Masterarbeit

RWTH Aachen

Die Auswirkungen verschiedener Preprocessing-Routinen und Branch-And-Cut-Frameworks auf das Lösungsverhalten der Time Bucket Formulation des Travelling Salesman Problems mit Zeitfenstern

Zur Erlangung des akademischen Grades eines
Master of Science in Business Administration

Vorgelegt von:
Stephan M.F. Beckhäuser
Matrikelnummer:
266643
Vorgelegt an der:
Rheinisch-Westfälischen
Technischen Hochschule Aachen
Vorgelegt am:
14.Juni 2013

Vorgelegt beim:
Lehrstuhl für
Operations Research
Betreuer:
Prof. Dr. Marco E. Lübbecke
Beratungsassistent:
M.Sc. Christian Puchert

Inhaltsverzeichnis

1. Einführung

In dieser Arbeit werden das *Problem des Handelsreisenden mit Zeitfenstern*, oder auch *Travelling Salesman Problem With Time Windows* (*TSPTW*), sowie verschiedenen, dem Gebiet des (Mixed-)Integer Programming entstammende Formulierungsvarianten für selbiges, vorgestellt. Im Rahmen dessen werden Techniken des allgemeinen Preprocessings sowie ein spezieller, LP-basierter Lösungsansatz ausführlicher diskutiert.

1.1. Schwerpunkte der Arbeit

Namentlich wird insbesondere auf die von *Dash et al.* [5] eingeführte *TSPTW-Time Bucket Relaxation* und ihre Einbettung in ein spezifisches Branch-And-Cut-Framework (B&C-Framework), um selbige in die *TSPTW*-Branch-And-Cut-*Time Bucket Formulation* zu überführen, eingegangen. Im Rahmen dessen werden sodann, basierend auf diversen Literaturinstanzen von *Ascheuer* [25] sowie *Dumas et al.* [28], unterschiedliche Kombinationen möglicher Preprocessing-Routinen, und deren Auswirkungen auf das Lösungsverhalten der in Rede stehenden Formulierung untersucht. In gleicher Weise sollen im Zusammenhang mit dem Branch-And-Cut-Framework Experimente zu den Auswirkungen verschiedener Schnittebenen-Varianten, Branching-Regeln und Primalheuristiken durchgeführt werden, wobei sowohl die von *Dash et al.* [5] vorgestellten Elemente, als auch neue, denkbare Ergänzungen derselben, eingeführt und betrachtet werden. Besonders im Kontext möglicher Schnittebenen- und Primalheuristik-Erweiterungen werden Ergebnisse präsentiert, welche bemerkenswerte Verbesserungspotentiale aufzeigen.

1.2. Aufbau der Arbeit

In Kapitel 2 werden eine verbale und eine formale Definition des *TSPTW* geliefert. Es folgen eine komplexitätstheoretische Untersuchung sowie eine Einordnung des Problems in Literatur und Praxis. Das Kapitel schließt mit einer Präsentation drei verschiedener Integer-Linear-Programming-Formulierungen des *TSPTW*, darunter die im Rahmen dieser Arbeit ausführlicher zu untersuchende *TSPTW-TBF*, beziehungsweise Branch-And-Cut-Framework unterstützte *TSPTW-TBR*. In Kapitel 3 wird sodann das allgemeine Preprocessing, welches zur Vereinfachung der betrachteten *TSPTW*-Instanzen verwendet werden soll, in seinen einzelnen Elementen vorgestellt. Im Anschluss beschreibt Kapitel 4 ausführlich das Branch-And-Cut-Framework, in welches die *TSPTW-TBR* eingebettet wird um in die Branch-And-Cut-Framework-*Time Bucket Formulation* überführt zu werden. Überdies werden die einzelnen Elemente der Initialen *TSPTW-TBR* -Preparation vorgestellt. Kapitel 5 präsentiert schließlich die Resultate der in Abschnitt 1.1 aufgeführten Experimente, denen die Branch-And-Cut-Framework-*Time Bucket Formulation* im Rahmen dieser Arbeit unterzogen wurde. Das Kapitel 6 schließt sodann mit einer Zusammenfassung der Ergebnisse sowie einer kritischen Würdigung des betrachteten Ansatzes.

2. TSPTW

In diesem Kapitel soll das *Problem des Handelsreisenden mit Zeitfenstern* einführend erläutert und definiert werden. Überdies werden seine Komplexität und Schwere untersucht. Sodann folgen Ausführungen zu seiner Literatur und seinen Anwendungen in der Praxis. Schließlich werden drei verschiedene Integer-Linear-Programming Formulations (ILP Formulations oder auch ILP-Formulierungen) des Problems vorgestellt.

2.1. Definitionen

Es sei angemerkt, dass sich in der Literatur keine einheitliche Definition des Problems findet, was nicht zuletzt der Tatsache geschuldet ist, dass verschiedene Praxisanwendungen unterschiedliche Subsumtionen der Problematik auf der Modellebene erfordern. Im Rahmen dieser Arbeit soll sich die Definition des Problems daher an den Basisliteratur-Werken orientieren, welchen die verwendeten Testinstanzen entstammen.

2.1.1. Verbale Definition

Das *TSPTW* ist das Problem des Bereisens einer Menge an gegebenen Städten, wobei jede Stadt nur genau einmal besucht werden darf, und dies in einem jeweils vorgegebenen Zeitfenster geschehen muss. Hierbei ist in aller Regel entweder die gesamte Reisezeit, oder die Ankunftszeit bei der letzten Stadt zu minimieren.[1] In einem Großteil der gängigen Literatur-Instanzen fallen die Reisezeit und die Reisekosten, welche der Verbindung von zwei Städten zugeordnet werden können, zusammen. In selteneren Fällen, wo dies nicht so ist, wird auch teilweise eine Hybrid-Kostenfunktion, zusammengesetzt aus der gesamten Reisezeit und den gesamten Reisekosten, minimiert.[2]

2.1.2. Formale Definition

Gegeben sei ein Graph $G = (V, A)$ mit $|V| = n$, wobei jeder Knoten $i \in V$ eine Stadt repräsentiert, welche im Zeitfenster $W_i = [R_i, D_i]$ verlassen werden muss, die Ankunft kann hingegen auch vor dem Beginn des Zeitfensters erfolgen. R_i bezeichnet hierbei die sogenannte "*Release time* " und D_i die sogenannte "*Deadline* " des Knotens i (aus Gründen der Übersicht sei $i := \{i\}$ eine vereinfachte Darstellung der Mengenschreibweise für einen einzelnen Knoten). Ferner werden jeder Kante $(i, j) \in A$ die Reisekosten $c_{i,j}$ und die Reisezeit $\theta_{i,j}$ zugeordnet. Wie bereits angedeutet, gilt auf den meisten Anwendungs-Instanzen, so auch jenen, welche in dieser Arbeit untersucht werden sollen, $\theta_{i,j} = c_{i,j}$. Um dem Allgemeinfall Rechnung zu tragen, soll im theoretischen Teil dieser Arbeit die in Rede stehende formale Distinguierung jedoch beibehalten werden. Ferner wird angenommen, dass sämtliche Daten integral sind, und alle Kantenkosten $c_{i,k}, c_{k,j}, c_{i,j}$, $(i, k), (k, j), (i, j) \in A$, der Dreiecksungleichung unterliegend, also metrisch sind.

