Effiziente Reoptimierung in Branch & Bound-Verfahren für die Steuerung von Aufzügen

Bachelorarbeit bei
Prof. Dr. Dr. h.c. mult. M. Grötschel

vorgelegt von
Jakob Witzig[1]
Technische Universität Berlin
Fachbereich Mathematik

Berlin, 24. Januar 2013

[1]Konrad Zuse Institut für Informationstechnik Berlin, witzig@zib.de

Hiermit versichere ich die selbstständige und eigenhändige Anfertigung dieser Arbeit an Eides statt.

Ort, Datum

Unterschrift

Einleitung

Heutzutage ist eine Vielzahl der mehrstöckigen Gebäude mit Personenaufzugsgruppen ausgestattet. Uns wohl bekannt sind die sogenannten *konventionellen Systeme*. Bei diesen Systemen betätigt jeder ankommende Passagier eine der beiden Richtungstasten und teilt dem dahinterstehenden Steuerungsalgorithmus seine gewünschte Startetage und Fahrtrichtung mit. Betreten wird der zuerst auf der Startetage ankommende Aufzug mit gleicher Fahrtrichtung und ausreichend Kapazität. Die entsprechende Zieletage wird dem System erst nach dem Betreten der Fahrgastkabine mitgeteilt. Neben diesen konventionellen Systemen gibt es Aufzugsgruppen mit *Zielrufsteuerung*. Die Besonderheit eines zielrufgesteuerten Systems ist, dass ein ankommender Passagier bereits auf der Startetage seine gewünschte Zieletage angibt und eine Rückmeldung vom System erhält, welchen Aufzug er nutzen soll. Diese Zuweisung durch das System hat das Ziel, die Warte- und Reisezeiten der Passagiere zu minimieren. Ein wesentlicher Faktor bei der Berechnung warte- und reisezeitminimaler Fahrpläne ist das momentane Verkehrsmuster. Eine Einteilung der Verkehrsszenarien lässt sich am besten bei Bürogebäuden vornehmen. So ist es typisch für die Morgenstunden, dass jeder Passagier auf einer Zugangsebene seine Fahrt beginnt und alle Passagiere die gleiche Fahrtrichtung haben. Unter einer Zugangsebene ist z. B. der Haupteingang oder ein Parkdeck zu verstehen. Ein weiterer wesentlicher Punkt bei Zielrufsystemen ist die Art der Zuweisung der Passagiere durch das System. Zum einen gibt es *unmittelbar zuweisende* (UZ-) Systeme. In einem UZ-System wird nach jeder Ankunft eines Passagiers eine Momentaufnahme des momentanen Verkehrs erstellt und es findet eine Neuplanung und Zuweisung statt. Eine solche Momentaufnahme werden wir im späteren Verkauf als Schnappschussproblem bezeichnen. Jeder Passagier bekommt im Anschluss an die Lösung des Schnappschussproblems eine Mitteilung vom System, z. B. über ein Display, welchen Aufzug er benutzen soll. Zum anderen gibt es *verzögert zuweisende* (VZ-) Systeme. In diesen Systemen wird die Erstellung und Lösung eines Schnappschussproblems bis kurz vor Ankunft eines Aufzuges auf einer Etage verzögert. In einem VZ-System teilt das System allen wartenden Passagieren die geplanten Zieletagen des ankommenden Aufzugs mit. Jeder Passagier, der einen Ruf getätigt hat und zu einer dieser Zieletagen fahren will, kann jetzt diesen Aufzug betreten. Durch die Verzögerung muss im Vergleich zu einem UZ-System eine weitaus größere Menge von Passagieren zugewiesen werden. Dadurch kann der Lösungsprozess bedeutend aufwändiger werden. Vorteil eines VZ-Systems ist hingegen der größere Freiheitsgrad bei der Optimierung, da aufgrund der späten Zuweisung die weitere Verkehrsentwicklung mit einbezogen werden kann.

Eine umfassende Untersuchung der Unterschiede zwischen UZ- und VZ-Systemen in Bürogebäuden wurde im Rahmen eines Forschungsprojektes am Konrad-Zuse-Zentrum für Informationstechnik Berlin (ZIB) durchgeführt. Im Rahmen dieses Forschungsprojektes und der Dissertation „Online Optimization: Probabilistic Analysis and Algorithm

Engineering“ [Hil10] von Dr. Benjamin Hiller wurde der Planungsalgorithmus Exact-Replan entwickelt. Dieser Algorithmus kann sowohl UZ-System als auch VZ-System steuern und ist speziell auf zielrufgesteuerte Personenaufzugsgruppen ausgerichtet. Um eine Zuweisung der neu angekommenen Passagiere durchführen zu können, erstellt der Exact-Replan ein Schnappschussproblem in Abhängigkeit der Zuweisungsstrategie (unmittelbar oder verzögert). Nach der Erstellung eines Schnappschussproblems wird zuerst eine heuristische Startlösung ermittelt. Im Anschluss daran wird mithilfe von LP- und IP-Techniken, wie Branch & Bound und Spaltengenerierung, ein im Bezug auf Warte- und Reisezeit möglichst guter Fahrplan ermittelt, wobei für jeden Aufzug separat eine Menge zulässiger Touren konstruiert wird. Die Lösung der Schnappschussprobleme ist mithilfe des Exact-Replan für UZ-Systeme sehr effizient möglich, sodass dieser in Echtzeit die Steuerung einer Personenaufzugsgruppe durchführen kann. Für VZ-Systeme hingegen kann der Lösungsprozess eines Schnappschussproblems sehr viel Zeit in Anspruch nehmen, sodass ein Einsatz in einer „realen“ Personenaufzugsgruppe momentan noch nicht praktikabel ist.

Da aber ein VZ-System aufgrund des größeren Freiheitsgrades interessant für die Praxis ist, wollen wir uns in den folgenden Kapiteln mit einer effizienteren Lösung dieser Art von Schnappschussproblemen befassen. Es genügt dabei den Lösungsprozess eines Schnappschussproblems zu betrachten. Das Ziel ist eine Reduzierung der benötigten Rechenzeit. Wie bereits erwähnt konstruiert Exact-Replan die Menge zulässiger Touren für jeden Aufzug separat, daher genügt es, den Prozess der Spaltengenerierung auch lediglich für einen Aufzug zu betrachten. Unter *Reoptimierung* verstehen wir die Konstruktion zulässiger Spalten in den jeweiligen Iterationsrunden der Spaltengenerierung innerhalb eines Schnappschussproblems. Als eine Iterationsrunde bezeichnet wir einer Menge zulässiger Touren mit negativen reduzierten Kosten. Eine *effiziente* Reoptimierung zeichnet sich durch die Wiederverwendung und Aufbereitung von Informationen aus vorangegangenen Iterationsrunden desselben Schnappschussproblems aus. Zu den wichtigen Informationen gehört der konstruierte Suchbaum der vorherigen Iterationsrunde mit seinen ausgeloteten (abgeschnittenen) Blättern sowie konstruierten Touren bzw. Spalten, welche in der Iterationsrunde ihrer Konstruktion nicht zur Lösung des Teilproblems der Spaltengenerierung beitrugen. Eine solche Wiederverwendung und Aufbereitung von Informationen nennen wir *Warmstart*.

Die Arbeit ist wie folgt gegliedert. Nach einem kurzen Überblick über die mathematischen Grundlagen (Kapitel 1) diskutieren wir allgemeine Hintergrundinformationen für die Steuerung von Personenaufzugsgruppen. Im darauffolgenden Kapitel 3 betrachten wir zunächst allgemeine Branch & Bound-Verfahren. Des Weiteren führen wir ein *Ähnlichkeitsmaß* ein, welches dazu dient, die Struktur von Bäume zu vergleichen. Motiviert ist die Konstruktion eines solches Maßes durch die Vermutung, dass die Suchbäume, welche bei der Lösung des Teilproblems der Spaltengenerierung entstehen, mit zunehmenden Iterationsrunden einen Großteil identischer Touren enthalten. Es wird sich herausstellen, dass die Vermutung korrekt ist. Aufgrund dieser Ähnlichkeit konstruieren wir anschließend ein Branch & Bound-Verfahren, welches Optimierungsprobleme mit identischen Lösungsräumen und sehr ähnlichen Zielfunktionen löst. Dazu nutzt dieses Verfahren bereits gewonnene Informationen vorheriger Lösungsprozesse, wir können es daher als warmstartend bezeichnen. Ziel ist es, möglichst effizient (optimale) Lösungen zu produzieren. Ein wesentlicher Vorteil dieses Verfahrens wird es

sein, dass wir nur einmal im Wurzelknoten beginnen und im weiteren Verlauf auf einer tieferen Ebene im Suchbaum starten können. Den bereits angesprochenen Algorithmus Exact-Replan wollen wir in Kapitel 4 näher erläutern. Dabei gehen wir speziell auf jene Details ein, welche zum Lösen eines Schnappschussproblems essentiell sind. Den Abschluss dieser Arbeit bildet die Anwendung der in Kapitel 3 gewonnen Erkenntnisse auf den Algorithmus Exact-Replan. Dazu werden wir im Verlauf dieses Abschnittes u. a. die spezielle Struktur der *unteren Schranke der reduzierten Kosten* der Knoten in den Suchbäumen ausnutzen und eine neue *Verzweigungsstrategie* präsentieren, sodass im Vergleich zum Exact-Replan bis zu 88 % weniger Knoten erzeugt werden, die gesamte Rechenzeit auf den schweren Verkehrsmustern auf bis zu 15 % reduziert und trotzdem Optimalität gewährleistet werden kann.

Inhaltsverzeichnis

1 Grundlagen und Definitionen **1**

2 Einführung in die Steuerung von Aufzugsgruppen **5**
2.1 Hintergrundinformationen 5
2.2 Schnappschussproblem 8
2.3 Interaktion zwischen Fahrgast und Steuerungsalgorithmus 10

3 Branch & Bound-Verfahren **11**
3.1 Verzweigungsstrategien 13
3.2 Regeln zur Knotenauswahl 14
3.3 Ein warmstartendes Branch & Bound-Verfahren 15
3.3.1 Konstruktion eines Ähnlichkeitsmaßes 16
3.3.2 Modifikation und Sicherung der Korrektheit 20

4 Der Exact-Replan-Algorithmus **25**
4.1 Das Set-Partitioning-Model 25
4.2 Berechnung einer Startlösung 26
4.3 Spaltengenerierung via Branch & Bound 27
4.3.1 Die Lasdon Schranke 27
4.3.2 Konstruktion des Suchbaumes 28
4.3.3 Klassifikation der Touren und Knoten 31

5 Der warmstartende Exact-Replan **35**
5.1 Die Grundversion des Warmstartenden Exact-Replan 35
5.2 Untere Schranke der reduzierten Kosten 41
5.3 Das Pseudo-Branching 43
5.4 Rechnerische Auswertung 49

A Testgebäude und Szenarien **59**

B Tabellen und Diagramme **61**
B.1 Ähnlichkeit von Branch & Bound-Bäumen 61
B.2 Exact-Replan vs. Exact-Replan+WS 66
B.3 Untere Schranke der reduzierten Kosten 81
B.4 Pseudo-Branching 83
B.5 Exact-Replan vs. Exact-Replan+WS+fastLB+PB 87

Literaturverzeichnis **101**

Kapitel 1

Grundlagen und Definitionen

Zu Beginn dieser Arbeit wollen wir grundlegende Definitionen und notwendige Zusammenhänge darlegen, welche im weiteren Verlauf dieser Arbeit eine wesentliche Rolle spielen.

Gemischt ganzzahlige Programme (MIP)

Definition 1.1 (Gemischt ganzzahliges Programm)**:**
Seien $m, n \in \mathbb{N}$, $\mathcal{Z}$, $\mathcal{R} \subseteq \mathcal{I}$ eine Partition der Variablenindexmenge $\mathcal{I} := \{1, \ldots, n\}$, $A \in \mathbb{R}^{m \times n}$, $c \in \mathbb{R}^n$ und $b \in \mathbb{R}^m$. Ein gemischt ganzzahliges Programm P lässt sich wie folgt definieren:

$$\begin{aligned} (P) \qquad \min c^T x & \\ Ax &\geq b \\ x_i &\in \mathbb{Z} \quad \forall i \in \mathcal{Z} \\ x_i &\in \mathbb{R} \quad \forall i \in \mathcal{R} \end{aligned} \tag{1.1}$$

Eine Variable x_i, mit $i \in \mathcal{Z}$, bezeichnen wir als *ganzzahlige* und jede Variable x_j, mit $j \in \mathcal{R}$, als *kontinuierliche* Variable. MIPs lassen sich anhand ihrer ganzzahligen und kontinuierliche Variablen klassifizieren. Gilt $\mathcal{R} = \mathcal{I}$, so gibt es ausschließlich kontinuierliche Variablen und wir sprechen von einem *linearen Programm* (LP). Sind hingegen alle Variablen ganzzahlig, $\mathcal{Z} = \mathcal{I}$, so bezeichen wir dies als ein *ganzzahliges Programm* (IP). Andernfalls sprechen wir von einem MIP.

Definition 1.2:
Gegeben sei ein LP gemäß der Definition 1.1. Wir sagen ein LP befindet sich in Standardform, falls

$$\begin{aligned} (P) \qquad \min c^T x & \\ Ax &= b \\ x_i &\in \mathbb{R}_+ \quad \forall i \in \mathcal{R} \end{aligned} \tag{1.2}$$

gilt.

Desweiteren können wir davon ausgehen, dass sich jedes LP in Standardform überführen lässt. Empfehlenswerte Literatur zu diesem Thema ist das Buch *"Linear Programming"* von Vasek Chvatal [Chv83].

Im weiteren Verlauf betrachten wir stets LPs in Standardform. Für eine einfachere Notation führen wir das Folgende ein:

- J bezeichne eine Teilmenge der Spaltenindizes $\{1, \ldots, n\}$, so bezeichnet $A_J \in \mathbb{R}^{m \times J}$ die auf die zu J korrespondierenden Spalten eingeschränkte Matrix.
- Sind B und N eine Partitionierung der Spaltenindizes, so kann $x \in \mathbb{R}^n$ auch als $(x_B, x_N) \in \mathbb{R}^B \times \mathbb{R}^N$ aufgefasst werden, wobei die Variablen entsprechend dem Paar (B, N) permutiert sind. Analog kann $A = \begin{pmatrix} A_B & A_N \end{pmatrix}$ geschrieben werden und damit insbesondere
$$Ax = \begin{pmatrix} A_B & A_N \end{pmatrix} \begin{pmatrix} x_B \\ x_N \end{pmatrix} = A_B x_B + A_N x_N.$$

Ein LP der Form (1.2) bezeichnen wir als *primales* Problem. Das zugehörige *duale* Problem ist gegeben durch

$$\begin{aligned} (D) \qquad & \max \pi^T b \\ & \pi^T A \leq c^T \\ & \pi_i \in \mathbb{R} \quad \forall i \in \mathcal{R} \end{aligned} \tag{1.3}$$

Definition 1.3 (reduzierte Kosten und Dualpreise)**:**
Gegeben sei ein LP in Standardform und A_B eine Basis (d. h. $B \subseteq \{1, \ldots, n\}$, $|B| = m$, sodass $A_B \in \mathbb{R}^{m \times m}$ regulär ist) von A. Der Vektor

$$\tilde{c}^T = c_N^T - c_B^T A_B^{-1} A_N$$

heißt reduzierte Kosten (bzgl. B). Die Komponenten $\tilde{c}_i$ von $\tilde{c}$ heißen reduzierte Kostenkoeffizienten.
$\pi^T = c_B^T A_B^{-1}$ bezeichnet den Dualpreisvektor und π_i den Dualpreis der i-ten Nebenbedingung.

Spaltengenerierung

Diese kurze Einführung in das Konzept der Spaltengenerierung ist ein Auszug aus dem Buch *"Column Generation"* von Guy Desauliers, Jacques Desrosiers und Maurice M. Solomon [DDS05] und soll lediglich die dahinterstehende Grundidee verdeutlichen.
Die Spaltengenerierung kommt immer dann zum Einsatz, wenn ein LP zu groß ist um es explizit zu beschreiben, d. h. es besitzt eine sehr große Anzahl an Variablen, wobei die Anzahl der Nebenbedingungen vergleichsweise gering ist. Lösen wir das LP mit einer Version des Simplex-Algorithmus, so sind die meisten Variablen eines LPs keine Basisvariablen und tragen mit dem Wert 0 nicht zum Zielfunktionswert bei. Nur ein kleiner Teil der Variablen reicht aus um das Problem optimal zu lösen. Wir betrachten wieder das Problem (1.2) mit $A = (a_{\cdot i})_{i \in \mathcal{I}}$

$$\begin{aligned} \min \sum_{i \in I} c_i x_i & \\ \sum_{i \in I} a_{\cdot i} x_i &= b \\ x_i &\geq 0 \quad \forall i \in \mathcal{I}. \end{aligned} \tag{1.4}$$

Die Schreibweise soll verdeutlichen, dass wir uns speziell für die Spalten $a_{\cdot i}$ interessieren. Wir nehmen an dieser Stelle an, dass die Variablenindexmenge $\mathcal{I}$, welche benötigt wird, um das Polytop explizit zu beschreiben, im Vergleich zur Anzahl der Nebenbedingungen sehr groß ist, jede Zeile der Matrix A korrespondiert zu einer Nebenbedingung. Wollen wir das Problem (1.4) mit dem Simplex-Algorithmus lösen, so beginnen wir mit einer Basis A_B und erhalten einen entsprechenden Zielfunktionswert. Das Ziel ist es nun den Zielfunktionswert durch die Aufnahme eines geeigneten $j \in N \subset \mathcal{I}$ in die Spaltenindexmenge B der Basis zu verbessern. Ein geeigneter Spaltenindex $j \in N$ ist ein Index mit negativen reduzierten Kosten, $\tilde{c}_j = c_j - \pi^T a_j < 0$. Da diese Suche für ein sehr großes $\mathcal{I}$ teuer ist, ist es die Idee der Spaltengenerierung, mit einer weitaus kleineren Menge $\tilde{\mathcal{I}}$ von Spaltenindizes zu starten. Das ursprüngliche Problem (1.4) bezeichnen wir deshalb als das *Master-Problem* (MP) und das Problem

$$\begin{aligned} \min \sum_{i \in \tilde{I}} c_i x_i & \\ \sum_{i \in \tilde{I}} a_{\cdot i} x_i &= b \\ x_i &\geq 0 \quad \forall i \in \tilde{\mathcal{I}} \subset \mathcal{I}, \end{aligned} \tag{1.5}$$

welches zunächst eine weitaus kleine Spaltenindexmenge $\tilde{\mathcal{I}}$ besitzt, als das *reduzierte Master-Problem* (RMP). In dieser bedeutend kleineren Indexmenge können die reduzierten Kosten mithilfe einer Enumeration sehr viel schneller berechnet werden. Nehmen wir an, dass π und x die optimalen dualen bzw. primalen Lösungen des momentanen RMPs sind. Des Weiteren sei $\mathcal{A}$ die implizit gegebene Menge von Spalten, welche sich noch nicht im RMP befinden. Eine Lösung des Teilproblems

$$\tilde{c} := \min\{\ c(\tilde{a}) - \pi^T \tilde{a} \,|\, \tilde{a} \in \mathcal{A}\ \} \tag{1.6}$$

löst das zuvor erwähnte Problem der Suche eines geeigneten Spaltenindex. Ist $\tilde{c} \geq 0$, d. h. es existieren keine $\tilde{c}_i < 0$, $i \in \mathcal{A}$, so stellt x eine optimale Lösung des RMPs und auch des MPs dar. Ist dies nicht der Fall, so fügen wir die Spalte a dem reduzierten Problem hinzu und nehmen den zugehörigen Spaltenindex in $\tilde{\mathcal{I}}$ auf. Anschließend wiederholen wir diesen Schritt, bis wir eine beweisbar optimale Primal- bzw. Duallösung x bzw. π erhalten haben. Das Lösen von (1.6) hängt stark vom zugrunde liegendem Problem ab und kann – wie wir im späteren Verlauf dieser Arbeit sehen – u. a. mit einem Branch & Bound-Verfahrens gelöst werden.

Korollar 1.4 ([DDS05])**:**
Ist $|\mathcal{A}| < \infty$, so liefert die Spaltengenerierung stets eine exakte Lösung.

Kapitel 2

Einführung in die Steuerung von Aufzugsgruppen

Dieses Kapitel soll dazu dienen eine kurze Einführung in die Steuerung von Aufzugsgruppen zu geben. Dabei wollen wir zuerst grundlegende und allgemeine Informationen geben, wie z. B. die verschiedenen Verkehrstypen und die Anforderungen an die Aufzüge von Seiten der Passagiere. Dabei beziehen wir uns auf die Dissertation *"Online Optimization: Probabilistic Analysis and Algorithm Engineering"* von Benjamin Hiller [Hil10] sowie auf das Buch von Gina Carol Barney *"Elevator Traffic Handbook: Theory and Practice"* [Bar02]. Anschließend führen wir das Konzept eines Schnappschussproblems und Fahrpläne ein und diskutieren deren Zulässigkeit. Das Prinzip des Schnappschnussproblems wird uns auch im nächsten Kapitel begegnen, denn dieses bildet die Grundlage für den Exact-Replan-Algorithmus [Hil10].

2.1 Hintergrundinformationen

Im Laufe dieses Abschnittes werden wir immer wieder *konventionelle Systeme* und *Systeme mit Zielrufsteuerung* gegenüberstellen. Bei einem konventionellen hat der Fahrgast lediglich die Möglichkeit die gewünschte Fahrtrichtung, nach oben oder unten, zu wählen. Ein System mit Zielrufsteuerung hingegen verlangt, anstatt der Fahrtrichtung, die gewünschte Zieletage und gibt dem Passagier eine Rückmeldung, welchen Aufzug er benutzen soll, um auf seine gewünschte Zieletage zu gelangen.

Jeder Fahrgast löst durch die Wahl der Fahrtrichtung oder Zieletage einen *Ruf* aus. In einem konventionellen System unterscheiden wir zwischen *Außen-* und *Innenrufen.* Ein Außenruf ist die Wahl der Fahrtrichtung und der Innenruf die Wahl der Zieletage nach dem Betreten des Aufzuges. In einem zielrufgesteuertem System gibt es nur eine Art von Ruf, welcher bei der Ankunft des Passagiers ausgelöst wird und sowohl die Fahrtrichtung als auch das Ziel der Fahrt beinhaltet.
In einem konventionellen System bestimmt der Steuerungsalgorithmus den Aufzug, der der Startetage eines noch nicht zugewiesenen Rufes am nächsten ist und dieselbe Fahrtrichtung besitzt, und weißt diesem den Ruf zu. Der Aufzug bedient in einem solchen System zuerst alle Rufe, welche die gleiche Fahrtrichtung haben wie er selbst und die von ihm erreicht werden können. Danach werden Rufe in entgegengesetzter Richtung bedient.

Weisen wir jeden Ruf auf diese Art zu, so ist dies ein sehr natürlicher Ansatz. Ein wesentlicher Vorteil ist die Beschränkung der Wartezeit, da wir genau abschätzen können wann ein Aufzug wieder an einer bestimmten Etage vorbei kommt. Nach [Bar02] lässt sich mit dieser Methode der Planung die Qualität des Services am besten einschätzen. Die kleine aber entscheidende Information, welche einem System mit Zielrufsteuerung zur Verfügung steht, ist die Zieletage, welche vor dem Betreten des Aufzuges angegeben wird. Durch diesen Informationsgewinn ist es möglich, die noch wartenden Passagiere so auf die Aufzüge zu verteilen, dass möglichst wenige Stopps entstehen und die Summe der Warte- und Reisezeiten verringert werden könnte.

In dieser Arbeit haben wir ausschließlich Bürogebäude untersucht. Unter einer Zugangsebene verstehen wir eine Etage über die das Gebäude betreten bzw. verlassen werden kann, dazu zählen u. a. Parkdecks und der Haupteingang von der Straße. Für diese Gebäude gibt es drei Grundtypen von Verkehrsmustern. Zusätzlich existieren noch drei weitere Verkehrsmuster, welche Mischformen dieser Grundtypen sind. Eine tabellarische Übersicht über die Grundtypen und die Zusammensetzung der Mischformen ist in Anhang A zu finden. Nichtsdestotrotz wollen wir eine zeitliche Einteilung der Verkehrsmuster geben.

Grundtypen

1) *Up Peak* – Dies ist der morgendliche Verkehr. Die Passagiere kommen alle auf einer der Zugangsebenen an und fahren von dort aus zu ihrer Zieletage. Dies ist nach [Bar02] das anspruchsvollste Verkehrsmuster.

2) *Down Peak* – Dies ist der Verkehr in den Abendstunden. Die Fahrgäste verlassen ihre Büros und fahren mit einem Aufzug zu einer der Zugangsebenen und verlassen das Gebäude.

3) *Interfloor* – Der Pendelverkehr zwischen den späten Morgenstunden und der Mittagszeit, sowie zwischen der Mittagszeit und den frühen Abendstunden. Die Menschen verlassen für kurze Zeit ihre Büros und nutzen die Aufzüge um ihren Tätigkeiten nachzugehen.

Mischformen

Für eine realistische Simulation des Verkehrs innerhalb eines Bürotages benötigen wir Mischformen, so z. B. der *Real Up Peak*. In diesem Verkehrsmuster beginnt nicht jeder Fahrgast seine Fahrt auf einer Zugangsebene, es gibt auch einige wenige Passagiere, welche zu einer Zugangsebene wollen oder sich innerhalb des Gebäudes bewegen. Analog der *Real Down Peak*. Die letzte Mischform ist der *Lunch Peak*, dieses Verkehrsmuster stellt den Verkehr zu den Pausenzeiten dar. Die Mitarbeiter fahren zu Beginn dieser Phase entweder zu den Zugangsebenen und verlassen das Gebäude oder sie fahren innerhalb des Gebäudes um z. B. die Kantine zu erreichen. Am Ende dieser Phase begeben sich alle Mitarbeiter wieder zu ihren Büros. Dieser Verkehrstyp ist demnach eine Mischung aus allen drei Grundtypen.

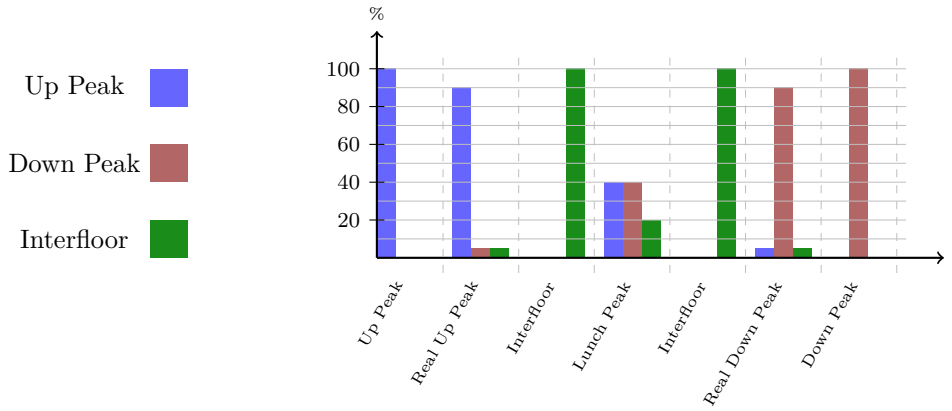


Abbildung 2.1: Darstellung eines möglichen Tagesablaufes der Verkehrsmuster und deren Zusammensetzung. Der Tag beginnt mit einem Up Peak Szenario, gefolgt von einer Real Up Peak Phase. Es schließt sich eine Interfloor Phase an, gefolgt vom Lunch Peak. Nach der Mittagspause erfolgt wieder der Interfloor-Verkehr. Die Abendstunden beginnen mit einem Real Down Peak Verkehrsmuster, gefolgt vom Down Peak.

Wie man nun leicht sieht, gibt es während des Up Peak Verkehrs für ein konventionelles System keine Möglichkeiten die Zuteilung der ankommenden Passagiere zu beeinflussen. Der Algorithmus kann lediglich die Haltezeit an einer der Zugangsebenen verlängern, sodass es zu einer besseren Auslastung der Fahrgastkabine kommen könnte. Aufzugsgruppen mit einer Zielrufsteuerung hingegen haben den Vorteil, dass das System die abzuarbeitenden Aufträge anhand ihrer Zieletagen auf die Aufzüge verteilen kann, sodass die Gesamtzahl der Stopps und damit auch die Summe der Warte- und Reisezeiten möglichst gering bleibt.

Wir wollen nun eine Übersicht über die zu beachtenden Anforderungen bei der Konstruktion der Fahrpläne geben. Diese Richtlinien können technischer Natur sein, Sicherheitsanforderungen, z. B. die Kapazität der Fahrgastkabine, oder auch die Erwartung des Fahrgastes.

Die folgenden natürlichen Verhaltensweisen eines Passagiers setzen wir, auch begründet durch unsere eigenen Erfahrungen, voraus. Nämlich, dass ein Fahrgast niemals einen Fahrstuhl betritt, welcher in die entgegengesetzte Fahrtrichtung fährt, er den Aufzug verlässt, sobald dieser die gewünschte Zieletage erreicht hat, und er seinen Ruf erneut tätigt, falls der ankommende Fahrstuhl nicht über ausreichende Kapazitäten verfügt. Aus diesen Verhaltensweisen der Passagiere können wir nun Anforderungen an den zu erstellenden Fahrplan ableiten. Im Weiteren unterscheiden wir zwischen zwei Arten von Stopps. Wir bezeichnen alle Stopps in denen Fahrgäste zusteigen als *Beladestopps* und alle Stopps in denen Fahrgäste den Aufzug verlassen als *Entladestopps*. Zusammenfassend halten wir fest:

(1) Ein Fahrstuhl nimmt niemals einen Fahrgast auf, welcher in die zur Fahrtrichtung entgegengesetzte Richtung fahren will.

(2) Ein Fahrstuhl muss alle seine geladenen Fahrgäste an ihren gewünschten Zieletagen abliefern, bevor er seine Fahrtrichtung ändern und Fahrgäste mit einer anderen Fahrtrichtung aufnehmen kann. Die Änderung der Fahrtrichtung erfolgt beim letzten Entladestopp.

(3) Jeder Passagier tätigt bei seiner Ankunft einen Ruf. Für diesen Ruf muss ein entsprechender Be- und Entladestopp existieren.

(4) Jeder Stopp ist ein Be- und/oder Entladestopp.

(5) Ist die Kapazität der Fahrgastkabine vollständig ausgeschöpft, so ist der nächste Stopp stets ein Entladestopp.

(6) Hat ein Fahrstuhl alle seine ihm zugeordneten Aufträge abgearbeitet, so verbleibt er auf der Etage des letzten Entladestopps.

Für eine detailliertere Beschreibung der Richtlinien und Anforderungen sei auf [Hil10] verwiesen.

2.2 Schnappschussproblem

In diesem Abschnitt, und auch in allen folgenden, betrachten wir ausschließlich verzögert zuweisende Zielrufsysteme. Wir wollen kurz erläutern was [Hil10] und wir unter einem Schnappschussproblem verstehen und behalten die Notation aus [Hil10] bei. Als ein Schnappschussproblem bezeichnen wir eine Momentaufnahme des aktuellen Verkehrs in dem Gebäude.

In einem Schnappschussproblem betrachten wir nicht mehr jeden Ruf separat, sondern fassen alle Passagiere und damit alle Rufe, welche die gleiche Start- und Zieletage haben zu sogenannten *Aufträgen* zusammen. Jede dieser Aufträge impliziert daher genau einen Entladestopp.

In einem Schnappschussproblem bezeichne $\mathcal{R}$ die Menge aller Aufträge. Wir unterscheiden innerhalb von $\mathcal{R}$ zwischen den Aufträgen $\mathcal{R}(e)$, diese wurden dem Aufzug e bereits zugewiesen, und den Aufträgen $\mathcal{R}_u$, diese müssen durch den Algorithmus auf die Aufzüge verteilt werden. Für einen Aufzug e bezeichnet $\mathcal{F}(e)$ die Menge aller Etagen, welche mit einem Entladestopp eines geladenen Passagiers korrespondieren. Ziel des Steuerungsalgorithmus ist es die Aufträge $\mathcal{R}_u$ auf die Aufzüge zu verteilen, sodass außerdem alle Entladestopps $\mathcal{F}(e)$ erfüllt werden. Die Menge aller zulässigen ersten Stopps, innerhalb einer solchen Tour, bezeichnen wir mit $\mathcal{F}_i(e)$ und die aktuelle Etage auf der der Fahrstuhl hält sei $f_0(e)$. Betrachten wir die Menge $\mathcal{E}$ aller Aufzüge, so können wir $\mathcal{F}(\mathcal{E}) := \{\mathcal{F}(e_1), \ldots, \mathcal{F}(e_{|\mathcal{E}|})\}$ definieren. Eine solche Lösung des Schnappschussproblems nennen wir Fahrplan.

Wir wollen an dieser Stelle noch auf die Zulässigkeitsbedingungen eines Fahrplanes, welchen wir am Ende jedes Lösungsprozesses eines Schnappschussproblems erhalten, eingehen. Dieser Fahrplan besteht aus den Touren der jeweiligen Fahrstühle. Für jeden Aufzug e bezeichne die aus den Stopps s_i, $0 \leq i \leq k$, bestehende Sequenz, $t = (s_0, \ldots, s_k)$ eine solche Tour. Dabei ist jeder Stopp s_i durch seine Zieletage ($s_i.\mathit{floor}$), die Fahrtrichtung ($s_i.\mathit{direction}$), die Menge der aktuell geladenen Rufe ($s_i.\mathit{current\_calls}$) sowie die Menge der aus- bzw. einzuladenden Fahrgäste ($s_i.\mathit{drops}$ bzw. $s_i.\mathit{pickups}$) und den folgenden Entladestopps ($s_i.\mathit{drop\_floors}$) charakterisiert. Des Weiteren bezeichne $\mathcal{D}(e)$ die Menge aller zulässigen Fahrtrichtungen nach dem ersten Stopp. Betrachten wir die Menge aller Aufzüge, so können wir auch hier $\mathcal{D}(\mathcal{E}) := \{\mathcal{D}(e_1), \ldots, \mathcal{D}(e_{|\mathcal{E}|})\}$ definieren.

Definition 2.1 (Zulässigkeit von Stopps [Hil10])**:**
Der erste Stopp eines Fahrplans heißt zulässig, falls

- $s_0.floor \in \mathcal{F}_i(e)$
- $s_0.direction \begin{cases} \in \mathcal{D}(e) & \text{, falls } s_0.floor = f_0(e) \\ = \text{ beliebig} & \text{, falls } \mathcal{F}(e) \text{ leer ist.} \end{cases}$
- $s_0.drop_floors = \mathcal{F}(e) \setminus s_0.floor \cup \{$Entladestopps, die durch den Halt auf dieser Etage entstehen$\}$.

Alle späteren Stopps müssen die folgenden Eigenschaften erfüllen, damit sie zulässig sind:

- Für die Entladestopps gibt es zwei Fälle:
  $s_i.drop\_floors \neq \emptyset$: $s_{i+1}.floor$ befindet sich zwischen $s_i.floor$ und der nächsten Etage in $s_i.drop\_floors$, welcher in der gleichen Fahrtrichtung liegt
  $s_i.drop\_floors = \emptyset$: $s_{i+1}.floor$ sowie die Fahrtrichtung können beliebig gewählt werden.
- $s_{i+1}.drop_floors = s_i.drop_floors \setminus s_{i+1}.floor \cup \{$Entladestopps, die durch den Halt auf dieser Etage entstehen$\}$.

Befindet sich ein Aufzug gerade auf der Fahrt von einer Etage zu einer anderen und ist dieser voll, so besteht $\mathcal{F}_i(e)$ lediglich aus dem nächstgelegenem Entladestopp. Hält der Aufzug e gerade auf einer Etage, so gilt offensichtlich $\mathcal{F}_i(e) = \{f_0(e)\}$. Zu beachten ist außerdem, dass in jeder zulässigen Tour der zugehörige Aufzug nur zwei Zustände einnehmen kann: entweder er hält gerade auf einer Etage oder er befindet sich in Fahrt zwischen zwei Etagen. Das gleiche Prinzip gilt für die Fahrtrichtungen ausgehend von der aktuellen Stoppetage. Um die Zulässigkeit einer Tour definieren zu können benötigen wir noch die Menge aller Aufträge, welche bereits in den Fahrplan aufgenommen wurden, bis einschließlich Stopp s_i, diese sei mit $P(s_i)$ bezeichnet.

Definition 2.2 (Zulässigkeit einer Tour [Hil10])**:**
Eine Tour $t = (s_0, \ldots, s_k)$ heißt zulässig, falls

- jeder Stopp s_i, $0 \leq i \leq k$, zulässig ist,
- es keine zwei aufeinander folgenden Stopps auf der gleichen Etage mit identischer Fahrtrichtung gibt,
- jeder Aufzug die ihm zugewiesenen Aufträge abgearbeitet hat, d. h. $\mathcal{R}(e) \subseteq P(s_k)$ und
- alle Aufträge und Entladestopps bedient wurden.

Nachdem wir nun die Zulässigkeit eines Stopps und die der einzelnen Touren eingeführt haben widmen wir uns der eigentlichen Definition eines Schnappschussproblems.

Definition 2.3 (Schnappschussproblem)**:**
Sei S ein Fahrplan in dem die Menge von Aufzügen $\mathcal{E}$ die Aufträge $\mathcal{R} \setminus \mathcal{R}_u$ bedient. Desweiteren bezeichne $\mathcal{F}(e_i)$ die Menge der zulässigen ersten Stoppetagen und $\mathcal{D}(e_i)$ die Menge aller zulässigen ersten Fahrtrichtungen für Aufzug $e_i \in \mathcal{E}$.
Das Tupel $(S, \mathcal{E}, \mathcal{R}, \mathcal{F}(\mathcal{E}), \mathcal{D}(\mathcal{E}))$ heißt Schnappschussproblem. $(S', \mathcal{E}, \mathcal{R}', \mathcal{F}'(\mathcal{E}), \mathcal{D}'(\mathcal{E}))$ heißt Lösung von $(S, \mathcal{E}, \mathcal{R}, \mathcal{F}(\mathcal{E}), \mathcal{D}(\mathcal{E}))$, falls ein Fahrplan S', in dem jeder Auftrag von genau einer Tour bedient wird, existiert und sowohl $\mathcal{R}' = \mathcal{R}$ als auch $\mathcal{R}'_u = \emptyset$ gilt.

2.3 Interaktion zwischen Fahrgast und Steuerungsalgorithmus

In einem konventionellem System wird jeder Ruf einzeln behandelt, d. h. jeder Auftrag entspricht genau einem Ruf. Die Ankunftszeit dieses Auftrages ist damit auch die Ankunftszeit des Rufes. Es existiert kein Ziel und die Fahrtrichtung des Auftrages ist über die vom Fahrgast gewählte Richtung (Pfeiltasten) gegeben. In [Hil10] werden für die Zielrufsysteme unmittelbar zuordnende (UZ-)Systeme und verzögert zuordnende (VZ-)Systeme untersucht.

Zu Beginn dieses Kapitels haben wir vorausgesetzt, dass ein Fahrgast stets den zuerst ankommenden Aufzug, welcher in die korrekte Fahrtrichtung fährt betritt. In einem UZ-System wählt der Passagier stets den ihm vom System zugewiesenen Aufzug. In einem VZ-System wählt der Passagier den ersten ankommenden Aufzug mit korrekter Fahrtrichtung, welcher an seiner gewünschten Zieletage hält.

UZ-Systeme In einem unmittelbar zuweisenden System werden alle noch nicht eingeladenen Rufe einer Etage, welche die gleiche Fahrtrichtung besitzen und im vorherigen Schnappschussproblem bereits einem Aufzug zugewiesen wurden, zu einem Auftrag zusammengefasst. Diese Aufträge müssen auch im aktuellen Schnappschussproblem von dem entsprechenden Aufzug bedient werden. Wartende Rufe, welche noch keinem Aufzug zugewiesen sind bilden jeweils einen separaten Auftrag und können frei auf die Aufzüge verteilt werden.

VZ-Systeme Ein verzögert zuweisendes System fasst Rufe mit gleicher Start- und Zieletage zu einem Auftrag zusammen. Aufträge, welche im vorherigen Schnappschussproblem einem Aufzug zugewiesen, jedoch noch nicht eingeladen, wurden, können im aktuellen Schnappschussproblem wieder frei auf die Aufzüge verteilt werden.

Kapitel 3

Branch & Bound-Verfahren

Bei der exakten Lösung $\mathcal{NP}$-schwerer Probleme – insbesondere bei der Lösung von IPs – stellen Branch & Bound-Verfahren eine Standardtechnik dar. Ein solches Verfahren ist eine geschickte und systematische Durchsuchung der (gesamten) Lösungsmenge nach einer Optimallösung. Für ein gegebenes Problem

$$(P) \qquad \min c(x) \qquad x \in \mathcal{P}^0, \quad |\mathcal{P}^0| < \infty$$

sei $\mathcal{P}^0$ der vollständige Lösungsraum, welcher implizit gegeben ist. Für jede Teilmenge $\mathcal{P} \subseteq \mathcal{P}^0$ liefert die Funktion

$$heur_{\mathcal{P}} \,:\, 2^{\mathcal{P}^0} \to \mathcal{P} \cup \{\perp\} \tag{3.1}$$

entweder eine (heuristisch konstruierte) zulässige Lösung $x^*_{\mathcal{P}}$ oder gibt an, dass die Konstruktion einer Lösung fehlgeschlagen ist. Ein Fehlschlag wird durch den Funktionswert $\perp$ repräsentiert. Des Weiteren sei durch die Funktion

$$lb_c \,:\, 2^{\mathcal{P}^0} \to \mathbb{R} \tag{3.2}$$

für jedes $\mathcal{P} \subseteq \mathcal{P}^0$ eine untere Schranke, in Abhängigkeit von der Zielfunktion c, gegeben. Allgemein lassen sich Branch & Bound-Verfahren wie folgt darstellen:

1. Initialisiere eine obere Schranke $U = \infty$ oder konstruiere eine zulässige Lösung x^* von $\mathcal{P}^0$ und setze $U = c(x^*)$.

2. Die Menge der Teilprobleme bezeichnen wir mit K und setzen $K = \{\mathcal{P}^0\}$.

3. $K = \emptyset \to$ Abbruch und gib die obere Schranke U und, falls $U < \infty$, die Optimallösung x^* zurück, ansonsten wähle $\mathcal{P} \in K$.

4. $lb_c(\mathcal{P}) \geq U \to$ keine Lösung in $\mathcal{P}$ ist besser als die bisher beste Lösung. Entferne $\mathcal{P}$ aus K. Gehe zu Schritt 3.

5. $lb_c(\mathcal{P}) < U$:

   (a) $heur(\mathcal{P}) = \perp$: Partitioniere $\mathcal{P}$ in (geeignete) Teilprobleme $\mathcal{P}^1, \ldots, \mathcal{P}^k$ und füge diese zu K hinzu. Entferne $\mathcal{P}$ aus der Menge der Teilprobleme. Gehe zu Schritt 3.

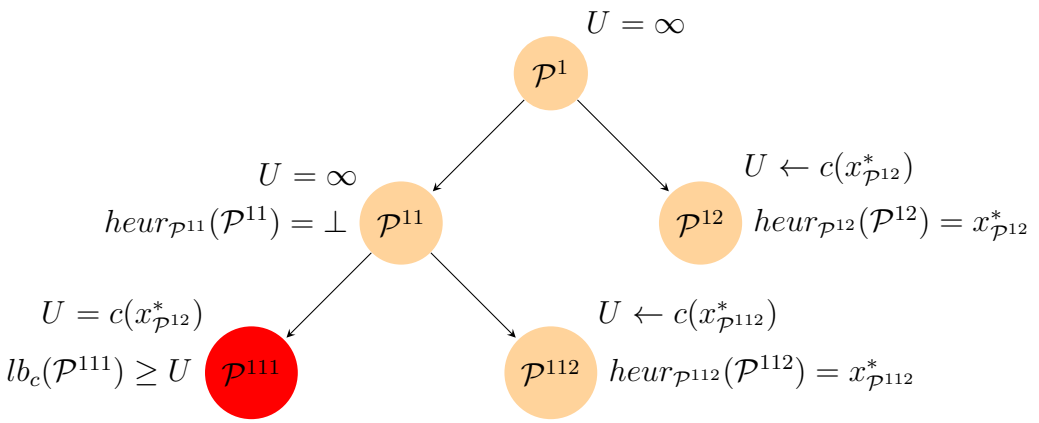


Abbildung 3.1: In diesem Suchbaum wird das Problem $\mathcal{P}^1$ in zwei Teilprobleme $\mathcal{P}^{11}$ und $\mathcal{P}^{12}$ aufgeteilt. Die Funktion *heur* liefert für das Problem $\mathcal{P}^{11}$ keine Lösung. Das Problem muss gemäß Schritt 5a partitioniert werden. Die Teilprobleme $\mathcal{P}^{111}$ und $\mathcal{P}^{112}$ werden der Menge der Teilprobleme K hinzugefügt. Für das Teilproblem $\mathcal{P}^{12}$ liefert die Funktion *heur* eine zulässige Lösung $x^*_{\mathcal{P}^{12}}$ für die $lb_c(\mathcal{P}^{12}) = c(x^*_{\mathcal{P}^{12}})$ gilt. Wir haben somit eine bessere Lösung gefunden. Im Anschluss betrachten wir $\mathcal{P}^{111}$: Die untere Schranke $lb_c(\mathcal{P}^{111})$ ist nicht besser als die globale obere Schranke U, das Problem kann ausgelotet werden. Die Teilmenge besteht nur noch aus $\mathcal{P}^{112}$, dieses liefert eine weitere Verbessung der oberen Schranke. $K = \emptyset$ und die Optimallösung lautet $x^*_{\mathcal{P}^{112}}$ mit dem Zielfunktionswert $c(x^*_{\mathcal{P}^{112}})$.

(b) $heur(\mathcal{P}) = x^*_{\mathcal{P}}$: Wir haben eine zulässige Lösung gefunden. Aktualisiere $U = c(x^*_{\mathcal{P}})$ und $x^* = x^*_{\mathcal{P}}$. Es muss sichergestellt werden, dass keine Lösung $x' \in \mathcal{P}$ existiert, sodass $c(x^*_{\mathcal{P}}) > c(x')$ gilt.

   i. $c(x^*_{\mathcal{P}}) = lb_c(\mathcal{P}) \rightarrow x^*_{\mathcal{P}}$ ist die beste in $\mathcal{P}$ enthaltene Lösung. Entferne $\mathcal{P}$ aus K. Gehe zu Schritt 3.

   ii. $c(x^*_{\mathcal{P}}) > lb_c(\mathcal{P}) \rightarrow$ Wir können aus dieser Lösung keine Schlussfolgerungen ziehen. Partitioniere $\mathcal{P}$ in (geeignete) Teilprobleme $\mathcal{P}^1, \ldots, \mathcal{P}^k$ und füge diese zu K hinzu. Entferne $\mathcal{P}$ aus der Menge der Teilprobleme. Gehe zu Schritt 3.

Das Entfernen eines Teilproblems $\mathcal{P}$ in Schritt 4 und 5 bezeichnen wir im Folgenden als *Ausloten*.

Definition 3.1:
Seien $\mathcal{P}', \mathcal{P}'' \subseteq \mathcal{P}^0$ zwei Lösungsräume und P', P'' die dazugehörigen Probleme. Wir sagen das Problem P'' ist von dem Problem P' abgeleitet, falls der Lösungsraum $\mathcal{P}''$ von P'' eine Teilmenge des Lösungsraumes $\mathcal{P}'$ von P' ist, d. h. $\mathcal{P}'' \subset \mathcal{P}'$.

