanzen in der vorgegebenen Zeit zu lösen. LocalSolver hat zusätzlich bei jedem Datensatz den besten Wert erreicht.

Wie man in Abbildung REF erkennen kann, ist ILP2 der einzige Ansatz, der nicht die vollen 60 Stunden braucht. Die einzelnen Werte der Ansätze pro Datensatz sind im Anhang REF abgebildet. Mit den Daten im Anhang REF können alle Abbildungen in diesem Kapitel generiert werden.



Beschriftung



Beschriftung

4.3 Laufzeit

Wie im vorherigen Kapitel [REF](#) bereits erwähnt wurde, ist ILP2 der einzige Ansatz, der bei einzelnen Datensätzen nicht die maximale Zeit braucht. Der Grund dafür ist, dass bei kleinen Datensätzen die gefundene Lösung der bestmöglichen Lösung entspricht und dann der Lösungsvorgang beendet wird. Dieser Beweis kann innerhalb der vorgegebenen Zeit nur bei den Datensätzen 1 und 3 erbracht werden. Bei den anderen Datensätzen wird die ganze Zeit gebraucht, da entweder die gefundene Lösung nicht optimal ist oder die Lösung nicht bewiesen optimal ist.

Alle anderen Ansätze terminieren erst beim Erreichen des Zeitlimits. Der Gründe dafür sind entweder, dass die Lösung nicht bewiesen optimal ist oder dass abgesehen vom Zeitlimit kein anderes Abbruchkriterium definiert ist.

4.4 Vergleich manuell geplante Route

Im Kapitel [REF](#) wird klar, wie die Ansätze betreffend Lösungsqualität gegeneinander abschneiden. Jedoch sagt das nichts darüber aus, ob diese Qualität der Lösungen auch für die praktische Anwendung ausreicht. In diesem Kapitel wird der Frage nachgegangen, ob die Ansätze bessere Lösungen erzeugen, als eine händische Planung.

Um objektiv Vergleichen zu können, sind die Kennzahlen der manuellen Lösungen berechnet worden, um daraus den Wert der Zielfunktion zu berechnen. Gegenstand dieser Betrachtung sind ausschliesslich die Datensätze 7 und 8, weil diese die einzigen realen Datensätze sind. Da die Chläuse bei der Planung nicht mit denselben Wegzeiten gerechnet haben, wird aus der

realen Routenplanung nur die Besuchszeit der ersten Familien übernommen. Die nachfolgenden Familien werden dann der Reihe nach hinzugefügt, wobei die exakten Wegzeiten aus den Datensätzen verwendet werden. Im Normalfall wird die Lösung durch dieses Verfahren besser. Die angenäherte Lösung könnte jedoch schlechter werden, wenn durch die Zeitverschiebung ein Besuch in der Nichtverfügbarkeit besucht wird oder Besuche aus dem Wunschzeitraum verschoben werden. Durch eine Prüfung beim Erzeugen der Kennzahlen wurde sichergestellt, dass diese beiden Fälle nicht eintreten. Dieses Verfahren wurde gewählt, um zu lange Wegzeiten zu verkürzen und so einen möglichst fairen Vergleich zu ermöglichen.

Abbildungen REF und REF zeigen die Kosten der Lösungen pro Ansatz. Die Kosten sind dabei farblich unterteilt, sodass die Aufteilung der Kosten sichtbar ist. Kürzere Balken sind besser als längere, da die Länge proportional zu den Kosten ist.

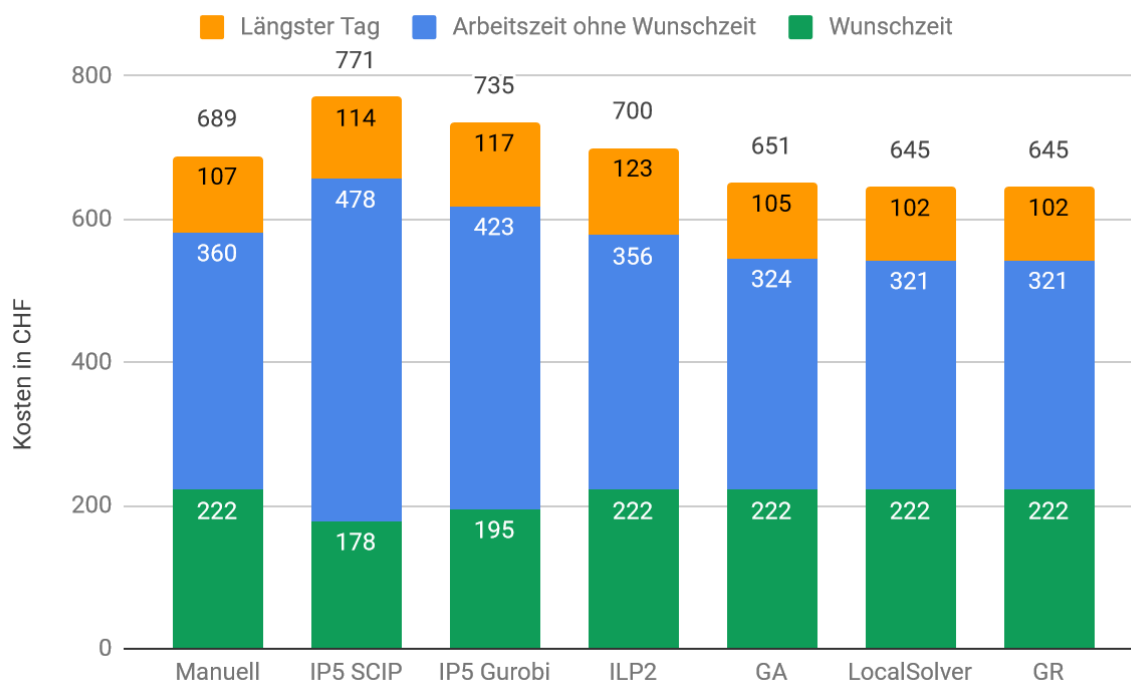
Beim Datensatz 7, welcher dem Jahr 2017 entspricht, können lediglich die Ansätze GA, LocalSolver und Google Routing eine bessere Lösung vorweisen, verglichen mit der manuellen Planung. Die möglichen Ersparnisse belaufen sich auf bis zu 44 CHF was gerundet 6.4% entspricht.

Beim Datensatz 8, welcher dem Jahr 2018 entspricht, ist die manuelle Planung die schlechteste Lösung. Ein Hauptgrund ist, dass bei der manuellen Lösung Wege ausserhalb der Arbeitszeit getätigt werden, was die Zielfunktion mit 120 CHF pro Stunde bestraft. Beim Datensatz 8 betragen die möglichen Ersparnisse 182 CHF was gerundet 23% entspricht.

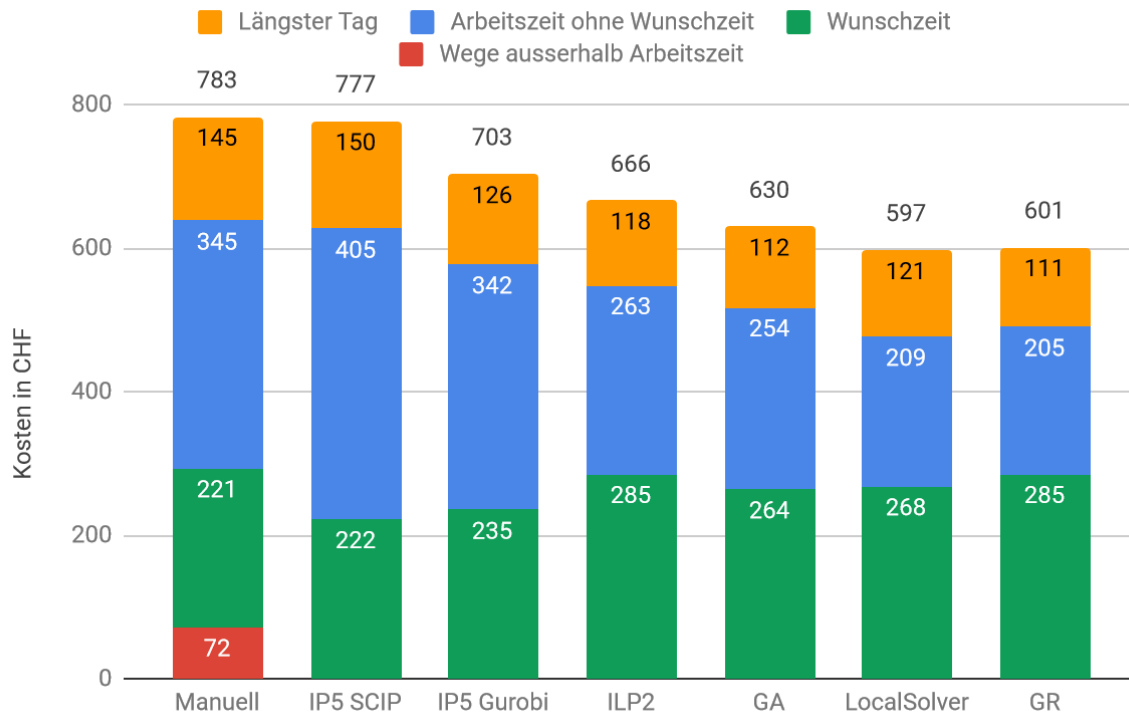
Vergleich mit LocalSolver

Generelle Aussage, ILP nicht so toll, andere alle besser

Ansätze brauchen weniger Zeit



Beschriftung Datensatz 7 (2017)