Es existieren sodann die zwei Knoten $p, q \in V$, denen eine Sonderstellung zukommt. Jede zulässige Tour startet in Knoten p, der somit einmal verlassen und keinmal erreicht wird,

[1] vgl. hierzu [5]
[2] siehe hierzu beispielsweise [4]

und endet in Knoten q, der somit einmal erreicht und keinmal verlassen wird. Folgerichtig ist im Fall des *TSPTW*, im Gegensatz zum Fall des klassischen *Travelling Salesman Problems* (*TSP*), namentlich dem Problem des Auffindens eines kostennminimalen Hamilton-Kreises in einem gewichteten Graphen, stattdessen ein Hamilton-Weg in G gesucht. Zwar können p und q, je nach Instanz, auch de facto ein- und dieselbe Stadt, beziehungsweise ein- und dasselbe Depot darstellen[3], formal und im Modell sollen die beiden Knoten in dieser Arbeit jedoch stets unterschieden werden. Überdies finden sich in der Literatur ebenfalls vereinzelte *TSPTW*-Instanzen, die a priori vorgegebener Start- und Endknoten zur Gänze entbehren.[4] Diese können jedoch durch das Einführen von zwei künstlichen Knoten p und q mit den Zeitfenstern $W_p = [argmin_{i \in V}\{R_i\} - 1, argmin_{i \in V}\{R_i\} - 1]$ und $W_q = [argmax_{i \in V}\{D_i\} + 1, argmax_{i \in V}\{D_i\} + 1]$, sowie den künstlichen Kanten (p, i) und (i, q) mit $c_{p,i} = c_{i,q} = 0$ und $\theta_{p,i} = \theta_{i,q} = 1$ für alle $i \in V$ in die oben beschriebenen Formulierungsparadigmen überführt werden, da die ergänzten Knoten aufgrund der Parameterbelegungen augenscheinlich zielfunktions- und somit entscheidungsneutral sind, sich in jeder zulässigen Lösung aber zugleich, wegen W_p und W_q, erzwungener Weise als Start- (p) und Endknoten (q) einstellen werden.[5] Insbesondere gilt stets $(q, p) \notin A$.

Sodann, gegeben der Knoten $k \in V \setminus \{q\}$ ist der direkte Vorgängerknoten von Knoten i in der Tour, sollen die entsprechende Ankunfts- und Startzeit bei i wie folgt definiert werden:

$$Ankunftszeit_i = Startzeit_k + \theta_{k,i} \quad \forall\ i \in V \setminus \{p\} \tag{2.1}$$

$$Startzeit_i = \max\{Startzeit_k + \theta_{k,i}, R_i\} \quad \forall\ i \in V \setminus \{p, q\} \tag{2.2}$$

Im Gegensatz zur Lesart in vereinzelten Literaturvorlagen[6] sollen in dieser Arbeit die in den Gleichungen (2.1) und (2.2) eingeführten Begriffe strikt getrennt werden. Präzise fallen, gemäß (2.2), Ankunfts- und Starzeit eines Knotens $i \in V \setminus \{p\}$ genau dann auseinander, wenn $Startzeit_k + \theta_{k,i} < R_i$, also die Ankunftszeit bei i vor dem Beginn des Zeitfensters von i liegt.

Ferner seien die Sonderfälle

$$Ankunftszeit_p = Startzeit_p = R_p \tag{2.3}$$

$$Startzeit_q = NULL \tag{2.4}$$

definiert. Die Deklaration (2.3) ist augenscheinlich legitim, da es nicht sinnvoll wäre, den Startknoten p später als frühestmöglich zu verlassen. Zuweisung (2.4) folgt unmittelbar aus der Tatsache, dass der Zielnoten q nicht mehr verlassen wird.

Basierend auf dieser Nomenklatur soll unter einer zulässigen *TSPTW*-Tour nun ein Hamilton-Weg von p nach q durch G verstanden werden, auf welchem für alle $i \in V \setminus \{q\}$ die $Startzeit_i$ im entsprechenden Zeitfenster W_i des jeweiligen Knotens liegt.

Eine optimale *TSPTW*-Tour soll überdies eine zulässige *TSPTW*-Tour $\mathcal{T} \subseteq A$ sein, welche die Zielfunktion $\sum_{\mathcal{T}} c_{i,j}$, also die gesamten Reisekosten, minimiert.

Ist der zugrunde liegende Graph G gerichtet, so spricht man auch vom *Gerichteten-TSPTW* oder *Arrowed-TSPTW* (*ATSPTW*). Da ein etwaiger ungerichteter Graph G_{sym} jedoch ohne Weiteres, durch eine Dopplung der Kanten, in einen äquivalenten, gerichteten

[3] vgl. hierzu beispielsweise die Instanzen von *Dumas et al.* [28]
[4] vgl. hierzu [5]
[5] vgl. hierzu [5]
[6] vgl. hierzu beispielsweise [4]

Graphen G überführt werden kann[7], sei in dieser Arbeit grundsätzlich vom gerichteten *TSPTW* gesprochen und auf eine derartige Distinguierung verzichtet.[8]

2.2. Komplexität und Schwere

Im Folgenden sollen, basierend auf Reduktionsketten, grob die Schwere, sowie die daraus resultierenden Indikationen für die Komplexität[9] des *TSPTW* untersucht, und den entsprechenden Eigenschaften des klassischen TSP gegenüber gestellt werden.

Für das Problem der Erfüllbarkeit von aussagenlogischen Formeln in konjunktiver Normalform mit endlich vielen Literalen pro Klausel, namentlich das *Satisfiability-Problem* (*SAT*), konnte von *Cook* [49] als erstes Problem in der Geschichte der theoretischen Informatik nachgewiesen werden, dass es $\mathcal{NP}$-vollständig, und damit insbesondere auch $\mathcal{NP}$-schwer ist. Dies bedeutet in praxisrelevanter Hinsicht unter anderem, dass für dieses Problem kein effizienter Algorithmus (namentlich ein Algorithmus, dessen Laufzeitkomplexität, in Abhängigkeit aller Eingabegrößen der Instanz-Attribute, durch ein Polynom beschränkt ist - im Folgenden auch als "polynomiell beschränkter Algorithmus" bezeichnet) gefunden werden kann, es sei denn $\mathcal{P}$=$\mathcal{NP}$, was bis heute zwar weder gezeigt noch widerlegt werden konnte, jedoch im generellen Meinungsbild der modernen Literatur zur theoretischen Informatik hinlänglich bezweifelt wird.[10] Daher soll den Folgeargumentationen die Behauptung $\mathcal{P}\neq\mathcal{NP}$ zugrunde gelegt werden.

Ferner gilt für *SAT*, sowie das Entscheidungsproblem der Erfüllbarkeit von aussagenlogischen Formeln in konjunktiver Normalform mit höchstens drei Literalen pro Klausel, namentlich das *3-Satisfiability-Problem* (*3-SAT*), und das Entscheidungsproblem, ob ein gegebener Graph $G = (V, A)$ einen Hamiltonkreis enthält, namentlich das *Hamiltonian Circuit-Problem* (*HC*), der Zusammenhang $SAT \leq_P$ *3-SAT* $\leq_P$ *HC*[11], woraus unmittelbar die $\mathcal{NP}$-Schwere von *HC* folgt, da diese Reduktionskette impliziert, dass *HC* "mindestens so schwer" ist wie *SAT*. Basierend auf diesen Erkenntnissen lassen sich die $\mathcal{NP}$-Schwere des *TSP*, und daraufhin die $\mathcal{NP}$-Schwere des *TSPTW* nachweisen.

Satz 1. Aus der $\mathcal{NP}$-Schwere von *HC* ergibt sich die $\mathcal{NP}$-Schwere des *TSP*.

Beweis. Es wird gezeigt, dass $HC \leq_P TSP$, woraus die Behauptung folgt.