Die Menge aller Teilprobleme, welche im Laufe des Lösungsprozesses von P erzeugt werden, kann als Knotenmenge V eines gerichteten Baumes T mit Wurzelknoten P interpretiert werden. Dieser Wurzelknoten kann von jedem Teilproblem P^i aus über einen eindeutigen Pfad erreicht werden. Diesen Baum bezeichnen wir im Folgenden als Branch & Bound-Baum oder Suchbaum. Zwischen zwei Knoten v und w existiert genau dann eine Kante $e = \{v, w\}$, wenn $\mathcal{P}^v \subset \mathcal{P}^w$ oder $\mathcal{P}^w \subset \mathcal{P}^v$ gilt. Die Gesamtheit aller Kanten in T bezeichnen wir mit E. Reden wir in Zukunft von einem Knoten v, so ist damit zugleich – in Abhängigkeit vom Kontext – das korrespondierende Problem P^v bzw. der Lösungsraum $\mathcal{P}^v$ gemeint.

Das am Anfang gefallene Schlagwort, um die Grundidee der Branch & Bound-Verfahren zu beschreiben, war „*geschickt*“. Dies bezieht sich auf die Partitionierung eines Problems in Teilprobleme und auf die Auswahl eines Knotens in K.

Wenn wir uns an die Punkte 5a und 5(b)ii aus dem Schema für Branch & Bound-Verfahren zurückerinnern, dann ist dort die Entscheidung zu treffen wie ein Problem in Teilprobleme partitioniert werden soll. Die Strategie nach der eine solche Entscheidung getroffen wird, bezeichnet man als *Verzweigungsstrategie*.

3.1 Verzweigungsstrategien

Definition 3.2 (Verzweigungsstrategie)**:**
Sei P ein kombinatorisches Optimierungsproblem mit einem endlichen Lösungsraum $\mathcal{P}$ und c eine Zielfunktion. Eine Abbildung

$$\mathfrak{V}_c : 2^{\mathcal{P}} \to 2^{2^{\mathcal{P}}},\ \mathcal{P}^i \mapsto \{\hat{\mathcal{P}}^1, \ldots, \hat{\mathcal{P}}^k\},$$

welche die Eigenschaften

1. $\hat{\mathcal{P}}^i \cap \hat{\mathcal{P}}^j = \emptyset$ für $i, j = 1, \ldots, k,\ i \neq j$
2. $\bigcup_{i=1}^{k} \hat{\mathcal{P}}^i = \mathcal{P}^i$

erfüllt, heißt Verzweigungsstrategie.

Bemerkung 3.3:
Es existieren auch Verzweigungsstrategien, die die Eigenschaft 1 nicht erfüllen. Da wir im weiteren Verlauf dieser Arbeit jedoch stets mit Verzweigungsstrategien arbeiten, welche paarweise disjunkte Mengen $\hat{\mathcal{P}}^i$ und $\hat{\mathcal{P}}^j$ erzeugen, wollen wir diese Eigenschaft fordern.

In der Literatur gibt es eine Vielzahl an Verzweigungsstrategien, wobei sich eine gute Verzweigungsstrategie dadurch auszeichnet, dass aus Lösungen der Teilprobleme neue obere und untere Schranken ermittelt werden.
Über Verzweigungsstrategien für ein ganz allgemeines Branch & Bound-Verfahren für ein beliebiges kombinatorisches Optimierungsproblem lässt sich nicht viel aussagen, da diese sehr stark vom zugrunde liegenden Problem abhängen. Wir wollen jedoch zwei Verzweigungsstrategien für einen typischen Anwendungsfall – das Lösen von gemischt ganzzahligen Programmen (MIPs) – von Branch & Bound-Algorithmen vorstellen.

Verzweigung auf franktionalster Variable

Unser zugrunde liegendes Problem ist ein MIP wie aus Kapitel 1 Definition 1.1. Mit $\mathcal{Z}$ bezeichnen wir wieder die Indexmenge aller ganzzahligen Variablen. Wie der Name schon sagt, wählen wir die Variable x_i^v mit $i \in \mathcal{Z}$ mit dem fraktionalsten Wert in der relaxierten LP-Lösung x_{LP}^v in Knoten v. Die Relaxierung entspricht dem Weglassen der Ganzzahligkeitsbedingung an alle Variablen x_i mit $i \in \mathcal{Z}$. Der Zielfunktionswert von x_{LP}^v stellt zugleich eine untere Schranke an den Zielfunktionswert des nicht-relaxierten

Problems dar. Sei x_i^v der Wert zur Variable i. Wir wählen also eine Variable i, sodass

$$i = \arg\min_{j \in \mathcal{Z}} \left\{ \left| \left| x_j^v - \lfloor x_j^v \rfloor \right| - \frac{1}{2} \right| \right\}$$

gilt. Eine Partitionierung des zum Knoten v gehörigen Lösungsraumes erfolgt durch Fixierung der i-ten Variable auf $\lceil x_i^v \rceil$ bzw. $\lfloor x_i^v \rfloor$. Ein großer Vorteil dieser Strategie ist ihre Einfachheit. Aus [AKM04] geht jedoch hervor, dass es hinsichtlich der Performance keinen Unterschied zur randomisierten Auswahl der Variablen gibt.

Strong Branching

Beim Strong Branching werden die Variablen versuchsweise verzweigt. Das heißt es wird diejenige Variable bestimmt, welche den größten Zuwachs im Zielfunktionswert bewirkt, bevor in diese verzweigt wird. Diese Strategie bewirkt eine deutliche Reduzierung der Knoten, jedoch ist das Testen sehr aufwändig, da im vollständigen Strong Branching die LP-Relaxierung bzgl. jeder Variable gelöst wird [AKM04]. Um den gesamten Prozess zu beschleunigen kann auch nur auf einer Teilmenge aller möglichen Variablen getestet werden. Ein anderer Ansatz ist, dass die LP-Relaxierung nicht vollständig gelöst wird, d. h. es werden nur ein paar Simplex-Iterationen durchgeführt, um anschließend eine Tendenz zu erkennen. Analog zur obigen Verzweigungsstrategie erfolgt die Partitionierung durch Fixierung der i-ten Variable auf die nächst kleine bzw. größere ganze Zahl.

3.2 Regeln zur Knotenauswahl

Blicken wir wieder zurück auf das anfangs erwähnte Schema, so ist neben der Entscheidung über die Art und Weise der Zerlegung eines Problems noch die Wahl des als nächstes zu untersuchenden Teilproblems zu fällen (Punkt 3 des Branch & Bound Schemas). Für eine solche Auswahlregel gibt es in der Literatur sehr viele Ansätze. Diese Regeln lassen sich u. a. in *statische* und *2-Phasen* Knotenauswahlregeln unterteilen. Zu den statischen Auswahlregeln zählt z. B. die Tiefensuche, welche sich auch unabhängig vom zugrunde liegenden Problem formulieren lässt.

Statische Knotenauswahl

Bei der Tiefensuche wird stets der Knoten ausgewählt, welcher zuletzt in die Menge der Teilprobleme aufgenommen wurde (LIFO, Last-In-First-Out). Bei diesem Vorgehen entwickelt sich der Suchbaum in der Regel mehr in die Tiefe. Ein Vorteil dieser Vorgehensweise ist, dass sich die aufeinanderfolgenden Probleme lediglich um eine oder sehr wenige Restriktion unterscheiden und somit „leichter“ zu lösen sind. Ein wesentlicher Nachteil hingegen ist, dass die Gefahr besteht, sehr viel Aufwand in Teilbäume zu investieren, welche keine Optimallösung enthalten.
Eine weitere statische Strategie ist die *Best-First-Regel* [Wau07]. So kann z. B. einer derjenigen Knoten ausgewählt werden, deren Vorgänger die größte Verbesserung der oberen Schranke lieferte. Im Gegensatz zur Tiefensuche wächst der Suchbaum in der Regel in die Breite. Eine unmittelbare Folge ist, dass die aufeinanderfolgenden Probleme keine wesentliche Ähnlichkeit aufweisen und somit „schwerer“ zu lösen sind. Die

bei der Tiefensuche auftretende Gefahr zu lange in einem Teilbaum zu suchen, welcher keine Optimallösung enthält, ist bei dieser Strategie wesentlich geringer.

2-Phasen-Regel [Wau07]

Das grundlegende Problem beim Branch & Bound ist ein Zielkonflikt: Zum Einen ist man an einer möglichst guten zulässigen Lösung interessiert und zum Anderen am Beweis, dass es keine bessere Lösung gibt. Um diesem Zielkonflikt entgegenzuwirken liegt eine Kombination der beiden bereits vorgestellten Regeln nah [Wau07]. Die Idee ist, zuerst mit der Tiefensuche eine möglichst gute zulässige Lösung zu suchen. Denn in der Tiefensuche weisen die aufeinanderfolgenden Probleme eine starke Ähnlichkeit auf. Die Hoffnung ist, dass diese Ähnlichkeit dazu führt, dass viele Probleme schnell gelöst werden können. Die Best-First-Regel folgt der Tiefensuche, durch diese sollen die Schranken verbessert und im Idealfall bewiesen werden, dass es keine bessere Lösung gibt.

Bis zu diesem Punkt haben wir stets ein Optimierungsproblem P mit einer festen Zielfunktion betrachtet. Erinnern wir uns an Kapitel 1 zurück, speziell an den Abschnitt über Spaltengenerierung, dann fällt auf, dass sich der komplette Lösungsraum $\mathcal{P}$ nicht verändert, die Zielfunktion des Teilproblems (1.6)

$$\min\{\ c(a) - \pi^T a \,|\, a \in \mathcal{A}\ \}$$

hingegen schon. Denn der Dualpreisvektor π ändert sich nach dem Hinzufügen einer neuen Spalte. Lassen sich die Spalten a mithilfe eines Branch & Bound-Verfahrens konstruieren, so stellt sich die Frage, ob es möglich ist Informationen aus vorherigen Suchbäumen zu nutzen, um die Konstruktion des momentanen Suchbaumes zu beschleunigen. Eine solche Wiederverwendung von Informationen bezeichnen wir als *Warmstart*. Ein warmstartendes Verfahren muss selbstverständlich eine genauso gute Lösung liefern, wie viele hintereinander ausgeführte nicht warmstartende Branch&-Bound-Verfahren.

3.3 Ein warmstartendes Branch & Bound-Verfahren

Im Folgenden betrachten wir wieder ein kombinatorisches Optimierungsproblem P mit dem endlichen Lösungsraum $\mathcal{P}^0$ sowie den Zielfunktionen $c_1, \ldots, c_n$:

$$(P) \quad \left.\begin{array}{l} \min c_i(x) \\ x \in \mathcal{P}^0 \end{array}\right\} \forall i \in \{1, \ldots, n\} \tag{3.3}$$

Für $\min_{x \in \mathcal{P}^0} c_i(x)$ wollen wir in Zukunft nur noch P_i schreiben.

Definition 3.4 (Abbruchbedingung)**:**
Eine Abbruchbedingung $\mathfrak{A}$ ist ein Entscheidungskriterium, welche die momentan beste Lösung eines Optimierungsproblem $\min_{x \in \mathcal{P}^0} c(x)$ als genügend gut verifiziert.

Wie der Name schon sagt wird der Lösungsprozess bei einer positiven Verifizierung beendet. Die zu den Problemen $P_1, \ldots, P_n$ aus (3.3) gehörige Abbruchbedingungen

seien gegeben durch $\mathfrak{A}_1, \ldots, \mathfrak{A}_n$, welche nicht zwangsläufig dem Beweis der Optimalität entsprechen. Beispielsweise kann die Anzahl erzeugter Knoten oder ein bestimmter Gap eine Abbruchbedingung darstellen. Mit $T_{i,\mathfrak{A}_i}$ bezeichnen wir im Folgenden den Suchbaum zu Problem P_i und Abbruchbedingung $\mathfrak{A}_i$. Im Weiteren Verlauf können wir davon ausgehen, dass einen Algorithmus existiert, welcher die Probleme $P_1, \ldots, P_n$ durch den iterativen Aufruf eines Branch & Bound-Verfahrens löst. Dieser Algorithmus übergibt dem Branch & Bound-Verfahren in der i-ten Iterationsrunde die Zielfunktion c_i, eine obere Schranke U und die initialisierte Menge der Teilprobleme, welche wir mit K^i bezeichnen.

Ein Warmstart macht natürlich nur dann Sinn, wenn die Suchbäume zweier aufeinanderfolgenden Probleme zu einem Großteil die gleichen Knoten und Ableitungsbeziehungen (siehe Definition 3.1) bzw. Verzweigungen besitzen. Diese (teilweise) Übereinstimmung wollen wir als *Ähnlichkeit* bezeichnen und im folgenden Abschnitt ein Maß konstruieren, welches diese misst.

3.3.1 Konstruktion eines Ähnlichkeitsmaßes

Bevor wir ein solches Maß für die Ähnlichkeit zweier Branch & Bound-Bäume konstruieren benötigen wir den Begriff des Pfades um die genaue Position eines Knotens innerhalb des Suchbaumes bestimmen zu können.

Definition 3.5 (Pfad)**:**
Sei $T = (V, E)$ ein Baum. Eine Folge $\sigma = (v_1, \ldots, v_n)$ von Knoten aus V mit den Eigenschaften

1. $v_i \neq v_j$ für alle $i \neq j$
2. $(v_i, v_{i+1}) \in E$ für alle $i \in \{1, \ldots, n-1\}$.

heißt *Pfad* in T. Für eine einfache Notation schreiben wir für die Sequenz $(v = v_1, \ldots, v_n)$ lediglich $\sigma(v)$, falls v_n der Wurzelknoten von T ist.

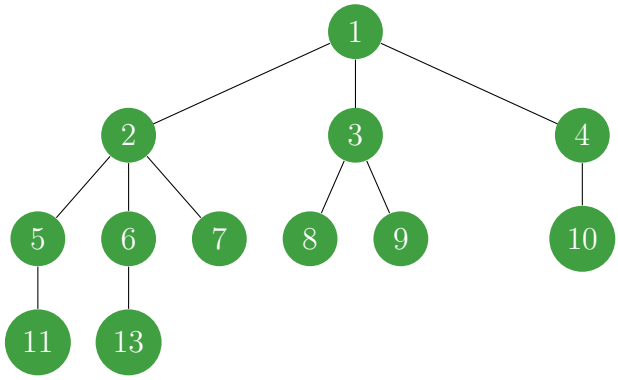
Am Ende des letzten Abschnittes wurde erwähnt, dass sich die Ähnlichkeit durch gemeinsame Knoten und Verzweigungen auszeichnet. Diese Ähnlichkeit lässt sich analog über den Pfad eines Knotens definieren.

Definition 3.6 (Ähnlichkeit von Bäumen)**:**
Seien T, T' zwei Bäume mit den jeweiligen Knotenmengen V und V'. Wir sagen T ist *ähnlich* zu T', falls eine nichtleere Knotenmenge $\tilde{V} \subseteq V$ existiert, sodass $\tilde{V} \subseteq V'$ mit $\sigma(v) \subseteq V' \, \forall v \in \tilde{V}$ gilt.

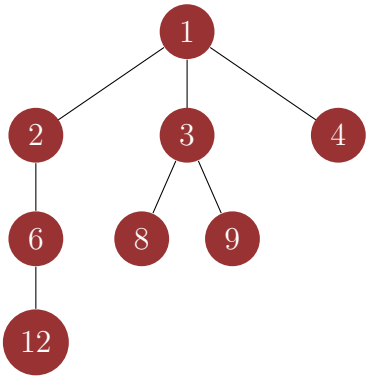
Definition 3.7 (Ähnlichkeitsmaß)**:**
Seien T, T' Bäume und $\mathfrak{T}$ die Menge aller Bäume. Des Weiteren sei $\alpha(T, T') := |\{v \in T \mid \sigma(v) \subset T'\}|$, der sogenannte *Ähnlichkeitsindex*. Die Abbildung

$$\Lambda \; : \; \mathfrak{T} \times \mathfrak{T} \to [0,1] \; , \; (T, T') \mapsto \begin{cases} \frac{\alpha(T,T')}{|V|+|V'|-\alpha(T,T')} & , V \neq \emptyset, \; V' \neq \emptyset \\ 0 & , \text{sonst} \end{cases}$$

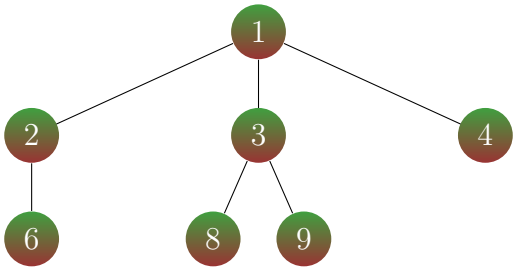
heißt Ähnlichkeitsmaß.



(a) Suchbaum T



(b) Suchbaum T'



(c) Gemeinsamer Suchbaum von T und T'

Abbildung 3.2: Durch das Vereinigen der beiden Suchbäume T und T' entsteht ein gemeinsamer Suchbaum (c). In diesem Beispiel ist die Teilmenge $\tilde{V}$ der Knotenmenge V von T durch die Knoten $1, 2, 3, 4, 6, 8$ und 9 gegeben, da für jeden dieser Knoten der zugehörige Pfad vollständig in T' enthalten ist, so ist z. B. $\sigma(1) = (1)$ und $\sigma(8) = (8, 3, 1)$. Andererseits gilt für Knoten 11, dass $\sigma(11) = (11, 5, 2, 1) \nsubseteq V'$, da $11 \notin V'$. In diesem Beispiel ist $\alpha(T, T') = 7$. Somit ergibt sich für die Ähnlichkeit $\Lambda(T, T') = \frac{7}{13}$. D. h. die Bäume T und T' haben etwas mehr als die Hälfte aller Knoten gemeinsam.

Ein Beispiel für die Ähnlichkeit und die Berechnung des Ähnlichkeitsmaßes zweier Bäume ist Abbildung 3.2.

Korollar 3.8:
Für zwei Bäume T, T' ist der Ähnlichkeitsindex α und das Ähnlichkeitsmaß Λ symmetrisch, d. h. es gilt $\alpha(T, T') = \alpha(T', T)$ und $\Lambda(T, T') = \Lambda(T', T)$.

Satz 3.9:
Das Ähnlichkeitsmaß Λ aus Definition 3.7 liefert für zwei beliebige Bäume T, T' einen Wert im Intervall $[0, 1]$.

Beweis: Seien T, T' zwei Bäume mit den Knotenmengen V und V'. O. B. d. A. können wir $|V| \geq |V'| \geq 1$ annehmen. Ist die Knotenmenge eines Baumes leer, so gilt per Definition $\Lambda(T, T') = 0$. Insbesondere gilt für den Ähnlichkeitsindex $\alpha(T, T') \in [0, |V'|]$.

„≤ 1“

$$\frac{\alpha(T, T')}{|V| + |V'| - \alpha(T, T')} \leq \frac{\alpha(T, T')}{|V'|} \leq 1$$

„≥ 0“

$$\frac{\overbrace{\alpha(T, T')}^{\geq 0}}{\underbrace{|V| + |V'| - \alpha(T, T')}_{>0}} \geq 0$$

□

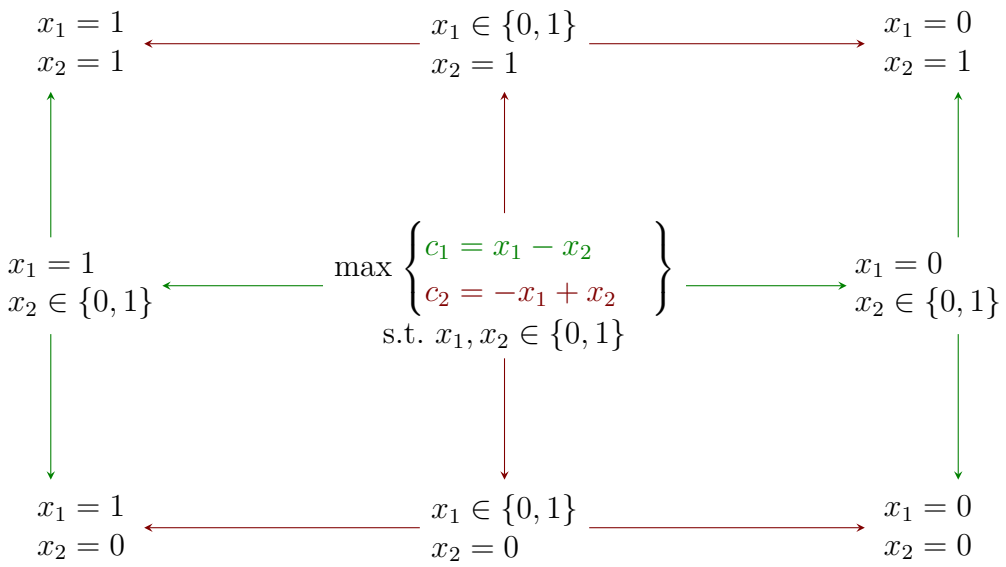
Damit wir aus dem Ähnlichkeitsmaß auch Schlussfolgerungen auf die Gemeinsamkeiten der untersuchten Suchbäume schließen können benötigen wir noch eine Eigenschaft der Verzweigungsstrategie, mit der die Bäume erzeugt wurden.

Definition 3.10 (Konsistenzbedingung)**:**
Sei $\mathcal{P}$ der endliche Lösungsraum eines kombinatorischen Optimierungsproblems und $c_1, \ldots, c_n$ Zielfunktionen. Eine Verzweigungsstrategie $\mathfrak{V}$ heißt *konsistent* bzgl. $\mathcal{P}$ und den Zielfunktionen $c_1, \ldots, c_n$, falls $\mathfrak{V}_{c_i}(\mathcal{P}) = \mathfrak{V}_{c_j}(\mathcal{P})$, für alle $i, j = 1, \ldots, n$ gilt.

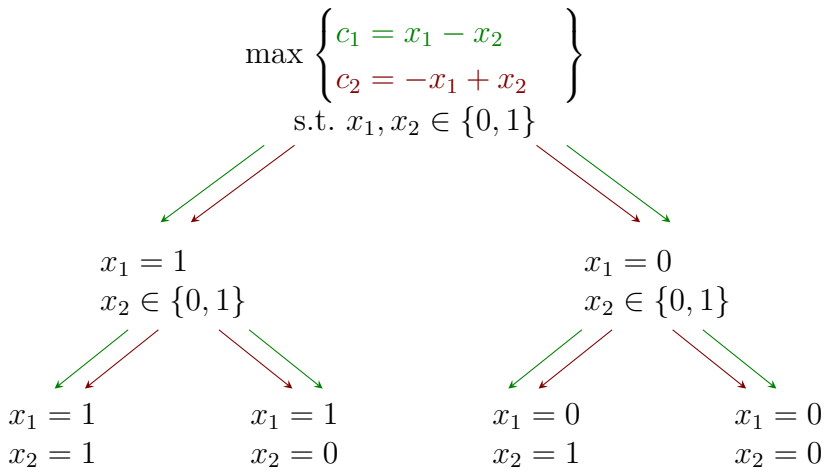
Im Falle einer konsistenten Verzweigungsstrategie schreiben lediglich $\mathfrak{V}$ und verzichten auf die zugehörige Zielfunktion im Index.

Korollar 3.11:
Gegeben sei ein kombinatorischen Optimierungsproblems vom Typ (3.3) mit den Zielfunktionen $c_1, \ldots, c_n$, den Abbruchbedingungen $\mathfrak{A}_1, \ldots, \mathfrak{A}_n$ sowie einer konsistente Verzweigungsstrategie $\mathfrak{V}$. Des Weiteren bezeichne T den zugrunde liegenden implizit gegebenen vollständigen Suchbaum. Für jeden Suchbaum $T_{i,\mathfrak{A}_i}$ gilt stets

1. $V[T_{i,\mathfrak{A}_i}] \subseteq V[T]$
2. $E[T_{i,\mathfrak{A}_i}] \subseteq E[T]$.



(a) Inkonsistente Verzweigungsstrategie



(b) Konsistente Verzweigungsstrategie

Abbildung 3.3: In Bild (a) nutzen wir eine inkonsistente Verzweigungensstrategie. Diese verzweigt nach jener Variable, welche den größten Fortschritt für den Zielfunktionswert bewirkt. Im Fall von Zielfunktion c_1 bewirkt die Fixierung von x_1 auf 1 bzw. 0 den größten Anstieg des Zielfunktionswertes. Betrachten wir Zielfunktion c_2, so bringt die Fixierung von x_2 auf 1 bzw. 0 den größten Anstieg des Zielfunktionswertes. In Bild (b) hingegen verzweigen wir stets nach dem kleinsten noch nicht verzweigten Variablenindex, diese Strategie ist offensichtlich konsistent bzgl. jeder Zielfunktion.

Die Korrektheit dieses Korollars macht man sich leicht klar, da wir uns stets in einem Teilbaum von T befinden, nur zu unterschiedlichen Zeitpunkten, welche durch die entsprechende Abbruchbedingung gegeben sind, abbrechen. Die Abbildung 3.3 verdeutlicht ebenfalls, dass der implizit gegebene vollständige Suchbaum T stets eine Obermenge jedes Suchbaumes $T_{i,\mathfrak{A}_\mathrm{i}}$ bzgl. Problem P^i bildet, falls $\mathfrak{V}$ konsistent ist.

Implementierung der Berechnung des Ähnlichkeitsmaßes Λ

Die Implementierung des Ähnlichkeitsindex erfolgt, im Gegensatz zu der Formulierung in Definition 3.7, rekursiv und ist in Algorithmus Similarity auf Seite 21 dargestellt. Dabei wird stets von den Wurzelknoten v bzw. v' der Suchbäume T bzw. T' ausgegangen. Sind die Wurzelknoten verschieden, so sind sich T und T' nicht ähnlich und es gilt $\Lambda(T, T') = 0$. Andernfalls betrachten wir alle Kinder der Knoten v und v', welche wir im Folgenden mit $child_T(v)$ bzw. $child_{T'}(v')$ bezeichnen wollen. Ist der Schnitt der beiden Knotenmengen $child_T(v)$ und $child_{T'}(v')$ nicht leer, so betrachten wir für jeden Knoten $v_i \in child_T(v) \cap child_{T'}(v')$ die Teilbäume $T(v_i)$ und $T'(v_i)$ von T bzw. T', welche den Knoten v_i als Wurzel haben. Diese Prozedur wiederholt sich bis wir in den Blätter von $T \cap T'$ angekommen sind, diese seien $V_{leaf} = \{v_1^{leaf}, \ldots, v_l^{leaf}\}$. Der Ähnlichkeitsindex ist nun gegeben durch $\alpha(T, T') = |\bigcup_{v_i^{leaf} \in V_{leaf}} \sigma(v_i^{leaf})|$ oder anders formuliert

$$\alpha(T, T') = \begin{cases} 1 + \sum\limits_{\tilde{v} \in child_T(v) \cap child_{T'}(v')} \alpha(T(\tilde{v}), T'(\tilde{v})) & , v = v' \\ 0 & , \text{ sonst.} \end{cases} \tag{3.4}$$

Beispiele für die Ähnlichkeit der Suchbäume innerhalb eines Schnappschussproblems sind in den Tabellen 3.1a und 3.1b auf Seite 21 sowie in Anhang B.1 zu finden. Für die Gebäude A,B und C ist je ein Schnappschussproblem pro Instanz aufgeführt. Es ist dabei sehr gut zu sehen, dass die Ähnlichkeit der Suchbäume, mit Ausnahme kleiner Schwankungen, stets zunimmt. Ebenfalls erkennbar ist ein plötzlicher Sprung in der Ähnlichkeit, sodass die Suchbäume in den letzten Iterationsrunden Ähnlichkeiten über 85 % aufweisen. Diese Ergebnisse rechtfertigen auch die Konstruktion eines warmstartenden Branch & Bound-Verfahrens, wie wir es im folgenden Abschnitt vorstellen.

3.3.2 Modifikation und Sicherung der Korrektheit

Um aus dem bisherigem Branch & Bound-Verfahren ein warmstartendes zu konstruieren und dieses zu beschreiben, benötigen wir noch etwas mehr Notation.

- Unter einer *Iterationsrunde* verstehen wir im Folgenden den Lösungsprozess eines Problems P^i bzgl. der Zielfunktion c_i bis zur Erfüllung der Abbruchbedingung $\mathfrak{A}_\mathrm{i}$.
- Wie zu Beginn dieses Abschnittes erwähnt bezeichnen wir die Menge der Teilprobleme in Iteration i mit K^i.
- Die Menge aller Teilprobleme, welche aufgrund der oberen Schranke in Iterationsrunde i ausgelotet wurden, bezeichnen wir mit $\tilde{K}^i$.

Algorithmus 1: Rekursiver Ablauf für die Berechnung des Ähnlichkeitsmaßes zweier Bäume T und T' mit den Würzelknoten v und v'.

1 **Algorithmus :** Similarity

Input : Knotenmengen $\tilde{V} \subseteq V$ und $\tilde{V}' \subseteq V'$ von T und T'. Beim ersten Aufruf dieser Methode gilt $\tilde{V} = \{v\}$ bzw. $\tilde{V}' = \{v'\}$.

Output : $\Lambda(T, T')$

2 $\alpha \leftarrow 0$

3 **foreach** $\tilde{v} \in \tilde{V}$ **do**

4 **if** $\tilde{v} \in \tilde{V}'$ **then**

5   **if** $|child_T(\tilde{v})| = 0$ *or* $|child_{T'}(\tilde{v})| = 0$ **then**

6   **return** $\alpha \leftarrow \alpha + 1$

7   **else**

8   **return** $\alpha \leftarrow \alpha + \mathsf{Similarity}(child_T(\tilde{v}), child_{T'}(\tilde{v})) + 1$

9 **return** $\frac{\alpha}{|V|+|V'|-\alpha}$

T_i	1	2	3	4	5	6	7
1	1.000	0.2581	0.1048	0.1044	0.1069	0.1030	0.1039
2	0.2581	1.000	0.2591	0.2637	0.2658	0.2604	0.2626
3	0.1048	0.2591	1.000	0.5706	0.5105	0.5537	0.5629
4	0.1044	0.2637	0.5706	1.000	0.8484	0.9098	0.9286
5	0.1069	0.2658	0.5105	0.8484	1.000	0.9187	0.8965
6	0.1030	0.2604	0.5537	0.9098	0.9187	1.000	0.9778
7	0.1039	0.2626	0.5629	0.9286	0.8965	0.9778	1.000

(a) Gebäude A, Tabellarische Darstellung der Ähnlichkeit Λ zwischen Suchbäume T_i und T_j innerhalb eines Schnappschussproblems einer Real Down Peak Instanz.

T_i	1	2	3	4	5	6	7
1	1.000	0.1138	0.1287	0.1455	0.1486	0.1415	0.1431
2	0.1138	1.000	0.2996	0.2869	0.2726	0.2520	0.2560
3	0.1287	0.2996	1.000	0.3394	0.3149	0.2919	0.2964
4	0.1455	0.2869	0.3394	1.000	0.8427	0.8365	0.8542
5	0.1486	0.2726	0.3149	0.8427	1.000	0.8324	0.8157
6	0.1415	0.2520	0.2919	0.8365	0.8324	1.000	0.9823
7	0.1431	0.2560	0.2964	0.8542	0.8157	0.9823	1.000

(b) Gebäude A, Tabellarische Darstellung der Ähnlichkeit Λ zwischen Suchbäume T_i und T_j innerhalb eines Schnappschussproblems einer Up Peak Instanz

Tabelle 3.1: Die Tabellen 3.1a und 3.1b sind Beispiele für die zunehmende Ähnlichkeit der Suchbäume im Laufe aufeinanderfolgender Iterationsrunden. Die Tabellen sind wie folgt zu interpretieren: Der Eintrag (i, j) gibt an, wie viel Prozent der Knoten von Suchbaum T_i und T_j in beiden Bäumen enthalten sind. So sind z. B. 84.84 % der Knoten von T_4 und T_5 in beiden Suchbäumen enthalten sind.

- Mit Σ bezeichnen wir die Menge aller gefunden Lösungen, die zu keiner Verbesserung der oberen Schranke führten.

Klar ist nun, dass zu Beginn der ersten Iterationsrunde $\tilde{K}^1$ und Σ leer sind und $K^1 = \{\mathcal{P}^0\}$ gilt.

Befinden wir uns in einer Iterationsrunde i und wollen zur Iterationsrunde $i+1$ übergehen, so benötigen wir zur Sicherung der Korrektheit zwei Schritte: Die Aktualisierung und die Neubewertung.

1. Unter einer *Aktualisierung* verstehen wir die Neuberechnung aller Kosten, der Knoten in K^i und $\tilde{K}^i$, sowie die der Lösungen in Σ bzgl. der Zielfunktion c_{i+1}. Damit können sich die unteren Schranken der Knoten in K^i und $\tilde{K}^i$ ändern. Um in der folgenden Iterationsrunde mit einer möglichst guten oberen Schranke zu starten setzten wir
$$U = \min_{x \in \Sigma} c_{i+1}(x)$$

2. Unter einer *Neubewertung* verstehen wir eine Prüfung, ob jeder Knoten in K^i und $\tilde{K}^i$ sich in der richtigen Menge befindet. Denn durch den Wechsel von c_i auf c_{i+1} können sich sowohl die unteren Schranken in den Knoten als auch die obere Schranke U geändert haben. Demzufolge kann es Knoten $v \in K^i$ geben, sodass für die korrespondierenden Lösungsräume $\mathcal{P}^v$ nach der Aktualisierung $lb_{c_{i+1}}(\mathcal{P}^v) \geq U$ gilt. Diese Knoten müssen ausgelotet werden. Analog kann es Knoten $v \in \tilde{K}^i$ geben, sodass $lb_{c_{i+1}}(\mathcal{P}^v) < U$ gilt. Die Aufnahme solcher Knoten in K^i bezeichnen wir als *Wiederbelebung*.

Die aktualisierte und neubewertete Menge der noch zu untersuchenden und die der ausgeloteten Teilprobleme bezeichnen wir nun mit K^{i+1} bzw. $\tilde{K}^{i+1}$. Während jeder Iterationsrunde bzw. nach der Neubewertung gilt stets $K^i \cap \tilde{K}^i = \emptyset$ bzw. $K^{i+1} \cap \tilde{K}^{i+1} = \emptyset$.

Bemerkung 3.12:
Für jedes Blatt v im aktuellen Suchbaum gilt zu jeder Zeit des Algorithmus
$$v \in K^i \quad \dot{\vee} \quad v \in \tilde{K}^i$$

Wir wollen nun das warmstartende Branch & Bound-Verfahren schematisch darstellen. Wie zu Beginn können wir davon ausgehen, dass ein Algorithmus existiert, welcher die Probleme $P_1, \ldots, P_n$ durch den iterativen Aufruf eines Branch & Bound-Verfahrens löst. Zusätzlich möge dieser Algorithmus die Aktualisierung und Neubewertung korrekt durchführen. Es genügt daher das Verfahren für eine Iterationsrunde i vorzustellen. Die Eingabe für das warmstartende Branch & Bound-Verfahren zu Beginn der i-ten Iterationsrunde ist die Folgende

- Zielfunktion c_i
- Menge der bisher gefundenen Lösungen Σ, wobei diese im Fall $i = 1$ leer ist.
- $i = 1$: Menge der Teilprobleme $K^1 = \{\mathcal{P}^0\}$ und $\tilde{K}^1 = \emptyset$.

  $i \geq 2$: Die aktualisierten und neubewerteten Mengen der Teilprobleme K^i und $\tilde{K}^i$.

- $i = 1$: Heuristisch berechnete Lösung x^* mit $U = c_i(x^*)$ oder $U = \infty$.

  $i \geq 2$: Obere Schranke $U = \min_{x \in \Sigma} c_i(x)$ und dazugehörige Lösung $x^* = \arg\min_{x \in \Sigma} c_i(x)$.

- Abbruchbedingung $\mathfrak{A}_i$

Zusätzlich zur schematischen Darstellungen sind alle für den Warmstart notwendigen Ergänzungen kursiv hervorgehoben.

1. $K^i = \emptyset$ *oder* $\mathfrak{A}_i$ *ist erfüllt* $\rightarrow$ Abbruch und gib, falls vorhanden, beste Lösung x^* zurück, ansonsten wähle $\mathcal{P} \in K^i$.

2. $lb_{c_i}(\mathcal{P}) \geq U \rightarrow$ keine Lösung in $\mathcal{P}$ ist besser als die bisher beste Lösung. Entferne $\mathcal{P}$ aus K^i *und füge* $\mathcal{P}$ *zu* $\tilde{K}^i$ *hinzu.* Gehe zu Schritt 1.

3. $lb_{c_i}(\mathcal{P}) < U$:

   (a) $heur(\mathcal{P}) = \perp$: Partitioniere $\mathcal{P}$ in (geeignete) Teilprobleme $\mathcal{P}^1, \ldots, \mathcal{P}^k$ und füge diese zu K^i hinzu. Entferne $\mathcal{P}$ aus der Menge der Teilprobleme. Gehe zu Schritt 1.

   (b) $heur(\mathcal{P}) = x^*_{\mathcal{P}}$: Wir haben eine zulässige Lösung gefunden. Aktualisiere $U = c(x^*_{\mathcal{P}})$ und $x^* = x^*_{\mathcal{P}}$. Es muss sichergestellt werden, dass keine Lösung $x' \in \mathcal{P}$ existiert, sodass $c_i(x^*_{\mathcal{P}}) > c_i(x')$ gilt.

      i. $c_i(x^*_{\mathcal{P}}) = lb_{c_i}(\mathcal{P}) \rightarrow x^*_{\mathcal{P}}$ ist die beste in $\mathcal{P}$ enthaltene Lösung. Entferne $\mathcal{P}$ aus K *und füge* $\mathcal{P}$ *zu* $\tilde{K}^i$ *hinzu, sofern* $\mathcal{P}$ *noch verzweigt werden kann.* Gehe zu Schritt 1.

      ii. $c(x^*_{\mathcal{P}}) > lb_c(\mathcal{P}) \rightarrow$ Wir können aus dieser Lösung keine Schlussfolgerungen ziehen. Partitioniere $\mathcal{P}$ in (geeignete) Teilprobleme $\mathcal{P}^1, \ldots, \mathcal{P}^k$ und füge diese zu K^i hinzu. Entferne $\mathcal{P}$ aus der Menge der Teilprobleme. Gehe zu Schritt 3.

Bemerkung 3.13 (Ergänzung zu Schritt 3(b)i):
In einem regulären Branch & Bound-Verfahren würden wir an dieser Stellen das Teilproblem $\mathcal{P}$ *verwerfen, denn wir haben die beste Lösung* $x^*_{\mathcal{P}}$ *innerhalb* $\mathcal{P}$ *bereits gefunden. Wechseln wir jedoch die Zielfunktion, so kann es möglich sein, dass weitere Partitionierungen des Teilproblems zu einer besseren Lösung bzgl. der neuen Zielfunktion führen.*

Das so modifizierte Verfahren wird als *warmstartend* bezeichnet, da nach jedem Wechsel der Zielfunktion von c_i auf c_{i+1} die Menge der Teilmenge K^{i+1} nicht aus dem gesamten Lösungsraum $\mathcal{P}^0$ besteht, sondern aus allen Teilproblemen $\mathcal{P}^v$, welche zu einem Blatt v des Suchbaumes am Ende der i-ten Iterationsrunde korrespondieren und für die $lb_{c_{i+1}}(\mathcal{P}^v) < U$ gilt.

Satz 3.14 (Korrektheit des warmstartenden Branch & Bound-Algorithmus)**:**
Der erweiterte Branch & Bound-Algorithmus arbeitet korrekt.

Beweis: Um die Korrektheit des warmstartenden Branch & Bound-Algorithmus zu beweisen, können wir o. B. d. A. annehmen, dass die Abbruchbedingung dem Beweis der Optimalität entspricht.

Betrachten wir den Beginn der i-ten Iterationensrunde. Aufgrund der Konstruktion der Mengen Σ, K^i und $\tilde{K}^i$ können wir feststellen, dass sich eine Optimallösung des Problems P^i in einer der drei Mengen befindet.
Setzen wir eine korrekte Aktualisierung und Neubewertung der Mengen K^{i-1} und $\tilde{K}^{i-1}$ voraus, so gilt $lb_{c_i}(\mathcal{P}) < U = \min_{x \in \Sigma} c_i(x)$ für alle $\mathcal{P} \in K^i$ bzw. $lb_{c_i}(\mathcal{P}') \geq U$ für alle $\mathcal{P}' \in \tilde{K}^i$. Gelangen wir im Laufe des oben präsentierten Verfahrens stets in die Schritte 2 oder 3a, so ist $x^* = \arg\min_{x \in \Sigma} c_i(x)$ die gesuchte Optimallösung. Andernfalls finden wir bei jedem Ausführen von Schritt 3b eine zulässige Lösung, welche die obere Schranke verbessert und die neue beste Lösung darstellt. $\square$

Kapitel 4

Der Exact-Replan-Algorithmus

In der Einleitung wurde bereits erwähnt, dass wir den Exact-Replan-Algorithmus aus [Hil10] als Grundlage für eine effiziente Reoptimierung nutzen. In diesem Kapitel soll es um die wesentlichen Bestandteile dieses Algorithmus gehen. Zu Beginn stellen wir das zugrundeliegende mathematische Modell vor. Im Anschluss daran betrachten wir in den Algorithmus und diskutieren, wie mithilfe einer Heuristik eine Startlösung und mithilfe einer weiteren Heuristik eine untere Schranke für die minimalen reduzierten Kosten eines Fahrplans berechnet wird. Als Letztes wollen wir uns mit der Spaltengenerierung innerhalb des Reoptimierungsprozesses beschäftigen. Dabei gehen wir insbesondere auf die Konstruktion neuer Spalten ein.

4.1 Das Set-Partitioning-Model

Wie wir in den vorherigen Kapiteln bereits festgestellt haben, besteht ein Fahrplan aus mehreren Touren, eine für jeden Aufzug. $\mathcal{T}(e)$ sei die Menge aller zulässigen Touren für Aufzug e und wir definieren $\mathcal{T} := \bigcup_{e \in \mathcal{E}} \mathcal{T}(e)$. In einem zulässigen Fahrplan $S \in \mathcal{S}$ wird Auftrag Ruf von genau einer Tour bedient. Um diese Eigenschaft zu modellieren wird in [Hil10] eine Entscheidungsvariable $x_t \in \{0, 1\}$ eingeführt. Des Weiteren bezeichnet $c(t)$ die Kosten der Tour t von Aufzug e. Das dem Exact-Replan-Algorithmus zugrunde liegende Set-Partitioning-Modell lautet:

$$\min \sum_{t \in \mathcal{T}} c(t) x_t \tag{4.1}$$

$$\sum_{t \in \mathcal{T} \colon \varrho \in t} x_t = 1 \quad \forall \varrho \in \mathcal{R}_u \tag{4.2}$$

$$\sum_{t \in \mathcal{T}(e)} x_t = 1 \quad \forall e \in \mathcal{E} \tag{4.3}$$

$$x_t \in \{0, 1\} \quad \forall t \in \mathcal{T} \tag{4.4}$$

Die Zuweisung eines jeden Auftrags zu genau einer Tour bzw. genau einer Tour zu für jeden Aufzug wird durch die Nebenbedingungen (4.2) bzw. (4.3) sichergestellt. In Kapitel 2 haben wir einige Restriktionen für eine zulässige Tour und damit auch für die Menge der zulässigen Touren erörtert. Hinzu kommt noch, dass die Größe der Menge $\mathcal{T}$ exponentiell zur Anzahl aller noch nicht geladenen Aufträge ist, d. h. $|\mathcal{T}(e)| \in \Omega(2^{|\mathcal{R}_u|}) \, \forall e \in \mathcal{E}$. Diese beiden Punkte führen dazu, dass für großes $\mathcal{R}_u$ das

obige Modell nicht ohne Weiteres mit einem „standard“ IP-Löser gelöst werden kann. Abhilfe schafft die in Abschnitt 4.3 beschriebene Spaltengenerierung in Verbindung mit einem auf das Problem zugeschnitten Branch & Bound-Verfahren. Bevor wir mit der Spaltengenerierung beginnen können, benötigen wir jedoch noch eine Startlösung, welche uns zum einen eine obere Schranke.

4.2 Berechnung einer Startlösung

In einem Schnappschussproblem eines VZ-Systems betrachten wir stets die Menge $\mathcal{R}$ aller Aufträge, welche sich entweder bereits in einem Aufzug befinden oder durch den Algorithmus auf die jeweiligen Aufzüge verteilt werden müssen, wobei wir letztere als die Menge $\mathcal{R}_u$ zusammenfassen. Um eine zulässige Startlösung berechnen zu können benötigen wir den Fahrplan S aus dem vorherigen Schnappschussproblem. Dieser Fahrplan bedient alle bereits geladenen Aufträge $\mathcal{R} \setminus \mathcal{R}_u$, welche sich noch im System befinden.
Um eine möglichst „gute“ Startlösung zu berechnen werden die noch nicht zugewiesenen Aufträge anhand ihrer aktuellen Wartezeit τ_{wait} nicht aufsteigend sortiert, sodass für alle Aufträge ϱ_i und ϱ_j, aus $\mathcal{R}_u = \{\varrho_1, \ldots, \varrho_{|\mathcal{R}_u|}\}$, stets $\tau_{\text{wait}}(\varrho_i) \geq \tau_{\text{wait}}(\varrho_j)$ gilt, falls $i < j$ ist. Die Aufträge werden nun, beginnend mit dem am längsten wartenden, „gierig“ auf die Aufzüge verteilt. Das bedeutet: In der ersten Iteration betrachten wir alle Aufzüge $e \in \mathcal{E}$ und bestimmen den Aufzug e_i, welcher durch die Aufnahme von ϱ_1 den geringsten Kostenanstieg im Vergleich zu seinem alten Fahrplan $S(e_i)$ in S hat. Diesem Aufzug wird nun der Auftrag ϱ_1 zugewiesen. Anschließend aktualisieren wir den Fahrplan S und führen diese Prozedur (Algorithmus BestInsert) für $\varrho_2, \ldots, \varrho_{|\mathcal{R}_u|}$ durch. Der so erzeugte Fahrplan ist zulässig und der dazugehörige Zielfunktionswert liefert eine obere Schranke für die Optimallösung der Instanz $(S, \mathcal{E}, \mathcal{R}, \mathcal{F}(\mathcal{E}), \mathcal{D}(\mathcal{E}))$.

Algorithmus 2: Algorithmus zur Berechnung einer zulässigen Startlösung.