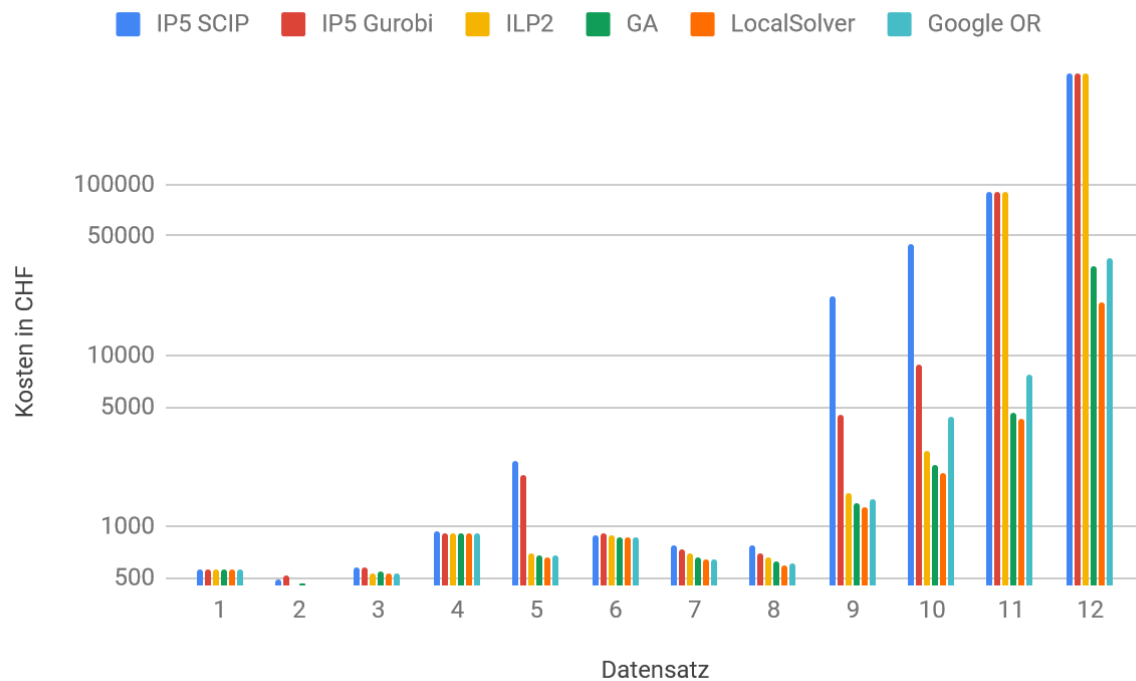
Beschriftung Datensatz 8 (2018)

Rundungsdifferenzen sind bei "Längster Tag" addiert

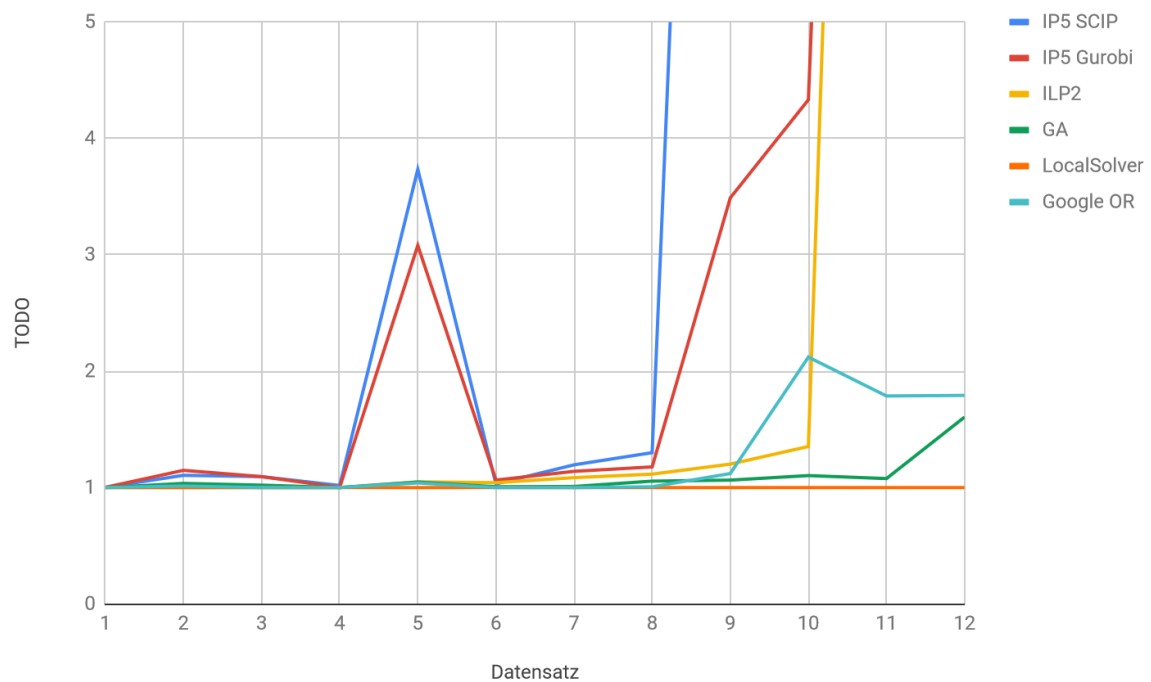
Nochmal prüfen, ob Gesamtkosten stimmen

4.5 Skalierbarkeit

Wie in Abbildung REF ersichtlich ist, variiert die Skalierbarkeit der Ansätze stark. Bei der Abbildung REF ist zu beachten, dass die vertikale Achse logarithmisch skaliert ist.



Beschriftung. Y log-skaliert.



Alternative

Die Ansätze IP5 SCIP und ILP5 Gurobi skalieren allgemein am schlechtesten und bilden die erste Gruppe in Sachen Skalierbarkeit. Das wird bereits beim Datensatz 5 klar, bei welchem die Lösungen im Vergleich zu den anderen Ansätzen über Faktor 3 zu teuer ist. Die maximal empfohlene Problemgrösse liegt also bei 10 Besuchen. Es muss jedoch erwähnt werden, dass teilweise auch grössere Probleminstanzen gelöst werden können.

Google OR-Routing hat den optimalen Anwendungsbereich bis Datensatz 9, der 50 Besuche hat. Dies ist auch an den verwendeten Strategien, die in **Kapitel X** beschrieben sind, ersichtlich. Bei Probleminstanzen, die mehr als 50 Besuche haben, kommt die Strategie zum Zug, bei welcher die Zeitfenster ignoriert werden. Das heisst die Leistungsfähigkeit reicht dann lediglich dafür aus, ein VRP mit beschränkten Tagen zu lösen. Die Kosten der Lösungen der Datensätze 10 - 12 sind dann um knapp Faktor zwei schlechter als bei LocalSolver. Bei den anderen Datensätzen sind die Abweichungen zu LocalSolver unter 13%.

Bis Datensatz 10 mit 100 Besuchen kann ILP2 eingesetzt werden. Bei den Datensatz 11 kann keine Lösung innerhalb des Zeitlimits gefunden werden. Bei Datensatz 12 reichen die 2 GB Arbeitsspeicher der Testmaschine nicht mehr und es kann keine Lösung mehr berechnet werden.

Der genetische Algorithmus kann ohne Bedenken bis Datensatz 11 mit 200 Besuchen verwendet werden. Erst bei Datensatz 12 reicht die Zeit nicht mehr und die Lösung wird gerundet 60% teurer als diejenige von LocalSolver.