Gegeben sei eine Instanz des *Hamiltonian Circuit-Problems* auf einem beliebigen Graphen $G = (V, A)$. Konstruiere den Hilfsgraphen $G' = (V, A')$ mit $A' = \{(i, j) : i \in V, j \in V\}$ (Laufzeit-Komplexität: $O(|V|^2)$) und der Kanten-Gewichtungsfunktion $c : A' \to \mathbb{R}$ mit $c(e) = 0$ für $e \in A$ und $c(e) = 1$ sonst (Laufzeit-Komplexität: $O(|V|^2)$). Die Lösung der so resultierenden *TSP*-Instanz auf dem Hilfsgraphen G' hat genau dann den Zielfunktionswert $ZF = 0$, wenn der Ursprungsgraph G hamiltonisch war, da in diesem Fall jeder Hamiltonweg in G einer optimalen *TSP*-Tour in G' entspricht. G war indes genau dann nicht hamiltonisch, wenn ein Zielfunktionswert $ZF > 0$ resultiert, da hier mindestens eine Kante $(i, j) \notin A$ mit $c(i, j) = 1$ in die Lösung aufgenommen werden muss, um die *TSP*-Tour zu schließen. □

Satz 2. Aus der $\mathcal{NP}$-Schwere des *TSP* ergibt sich die $\mathcal{NP}$-Schwere des *TSPTW*.

[7] siehe hierzu beispielsweise die Instanzen von [28], welche von Natur aus zunächst symmetrisch sind
[8] vgl. zu diesem Abschnitt 2.1.2 auch [5]
[9] gegeben $\mathcal{P}\neq\mathcal{NP}$
[10] siehe hierzu beispielsweise [14]
[11] siehe hierzu beispielsweise [64]

Beweis. Es wird gezeigt, dass $TSP \leq_P TSPTW$, woraus die Behauptung folgt.

Gegeben sei eine Instanz des *Travelling Salesman Problems* auf einem beliebigen Graphen $G = (V, A)$ mit der Kantengewichtungs-Funktion $c : A \to \mathbb{R}$. Zerlege einen beliebigen Knoten $i_0 \in V$ in zwei Hilfsknoten p und q (Laufzeit-Komplexität: $O(1)$). Setze $\theta_{i,j} = c_{i,j}$ (Laufzeit-Komplexität: $O(|V|^2)$). Ergänze für jeden Knoten $i \in V$ ein Zeitfenster $W_i = [0, +\infty]$ (Laufzeit-Komplexität: $O(|V|)$). Mit p als Startknoten und q als Endknoten kodiert die Knoten-Permutation, welche der optimalen Lösung der so resultierenden $TSPTW$-Instanz zugeordnet werden kann, bei Entfallen des Knotens q (Laufzeit-Komplexität: $O(1)$) und der Rücküberführung von p in den Ursprungsknoten i_0 (Laufzeit-Komplexität: $O(1)$) eine optimale Lösung der ursprünglichen TSP-Instanz. □

Folglich ist das $TSPTW$ "mindestens so schwer" wie das $\mathcal{NP}$-schwere TSP. Somit ist es nicht möglich, für das $TSPTW$ einen polinomiell beschränkten Algorithmus anzugeben.

Überdies konnte *Savelsbergh* [41] zeigen, dass für jede Instanz des $\mathcal{NP}$-vollständigen[12] Entscheidungsproblems, ob ein Set bestehend aus $3m$ ($m \in \mathbb{N}$) Integer-Zahlen in dreielementige Subsets zerlegt werden kann, sodass sämtliche dieser Subsets im Hinblick auf die Summe der enthaltenen drei Zahlen denselben Wert aufweisen, namentlich das *3-Partitioning-Problem* (*3-PAR*), eine in polynomieller Zeit konstruierbare $TSPTW$-Instanz mit metrischer Distanzmatrix existiert, sodass die ursprüngliche *3-PAR*-Instanz genau dann lösbar ist, wenn die konstruierte $TSPTW$-Instanz eine zulässige Lösung besitzt. Daraus folgt unmittelbar, dass das Entscheidungsproblem, ob eine metrische $TSPTW$-Instanz eine zulässige Lösung hat, $\mathcal{NP}$-schwer sein muss. Dies wiederum impliziert, dass im Allgemeinfall des metrischen $TSPTW$ sogar das Auffinden einer zulässigen Lösung $\mathcal{NP}$-schwer ist[13], da die Aufgabe, eine zulässige Lösung anzugeben, klarerweise das Entscheidungsproblem, ob selbige existiert, impliziert. Insbesondere schließt diese Erkenntnis von vorne herein die Frage nach einem polynomiell beschränkten Approximations-Algorithmus für das allgemeine metrische $TSPTW$ aus, wogegen das TSP im allgemeinen metrischen Fall, mittels des Algorithmus' von *Christophides*, sogar mit einem konstanten Gütefaktor von $\alpha = \frac{3}{2}$ in polynomieller Zeit approximierbar ist[14].

2.3. Literatur und Anwendungen in der Praxis

Die Problematik des $TSPTW$ ist in der Literatur bis heute deutlich weniger intensiv bearbeitet als die Problematik des TSP: Während das *Travelling Salesman Problem* bereits seit den 1950er Jahren eine intensive Untersuchung in der Literatur zur angewandten Mathematik erfährt[15], ist dem *Travelling Salesman Problem mit Zeitfenstern* in dieser Hinsicht erst seit Beginn der 1980er Jahre eine größere Aufmerksamkeit zuteil geworden.

Seitdem wurden einige exakte Lösungsansätze präsentiert, welche auf unterschiedliche mathematische Instrumente des Operations Research zurück greifen. So existieren diverse Vorschläge, welche auf erweiterten, beziehungsweise modulierten Branch-And-Bound-Frameworks basieren, wie besiepielsweise die Ansätze von *Christofides, Mingozzi und Toth* [45], *Baker* [43] sowie *Langevin, Desrochers, Desrosiers, Gélinas und Soumis* [31]. Überdies finden sich in der Literatur vereinzelte Lösungsvarianten, welche auf der Technik des

[12] zur $\mathcal{NP}$-Vollständigkeit von *3-SAT* siehe beispielsweise [47]

[13] während sich dieselbe Aufgabe im Fall des TSP indes, durch die Angabe einer beliebigen Permutation über die Knotenmenge V, noch trivial gestaltet

[14] für nähere Ausführungen zum Thema Approximations-Algorithmen, zum Begriff des Gütefaktors sowie zum Algorithmus von *Christophides* siehe beispielsweise [14]

[15] vgl. hierzu beispielsweise [53]

Dynamic-Programmings fußen, wie beispielsweise jene von *Dumas, Desrosiers, Gelinas und Solomon* [28] sowie *Mingozzi, Bianco und Ricciardelli* [22]. Da das *TSPTW* sowohl die Problematik des *Schedulings* als auch die des *Rootings* auf sich vereint, womit es günstige Ansatzpunkte für die Technik des *Constrained Programmings* bietet, fand es ebenfalls in diesem Bereich Beachtung, so beispielsweise in den Werken von *Caseau und Koppstein* [33] sowie *Pesant, Gendreau, Potvin und Rousseau* [19],[17]. Indes präsentierten *Focacci, Lodi und Milano* [11] einen Hybridansatz aus klassischen Operations Research Techniken und *Constrained Programming*.[16]

Überdies finden sich seit dem Beginn der 1990er Jahre zunehmend Vorschläge für (nichtpolinomielle) Heuristiken zur Approximation des *TSPTW*, so zum Beispiel in den Publikationen von *Gendreau, Hertz und Laporte* [34], *Carlton und Barnes* [23], *Gendreau, Hertz, Laporte und Stan* [18],*Calvo* [16], *Ohlmann und Thomas* [8], *López-Ibáñez und Blum* [1] sowie *da Silva und Urrutia* [2].[17]

Ein Hauptgrund für die vermehrte Untersuchung des *TSPTW* im Verlauf der letzten drei Jahrzehnte kann vor allem in der erheblichen Praxisrelevanz gesehen werden, welche es für verschiedene moderne Problematiken der Wirtschaft sowie der Industrie aufweist. So ist es als "Basis-Problem"des zeitlich restringierten Routings auf viele logistische Applikationen und dynamische Konstruktionsvorgänge anwendbar.