1 **Algorithmus :** BestInsert
Input : $\mathcal{E}$, $\mathcal{R}$, initialisierter Fahrplan S für die Aufträge $\mathcal{R} \setminus \mathcal{R}_u$
Output : Fahrplan S für die Aufträge $\mathcal{R}$

2 Sortiere $\mathcal{R}_u$ nach nicht aufsteigenden Wartezeiten
// $\forall i \leq j \,:\, \tau_{\text{wait}}(\varrho_i) \geq \tau_{\text{wait}}(\varrho_j)$
3 $i \leftarrow 1$
4 **for** $\varrho_i \in \mathcal{R}_u$ **do**
5 $\quad \underline{c} \leftarrow \infty$ // Minimale Kosten für das Aufnehmen von ϱ_i bzgl. S
6 $\quad e_{best} \leftarrow \emptyset$ // Kostengünstigster Aufzug, welcher ϱ_i bedient
7 $\quad$ **foreach** $e \in \mathcal{E}$ **do**
8 $\quad\quad S'(e) \leftarrow S(e) \cup \{\varrho_i\}$ // Fahrplan für Aufzug e mit ϱ_i
9 $\quad\quad$ **if** $c(S'(e)) - c(S(e)) < \underline{c}$ **then**
10 $\quad\quad\quad \underline{c} \leftarrow c(S'(e)) - c(S(e))$
11 $\quad\quad\quad e_{best} \leftarrow e$
12 $\quad S(e_{best}) \leftarrow S(e_{best}) \cup \{\varrho_i\}$ // Weise ϱ_i Aufzug e_{best} zu
13 $\quad i \leftarrow i + 1$
14 **return** S

Wir haben nun eine sehr kleine Spaltenindexmenge $\tilde{\mathcal{T}}$ konstruiert, reduzierten das Problem (4.1) – (4.4) auf $\tilde{\mathcal{T}}$ und definieren $\mathcal{T}' := \mathcal{T} \setminus \tilde{\mathcal{T}}$. Der Zielfunktionswert des durch den BestInsert bestimmten Fahrplans stellt eine obere Schranke c^0 dar. Wir können jetzt mit der Spaltengenerierung und dem Lösungsprozess bis zum Beweis der Optimalität beginnen. Für die Bestimmung der Qualität der momentan besten Lösung, zu diesem Zeitpunkt c^0, führen wir noch eine untere Schranke $\underline{\tilde{c}}$ für die minimalen reduzierten Kosten des Fahrplans ein, mit welcher wir den kommenden Abschnitt auch beginnen wollen.

4.3 Spaltengenerierung via Branch & Bound

In diesem Abschnitt betrachten wir stets einen Aufzug e. Das bedeutet, wir führen die Spaltengenerierung und damit auch die Reoptimierung für jeden Aufzug separat durch und fügen erst dann alle gefunden Spalten dem IP hinzu. Wir behalten die Notation aus Kapitel 3 bei, d. h. unter einer *Iterationsrunde* verstehen wir wieder den Lösungsprozess bis zum Abbruch aufgrund einer Abbruchbedingung $\mathfrak{A}$.

4.3.1 Die Lasdon Schranke

Der vorherige Abschnitt beschreibt wie eine zulässige Lösung und damit auch eine obere Schranke für ein Schnappschussproblem berechnet wird. Um die Qualität der Lösung in jeder Iterationsrunde bestimmen zu können wird noch eine untere Schranke benötigt. Für eine solche untere Schranken gibt es neben der in [Hil10] diskutierten Schranke von Tanaka et al. [TUA05] auch eine Schranke, welche auf dem Satz 4.1 von Lasdon beruht. An dieser Stelle verzichten wir auf eine nähere Erläuterung der Schranke von Tanaka et al. und wollen nur auf die Schranke von Lasdon eingehen, da wir in unseren Berechnungen lediglich mit dieser arbeiten.

Satz 4.1 ([Las70])**:**
Sei wieder $\tilde{\mathcal{T}}$ die Menge aller bereits konstruierten Touren, welche das aktuelle Schnappschussproblem definieren, $(x_t^*)_{t\in\mathcal{T}}$ sei eine Optimallösung des momentanen Schnappschussproblems und $(\pi_\varrho^*)_{\varrho\in\mathcal{R}}$ sowie $(\pi_e^*)_{e\in\mathcal{E}}$ die dazugehörigen Dualpreisvektoren. Des Weiteren sei $\underline{\tilde{c}}(e)$ eine untere Schranke der minimalen reduzierten Kosten aller zulässigen Fahrpläne für Aufzug e:

$$\underline{\tilde{c}}(e) \leq \min_{t\in\tilde{\mathcal{T}}(e)} \left\{ c(t) - \pi_e^* - \sum_{\varrho\in\mathcal{R}\cap t} \pi_\varrho^* \right\}. \tag{4.5}$$

Die Kosten c^* jeder Optimallösung der LP-Relaxierung von (4.1)-(4.5) sind durch

$$c^* \geq \sum_{t\in\tilde{\mathcal{T}}} c(t)x_t^* + \sum_{e\in\mathcal{E}} \underline{\tilde{c}}(e) \tag{4.6}$$

beschränkt.

Erinnern wir uns an die Spaltengenerierung aus Kapitel 1 zurück. Die Aufgabe ist in jeder Iterationsrunde das Teilproblem (1.6)

$$\tilde{c} := \min\{\ c(t) - \pi^T t \,|\, t \in \mathcal{T}'\ \}$$

zu lösen. Im gleichen Abschnitt haben wir bereits angesprochen, dass sich dieses Teilproblem auch mit einem Branch & Bound-Verfahren lösen lässt. Mit diesem Lösungsprozess wollen wir uns auf den nächsten Seiten beschäftigen, dabei gehen wir auf die Konstruktion des Suchbaumes, insbesondere auf die Verzweigungsstrategie, ein und werden uns dann mit der Klassifizierung der gefunden Spalten bzw. Touren beschäftigen.

4.3.2 Konstruktion des Suchbaumes

Zu Beginn jeder Iterationsrunde erhalten wir einen neuen Dualpreisvektor π. Mit Hilfe des folgenden Satzes können wir in jedem Knoten v eine Teilmenge der optionalen Aufträge bestimmen, sodass alle Touren die mindestens einen dieser Aufträge bedienen nicht zur Lösung des Teilproblems (1.6) benötigt werden.

Satz 4.2:
Betrachte einen Aufzug e und seine optionalen Aufträge $O_v \subseteq \mathcal{R}_u$ sowie den aktuellen Dualpreisvektor π und die untere Schranke $\underline{c}$ der Primalkosten. Für jeden Auftrag ϱ stellt $\tilde{\underline{c}}(\varrho) = \underline{c}(\varrho) - \pi_\varrho$ eine untere Schranke der reduzierten Kosten dar. Existiert eine Teilmenge O'_v der optionalen Aufträge, sodass für jeden Auftrag innerhalb O'_v die untere Schranke der reduzierten Kosten nicht negativ ist, so gibt es für das Teilproblem (1.6) eine optimale Tour, welche keinen der Aufträge in O'_v bedient.

Beweis: Mit $A_v \subseteq \mathcal{R}_u$ bezeichnen wir alle Aufträge, welche dem Aufzug bereits zugewiesen wurden und mit $O_v = \mathcal{R}_u \setminus A_v$ alle optionalen Aufträge. Betrachten wir die Berechnung der unteren Schranke der reduzierten Kosten einer Tour t in einem Knoten v:

$$\begin{aligned}
\tilde{\underline{c}}(v) &= c(t_v) && \text{(bisherige Kosten der Tour)}\\
&+ \sum_{\varrho \in A_v} \underline{c}(\varrho) && \text{(zugewiesene Aufträge)}\\
&+ \sum_{\varrho \in O_v} (\underline{c}(\varrho) - \pi_\varrho) && \text{(optionale Aufträge)}\\
&- \pi_e . && \text{(Dualpreise Aufzug } e\text{)}
\end{aligned}$$

Die Summe über die optionalen Aufträge können wir wie folgt aufspalten und nach unten abschätzen:

$$\begin{aligned}
\sum_{\varrho \in O_v} (\underline{c}(\varrho) - \pi_\varrho) &= \sum_{\varrho \in O_v \,:\, \underline{c}(\varrho) - \pi_\varrho < 0} (\underline{c}(\varrho) - \pi_\varrho) + \overbrace{\sum_{\varrho \in O_v \,:\, \underline{c}(\varrho) - \pi_\varrho \geq 0} (\underline{c}(\varrho) - \pi_\varrho)}^{\text{Summe über alle Aufträge in } O'_v}\\
&\geq \sum_{\varrho \in O_v \,:\, \underline{c}(\varrho) - \pi_\varrho < 0} (\underline{c}(\varrho) - \pi_\varrho) .
\end{aligned}$$

□

Die Verzweigungsstrategie

Neben dem Kriterium für die untere Schranke der reduzierten Kosten aus Satz 4.2 muss jeder Auftrag in jedem Knoten, welcher in dieser Iterationsrunde von Aufzug e aufgenommen werden soll, eine der beiden folgenden Bedingungen erfüllen:

1. Der Aufzug e steht momentan auf Etage f:
   (a) Die Startetage von Auftrag ϱ ist f oder
   (b) die Startetage von ϱ liegt oberhalb bzw. unterhalb von f, falls die aktuelle Fahrtrichtung von e aufwärts bzw. abwärts ist.
2. Der Aufzug e befindet sich in Fahrt zwischen zwei Etagen:
   (a) Die Fahrtrichtung von e und ϱ ist identisch und
   (b) die Startetage von ϱ befindet sich in Fahrtrichtung hinter der nächsten erreichbaren Etage oder ist mit mit dieser identisch.

Nachdem wir die Menge der zulässigen Aufträge für einen Aufzug e ermittelt haben, ergeben sich die Verzweigungsmöglichkeiten aus allen verbindlichen Entladestopps, d. h. die Zieletage der geladenen Fahrgäste, und den Startetagen der optionalen Aufträgen. Diese Verzweigungsmöglichkeiten eines Knotens v bezeichnen wir im Folgenden mit $\mathcal{V}_{opt}(v)$. Die Verzweigungsstrategie im Exact-Replan ist demnach die Entscheidung, welche Etage als nächsten angefahren wird. Aufgrund der Definition einer zulässigen Tour bzw. eines zulässigen Stopps gibt es in jedem Knoten, welcher über verbindliche Entladestopps verfügt, immer die Möglichkeit direkt bis zum nächst gelegenen verbindlichen Entladestopp zu fahren und keine weiteren Fahrgäste aufzunehmen. Gibt es in einem Knoten keine verbindlichen Entladestopps mehr, so haben wir eine zulässige Tour konstruiert und können diese dem Knoten entnehmen. Ein Beispiel für einen vollständigen Suchbaum ist Abbildung 4.1 (a). Für eine vereinfachte Darstellung nehmen wir in diesem Beispiel an, dass jeder Auftrag genau einem Ruf entspricht. Betrachten wir einen Aufzug e, welcher momentan auf Etage 5 hält und den Auftrag ϱ_6 geladen hat, dessen Zieletage ist 7. Des Weiteren erfüllen die Aufträge ϱ_2 und ϱ_4 die Kondition 1b, mit den Startetagen 6 bzw. 8 und der identischen Zieletage 9. Ausgehend vom momentanen Stopp auf Etage 5 gibt es nun zwei Verzweigungsmöglichkeiten. Entweder fährt der Aufzug e direkt zu Etage 7 und lädt ϱ_6 aus oder zu Etage 6 und nimmt Auftrag ϱ_2 auf. Die Etage 8 stellt keine zulässige Verzweigungsmöglichkeit dar, da wir sonst an der Zieletage von ϱ_6 vorbei fahren würden. Im ersten Fall existieren keine weiteren Stoppverpflichtungen, wir erinnern uns an Definition 2.2 aus Kapitel 2.2, somit haben wir in diesem Knoten eine zulässige Tour konstruiert. Ausgehend von diesem Knoten gibt es wieder zwei Verzweigungsmöglichkeiten: entweder zurück zu Etage 6 um Auftrag ϱ_2 einzuladen oder weiter zu Etage 8 um ϱ_4 aufzunehmen. In diesen beiden Knoten gibt es keine zulässige Tour, da durch die Aufnahme eines Auftrags ein Entladestopp entsteht. Im zweiten Fall nehmen wir den Auftrag ϱ_2 auf Etage 6 auf. Es existieren somit noch zwei offene Entladestopps und damit auch keine zulässige Tour in diesem Knoten. In diesem Fall gibt es nur eine Verzweigungsmöglichkeit: Das Ausladen von ϱ_6 auf Etage 7. Auch hier gibt es aufgrund des geladenen Auftrags ϱ_2 noch keine zulässige Tour. Nach dem gleichen Prinzip verzweigen wir alle weiteren so erzeugten Knoten, bis kein Blatt des Suchbaumes über mögliche Verzweigungsmöglichkeiten verfügt. Jedes dieser Blätter enthält somit auch eine zulässige Tour.

In der Abbildung 4.1 (a) ist zu erkennen, dass nicht jeder erzeugter Knote eine zulässige Tour beinhaltet. Die zulässigen Touren sind in den Abbildungen 4.1 (b) – (j) dargestellt. Haben wir eine zulässige Tour gefunden, so muss entschieden werden ob diese das Teilproblem (1.6) löst.

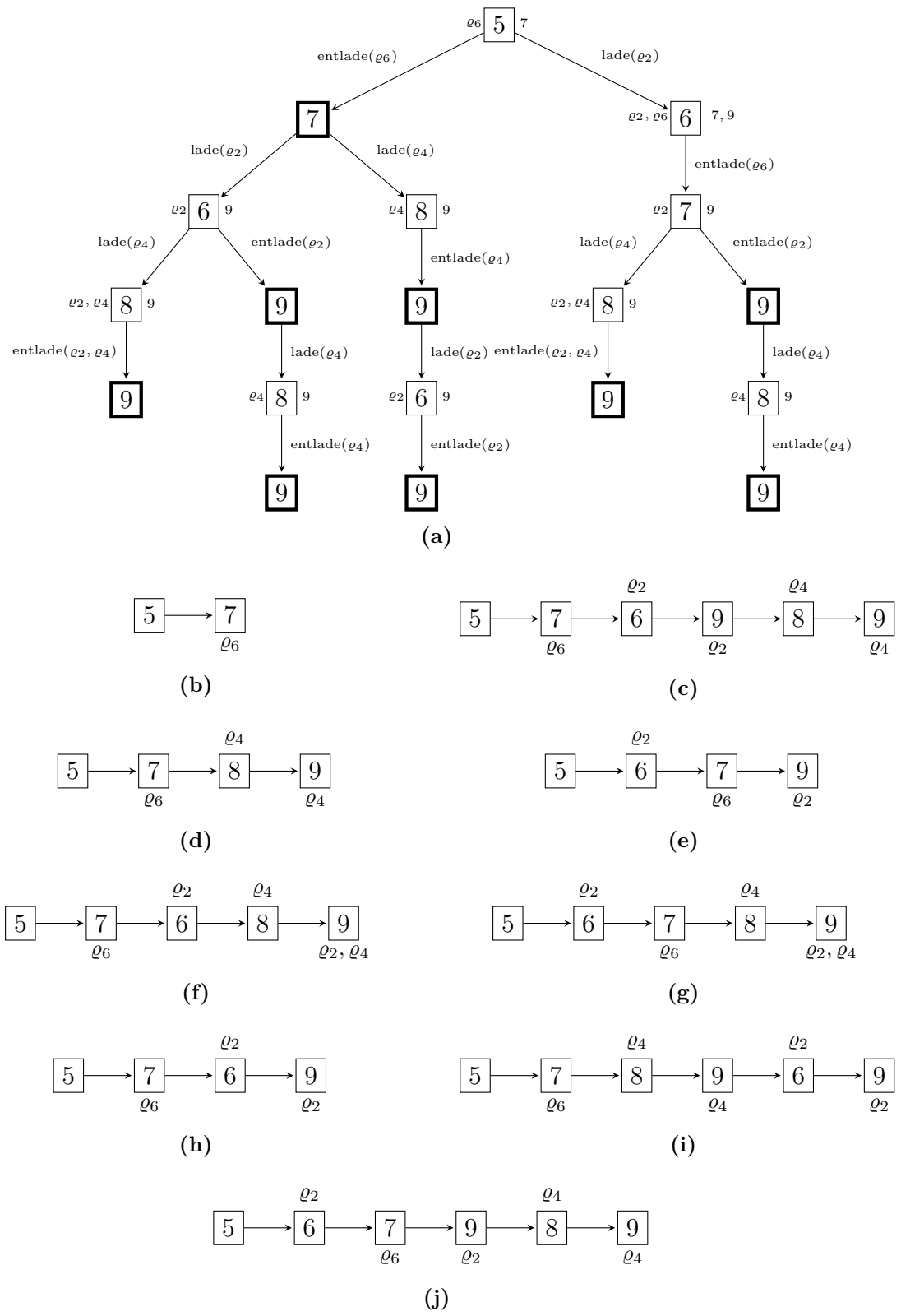


Abbildung 4.1: Abbildung (a) ist die Darstellung eines vollständigen Suchbaumes, bei dem der dazugehörige Aufzug momentan auf Etage 5 steht und bereits Auftrag ϱ_6, mit der Zieletage 7, geladen hat. Noch zu bedienen sind die Auftrage $\varrho_2 : 6 \to 9$ und $\varrho_4 : 8 \to 9$. Jeder Knoten der eine zulässige Tour enthält ist hervorgehoben. Die verpflichtenden Entladestopps sind rechts und die geladenen Aufträge links neben jedem Knoten vermerkt. In den Abbildungen (b) – (j) sind alle zulässigen Touren dargestellt. Oberhalb der Knoten sind die zusteigenden und unterhalb die aussteigenden Aufträge vermerkt.

4.3.3 Klassifikation der Touren und Knoten

Im Gegensatz zu einem standard Branch & Bound-Verfahren sind wir im Falle des Exact-Replan nicht an der besten Lösung interessiert, sondern an einem Pool von Lösungen. Diesen Lösungspool wollen wir mit χ bezeichnen. In Kapitel 3.3 haben wir den Begriff der Abbruchbedingung eingeführt. Eine solche Abbruchbedingung finden wir im Branch & Bound-Verfahren des Exact-Replan wieder. Die Abbruchbedingung $\mathfrak{A}$ lautet in jeder Iterationsrunde: *Beende die Suche nach zulässigen Touren, falls* $|\chi| = \mu \in \mathbb{N}$. Die Wahl des Parameters μ kann beliebig erfolgen. Es besteht auch die Möglichkeit μ dynamisch zu wählen, z. B. anhand der Anzahl gefundener Touren in der letzten Iterationsrunde. Rufen wir uns die Spaltengenerierung sowie das Teilproblem (1.6) wieder in Erinnerung. Dieses Problem lässt sich, aufgrund der Nutzung einer unteren Schranke der Primalkosten wie folgt umformulieren

$$\underline{\tilde{c}} := \min\{\ \underline{c}(t) - \pi^T t \mid t \in \mathcal{T}'\ \}.$$

Wir erinnern uns an $\mathcal{T}' := \mathcal{T} \setminus \tilde{\mathcal{T}}$, wobei $\mathcal{T}$ die implizit gegebene Menge aller Touren und $\tilde{\mathcal{T}}$ die Menge aller bisher konstruierten Touren bezeichnet. Neben der Entscheidung, ob eine (zulässige) Tour in den Lösungspool aufgenommen werden soll oder nicht, bedarf es noch der Entscheidung, ob ein Knoten ausgelotet oder weiter verzweigt werden soll, wodurch neue (zulässige) Touren entstehen. Um diese Entscheidung fällen zu können, benötigen wir eine obere Schranke θ, welche zu Beginn jeder Iterationsrunde mit dem Wert 0 initialisiert wird. Wir können für die obere Schranke den Wert 0 wählen, da nur Touren mit negativen reduzierten Kosten zu einer Verbesserung des Zielfunktionswertes führen. Betrachten wir einen Knoten v, welcher eine zulässige Tour enthält, und seine untere Schranke der reduzierten Kosten $\underline{\tilde{c}}(v)$. Gilt $\underline{\tilde{c}}(v) \geq \theta$, so können wir diesen Knoten ausloten, denn ein weiteres Verzweigen bedeutet auch die Aufnahme eines weiteren Fahrgastes, wodurch die untere Schranke der reduzierten Kosten nicht verkleinert werden kann. Enthält der Knoten keine zulässige Tour, so wird dieser weiter verzweigt. Ist $\underline{\tilde{c}}(v)$ kleiner als θ, betrachten wir die in v enthaltene zulässige Tour und die untere Schranke der reduzierten Kosten $\tilde{c}(t)$. Hier gilt wieder das gleiche Kriterium:

1. $\tilde{c}(t) \geq \theta \Longrightarrow$ die Tour ist nicht „günstig“ und wird verworfen.
2. $\tilde{c}(t) < \theta \Longrightarrow$ die Tour muss für eine genaue Klassifikation weiter untersucht werden.

Der interessante Fall ist 2. Aufgrund der Abbruchbedingung $\mathfrak{A}$ gilt $|\chi| < \mu$ und die untere Schranke der reduzierten Kosten von t ist klein genug. Fest steht nun auch, dass die Tour dem Lösungspool hinzufügt wird. Geprüft werden muss noch, ob es bereits eine weitere Tour in χ gibt, welche die gleiche Menge optionaler Aufträge bedient. Ist dies der Fall, so kann diese aus χ entfernt werden. Im Anschluss wird die obere Schranke θ aktualisiert, in dem diese auf die untere Schranke der reduzierten Kosten $\tilde{c}(t)$ der neuen und damit günstigsten Tour gesetzt wird. Der Knoten v wird anschließend weiter verzweigt. Eine Darstellung dieser Klassifikation der Knoten und Touren ist im Algorithmus CheckNode auf Seite 33 dargestellt.

Zusammenfassung: Der Exact-Replan-Algorithmus

Nachdem wir in den Abschnitten 4.1 – 4.3 die Komponenten des Exact-Replan-Algorithmus und ihre Funktionsweise erläutert haben wollen wir eine schematische Übersicht des gesamten Ablaufes geben.

1. Setze $i \leftarrow 1$ und berechne eine Startlösung, wie in 4.2 beschrieben. Die Menge der konstruierten Touren bezeichnen wir mit $\tilde{\mathcal{T}}$

2. Reduziere den Lösungsraum des Problems (4.1) - (4.4) auf $\tilde{\mathcal{T}}$ und löse die LP Relaxierung des reduzierten Problems. Die Lösung sei mit x^* und der Lösungswert mit c^i bezeichnet.

3. Berechne die Lasdon-Schranke $\underline{\tilde{c}}(\mathcal{E}) = \sum_{e \in \mathcal{E}} \underline{\tilde{c}}(e)$ sowie $c(\tilde{\mathcal{T}}) = \sum_{t \in \tilde{\mathcal{T}}} c(t) x_t^*$. Beende den Algorithmus, falls $c^i \leq c(\mathcal{T}) + \underline{\tilde{c}}(\mathcal{E})$ gilt, ansonsten setzte $i \leftarrow i + 1$ und gehe zu 4.

4. Für jeden Aufzug $e \in \mathcal{E}$:

   (a) Setze $\theta = 0$

   (b) Konstruiere einen Pool χ zulässiger Touren wie in 4.3 beschrieben und beende die Konstruktion des Suchbaumes, falls dieser komplett durchsucht ist oder die Abbruchbedingung $|\chi| = \mu$ erfüllt ist.

   (c) $\tilde{\mathcal{T}} \leftarrow \tilde{\mathcal{T}} \cup \chi$

5. Löse die LP Relaxierung des auf $\tilde{\mathcal{T}}$ reduzierten Problems. Der dazugehörige Zielfunktionswert c^i stellt zugleich eine neue obere Schranke dar.

6. Beende den Algorithmus, falls alle möglichen Touren aller Aufzüge konstruiert wurden. Ansonsten gehe zu Schritt 3.

Algorithmus 3: Algorithmus zur Klassifikation konstruierter Knoten und Touren. Dieser fällt die Entscheidung, ob ein Knoten weiter verzweigt bzw. eine Tour in den Lösungspool χ aufgenommen werden soll.

1 **Algorithmus :** CheckNode

Input : Knoten $v \in K$.

Output : *True*, falls der Knoten v weiter verzweigt werden soll, sonst *false*.

2 **if** $\underline{\tilde{c}}(v) < \theta$ **then**

3 $\quad t \leftarrow v.get\_tour$ `// Die in` v `enthaltene Tour.`

4 $\quad$ **if** t *ist zulässig* **then**

5 $\quad\quad$ **if** $\tilde{c}(t) < \theta$ **then**

1. Füge die Tour dem Lösungspool hinzu
2. Falls eine weitere Tour in χ existiert, die die gleiche Menge optionaler Aufträge bedient, so lösche diese.
3. Aktualisiere die obere Schranke: $\theta = \tilde{c}(t)$

6 $\quad$ **if** $\mathcal{V}_{opt}(v) = \emptyset$ **then**

7 $\quad\quad$ Lösche v `// Da` v `nicht weiter partitioniert werden kann.`

8 $\quad\quad$ **return** false

9 **else**

10 $\quad$ **return** false

11 **return** true

Kapitel 5

Der warmstartende Exact-Replan

Im Laufe dieses Kapitels wollen wir den Exact-Replan-Algorithmus für VZ-Systeme dahin gehend weiterentwickeln, dass dieser die Konstruktion der Touren mithilfe eines warmstartenden Branch & Bound-Verfahrens durchführt. Motivation für diese Weiterentwicklung ist die Vermutung, dass sich die Suchbäume innerhalb eines Schnappschussproblems mit zunehmender Anzahl von Iterationsrunden nur noch in wenigen Knoten und vereinzelten Teilbäumen unterscheiden. Ist dies der Fall, so könnte durch die Verwendung eines warmstartenden Branch & Bound-Verfahrens die mehrfache Konstruktion gleicher Knoten und Teilbäume vermieden und Zeit eingespart werden.

In Kapitel 3 haben wir bereits ein Ähnlichkeitsmaß für Branch & Bound-Bäume konstruiert und in den Tabellen 3.1a und 3.1b auf Seite 21 gesehen, dass die durch den Exact-Replan konstruierten Suchbäume mit zunehmender Anzahl von Iterationsrunden ähnlicher werden. Aus diesem Grund wollen wir zunächst einen einfachen und intuitiven Ansatz zur Realisierung des Warmstarts vorstellen. Aufgrund der daraus resultierenden Rechenergebnisse ist eine weitere Modifikation jedoch unerlässlich. Dazu betrachten wir in Abschnitt 5.2 die Struktur der unteren Schranke der reduzierten Kosten etwas genauer und werden abschließend in Abschnitt 5.3 eine neue Verzweigungsstrategie vorstellen.

5.1 Die Grundversion des Warmstartenden Exact-Replan

Nachdem wir nun viel über die Theorie von Branch & Bound-Verfahren bzgl. verschiedener Zielfunktionen und die Konstruktion eines warmstartenden Verfahrens sowie die Ähnlichkeit zwischen aufeinanderfolgenden Suchbäume im Exact-Replan erfahren haben, kommen wir zum eigentlichen Kernstück dieser Arbeit: Die Anwendung der bisher erläuterten Theorie und die damit verbundene Modifikation des Exact-Replan zu einem warmstartenden Algorithmus. Auch in diesem Abschnitt betrachten wir für die Erläuterung der Änderungen lediglich ein Schnappschussproblem und einen Aufzug. Eine Iterationsrunde ist somit wieder die Konstruktion des Suchbaumes bis zur Erfüllung der entsprechenden Abbruchbedingung. Des Weiteren bezeichnet K wieder die Menge der Teilprobleme, $\tilde{K}$ die der ausgeloteten Knoten und χ den Lösungspool der gefunden Touren. Die Abbruchbedingung ändert sich im Vergleich zum Exact-Replan nicht, sie lautet demnach wieder: *Beende die Suche nach zulässigen Touren, falls* $|\chi| = \mu \in \mathbb{N}$.

Die Entwicklung des warmstartenden Exact-Replan-Algorithmus beginnt mit einer Grundversion, welche die direkte Umsetzung der in Abschnitt 3.3 erläuterten Modifikationen ist. Somit wird die bisherige Konstruktion und Suche nach Touren, welche den Zielfunktionswert verbessern, an das konstruierte Branch & Bound-Verfahren angepasst.

Wir haben am Ende von Abschnitt 3.3.1 bereits gesehen, dass aufeinanderfolgende Suchbäume im Exact-Replan von Iterationsrunde zu Iterationsrunde immer ähnlicher werden. Die Idee hinter dem warmstartenden Exact-Replan-Algorithmus ist es, die bereits gewonnen Informationen der vorherigen Iterationsrunde zu nutzen um Arbeit zu sparen, z. B. bei der Konstruktion des Suchbaumes. Denn wir haben in Korollar 3.11 gesehen, dass jeder Suchbaum einer Iterationsrunde Teilbaum eines gemeinsamen vollständigen Baumes ist.

Zur Sicherung der Korrektheit wurde im Abschnitt 3.3, „Eines warmstartendes Branch & Bound-Verfahren“, zu Beginn jeder neuen Iterationsrunde eine Neuberechnung der unteren Schranke der reduzierten Kosten aller ausgeloteten Knoten und verworfenen Touren gefordert. Diese Neuberechnung setzen wir in Form von Algorithmus InitSearch auf Seite 37 um. Wir bezeichnen diesen Vorgang als die *Initialisierung des Warmstartes*. Befinden wir uns in der ersten Iterationsrunde, so wird wie gewohnt fortgefahren, d. h. die konstruierten Wurzelknoten werden mit den Methoden aus Kapitel 4 verzweigt und bewertet. In allen anderen Iterationsrunden $i > 1$ gibt es jeweils drei Phasen.

- Aktualisierung der Kosten der in Σ gespeicherten Lösungen.
- Aktualisierung aller bereits ausgeloteten Knoten in $\tilde{K}^{i-1}$.
- Aktualisierung der noch nicht ausgeloteten und zu verzweigenden Blätter K^{i-1} des Suchbaumes der vorherigen Iterationsrunde $i-1$.

Die Reihenfolge, in der diese Phasen durchgeführt werden, kann beliebig gewählt werden. Es besteht z. B. die Möglichkeit, die Aktualisierung der gespeicherten Touren Σ nur in den ersten Iterationsrunden und ganz am Ende durchzuführen, also eine dynamische Gestaltung. In dieser Arbeit haben wir uns dafür entschieden in jeder Iterationsrunde jede Phase durchzuführen und dies in der oben angeführten Reihenfolge, also ein statisches Vorgehen.

Um zwischen Entscheidungen aus der $(i-1)$-ten Iterationsrunde und der i-ten besser unterscheiden zu können bezeichnen wir die entsprechenden Schranken mit θ^{i-1} und θ^i. In der ersten Phase betrachten wir alle zwischengespeicherten Touren $t \in \Sigma$. Für diese Touren galt in der vorherigen Iterationsrunde $\tilde{c}_{i-1}(t) \geq \theta^{i-1}$. Nun beginnt die Neuberechnung der reduzierten Kosten von t. Die reduzierten Kosten einer Tour t sind durch $\tilde{c}_i(t) = c_i(t) - \sum_{\varrho \in t \cap \mathcal{R}} \pi_\varrho - \pi_e$ gegeben. Sind die reduzierten Kosten kleiner als θ^i, muss eine Klassifikation der Tour stattfinden. Im Exact-Replan aus [Hil10] gilt zum Zeitpunkt einer Klassifikation stets $|\chi| < \mu$. Während der Initialisierung des Warmstartes können wir diese Abbruchbedingung jedoch außer Acht lassen, wenn die Neuberechnung der Kosten vergleichsweise günstig ist. Es werden nur Touren aus χ entfernt und wieder zu Σ hinzugefügt, falls eine weitere Tour in Σ gefunden wird, die

Algorithmus 4: Pseudocode zur Initialisierung des Warmstarts. Dabei werden die unteren Schranken der reduzierten Kosten der zwischengespeicherten Knoten und Touren aktualisiert und mit dem Schwellenwert θ verglichen.

1 **Algorithmus :** InitSearch

Input : Iterationsrunde $i \geq 2$, ausgelotete Knotenmenge $\tilde{K}^{i-1}$, bereits gefundene Touren Σ, Blätter K^{i-1} des Suchbaumes der vorherigen Iterationsrunde.

Output : Status: *Genügend zulässige Touren gefunden* oder *Baum initialisiert*, und aktualisierte Mengen $K^i, \tilde{K}^i$ und Σ.

2 $\theta \leftarrow 0$
3 **foreach** $t \in \Sigma$ **do**
4 Aktualisiere $\tilde{c}_i(t)$
5 **if** $\tilde{c}_i(t) < \theta$ **then**
6 **if** *t wurde zu χ hinzugefügt* **then**
1. Aktualisiere $\theta = \tilde{c}_i(t)$.
2. Aktualisiere $\Sigma \leftarrow \Sigma \setminus t$.
7 **if** $|\chi| \geq \mu$ **then**
8 **return** Genügend zulässige Touren gefunden
9 $\hat{K} \leftarrow \emptyset$ `// Wiederbelebte Knoten aus` $\tilde{K}^i$
10 **foreach** $v \in \tilde{K}^i$ **do**
11 Aktualisiere $\underline{\tilde{c}}(v)$
12 **if** $\underline{\tilde{c}}(v) < \theta$ **then**
13 $\tilde{K}^i \leftarrow \tilde{K}^i \setminus v$
14 $\hat{K} \leftarrow \hat{K} \cup v$
15 **foreach** $v \in K^i$ **do**
16 Aktualisiere $\underline{\tilde{c}}(v)$
17 **if** $\underline{\tilde{c}}(v) \geq \theta$ **then**
18 $K^i \leftarrow K^i \setminus v$
19 $\tilde{K}^i \leftarrow \tilde{K}^i \cup v$
20 $K^{i+1} \leftarrow K^i \cup \hat{K}$
21 $\tilde{K}^{i+1} \leftarrow K^i$
22 **return** Baum initialisiert

die gleichen Aufträge bedient und geringere reduzierten Kosten hat. Diese Phase ist in Algorithmus InitSearch, Zeile $2-5$ zu finden. In dieser Phase betrachten wir alle in Σ gespeicherten Touren und beenden die Initialisierung, falls nach der Aktualisierung aller Touren $|\chi| \geq \mu$ gilt. Diese Vorgehensweise ist jedoch nicht zwingend, es kann auch nur eine gewisse Anzahl an Touren überprüft werden oder nur Touren, die aufgrund eines bestimmten Kriteriums ausgewählt wurden. Für die Korrektheit des Algorithmus müssen jedoch spätestens in der letzten Iterationsrunde alle Touren überprüft werden.

Phase 2, welche in Algorithmus InitSearch in den Zeilen $9-14$ zu finden ist, befasst sich mit der Aktualisierung der unteren Schranke der ausgeloteten Knoten $v \in \tilde{K}^i$. Für jeden Knoten $v \in \tilde{K}^i$ galt in der letzten Iterationsrunde $\underline{\tilde{c}}_{i-1}(v) \geq \theta^{i-1}$. Es erfolgt eine Neuberechnung der reduzierten Kosten von v. Gilt nach dieser Neuberechnung $\underline{\tilde{c}}_i(v) < \theta^i$, so muss der Knoten weiter verzweigt werden, wir nehmen v aus diesem Grund wieder in die Menge der Teilprobleme auf und entfernen ihn aus $\tilde{K}^i$.

Die dritte und letzte Phase (Zeile $14-18$) überprüft die Menge der Teilprobleme K^i der letzten Iterationsrunde. In ihr wird entschieden, ob ein Knoten v ausgelotet und damit zu $\tilde{K}^i$ hinzugefügt wird oder ob dieser weiter verzweigt werden soll. Eine Auslotung erfolgt, falls die untere Schranke $\underline{\tilde{c}}_i(v)$ nicht kleiner als θ^i ist. Bei der Implementierung ist darauf zu achten, dass Knoten, die in Phase 2 wieder in die Menge der Teilprobleme aufgenommen wurden, nicht ein zweites Mal in Phase 3 überprüft werden. Damit würde unnötige Mehrarbeit entstehen, die jedoch nichts an der Korrektheit ändert. Die aktualisierte Menge der Teilprobleme und die der ausgeloteten Knoten bezeichnen wir mit K^{i+1} und $\tilde{K}^{i+1}$.

Nachdem die Initialisierung des Warmstarts abgeschlossen ist, beginnt die Konstruktion neuer Touren wie im Exact-Replan aus [Hil10], welchen wir in Kapitel 4 erläutert haben – wir springen direkt zur Verzweigung und Bewertung der Teilmenge K (Abschnitt 4.3). Mit Exact-Replan+WS bezeichnen wir im Folgenden die Grundversion des warmstartenden Exact-Replan.

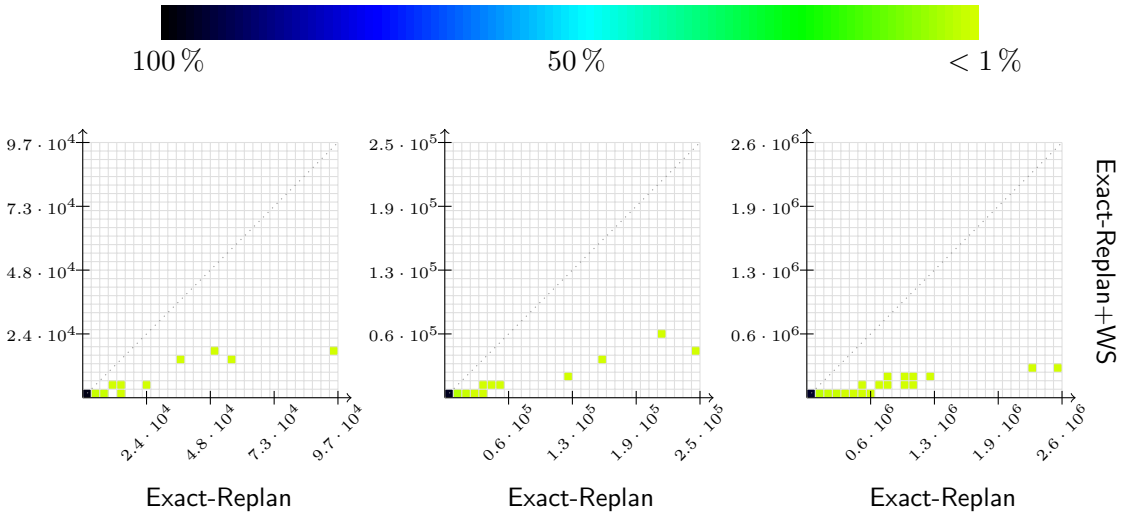
Exact-Replan vs. Exact-Replan+WS

In den nachfolgenden Tabellen präsentieren wir einen ersten Vergleich zwischen Exact-Replan und Exact-Replan+WS für Gebäude A. Eine Auswertung für die Gebäude B und C ist in Anhang B.2 ab Seite 66 zu finden. In der Tabelle 5.1 ist in der ersten Spalte die gesamte Rechenzeit für die Instanz angegeben, d. h. summiert über alle Schnappschussprobleme und Aufzüge. Das gleiche trifft für die zweite Spalte zu, in dieser stellen wir die benötigte Zeit für die Berechnung der unteren Schranke der erzeugten Knoten. Die Spalte *anteilige Rechenzeit* ist das Verhältnis der Rechenzeit des Exact-Replan zur Rechenzeit des Exact-Replan+WS, also die relative Rechenzeit der Grundversion im Vergleich zur Version aus [Hil10]. Eine Zeitersparnis (< 1) ist durch grün eingefärbte und ein Zeitverlust (≥ 1) ist durch rot eingefärbte Werte verdeutlicht. Zusätzlich sind kürzere Rechenzeiten hervorgehoben. Die Instanzen des (Real) Down bzw. (Real) Up Peaks sind mit (R)DP bzw. (R)UP abgekürzt. Die Abkürzungen IF und LP stehen für den Interfloor und den Lunch Peak. Die Zahlen 1125, 1047 und 2026 entsprechen der Anzahl der ankommenden Passagiere innerhalb einer Stunde. Weitere Informationen, wie z. B. die Anzahl der Schnappschussprobleme innerhalb einer Instanz, sind in Anhang A zu finden.

Instanz	Exact-Replan		Exact-Replan+WS		
	Gesamt	LB	Gesamt	LB	anteilige Rechenzeit
DP 1125	0:01:08	0:00:04	**0:01:01**	0:00:11	0.897
DP 1407	**0:02:14**	0:00:09	**0:02:13**	0:00:35	0.993
DP 2026	**0:13:44**	0:01:32	0:20:26	0:09:41	1.488
IF 1125	**0:01:57**	0:00:09	0:02:04	0:00:38	1.060
IF 1407	**0:08:35**	0:00:58	0:16:17	0:08:15	1.897
IF 2026	**7:23:41**	1:18:14	20:24:05	14:44:22	2.759
LP 1125	0:00:52	0:00:02	**0:00:46**	0:00:06	0.885
LP 1407	0:02:26	0:00:10	**0:02:23**	0:00:39	0.979
LP 2026	**1:52:44**	0:16:39	2:31:22	1:38:29	1.343
RDP 1125	0:00:55	0:00:02	**0:00:48**	0:00:07	0.873
RDP 1407	**0:01:59**	0:00:08	**0:01:59**	0:00:30	1.000
RDP 2026	**0:12:18**	0:01:25	0:20:42	0:10:06	1.683
RUP 1125	0:03:23	0:00:22	**0:01:46**	0:00:32	0.522
RUP 1407	0:09:45	0:01:10	**0:04:56**	0:01:48	0.506
RUP 2026	3:31:30	0:30:01	**1:08:29**	0:34:05	0.324
UP 1125	0:01:45	0:00:09	**0:00:54**	0:00:09	0.514
UP 1407	0:45:02	0:06:21	**0:17:55**	0:06:38	0.398
UP 2026	29:36:37	5:29:46	**8:19:39**	4:42:35	0.281

Tabelle 5.1: Laufzeitvergleich zwischen dem Exact-Replan und der Grundversion des warmstartenden Algorithmus (Exact-Replan+WS) auf den Instanzen des Gebäude A. Die Zeiten sind in dem Format hh:mm:ss angegeben. Unter *Gesamt* ist die benötigte Rechenzeit aller Schnappschussprobleme aufaddiert. *LB* ist die benötigte Zeit für die Aktualisierung der unteren Schranke der reduzierten Kosten der zwischengespeicherten Knoten und Touren. Unter *anteilige Rechenzeit* ist der Quotient (Rechenzeit Exact-Replan+WS)/(Rechenzeit Exact-Replan) zu verstehen.

Der Tabelle 5.1 ist zu entnehmen, dass jeweils die beiden leichtesten Instanzen mit 1125 und 1407 Fahrgästen sowohl vom Exact-Replan als auch vom Exact-Replan+WS sehr schnell gelöst werden. Die Rechenzeiten in der Instanz Down Peak 1404 weisen eine Differenz von einer Sekunde auf. Diese Differenz könnte auch auf Messfehler zurückzuführen sein. Wir können daher annehmen, dass beide Algorithmen auf dieser Instanz gleich schnell sind. Auf der Up Peak Instanz mit 1407 Fahrgästen ist bereits ein deutlicher Unterschied zu erkennen. Diese Instanz wird vom Exact-Replan ebenfalls in unter einer Stunde Rechenzeit gelöst – eine Stunde entspricht der simulierten Betriebsdauer – kommt dieser jedoch bedeutend näher, als alle anderen Instanzen mit 1407 oder weniger Fahrgästen. An den 1125er und 1407er Instanzen ist ebenfalls abzulesen, dass Exact-Replan+WS bereits minimale Vorteile gegenüber dem Exact-Replan aufweist. Minimale Unterschiede von etwa einer Sekunde können jedoch aufgrund von Messungenauigkeiten entstehen. Ausgenommen davon sind jedoch die Interfloor Instanzen sowie Real Down Peak 1407. Im Gegensatz dazu kommt es auf der Interfloor Instanz mit 2026 Fahrgästen zu einem Zeitverlust von 175.9 %. Ein wesentlicher Grund dafür ist die gestiegene Zeit, welche in die Aktualisierung der unteren Schranken der zwischengespeicherten Knoten investiert werden muss. Schauen wir wieder auf die Interfloor Instanz mit 2026 Fahrgästen, so erhöht sich die benötigte Zeit für die Aktualisierung



(a) Real Up Peak 1125 **(b)** Real Up Peak 1407 **(c)** Real Up Peak 2026

Abbildung 5.1: In den oben stehenden Abbildungen ist die Anzahl der erzeugten Knoten, welche für die Lösung eines Schnappschussproblems mit dem Exact-Replan Algorithmus benötigt wird, der Anzahl der benötigten Knoten zur Lösung des gleichen Schnappschussproblems mit dem Exact-Replan+WS Algorithmus gegenübergestellt. Die horizontale Achse repräsentiert die Anzahl der durch den Exact-Replan erzeugten Knoten. Auf der vertikalen Achse kann die Anzahl der durch den Exact-Replan+WS erzeugten Knoten abgelesen werden. Aufgrund der großen Datenmenge wurde das Koordinatensystem gerastert. Die Färbung eines jeden Feldes $[x, x+\varepsilon] \times [y, y+\varepsilon]$ repräsentiert den prozentualen Anteil der Schnappschussprobleme, für deren Lösung der Exact-Replan zwischen x und $x+\varepsilon$ viele Knoten und der Exact-Replan+WS zwischen y und $y+\varepsilon$ viele Knoten erzeugt hat. Die gestrichelte Diagonale repräsentiert den Bereich in dem es keine signifikanten Veränderungen gibt. Die Diagramme sind wie folgt zu interpretieren. Je flacher der Verlauf der Punktwolke ist, desto größer sind die Einsparungen an erzeugten Knoten durch die Verwendung des Warmstarts.

der unteren Schranke der reduzierten Kosten um einen Faktor von etwa $11,3$. Auf Real Up Peak 2026 erhöht sich die benötigte Zeit für die Aktualisierung zwar auch, trotzdem kann die Gesamtzeit so weit herabgesetzt werden, dass diese fast der Simulationsdauer von einer Stunde entspricht. Der Exact-Replan benötigte hier mehr als dreimal so lange. Dieser positive Effekt liegt an der verringerten Anzahl der erzeugter Knoten. Eine grafische Darstellung des Vergleiches zwischen den mit Exact-Replan erzeugten Knoten mit den Exact-Replan+WS erzeugten Knoten ist in Abbildung 5.1 zu finden. Die durch den Warmstart erhoffte Zeit- und Arbeitsersparnis ist besonders deutlich auf allen Up Peak Instanzen zu erkennen. So kommt es auf Up Peak 2026 aufgrund signifikant weniger erzeugter Knoten sogar zu einem einer Zeitersparnis von 71.9 %.

Die Auswertung der Ergebnisse des Exact-Replan+WS für die Gebäude B und C sind im Anhang B.2 zu finden. In diesem Abschnitt ist ebenfalls eine Gegenüberstellung der Anzahl der erzeugten Knoten in den jeweiligen Schnappschüssen zu finden, analog zu den Abbildungen 5.1 (a) - (c).

Wir haben nun festgestellt, dass durch den Exact-Replan+WS bereits viele Knoten eingespart werden können. Betrachten wir die benötigte Zeit für die Aktualisierung der unteren Schranken der zwischengespeicherten Knoten, so können wir feststellen, dass durch diesen Anstieg der gewonnen Vorteil auf fast allen Instanzen vollständig

aufgebraucht bzw. überschritten wird. Diesem Problem wollen wir uns nun im nächsten Abschnitt widmen.