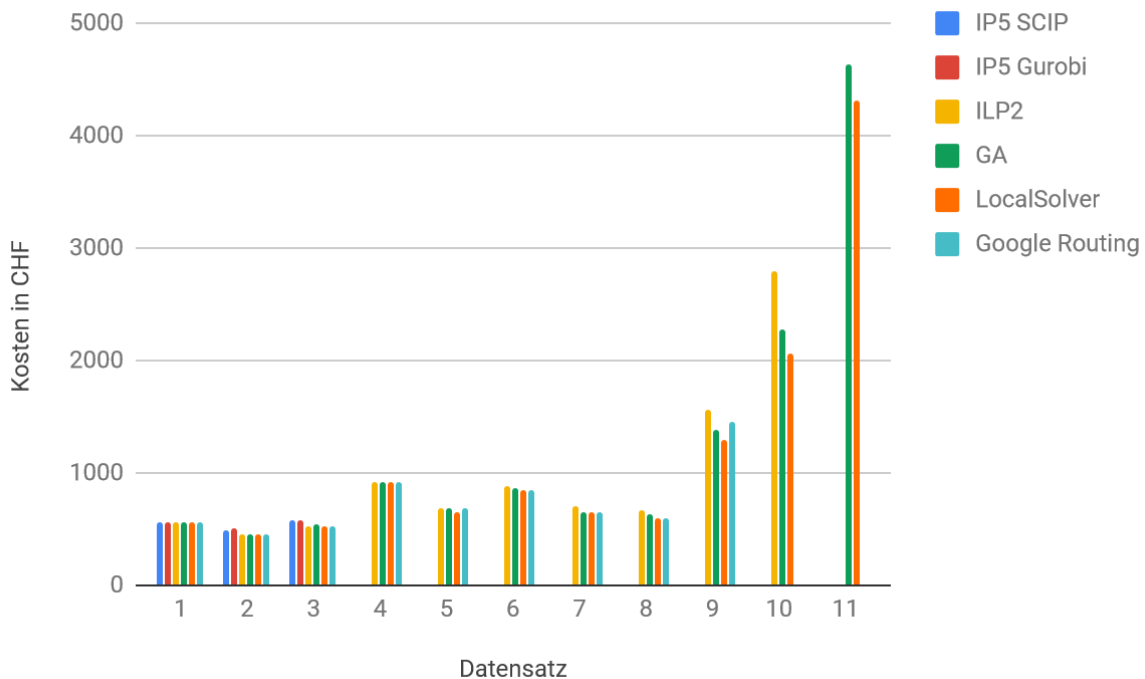
Mit LocalSolver, dem besten Ansatz, kann als einziges eine gute Lösung für den Datensatz 12 berechnet werden. Aufgrund der Kosten pro Besuch, die bei LocalSolver bei den Datensätzen 10-12 alle um 21 CHF liegen, wird davon ausgegangen, dass auch noch grössere Probleminstanzen in nützlicher Frist optimiert werden könnten.

Auswertung Skalierbarkeit der Resultate, maxima aufzeigen (sofern gefunden)

4.6 Lösungsqualität

Optimaler Anwendungsbereich pro Solver

In diesem Kapitel wird die Lösungsqualität der einzelnen Ansätze innerhalb des optimalen Anwendungsbereichs untersucht. Die optimalen Anwendungsbereiche sind im Kapitel **REF** beschrieben. Die Abbildung **REF** zeigt die Kosten der Lösungen wobei die Kosten nur eingetragen sind, wenn der Datensatz im optimalen Anwendungsbereich des jeweiligen Ansatzes liegt.



Beschriftung

Obwohl sich die Ansätze IP5 SCIP und IP5 Gurobi beim Datensatz 1 nur um einen Punkt von der besten Lösung unterscheiden, kann bei diesen Ansätzen nicht von einer guten Lösungsqualität gesprochen werden. Grund dafür ist, dass die Lösungen bei den Datensätzen 2 und 3 über 9% schlechter sind, als bei LocalSolver.

Die Lösungsqualität vom ILP2 ist stark abhängig von der Problemgrösse, auch innerhalb des optimalen Anwendungsbereichs. Lösungen der Datensätze 1 bis 4 entsprechen den besten gefunden Lösungen. Bei den Datensätzen 5 und 6 sind die Lösungen weniger als 5% schlechter als bei LocalSolver. Die Datensätze 7 und 8 werden dann gerundet 9% und 11% schlechter gelöst. Der Datensatz 9 wird dann noch schlechter gelöst, genauer gesagt, weichen die Kosten verglichen mit der besten Lösung um gerundet 20% ab. Die grösste Abweichung mit gerundet 35% tritt dann beim Datensatz 10 auf.

Der genetische Algorithmus hat bei den Datensätzen 1 bis 7 eine Abweichung von unter 5% verglichen mit LocalSolver. Bis zum Datensatz 11 sind die Lösungen vom GA maximal 11% schlechter als von LocalSolver.

Der Ansatz LocalSolver erreicht bei allen Datensätzen den Bestwert und geht bei der Lösungsqualität klar als Sieger hervor. Dabei sind die Lösungen von den Datensätzen 1, 3 und 4 bewiesen die bestmöglichen, da der erhaltene Wert gemäss ILP2 bewiesen optimal ist.

Der Google Routing Ansatz hat bei den Datensätzen 1, 3, 4 und 6 die beste bekannte Lösung erzeugt. Allgemein weichen die Lösungen bis Datensatz 8 nur um maximal 4% verglichen mit LocalSolver ab. Erst bei Datensatz 9 beträgt dann der Unterschied zur besten Lösung gerundet 12%.

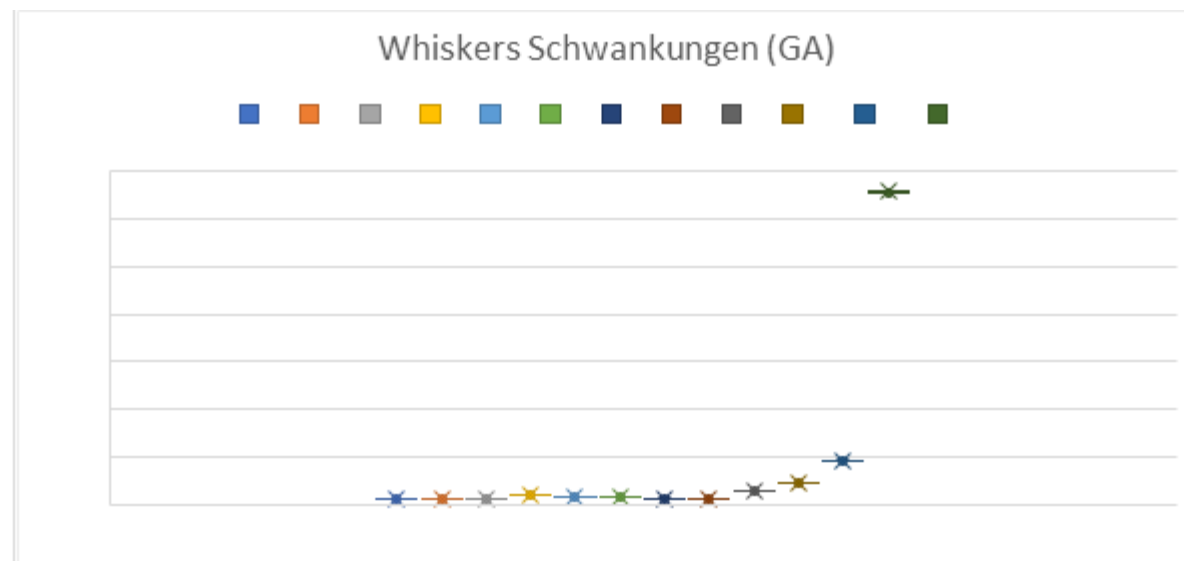
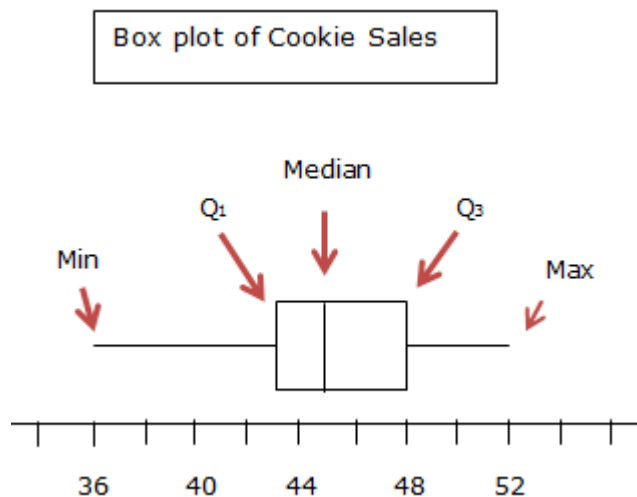
Im optimalen Anwendungsbereich