Für eine Zusammenfassung moderner Routingprobleme mit Zeitfenstern sei, *Dash et al.* [5] folgend, beispielsweise an *Cordeau, Desaulniers, Desrosiers, Solomon und Soumis* [10] sowie *Desrosiers, Dumas, Solomon und Soumis* [27] verwiesen. Mögliche Anwendungen im Rahmen dynamischer Produktionsvorgänge in der Industrie finden sich beispielsweise in *Ascheuer* [25], wo die zeitlich restringierte Planung eines Stapelkranes in einem "single aisled automatic storage system" betrachtet und mittels des *TSPTW* modelliert wird.[18] Überdies ist die Einbindungsmöglichkeit von Zeitfenstern, laut *Cook*[19], die am häufigsten nachgefragte Zusatzapplikation für den populären *TSP*-Solver `Concorde` [56], was als ein weiterer Indikator für die erhebliche Praxisrelevanz des *TSPTW* betrachtet werden kann.

2.4. ILP Formulations

In diesem Abschnitt sollen drei verschiedene ILP-Formulierungen des *TSPTW* kurz vorgestellt werden.

Hierfür seien, in Anlehnung an *Dash et al.* [5], die folgenden Notations-Konventionen getroffen: Für einen Knoten $i \in V$ bezeichnet der Operator $V^+(i)$ die Menge aller Knoten, welche von i aus erreichbar sind, formal ist also $V^+(i) = \{j \in V : (i,j) \in A\}$. Vice versa bezeichnet $V^-(i)$ die Menge aller Knoten, von denen aus i erreicht werden kann, formal ist also $V^-(i) = \{k \in V : (k,i) \in A\}$.

2.4.1. Big-M Formulation

Mit der Entscheidungsvariablen $s_i \in \mathbb{Z}$, welche jedem Knoten $i \in V$ eine Startzeit $t_i = s_i$ zuordnet, lautet die *Big-M Formulation* des *TSPTW* (*TSPTW-BMF*)[20]

[16] vgl. zu diesem Absatz auch die entsprechenden Ausführungen von *Dash et al.* [5]
[17] vgl. hierzu beispielsweise [3]
[18] siehe hierzu auch die Ausführungen in Abschnitt 5.2
[19] im Rahmen einer persönlichen Kommunikation mit *Dash et al.* [5]
[20] vgl. hierzu [13]

$$\begin{aligned}
\min \quad & \sum_{(i,j)\in V} c_{i,j}x_{i,j} \\
\text{s.t.} \quad & \sum_{j\in V^+(i)} x_{i,j} = 1 \quad \forall\ i \in V && (2.5)\\
& \sum_{k\in V^-(i)} x_{k,i} = 1 \quad \forall\ i \in V && (2.6)\\
& s_i + \theta_{i,j} - (1 - x_{i,j})M_{i,j} \leq s_j \quad \forall\ (i,j) \in A && (2.7)\\
& R_i \leq s_i \leq D_i \quad \forall\ i \in V && (2.8)\\
& x_{i,j} \in \{0,1\} \quad \forall\ (i,j) \in A. && (2.9)
\end{aligned}$$

wobei $x_{i,j}$ genau dann den Wert 1 annimmt, wenn die Kante $(i,j) \in A$ Element der Lösungs-Tour ist und für jede Kante $(i,j) \in A$ ein möglichst kleines Big-M zu $M_{i,j} = D_i - R_j + \theta_{i,j}$ gewählt wird.

Restriktion (2.5) fordert, dass jede Stadt, außer der Start-Stadt p, genau einmal besucht, Restriktion (2.6) indes fordert, dass jede Stadt, außer der End-Stadt q, genau einmal verlassen wird.

Unter der Annahme, dass $\theta_{i,j} > 0$, erzwingt die Ungleichung (2.7), basierend auf dem aus der Scheduling-Literatur bekannten Mechanismus[21], sodann, dass die Besuchszeiten s_i entlang der Tour streng monoton steigend sind: Gegeben die Kante $(i,j) \in A$ wird in die Tour aufgenommen (ist also $x_{i,j} = 1$), so resultiert aus der entsprechenden Restriktion (2.7), dass $s_i + \theta_{i,j} \leq s_j$, woraus mit $\theta_{i,j} > 0$ folgt, dass wie gefordert $s_i < s_j$. Ist selbige Kante (i,j) indes nicht in der Tour (ist also $x_{i,j} = 0$), resultiert aus der entsprechenden Restriktion (2.7), dass $s_i + \theta_{i,j} - D_i + R_j - \theta_{i,j} \leq s_j$, also $R_j - D_j \leq s_j - s_i$. Da die Differenz der generell frühest möglichen Startzeit bei j und der generell spätest möglichen Startzeit bei i, namentlich $R_j - D_i$, klarerweise eine untere Schranke auf die mögliche Differenz der beiden Startzeiten von Knoten j und Knoten i, namentlich $s_j - s_i$, ist, wird die in Rede stehende Restriktion in diesem Fall somit, wie zu wünschen ist, redundant.

Zudem stellt die Restriktion (2.7) implizit sicher, dass jede zulässige Lösung der *TSPTW-BMF* frei von Subtouren ist.

Lemma 1. Eine zulässige Lösung der *Big-M Formulation* des *TSPTW* ist wegen (2.7) frei von Subtouren und entbehrt somit der klassischen *Subtour Elemination-Constraints*.

Beweis. Sei (x^*, t^*) eine zulässige Lösung der *TSPTW-BMF*, welche Subtouren aufweist, so muss es mindestens eine Subtour geben, die weder den Knoten p, noch den Knoten q enthält. In dieser Tour müsste exakt ein Knoten $k \in V \setminus \{p,q\}$ gleichzeitig sowohl Start- als auch Endknoten sein. Da, gegeben $\theta_{i,j} > 0$[22], die Startzeitpunkte s_i, gemäß Gleichung (2.7), entlang des Pfades streng monoton steigend sind und jedem Knoten nur exakt eine Startzeit zugeordnet wird, kann ein solches $k \in V \setminus \{p,q\}$ jedoch nicht existieren, woraus das Lemma folgt. □

Somit ist, in Verbindung mit den Restriktionen (2.5) und (2.6) sichergestellt, dass die Lösung der *TSPTW-BMF* einen Hamiltonweg von p nach q durch G beschreibt.

[21] siehe hierzu beispielsweise [66]

[22] tatsächlich ist es auf Instanzen mit der vereinzelten Parameterbelegung $\theta_{i,j} = 0$, so zum Beispiel jenen von *Dumas et al.* [28], im Fall der *TSPTW-BMF* nötig, für selbige Knotentupel ausdrücklich die jeweilige *Subtour Elimination Constraint* zu ergänzen

Die Restriktion (2.8) stellt sodann sicher, dass für alle $i \in V$ die Startzeiten s_i im entsprechenden Knotenzeitfenster $W_i = [R_i, D_i]$ liegen.

Daher entspricht offenbar jede zulässige Lösung der *TSPTW-BMF* einer zulässigen Lösung der zugrunde liegenden *TSPTW*-Instanz und umgekehrt.