5.2 Aktualisieren der unteren Schranken der reduzierten Kosten der Knoten

Im Gegensatz zum Exact-Replan-Algorithmus aus [Hil10] muss im Exact-Replan+WS die untere Schranke der Kosten eines Knotens v nicht nur einmal, sondern zu Beginn jeder neuen Iterationsrunde berechnet werden. Für eine einfachere Notation unterteilen wir die Aufträge $\mathcal{R}_u$ eines Knotens v in bereits zugewiesene, aber noch nicht geladene A_v und in optionale Aufträge O_v. Wie im vorherigen Kapitel ist $\underline{c}$ die untere Schranke der Primalkosten sowie π_ϱ bzw. π_e der Dualpreis für Auftrag ϱ bzw. Aufzug e. Des Weiteren bezeichnen wir die in v enthaltene Tour mit t_v. Jeder Knotens v besitzt mindestens reduzierte Kosten von

$$\begin{aligned}
\underline{\tilde{c}}(v) &= c(t_v) && \text{(bisherige Kosten der Tour)}\\
&+ \sum_{\varrho \in A_v} \underline{c}(\varrho) && \text{(zugewiesene Aufträge)}\\
&+ \sum_{\varrho \in O_v \,:\, \underline{c}(\varrho) - \pi_\varrho < 0} (\underline{c}(\varrho) - \pi_\varrho) && \text{(optionale Aufträge)}\\
&- \pi_e. && \text{(Dualpreise Aufzug } e\text{)}
\end{aligned}$$

Betrachten wir die jeweiligen Summanden, so stellen wir fest, dass die Kosten der bisherigen Tour sowie die unteren Schranken Kosten der bereits zugewiesenen Aufträge in jeder Iterationsrunde konstant sind. Dazu sehen wir uns die jeweiligen Summanden genauer an. Die Kosten einer Tour sind in [Hil10] durch

$$c(t_v) = \sum_{c \in \mathcal{C}(t_v)} (\lambda_\omega \tau_{\text{wait}}(c) + \lambda_r \tau_{\text{ride}}(c) + \lambda_t \tau_{\text{travel}}(c))$$

gegeben. Wobei $\mathcal{C}(t_v)$ die Menge der einzelnen Zielrufe ist, welche sich sich bereits im Aufzug befindet. Des Weiteren sind $\tau_{\text{wait}}, \tau_{\text{ride}}$ und τ_{travel} die Warte-, Kabinen- und Reisezeiten der einzelnen Zielrufe $c \in \mathcal{C}$. Als Wartezeit wird die Zeitspanne zwischen der Auslösung des Rufes und dem Einladen durch einen entsprechenden Aufzug bezeichnet, als Kabinenzeit wird die Zeit innerhalb eines Aufzuges bezeichnet, d. h. die Zeitspanne zwischen Ein- und Ausladen. Die Reisezeit ist die Summe beider Zeiten. $\lambda_\omega, \lambda_r$ und λ_t sind Gewichte für die einzelnen Zeiten. Die Kosten eines Auftrages, egal ob dieser bereits zugewiesen wurde oder nicht, sind in [Hil10] durch die Summe unterer Schranken für die Warte-, Kabinen- und Reisezeiten gegeben. Auf die genaue Berechnung dieser Schranken wollen wir an dieser Stelle nicht eingehen und verweise daher auf [Hil10], es sei jedoch angemerkt, dass sich diese Schranken aus den Fahrzeiten zwischen den entsprechenden Start- und Zieletagen, der in einem Ruf ϱ enthaltenen Zielrufe c sowie der Stopp- und Ladezeiten des entsprechenden Aufzugs zusammensetzen. Diese Werte sind innerhalb eines Schnappschussproblems offensichtlich konstant.
In zwei aufeinanderfolgenden Iterationsrunden ändern sich somit höchstens die Dualpreise π_ϱ der Aufträge $\varrho \in O_v$ sowie der Dualpreis π_e des dazugehörigen Aufzuges e. Wir können somit die untere Schranke der reduzierten Kosten eines Knotens als die

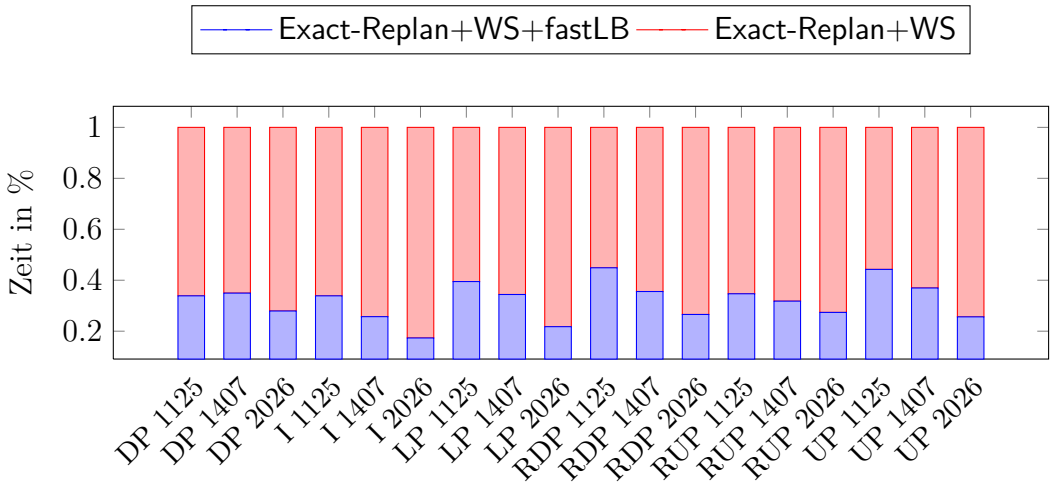


Abbildung 5.2: Gebäude A. Grafische Darstellung der Reduzierung der benötigten Rechenzeit für Aktualisierung der unteren Schranke der reduzierten Kosten unter Ausnutzung der in Abschnitt 5.2 erläuterten Struktur.

Summe aus den Dualpreisen, π_e und π_ϱ, und einer unteren Schranken der Primalkosten $\underline{c}$ darstellen:

$$\underline{\tilde{c}}(v) = \underbrace{c(t_v) + \sum_{\varrho \in A_v} \underline{c}(\varrho) + \sum_{\varrho \in O_v \,:\, \underline{c}(\varrho) - \pi_\varrho < 0} \underline{c}(\varrho)}_{=\underline{c}(v)} - \sum_{\varrho \in O_v \,:\, \underline{c}(\varrho) - \pi_\varrho < 0} \pi_\varrho - \pi_e .$$

Mit dieser Erkenntnis genügt es für jeden Knoten v seine untere Schranke der Primalkosten $lbc(v)$ zu speichern und in jeder Iterationsrunde von diesen die entsprechenden Dualpreise abzuziehen, anstelle einer vollständigen Neuberechnung der unteren Schranke der reduzierten Kosten. Diese Eigenschaft haben wir uns zunutze gemacht. Dieser kleine aber feine Unterschied in der Implementierung hat signifikante Auswirkungen. Wir haben damit ein erstes und auch wichtiges Kriterium für eine effizientere Reoptimierung entwickelt. Sprechen wir vom warmstartenden Exact-Replan in Verbindung mit der schnellen Aktualisierung, so schreiben wir Exact-Replan+WS+fastLB. Für das Gebäude A ist eine Gegenüberstellung der benötigten Aktualisierungszeiten in Abbildung 5.2 zu sehen. Eine analoge Darstellung für die Gebäude B und C ist in Anhang B.3 zu finden. In dem Diagramm ist die Zeit für die Berechnung der unteren Schranke ohne die Ausnutzung der speziellen Struktur auf 1 skaliert. Die benötigte Rechenzeit mit Ausnutzung dieser Struktur wurde in Relation dazu gesetzt.

In Abschnitt 4.3.2 haben wir über die Konstruktion des Suchbaumes im Exact-Replan gesprochen. Die Verzweigungsmöglichkeiten sind dabei die anzufahrenden Etagen, entweder um einen Auftrag aufzunehmen oder um einen Auftrag auszuladen. Dabei kamen aufgrund von Satz 4.2 nur Etagen in Frage, welche Startetage eines Auftrages ϱ sind, deren untere Schranke der Kosten $\underline{c}(\varrho)$ vollständig durch den Dualpreis π_ϱ gedeckt wird, d. h. $\underline{c}(\varrho) - \pi_\varrho < 0$. In dem bisherigen Warmstart können wir diesen Satz nicht anwenden und haben damit einen bedeutend größeren Pool von Verzweigungsmöglichkeiten. In dem folgenden Abschnitt wollen wir erörtern warum dieser Satz nicht bedenkenlos im Warmstart angewendet werden kann und wie der Verzweigungsprozess (Branching) modifiziert werden muss um die Erkenntnisse aus Satz 4.2 nutzen zu können.

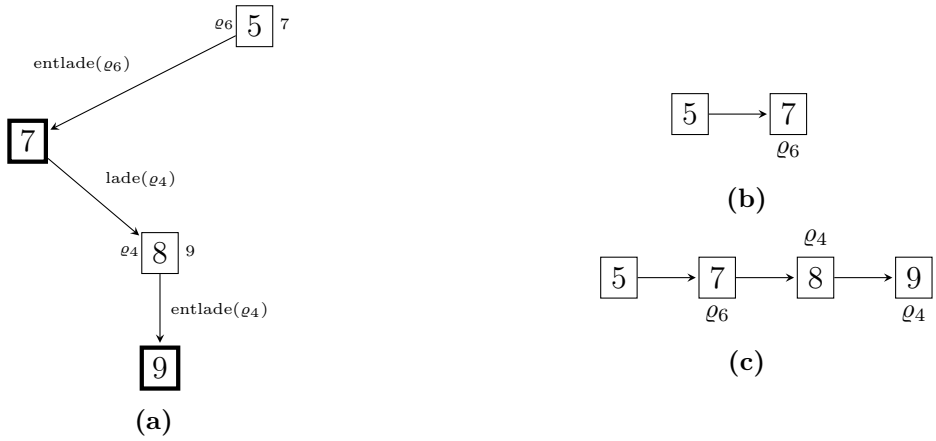


Abbildung 5.3: Darstellung des Suchbaumes unter Verwendung des Exact-Replan+WS+fastLB und Satz 4.2. Im Aufzug befindet sich ϱ_6, mit der Zieletage 7. Noch nicht bediente Rufe sind $\varrho_2 : 6 \to 9$ und $\varrho_4 : 8 \to 9$, wobei $\underline{c}(\varrho_2) - \pi_{\varrho_2} \geq 0$ und $\underline{c}(\varrho_4) - \pi_{\varrho_4} < 0$ gilt. Die Start- und Zieletage von Auftrag ϱ_2 stellt aufgrund von Satz 4.2 keine Verzweigungsoption dar. Nachdem die Tour (c) gefunden wurde ist die Abbruchbedingung erfüllt. Dieser Suchbaum ist gleichzeitig der vollständige Suchbaum.

5.3 Das Pseudo-Branching

Die Problematik, die entsteht, wenn wir Satz 4.2 auch innerhalb des Warmstarts anwenden, ist der Verlust der Korrektheit. Um dieses Problem zu verdeutlichen, betrachten wir wieder das Beispiel 4.1 für einen vollständigen Suchbaum aus Abschnitt 4.3.2. In diesem Beispiel befindet sich der Aufzug auf Etage 5 und hat den Auftrag ϱ_6 mit der Zieletage 7 geladen. Die Aufträge ϱ_2 und ϱ_4 wollen auf Etage 6 bzw. 8 zusteigen und haben die gemeinsame Zieletage 9. Als Abbruchbedingung wählen wir $|\chi| = 2$. Wir suchen demnach in jeder Iterationsrunde maximal 2 neue Touren und beenden diese Iterationsrunde, falls $|\chi| = 2$ gilt oder es keine weiteren Touren gibt. In der ersten Iterationsrunde seien die Dualpreise π_{ϱ_2} und π_{ϱ_4} so, dass $\underline{c}(\varrho_2) - \pi_{\varrho_2} \geq 0$ und $\underline{c}(\varrho_4) - \pi_{\varrho_4} < 0$ gilt und erst in der darauf folgenden Iterationsrunde möge sich der Dualpreis π_{ϱ_2} so ändern, dass $\underline{c}(\varrho_2) - \pi_{\varrho_2} < 0$ gilt. Wenden wir nun den Satz über die untere Schranke der reduzierten Kosten an, so kommen die Startetage von Auftrag ϱ_2 nicht als Verzweigungsoptionen im Wurzelknoten – und damit in allen weiteren Knoten – in Frage. Die Etagen 6, 8 und 9 sind somit alle Verzweigungsoptionen, die je nach Position des Aufzuges und verpflichtenden Entladestopps möglich sind. In der ersten Iterationsrunde entsteht der Suchbaum 5.3 (a) mit den zulässigen Touren (b) und (c), für diese setzen wir voraus, dass ihre reduzierten Kosten negativ genug sind, so dass beide Touren in den Lösungspool χ aufgenommen werden. Am Ende der ersten Iterationsrunde ist Abbruchbedingungen erfüllt, der Lösungspool χ hat seine maximale Kardinalität erreicht und es können keine weiteren Touren konstruiert werden. Die Generierung neuer Touren und der damit korrespondierenden Spalten wird für diesen Aufzug somit beendet.

Betrachten wir nun das Vorgehen des Exact-Replan. Die erste Iterationsrunde verläuft identisch. Für Auftrag ϱ_2 gilt in der zweiten Iterationsrunde ebenfalls $\underline{c}(\varrho_2) - \pi_{\varrho_2} < 0$. Der Pool an potentiellen Verzweigungsoptionen ist damit 6, 7, 8 und 9. Im Laufe der zweiten Iterationsrunden wird ein Baum wie in Abbildung 5.4 (a) dargestellt konstruiert. Die gefundenen zulässigen Touren sind 5.4 (b) und (c). Die Touren $5 \to 7$ und $5 \to 7 \to 8 \to 9$ wurden bereits in der vorherigen Iterationsrunde gefunden und werden nun nicht mehr beachtet, müssen jedoch – da diese Teil anderer Touren sind – konstru-

iert werden. Analog verläuft die dritte Iterationsrunde. Der dazugehörige Suchbaum und die gefundenen zulässigen Touren sind den Abbildungen 5.4 (d) - (f) dargestellt. Jede weitere Iterationsrunde verläuft nach dem gleichen Prinzip, bis der Baum entweder vollständige durchsucht wurde oder die Optimalität mithilfe der in 4.3.1 eingeführten Lasdon-Schranke bewiesen ist.
Durch dieses kleine Beispiel ist deutlich geworden, dass der Exact-Replan+WS+fastLB in Verbindung mit Satz 4.2 unter Umständen nur einen Teil aller zulässigen Touren konstruieren kann. Eine unmittelbare Folge ist der Verlust der Korrektheit. Möglicherweise werden alle Touren gefunden, welche zu einer optimalen Lösung gehören, die Verifizierung als eine optimale Lösung ist jedoch nicht möglich.

Um das eben beschriebene Problem zu lösen und zugleich die Vorteile des Satzes über die untere Schranke der reduzierten Kosten nutzen zu können, modifizieren wir das Branchingverfahren. Im Exact-Replan und Exact-Replan+WS+fastLB wird ein Knoten stets vollständig partitioniert, d. h. jede zulässige Verzweigungsoption (anfahrbare Etage f) in $\mathcal{V}_{opt}(v)$ wird wahrgenommen, es werden somit stets $|\mathcal{V}_{opt}(v)|$ neue Knoten erzeugt. Jede Etage $f \in \mathcal{V}_{opt}(v)$ korrespondiert zu einer Start- oder Zieletage. In Algorithmus Branch auf Seite 48, in dem die genaue Prozedur beschrieben ist, wird in den Zeilen 2 bis 4 die direkte Weiterfahrt zum nächstmöglichen Entladestopp beschrieben. Die Fahrt zu einer Etage auf der ein noch nicht zugewiesener Auftrag aufgenommen werden kann, ist in den Zeilen 5 bis 11 zu finden. Jeder Knoten, der einmal Teil der Eingabemenge von Algorithmus Branch war, kann nach seiner Partitionierung verworfen werden. Um die Modifikation des Branchings beschreiben zu können benötigen wir noch etwas Notation:

- $O_v^< := \{\varrho \in O_v \mid \varrho.start\_floor \in \mathcal{V}_{opt}(v) \text{ und } \underline{c}(\varrho) - \pi_\varrho < 0\} \subseteq O_v$
- $O_v^\geq := \{\varrho \in O_v \mid \varrho.start\_floor \in \mathcal{V}_{opt}(v) \text{ und } \underline{c}(\varrho) - \pi_\varrho \geq 0\} \subseteq O_v$
- Ein Knoten v heißt *partitioniert*, falls $O_v^\geq = O_v^< = \emptyset$ gilt.
- Ein Knoten v heißt *pseudo-partitioniert*, falls $O_v^\geq \neq \emptyset$ und $O_v^< = \emptyset$ gilt.

Betrachten wir nun das Branchingverfahren an einem beliebigen Knoten v. Wir setzen an dieser Stelle voraus, dass sowohl $O_v^<$ als auch $O_v^\geq$ nicht leer sind. Im Laufe des Abschnittes haben wir festgestellt, dass wir die Menge der Verzweigungsoptionen $\mathcal{V}_{opt}(v)$ eines Knotens nicht verändern können ohne die Korrektheit zu gefährden. Aus diesem Grund betrachten wir im Folgenden in Algorithmus Branch in Zeile 5 lediglich die Menge $O_v^<$. Von diesen Aufträgen wissen wir, dass sie dem Satz 4.2 genügen. Haben wir nun alle Knoten erzeugt, die zu einer Startetage eines Auftrages in $O_v^<$ gehören, so ist v pseudo-partitioniert. Diesen Knoten dürfen wir an diesem Punkt noch nicht verwerfen, sondern fügen v der Menge ausgeloteten Knoten $\tilde{K}$ hinzu. Haben sich in einer der folgenden Iterationsrunden die Dualpreise mancher Aufträge in $O_v^\geq$ so weit geändert, dass für diese $\underline{c}(\varrho) - \pi_\varrho < 0$ gilt, fügen wir diese Aufträge der Menge $O_v^<$ hinzu und entfernen diese aus $O_v^\geq$. Der Knoten v kann in dieser Iterationsrunde wiederbelebt und weiter partitioniert werden. Ein solches Vorgehen bezeichnen wir als *Pseudo-Branching*.

In Abschnitt 4.3.3 haben wir den Algorithmus CheckNode vorgestellt. Dieser bekommt einen Knoten v übergeben und entscheidet u. a. ob dieser Knoten weiter verzweigt werden soll. Wir können diesen Algorithmus dahingehend erweitern, dass wir zu Beginn

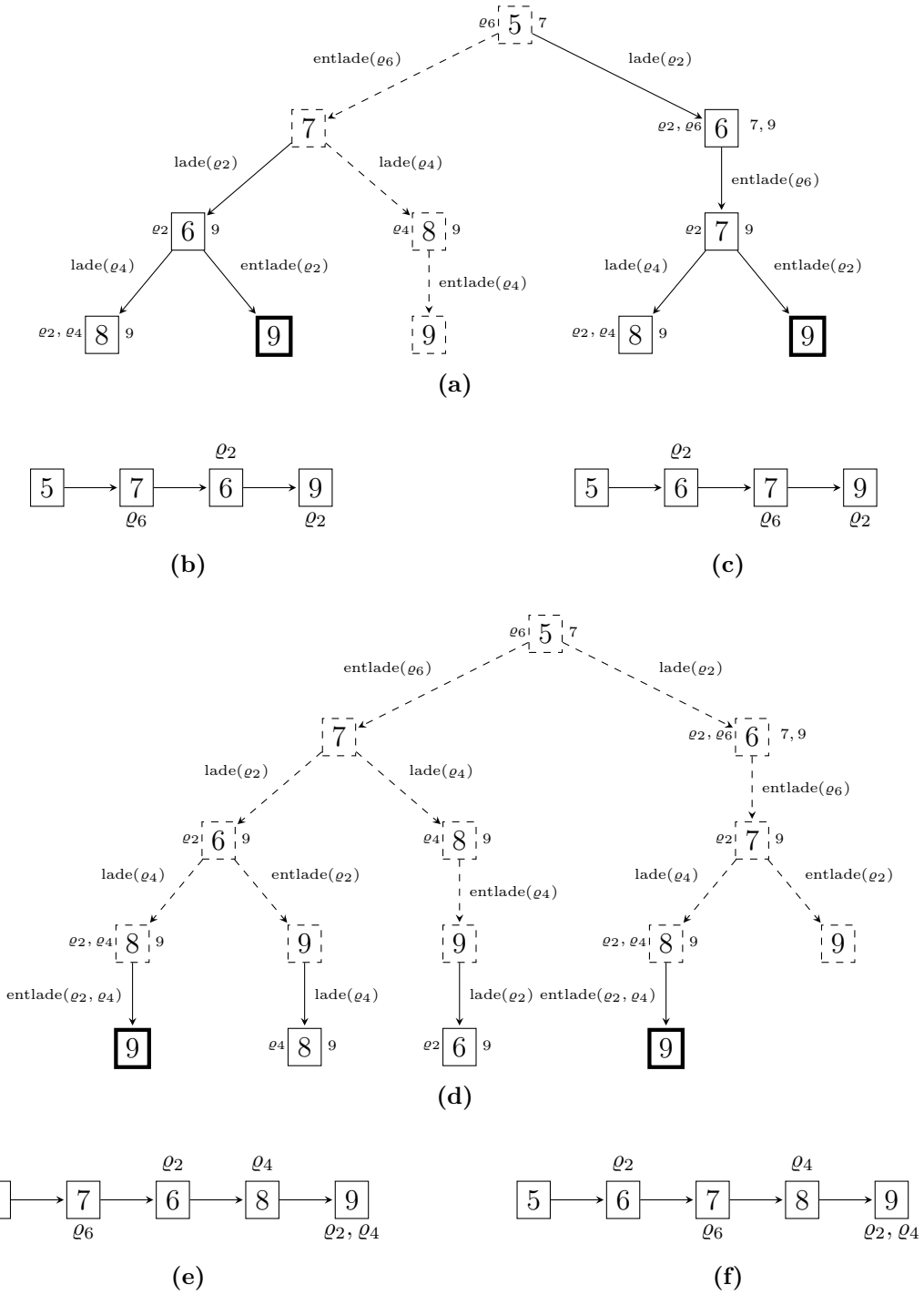


Abbildung 5.4: Darstellung des Suchbaumes bis zum Ende der zweiten Iterationsrunde (a) und bis zum Ende der dritten Iterationsrunde (d) unter Verwendung des Exact-Replan. Im Aufzug befindet sich ϱ_6, mit der Zieletage 7. Noch nicht bediente Rufe sind $\varrho_2 : 6 \to 9$ und $\varrho_4 : 8 \to 9$. Der zur vorherigen Iterationsrunde identische Teilbaum ist gestrichelt dargestellt. Die zur Erfüllung der Abbruchbedingung konstruierten Touren sind (b), (c), (e) und (f).

die Mengen $O_v^<$ und $O_v^\geq$ bestimmen. Gilt für den übergebenen Knoten $O_v^< = \emptyset$, so kann der Algorithmus CheckNode beendet werden, es ist keine Berechnung der unteren Schranke der reduzierten Kosten notwendig, und der Knoten wird der Menge $\tilde{K}$ hinzugefügt. Analog können wir bei der Aktualisierung und Neubewertung zu Beginn jeder neuen Iterationsrunde die Mengen $O_v^<$ und $O_v^\geq$ bestimmen. Gilt nun $O_v^< = \emptyset$, so kann der Knoten ausgelotet werden bzw. verbleibt in $\tilde{K}$.

Um die Vorgehensweise des Pseudo-Branchings zu verdeutlichen nehmen wir uns wieder das Beispiel vom Anfang dieses Abschnittes, auf die konstruierten Touren gehen wir jedoch noch einmal genauer ein.

Zur Erinnerung: Der Auftrag ϱ_6 ist bereits geladen und hat die Zieletage 6. Zusätzlich existieren die Aufträge $\varrho_2 : 6 \to 9$ und $\varrho_4 : 8 \to 9$. Die Abbruchbedingung sei wieder $|\chi| = 2$. Die Zerlegung der optionalen Aufträge ergibt $O_v^< = \{\varrho_4\}$ und $O_v^\geq = \{\varrho_2\}$.

In der ersten Iterationsrunde starten wir mit dem Wurzelknoten v_1, dieser repräsentiert die momentane Position des Aufzuges auf Etage 5. Da $\varrho_2 \in O_v^\geq$ können wir nur einen Kindsknoten von v_1 erzeugen, dieser sei v_2 und repräsentiert den Entladestopp für ϱ_6 auf Etage 7, diesen nehmen wir in die Menge der Teilprobleme auf, $K^1 = \{v_2\}$. Der Knoten v_1 ist nun pseudo-partitioniert und wird ausgelotet, es gilt $\tilde{K}^1 = \{v_1\}$. Wir setzten an dieser Stelle voraus, dass die Klassifizierung von v_2 eine weitere Partitionierung ergibt. Die Zerlegung der optionalen Aufträge von v_2 lautet ebenfalls $O_v^< = \{\varrho_4\}$ und $O_v^\geq = \{\varrho_2\}$. Analog zur Partitionierung von v_1 können wir nur den Knoten v_3 erzeugen, welcher einem Beladestopp von Auftrag ϱ_4 auf Etage 8 entspricht. Somit ist auch v_2 pseudo-partitioniert, es gilt nun $K^1 = \{v_3\}$ und $\tilde{K} = \{v_1, v_2\}$. Da wir in v_3 noch keine zulässige Tour haben, wird dieser weiter partitioniert. An dieser Stelle haben wir jedoch nur eine Möglichkeit: die direkte Weiterfahrt zu Etage 9 um Auftrag ϱ_4 auszuladen. Diesen Knoten bezeichnen wir mit v_4, welcher jetzt der einzige Knoten in der Menge der Teilmengen ist. $\tilde{K}^1$ bleibt unverändert. Die Knoten v_2 und v_4 enthielten zulässige Touren, welche wie zu Beginn dem Lösungspool hinzugefügt wurden. Die Abbruchbedingung ist somit erfüllt.

Wie im obigen Beispiel nehmen wir an, dass sich am Ende der ersten Iterationsrunde der Dualpreis von Auftrag ϱ_2 so geändert hat, dass dieser die untere Schranke der Primalkosten vollständig deckt. Die Zerlegung der optionalen Aufträge ist somit $O_v^< = \{\varrho_2\}$ und $O_v^\geq = \emptyset$. Aus diesem Grund müssen wir gemäß der Konstruktion eines warmstartenden Branch & Bound-Verfahrens eine Aktualisierung und Neubewertung der Knoten vornehmen. Wir nehmen an, die unteren Schranken aller Knoten in K^1 und $\tilde{K}^1$ sind so niedrig, dass nach der Aktualisierung und Neubewertung $K^2 = \{v_1, v_2, v_4\}$ gilt. Ausgehend von Knoten v_1 besteht lediglich Möglichkeit zu Etage 6 zu fahren. Diesen Knoten bezeichnen wir als v_5 und er repräsentiert den zu ϱ_2 gehörigen Beladestopp. Dieser Knoten ist weder partitioniert noch pseudo-partitioniert, er wird somit in die Menge der Teilprobleme aufgenommen. Auch für Knoten v_2 existiert eine offene Verzweigungsoption, die Fahrt zu Etage 6, auf der ebenfalls Auftrag ϱ_2 aufgenommen wird. Dieser Knoten sei mit v_6 bezeichnet. Nach diesem Schritt beinhaltet die Menge der Teilprobleme die Knoten v_4, v_5 und v_6. Diese werden analog zu den Knoten v_1, v_2 und v_3 partitioniert. Am Ende der zweiten Iterationsrunde wurde ein Suchbaum wie Abbildung 5.4 (a) konstruiert.

Satz 5.1:
Sei S ein optimaler Fahrplan, welcher durch den Exact-Replan *aus [Hil10] berechnet wurde. Der* Exact-Replan+WS+fastLB *in Verbindung mit dem Pseudo-Branching liefert einen Fahrplan $\tilde{S}$, für den*

$$c(S) = c(\tilde{S})$$

gilt.

Beweis: Sei $e \in \mathcal{E}$ ein beliebiger Aufzug und T_i der Suchbaum am Ende der i-ten Iterationsrunde im Exact-Replan. Des Weiteren betrachten wir den Baum $\tilde{T}_i$, welcher durch den Exact-Replan+WS+fastLB in Verbindung mit dem Pseudo-Branching in der i-ten Iterationsrunde konstruiert wurde. Da beide Algorithmen (Exact-Replan und Exact-Replan+WS+fastLB mit dem Pseudo-Branching) das gleiche Abbruchkriterium haben und die Auswahl der Knoten deterministisch erfolgt, können wir davon ausgehen, dass das Abbruchkriterium der vollständigen Konstruktion des Suchbaumes entspricht.

Behauptung: $T_i \subseteq \tilde{T}_i$, für jede Iterationsrunde i.

Angenommen es existiert ein Knoten $v \in T_i$, welcher nicht in $\tilde{T}_i$ enthalten ist. Das bedeutet, es gibt einen Vorgänger u von v, welcher im Laufe des Pseudo-Branchings ausgelotet wurde. Nach Konstruktion loten wir einen Knoten genau dann aus, wenn dieser vollständig partitioniert oder pseudo-partitioniert ist. Wäre u vollständig partitioniert, so erhalten wir einen Widerspruch zur Existenz von v in T_i. Somit kann u lediglich pseudo-partitioniert sein. Pseudo-partitioniert bedeutet, dass $O_u^{\geq} \neq \emptyset$ und $O_u^{<} = \emptyset$ gelten muss. Der Knoten v korrespondiert zu einer Verzweigung auf die Etage f, welche die Start- oder Zieletage eines Auftrages ϱ ist, für den $\underline{c}(\varrho) - \pi_\varrho < 0$ gilt. Andernfalls wäre f keine Verzweigungsoption in der aktuellen Iterationsrunde des Exact-Replan. Per Konstruktion der Menge $O_u^{<}$ beinhaltet diese den Auftrag ϱ und kann nicht leer sein. Daraus folgt, dass u auch nicht pseudo-partitioniert ist. Dies ist ein Widerspruch zur Annahme $v \notin \tilde{T}_i$. □

Durch die Einsparung von Knoten werden im gleichen Moment auch Touren eingespart, deren untere Schranke der reduzierten Kosten zu groß sind, um diese dem Lösungspool χ hinzuzufügen. Durch all diese Einsparungen sind die zu Beginn jeder Iterationsrunde zu aktualisierenden Mengen $\tilde{K}$ und Σ weitaus kleiner, dies verkürzt offensichtlich auch die Initialisierungszeit des Warmstarts. Ein weiterer positiver Aspekt, welchen wir in dieser Arbeit jedoch außer Acht gelassen haben, ist die Verringerung der notwendigen Ressourcen, wie z. B. Speicherplatz für die Verwaltung der zwischengespeicherten Daten. Den Algorithmus Exact-Replan+WS+fastLB in Verbindung mit dem Pseudo-Branching wollen wir im Folgenden mit Exact-Replan+WS+fastLB+PB bezeichnen. Für das Gebäude A sind die prozentualen Einsparungen im Vergleich zum Exact-Replan+WS+fastLB in Abbildung 5.5 zu sehen.
Die prozentualen Einsparungen der erzeugten Knoten, zwischengespeicherten Touren und der Initialisierungszeit des Warmstartes sind summiert über alle Aufzüge und alle Schnappschussprobleme. Des Weiteren ist der jeweilige Mittelwert angegeben. Durch die Technik des Pseudo-Branchings werden im Mittelwert ca. 23 % der gespeicherten Touren eingespart. Bei den erzeugten Knoten sind es sogar 31 % und die Zeit für die Initialisierung des Warmstartes senkt sich um ca. 25 %.

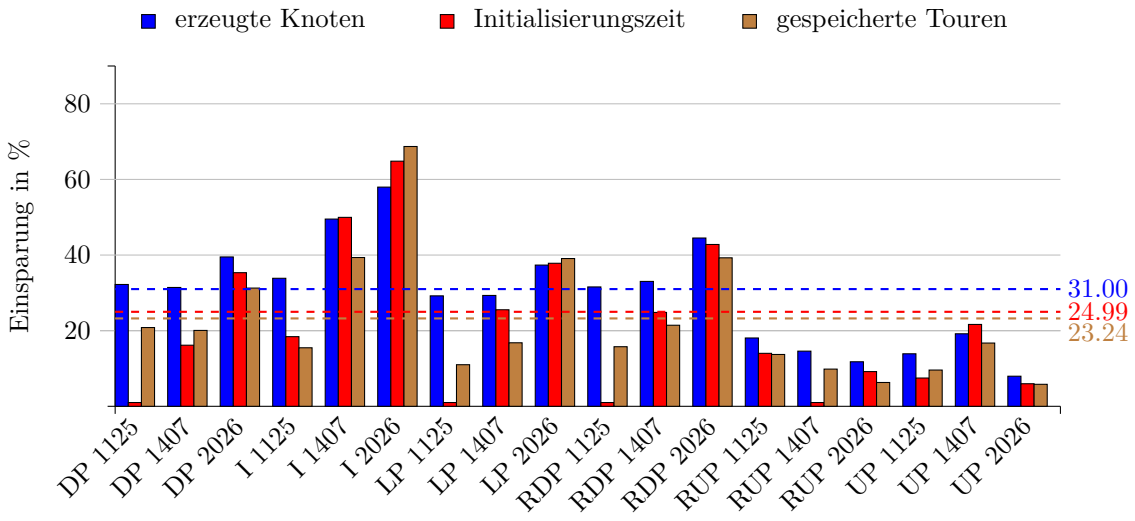


Abbildung 5.5: Prozentuale Einsparungen der erzeugten Knoten, zwischengespeicherten Touren und der Initialisierungszeit des Warmstartes im Exact-Replan+WS+fastLB+PB im Vergleich zu Exact-Replan+WS+fastLB für das Gebäude A. Die Mittelwerte der Ersparnisse sind in den jeweiligen Farben und gestrichelt dargestellt.

Algorithmus 5: Aufspalten eines Knotens v zur Konstruktion neuer zulässiger Touren.

1 **Algorithmus :** Branch

Input : Knoten v mit enthaltener Tour t, welcher verzweigt werden soll und die Menge $\mathcal{V}_{opt}(v)$ aller möglichen Verzweigungsoptionen.

Output : Eine Menge $\hat{V}$ bestehend aus Kinderknoten von v

2 $\hat{V} \leftarrow \emptyset$

3 **if** $A_v \neq \emptyset$ **then**

4 Erzeuge neuen Kinderknoten $\hat{v}$ und erweitere die in ihm enthalte Tour durch einen Entladestopp an der nächst möglichen Etage.

5 $\hat{V} \leftarrow \hat{V} \cup \hat{v}$

6 **foreach** $\varrho \in O_v$ **do**

7 $f \leftarrow \varrho.start\_floor$ `// Startetage von Auftrag` ϱ

8 **if** $f \in \mathcal{V}_{opt}(v)$ **then**

9 Erzeuge neuen Kinderknoten $\hat{v}$ von v und erweitere die in $\hat{v}$ enthalte Tour durch einen Beladestopp auf Etage f.

10 $\hat{V} \leftarrow \hat{V} \cup \hat{v}$

11 $\mathcal{V}_{opt}(v) \leftarrow \mathcal{V}_{opt}(v) \setminus f$

12 $O_v \leftarrow O_v \setminus \varrho$

13 **return** $\hat{V}$

5.4 Rechnerische Auswertung

Im Laufe von Kapitel 5 haben wir bereits erste Zwischenergebnisse für das Gebäude A präsentiert. All diese und auch alle folgenden Berechnungen des Gebäudes A sowie die der Gebäude B und C wurden unter Linux auf einem Intel Core 2 Quad CPU Q9550 @ 2.83 GHz mit einem Arbeitsspeicher von 16 GB durchgeführt. Wir wollen in diesem letzten Abschnitt – neben einem kurzen Résumé der Abschnitte 5.1 und 5.2 – insbesondere auf die letzte Änderung des Exact-Replan eingehen, denn diese habe wir, im Gegensatz zur Grundversion des Warmstartes aus Kapitel 5.1 und zur schnellen Aktualisierung der unteren Schranken aus Kapitel 5.2, nur dem in dieser Arbeit konstruierten Exact-Replan+WS+fastLB gegenübergestellt.

In [Hil10] und im Exact-Replan sowie in unseren Berechnungen werden für die Gewichte der Warte-, Kabinen- und Reisezeit die Werte $\lambda_\omega = \lambda_t = 1$ und $\lambda_r = 0$ gewählt. Mit dieser Wahl der Gewichte wird die Wartezeit doppelt gezählt und somit am meisten berücksichtigt.

Die Grundversion des warmstartenden Exact-Replan aus Kapitel 5.1 beinhaltete lediglich alle notwendigen Modifikationen, um weiterhin Korrektheit garantieren zu können. Diese notwendigen Änderungen haben wir durch die Konstruktion eines warmstartenden Branch & Bound-Verfahrens in Kapitel 3 hergeleitet. Diese Grundversion liefert auf den Instanzen mit 80 % und 100 % der Passagierauslastung bereits gute Ergebnisse und verkürzte die benötigte Rechenzeit. An dieser Stelle sei jedoch gesagt, dass auch der Exact-Replan diese Instanzen in angemessener Zeit löst. Weitaus interessanter sind die Instanzen mit 144 % der Passagierauslastung, da diese vom Exact-Replan nicht in der simulierten Zeit von einer Stunde gelöst werden können. Abgesehen von den Real Up Peak und Up Peak Instanzen benötigt der Exact-Replan+WS die bis zu 3,687-fache Rechenzeit. Diese Laufzeitvergleiche sind in der Tabelle 5.1 auf Seite 39 bzw. in den Tabellen B.17 und B.18 im Anhang auf den Seiten 67 und 68 zu finden. Ein wesentlicher Grund für die gestiegene Laufzeit ist die aufwendige Aktualisierung der unteren Schranken der zwischengespeicherten Knoten. Dieses Problem kann durch Ausnutzung der speziellen Struktur der unteren Schranken weitgehend behoben werden. In der Abbildung 5.2 auf Seite 42 konnten wir bereits sehen, dass die anteilige Rechenzeit für die Aktualisierung der unteren Schranken um mindestens 50 % reduziert werden kann. Für die Gebäude B und C erhalten wir sehr ähnliche Resultate. Eine signifikante Beschleunigung ist auf der Instanz Interfloor 1872 des Gebäudes C zu sehen. Hier kann die benötigte Zeit auf etwa 20 % reduziert werden, es wurden somit „nur“ 4:50 Stunden statt 24:30 Stunden für die Aktualisierung benötigt. Diese Zeiten sind jedoch immer noch jenseits der gewünschten Ergebnisse. Eine grafische Darstellung der reduzierten Aktualisierungszeit der Gebäude B und C ist in den Abbildungen auf Seite 82 zu finden.

Im vorherigen Abschnitt dieses Kapitels haben wir eine Modifikation des Branch&-Bound-Verfahrens vorgestellt. Das so veränderte Verfahren bezeichnen wir als Pseudo-Branching. In der Abbildung 5.5 sowie in den Abbildungen B.37a und B.37b sind die Einsparungen gegenüber dem bisherigen Verzweigungsverfahren in Verbindung mit dem Exact-Replan+WS+fastLB für die Gebäude A, B und C dargestellt. Die entsprechenden Daten sind in den Tabellen B.19, B.20 und B.21 aufgelistet. Betrachten wir die durchschnittlichen Einsparungen über alle Instanzen eines Gebäudes, so ist erkennbar, dass die Technik des PseudoBranchings die meisten Einsparungen in der Anzahl der generierten Knoten bewirkt, gefolgt von der Initialisierungszeit und den zwischenge-

speicherten Touren. Vergleichen wir die Durchschnittswerte aller Gebäude, so stellen wir eine sehr große Ähnlichkeit fest. Diese Ähnlichkeit weist auf eine konstante Effektivität des Pseudo-Branchings hin. Vergleichen wir die einzelnen Instanzen, so können wir erkennen, dass es besonders auf den Interfloor Instanzen zu Einsparungen kommt. Auf den Real Up Peak und Up Peak Instanzen halten sich die Einsparungen hingegen in Grenzen. Wobei nicht außer Acht gelassen werden darf, dass wir bereits mit der Grundversion Exact-Replan+WS die Rechenzeit auf diesen Instanzen stark reduzieren konnten. In den übrigen Instanzen können wir Einsparungen nahe dem Mittelwert verzeichnen.

Nachdem wir in den Kapiteln 5.2 und 5.3 stets Verbesserungen in der Laufzeit, der Anzahl generierter Knoten sowie zwischengespeicherter Touren beobachten konnten, wollen wir den konstruierten Exact-Replan+WS+fastLB+PB mit Exact-Replan vergleichen. Für diesen Vergleich betrachten wir die Tabellen 5.2 - 5.7 auf den Seiten 52 - 57. Zu jedem der drei Gebäude gibt es zwei Tabellen. Die erste Tabelle gibt einen Überblick über die Laufzeiten. In diesen Tabellen ist neben der Rechenzeit auch die für das Branching bzw. die für die Aktualisierung der unteren Schranken benötigte Zeit angegeben. Des Weiteren ist in den Laufzeittabellen die anteilige Rechenzeit des Exact-Replan+WS+fastLB+PB hinsichtlich des Exact-Replan angegeben. Wie auch schon den vorhergehenden Abschnitten ist eine Reduzierung der Rechenzeit grün und eine Erhöhung rot markiert. Zum Anderen haben wir eine Übersicht über die benötigten Iterationsrunden, generierten Knoten sowie zwischengespeicherten Touren und der Initialisierungszeit des Warmstartes zu jedem Gebäude angegeben.

Beginnen wir mit einer Analyse der Ergebnisse für Gebäude A. Betrachten wir zuerst die anteiligen Rechenzeiten. Wir stellen fest, dass es in jeder Instanz zu einer Reduzierung der Laufzeit kommt. Wir erreichen dabei eine Reduzierung um mindestens 31 % bei Lunch Peak 1125 und bis zu 85 % bei Up Peak 2026. Es ist ebenso erkennbar, dass die benörigte Zeit für die Konstruktion des Suchbaums in jeder Instanz zurück geht. Hingegen liegt die Aktualisierungszeit der unteren Schranken im Exact-Replan+WS+fastLB+PB nicht auf allen Instanzen unter der des Exact-Replan. Die Verschlechterungen auf den vereinzelten Instanzen sind jedoch minimal. Der Exact-Replan benötigt für 4 von 18 Instanzen mehr als die simulierte Zeit von einer Stunde. Mithilfe des Exact-Replan+WS+fastLB+PB lassen sich zwei weitere Instanzen in unter einer Stunde lösen, Lunch Peak 2026 und Real Up Peak 2026. Signifikanter ist die Verbesserung der Rechenzeit auf der Instanz Real Up Peak 2026; es ist uns gelungen, die benötigte Zeit um 79.1 % von 3:31 Stunden auf 40 Minuten zu reduzieren.

In Abschnitt 5.3 haben wir festgestellt, dass durch das Pseudo-Branching sehr viele Knoten eingespart werden können. Schauen wir auf die generierten Knoten in Tabelle 5.3, so haben sich diese im Vergleich zum Exact-Replan bedeutend reduziert. Diese Reduzierung weist darauf hin, dass sich die Vermutung über die Ähnlichkeit der Suchbäume bestätigt. Deutlich werden die Veränderungen in den Instanzen Interfloor 2026 und Up Peak 2026. So ist es uns gelungen durchschnittlich $2772,96$ anstatt $10426,06$ bzw. $3564,73$ anstatt $29064,2$ Knoten pro Aufzug und Schnappschussproblem zu erzeugen. Ebenfalls können wir feststellen, dass die Initialisierungszeit für den Warmstart durchschnittlich unter einer Sekunde liegt. Es ist demnach – für einen einzelnen Aufzug und Schnappschussproblem – sehr günstig eine Aktualisierung und Neubewertung durchzuführen.

Bei einer Betrachtung der Laufzeittabelle für Gebäude B können wir die gleichen Effekte feststellen. Auch hier können wir mithilfe des Exact-Replan+WS+fastLB+PB die Rechenzeit in jeder Instanz reduzieren. Im Unterschied zu den Ergebnissen von Gebäude A gelang es uns nicht, eine der Instanzen, welche mehr als eine Stunde Rechenzeit im Exact-Replan benötigt, in unter einer Stunde zu lösen. Nichtsdestotrotz liegen die Einsparungen zwischen 19.5 % und 75.5 %. Eine Analyse von Tabelle 5.5 liefert die gleichen Erkenntnisse wie die Analyse von Tabelle 5.3. Die Anzahl der durchschnittlich generierten Knoten nimmt in jeder Instanz ab und die Initialisierung des Warmstartes ist für jeden Aufzug günstig. In der Summe kann diese Zeit jedoch auf bis zu 57 Minuten anhäufen (Interfloor 4723).

Um die Auswertung abzuschließen, betrachten wir die Tabellen 5.6 und 5.7 für das Gebäude C. In der ersten Tabelle sehen wir wieder die Laufzeiten. Auch hier können wir in jeder Instanz eine Reduzierung der Rechenzeit erzielen. Die geringste Verbesserung erzielen wir in der Instanz Real Down Peak 1040 (29.7 %) die größte Zeitersparnis können wir auf der Instanz Up Peak 1872 verzeichnen (81.7 %). Analog zu den beiden anderen Gebäuden gibt es auch hier Instanzen auf denen die Aktualisierungs- und Neubewertungszeit leicht zunimmt, diese Zuwächse sind jedoch minimal. Betrachten wir die Ergebnisse des Exact-Replan und die vier Instanzen, die nicht in unter einer Stunde gelöst werden können. Von diesen vier Instanzen kann eine mit dem Exact-Replan+WS+fastLB+PB in unter einer Stunde gelöst werden, Lunch Peak 1872. Die Lösungszeit der Instanz Real Up Peak 1872 ist hingegen lediglich 2:10 Minuten über der simulierten Zeit von einer Stunde. Hinsichtlich der generierten Knoten können wir ebenfalls auf allen Instanzen Einsparungen erzielen. Signifikante Beispiele sind wieder die Interfloor und Up Peak Instanzen mit 144 % der Passagierauslastung. Auf diesen beiden Instanzen können wir die erzeugten Knoten von $23364,4$ bzw. $36194,99$ auf $4976,09$ bzw. $5466,91$ reduzieren.

Zusammenfassung

Abschließend können wir sagen, dass die präsentierten Ergebnisse ein sehr positives Fazit zulassen. Auch wenn es uns nicht gelungen ist jede Instanz in unter einer Stunde zu lösen, so konnten wir jedoch alle Laufzeiten um mindestens 19.5 % und bis zu 81.7 % verbessern. Das Lösen der Instanzen in unter einer Stunde Rechenzeit ist zwar ein notwendiges Kriterium für die Echtzeitlauffähigkeit, jedoch kein hinreichendes. Ein wesentlicher Bestandteil des Exact-Replan+WS+fastLB+PB ist die Technik des Pseudo-Branchings. Durch diese wird, wie wir in der obigen Auswertung gesehen haben, eine signifikante Anzahl an Knoten eingespart, was unmittelbar zu einer Reduzierung der Initialisierungszeit des Warmstartes führt. Selbstverständlich ist auch die Ausnutzung der Struktur der unteren Schranke der reduzierten Kosten der Knoten ein wesentlicher Faktor für eine effiziente Reoptimierung.
An dieser Stelle sei noch erwähnt, dass wir davon ausgehen, dass sich die Erkenntnisse dieser Arbeit auch auf allgemeinere Branch & Bound-Verfahren übertragen lassen. Ein allgemeineres Branch & Bound-Verfahren lässt es möglicherweise zu, dass komplette Teilbäume des Suchbaums aufgrund von Unzulässigkeit für alle folgenden Iterationsrunden ausgelotet werden können. In dem von uns betrachteten Problem ist dies nicht der Fall, da der Suchbaum genau so konstruiert wird, dass stets noch zulässige Touren existieren.