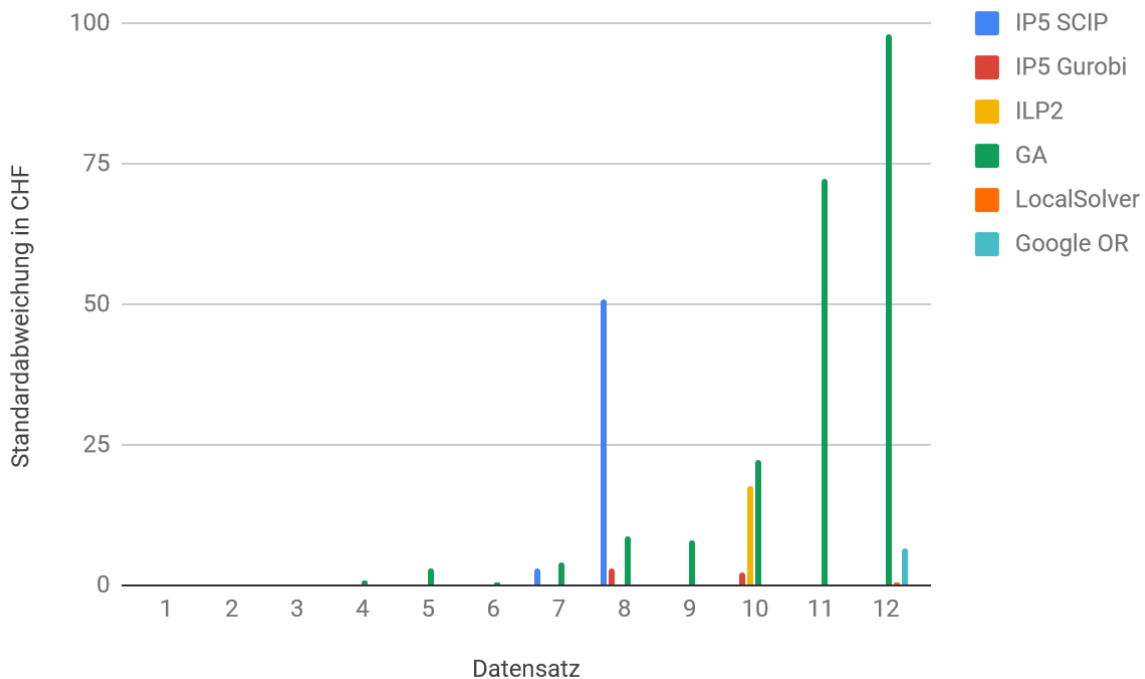
Ev. sagen dass Datensätze 1-n optimal gelöst sind

4.8 Lösungsstabilität

Wie im Kapitel REF beschrieben wird, gibt es pro Lösungsansatz fünf Durchläufe um allfällige Lösungsinstabilitäten aufzudecken. Wie Abbildung REF zeigt, wäre das nicht für alle Ansätze nötig. Abbildung REF zeigt die Standardabweichung der fünf Durchläufe pro Lösungsansatz und Datensatz.

Für Diagramm entscheiden





Beschriftung

Alle drei ILP-Ansätze sind meistens stabil. Bei den Lösungen vom IP5 SCIP treten nur bei den Datensätzen 7 und 8 Lösungsinstabilitäten auf. Die beste Lösung von Datensatz 8 des Ansatz IP5 SCIP kostet 669 CHF und die schlechteste kostet 777 CHF, was einer Differenz von 108 CHF entspricht. Beim IP5 Gurobi treten bei Datensätzen 8 und 10 geringe Lösungsinstabilitäten auf. Die Standardabweichungen beim IP5 Gurobi sind in beiden Fällen unter 3 CHF. Der Lösungsansatz ILP2 verhält sich bis zu dem Datensatz 10 stabil. Bei dem Datensatz 10 gibt es grössere Unterschiede zwischen den Lösungen. **TODO: Um das Zeitlimit werden neue Lösungen gefunden (Gurobi log bei ~7050 / 7200s) bei anderen Datensätzen wurde lange kein neuer upper bound gefunden.**

Beim genetischen Algorithmus treten ab Datensatz 4 Lösungsinstabilitäten auf. Diese Lösungsinstabilitäten nehmen mit der Grösse der Datensätze zu. So haben zum Beispiel die einzelnen Kosten der Lösungen von Datensatz 12 eine Standardabweichung von gerundet 98 CHF. Über alle Datensätze betrachtet sind die Kostenunterschiede durch Instabilitäten pro Familie nicht gravierend. Denn die Besuche einer Familie kosten wegen der Lösungsinstabilitäten maximal 0.85 CHF mehr, dies bei Preisen zwischen 17 CHF und 56 CHF pro Familie.

Die Ansätze LocalSolver und Google Routing sind grundsätzlich sehr stabil und haben nur beim Datensatz 12 Abweichungen. Bei LocalSolver war beim Datensatz 12 in einem Durchgang die Lösung um 1 CHF teurer. Beim Google Routing war beim Datensatz 12 in einem Durchgang die Lösung 15 CHF günstiger. Die Kosten der Lösungen beim Datensatz 12 betragen in beiden Fällen über 20'000 CHF, was bedeutet, dass die beiden Abweichung unter einem Promille sind.

1. Diagramm mit stdev
2. Diskussion, dass alle relativ stabil sind ausser ga

3. diskussion GA, diagramm mit differenz / visit skaliert -> aussage: gar nicht so schlimm, max- 2CHF pro visit, bei avg Visit cost von 20 CHF ist das etwa 10%.

4.7 Auswirkung Wunschzeiten und Nichtverfügbarkeit

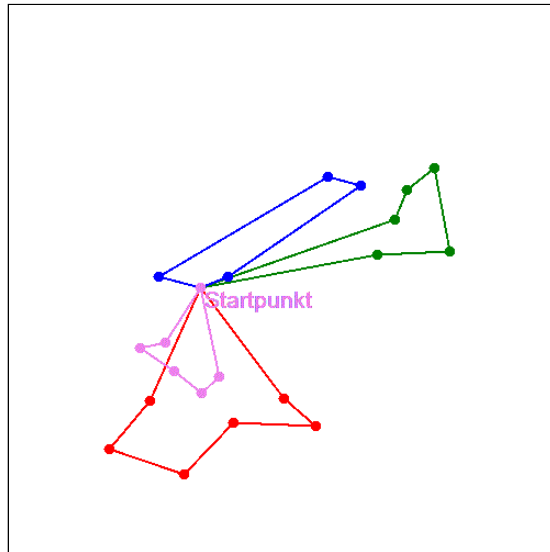
Um die Auswirkungen der Wunschzeiten und Nichtverfügbarkeiten zu analysieren wurden drei neue Datensätze generiert. Sie werden nachfolgend DSN, DSU und DSD genannt. Die Datensätze haben alle dieselben 20 Familien, die besucht werden müssen. Es werden jeweils 2 Chläuse an 2 Tagen eingesetzt, wobei kein Chlaus eine Pause eingetragen hat. Bei DSU ist für die ersten 10 Familien am ersten Tag und für die zweiten 10 am zweiten Tag Nichtverfügbarkeit eingetragen. Bei DSD haben die ersten 10 Familien am ersten Tag Wunschzeiten und die zweiten 10 am zweiten Tag. Die Wunschzeiten und Nichtverfügbarkeiten werden analog dem Kapitel REF generiert. Die Datensätze werden mit jedem Lösungsansatz fünfmal berechnet. Als Zeitlimite sind 5 Minuten gesetzt um die Auswirkungen besser aufzeigen zu können. Als Messwert wird dann das dritte Quartil genommen.