2.4.2. Time Indexed Formulation

Da per Annahme sämtliche Daten integral sind, kann der Entscheidungsraum über die Startzeitpunkte mittels diskreter Zeitslots $t \in W_i$ modelliert werden:

Mit der Entscheidungsvariablen $z_i^t \in \{0,1\}$, welche genau dann den Wert 1 annimmt, wenn Knoten $i \in V$ die Startzeit (beziehungsweise Besuchszeit) $t \in W_i$ zugeordnet wird, sowie der Entscheidungsvariablen $y_{i,j}^t \in \{0,1\}$, welche genau dann den Wert 1 annimmt, wenn Knoten $i \in V \setminus \{q\}$ zum Startzeitpunkt $t \in W_i$ in Richtung des Knotens $j \in V \setminus \{q\}$ verlassen wird und in der Formulierung nur präsent ist, wenn $t + \theta_{i,j} \leq D_j$ (wenn sie also einen Reiseschritt bezeichnet, bei welchem der Folgeknoten j vor Ablauf seiner *Deadline* erreicht werden kann), lautet die Time Indexed-Formulierung des *TSPTW* (*TSPTW-TIF*)[23] sodann

$$\min \quad \sum_{(i,j)\in V} c_{i,j} x_{i,j}$$

$$\text{s.t.} \quad \sum_{t\in W_i} z_i^t = 1 \qquad \forall\ i \in V \tag{2.10}$$

$$\sum_{j\in V^+(i)} y_{i,j}^t = z_i^t \qquad \forall\ i \in V \setminus \{q\}, t \in W_i \tag{2.11}$$

$$\sum_{k\in V^-(i)} \sum_{\tau\in I_k(i,t)} y_{k,i}^\tau = z_i^t \qquad \forall\ i \in V \setminus \{p\}, t \in W_i \tag{2.12}$$

$$\sum_{t\in W_i} y_{i,j}^t = x_{i,j} \qquad \forall\ (i,j) \in A \tag{2.13}$$

$$x_{i,j}, y_{i,j}^t, z_i^t \in \{0,1\} \qquad \forall\ i \in V, \forall\ (i,j) \in A, \forall\ t \in W_i. \tag{2.14}$$

Hierbei enthält das Hilfs-Set $I_k(i,t)$ die Menge aller in Frage kommenden Startzeitpunkte $\tau \in W_k$ bei Knoten $k \in V \setminus \{q\}$, gegeben die Startzeit bei Knoten $i \in V \setminus \{q\}$ ist t und Knoten k der unmittelbare Vorgänger von i in der Tour. Präziser ist $\tau \in I_k(i,t)$ genau dann, wenn $\tau \in W_k$ und $\max\{\tau + \theta_{k,i}, R_i\} = t$ mit $t \in W_i$. Gegeben die Startzeit t bei i ist R_i, kann τ folgerichtig jede Startzeit bei Knoten k sein, deren zugehörige Ankunftszeit $\tau + \theta_{k,i}$ bei Knoten i kleiner oder gleich R_i ist, da in diesem Fall, wie oben bereits ausgeführt, Ankunfts- und Startzeit bei i auseinanderfallen, und die resultierende Diskrepanz als Wartezeit behandelt wird. Für $D_i \geq t > R_i$ ist hingegen das eindeutig zuzuordnende τ gerade jenes, welches dem Zusammenfallen von Ankunfts- und Startzeit bei i Rechnung trägt, also die Gleichung $\tau + \theta_{k,i} = t$ erfüllt.

Augenscheinlich impliziert die *Time Indexed Forumlation* die *Big-M Forumlation*:

Gleichung (2.10) stellt sicher, dass jedem Knoten $i \in V \setminus \{q\}$ ein eindeutiger Startzeitpunkt $t_i (= s_i)$ zugeordnet wird. Gleichung (2.11) gewährleistet, dass jeder Knoten, zu genau diesem Startzeitpunkt, exakt einmal, in Richtung eines anderen Knotens $j \in V \setminus \{p\}$ verlassen wird. Gleichung (2.12) bildet mittels des Hilfs-Sets $I_k(i,t)$ implizit die Einhaltung des Zeitfensters von i, sowie die adäquate Zuordnung eines entsprechenden Startzeitpunk-

[23] vgl. hierzu [6]

tes $\tau \in I_k(i,t)$ beim Vorgängerknoten k ab und erzwingt zugleich, dass jedes $i \in V \setminus \{p\}$ nur exakt einmal besucht wird, und zwar ebenfalls gerade zum ausgewählten Besuchszeitpunkt t. Somit subsumieren diese Ungleichungen die Restriktionen (2.5), (2.6), (2.7) und (2.8). Insbesondere wird daher eine Lösung der *TSPTW-TIF* ebenfalls stets frei von Subtouren sein. Gleichung (2.13) sodann verknüpft die Entscheidungsvariablen $y^t_{i,j}$ mit den Entscheidungsvariablen $x_{i,j}$, welche zur Evaluierung der Zielfunktion dienen.

Sei $\widehat{s}_i \in \mathbb{Z}^{|V|}$. Nach der Überführung $\widehat{s}_i = \sum_{t \in W_i} t z^t_i$, womit im Vektor $\widehat{s}_i$ nun die effektiv gewählten Start- beziehungsweise Besuchszeiten des jeweiligen Knotens i abzulesen sind, entspricht jede zulässige Lösung $(x, \widehat{s})$ der *TSPTW-TIF* also einer zulässigen Lösung (x, s) der *TSPTW-BMF*, was ferner bedeutet, dass sie ebenfalls stets eine zulässigen Lösung für die zugrunde liegende *TSPTW*-Instanz darstellt.[24]

Im Gegensatz zur *TSPTW-BMF* greift die *TSPTW-TIF* jedoch ausschließlich auf Binär-Variablen zurück und entbehrt somit einer Verwendung des "Big-M", von welcher bekannt ist, dass sie numerische Probleme verursachen, und, aufgrund des erhöhten "Solver-Spielraumes", verhältnismäßig schwache Duale Schranken sowie nachteiliges Verhalten der LP-Relaxation im Branch-And-Cut-Verfahren bergen kann.[25]

Insbesondere implizieren die Ausführungen dieses Abschnitts, dass die *TSPTW-TIF* die *TSPTW-BMF* in dem Sinne dominiert, dass die LP-Relaxation der *TSPTW-TIF* stets mindestens so gute Duale Schranken liefert wie die LP-Relaxation der *TSPTW-BMF*.[26]

2.4.3. Time Bucket Formulation

Um die *Time Bucket Formulation*, wie sie in *Dash et al.* [5] eingeführt wird, vorzustellen, sei zunächst die Idee der Buckets erläutert. Sodann werden, basierend hierauf, die *Time Bucket Realxation* und ihre Erweiterung zur *Time Bucket Formulation* präsentiert.

2.4.3.1. Bucket-Definition

Buckets $b \subseteq W_i$ fassen die Zeitslots $t \in W_i$ eines Knotens i zu "Päckchen" zusammen.

In Abbildung 2.1, welche den beschriebenen Sachverhalt visualisieren soll, repräsentieren die Zwischenräume auf dem oberen Strahl die diskreten Zeitslots, welche dem Zeitfenster von Knoten i zugeordnet werden können, die Zwischenräume auf dem unteren Strahl indes deren entsprechende Vereinigungen zu Buckets.