Instanz	Exact-Replan			Exact-Replan+WS+fastLB+PB			
	Rechenzeit	Branching	LB	Rechenzeit	Branching	LB	anteilige Rechenzeit
DP 1125	0:01:08	0:00:24	0:00:04	**0:00:44**	**0:00:12**	**0:00:03**	0.642
DP 1407	0:02:14	0:00:59	0:00:09	**0:01:26**	**0:00:29**	**0:00:08**	0.640
DP 2026	0:13:44	0:08:40	**0:01:32**	**0:08:34**	**0:04:04**	0:01:38	0.624
I 1125	0:01:57	0:00:55	**0:00:09**	**0:01:16**	**0:00:25**	**0:00:09**	0.650
I 1407	0:08:35	0:05:19	**0:00:58**	**0:05:28**	**0:02:31**	0:01:03	0.637
I 2026	7:23:41	5:06:55	1:18:14	**3:22:04**	**1:30:39**	**1:06:57**	0.455
LP 1125	0:00:52	0:00:16	**0:00:02**	**0:00:36**	**0:00:07**	**0:00:02**	0.690
LP 1407	0:02:26	0:01:10	**0:00:10**	**0:01:34**	**0:00:32**	**0:00:10**	0.641
LP 2026	1:52:44	1:19:16	0:16:39	**0:47:07**	**0:22:28**	**0:13:27**	0.418
RDP 1125	0:00:55	0:00:15	**0:00:02**	**0:00:38**	**0:00:08**	**0:00:02**	0.689
RDP 1407	0:01:59	0:00:50	**0:00:08**	**0:01:19**	**0:00:26**	**0:00:08**	0.669
RDP 2026	0:12:18	0:07:35	**0:01:25**	**0:07:51**	**0:03:36**	0:01:30	0.639
RUP 1125	0:03:23	0:02:08	0:00:22	**0:01:14**	**0:00:30**	**0:00:09**	0.364
RUP 1407	0:09:45	0:06:35	0:01:10	**0:03:19**	**0:01:36**	**0:00:30**	0.340
RUP 2026	3:31:30	2:33:15	0:30:01	**0:40:01**	**0:21:38**	**0:08:43**	0.189
UP 1125	0:01:45	0:00:57	0:00:09	**0:00:45**	**0:00:15**	**0:00:03**	0.426
UP 1407	0:45:02	0:32:28	0:06:21	**0:11:10**	**0:06:32**	**0:01:58**	0.248
UP 2026	29:36:37	20:59:35	5:29:46	**4:26:44**	**2:31:30**	**1:05:41**	0.150

Tabelle 5.2: Auswertung der Rechenzeiten für das Gebäude A mit dem Exact-Replan und dem modifizierten Exact-Replan+WS+fastLB+PB. Im Vergleich stehen neben der gesamten Rechenzeit auch die für den Verzweigungsprozess (Branching) und für die Aktualisierung der unteren Schranke (LB) benötigten Zeiten. Alle Zeiten sind in dem Format h:mm:ss angegeben. Die besten Werte werden sind in jeder Kategorie hervorgehoben. Zusätzlich ist ein ein Zeitgewinn (< 1), d. h. Reduzierung der Rechenzeit, grün markiert, ein Zeitverlust (≥ 1) ist rot markiert.

Instanz	Exact-Replan				Exact-Replan+WS+fastLB+PB							
	Iterationsrunden		gen. Knoten		Iterationsrunden		gen. Knoten		gesp. Touren		Initialisierung	
	∅	Σ	∅	Σ	∅	Σ	∅	Σ	∅	Σ	∅	Σ
DP 1125	**3.88**	22422	39.57	228649	**3.88**	**22410**	**17.13**	**98957**	5.37	31046	0:00:00	0:00:02
DP 1407	4.87	33096	73.92	502501	**4.84**	**32916**	**32.45**	**220589**	10.53	71583	0:00:00	0:00:05
DP 2026	7.18	66480	391.18	3623926	**7.10**	**65760**	**168.13**	**1557577**	59.53	551452	0:00:00	0:01:04
I 1125	4.88	28674	71.14	418278	**4.81**	**28290**	**30.35**	**178468**	9.99	58725	0:00:00	0:00:05
I 1407	7.28	51306	296.45	2088224	**7.07**	**49830**	**129.78**	**914198**	49.64	349688	0:00:00	0:00:43
I 2026	15.48	145902	10426.06	98276004	**14.84**	**139920**	**2772.96**	**26137964**	1227.76	11572900	0:00:00	0:51:50
LP 1125	**3.63**	**21354**	26.28	154550	**3.63**	21360	**11.33**	**66612**	3.47	20428	0:00:00	0:00:01
LP 1407	4.96	34896	75.47	530705	**4.94**	**34770**	**31.92**	**224464**	10.40	73132	0:00:00	0:00:06
LP 2026	9.94	93516	2774.93	26106505	**9.70**	**91266**	**748.98**	**7046439**	270.38	2543689	0:00:00	0:09:58
RDP 1125	**3.58**	**21054**	27.77	163105	3.60	21138	**12.06**	**70848**	3.62	21267	0:00:00	0:00:01
RDP 1407	**4.54**	**31914**	61.80	434206	**4.54**	**31890**	**27.75**	**194941**	9.17	64413	0:00:00	0:00:04
RDP 2026	7.13	66930	337.59	3169995	**7.04**	**66078**	**146.38**	**1374500**	55.19	518192	0:00:00	0:01:01
RUP 1125	**3.73**	21594	125.26	724518	**3.73**	**21552**	**30.67**	**177410**	8.86	51255	0:00:00	0:00:06
RUP 1407	5.16	36072	308.59	2155196	**5.04**	**35232**	**73.65**	**514378**	21.91	153045	0:00:00	0:00:19
RUP 2026	9.43	96990	3914.48	40256544	**8.95**	**92058**	**553.04**	**5687509**	178.08	1831355	0:00:00	0:06:32
UP 1125	3.32	19224	65.54	379862	**3.29**	**19062**	**16.91**	**98028**	4.37	25315	0:00:00	0:00:02
UP 1407	4.92	34242	1305.92	9081366	**4.76**	**33114**	**259.06**	**1801496**	80.46	559487	0:00:00	0:01:10
UP 2026	9.45	93876	29064.20	288781869	**9.18**	**91212**	**3564.73**	**35419129**	945.78	9397270	0:00:00	0:44:12

Tabelle 5.3: Gegenüberstellung der beiden Algorithmen Exact-Replan und Exact-Replan+WS+fastLB+PB für das Gebäude A. Dabei werden die benötigten Iterationsrunden und generierten Knoten verglichen. Jeweils im Mittel und einmal summiert über alle Aufzüge und Schnappschüsse. Desweiteren sind die zwischengespeicherten Touren und die für die Initialisierung des Warmstarts benötigte Zeit angegeben – ebenfalls gemittelt und summiert.

Instanz	Exact-Replan			Exact-Replan+WS+fastLB+PB			
	Rechenzeit	Branching	LB	Rechenzeit	Branching	LB	anteilige Rechenzeit
DP 2624	0:01:45	0:00:21	0:00:05	**0:01:07**	**0:00:12**	**0:00:03**	0.641
DP 3280	0:02:27	0:00:38	0:00:07	**0:01:32**	**0:00:20**	**0:00:06**	0.626
DP 4723	0:04:53	0:01:46	0:00:20	**0:03:01**	**0:00:54**	**0:00:16**	0.618
I 2624	0:06:43	0:03:15	**0:00:35**	**0:04:46**	**0:01:47**	0:00:43	0.710
I 3280	0:17:58	0:10:08	**0:01:50**	**0:14:47**	**0:06:25**	0:03:04	0.823
I 4723	7:21:47	5:08:27	**1:06:07**	**4:30:25**	**2:06:38**	1:29:07	0.612
LP 2624	0:03:37	0:01:23	**0:00:13**	**0:02:27**	**0:00:46**	0:00:13	0.680
LP 3280	0:18:07	0:10:18	0:01:51	**0:14:35**	**0:06:24**	**0:03:03**	0.805
LP 4723	0:40:17	0:25:37	**0:04:33**	**0:33:11**	**0:16:24**	0:07:21	0.824
RDP 2624	0:01:56	0:00:27	0:00:05	**0:01:14**	**0:00:14**	**0:00:04**	0.642
RDP 3280	0:03:15	0:00:59	0:00:11	**0:02:05**	**0:00:31**	**0:00:09**	0.640
RDP 4723	0:08:19	0:03:47	0:00:41	**0:05:17**	**0:01:54**	**0:00:40**	0.637
RUP 2624	0:03:31	0:01:37	0:00:16	**0:01:50**	**0:00:32**	**0:00:08**	0.522
RUP 3280	0:07:27	0:04:07	0:00:43	**0:03:19**	**0:01:14**	**0:00:21**	0.445
RUP 4723	10:21:49	7:15:49	1:19:58	**2:55:39**	**1:32:35**	**0:33:36**	0.282
UP 2624	0:03:21	0:01:40	0:00:17	**0:01:39**	**0:00:31**	**0:00:07**	0.495
UP 3280	0:07:07	0:04:10	0:00:42	**0:02:51**	**0:01:06**	**0:00:15**	0.402
UP 4723	7:44:17	5:39:59	0:51:18	**1:53:54**	**1:08:09**	**0:13:41**	0.245

Tabelle 5.4: Auswertung der Rechenzeiten für das Gebäude B mit dem Exact-Replan und dem modifizierten Exact-Replan+WS+fastLB+PB. Im Vergleich stehen neben der gesamten Rechenzeit auch die für den Verzweigungsprozess (Branching) und für die Aktualisierung der unteren Schranke (LB) benötigten Zeiten. Alle Zeiten sind in dem Format h:mm:ss angegeben. Die besten Werte werden sind in jeder Kategorie hervorgehoben. Zusätzlich ist ein ein Zeitgewinn (< 1), d. h. Reduzierung der Rechenzeit, grün markiert, ein Zeitverlust (≥ 1) ist rot markiert.

Instanz	Exact-Replan				Exact-Replan+WS+fastLB+PB							
	Iterationsrunden		gen. Knoten		Iterationsrunden		gen. Knoten		gesp. Touren		Initialisierung	
	∅	Σ	∅	Σ	∅	Σ	∅	Σ	∅	Σ	∅	Σ
DP 2624	**3.38**	**49472**	19.11	279743	**3.38**	49528	**8.36**	**122346**	2.28	33395	0:00:00	0:00:01
DP 3280	**3.70**	**63024**	25.27	430789	**3.7**	63072	**10.97**	**187027**	3.19	54426	0:00:00	0:00:03
DP 4723	4.47	102560	42.22	969765	**4.44**	**102080**	**18.35**	**421380**	5.63	129366	0:00:00	0:00:08
I 2624	6.14	91760	100.27	1498438	**5.99**	**89448**	**48.58**	**726023**	16.08	240374	0:00:00	0:00:23
I 3280	8.14	142032	246.01	4294423	**7.89**	**137784**	**136.23**	**2377981**	46.14	805359	0:00:00	0:01:46
I 4723	14.71	323784	4637.07	1020896	**13.76**	**302912**	**1681.62**	**37022619**	571.38	12579599	0:00:00	0:57:12
LP 2624	4.72	69680	48.40	713924	**4.68**	**69032**	**23.09**	**340569**	7.77	114577	0:00:00	0:00:07
LP 3280	8.05	140664	247.52	4322721	**7.79**	**136000**	**133.77**	**2336153**	45.48	794289	0:00:00	0:01:45
LP 4723	8.74	193856	441.79	9797134	**8.39**	**186088**	**249.33**	**5529036**	81.51	1807648	0:00:00	0:04:27
RDP 2624	**3.59**	**53624**	21.29	318180	3.61	53920	**9.16**	**136908**	2.67	39919	0:00:00	0:00:02
RDP 3280	**4.31**	**75312**	34.22	597329	4.32	75400	**14.77**	**257740**	4.66	81432	0:00:00	0:00:05
RDP 4723	5.69	127840	81.17	1824069	**5.61**	**126128**	**35.93**	**807444**	11.49	258266	0:00:00	0:00:20
RUP 2624	3.82	57912	46.19	699567	**3.8**	**57480**	**14.86**	**224998**	4.07	61634	0:00:00	0:00:04
RUP 3280	4.27	76040	89.98	1602391	**4.25**	**75632**	**26.14**	**465413**	7.63	135853	0:00:00	0:00:11
RUP 4723	9.24	861592	1218.45	113637742	**9.0**	**838968**	**257.86**	**24048890**	79.26	7391931	0:00:00	0:19:40
UP 2624	3.04	44504	46.68	683042	**3.02**	**44240**	**14.28**	**208875**	3.69	54017	0:00:00	0:00:03
UP 3280	3.60	65144	81.75	1480686	**3.57**	**64608**	**22.01**	**398563**	6.02	109093	0:00:00	0:00:08
UP 4723	6.25	663120	614.23	65196922	**6.13**	**650680**	**121.03**	**12847125**	37.48	3978010	0:00:00	0:07:47

Tabelle 5.5: Gegenüberstellung der beiden Algorithmen Exact-Replan und Exact-Replan+WS+fastLB+PB für das Gebäude B. Dabei werden die benötigten Iterationsrunden und generierten Knoten verglichen. Jeweils im Mittel und einmal summiert über alle Aufzüge und Schnappschüsse. Desweiteren sind die zwischengespeicherten Touren und die für die Initialisierung des Warmstarts benötigte Zeit angegeben – ebenfalls gemittelt und summiert.

Instanz	Exact-Replan			Exact-Replan+WS+fastLB+PB			
	Rechenzeit	Branching	LB	Rechenzeit	Branching	LB	anteilige Rechenzeit
DP 1040	0:01:12	0:00:30	0:00:05	**0:00:50**	**0:00:16**	**0:00:04**	0.697
DP 1300	0:03:00	0:01:36	**0:00:15**	**0:02:01**	**0:00:47**	0:00:16	0.673
DP 1872	0:23:19	0:15:50	0:02:46	**0:13:30**	**0:06:37**	**0:02:44**	0.579
I 1040	0:02:04	0:01:03	**0:00:09**	**0:01:22**	**0:00:30**	0:00:10	0.659
I 1300	0:13:50	0:09:14	**0:01:40**	**0:08:04**	**0:03:51**	0:01:42	0.583
I 1872	13:18:28	9:14:45	2:20:11	**4:58:13**	**2:09:38**	**1:38:03**	0.373
LP 1040	0:00:59	0:00:23	**0:00:03**	**0:00:39**	**0:00:10**	**0:00:03**	0.670
LP 1300	0:04:16	0:02:30	**0:00:24**	**0:02:37**	**0:01:06**	**0:00:24**	0.614
LP 1872	2:07:14	1:30:31	0:17:56	**0:52:26**	**0:25:35**	**0:13:43**	0.412
RDP 1040	0:00:48	0:00:14	0:00:02	**0:00:33**	**0:00:08**	**0:00:01**	0.703
RDP 1300	0:01:55	0:00:54	**0:00:08**	**0:01:18**	**0:00:28**	**0:00:08**	0.679
RDP 1872	0:20:43	0:13:49	**0:02:23**	**0:12:37**	**0:06:04**	0:02:31	0.609
RUP 1040	0:02:22	0:01:24	0:00:14	**0:00:56**	**0:00:20**	**0:00:05**	0.397
RUP 1300	0:14:16	0:09:59	0:01:46	**0:03:57**	**0:01:55**	**0:00:41**	0.277
RUP 1872	5:35:54	4:02:54	0:48:32	**1:02:10**	**0:32:00**	**0:14:53**	0.185
UP 1040	0:27:33	0:19:49	0:04:03	**0:05:57**	**0:03:14**	**0:01:08**	0.216
UP 1300	0:36:17	0:26:02	0:05:03	**0:08:30**	**0:04:47**	**0:01:29**	0.235
UP 1872	21:55:22	15:59:02	3:18:59	**4:00:37**	**2:23:19**	**0:48:38**	0.183

Tabelle 5.6: Auswertung der Rechenzeiten für das Gebäude C mit dem Exact-Replan und dem modifizierten Exact-Replan+WS+fastLB+PB. Im Vergleich stehen neben der gesamten Rechenzeit auch die für den Verzweigungsprozess (Branching) und für die Aktualisierung der unteren Schranke (LB) benötigten Zeiten. Alle Zeiten sind in dem Format h:mm:ss angegeben. Die besten Werte werden sind in jeder Kategorie hervorgehoben. Zusätzlich ist ein ein Zeitgewinn (< 1), d. h. Reduzierung der Rechenzeit, grün markiert, ein Zeitverlust (≥ 1) ist rot markiert.

Instanz	Exact-Replan				Exact-Replan+WS+fastLB+PB							
	Iterationsrunden		gen. Knoten		Iterationsrunden		gen. Knoten		gesp. Touren		Initialisierung	
	∅	Σ	∅	Σ	∅	Σ	∅	Σ	∅	Σ	∅	Σ
DP 1040	**4.26**	**18985**	59.17	263590	4.29	19095	**26.98**	**120186**	8.96	39910	0:00:00	0:00:02
DP 1300	**5.3**	**28620**	136.52	735169	5.32	28635	**60.12**	**323762**	21.02	113215	0:00:00	0:00:10
DP 1872	8.38	60730	855.38	6197206	**8.28**	**59980**	**333.89**	**2419011**	131.23	950740	0:00:00	0:02:07
I 1040	5.62	25195	104.09	466330	**5.56**	**24930**	**45.08**	**201937**	14.79	66238	0:00:00	0:00:06
I 1300	8.48	47030	625.44	3468091	**8.27**	**45830**	**243.94**	**1352675**	92.52	513034	0:00:00	0:01:18
I 1872	18.83	138985	23364.40	172429274	**18.03**	**133065**	**4976.09**	**36723532**	2508.98	18516296	0:00:00	1:29:37
LP 1040	**4.06**	**18305**	42.81	193081	4.08	18385	**17.94**	**80929**	5.60	25240	0:00:00	0:00:02
LP 1300	6.14	33300	189.83	1029829	**6.01**	**32600**	**76.14**	**413069**	26.86	145734	0:00:00	0:00:18
LP 1872	11.90	88205	4007.18	29693217	**11.50**	**85235**	**1072.42**	**7946631**	412.69	3058049	0:00:00	0:11:33
RDP 1040	**3.70**	16565	32.81	146984	**3.70**	**16555**	**14.69**	**65819**	4.73	21180	0:00:00	0:00:01
RDP 1300	4.92	27320	79.30	440128	**4.90**	**27205**	**37.42**	**207706**	12.65	70225	0:00:00	0:00:05
RDP 1872	8.38	61840	726.55	5361931	**8.29**	**61195**	**299.96**	**2213710**	125.25	924371	0:00:00	0:01:58
RUP 1040	4.07	18340	118.08	532535	**4.04**	**18200**	**28.62**	**129058**	8.13	36687	0:00:00	0:00:04
RUP 1300	6.01	32675	590.21	3207806	**5.88**	**31935**	**113.48**	**616756**	35.73	194170	0:00:00	0:00:32
RUP 1872	10.32	81580	8135.78	64313330	**9.83**	**77725**	**1076.63**	**8510726**	352.99	2790382	0:00:00	0:13:38
UP 1040	4.36	19520	1182.00	5289461	**4.32**	**19335**	**195.35**	**874190**	63.06	282174	0:00:00	0:00:56
UP 1300	5.71	31075	1390.10	7562155	**5.55**	**30215**	**253.70**	**1380109**	76.73	417402	0:00:00	0:01:02
UP 1872	9.35	53150	36194.99	205768520	**9.05**	**51425**	**5466.91**	**31079362**	1743.75	9913225	0:00:00	0:34:55

Tabelle 5.7: Gegenüberstellung der beiden Algorithmen Exact-Replan und Exact-Replan+WS+fastLB+PB für das Gebäude C. Dabei werden die benötigten Iterationsrunden und generierten Knoten verglichen. Jeweils im Mittel und einmal summiert über alle Aufzüge und Schnappschüsse. Des Weiteren sind die zwischengespeicherten Touren und die für die Initialisierung des Warmstarts benötigte Zeit angegeben – ebenfalls gemittelt und summiert.

Anhang A

Testgebäude und Szenarien

In der Einleitung wurde bereits erwähnt, dass der Exact-Replan Algorithmus im Rahmen eines Forschungsprojektes und der Dissertation [Hil10] entwickelt wurde. Die in dieser Arbeit getesteten Instanzen wurden von dem entsprechenden Industriepartner bereitgestellt und mit dem Programm ELEVATE [Ltd] generiert.

Als Grundlage stehen drei Gebäude mit je 18 Szenarien zur Verfügung. Die Testinstanzen unterteilen sich in die Verkehrsmuster Down Peak (DP), Real Down Peak (RDP), Interfloor (I), Lunch Peak (LP), Real Up Peak (RUP) und Up Peak (UP). Jedes dieser sechs Verkehrsmuster wurde mit 80 %, 100 % und 144 % der Passagierauslastung simuliert.

	Gebäude A	**Gebäude B**	**Gebäude C**
Etagen	23	12	25
Aufzüge	6	8	5
80 % Passagiere/h	1125	2624	1040
100 % Passagiere/h	1407	3280	1300
144 % Passagiere/h	2026	4723	1872

Tabelle A.1: Eigenschaften der Gebäude A,B und C.

In der Tabelle A.3 ist die Anzahl der aufgenommenen Schnappschussprobleme in den einzelnen Szenarien der Gebäude A,B und C zu sehen. Die Schnappschussprobleme sowie deren Anzahl sind unter Verwendung der in dieser Arbeit vorgestellten Algorithmen identisch.

In Abschnitt 2.1 haben wir die verschiedenen Verkehrsmustern innerhalb eines Bürogebäudes vorgestellt. Die Verkehrsmuster Up Peak, Down Peak und Interfloor haben wir als die Grundformen beschrieben. Real Up Peak, Real Down Peak und Lunch Peak hingegen bezeichneten wir als Mischformen, die genaue Zusammensetzung ist in der Tabelle A.2zu sehen.

	Real Up Peak	**Real Down Peak**	**Lunch Peak**
Up Peak	90 %	5 %	40 %
Interfloor	5 %	5 %	20 %
Down Peak	5 %	90 %	40 %

Tabelle A.2: Zusammensetzung der Mischformen Real Up Peak, Real Down Peak und Lunch Peak aus den Grundformen der Verkehrsmuster Up Peak, Interfloor und Down Peak.

Instanz	**Gebäude A**	**Gebäude B**	**Gebäude C**
Down Peak 80 %	963	1830	891
Down Peak 100 %	1133	2131	1077
Down Peak 144 %	1544	2871	1449
Interloor 80 %	980	1868	896
Interloor 100 %	1174	2182	1109
Interloor 144 %	1571	2752	1476
Lunch Peak 80 %	980	1844	902
Lunch Peak 100 %	1172	2183	1085
Lunch Peak 144 %	1568	2772	1482
Real Down Peak 80 %	979	1868	896
Real Down Peak 100 %	1171	2182	1110
Real Down Peak 144 %	1565	2809	1476
Real Up Peak 80 %	964	1893	902
Real Up Peak 100 %	1164	2226	1087
Real Up Peak 144 %	1714	11658	1581
Up Peak 80 %	966	1829	895
Up Peak 100 %	1159	2264	1088
Up Peak 144 %	1656	13268	1137

Tabelle A.3: Übersicht über die Anzahl der Schnappschussprobleme in den verschiedenen Instanzen auf den drei Gebäuden.

Anhang B

Tabellen und Diagramme

B.1 Ähnlichkeit von Branch & Bound-Bäumen

Dieser Abschnitt befasst sich mit den Auswertungen des Ähnlichkeitmaßes für Branch & Bound-Bäume, welches in Abschnitt 3.3.1 eingeführt wurde. Dabei betrachten wir die Ähnlichkeit Λ der Suchbäume T_i und T_j der i-ten und j-ten Iterationsrunde eines Schnappschussproblems. In den folgenden Tabellen sind die Werte über alle Aufzüge gemittelt. Für eine bessere Übersicht sind in den Tabellen die Werte anhand ihrer Ähnlichkeit eingefärbt. Hellgrün bedeutet eine Ähnlichkeit nahe 0, eine Einfärbung in dunkelblau spiegelt eine Ähnlichkeit nahe 1 wieder. Werte im Intervall $[0, 1]$ gehen schrittweise von hellgrün in Dunkelblau über.

Zu jedem Testgebäude und jedem Verkehrsmuster ist eine Tabelle über die Ähnlichkeit der Suchbäume eines Schnappschussproblems angegeben. In den nachfolgenden Tabellen ist zu sehen, dass die Ähnlichkeit – bis auf vereinzelte Schwankungen – stets zunimmt. Ebenfalls auffällig ist, dass in (nahezu) jeder Instanz die letzten Suchbäume eine Ähnlichkeit von über 80 % haben. Die hier gewonnenen Erkenntnisse bestätigen die in Abschnitt 3.3.1 formulierte Vermutung und rechtfertigt somit einen Warmstart des Exact-Replan unter Verwendung der Blätter des Suchbaumes der vorherigen Iterationsrunde.

Gebäude A

T_i	1	2	3	4	5
1	1.000	0.2774	0.2340	0.2057	0.2057
2	0.2774	1.000	0.3126	0.3304	0.3304
3	0.2340	0.3126	1.000	0.6675	0.6675
4	0.2057	0.3304	0.6675	1.000	1.0000
5	0.2057	0.3304	0.6675	1.0000	1.000

Tabelle B.1: Gebäude A, Down Peak

T_i	1	2	3	4	5	6
1	1.000	0.3170	0.3231	0.2878	0.2990	0.2990
2	0.3170	1.000	0.2995	0.2854	0.2874	0.2874
3	0.3231	0.2995	1.000	0.6710	0.6301	0.6385
4	0.2878	0.2854	0.6710	1.000	0.8576	0.8576
5	0.2990	0.2874	0.6301	0.8576	1.000	0.9792
6	0.2990	0.2874	0.6385	0.8576	0.9792	1.000

Tabelle B.2: Gebäude A, Real Up Peak

T_i	1	2	3	4	5	6	7	8	9
1	1.000	0.0273	0.0563	0.0130	0.0468	0.0492	0.0465	0.0489	0.0490
2	0.0273	1.000	0.0182	0.0117	0.0446	0.0358	0.0318	0.0342	0.0322
3	0.0563	0.0182	1.000	0.0230	0.0399	0.0748	0.0715	0.0720	0.0742
4	0.0130	0.0117	0.0230	1.000	0.0944	0.0423	0.0392	0.0380	0.0370
5	0.0468	0.0446	0.0399	0.0944	1.000	0.1926	0.1960	0.2121	0.1961
6	0.0492	0.0358	0.0748	0.0423	0.1926	1.000	0.6370	0.5670	0.6128
7	0.0465	0.0318	0.0715	0.0392	0.1960	0.6370	1.000	0.8225	0.8992
8	0.0489	0.0342	0.0720	0.0380	0.2121	0.5670	0.8225	1.000	0.9036
9	0.0490	0.0322	0.0742	0.0370	0.1961	0.6128	0.8992	0.9036	1.000

Tabelle B.3: Gebäude A, Interfloor

T_i	1	2	3	4	5	6	7	8	9	10	11
1	1.000	0.0155	0.0579	0.0689	0.0667	0.0495	0.0551	0.0529	0.0540	0.0579	0.0559
2	0.0155	1.000	0.0370	0.0259	0.0227	0.0227	0.0230	0.0198	0.0217	0.0221	0.0213
3	0.0579	0.0370	1.000	0.0958	0.1301	0.1160	0.1040	0.0977	0.1077	0.1153	0.1132
4	0.0689	0.0259	0.0958	1.000	0.2029	0.1993	0.2199	0.1984	0.1927	0.1893	0.1887
5	0.0667	0.0227	0.1301	0.2029	1.000	0.3460	0.5474	0.5250	0.5493	0.5442	0.5463
6	0.0495	0.0227	0.1160	0.1993	0.3460	1.000	0.3614	0.2875	0.3249	0.3125	0.3119
7	0.0551	0.0230	0.1040	0.2199	0.5474	0.3614	1.000	0.6268	0.6506	0.6057	0.6021
8	0.0529	0.0198	0.0977	0.1984	0.5250	0.2875	0.6268	1.000	0.7672	0.7449	0.7392
9	0.0540	0.0217	0.1077	0.1927	0.5493	0.3249	0.6506	0.7672	1.000	0.8876	0.8711
10	0.0579	0.0221	0.1153	0.1893	0.5442	0.3125	0.6057	0.7449	0.8876	1.000	0.9746
11	0.0559	0.0213	0.1132	0.1887	0.5463	0.3119	0.6021	0.7392	0.8711	0.9746	1.000

Tabelle B.4: Gebäude A, Lunch Peak

Gebäude B

T_i	1	2	3	4	5	6	7
1	1.000	0.1051	0.0697	0.0699	0.0506	0.0508	0.0512
2	0.1051	1.000	0.1032	0.1006	0.0599	0.0600	0.0602
3	0.0697	0.1032	1.000	0.4377	0.3729	0.3635	0.3730
4	0.0699	0.1006	0.4377	1.000	0.4722	0.4778	0.4622
5	0.0506	0.0599	0.3729	0.4722	1.000	0.9276	0.9736
6	0.0508	0.0600	0.3635	0.4778	0.9276	1.000	0.9510
7	0.0512	0.0602	0.3730	0.4622	0.9736	0.9510	1.000

Tabelle B.5: Gebäude B, Down Peak

T_i	1	2	3	4
1	1.000	0.3206	0.2516	0.2447
2	0.3206	1.000	0.2544	0.2480
3	0.2516	0.2544	1.000	0.9417
4	0.2447	0.2480	0.9417	1.000

Tabelle B.6: Gebäude B, Real Down Peak

T_i	1	2	3	4	5	6
1	1.000	0.0611	0.0036	0.1421	0.1378	0.1418
2	0.0611	1.000	0.0220	0.0531	0.0530	0.0526
3	0.0036	0.0220	1.000	0.0050	0.0052	0.0053
4	0.1421	0.0531	0.0050	1.000	0.9173	0.8969
5	0.1378	0.0530	0.0052	0.9173	1.000	0.9337
6	0.1418	0.0526	0.0053	0.8969	0.9337	1.000

Tabelle B.7: Gebäude B, Up Peak

T_i	1	2	3	4
1	1.000	0.4590	0.4696	0.5557
2	0.4590	1.000	0.9281	0.7586
3	0.4696	0.9281	1.000	0.7797
4	0.5557	0.7586	0.7797	1.000

Tabelle B.8: Gebäude B, Real Up Peak

T_i	1	2	3	4
1	1.000	0.2029	0.2092	0.2040
2	0.2029	1.000	0.5382	0.5499
3	0.2092	0.5382	1.000	0.9479
4	0.2040	0.5499	0.9479	1.000

Tabelle B.9: Gebäude B, Interfloor

T_i	1	2	3	4	5	6
1	1.000	0.1734	0.1724	0.1474	0.1490	0.1448
2	0.1734	1.000	0.8577	0.7918	0.7813	0.7689
3	0.1724	0.8577	1.000	0.8625	0.8514	0.8414
4	0.1474	0.7918	0.8625	1.000	0.9844	0.9750
5	0.1490	0.7813	0.8514	0.9844	1.000	0.9594
6	0.1448	0.7689	0.8414	0.9750	0.9594	1.000

Tabelle B.10: Gebäude B, Lunch Peak

Gebäude C

T_i	1	2	3	4	5	6	7	8
1	1.000	0.1745	0.1519	0.0948	0.1689	0.1555	0.1637	0.1637
2	0.1745	1.000	0.3035	0.1116	0.2103	0.1835	0.1826	0.1826
3	0.1519	0.3035	1.000	0.1251	0.1740	0.1497	0.1506	0.1506
4	0.0948	0.1116	0.1251	1.000	0.1791	0.1305	0.1272	0.1272
5	0.1689	0.2103	0.1740	0.1791	1.000	0.5616	0.5811	0.5811
6	0.1555	0.1835	0.1497	0.1305	0.5616	1.000	0.6986	0.6986
7	0.1637	0.1826	0.1506	0.1272	0.5811	0.6986	1.000	1.0000
8	0.1637	0.1826	0.1506	0.1272	0.5811	0.6986	1.0000	1.000

Tabelle B.11: Gebäude C, Down Peak

T_i	1	2	3	4	5	6	7
1	1.000	0.2687	0.0713	0.1788	0.1757	0.1677	0.1686
2	0.2687	1.000	0.0746	0.2377	0.1726	0.1795	0.1687
3	0.0713	0.0746	1.000	0.1257	0.0557	0.0608	0.0556
4	0.1788	0.2377	0.1257	1.000	0.2609	0.2627	0.2615
5	0.1757	0.1726	0.0557	0.2609	1.000	0.8334	0.8506
6	0.1677	0.1795	0.0608	0.2627	0.8334	1.000	0.9255
7	0.1686	0.1687	0.0556	0.2615	0.8506	0.9255	1.000

Tabelle B.12: Gebäude C, Real Down Peak

T_i	1	2	3	4	5	6	7	8	9
1	1.000	0.2700	0.3254	0.2693	0.2693	0.2676	0.2575	0.2586	0.2598
2	0.2700	1.000	0.3027	0.2703	0.2703	0.2686	0.2581	0.2592	0.2604
3	0.3254	0.3027	1.000	0.7103	0.7103	0.7008	0.6598	0.6639	0.6683
4	0.2693	0.2703	0.7103	1.000	1.0000	0.9810	0.9051	0.9124	0.9204
5	0.2693	0.2703	0.7103	1.0000	1.000	0.9810	0.9051	0.9124	0.9204
6	0.2676	0.2686	0.7008	0.9810	0.9810	1.000	0.9233	0.9305	0.9386
7	0.2575	0.2581	0.6598	0.9051	0.9051	0.9233	1.000	0.9902	0.9805
8	0.2586	0.2592	0.6639	0.9124	0.9124	0.9305	0.9902	1.000	0.9897
9	0.2598	0.2604	0.6683	0.9204	0.9204	0.9386	0.9805	0.9897	1.000

Tabelle B.13: Gebäude C, Up Peak

T_i	1	2	3	4	5	6	7
1	1.000	0.3826	0.3242	0.2057	0.1973	0.1879	0.1920
2	0.3826	1.000	0.3586	0.2694	0.2608	0.2510	0.2560
3	0.3242	0.3586	1.000	0.4123	0.4000	0.3955	0.3931
4	0.2057	0.2694	0.4123	1.000	0.9430	0.8887	0.9082
5	0.1973	0.2608	0.4000	0.9430	1.000	0.9387	0.9623
6	0.1879	0.2510	0.3955	0.8887	0.9387	1.000	0.9277
7	0.1920	0.2560	0.3931	0.9082	0.9623	0.9277	1.000

Tabelle B.14: Gebäude C, Real Up Peak

T_i	1	2	3	4	5	6	7	8	9
1	1.000	0.0388	0.0378	0.0453	0.0312	0.0494	0.0504	0.0454	0.0461
2	0.0388	1.000	0.0462	0.0448	0.0330	0.0369	0.0363	0.0329	0.0339
3	0.0378	0.0462	1.000	0.1406	0.1966	0.1386	0.1470	0.1500	0.1503
4	0.0453	0.0448	0.1406	1.000	0.1999	0.2879	0.2910	0.2793	0.3004
5	0.0312	0.0330	0.1966	0.1999	1.000	0.3872	0.4049	0.4369	0.4109
6	0.0494	0.0369	0.1386	0.2879	0.3872	1.000	0.8049	0.8391	0.8524
7	0.0504	0.0363	0.1470	0.2910	0.4049	0.8049	1.000	0.8714	0.8901
8	0.0454	0.0329	0.1500	0.2793	0.4369	0.8391	0.8714	1.000	0.8931
9	0.0461	0.0339	0.1503	0.3004	0.4109	0.8524	0.8901	0.8931	1.000

Tabelle B.15: Gebäude C, Interfloor

T_i	1	2	3	4	5	6	7
1	1.000	0.0981	0.2076	0.1388	0.1531	0.1347	0.1243
2	0.0981	1.000	0.1176	0.0945	0.1061	0.0914	0.0833
3	0.2076	0.1176	1.000	0.4030	0.4373	0.3319	0.3148
4	0.1388	0.0945	0.4030	1.000	0.5382	0.5089	0.4765
5	0.1531	0.1061	0.4373	0.5382	1.000	0.6684	0.6053
6	0.1347	0.0914	0.3319	0.5089	0.6684	1.000	0.9250
7	0.1243	0.0833	0.3148	0.4765	0.6053	0.9250	1.000

Tabelle B.16: Gebäude C, Lunch Peak

B.2 Exact-Replan vs. Exact-Replan+WS

Rechenzeiten

In den folgenden Tabellen und Diagrammen wollen wir den Exact-Replan und Exact-Replan+WS vergleichen. Die Rechenzeiten der Gebäude B und C liefern das gleiche Ergebnis wie auf Gebäude A. Differenzen in den Rechenzeiten von einer Sekunden können auf Messungenauigkeiten zurückzuführen sein, sodass wir davon ausgehen, dass beide Algorithmen auf den entsprechenden Instanzen gleich schnell sind. Der Exact-Replan+WS ist auf den Instanzen mit 80 % und 100 % Passagiere etwas schneller als der Exact-Replan. Auf den Instanzen mit 144 % hingegen kommt es besonders auf den Interfloor Instanzen zu einem erheblichen Zeitverlust. Grund dafür sind wieder die gestiegenen Zeiten, welchen für die Aktualisierung der unteren Schranke benötigt werden. Ausnahmen bilden auch auf diesen Gebäuden die (Real) Up Peak Instanzen.

Instanz	Exact-Replan		Exact-Replan+WS		
	Gesamt	LB	Gesamt	LB	anteilige Rechenzeit
DP 2624	0:01:45	0:00:05	**0:01:31**	0:00:12	0.867
DP 3280	0:02:27	0:00:07	**0:02:09**	0:00:22	0.878
DP 4723	0:04:53	0:00:20	**0:04:36**	0:01:00	0.942
IF 2624	**0:06:43**	0:00:35	0:10:33	0:04:32	1.571
IF 3280	**0:17:58**	0:01:50	0:46:58	0:25:27	2.614
IF 4723	**7:21:47**	1:06:07	27:08:45	20:05:43	3.687
LP 2624	**0:03:37**	0:00:13	**0:03:36**	0:00:52	0.995
LP 3280	**0:18:07**	0:01:51	0:47:40	0:26:20	2.631
LP 4723	**0:40:17**	0:04:33	1:39:00	0:54:17	2.458
RDP 2624	0:01:56	0:00:05	**0:01:42**	0:00:15	0.879
RDP 3280	0:03:15	0:00:11	**0:03:02**	0:00:37	0.933
RDP 4723	**0:08:19**	0:00:41	0:09:37	0:03:08	1.156
RUP 2624	0:03:31	0:00:16	**0:02:17**	0:00:25	0.649
RUP 3280	0:07:27	0:00:43	**0:04:23**	0:01:05	0.588
RUP 4723	10:21:49	1:19:58	**5:35:06**	2:44:14	0.539
UP 2624	0:03:21	0:00:17	**0:01:58**	0:00:19	0.587
UP 3280	0:07:07	0:00:42	**0:03:30**	0:00:43	0.492
UP 4723	7:44:17	0:51:18	**2:28:34**	0:44:38	0.320

Tabelle B.17: Laufzeitvergleich zwischen dem Exact-Replan und der Grundversion des warmstartenden Algorithmus (Exact-Replan+WS) auf den Instanzen des Gebäude B. Die Zeiten sind in dem Format hh:mm:ss angegeben. Unter *Gesamt* ist die benötigte Rechenzeit aller Schnappschussprobleme aufaddiert. *LB* ist die benötigte Zeit für die Aktualisierung der unteren Schranke der reduzierten Kosten der zwischengespeicherten Knoten und Touren. Unter *anteilige Rechenzeit* ist der Quotient (Rechenzeit Exact-Replan+WS)/(Rechenzeit Exact-Replan) zu verstehen.

Instanz	Exact-Replan		Exact-Replan+WS		
	Gesamt	LB	Gesamt	LB	anteilige Rechenzeit
DP 1040	**0:01:12**	0:00:05	0:01:09	0:00:15	0.958
DP 1300	**0:03:00**	0:00:15	0:03:40	0:01:13	1.222
DP 1872	**0:23:19**	0:02:46	0:33:26	0:16:38	1.434
IF 1040	**0:02:04**	0:00:09	0:02:20	0:00:48	1.129
IF 1300	**0:13:50**	0:01:40	0:23:15	0:12:13	1.681
IF 1872	**13:18:28**	2:20:11	33:23:53	24:26:30	2.511
LP 1040	0:00:59	0:00:03	**0:00:53**	0:00:11	0.898
LP 1300	**0:04:16**	0:00:24	0:05:01	0:02:02	1.176
LP 1872	**2:07:14**	0:17:56	2:53:54	1:49:09	1.367
RDP 1040	**0:00:48**	0:00:02	0:00:41	0:00:05	0.854
RDP 1300	**0:01:55**	0:00:08	0:01:57	0:00:30	1.017
RDP 1872	**0:20:43**	0:02:23	0:31:06	0:15:03	1.501
RUP 1040	0:02:22	0:00:14	**0:01:16**	0:00:19	0.535
RUP 1300	0:14:16	0:01:46	**0:06:17**	0:02:39	0.440
RUP 1872	5:35:54	0:48:32	**1:51:10**	1:01:30	0.331
UP 1040	0:27:33	0:04:03	**0:09:03**	0:03:47	0.328
UP 1300	0:36:17	0:05:03	**0:12:51**	0:04:59	0.354
UP 1872	21:55:22	3:18:59	**6:13:42**	2:41:56	0.284

Tabelle B.18: Laufzeitvergleich zwischen dem Exact-Replan und der Grundversion des warmstartenden Algorithmus (Exact-Replan+WS) auf den Instanzen des Gebäude C. Die Zeiten sind in dem Format hh:mm:ss angegeben. Unter *Gesamt* ist die benötigte Rechenzeit aller Schnappschussprobleme aufaddiert. *LB* ist die benötigte Zeit für die Aktualisierung der unteren Schranke der reduzierten Kosten der zwischengespeicherten Knoten und Touren. Unter *anteilige Rechenzeit* ist der Quotient (Rechenzeit Exact-Replan+WS)/(Rechenzeit Exact-Replan) zu verstehen.

Generierte Knoten in den Schnappschussproblemen

In den folgenden Abbildungen ist die Anzahl der erzeugten Knoten, welche für die Lösung eines Schnappschussproblems mit dem Exact-Replan Algorithmus benötigt wird, der Anzahl der benötigten Knoten zur Lösung des gleichen Schnappschussproblems mit dem Exact-Replan+WS Algorithmus gegenübergestellt. Die horizontale Achse repräsentiert die Anzahl der durch den Exact-Replan erzeugten Knoten. Auf der vertikalen Achse kann die Anzahl der durch den Exact-Replan+WS erzeugten Knoten abgelesen werden. Aufgrund der großen Datenmenge wurde das Koordinatensystem gerastert. Die Färbung eines jeden Feldes $[x, x+\varepsilon] \times [y, y+\varepsilon]$ repräsentiert den prozentualen Anteil der Schnappschussprobleme, für deren Lösung der Exact-Replan zwischen x und $x+\varepsilon$ viele Knoten und der Exact-Replan+WS zwischen y und $y+\varepsilon$ viele Knoten erzeugt hat. Die gestrichelte Diagonale repräsentiert den Bereich in dem es keine signifikanten Unterschiede gibt. Die Diagramme sind wie folgt zu interpretieren: Je flacher der Verlauf der Punktwolke ist, desto größer sind die Einsparungen an erzeugten Knoten durch die Verwendung des Warmstarts.

Gebäude A

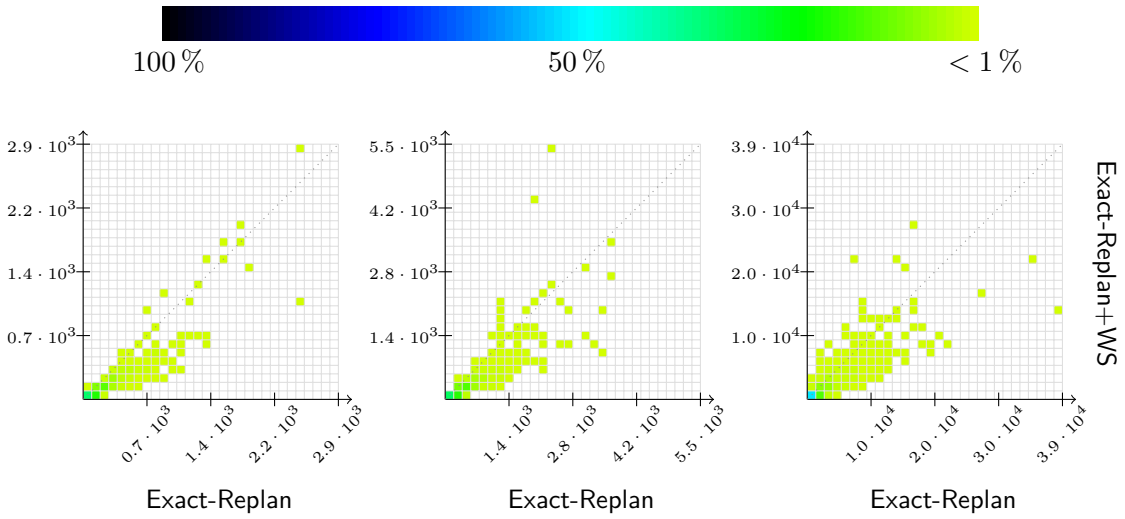


Abbildung B.1: Down Peak Verkehr, 1125, 1407 und 2026.

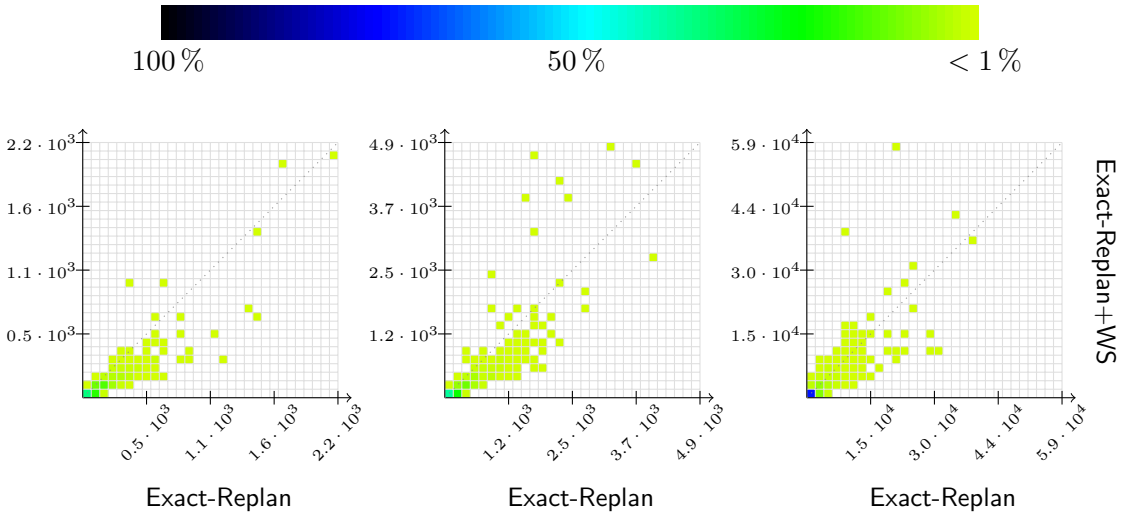


Abbildung B.2: Real Down Peak Verkehr, 1125, 1407 und 2026.