Lösungsansatz	DSN	%	DSU	%	DSD	%
IP5 SCIP	855	7.68	823	2.24	761	32.35
ILP Gurobi	801	0.88	823	2.24	781	35.83
ILP2	794	0	831	3.23	609	5.91
GA	807	1.64	836	3.85	624	8.52
LocalSolver	794	0	805	0	575	0
Google OR	794	0	805	0	581	1.04

TABLE REF

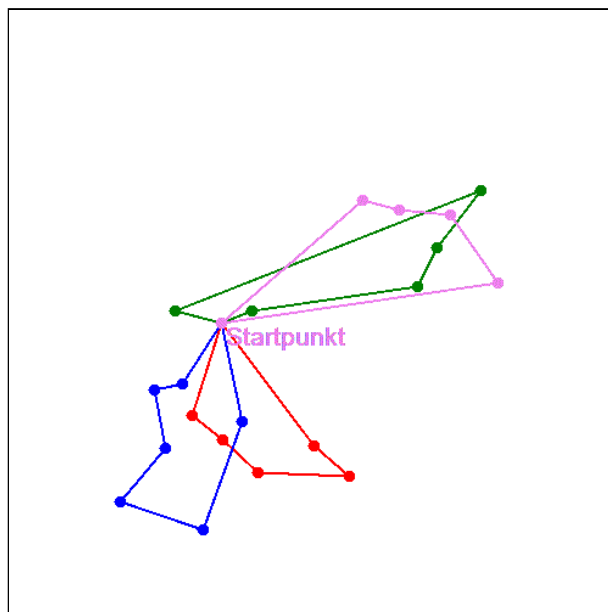
In der Tabelle REF sind die dritten Quartile der einzelnen Durchläufe aufgeführt. Zu erkennen ist, dass in der kürzeren Zeit nur der LocalSolver, das Google OR Routing und das ILP2 das Optimum für den Datensatz DSN erreicht haben. Die berechnete Lösung ist bewiesen optimal REF Anhang DSN Gurobi Log. Der genetische Algorithmus wurde für Berechnungen mit 20 Besuchen innerhalb von 20 Minuten parametrisiert, weshalb er hier nicht die beste bekannte Lösung erreicht.

Das IP5 SCIP sowie das IP5 Gurobi konnten das Problem nicht optimal lösen.



DSN bewiesen optimale Lösung

Kommen Nichtverfügbarkeiten hinzu, so sieht man beim IP5 Gurobi **REF LOG ANHANG** sowie beim IP5 SCIP, dass der erste Teil, das Clustering, innerhalb weniger Sekunden gelöst werden kann. Die Nichtverfügbarkeit hilft hier das Problem schneller zu lösen, weil damit die Aufteilung der Besuche mit einem Hard-constraint einfacher gemacht werden kann. Durch diese Aufteilung wird die Lösung jedoch stärker eingeschränkt und sie weicht um 2.24% von der besten bekannten Lösung ab. Obwohl das ILP2 ebenfalls als ganzzahliges lineares Programm formuliert wurde, erschwert die Einführung von Nichtverfügbarkeiten das Problem. Ohne Zeiteinschränkungen werden beim ILP2 424 Zeilen und 1853 Spalten benötigt. Mit den Zeiteinschränkungen von DSU werden mehr als doppelt so viele Zeilen (871) und etwa 20% mehr Spalten (2145) erstellt.

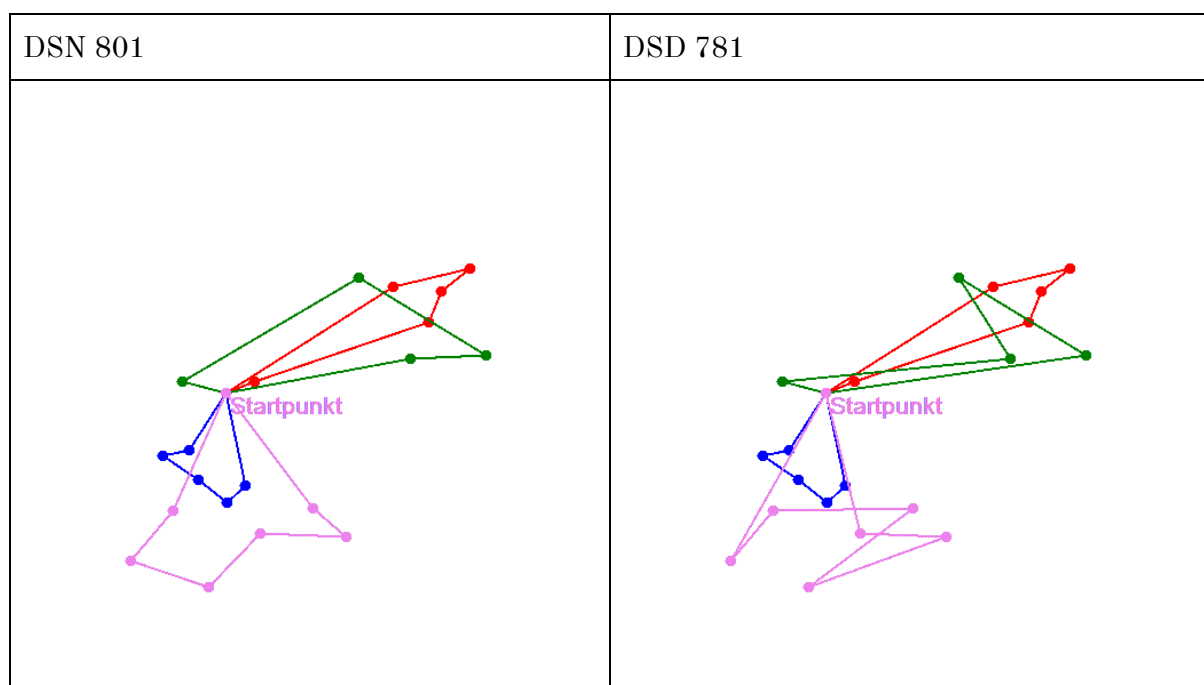


DSU beste bekannte Lösung

In der Abbildung REF sieht man, dass die beste bekannte Lösung des Datensatzes DSU von der besten bekannten Lösung des Datensatzes DSN stark abweicht. Es stimmen pro Route nie mehr als drei Punkte überein. Dies ist ein weiterer Grund, weshalb das ILP2 schlecht performt, da 10% der Zeit für das Lösen des VRP verwendet werden. Der genetische

Algorithmus verhält sich ähnlich wie beim Datensatz DSN. Mit dem genetischen Algorithmus ist es jedoch nicht möglich die beste bekannte Lösung zu erreichen, da die Startzeiten der Routen immer zum Tagesstart gesetzt sind und die beste bekannte Lösung erst später startet. Das Google Routing sowie der LocalSolver haben bei dieser Probleminstanz die beste bekannte Lösung berechnet. Der Rechenaufwand wird für die beiden Lösungsansätze grösser. Beim Google Routing werden grössere Datensätze, ab 51 Besuchen, aufgrund des zu grossen Rechenaufwand ohne Nichtverfügbarkeiten gerechnet.

Eine weitaus grössere Auswirkung haben die Wunschzeiten. Man sieht eine deutliche Verschlechterung der Lösungen aller Lösungsansätze mit der Ausnahme von LocalSolver. Durch die vorherige Aufteilung der Routen im IP5 SCIP und IP5 Gurobi ist eine gute Lösung nicht mehr möglich. Beide Lösungsansätze weichen um mehr als 30% von der best bekannten Lösung ab.



Tabellenbeschriftung

Man sieht in der Abbildung [REF](#), dass die berechneten Routen für den Datensatz DSD dieselben Besuche enthalten wie die Lösung für den Datensatz DSN. Der Unterschied liegt lediglich in der Reihenfolge wie diese besucht werden.