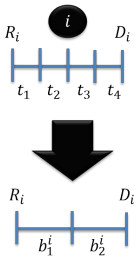

Abbildung 2.1.: Verbildlichung der Bucket-Definition anhand eines beliebigen Knotens i

Formal wird das Gesamtzeitfenster W_i jedes Knotens $i \in V$ in ein Set von Buckets, namentlich $B_i = \{b^i_1, \ldots, b^i_L\}$, unterteilt, wobei jede Bucket $b^i_l = [r_{b^i_l}, d_{b^i_l}]$ eine eigene *Releasetime* $r_{b^i_l}$ und eine eigene *Deadline* $d_{b^i_l}$ besitzt. An die Buckets b^i_l sind die folgenden

[24] für einen streng formalen Beweis zu diesen Behauptungen sei an *Dash et al.* [5] verwiesen
[25] siehe hierzu beispielsweise [20]
[26] siehe hierzu auch [5]

Anforderungen gestellt: (1.) $r_{b_1^i} = R_i$, was bedeutet, dass die erste Bucket eines Knotens zu dessen *Releasetime* beginnen muss, (2.) $d_{b_L^i} = D_i$, was bedeutet, dass die letzte Bucket eines Knotens zu dessen *Deadline* enden muss, und (3.) $r_{b_{l+1}^i} > d_{b_l^i}$ für alle $i \in V$, was bedeutet, dass die Zeitfenster der Buckets disjunkt sein müssen. Ferner wird $W_i \neq \bigcup_{b \in B_i}[r_b, d_b]$ gestattet, womit Lücken zwischen den verschiedenen Buckets zugelassen sind. Dies ist opportun, da innerhalb eines Knotenzeitfensters W_i Zeitslots $t \in W_i$ existieren können, die der Salesman in keiner zulässigen Lösung wahrnehmen kann. Wenn besagte Zeitslots von keiner der verwendeten Buckets erfasst werden, hat dies folgerichtig keinerlei Auswirkung auf die optimale Lösung. Angelehnt an das Schema in Abbildung 2.1 wird in Abbildung 2.2 eine entsprechende Bucket-Unterteilung visualisiert.

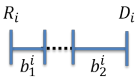

Abbildung 2.2.: Verbildlichung einer Bucket-Unterteilung mit $W_i \neq \bigcup_{b \in B_i}[r_b, d_b]$

Insbesondere ist zu beachten, dass nicht der gesamte zugrunde liegende Problem-Zeitraum $\bigcup_{i \in V} W_i$ in eine generelle Zusammensetzung aus Buckets zerlegt wird, sondern vielmehr jedes Knotenzeitfenster W_i eine individuelle Bucket-Unterteilung erfährt, und die Buckets im Rahmen dessen zudem verschiedene Größen aufweisen können.

2.4.3.2. Time Bucket Relaxation

Basierend auf der gerade eingeführten Idee der Buckets lässt sich nun, in Analogie zu der *Time Indexed Formulation*, die *Time Bucket Relaxation* definieren:

Mit der Entscheidungsvaribalen $z_i^b \in \{0,1\}$, welche genau dann den Wert 1 annimmt, wenn Knoten $i \in V \setminus \{q\}$ zur Bucket $b \in B_i$ besucht wird, sowie der Entscheidungsvariablen $y_{i,j}^b \in \{0,1\}$, welche genau dann den Wert 1 annimmt, wenn Knoten $i \in V \setminus \{q\}$ zur Bucket $b \in B_i$ in Richtung des Knotens $j \in V \setminus \{q\}$ verlassen wird und in der Formulierung nur präsent ist, wenn $r_b + \theta_{i,j} \leq D_j$ (wenn sie also einen Reiseschritt bezeichnet, bei welchem der Folgeknoten j vor Ablauf seiner *Deadline* erreicht werden kann), lautet die *Time Bucket Relaxation* des *TSPTW* (*TSPTW-TBR*)[27] sodann

$$\min \quad \sum_{(i,j) \in V} c_{i,j} x_{i,j}$$

$$\text{s.t.} \quad \sum_{b \in B_i} z_i^b = 1 \qquad \forall\ i \in V \tag{2.15}$$

$$\sum_{j \in V^+(i)} y_{i,j}^b = z_i^b \qquad \forall\ i \in V \setminus \{q\}, b \in B_i \tag{2.16}$$

$$\sum_{k \in V^-(i)} \sum_{\beta \in \mathcal{I}_k(i,b)} y_{k,i}^\beta = z_i^b \qquad \forall\ i \in V \setminus \{p\}, b \in B_i \tag{2.17}$$

$$\sum_{b \in B_i} y_{i,j}^b = x_{i,j} \qquad \forall\ (i,j) \in A \tag{2.18}$$

$$x_{i,j}, y_{i,j}^b, z_i^b \in \{0,1\} \qquad \forall\ i \in V, \forall\ (i,j) \in A, \forall\ b \in B_i. \tag{2.19}$$

Hierbei bezeichnet das Hilfs-Set $\mathcal{I}_k(i,b)$ die Menge aller in Frage kommenden Buckets $\beta \in B_k$ bei Knoten $k \in V \setminus \{q\}$, gegeben die gewählte Bucket bei Knoten $i \in V \setminus \{q\}$ ist

[27] vgl. hierzu [5]

b und Konten k der unmittelbare Vorgänger von i in der Tour. Präziser ist $\beta \in \mathcal{I}_k(i, b_\ell)$ genau dann, wenn $\beta \in B_k$ und

$$d_{b_{\ell-1}} < r_\beta + \theta_{k,i} \leq d_{b_\ell} \tag{2.20}$$

mit $b \in B_i$, wobei $d^i_{b_0} = -\infty$ für alle $i \in V$. Offenbar ist wegen (2.20) die effektiv im Modell zugeordnete Startzeit bei Bucket β stets r_β, ungeachtet dessen, wann diese Bucket tatsächlich erreicht wurde. Dementsprechend berechnet sich die Ankunftszeit beim Folge-Knoten i zu $r_\beta + \theta_{k,i}$. Als gewählte Bucket bei i wird sodann jenes b zugeordnet, dessen Zeitfenster die besagte Ankunftszeit enthält. Da $d^i_{b_0} = -\infty$ gilt, kann zudem jede erste Bucket b^i_1 eines Knotens i beliebig früh erreicht werden. Verlassen wird sie jedoch, wie alle anderen Buckets auch, zu ihrer *Releasetime*. Dies trägt genau dem Effekt der Wartezeit Rechnung, welche sich einstellt, wenn ein Knoten vor Beginn seines Zeitfensters erreicht wird.

Gemäß der gerade diskutierten Definition ist die Modellierung der *Time Bucket Relaxation* also um den folgenden Effekt gelockert: Ein Knoten $i \in V \setminus \{p, q\}$ kann früher verlassen werden, als er erreicht wurde, und zwar genau dann, wenn für die Ankunftszeit bei der entsprechende Bucket $b_i \in B_i$ gilt, dass $r_\beta + \theta_{k,i} > r_{b_i}$. Die daraus resultierende, zeitliche Diskrepanz $r_\beta + \theta_{k,i} - r_{b_i}$ bezeichnen *Dash et al.* folgerichtig auch als "*negative waiting time*" (oder "*negative Wartezeit*") bei Bucket b_i.

Überdies muss eine zulässige Lösung der *TSPTW-TBR* nicht zwingend frei von Subtouren sein. Um dies einzusehen, soll das folgende Beispiel aus *Dash et al.* [5] betrachtet werden: Gegeben seien zwei Knoten $i, j \in V$ welche deckungsgleiche Zeitfenster $W_i = W_j$ aufweisen, die jeweils in nur eine Bucket $b^i_1 = b^j_1 = [R_i, D_i] = [R_j, D_j]$ unterteilt wurden. Es gelte ferner $R_i + \theta_{i,j} = D_j$ und $R_j + \theta_{j,i} = D_i$. Eine Subtour, welche von i nach j und von dort wieder zurück nach i führt, wird nun von den *TBR*-Restriktionen und der Definition des Sets $\mathcal{I}_k(i, b)$ offensichtlich nicht ausgeschlossen.