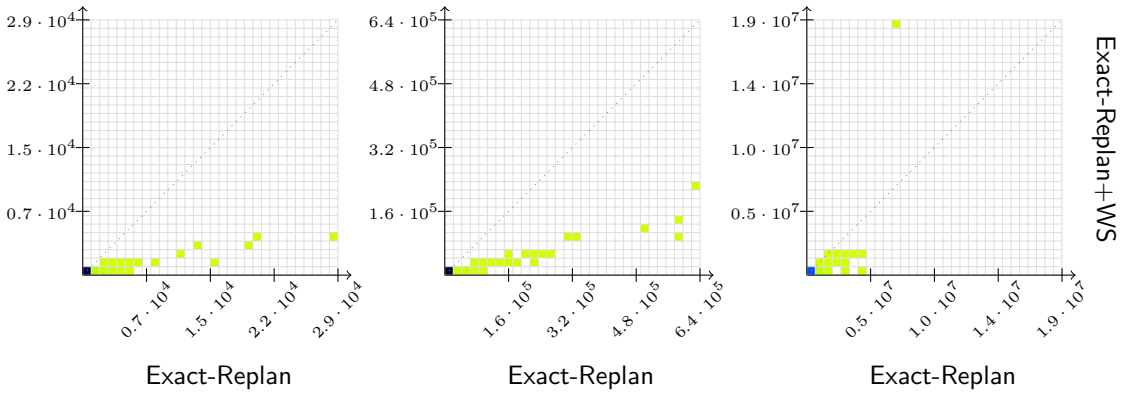


Abbildung B.3: Up Peak Verkehr, 1125, 1407 und 2026.

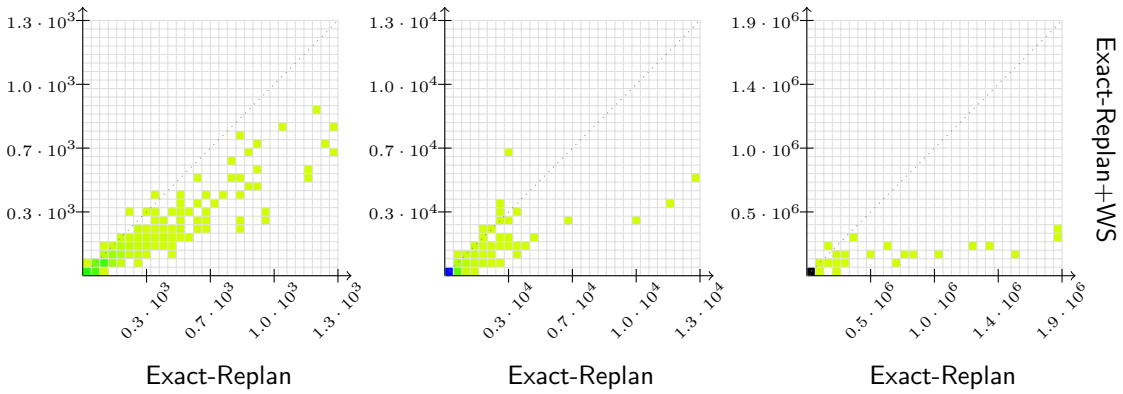


Abbildung B.4: Lunch Peak Verkehr, 1125, 1407 und 2026.

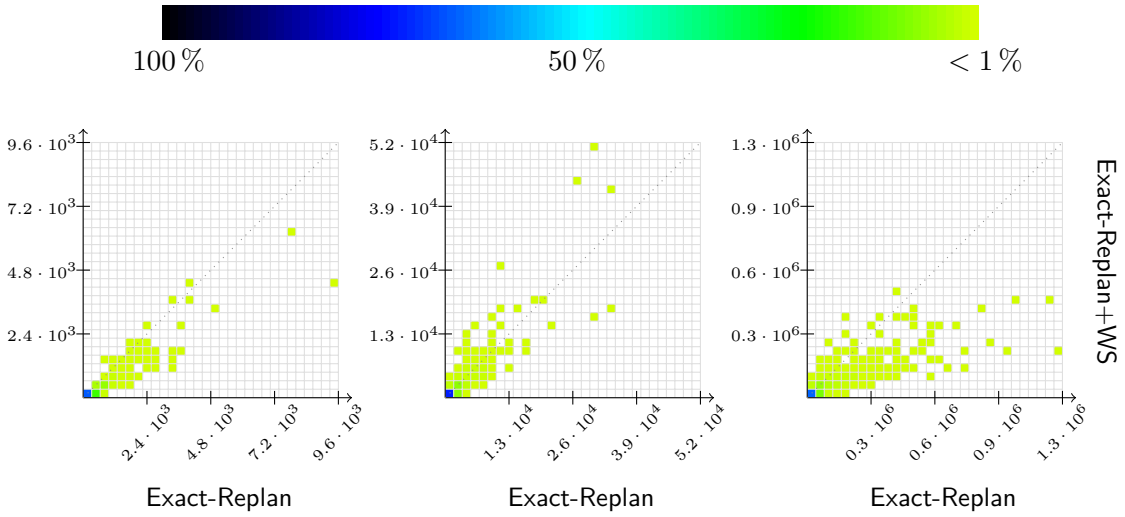


Abbildung B.5: Interfloor Verkehr, 1125, 1407 und 2026.

Gebäude B

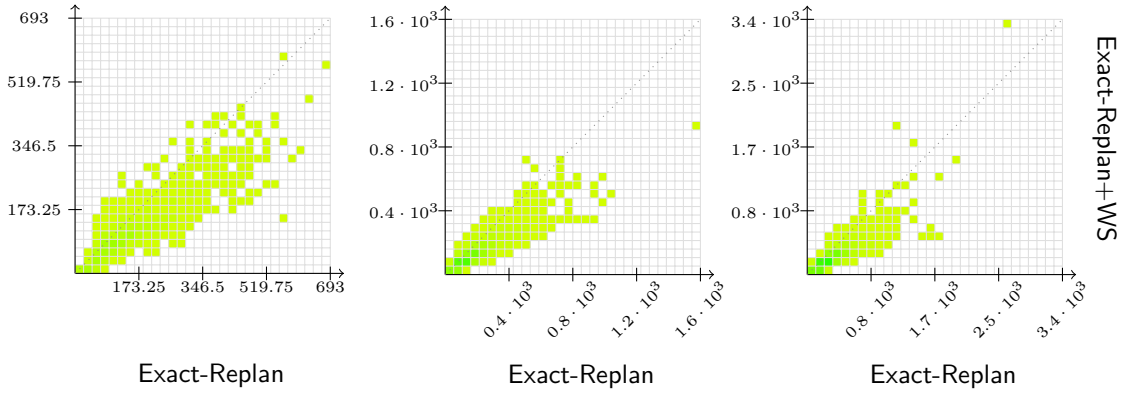


Abbildung B.6: Down Peak Verkehr, 2624, 3280 und 4723.

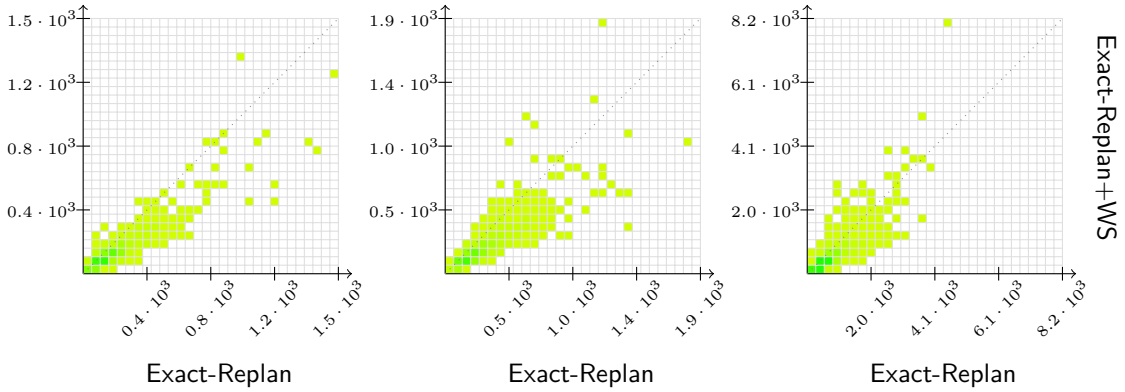


Abbildung B.7: Real Down Peak Verkehr, 2624, 3280 und 4723.

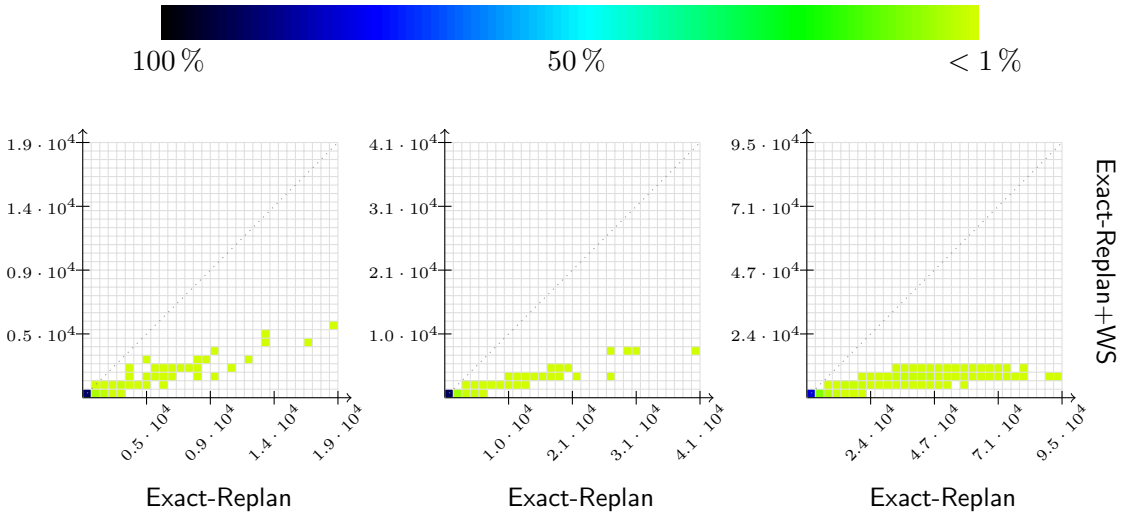


Abbildung B.8: Up Peak Verkehr, 2624, 3280 und 4723.

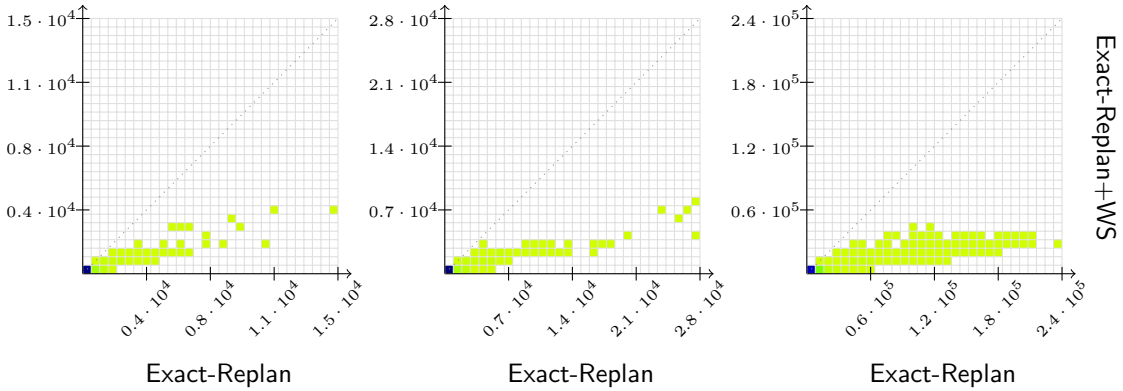


Abbildung B.9: Real Up Peak Verkehr, 2624, 3280 und 4723.

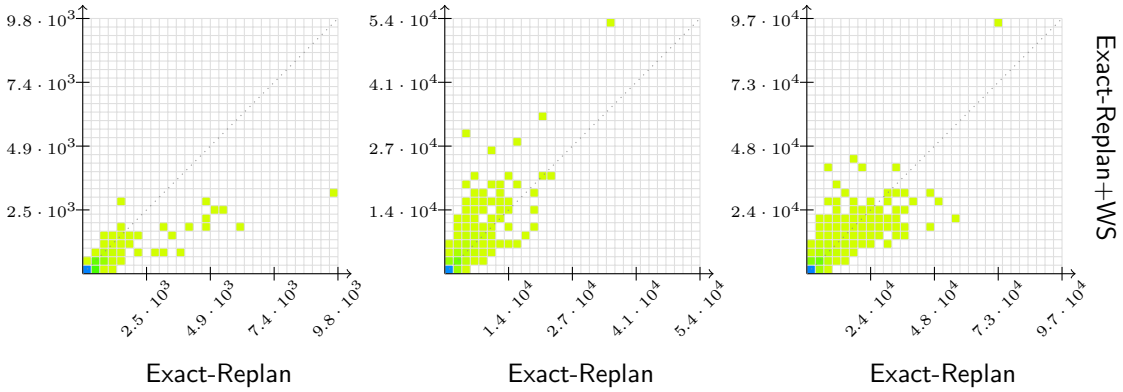


Abbildung B.10: Lunch Peak Verkehr, 2624, 3280 und 4723.

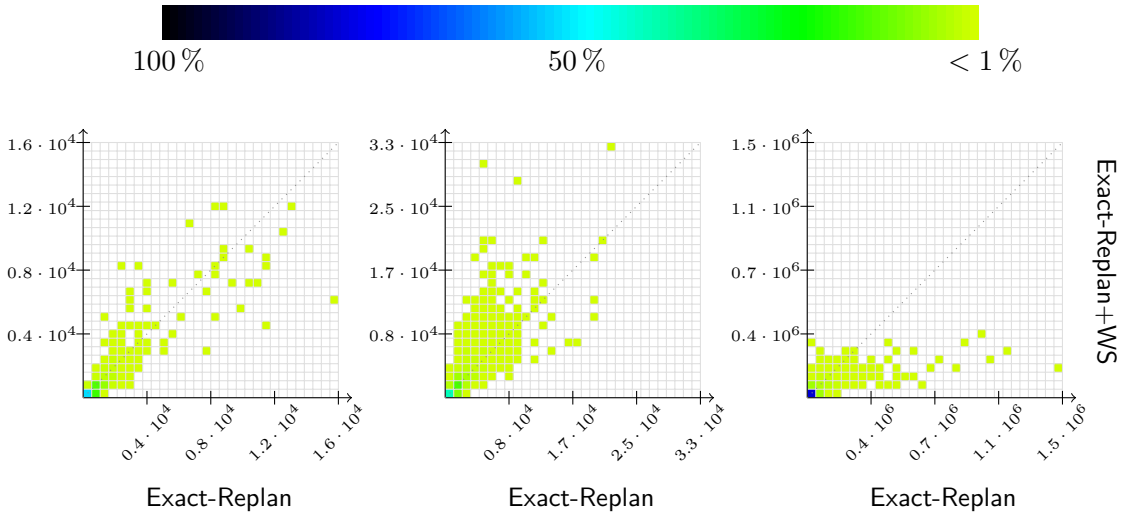


Abbildung B.11: Interfloor Verkehr, 2624, 3280 und 4723.

Gebäude C

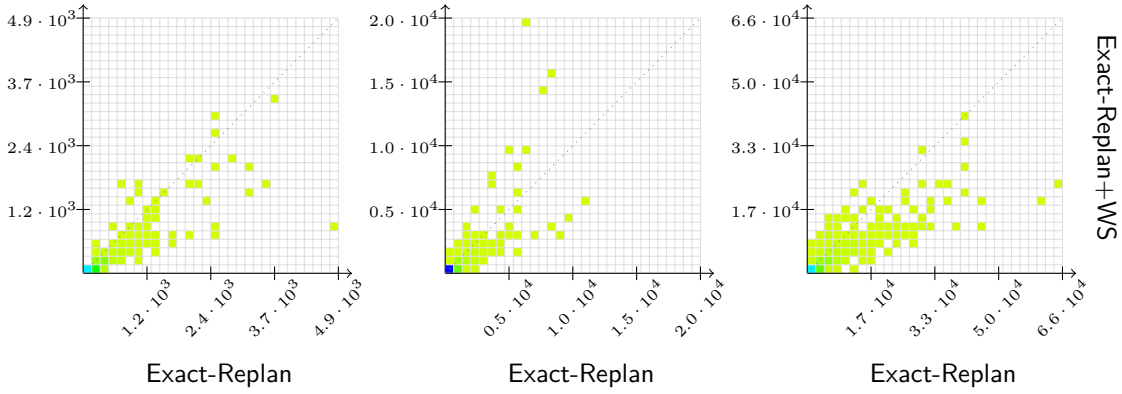


Abbildung B.12: Down Peak Verkehr, 1040, 1300 und 1872.

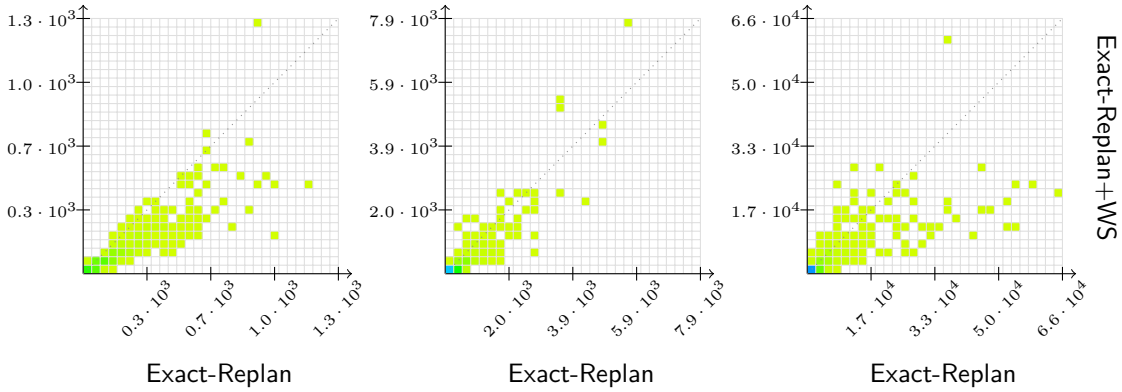


Abbildung B.13: Real Down Peak Verkehr, 1040, 1300 und 1872.

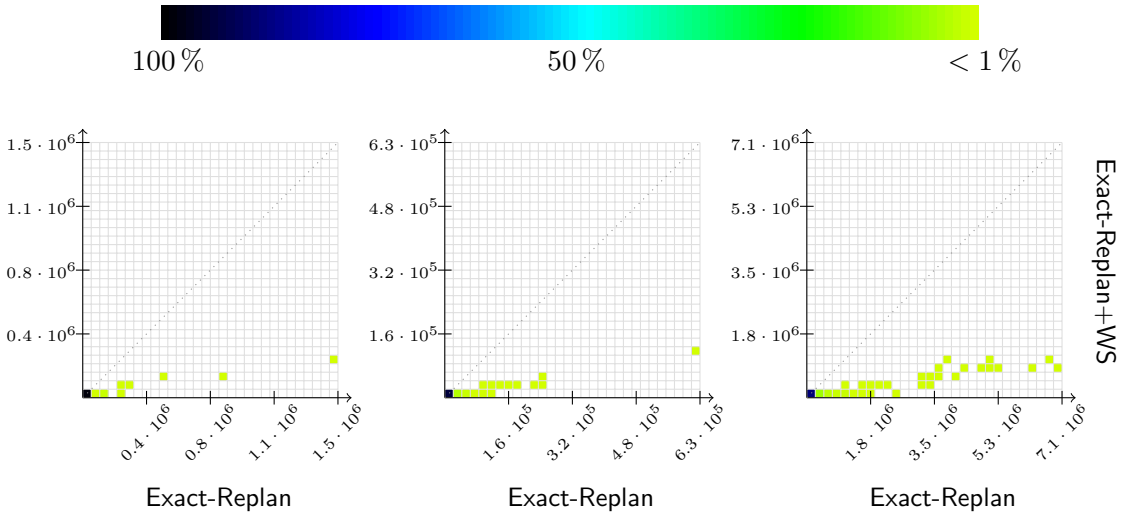


Abbildung B.14: Up Peak Verkehr, 1040, 1300 und 1872.

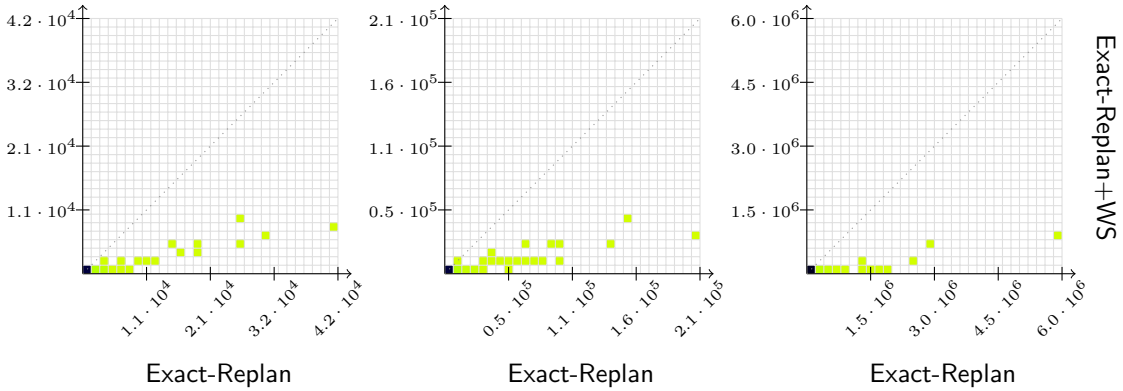


Abbildung B.15: Real Up Peak Verkehr, 1040, 1300 und 1872.

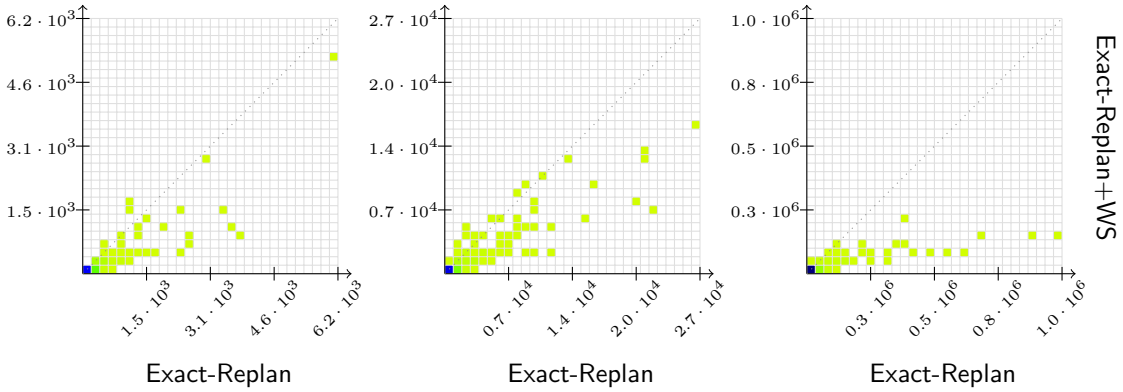


Abbildung B.16: Lunch Peak Verkehr, 1040, 1300 und 1872.

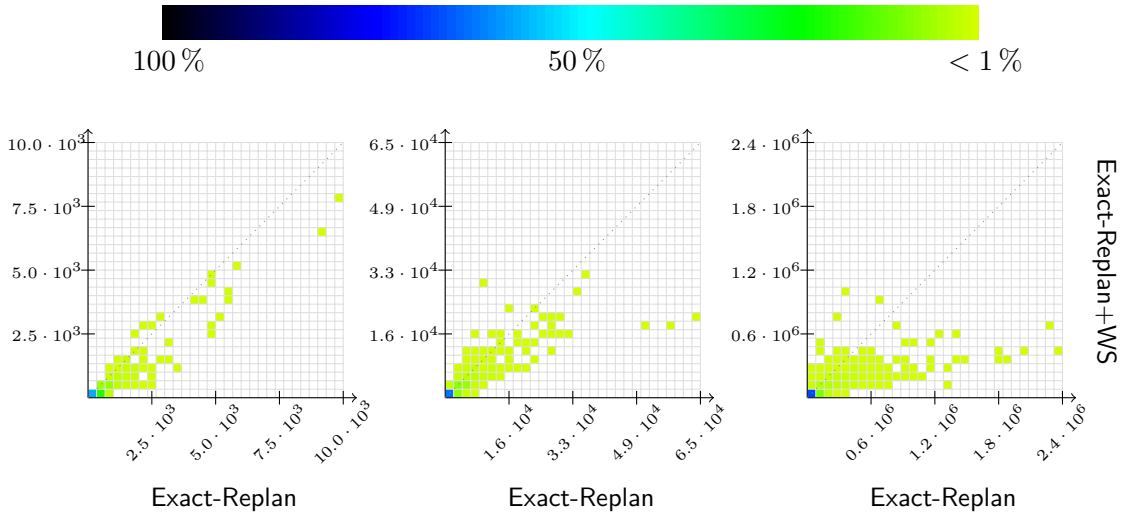


Abbildung B.17: Interfloor Verkehr, 1040, 1300 und 1872.

Rechenzeiten in den Schnappschussproblemen

In den folgenden Abbildungen ist die Rechenzeit in Sekunden, welche für die Lösung eines Schnappschussproblems mit dem Exact-Replan Algorithmus benötigt wird, der Rechenzeit zur Lösung des gleichen Schnappschussproblems mit dem Exact-Replan+WS Algorithmus gegenübergestellt. Die horizontale Achse repräsentiert die Rechenzeiten des Exact-Replan. Auf der vertikalen Achse könnten die Rechenzeiten des Exact-Replan+WS abgelesen werden. Aufgrund der großen Datenmenge wurde das Koordinatensystem gerastert. Die Färbung eines jeden Feldes $[x, x+\varepsilon] \times [y, y+\varepsilon]$ repräsentiert den prozentualen Anteil der Schnappschussprobleme, für deren Lösung der Exact-Replan zwischen x und $x+\varepsilon$ Sekunden und der Exact-Replan+WS zwischen y und $y+\varepsilon$ Sekunden benötigt. Die gestrichelte Diagonale repräsentiert den Bereich in dem es keine signifikanten Unterschiede gibt. Die Diagramme sind wie folgt zu interpretieren: Je flacher der Verlauf der Punktwolke ist, desto größer sind die Zeiteinsparungen die sich durch die Verwendung des Warmstarts ergeben.

Gebäude A

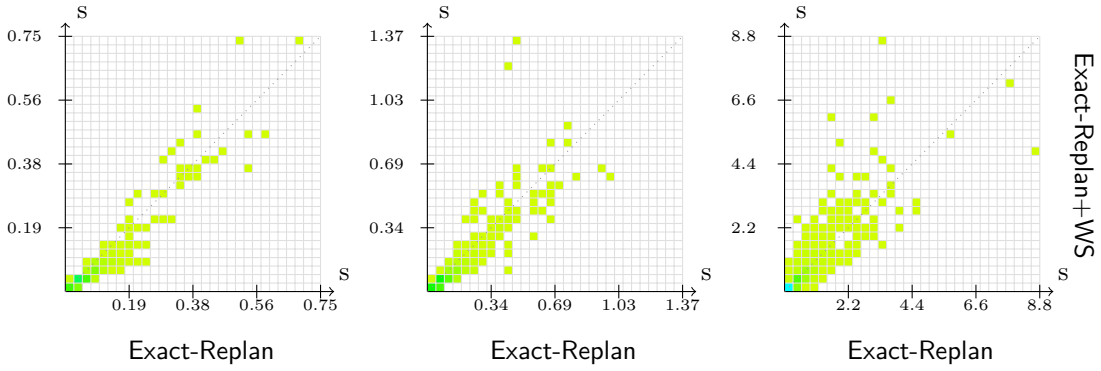


Abbildung B.18: Down Peak Verkehr, 1125, 1407 und 2026.

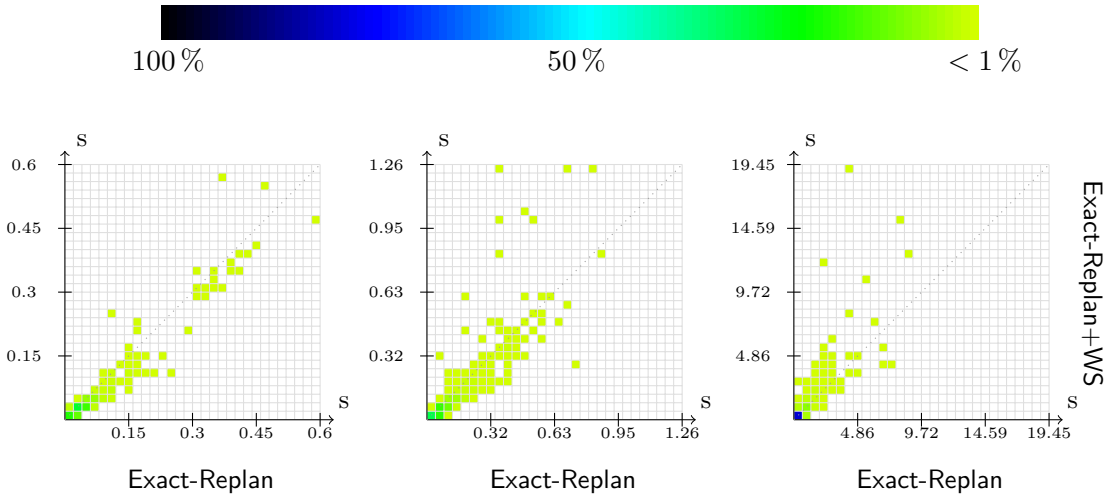


Abbildung B.19: Real Down Peak Verkehr, 1125, 1407 und 2026.

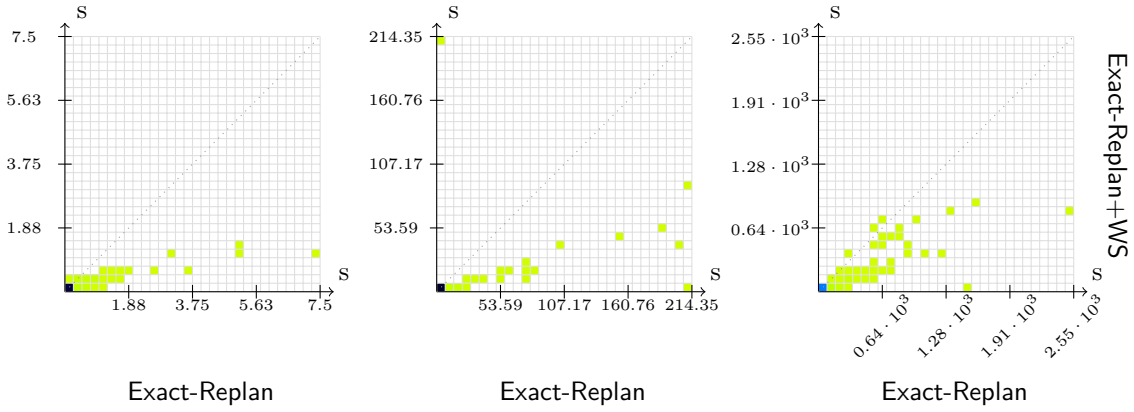


Abbildung B.20: Up Peak Verkehr, 1125, 1407 und 2026.

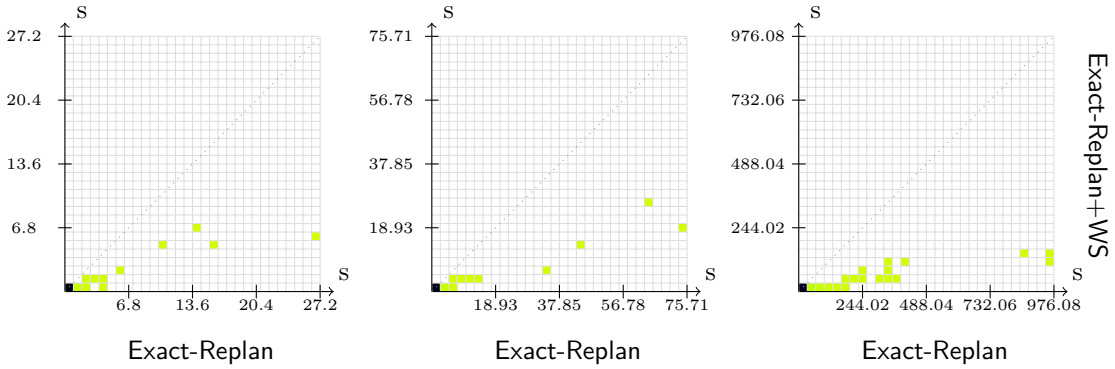


Abbildung B.21: Real Up Peak Verkehr, 1125, 1407 und 2026.

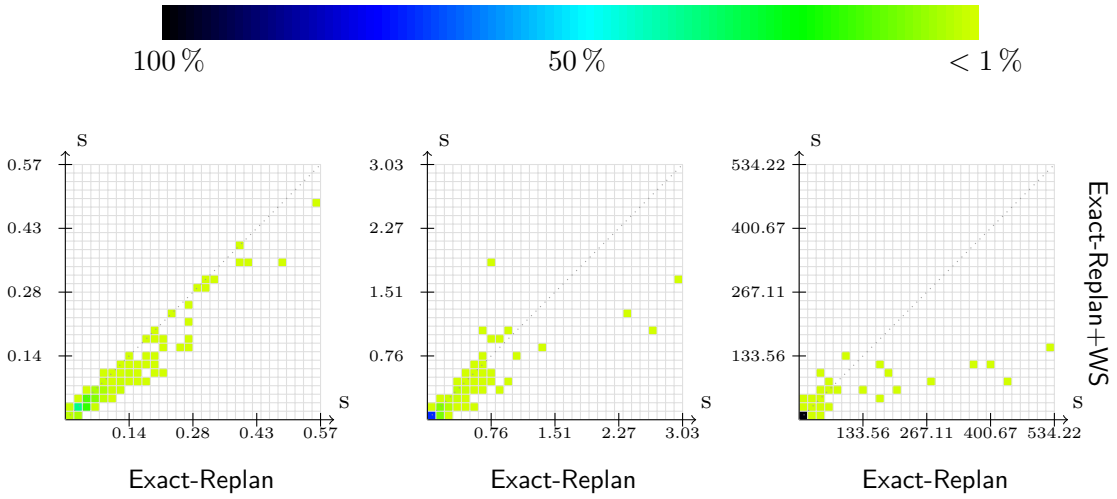


Abbildung B.22: Lunch Peak Verkehr, 1125, 1407 und 2026.

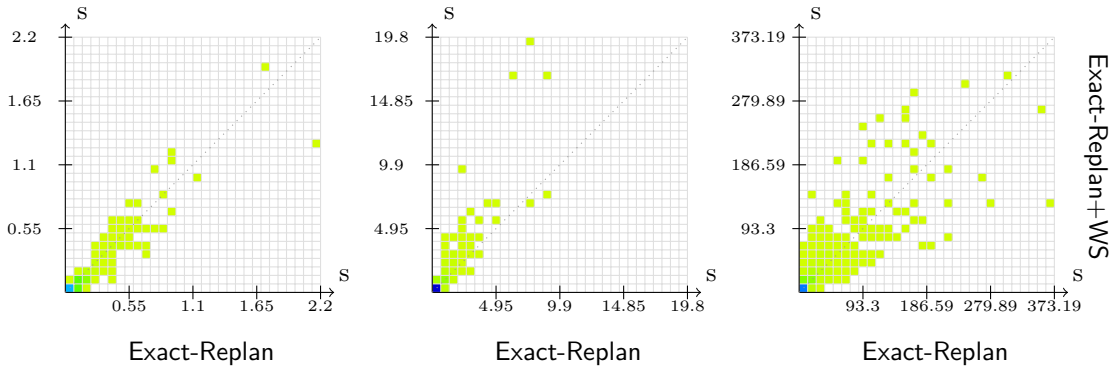


Abbildung B.23: Interfloor Verkehr, 1125, 1407 und 2026.

Gebäude B

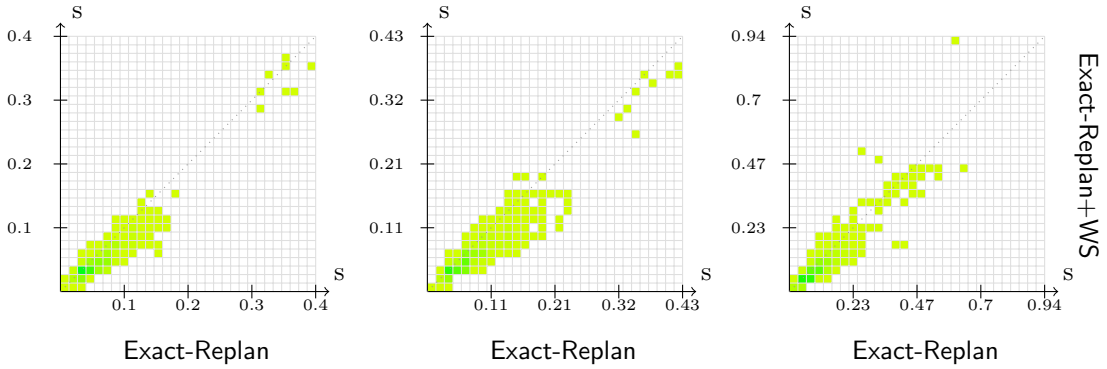


Abbildung B.24: Down Peak Verkehr, 2624, 3280 und 4723.

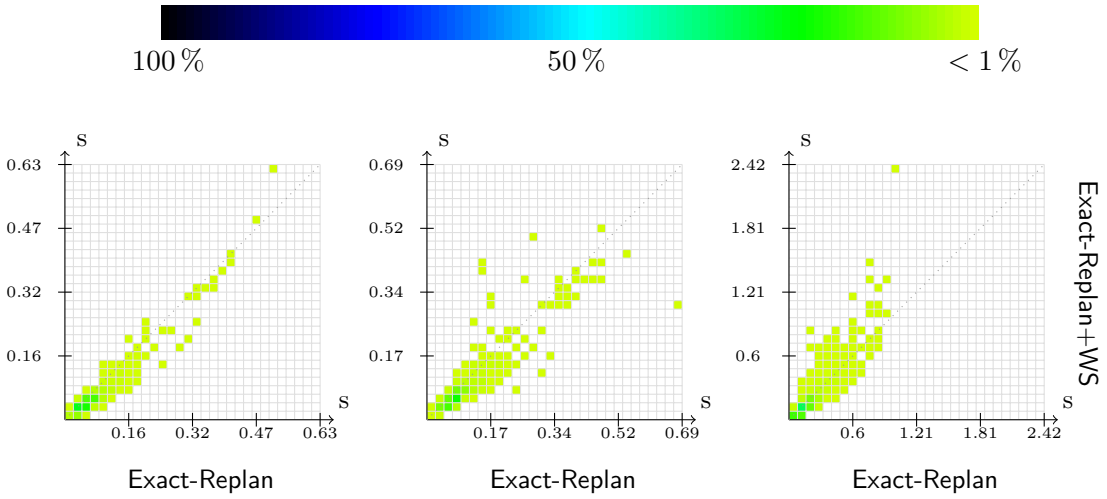


Abbildung B.25: Real Down Peak Verkehr, 2624, 3280 und 4723.

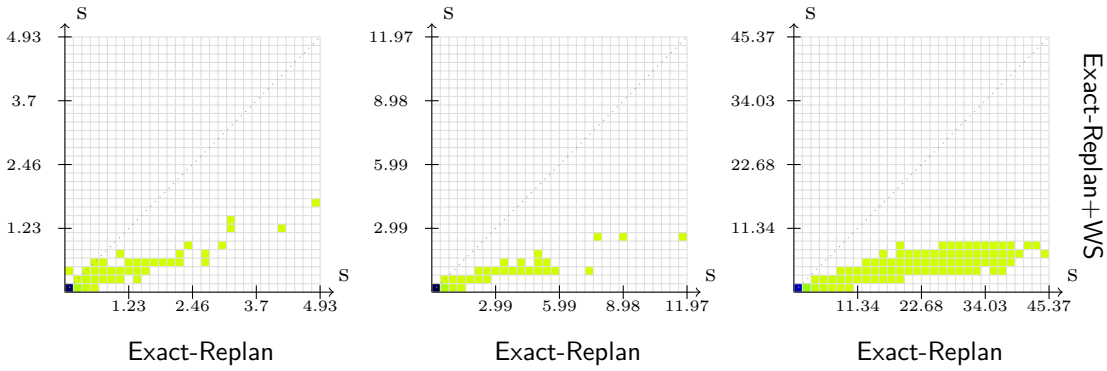


Abbildung B.26: Up Peak Verkehr, 2624, 3280 und 4723.

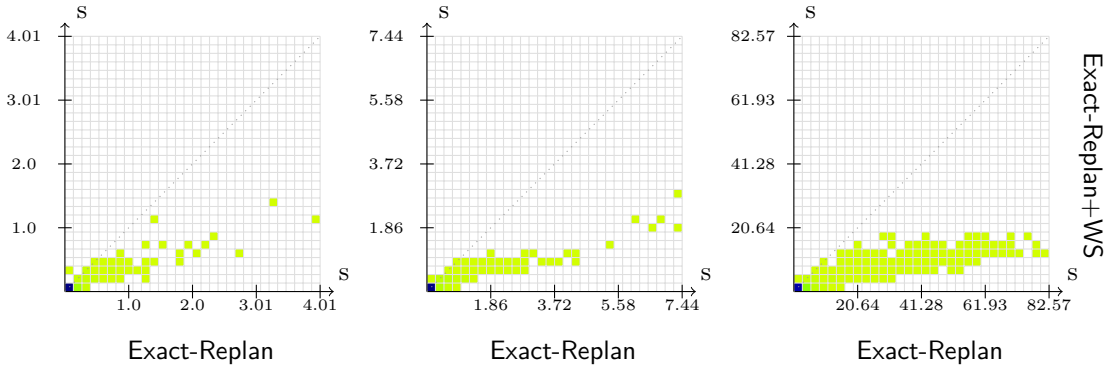


Abbildung B.27: Real Up Peak Verkehr, 2624, 3280 und 4723.

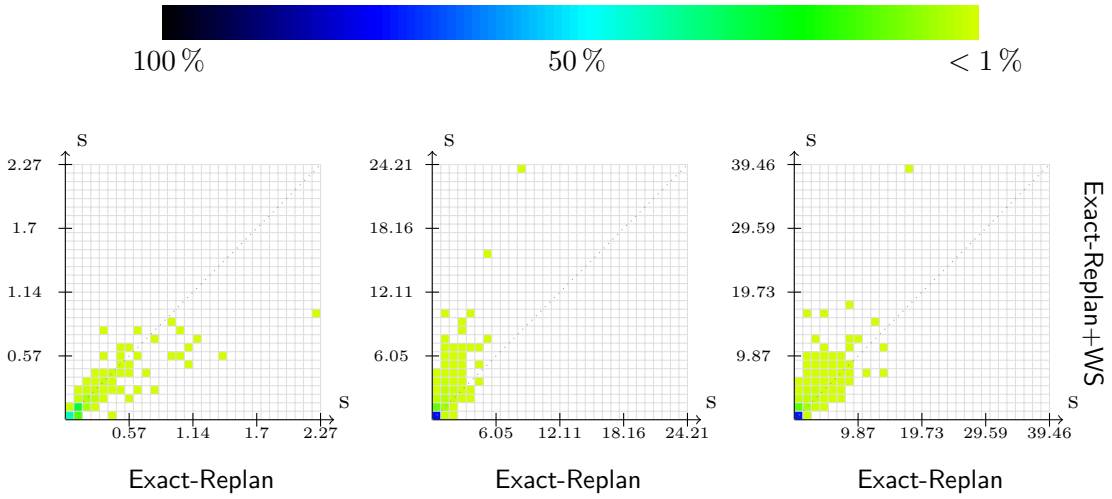


Abbildung B.28: Lunch Peak Verkehr, 2624, 3280 und 4723.

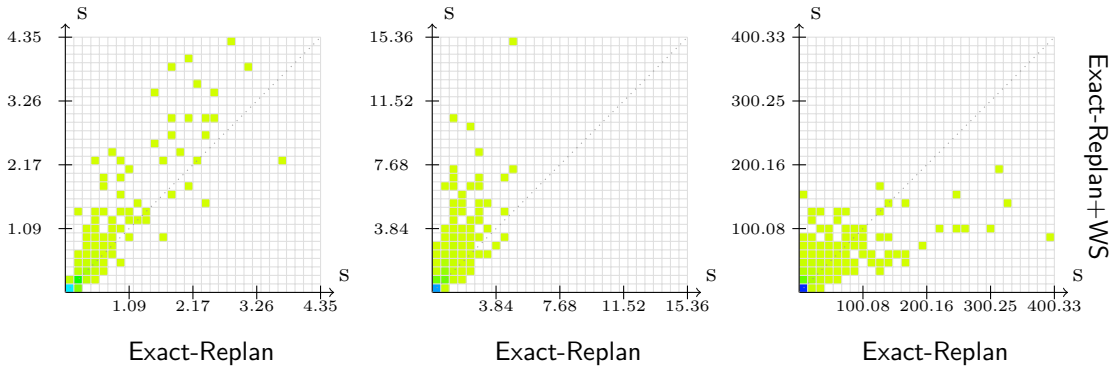


Abbildung B.29: Interfloor Verkehr, 2624, 3280 und 4723.

Gebäude C

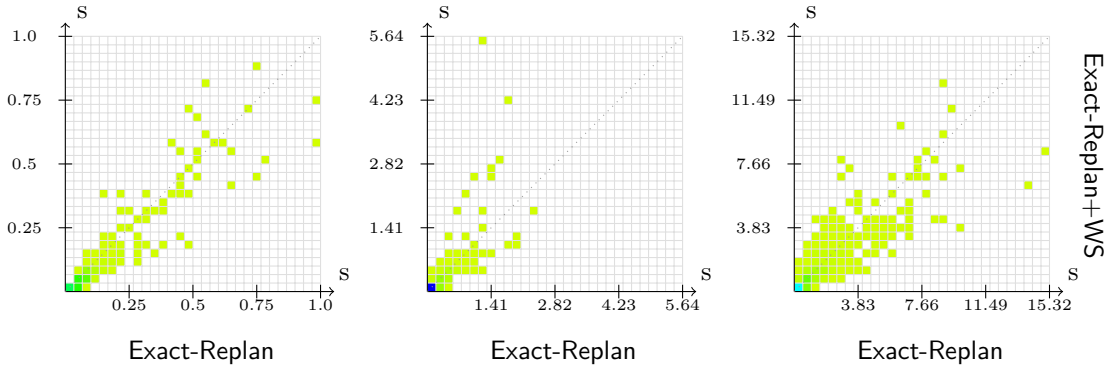


Abbildung B.30: Down Peak Verkehr, 1040, 1300 und 1872.

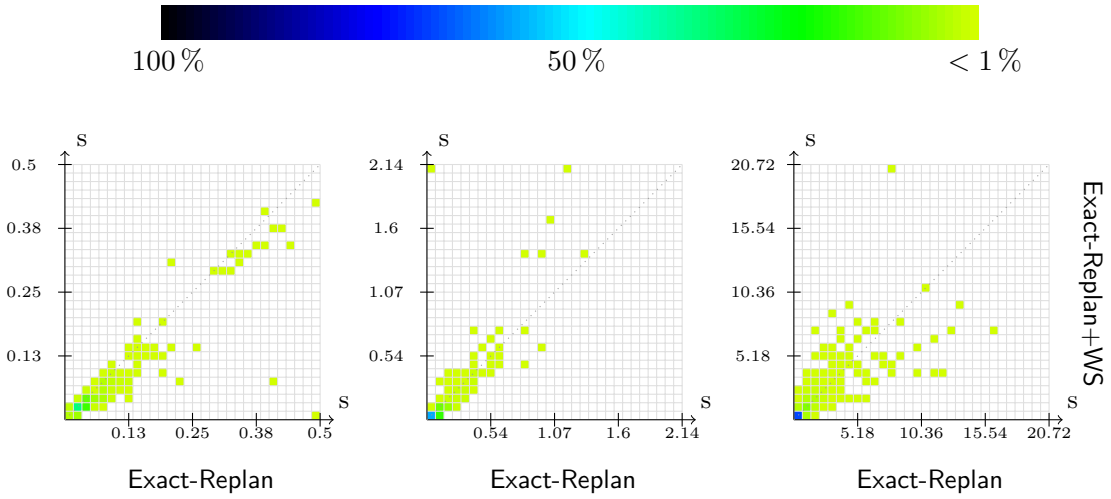


Abbildung B.31: Real Down Peak Verkehr, 1040, 1300 und 1872.