Resultatvergleich DSN/DSD für IP5 Gurobi, LocalSolver

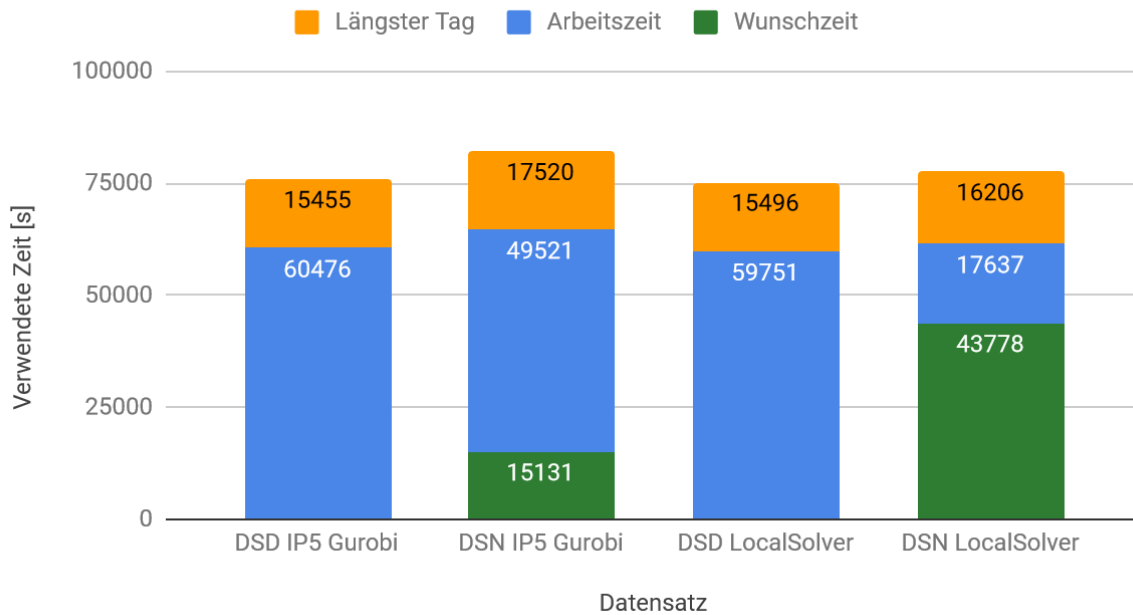


Diagramm Beschriftung

Vergleicht man die Resultate zwischen dem IP5 Gurobi und der besten bekannten Lösung, so wird der Unterschied der erfüllten Wunschzeit deutlich. Während bei der vom LocalSolver berechneten Route gerundet nur 28 Minuten mehr Arbeitszeit als bei der Lösung zum Datensatz DSD anfällt, wird bei der IP5 Gurobi Lösung 1h 10 Minuten Mehraufwand generiert.

Für den ILP2 Ansatz ist die Wunschzeit ebenfalls hinderlich. Das Resultat weicht stärker von der bekannten besten Lösung ab als bei den Berechnungen mit Nichtverfügbarkeiten oder ohne Zeitfenster. Das Google Routing erreicht mit den Wunschzeiten nicht mehr die beste bekannte Lösung. Ein Grund ist hier die Implementation des Algorithmus. Die Wunschzeiten werden nicht exakt berechnet sondern es wird der Abstand zur Wunschzeit linear bestraft. Dies führt dazu, dass die Besuche möglichst nahe an die Wunschzeiten gelegt werden. Dadurch entsteht hier eine Abweichung von einem Prozent zur besten bekannten Lösung. Aus der Analyse der [REF Kapitel Google OR auswahl der Strategien](#) geht hervor, dass auch bei diesem Lösungsansatz der Rechenaufwand für Wunschzeiten grösser ist als für Nichtverfügbarkeiten. Der genetische Algorithmus verschlechtert sich ebenfalls stärker. Aufgrund des kleinen Gewichts von Wunschzeiten werden in den ersten Generationen Lösungen bevorzugt, welche geringe Wegkosten und keine Nichtverfügbarkeit haben. Dadurch werden Individuen mit Wunschzeit aussortiert. Der Lösungsansatz mit dem LocalSolver berechnet erneut das beste bekannte Resultat. Der Rechenaufwand für Nichtverfügbarkeiten und Wunschzeit bleibt gleich.

4.8 Zusätzliche Chläuse

Erst am Schluss machen

LocalSolver, GA, Google Routing implementieren zusätzliche Chläuse. Es sollen alle Datensätze nochmal berechnet werden, wobei für die maximale Anzahl Chläuse die Anzahl Besuche (Familien + Pausen) genommen wird. So viele Chläuse sind nötig, um die theoretisch optimale Lösung immer erreichen zu können. Dabei sollte man sehen, dass die Performance deutlich schlechter wird.

Ev. müsste man hier einen Datensatz erstellen, der zusätzliche Chläuse braucht.

4.9 Preis/Lizenzkosten

	Akademisch	Nicht kommerziell	Kommerziell
IP5 SCIP	Gratis	Gratis	Auf Anfrage
IP5 Gurobi	Gratis	Auf Anfrage	Auf Anfrage
ILP	Gratis	Auf Anfrage	Auf Anfrage
GA	Gratis	Gratis	Gratis
LocalSolver	Gratis	Auf Anfrage	Auf Anfrage
Google Routing	Gratis	Gratis	Gratis

<https://scip.zib.de/index.php#license>

<http://www.gurobi.com/downloads/licenses/license-center>

<https://www.localsolver.com/pricing.html>

<https://github.com/google/or-tools#license>

Der Ansatz IP5 SCIP basiert auf Google OR-Tools sowie SCIP. SCIP ist unter der “ZIB Academic License” lizenziert und darf von akademischen und nicht kommerziellen Institutionen kostenfrei genutzt werden.

Die Ansätze IP5 Gurobi und ILP2 benötigen beide Gurobi und sind somit nur für akademische Nutzung kostenfrei. Andere Lizenzen erhält man vom Gurobi Vertrieb.

Der genetische Algorithmus wurde im Rahmen dieses Projekts von Grund auf entwickelt und darf somit kostenfrei verwendet werden.

LocalSolver bietet ebenfalls kostenlose Lizenzen für akademische Nutzer. Diese sind jedoch auf ein physikalisches Gerät beschränkt und müssen jeden Monat erneuert werden.

Google Routing, welches auf den Google OR-Tools basiert, steht unter der “Apache License 2.0” und darf kostenlos verwendet werden.

REFs

Aufteilen in: Akademisch, non Profit, kommerziell

4.10 Zusammenfassung

Nach Ansatz gruppiert stärken und schwächen aufzeigen

IP5 SCIP

IP5 Gurobi

ILP2

GA

LocalSolver

Google Routing

5. Schlussfolgerung / Empfehlung

Empfehlung Chlausgesellschaft (jetzt)

Aus der Evaluation geht hervor, dass LocalSolver der beste Ansatz ist. So ist die Empfehlung für die Jungwacht Niederwil, diesen Ansatz zu benutzen. Dabei spielt jedoch der Preis noch eine grosse Rolle. LocalSolver ist nur für akademische Anwendungszwecke kostenfrei. Möchte die Jungwacht LocalSolver benutzen, muss das Unternehmen hinter LocalSolver betreffend einer Lizenz angefragt werden. Dabei ist jedoch wichtig, dass sich eine kostenpflichtige Lizenz praktisch nicht lohnt. Betrachtet man die Datensätze 7 und 8, ergeben sich mit LocalSolver mögliche Einsparungen von maximal 4 CHF verglichen mit Google Routing.

Das führt auch gleich zur alternativen Empfehlung, Google Routing zu benutzen, falls die Lizenz von LocalSolver etwas kostet. Wie im Kapitel Evaluation beschrieben, sind die Resultate vom Google Routing bei den Datensätzen 1 bis 8 ähnlich gut wie diejenigen von LocalSolver. Jedoch fallen für das Google Routing keinerlei Lizenzgebühren an.

Falls jedoch die Anmeldungen bei der Chlausgesellschaft in Zukunft deutlich steigen, kommt eine kostenpflichtige Lizenz bei LocalSolver wieder ins Spiel. Beispielsweise beim Datensatz 9 ist die Lösung von LocalSolver 156 CHF günstiger als diejenige vom Google Routing. Sollten die Lizenzkosten für LocalSolver deutlich höher sein, als diese 156 CHF, lohnt sich der Einsatz bei Datensatz 9 nicht mehr. In dem Fall, dass mehr als 34 Besuche geplant werden sollen, lohnt sich dann der Einsatz des genetischen Algorithmus. Eine andere Möglichkeit ist auch, grössere Datensätze zuerst von Hand in mehrere kleinere Datensätze zu Unterteilen, sodass diese kleineren Datensätze dann vom Google Routing effizient gelöst werden können. Eine solche Aufteilung birgt jedoch die Gefahr, dass die besten Lösungen aufgrund einer ungünstigen Aufteilung, ausgeschlossen werden.

Um das Produkt für den Kunden möglichst robust zu machen, lautet die Empfehlung wie folgt. Sofern Preislich vertretbar, soll LocalSolver benutzt werden. Ob sich eine Lizenz lohnt, muss anhand der Ergebnisse der Evaluation abgeschätzt werden.