Indes wird die *TSPTW-TBR* einer *TSPTW*-Instanz in aller Regel jedoch deutlich kompakter als die entsprechende *TSPTW-TIF* ausfallen, da unter anderem in den meisten Fällen der Bucket-Unterteilung offenbar $|\mathcal{I}_k(i, b)| \ll |I_k(i, t)|$ gilt.

Insbesondere folgt aus den obigen Ausführungen, dass die *Time Bucket Relaxation nicht* äquivalent zur *Time Indexed Formulation* ist, da zwar klarerweise gilt, dass jeder zulässigen Lösung der zugrunde liegenden *TSPTW*-Instanz stets eine zulässige Lösung der *TSPTW-TBR* zugeordnet werden kann, eine zulässige Lösung für die *TSPTW-TBR* jedoch *keine* zulässige Lösung für die zugrunde liegende *TSPTW*-Instanz liefern muss. Somit impliziert die *Time Bucket Relaxation* also die *Time Indexed Formulation*, die Gegenimplikation gilt jedoch nicht.

2.4.3.3. Mögliche Cuts für Subtouren und Unzulässige Pfade

Im Folgenden sollen mögliche Ergänzungen der *Time Bucket Relaxation* diskutiert werden, durch welche selbige in die *Time Bucket Formulation* überführt werden kann.

Es notiere für zwei Knoten-Mengen $S, S' \subset V$ der Operator $\delta(S, S')$ die Menge aller Kanten, welche in S starten und in S' enden, formal ist also $\delta(S, S') = \{(i, j) \in A : i \in S, j \in S'\}$. Ferner notiere $\delta^+(S)$ die Menge aller Kanten, welche aus S herausführen, formal ist also $\delta^+(S) = \delta(S, V \setminus S)$. Entsprechend beschreibe $\delta^-(S)$ die Menge aller Kanten, welche in S hineinführen, formal ist also $\delta^-(S) = \delta(V \setminus S, S)$. Überdies sei $\overline{S}$ eine alternative Schreibweise für $V \setminus S$.

Unterbindung der Subtouren Wie bereits in Abschnitt 2.4.3.2 diskutiert, kann eine zulässige Lösung der *TSPTW-TBR*, bedingt durch den Effekt der *negative waiting time*, Subtouren aufweisen. Eine theoretisch mögliche Behebung dieser Problematik besteht darin, das ILP um die aus der Literatur bekannten[28], klassischen (gerichteten) *Subtour Elimination-Constraints* (*SECs*), namentlich

$$\sum_{(i,j)\in\delta^+(S)} x_{i,j} \geq 1 \quad \forall\ S \subseteq V \setminus \{q\} \tag{2.21}$$

zu erweitern, wobei selbige um eine geringfügige Adaptierung an die Modellierungsparadigmen des *TSPTW* moduliert sind. Namentlich ist hierbei der Knoten q von sämtlichen Subsets S ausgeschlossen, da er, gemäß Voraussetzung, den Endknoten jeder Tour darstellt und somit nicht mehr verlassen wird. Es ist unmittelbar einzusehen, dass das Ergänzen der Ungleichungen (2.21) die in Rede stehende *TSPTW-TBR* von Subtouren befreit.

Jedoch bewegt sich die Anzahl hierzu notwendiger, zusätzlicher Restriktionen, wie im Fall der klassischen *SECs* , in der Größenordnung $O(\mathfrak{P}(S)) = O(2^{|S|})$.

Unterbindung der Unzulässigen Pfade Der Effekt der *negative waiting time* kann überdies dazu führen, dass eine Tour, bezogen auf die Zeitfenster W_i der Knoten i, in der *TSPTW-TBR* zulässig ist, in der zugrunde liegenden *TSPTW*-Instanz jedoch (da der Solver entlang der Tour mittels *negative waiting times* "Zeit gewinnen" konnte) eine Überschreitung diverser *Deadlines* D_i aufweist, womit sie, auch wenn sie frei von Subtouren ist, unzulässig wäre. Präzise soll ein (Teil-)Pfad $P = (v_1, v_2, ..., v_h)$ (in Anlehnung an *Ascheuer et al.* [13] als geordnetes Set von Knoten in V angegeben), welcher in keiner zulässigen Lösung enthalten sein kann, als "Unzulässiger Pfad" oder auch "*infeasible path*" bezeichnet werden. Augenscheinlich ist somit insbesondere jeder Pfad, auf welchem in der tatsächlichen *TSPTW*-Tour mindestens einer der Knoten v_i außerhalb seines Zeitfensters W_i verlassen würde, ein *infeasible path*, was in der praktischen Umsetzung noch auszunutzen sein wird.

Eine theoretisch mögliche Vermeidung der Präsenz Unzulässiger Pfade in der ILP-Lösung ist das Hinzufügen der von *Ascheuer* [25] vorgestellten und von *Dash et al.* [5] aufgegriffenen "*infeasible path-Constraints*" (*IPCs*), namentlich

$$\sum_{i=1}^{h-1} x_{v_i,v_{i+1}} \leq h-2 \quad \forall\ P \in \Pi \tag{2.22}$$

wobei Π die Menge aller Unzulässigen (Teil-)Pfade P bezeichnen soll. Augenscheinlich verbieten diese, da $h-2 = |A_P| - 1$, wobei A_P die Menge aller in diesem (Teil-)Pfad verwendeten Kanten bezeichnen soll, für jeden Unzulässigen (Teil-)Pfad P ausdrücklich die Auswahl von mindesten einer der Kanten $(v_i, v_{i+1}) \in A_P$ in der ILP-Lösung, womit diese im Endeffekt frei von allen Unzulässigen Pfaden in Π sein wird.

Offensichtlich bewegt sich die Anzahlt hierzu notwendiger, zusätzlicher Restriktionen jedoch in der Größenordnung $O(|\Pi|) = O(n!)$.

Eine um die Restriktionen (2.21) und (2.22) erweiterte *Time Bucket Relaxation* liefert nun klarerweise eine zulässige Lösung für die korrespondierende *TSPTW*-Instanz und darf folgerichtig auch als *Time Bucket Formulation* bezeichnet werden. Selbige ist also nun-

[28] siehe hierzu beispielsweise [26] oder [5]

mehr gleichwertig mit der *TSPTW-TIF* sowie der *TSPTW-BMF* . Durch das Ergänzen der besagten Ungleichungen erlangt das resultierende ILP jedoch eine unter praktischen Gesichtspunkten völlig indiskutable Größe. Insbesondere ist hierdurch die bereits diskutierte vorteilhafte Kompaktheit der ursprünglichen *TSPTW-TBR* zunichte.

Eine praxistaugliche Lösung für diese Problematik ist, wie sich zeigen wird, das Einbetten der Restriktionen (2.21) und (2.22), beziehungsweise hierauf basierender, erweiterter Schnittebenen, in ein Branch-And-Cut-Framework (B&C-Framework), wie es in *Dash et al.* [5] vorgestellt wird. Nähere Ausführungen hierzu folgen in Abschnitt 4.2.

3. Allgemeines Preprocessing

Das prinzipielle Ziel des allgemeinen Preprocessings, wie es in dieser Arbeit präsentiert werden soll, ist, gegeben eine *TSPTW*-Instanz, das Straffen der Knoten-Zeitfenster (präziser das Erhöhen der R_i und das Verringern der D_i, durch das jeweilige Ausschließen von Knoten-Zeitslots, welche im Hinblick auf eine zulässige Lösung obsolet sind), sowie das Verringern der Kantenmenge A (präziser das Entfernen von Kanten (i, j) aus A, welche bezogen auf die Zeitfenster- und Distanzkonstellation der Städte niemals in einer zulässigen Tour enthalten sein können).