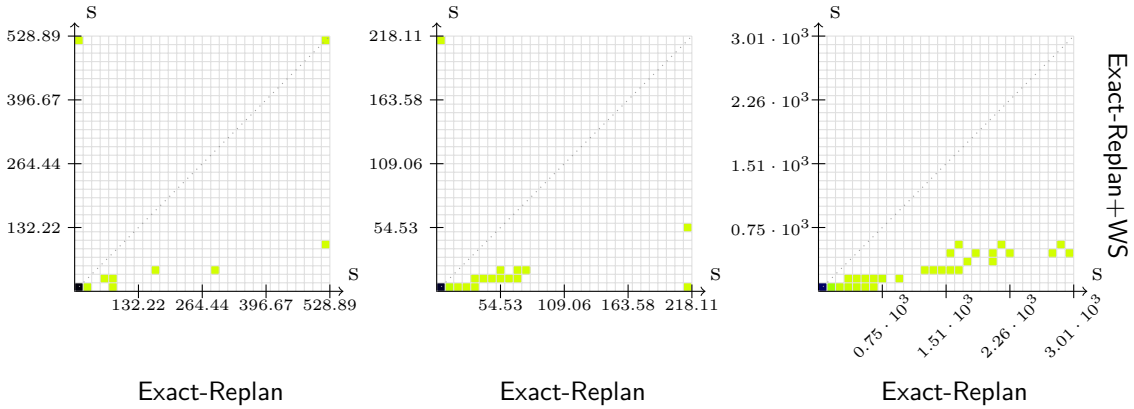


Abbildung B.32: Up Peak Verkehr, 1040, 1300 und 1872.

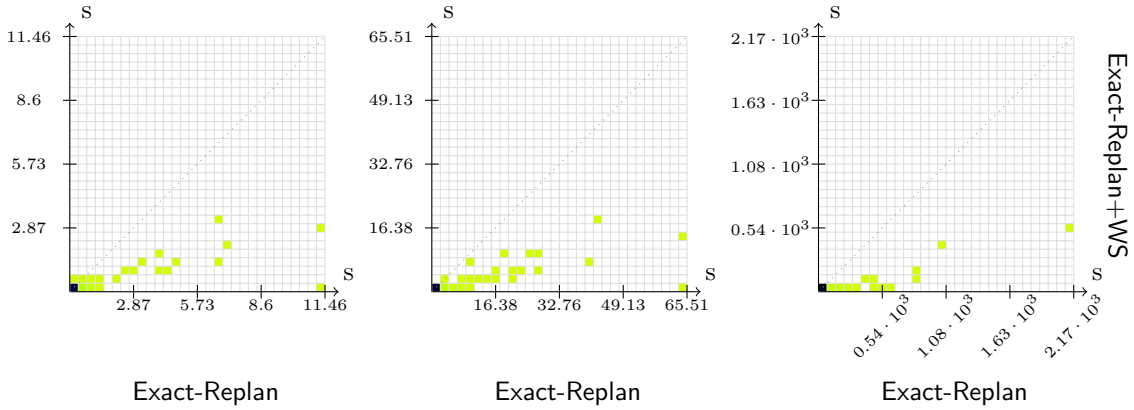


Abbildung B.33: Real Up Peak Verkehr, 1040, 1300 und 1872.

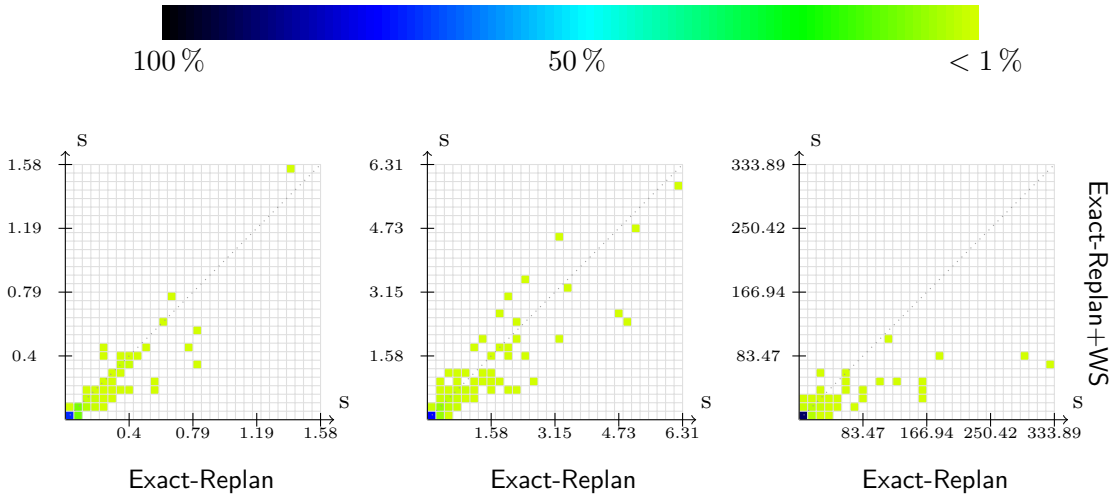


Abbildung B.34: Lunch Peak Verkehr, 1040, 1300 und 1872.

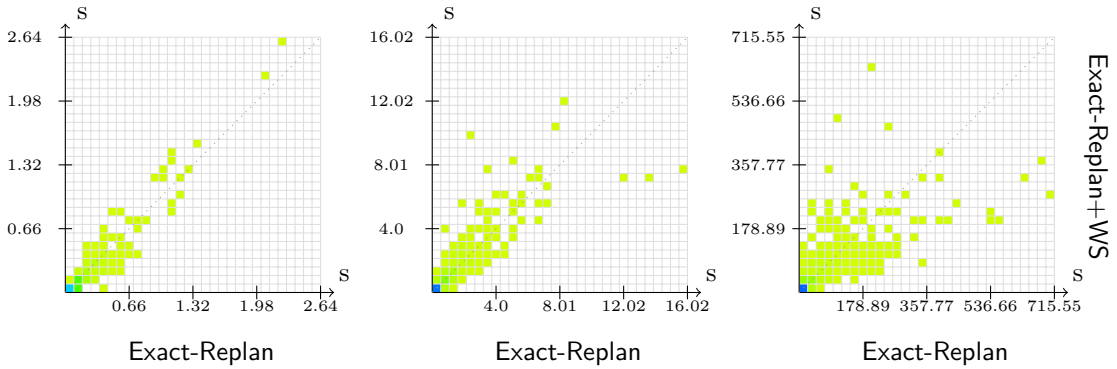
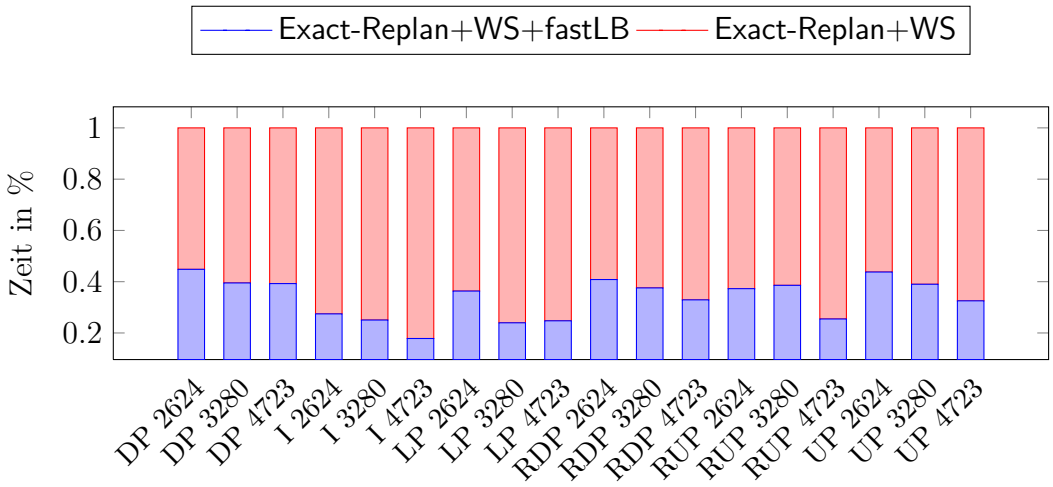


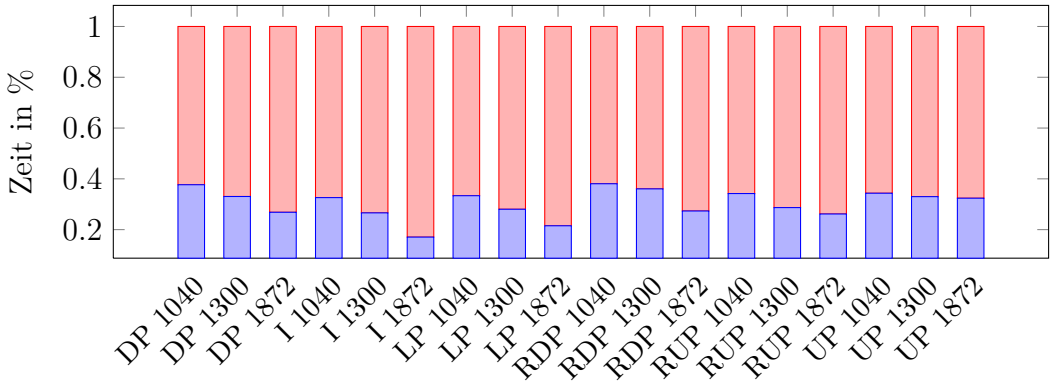
Abbildung B.35: Interfloor Verkehr, 1040, 1300 und 1872.

B.3 Aktualisieren der unteren Schranke der reduzierten Kosten der Knoten

In den Abbildungen B.36a und B.36b sind die Zeiten für die Aktualisierung der unteren Schranke der reduzierten Kosten im Exact-Replan+WS+fastLB dem Exact-Replan+WS gegenübergestellt. Die Aktualisierungszeiten des Exact-Replan+WS sind in jeder Instanz auf 1 skaliert. Es ist deutlich zu sehen, dass die benötigte Zeit auf jeder Instanz signifikant reduziert werden konnte. Ebenfalls erkennbar ist, dass mit steigender Passagierauslastung die prozentuale Einsparung zunimmt. So kommt es aufgrund der Ausnutzung der speziellen Struktur der unteren Schranke der reduzierten Kosten auf der Lunch Peak Instanz mit 1040 Passagieren zu einer Einsparung von etwa 60 %, bei 1300 Passagieren etwa 70 % und mit 1872 Passagieren von etwa 80 %.



(a) Gebäude B.

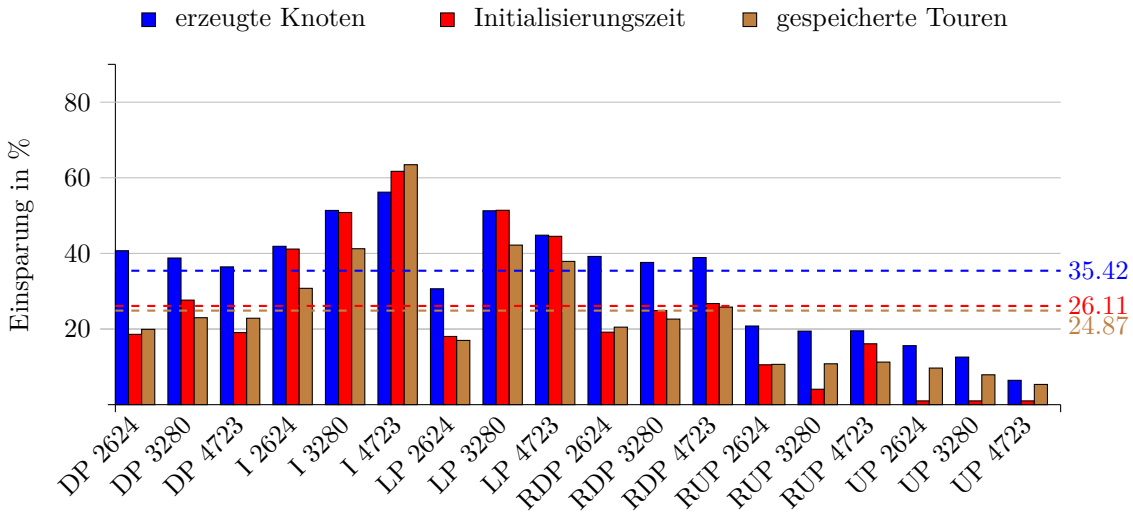


(b) Gebäude C.

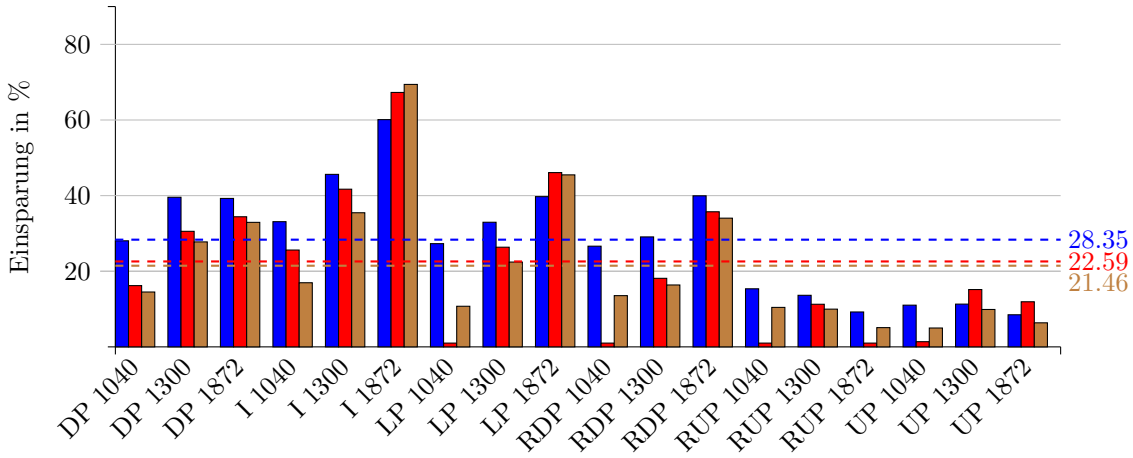
Abbildung B.36: Grafische Darstellung der Reduzierung der benötigten Rechenzeit für Aktualisierung der unteren Schranke der reduzierten Kosten unter Ausnutzung der in Abschnitt 5.2 erläuterten Struktur.

B.4 Pseudo-Branching

In den Abbildungen B.37a und B.37b sind die prozentualen Einsparungen erzeugter Knoten, benötigter Initialisierungszeit des Warmstartes und zwischengespeicherter Touren, welche durch die Verwendung des Pseudo-Branchings entstehen auf den Gebäuden B und C dargestellt. Der Mittelwert über alle Verkehrsmuster ist durch eine horizontale gestrichelte Linie visualisiert. Als Vergleichsgrundlage dient Exact-Replan-+WS+fastLB. In der Tabelle B.19 sind die zugrundeliegenden Daten der Abbildung 5.5 für das Gebäude A zu sehen. Die Tabellen B.20 und B.21 beinhalten die zugrundeliegenden Daten der beiden folgenden Abbildungen.



(a) Gebäude B.



(b) Gebäude C.

Abbildung B.37: Prozentuale Einsparungen der erzeugten Knoten, zwischengespeicherten Touren und der Initialisierungszeit des Warmstartes im Exact-Replan+WS+fastLB+PB im Vergleich Exact-Replan+WS+fastLB für die Gebäude B und C. Die Mittelwerte der Ersparnisse sind in den jeweiligen Farben und gestrichelt dargestellt.

Instanz	Exact-Replan+WS+fastLB						Exact-Replan+WS+fastLB+PB					
	gen. Knoten		gesp. Touren		Initialisierung		gen. Knoten		gesp. Touren		Initialisierung	
	∅	Σ	∅	Σ	∅	Σ	∅	Σ	∅	Σ	∅	Σ
DP 1125	25.27	145987	6.79	39216	**0:00:00**	**0:00:02**	**17.13**	**98957**	**5.37**	**31046**	**0:00:00**	**0:00:02**
DP 1407	47.33	321722	13.17	89558	**0:00:00**	0:00:06	**32.45**	**220589**	**10.53**	**71583**	**0:00:00**	**0:00:05**
DP 2026	277.92	2574660	86.61	802365	**0:00:00**	0:01:39	**168.13**	**1557577**	**59.53**	**551452**	**0:00:00**	**0:01:04**
IF 1125	45.89	269809	11.82	69483	**0:00:00**	0:00:06	**30.35**	**178448**	**9.99**	**58725**	**0:00:00**	**0:00:05**
IF 1407	257.07	1810791	81.85	576556	**0:00:00**	0:01:26	**129.78**	**914165**	**49.64**	**349688**	**0:00:00**	**0:00:43**
IF 2026	6579.44	62017756	3923.40	36981978	**0:00:00**	2:27:24	**2765.65**	**26069061**	**1227.76**	**11572900**	**0:00:00**	**0:51:50**
LP 1125	16.00	94094	3.90	22952	**0:00:00**	**0:00:01**	**11.33**	**66612**	**3.47**	**20428**	**0:00:00**	**0:00:01**
LP 1407	45.16	317572	12.50	87898	**0:00:00**	0:00:08	**31.92**	**224464**	**10.40**	**73132**	**0:00:00**	**0:00:06**
LP 2026	1195.54	11247599	443.83	4175569	**0:00:00**	0:16:02	**748.97**	**7046335**	**270.38**	**2543689**	**0:00:00**	**0:09:58**
RDP 1125	17.63	103538	4.30	25245	**0:00:00**	**0:00:01**	**12.06**	**70848**	**3.62**	**21267**	**0:00:00**	**0:00:01**
RDP 1407	41.43	291098	11.67	81997	**0:00:00**	0:00:06	**27.75**	**194941**	**9.17**	**64413**	**0:00:00**	**0:00:04**
RDP 2026	263.77	2476782	90.86	853204	**0:00:00**	0:01:48	**146.38**	**1374488**	**55.19**	**518192**	**0:00:00**	**0:01:01**
RUP 1125	37.44	216526	10.27	59397	**0:00:00**	0:00:07	**30.67**	**177410**	**8.86**	**51255**	**0:00:00**	**0:00:06**
RUP 1407	86.24	602269	24.31	169750	**0:00:00**	0:00:23	**73.65**	**514378**	**21.91**	**153045**	**0:00:00**	**0:00:19**
RUP 2026	626.89	6446901	190.05	1954426	**0:00:00**	0:07:11	**553.04**	**5687509**	**178.08**	**1831355**	**0:00:00**	**0:06:32**
UP 1125	19.64	113842	4.83	27999	**0:00:00**	**0:00:02**	**16.91**	**98028**	**4.37**	**25315**	**0:00:00**	**0:00:02**
UP 1407	320.55	2229084	96.62	671897	**0:00:00**	0:01:29	**259.06**	**1801496**	**80.46**	**559487**	**0:00:00**	**0:01:10**
UP 2026	3873.02	38482321	1004.22	9977988	**0:00:00**	0:47:00	**3564.73**	**35419129**	**945.78**	**9397270**	**0:00:00**	**0:44:12**

Tabelle B.19: Die zu der Abbildung 5.5 gehörigen Daten für das Gebäude A. Hierbei wird die Anzahl der generierten Knoten, zwischengespeicherten Touren und die für die Initialisierung des Warmstarts benötigte Zeit des Exact-Replan+WS+fastLB mit den Werten des Exact-Replan+WS+fastLB+PB verglichen. Der Durchschnitt sowie die Summe ist über alle Aufzüge und Schnappschussprobleme gebildet wurden.Die Zeiten für die Initialisierung des Warmstartes sind in h:mm:ss angegeben.

Instanz	Exact-Replan+WS+fastLB						Exact-Replan+WS+fastLB+PB					
	gen. Knoten		gesp. Touren		Initialisierung		gen. Knoten		gesp. Touren		Initialisierung	
	∅	Σ	∅	Σ	∅	Σ	∅	Σ	∅	Σ	∅	Σ
DP 2624	14.09	206339	2.85	41701	**0:00:00**	0:00:02	**8.36**	**122346**	**2.28**	**33395**	**0:00:00**	**0:00:01**
DP 3280	17.92	305487	4.15	70686	**0:00:00**	0:00:04	**10.97**	**187027**	**3.19**	**54426**	**0:00:00**	**0:00:03**
DP 4723	28.86	662938	7.30	167698	**0:00:00**	0:00:10	**18.35**	**421380**	**5.63**	**129366**	**0:00:00**	**0:00:08**
IF 2624	83.58	1249077	23.23	347201	**0:00:00**	0:00:40	**48.58**	**726016**	**16.08**	**240374**	**0:00:00**	**0:00:23**
IF 3280	280.12	4889803	78.52	1370606	**0:00:00**	0:03:37	**136.22**	**2377872**	**46.14**	**805359**	**0:00:00**	**0:01:46**
IF 4723	3839.68	84534403	1563.90	34430873	**0:00:00**	2:29:24	**1681.96**	**37029993**	**571.38**	**12579599**	**0:00:00**	**0:57:12**
LP 2624	33.29	491058	9.36	138027	**0:00:00**	0:00:09	**23.09**	**340551**	**7.77**	**114577**	**0:00:00**	**0:00:07**
LP 3280	274.57	4795136	78.69	1374195	**0:00:00**	0:03:37	**133.77**	**2336214**	**45.48**	**794289**	**0:00:00**	**0:01:45**
LP 4723	451.81	10019304	131.25	2910494	**0:00:00**	0:08:01	**249.33**	**6529035**	**81.51**	**1807648**	**0:00:00**	**0:04:27**
RDP 2624	15.07	225164	3.36	50212	**0:00:00**	0:00:03	**9.16**	**136908**	**2.67**	**39919**	**0:00:00**	**0:00:02**
RDP 3280	23.67	413141	6.03	105251	**0:00:00**	0:00:06	**14.77**	**257740**	**4.66**	**81432**	**0:00:00**	**0:00:05**
RDP 4723	58.80	1321376	15.49	348062	**0:00:00**	0:00:28	**35.93**	**807391**	**11.49**	**258266**	**0:00:00**	**0:00:20**
RUP 2624	18.76	284089	4.56	68990	**0:00:00**	0:00:05	**14.86**	**224998**	**4.07**	**61634**	**0:00:00**	**0:00:04**
RUP 3280	32.44	577770	8.55	152311	**0:00:00**	0:00:12	**26.14**	**465413**	**7.63**	**135853**	**0:00:00**	**0:00:11**
RUP 4723	320.45	29886515	89.31	8329717	**0:00:00**	0:23:27	**257.86**	**24048861**	**79.26**	**7391931**	**0:00:00**	**0:19:40**
UP 2624	16.91	247480	4.09	59805	**0:00:00**	**0:00:03**	**14.28**	**208875**	**3.69**	**54017**	**0:00:00**	**0:00:03**
UP 3280	25.17	455897	6.54	118452	**0:00:00**	**0:00:08**	**22.01**	**398563**	**6.02**	**109093**	**0:00:00**	**0:00:08**
UP 4723	129.37	13731386	39.59	4202687	**0:00:00**	**0:07:32**	**121.03**	**12847125**	**37.48**	**3978010**	**0:00:00**	**0:07:32**

Tabelle B.20: Die zu der Abbildung B.37a gehörigen Daten für das Gebäude B. Hierbei wird die Anzahl der generierten Knoten, zwischengespeicherten Touren und die für die Initialisierung des Warmstarts benötigte Zeit des Exact-Replan+WS+fastLB mit den Werten des Exact-Replan+WS+fastLB+PB verglichen. Der Durchschnitt sowie die Summe ist über alle Aufzüge und Schnappschussprobleme gebildet wurden. Die Zeiten für die Initialisierung des Warmstartes sind in h:mm:ss angegeben.

Instanz	Exact-Replan+WS+fastLB						Exact-Replan+WS+fastLB+PB					
	gen. Knoten		gesp. Touren		Initialisierung		gen. Knoten		gesp. Touren		Initialisierung	
	∅	Σ	∅	Σ	∅	Σ	∅	Σ	∅	Σ	∅	Σ
DP 1040	37.49	167034	10.48	46686	**0:00:00**	0:00:03	**26.98**	**120186**	**8.96**	**39910**	**0:00:00**	**0:00:02**
DP 1300	99.50	535793	29.10	156712	**0:00:00**	0:00:14	**60.12**	**323762**	**21.02**	**113215**	**0:00:00**	**0:00:10**
DP 1872	549.67	3982376	195.65	1417485	**0:00:00**	0:03:14	**333.89**	**2419011**	**131.23**	**950740**	**0:00:00**	**0:02:07**
IF 1040	67.39	301896	17.81	79767	**0:00:00**	0:00:08	**45.08**	**201944**	**14.79**	**66238**	**0:00:00**	**0:00:06**
IF 1300	448.75	2488346	143.39	795092	**0:00:00**	0:02:14	**244.04**	**1353182**	**92.52**	**513034**	**0:00:00**	**0:01:18**
IF 1872	12449.26	91875548	8203.26	60540031	0:00:02	4:34:00	**4967.50**	**36660154**	**2508.98**	**18516296**	**0:00:00**	**1:29:37**
LP 1040	24.68	111313	6.27	28277	**0:00:00**	**0:00:02**	**17.94**	**80929**	**5.60**	**25240**	**0:00:00**	**0:00:02**
LP 1300	113.57	616138	34.63	187863	**0:00:00**	0:00:24	**76.14**	**413069**	**26.86**	**145734**	**0:00:00**	**0:00:18**
LP 1872	1779.61	13186883	757.06	5609790	**0:00:00**	0:21:26	**1072.63**	**7948162**	**412.69**	**3058049**	**0:00:00**	**0:11:33**
RDP 1040	20.02	89710	5.47	24500	**0:00:00**	**0:00:01**	**14.69**	**65819**	**4.73**	**21180**	**0:00:00**	**0:00:01**
RDP 1300	52.77	292879	15.13	83973	**0:00:00**	0:00:06	**37.42**	**207706**	**12.65**	**70225**	**0:00:00**	**0:00:05**
RDP 1872	499.48	3686186	189.89	1401411	**0:00:00**	0:03:04	**299.96**	**2213710**	**125.25**	**924371**	**0:00:00**	**0:01:58**
RUP 1040	33.81	152486	9.08	40962	**0:00:00**	**0:00:04**	**28.62**	**129058**	**8.13**	**36687**	**0:00:00**	**0:00:04**
RUP 1300	131.40	714186	39.68	215687	**0:00:00**	0:00:36	**113.48**	**616756**	**35.73**	**194170**	**0:00:00**	**0:00:32**
RUP 1872	1186.08	9375926	371.91	2939922	**0:00:00**	0:14:20	**1076.63**	**8510726**	**352.99**	**279038**	**0:00:00**	**0:13:38**
UP 1040	219.56	982543	66.37	296984	**0:00:00**	**0:00:55**	**195.35**	**874190**	**63.06**	**282174**	**0:00:00**	**0:00:55**
UP 1300	286.04	1556039	85.15	463208	**0:00:00**	0:01:14	**253.70**	**1380109**	**76.73**	**417402**	**0:00:00**	**0:01:02**
UP 1872	5974.32	33963983	1862.15	10586305	**0:00:00**	0:39:39	**5466.91**	**31079362**	**1743.75**	**9913225**	**0:00:00**	**0:34:55**

Tabelle B.21: Die zu der Abbildung B.37b gehörigen Daten für das Gebäude C. Hierbei wird die Anzahl der generierten Knoten, zwischengespeicherten Touren und die für die Initialisierung des Warmstarts benötigte Zeit des Exact-Replan+WS+fastLB mit den Werten des Exact-Replan+WS+fastLB+PB verglichen. Der Durchschnitt sowie die Summe ist über alle Aufzüge und Schnappschussprobleme gebildet wurden. Die Zeiten für die Initialisierung des Warmstartes sind in h:mm:ss angegeben.

B.5 Exact-Replan vs. Exact-Replan+WS+fastLB+PB

Rechenzeiten in den Schnappschussproblemen

In den folgenden Abbildungen ist die Rechenzeit in Sekunden, welche für die Lösung eines Schnappschussproblems mit dem Exact-Replan Algorithmus benötigt wird, der Rechenzeit zur Lösung des gleichen Schnappschussproblems mit dem Exact-Replan+WS+fastLB+PB Algorithmus gegenübergestellt. Die horizontale Achse repräsentiert die Rechenzeiten des Exact-Replan. Auf der vertikalen Achse könnten die Rechenzeiten des Exact-Replan+WS+fastLB+PB abgelesen werden. Aufgrund der großen Datenmenge wurde das Koordinatensystem gerastert. Die Färbung eines jeden Feldes $[x, x+\varepsilon] \times [y, y+\varepsilon]$ repräsentiert den prozentualen Anteil der Schnappschussprobleme, für deren Lösung der Exact-Replan zwischen x und $x+\varepsilon$ Sekunden und der Exact-Replan+WS+fastLB+PB zwischen y und $y+\varepsilon$ Sekunden benötigt. Die gestrichelte Diagonale repräsentiert den Bereich in dem es keine signifikanten Veränderungen gibt. Die Diagramme sind wie folgt zu interpretieren. Je flacher der Verlauf der Punktwolke ist, desto größer sind die Zeiteinsparungen die sich durch die Verwendung des Warmstarts ergeben.

Es ist zu erkennen, dass die Punktwolken zu sehr großen Teilen unterhalb der Geraden liegen. Des Weiteren wird deutlich, dass alle Felder, welcher viele Schnappschussprobleme repräsentieren, unterhalb oder auf der Geraden liegen. Diese Ergebnisse decken sich mit den Werten in den Tabellen 5.2, 5.4 und 5.6 auf den Seiten 52, 54 und 56.

Gebäude A

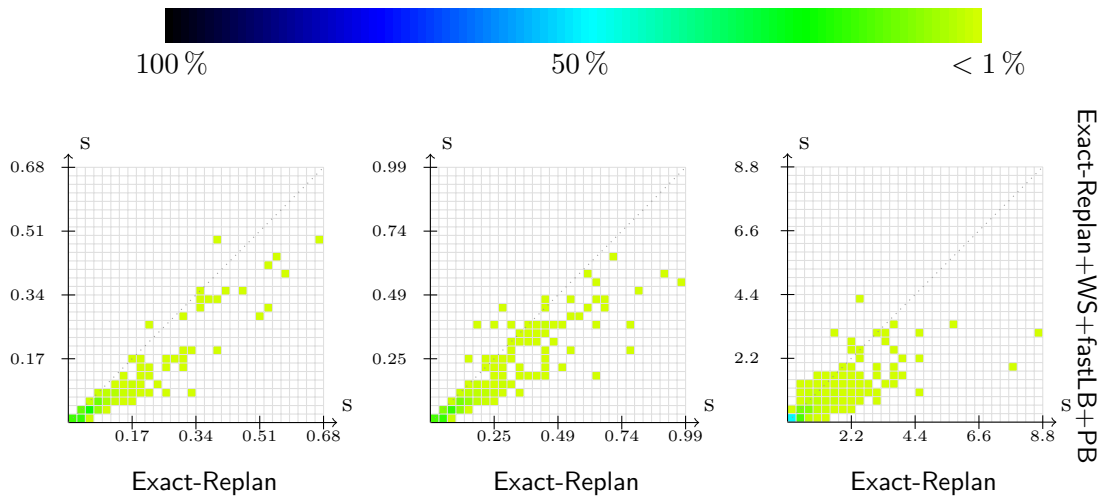


Abbildung B.38: Down Peak Verkehr, 1125, 1407 und 2026.

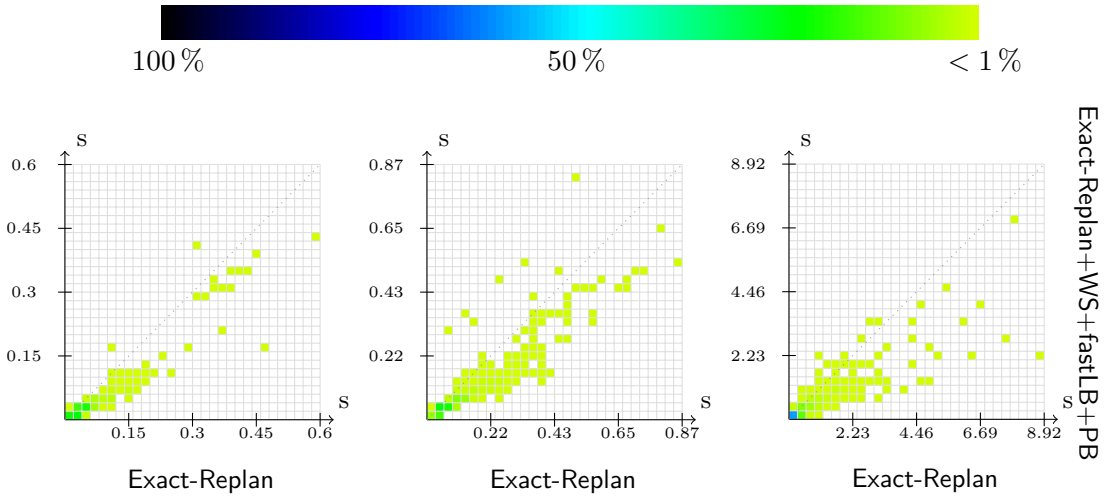


Abbildung B.39: Real Down Peak Verkehr, 1125, 1407 und 2026.

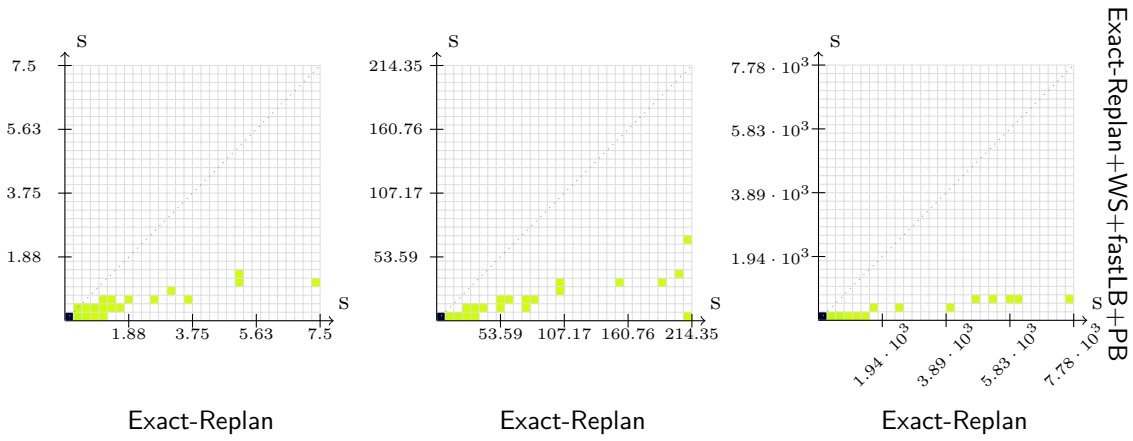


Abbildung B.40: Up Peak Verkehr, 1125, 1407 und 2026.

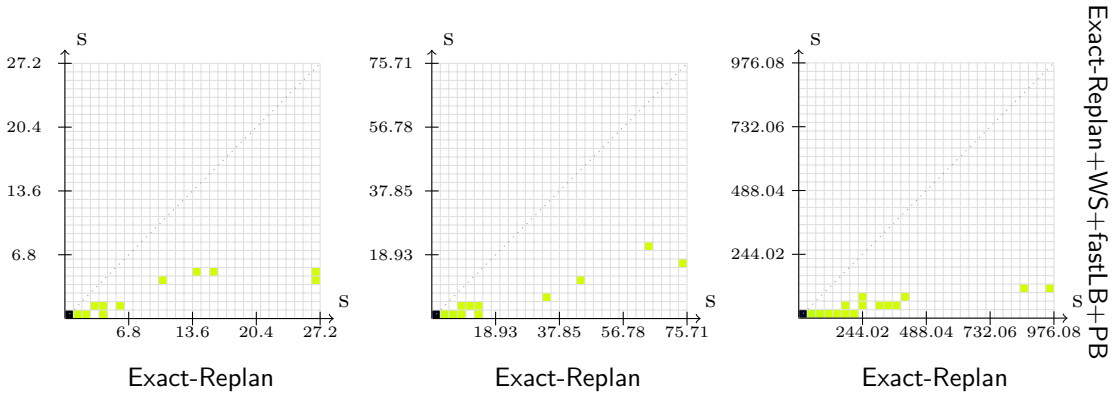


Abbildung B.41: Real Up Peak Verkehr, 1125, 1407 und 2026.

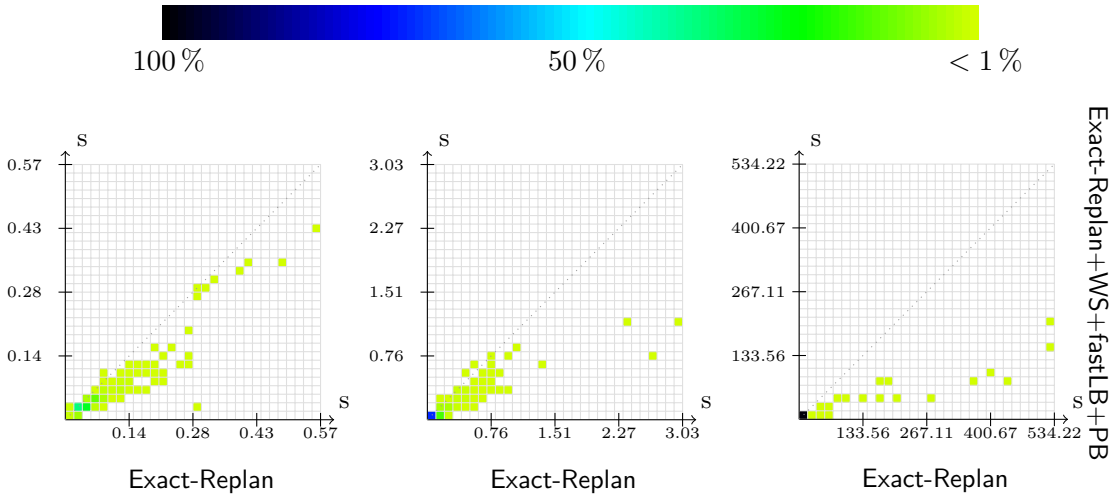


Abbildung B.42: Lunch Peak Verkehr, 1125, 1407 und 2026.

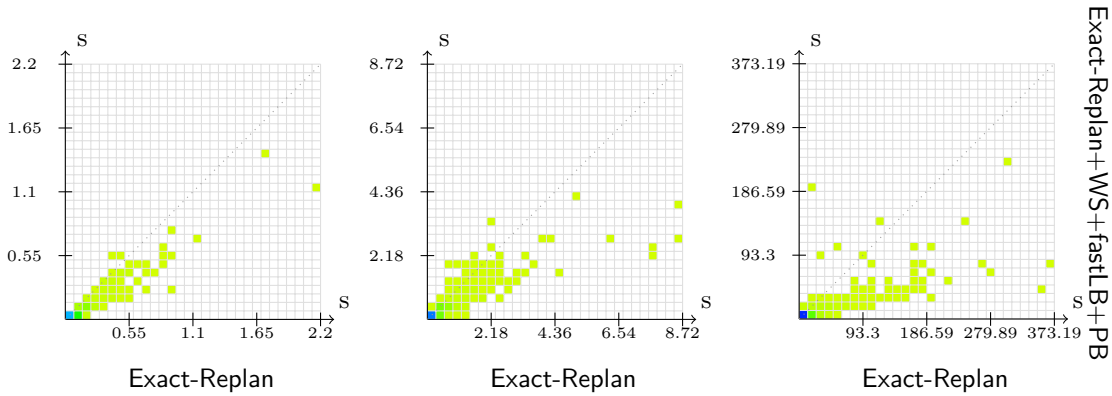


Abbildung B.43: Interfloor Verkehr, 1125, 1407 und 2026.

Gebäude B

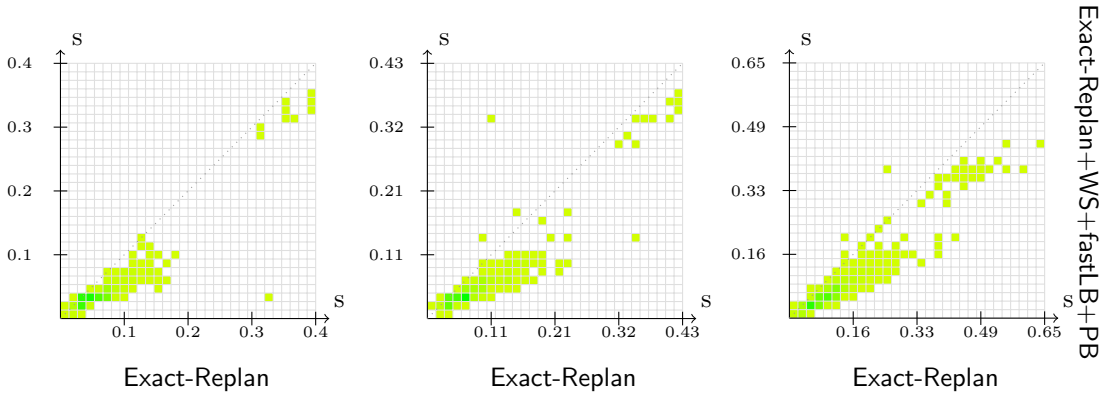


Abbildung B.44: Down Peak Verkehr, 2624, 3280 und 4723.

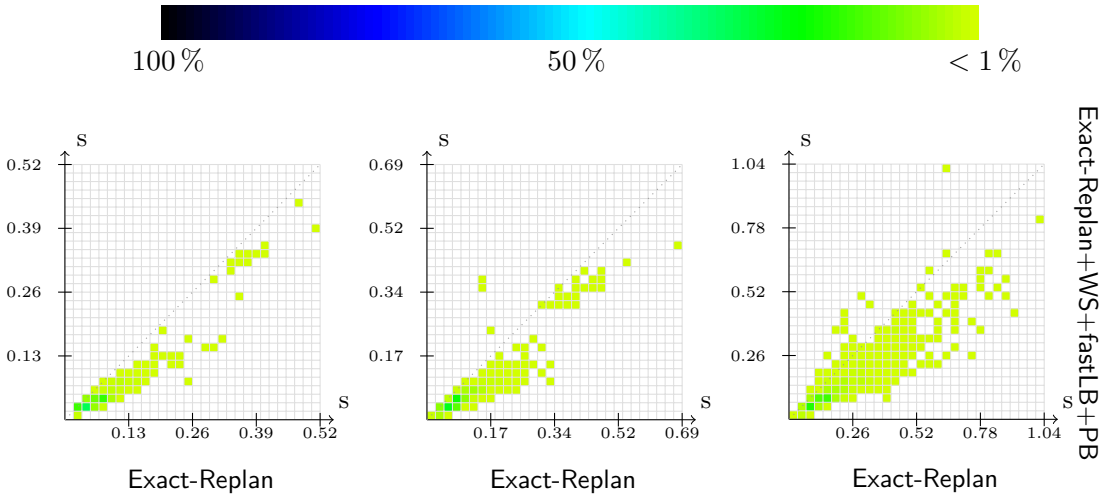


Abbildung B.45: Real Down Peak Verkehr, 2624, 3280 und 4723.

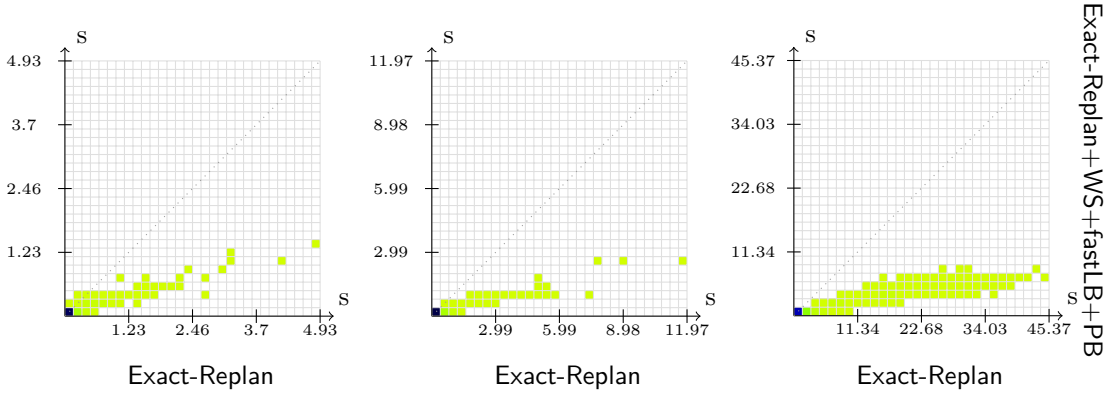


Abbildung B.46: Up Peak Verkehr, 2624, 3280 und 4723.

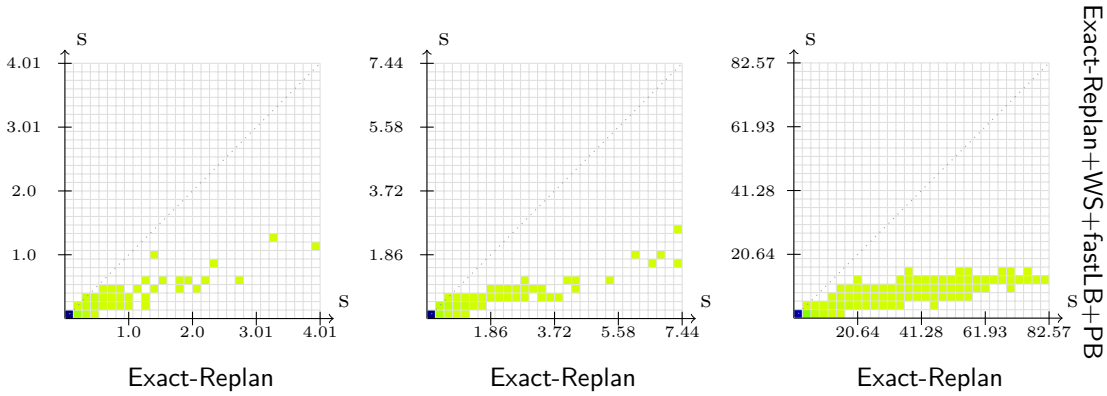


Abbildung B.47: Real Up Peak Verkehr, 2624, 3280 und 4723.

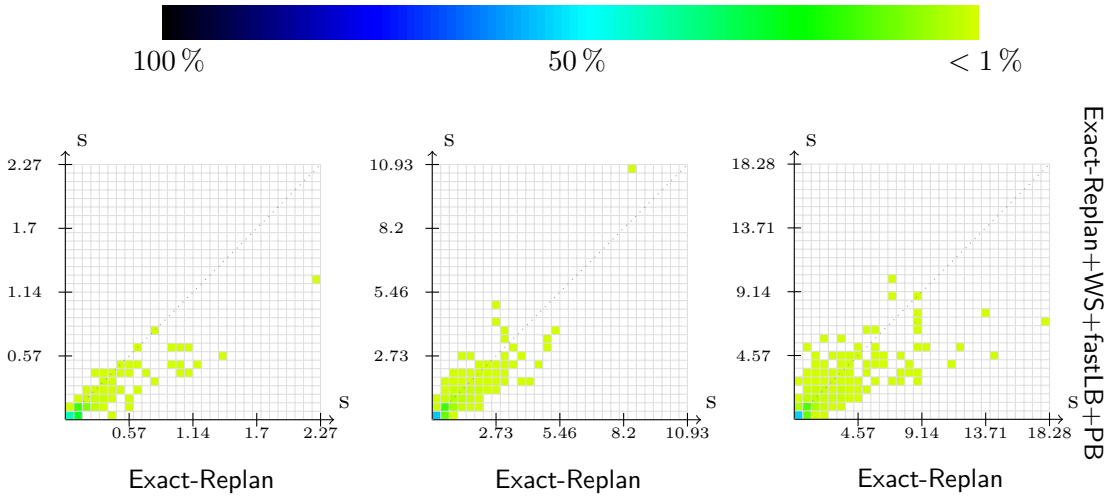


Abbildung B.48: Lunch Peak Verkehr, 2624, 3280 und 4723.