Sollte LocalSolver nicht in Frage kommen, wird empfohlen die Lösungen mit Google Routing und dem genetischen Algorithmus zu berechnen. Bestenfalls werden dabei beim Google Routing alle vier Strategien ausgeführt. Um den Instabilitäten des GA entgegenzuwirken könnte dieser Ansatz dreimal ausgeführt werden. Mit diesem Vorgehen würden dann die Berechnung maximal zehn Stunden dauern, was es dem Kunden ermöglicht, die Routen innerhalb einer Nacht berechnen zu lassen.

Unabhängig davon, ob am Schluss LocalSolver oder eine Kombination aus GA und Google Routing eingesetzt wird, sind wir überzeugt dass die Ansätze für die Chlausgesellschaft einen Mehrwert bieten. Denn durch die automatische Routenplanung werden die Routen nicht nur besser, sondern es entfällt auch ein grosser Teil der Handarbeit für die Planung. So können doppelt Kosten gespart werden. Das war beim Vorgängerprojekt, dem IP5, nicht der Fall, da die von Hand geplanten Routen tendenziell besser waren, als die berechneten.

Trotz des Umfangs dieser Arbeit, sind die Resultate stark auf die Problemstellung der Chlausengesellschaft Niederwil-Nesselnbach bezogen. Der Hauptgrund dafür ist neben der Zielfunktion, dass die 12 Datensätze der Evaluation ähnliche Eigenschaften aufweisen, wie die originalen Daten vom Kunden. So ist es möglich, dass sich die Ansätze ein anderes Lösungsverhalten zeigen, wenn beispielsweise die Besuche auf einem grösseren Gebiet verteilt sind oder die Besuche auf mehr als zwei Tage verteilt werden. Wie beim Solomon Benchmark REF erkennbar, gibt es mehrere Eigenschaften, die eine VRPTW Problemstellung ausmachen können. Beispiele sind, Datensätze bei denen die Routen durch Nichtverfügbarkeiten oder Wunschzeiten faktisch vorgegeben sind, Datensätze mit kurzen oder solche langen Wegen, Datensätze mit zufällig verteilten Besuchsorten oder solche mit lokalen Ansammlungen, Datensätze mit langen oder solche mit kurzen Tagen. Sollten die entwickelten Ansätze allgemeingültig verglichen werden, müssten solche Aspekte genauer untersucht werden.

Abhängigkeit Unavailable / Desired, empfehlungen/auswirkungen für Praxis

Vermutlich GA mit mehreren Runs, d.h. 12h laufzeit also 8 GA runs

Aufzeigen, dass eine Mögliche Lösung für grosse Probleme ist, dass sie manuell aufgeteilt werden. (EV. erst bei Schlussfolgerung)

Was könnte man noch alles machen?

Datensätze bei denen die Routen durch Unavailable vorgegeben werden

Same with Desired

Kurze/Lange Wege

Zufällig generierte Punkte vs. Cluster

Untersuchungen mit vielen Chläusen

6. Verzeichnisse

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

7. Ehrlichkeitserklärung

i. Anhang

Resultate Evaluations Durchläufe

IP5 SCIP

Datensatz	Avg	Min	Max	25-percentile	Measures
1	561	561	561	561	561,561,561,561,561
2	493	493	493	493	493,493,493,493,493
3	582	582	582	582	582,582,582,582,582
4	928	928	928	928	928,928,928,928,928
5	2446	2446	2446	2446	2446,2446,2446,2446,2446
6	882	882	882	882	882,882,882,882,882
7	769.2	764	771	771	770,771,770,764,771
8	732	669	777	776	777,751,687,776,669
9	22400	22400	22400	22400	22400,22400,22400,22400,22400
10	44800	44800	44800	44800	44800,44800,44800,44800
11	89600	89600	89600	89600	
12	448000	448000	448000	448000	

IP5 Gurobi

Datensatz	Avg	Min	Max	25-percentile	Measures
1	561	561	561	561	561,561,561,561,561,561
2	512	512	512	512	512,512,512,512,512,512
3	582	582	582	582	582,582,582,582,582,582
4	913	913	913	913	913,913,913,913,913,913
5	2018	2018	2018	2018	2018,2018,2018,2018,2018,2018
6	910	910	910	910	910,910,910,910,910,910
7	735	735	735	735	735,735,735,735,735,735
8	703	703	703	703	703,703,703,703,703,703
9	4518	4518	4518	4518	4518,4518,4518,4518,4518,4518
10	8924.3	8924	8925	8925	8925,8924,8924,8925,8924,8924
11	89600	89600	89600	89600	

12	448000	448000	448000	448000	
----	--------	--------	--------	---------------	--

ILP2

Datensatz	Avg	Min	Max	25-percentile	Measures
1	560	560	560	560	560,560,560,560,560
2	446	446	446	446	446,446,446,446,446
3	532	532	532	532	532,532,532,532,532
4	912	912	912	912	912,912,912,912,912
5	687	687	687	687	687,687,687,687,687
6	891	891	891	891	891,891,891,891,891
7	700	700	700	700	700,700,700,700,700
8	666	666	666	666	666,666,666,666,666
9	1556	1556	1556	1556	1556,1556,1556,1556,1556
10	2774.4	2756	2798	2787	2763,2798,2756,2787,2768
11	89600	89600	89600	89600	89600
12	448000	448000	448000	448000	

Genetischer Algorithmus

Datensatz	Avg	Min	Max	25-percentile	Measures
1	560	560	560	560	560,560,560,560,560
2	462	462	462	462	462,462,462,462,462
3	543	543	543	543	543,543,543,543,543
4	912.4	912	914	912	912,914,912,912,912
5	685.2	681	689	686	685,681,689,685,686
6	861.2	861	862	861	861,861,861,862,861
7	650.8	648	658	651	648,649,658,651,648
8	624.8	610	631	630	630,631,610,625,628
9	1376	1367	1388	1378	1367,1388,1378,1375,1372
10	2259.2	2221	2277	2272	2258,2221,2277,2268,2272
11	4586.6	4496	4663	4638	4526,4496,4638,4663,4610
12	32931.2	32802	33021	33000	33021,32851,32982,33000,32802

LocalSolver

Datensatz	Avg	Min	Max	25-percentile	Measures
-----------	-----	-----	-----	---------------	----------

1	560	560	560	560	560,560,560,560,560
2	446	446	446	446	446,446,446,446,446
3	532	532	532	532	532,532,532,532,532
4	912	912	912	912	912,912,912,912,912
5	655	655	655	655	655,655,655,655,655
6	855	855	855	855	855,855,855,855,855
7	645	645	645	645	645,645,645,645,645
8	597	597	597	597	597,597,597,597,597
9	1295	1295	1295	1295	1295,1295,1295,1295,1295
10	2060	2060	2060	2060	2060,2060,2060,2060,2060
11	4305	4305	4305	4305	4305,4305,4305,4305,4305
12	20547.2	20547	20548	20547	20547,20547,20547,20547,20548

Google OR Routing

Datensatz	Avg	Min	Max	25-percentile	Measures
1	560	560	560	560	560,560,560,560,560
2	452	452	452	452	452,452,452,452,452
3	532	532	532	532	532,532,532,532,532
4	912	912	912	912	912,912,912,912,912
5	681	681	681	681	681,681,681,681,681
6	855	855	855	855	855,855,855,855,855
7	645	645	645	645	645,645,645,645,645
8	601	601	601	601	601,601,601,601,601
9	1451	1451	1451	1451	1451,1451,1451,1451,1451
10	4369	4369	4369	4369	4369,4369,4369,4369,4369
11	7693	7693	7693	7693	7693,7693,7693,7693,7693
12	36817	36805	36820	36820	36805,36820,36820,36820,36820

IP5 Gurobi Logfile Clustering DSU

Gurobi 8.0.1 (win64, .NET) logging started 02/17/19 19:42:35

Academic license - for non-commercial use only

Optimize a model with 14902 rows, 5473 columns and 78549 nonzeros

Variable types: 1772 continuous, 3701 integer (3700 binary)

Coefficient statistics:

Matrix range [1e+00, 4e+03]

Objective range [8e-03, 1e-02]

Bounds range [1e+00, 2e+09]

RHS range [1e+00, 3e+04]

Warning: Model contains large bounds

Consider reformulating model or setting NumericFocus parameter
to avoid numerical issues.