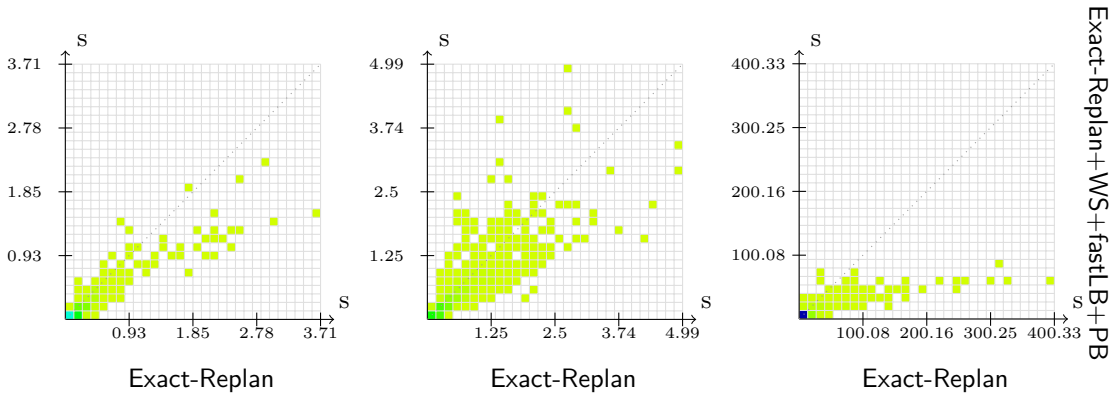


Abbildung B.49: Interfloor Verkehr, 2624, 3280 und 4723.

Gebäude C

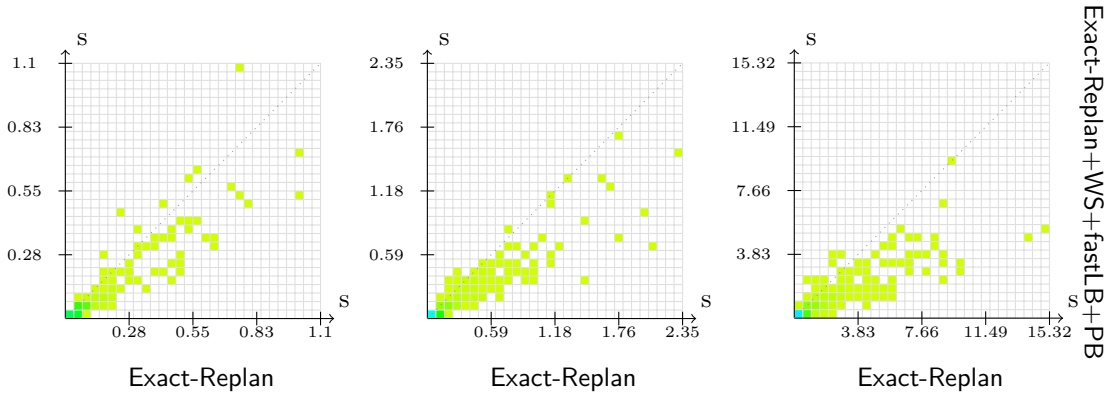


Abbildung B.50: Down Peak Verkehr, 1040, 1300 und 1872.

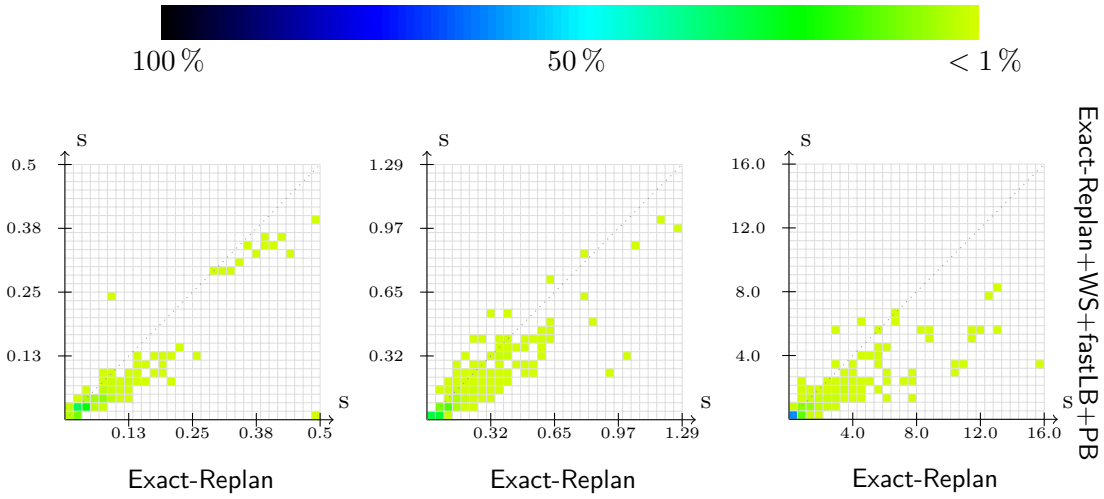


Abbildung B.51: Real Down Peak Verkehr, 1040, 1300 und 1872.

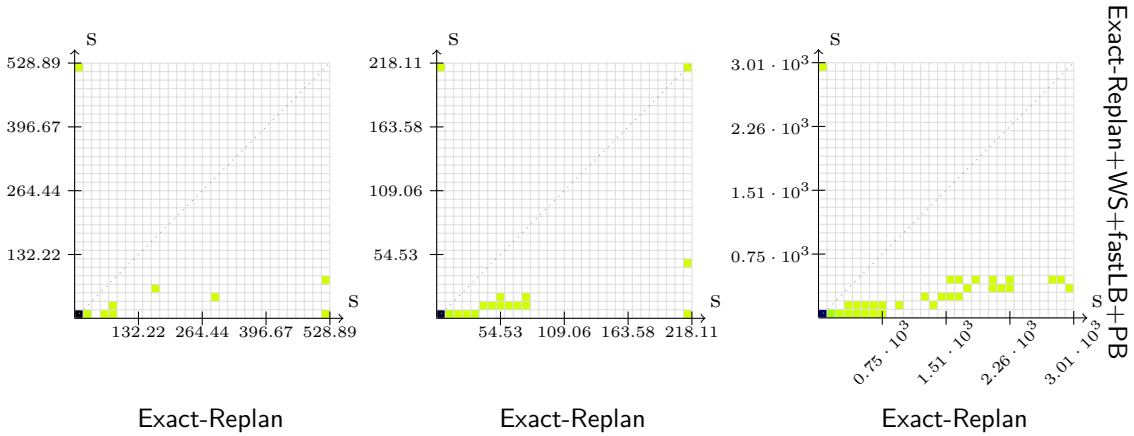


Abbildung B.52: Up Peak Verkehr, 1040, 1300 und 1872.

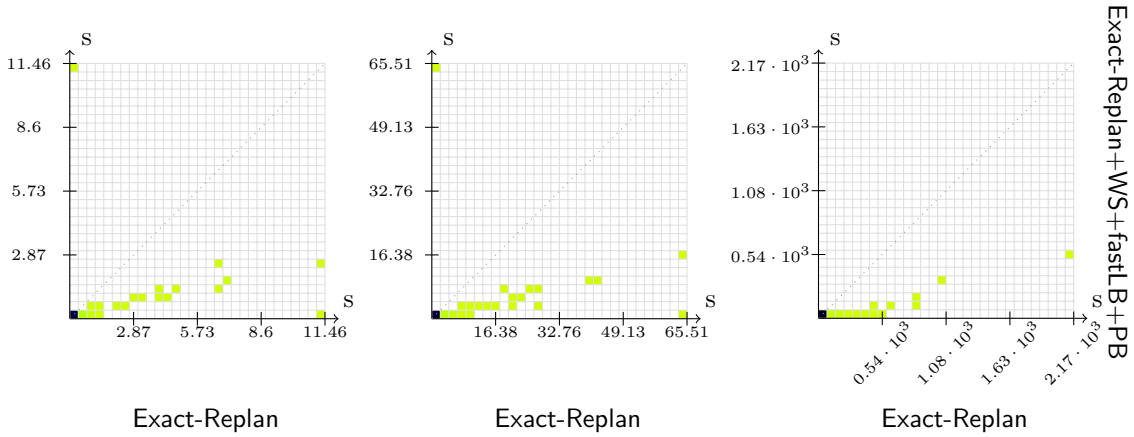


Abbildung B.53: Real Up Peak Verkehr, 1040, 1300 und 1872.

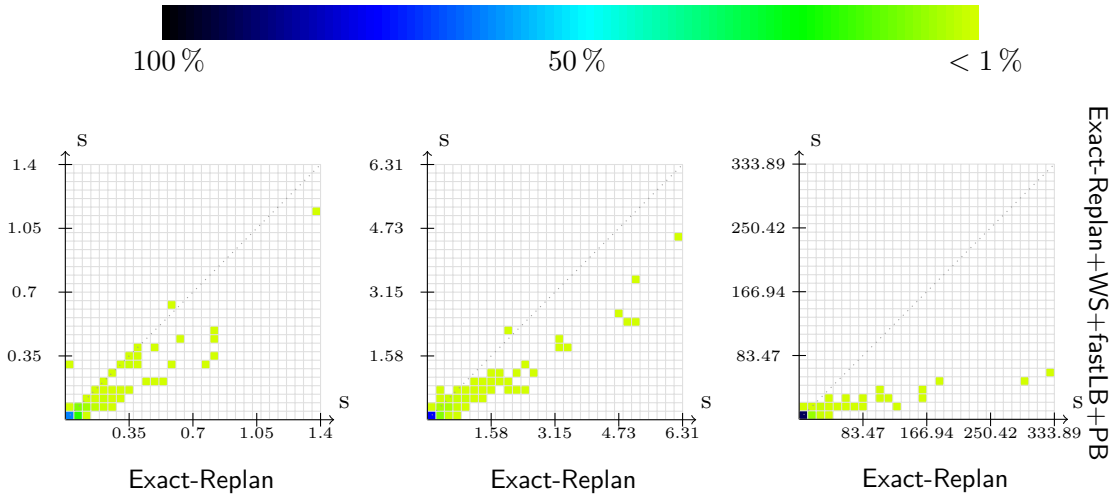


Abbildung B.54: Lunch Peak Verkehr, 1040, 1300 und 1872.

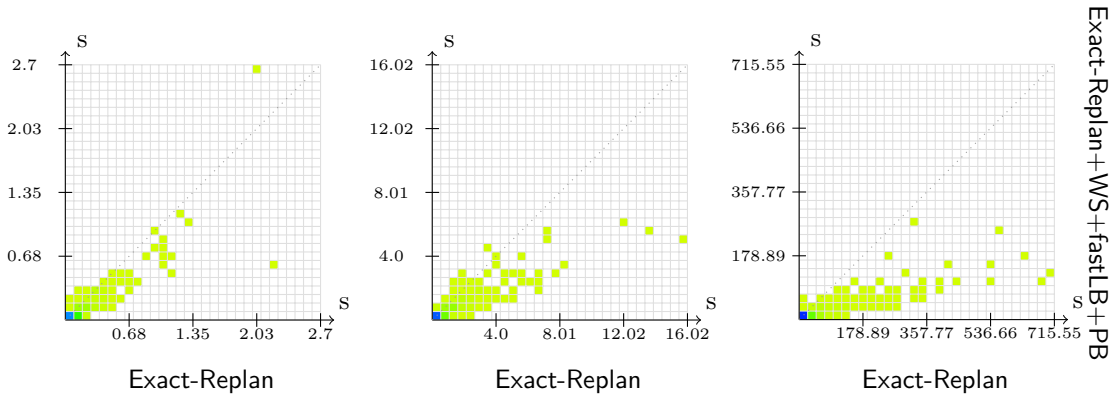


Abbildung B.55: Interfloor Verkehr, 1040, 1300 und 1872.

Generierte Knoten in den Schnappschussproblemen

In den folgenden Abbildungen ist die Anzahl der erzeugten Knoten, welche für die Lösung eines Schnappschussproblems mit dem Exact-Replan Algorithmus benötigt wird, der Anzahl der benötigten Knoten zur Lösung des gleichen Schnappschussproblems mit dem Exact-Replan+WS+fastLB+PB Algorithmus gegenübergestellt. Die horizontale Achse repräsentiert die Anzahl der durch den Exact-Replan erzeugten Knoten. Auf der vertikalen Achse kann die Anzahl der durch den Exact-Replan+WS+fastLB+PB erzeugten Knoten abgelesen werden. Aufgrund der großen Datenmenge wurde das Koordinatensystem gerastert. Die Färbung eines jeden Feldes $[x, x+\varepsilon] \times [y, y+\varepsilon]$ repräsentiert den prozentualen Anteil der Schnappschussprobleme, für deren Lösung der Exact-Replan zwischen x und $x+\varepsilon$ viele Knoten und der Exact-Replan+WS+fastLB+PB zwischen y und $y+\varepsilon$ viele Knoten erzeugt hat. Die gestrichelte Diagonale repräsentiert den Bereich in dem es keine signifikanten Unterschiede gibt. Die Diagramme sind wie folgt zu interpretieren: Je flacher der Verlauf der Punktwolke ist, desto größer sind die Einsparungen an erzeugten Knoten durch die Verwendung des Warmstarts.
In den folgenden Diagrammen ist zu erkennen, dass die Punktwolken zu sehr großen Teilen unterhalb der Geraden liegen. Des Weiteren wird deutlich, dass alle Felder,

welcher viele Schnappschussprobleme repräsentieren, unterhalb oder auf der Geraden liegen.

Gebäude A

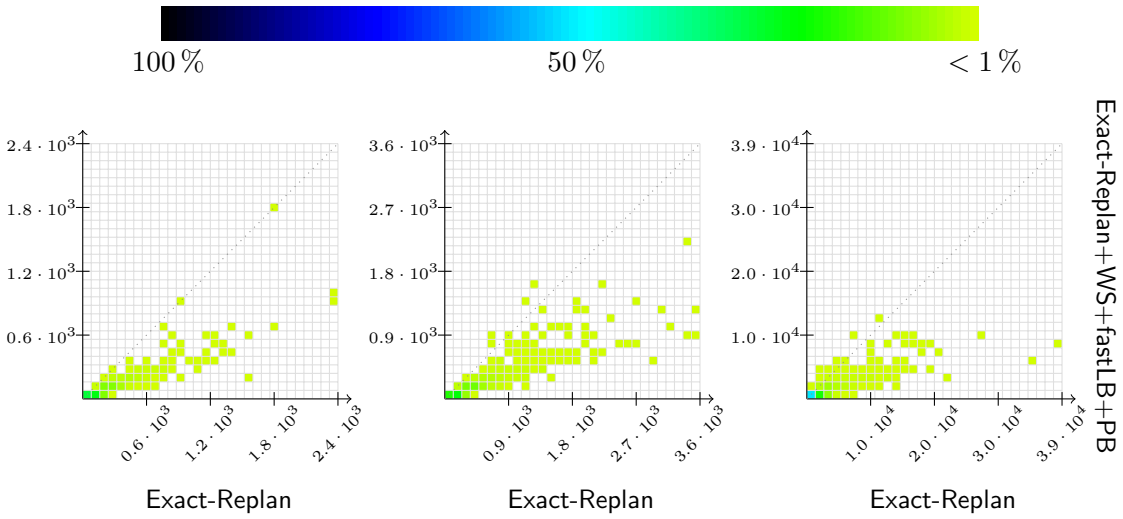


Abbildung B.56: Down Peak Verkehr, 1125, 1407 und 2026.

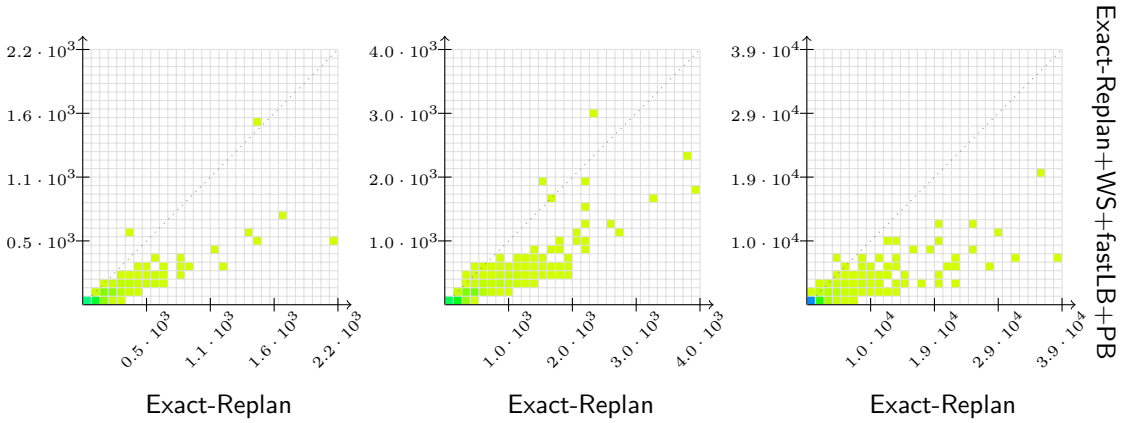


Abbildung B.57: Real Down Peak Verkehr, 1125, 1407 und 2026.

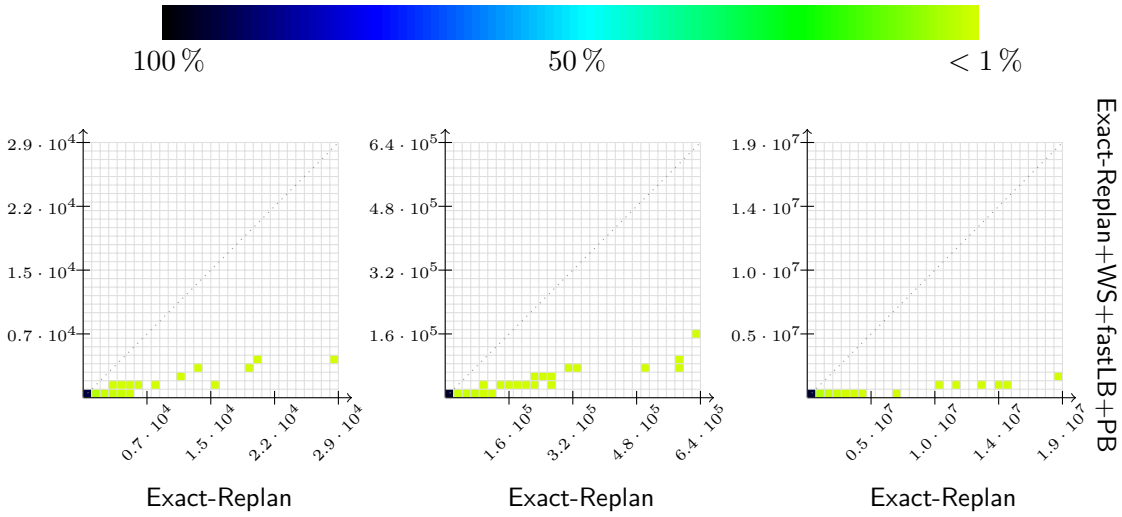


Abbildung B.58: Up Peak Verkehr, 1125, 1407 und 2026.

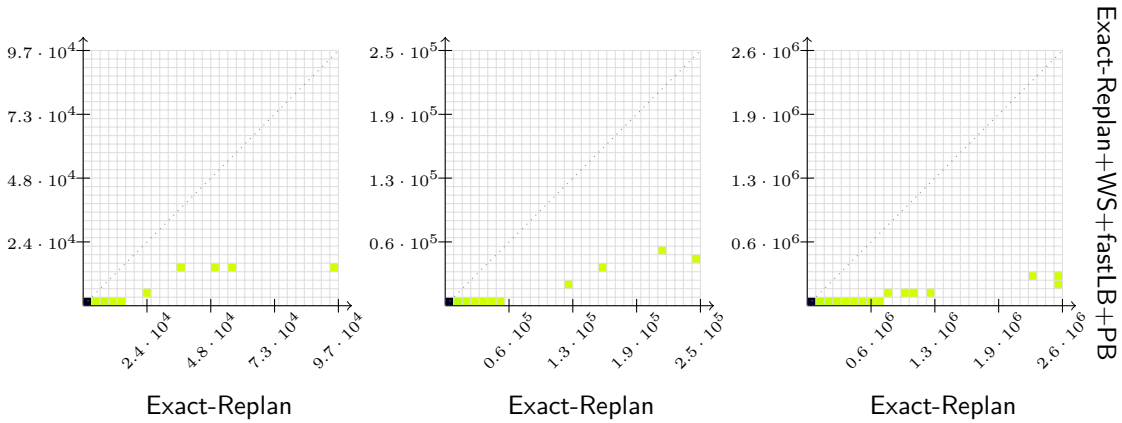


Abbildung B.59: Real Up Peak Verkehr, 1125, 1407 und 2026.

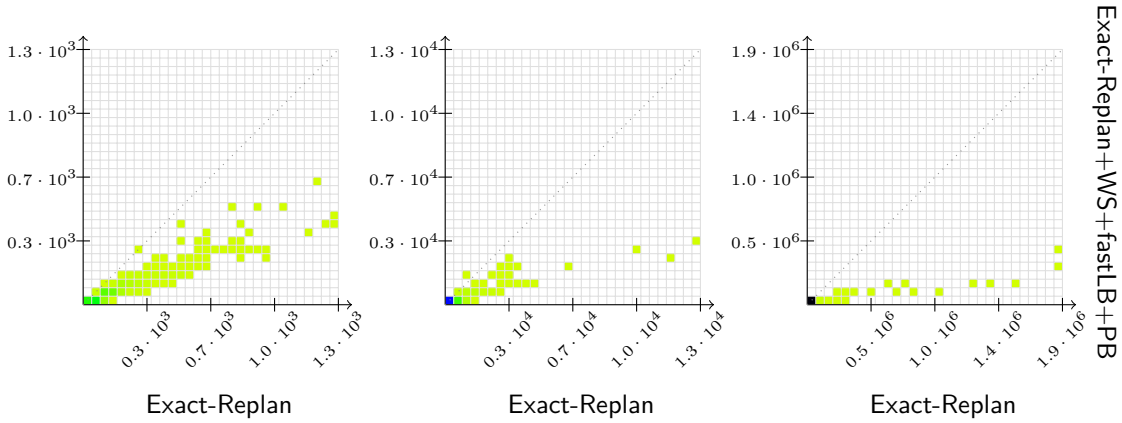


Abbildung B.60: Lunch Peak Verkehr, 1125, 1407 und 2026.

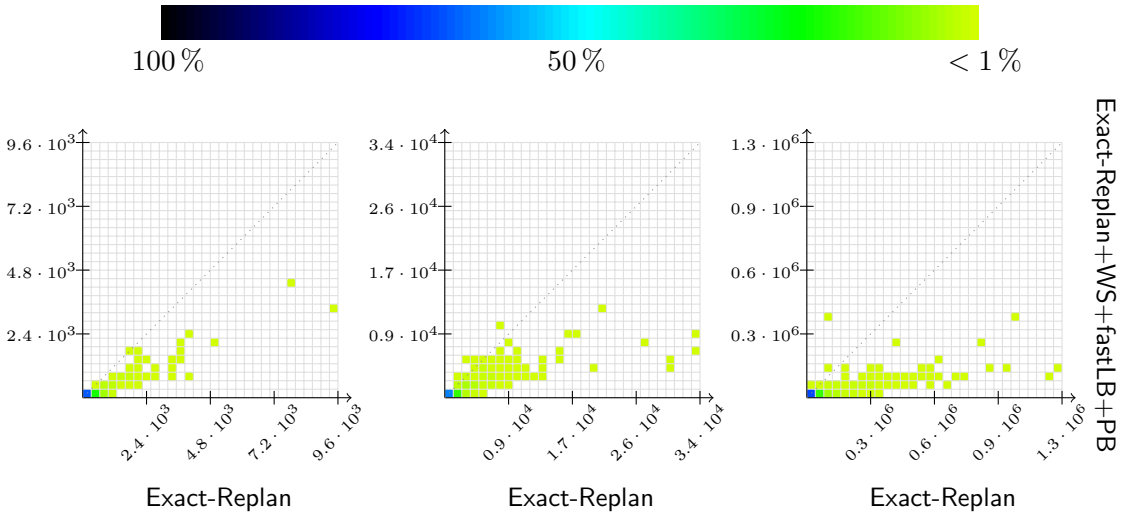


Abbildung B.61: Interfloor Verkehr, 1125, 1407 und 2026.

Gebäude B

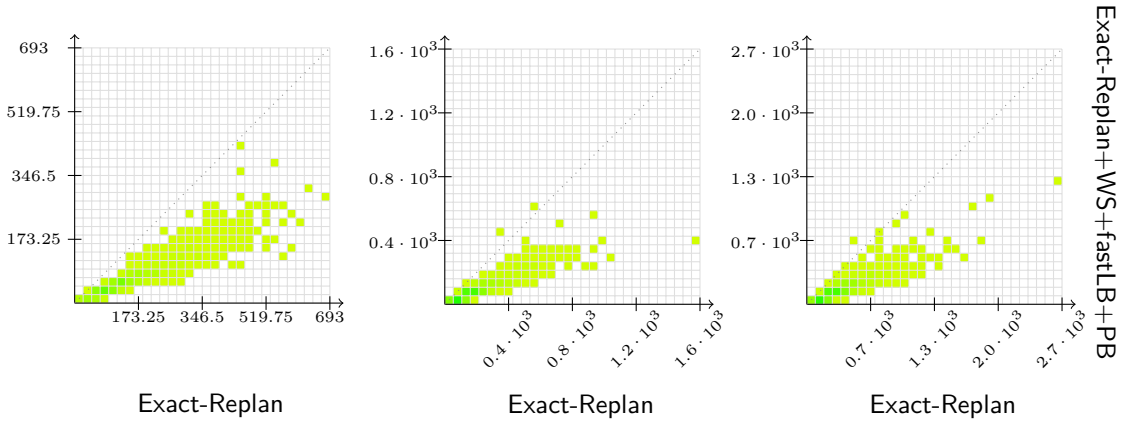


Abbildung B.62: Down Peak Verkehr, 2624, 3280 und 4723.

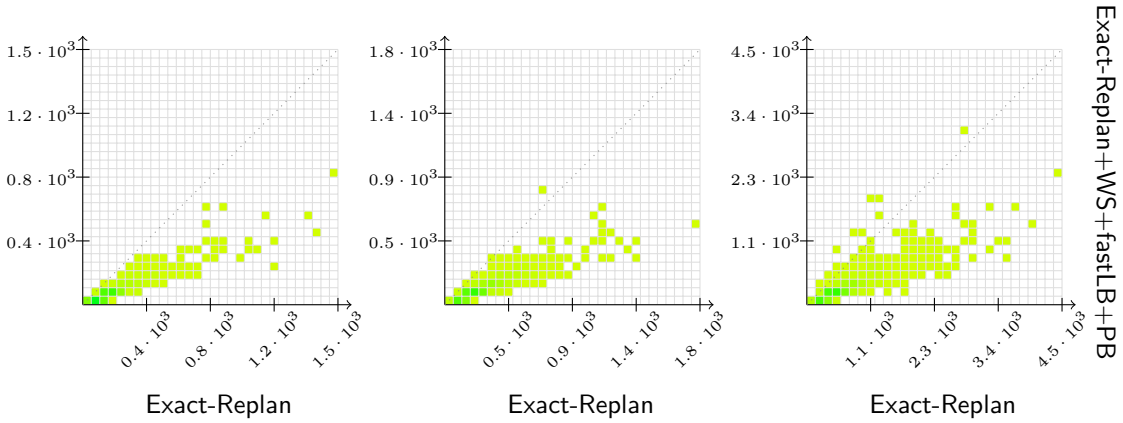


Abbildung B.63: Real Down Peak Verkehr, 2624, 3280 und 4723.

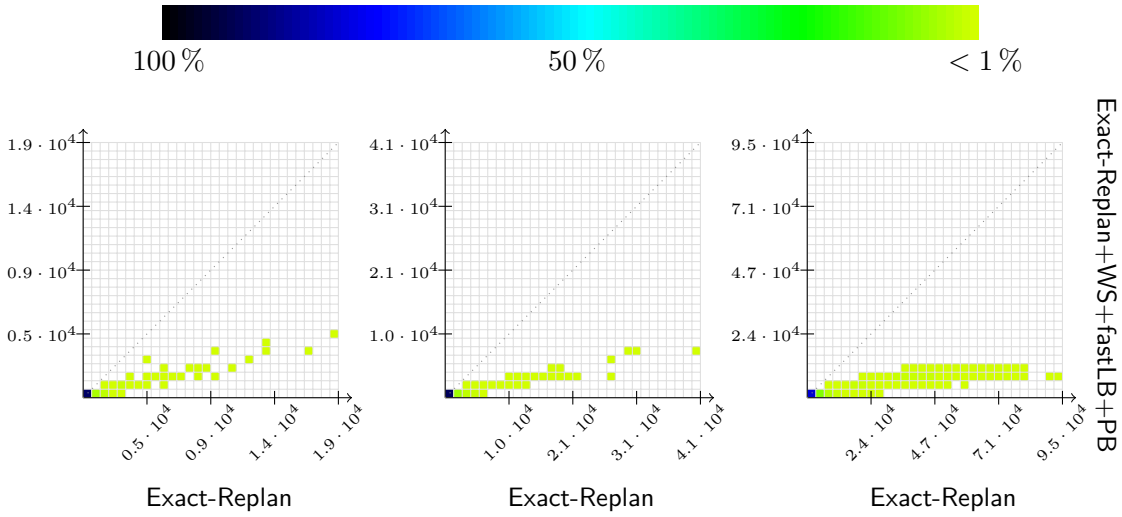


Abbildung B.64: Up Peak Verkehr, 2624, 3280 und 4723.

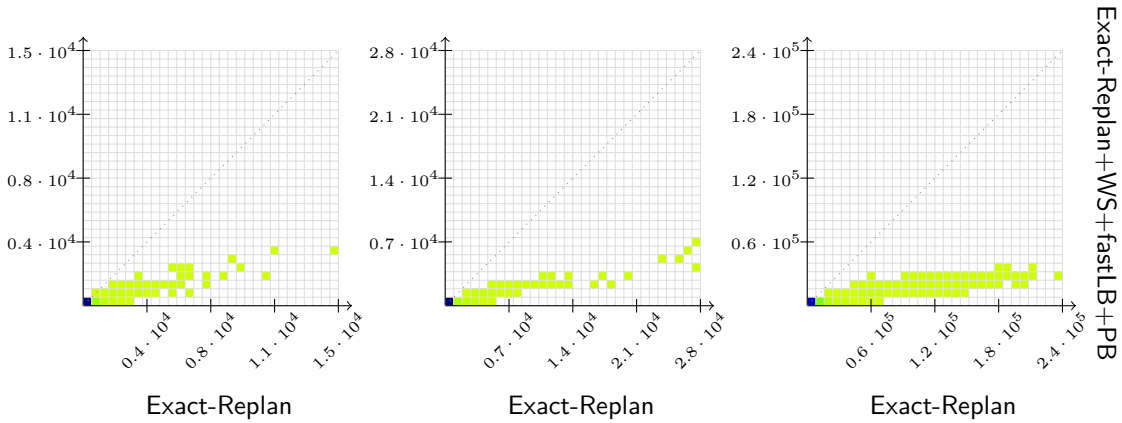


Abbildung B.65: Real Up Peak Verkehr, 2624, 3280 und 4723.

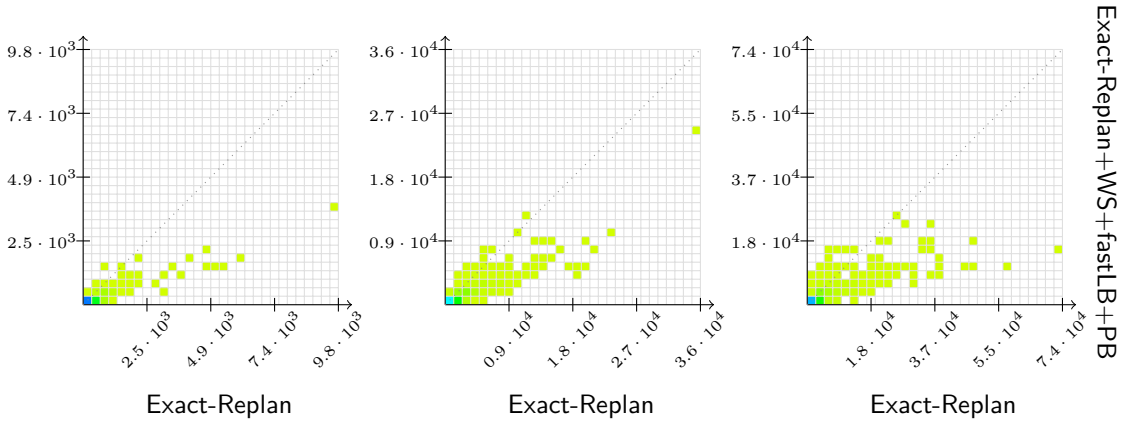


Abbildung B.66: Lunch Peak Verkehr, 2624, 3280 und 4723.

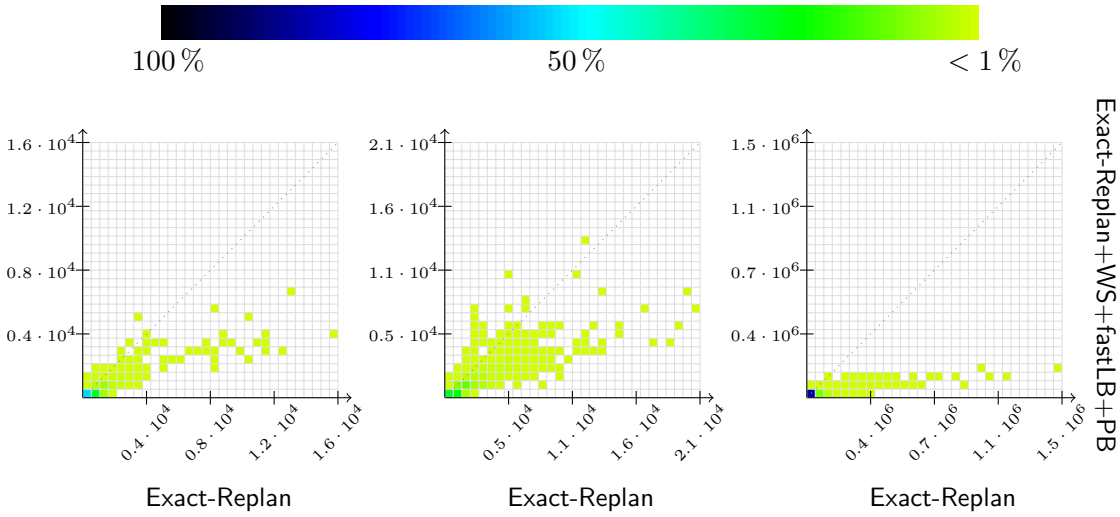


Abbildung B.67: Interfloor Verkehr, 2624, 3280 und 4723.

Gebäude C

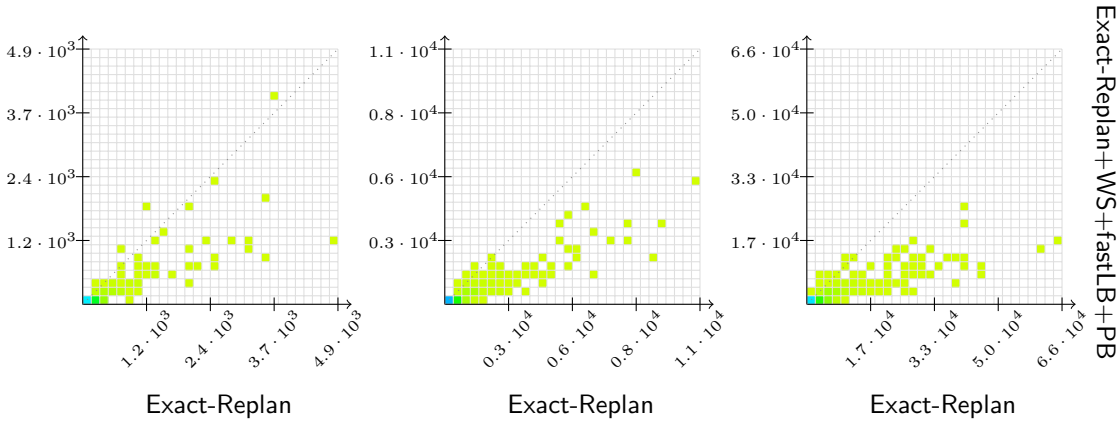


Abbildung B.68: Down Peak Verkehr, 1040, 1300 und 1872.

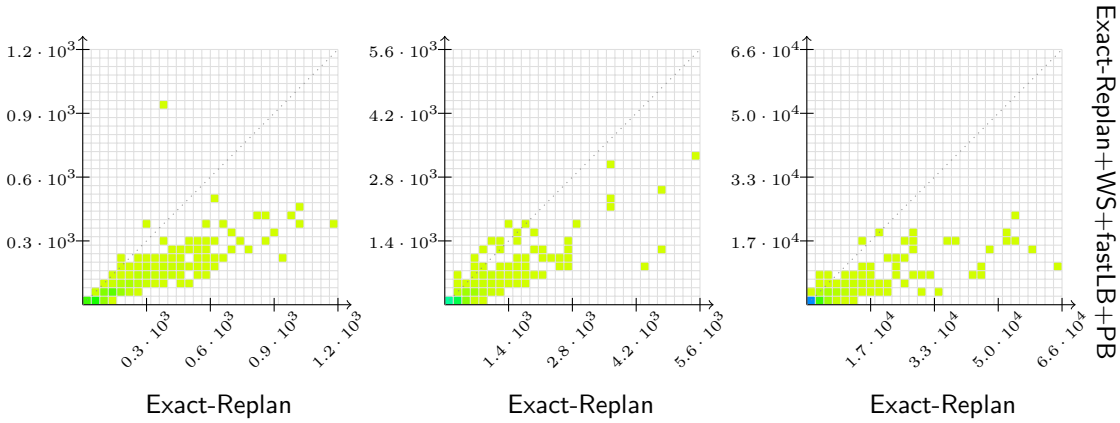


Abbildung B.69: Real Down Peak Verkehr, 1040, 1300 und 1872.

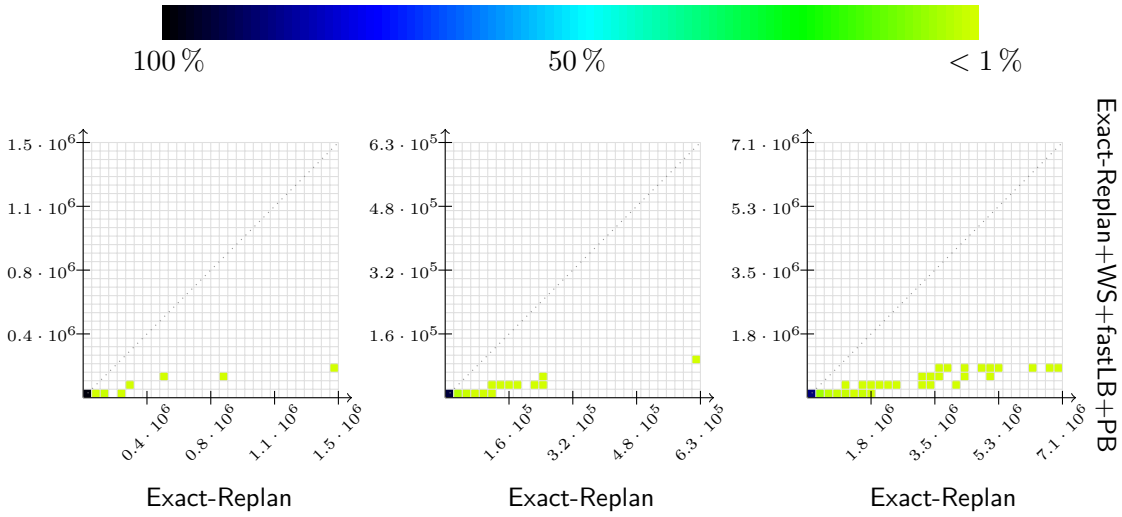


Abbildung B.70: Up Peak Verkehr, 1040, 1300 und 1872.

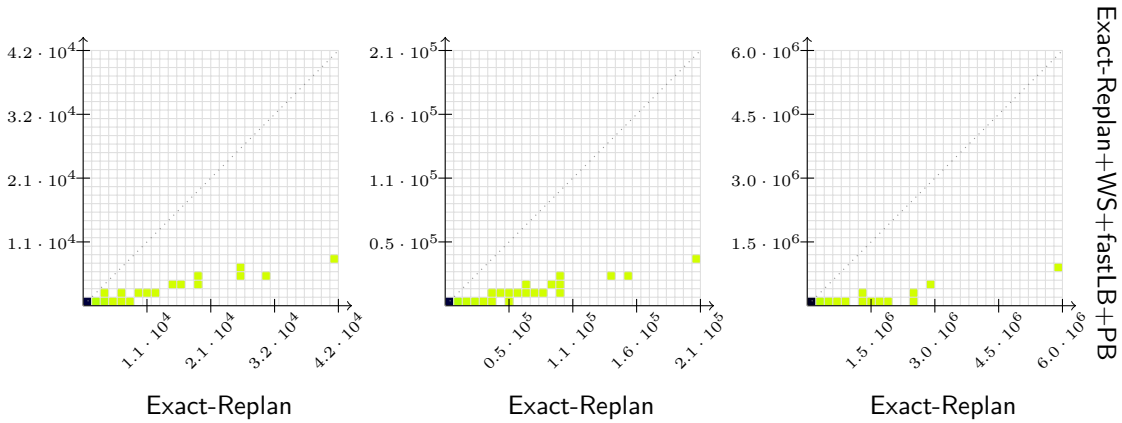


Abbildung B.71: Real Up Peak Verkehr, 1040, 1300 und 1872.

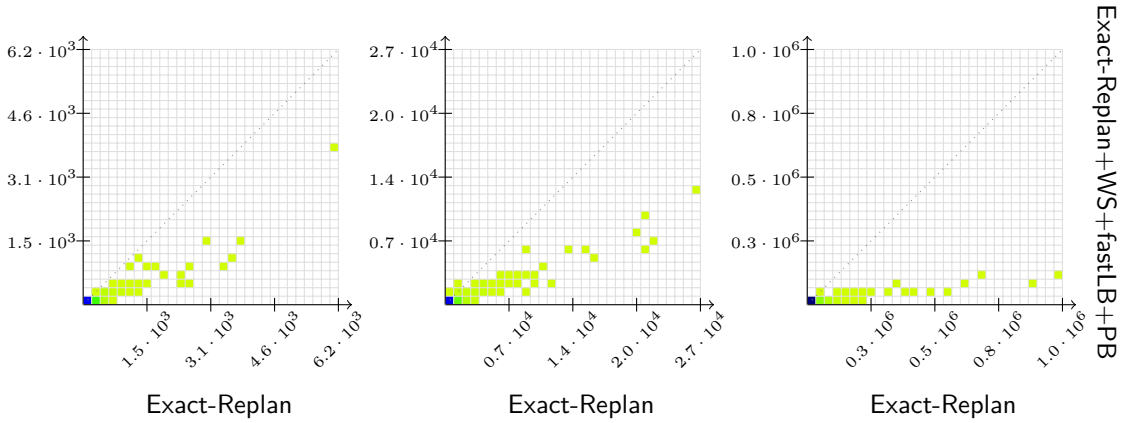


Abbildung B.72: Lunch Peak Verkehr, 1040, 1300 und 1872.

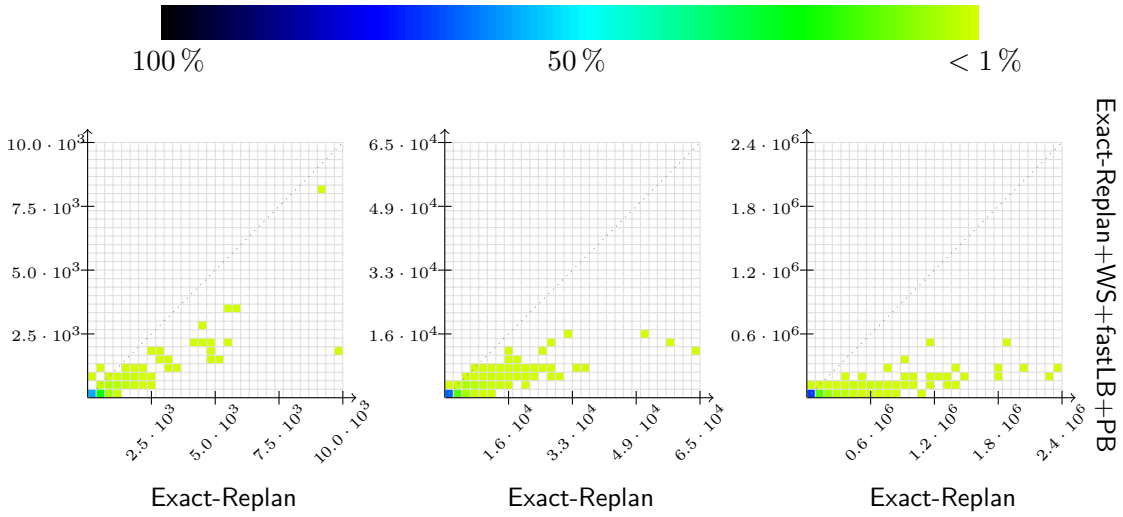


Abbildung B.73: Interfloor Verkehr, 1040, 1300 und 1872.

Literaturverzeichnis

[AKM04] Tobias Achterberg, Thorsten Koch, and Alexander Martin. *Branching Rules revisited.* ZIB Konrad-Zuse-zentrum für Informationstechnik, 2004.

[Bar02] Gina Carol Barney. *Elevator Traffic Handbook: Theory and Practice.* Taylor and FrancisTaylor and Francis, 2002.

[Chv83] Vasek Chvatal. *Linear Programming.* W.H. Freeman, 1983.

[DDS05] Guy Desauliers, Jacques Desrosiers, and Maurice M. Solomon. *Column Generation.* Springer, 2005.

[Hil10] Benjamin Hiller. *Online Optimization: Probabilistic Analysis and Algorithm Engineering.* TU Berlin, 2010.

[Las70] Leon S. Lasdon. *Optimization Theory for Large Systems.* Macmillan, 1970.

[Ltd] Peters Research Ltd. `https://www.peters-research.com/`.

[TUA05] Shunji Tanaka, Yukihiro Uraguchi, and Mituhiko Araki. *Dynamic optimization of the operation of single-car elevator systems with destination hall call registration: Part I. Formulation and simulations.* European J. Oper. Res., 2005.

[Wau07] Veronica Waue. *Entwicklung von Software zur Lösung von gemischtganzzahligen Optimierungsmodellen mit einem Branch and Cut Ansatz.* FU Berlin, 2007.