Presolve removed 13486 rows and 4682 columns

Presolve time: 0.16s

Presolved: 1416 rows, 791 columns, 8139 nonzeros

Variable types: 342 continuous, 449 integer (442 binary)

Root relaxation: objective 7.971517e+02, 589 iterations, 0.02 seconds

Nodes		Current Node			Objective Bounds			Work		
Expl	Unexpl	Obj	Depth	IntInf	Incumbent	BestBd	Gap	It/Node	Time	
H	0	0	797.15166	0	65	- 797.15166	-	-	0s	
	0	0			909.4611111	797.15166	12.3%	-	0s	
	0	0	797.15452	0	77	909.46111	797.15452	12.3%	-	0s
	0	0	799.44311	0	76	909.46111	799.44311	12.1%	-	0s
	0	0	800.23029	0	90	909.46111	800.23029	12.0%	-	0s
	0	0	800.23029	0	94	909.46111	800.23029	12.0%	-	0s
	0	0	800.54935	0	97	909.46111	800.54935	12.0%	-	0s
	0	0	800.62137	0	105	909.46111	800.62137	12.0%	-	0s
	0	0	800.92070	0	99	909.46111	800.92070	11.9%	-	0s
	0	0	800.93194	0	102	909.46111	800.93194	11.9%	-	0s
	0	0	800.93839	0	100	909.46111	800.93839	11.9%	-	0s
	0	0	800.93839	0	100	909.46111	800.93839	11.9%	-	0s
	0	0	801.05834	0	113	909.46111	801.05834	11.9%	-	0s
	0	0	801.05849	0	116	909.46111	801.05849	11.9%	-	0s
	0	0	801.09127	0	111	909.46111	801.09127	11.9%	-	0s
	0	0	801.34165	0	105	909.46111	801.34165	11.9%	-	0s
	0	0	801.35272	0	105	909.46111	801.35272	11.9%	-	0s
	0	0	801.35272	0	105	909.46111	801.35272	11.9%	-	0s
	0	0	801.35272	0	105	909.46111	801.35272	11.9%	-	0s
	0	0	801.35272	0	78	909.46111	801.35272	11.9%	-	0s
0	2	801.35272	0	78	909.46111	801.35272	11.9%	-	0s	
H	29	30			856.7194444	802.44154	6.34%	25.7	0s	
H	32	31			851.3861111	802.44154	5.75%	23.8	0s	
*	60	44			25	848.0527778	804.22339	5.17%	20.5	0s
H	523	207				847.5361111	814.07806	3.95%	16.9	1s
*	769	319			23	846.7194444	816.10304	3.62%	15.9	1s

* 771	307	21	845.3861111	816.10304	3.46%	15.9	1s
* 1339	409	18	844.0527778	819.87817	2.86%	16.2	2s
* 1340	369	17	840.0527778	819.87817	2.40%	16.2	2s
* 1657	410	21	839.5361111	821.14220	2.19%	16.1	2s

Cutting planes:

Gomory: 6

Cover: 31

Clique: 6

MIR: 11

StrongCG: 1

Flow cover: 29

Inf proof: 9

Zero half: 7

Explored 4371 nodes (64880 simplex iterations) in **4.30 seconds**

Thread count was 2 (of 2 available processors)

Solution count 10: 839.536 840.053 844.053 ... 909.461

Optimal solution found (tolerance 0.00e+00)

Best objective 8.395361111111e+02, best bound 8.395361111111e+02, gap **0.0000%**

ILP2 DSN Logfile Beweis best

Gurobi 8.0.0 (win64, .NET) logging started 02/19/19 00:20:12

Academic license - for non-commercial use only

Optimize a model with 424 rows, 1853 columns and 16528 nonzeros

Variable types: 1 continuous, 1852 integer (1852 binary)

Coefficient statistics:

Matrix range [1e+00, 4e+03]

Objective range [8e-03, 4e+01]

Bounds range [1e+00, 3e+04]

RHS range [1e+00, 1e+00]

MIP start produced solution with objective 964.706 (0.02s)

Loaded MIP start with objective 964.706

Presolve removed 40 rows and 88 columns

Presolve time: 0.03s

Presolved: 384 rows, 1765 columns, 11256 nonzeros

Variable types: 1 continuous, 1764 integer (1764 binary)

Root relaxation: objective 6.907556e+02, 383 iterations, 0.01 seconds

Nodes		Current Node		Objective Bounds				Work	
Expl	Unexpl	Obj	Depth	IntInf	Incumbent	BestBd	Gap	It/Node	Time
0	0	690.75556	0	168	964.70556	690.75556	28.4%	-	0s
0	0	690.96829	0	166	964.70556	690.96829	28.4%	-	0s
0	0	690.98798	0	157	964.70556	690.98798	28.4%	-	0s
0	0	692.36390	0	172	964.70556	692.36390	28.2%	-	0s
0	0	693.89350	0	185	964.70556	693.89350	28.1%	-	0s
0	0	698.33487	0	171	964.70556	698.33487	27.6%	-	0s
0	0	698.42739	0	168	964.70556	698.42739	27.6%	-	0s
0	0	698.50559	0	174	964.70556	698.50559	27.6%	-	0s
0	0	698.50682	0	164	964.70556	698.50682	27.6%	-	0s
0	0	698.50682	0	163	964.70556	698.50682	27.6%	-	0s
0	2	698.50682	0	163	964.70556	698.50682	27.6%	-	0s
*	406	391	77	908.5305556	706.77031	22.2%	27.1	1s	
*	409	391	78	905.0833333	706.77031	21.9%	27.0	1s	
*	523	453	70	893.2722222	706.77031	20.9%	25.5	1s	
...									
2411516	56756	infeasible	43	793.03327	787.64769	0.68%	24.5	2025s	
2417487	54700	infeasible	41	793.03327	787.75771	0.67%	24.5	2030s	
2423749	52273	791.32002	42	61	793.03327	787.88871	0.65%	24.5	2035s
2431087	49493	789.34532	47	67	793.03327	788.04068	0.63%	24.5	2040s
2437662	47016	cutoff	39	793.03327	788.18347	0.61%	24.5	2045s	
2444518	44233	cutoff	40	793.03327	788.34921	0.59%	24.5	2050s	
2451771	41217	cutoff	42	793.03327	788.52935	0.57%	24.5	2055s	
2458979	38015	cutoff	46	793.03327	788.73540	0.54%	24.5	2060s	
2466260	34742	cutoff	46	793.03327	788.95485	0.51%	24.4	2065s	
2473589	31098	cutoff	33	793.03327	789.22517	0.48%	24.4	2070s	
2481710	27111	cutoff	39	793.03327	789.52778	0.44%	24.4	2075s	
2489722	22790	cutoff	30	793.03327	789.88973	0.40%	24.4	2080s	
2498399	17535	cutoff	29	793.03327	790.39028	0.33%	24.4	2085s	
2505731	12390	792.38333	43	44	793.03327	790.98653	0.26%	24.3	2090s
2515669	3824	cutoff	44	793.03327	792.27222	0.10%	24.3	2095s	

Cutting planes:

Gomory: 3

Clique: 15

MIR: 9

Flow cover: 1

Zero half: 8

Lazy constraints: 491

Explored 2519351 nodes (61113883 simplex iterations) in 2096.95 seconds

Thread count was 4 (of 4 available processors)

Solution count 10: 793.033 793.033 793.033 ... 798.503

Optimal solution found (tolerance 1.00e-04)

Best objective 7.930332686883e+02, best bound 7.929544059760e+02, gap **0.0099%**

Test GA Parallel

Datenset	Gerät	Anzahl parallele Instanzen	Generationen
5	W541	1	990476
5	W541	1	998481
5	W541	3	783694
5	W541	3	797742
5	W541	3	808555
5	W541	4	693247
5	W541	4	693239
5	W541	4	687536
5	W541	4	708781
5	W541	5	612416
5	W541	5	631978
5	W541	5	630451
5	W541	5	600960
5	W541	5	634764
5	W541	6	551188
5	W541	6	538390
5	W541	6	543666
5	W541	6	539619
5	W541	6	544066
5	W541	6	509520
5	W541	7	472738
5	W541	7	473637
5	W541	7	474337
5	W541	7	470861
5	W541	7	474749
5	W541	7	482850
5	W541	7	483127
5	W541	8	420640

5	W541	8	421648
5	W541	8	421163
5	W541	8	418552
5	W541	8	421164
5	W541	8	419403
5	W541	8	418289
5	W541	8	420895
5	Server	1	528091
5	Server	2	577208
5	Server	2	576160
5	Server	3	403368
5	Server	3	323515
5	Server	3	429989