In diesem Kapitel seien einige gängige Techniken des allgemeinen Preprocessings vorgestellt.

3.1. Knotenpräzedenzen und Hilfsparameter

In einem ersten Schritt des Preprocessings wird versucht, möglichst präzise Informationen über Knotenpräzedenzen zu errechnen, welche von der Beschaffenheit der jeweiligen Instanz impliziert sind.

Gegeben sei ein Knoten $i \in V$ welcher in jeder zulässigen Tour *vor* einem anderen Knoten $j \in V \setminus \{i\}$ besucht werden muss, so spricht man davon, dass der Knoten i den Knoten j "*preceeded*". Man schreibt auch $i \prec j$. Symmetrischer Weise sei ein Knoten $i \in V$ gegeben, welcher in jeder zulässigen Tour *nach* einem anderen Knoten $k \in V \setminus \{i\}$ besucht werden muss, so spricht man davon, dass Knoten i den Knoten k "*succeeded*". Man schreibt auch $i \succ k$.

Um derartige Knotenrelationen für eine gegebene *TSPTW*-Instanz zu bestimmen, bedient man sich in der Regel diverser Hilfsparameter. Die folgenden Ausführungen stützen sich auf *Viergutz (2009)* [4] sowie *Dash et al.* [5], sind jedoch geringfügig moduliert, um Konsistenz mit der in dieser Arbeit verwendeten Nomenklatur zu wahren:

Es sei $EST(i, j)$ als die "*frühestmögliche Startzeit*", oder auch "*Earliest Starting Time*" bei Knoten $j \in V \setminus \{q\}$ definiert, gegeben der direkte Vorgänger von Knoten j ist Knoten $i \in V \setminus \{q\}$. Ferner sei $LST(i, j)$ als die "*spätestmögliche Startzeit*", oder auch "*Latest Starting Time*" bei Knoten $i \in V \setminus \{q\}$ definiert, gegeben der direkte Nachfolger von Knoten i ist Knoten $j \in V \setminus \{p\}$. Formal sind

$$EST(i, j) = \max\{R_i + \theta_{i,j}, R_j\} \quad \forall\ (i, j) \in A \setminus \{p, q\} \tag{3.1}$$

$$LST(i, j) = \min\{D_j - \theta_{i,j}, D_i\} \quad \forall\ (i, j) \in A \setminus \{p, q\}. \tag{3.2}$$

Deklaration (3.1) begründet sich wie folgt: Falls die frühestmögliche Ankunftszeit $R_i + \theta_{i,j}$ von i bei j vor der *Releasetime* von j liegt, so ist diese Releastime für die Betrachtung der frühestmöglichen Startzeit bei j entsprechend bindend, da der Knoten nicht vorher verlassen werden kann. Ist hingegen $R_i + \theta_{i,j} \geq R_j$, so ist die frühestmögliche Startzeit bei j gerade genau die frühestmögliche Ankunftszeit $R_i + \theta_{i,j}$.

Deklaration (3.2) ist unmittelbar einzusehen: Entweder ist $D_j - \theta_{i,j} < D_i$, so ist $D_j - \theta_{i,j}$ für die spätestmögliche Startzeit von i nach j bindend, da Knoten j sonst nicht mehr recht-

zeitig erreicht würde, im Gegenfall $D_j - \theta_{i,j} \geq D_i$ ist stattdessen D_i für die spätestmögliche Startzeit von i nach j entsprechend bindend, da i nicht nach D_i verlassen werden darf.

Für den Fall, dass im Hinblick auf die Zeitfensterkonstellation kein zulässiger Weg von i nach j existiert, also $R_i + \theta_{i,j} > D_j$ gilt, wird $EST(i,j) = LST(i,j) = +\infty$ gesetzt.

Aus den gerade definierten Hilfsparametern lassen sich sodann initiale Präzedenzerkenntnisse über die Knoten ableiten:

$$k \prec i \quad wenn \quad EST(i,k) > D_k \quad \forall \; k,i \in V \setminus \{p,q\} \tag{3.3}$$

Die Deklaration (3.3) folgt unmittelbar aus der Tatsache, dass Knoten k frühestens nach seiner *Deadline* verlassen werden könnte, wenn er hinter Knoten i besucht würde, womit er in einer zulässigen Lösung zwangsläufig vor i besucht werden muss.

Die so erworbenen, initialen Präzedenzkenntnisse können sodann, basierend auf einfachen, logischen Implikationen, verschärft werden:

$$Wenn \quad k \prec i \wedge i \prec j \quad gilt \;\; auch \;\; k \prec j \quad \forall \; i,j,k \in V \setminus \{p,q\} \tag{3.4}$$

Gegeben Knoten k ist ein obligatorischer Vorgänger von i, und i ist seines Zeichens ein obligatorischer Vorgänger von j, so ist k trivialerweise ebenfalls ein obligatorischer Vorgänger von j. Eben hierauf basiert die Regel (3.4).

Über die Deklaration (3.4) wird sodann iteriert, bis keine weiteren Präzedenzerkenntnisse mehr gefunden werden. Insbesondere impliziert dieses Vorgehen ebenfalls das Abdecken logischer Implikationsketten von größerer Länge.

Schlussendlich kann für alle $k,i \in V$ mit $k \prec i$ die Konstellation $i \succ k$ deklariert werden, da klarerweise $k \prec i \Leftrightarrow i \succ k$, womit ein Berechnen von successor-Relationen parallel zu den predecessor-Relationen eine Redundanz, und somit vergeudete CPU-Zeit, darstellen würde.

3.2. Kantenelimination

Im zweiten Schritt des Preprocessings werden sodann, unter anderem basierend auf den in Abschnitt 3.1 errechneten Knotenpräzedenzen, Kanten $(i,j) \in A$ gelöscht. Hierzu werden die folgenden Kanteneliminations-Regeln formuliert:

Gegeben seien zwei Knoten $i,j \in V$ mit $j \prec i$, so kann die Kante $(i,j) \in A$ aus A gelöscht werden. Dies ist einzusehen, da, laut Voraussetzung, Knoten j in jeder zulässigen Tour vor Knoten i besucht werden muss, und Kante (i,j), welche das direkte Gegenteil impliziert, somit keinesfalls in einer zulässigen Tour enthalten sein kann. Überdies, gegeben drei verschiedene Knoten $i,j,k \in V$, kann die Kante $(i,j) \in A$ aus A gelöscht werden, wenn

$$k \prec i \wedge j \prec k. \tag{3.5}$$

Gegeben Knoten k muss in jeder zulässigen Tour vor Knoten i besucht werden und Knoten j muss in jeder zulässigen Tour vor Knoten k besucht werden, womit effektiv auch gilt, dass in jeder zulässigen Tour j vor i zu besuchen ist, so kann die Kante (i,j) keinesfalls in einer zulässigen Tour enthalten sein. Folglich kann sie aus A gelöscht werden.

Eben dies bildet die Regel (3.5) ab. Ferner können die Kanteneliminations-Regeln

$$(k \prec i \vee k \prec j \vee R_i + \theta_{i,j} + \theta_{j,k} > D_k) \wedge (i \prec k \vee j \prec k \vee R_k + \theta_{k,i} + \theta_{i,j} > D_j) \quad (3.6)$$

$$EST(i,j) > LST(j,k) \wedge EST(k,i) > LST(i,j) \quad (3.7)$$

konstruiert werden, welche jeweils eine Löschbarkeit der Kante $(i,j) \in A$ implizieren, wobei $k \in V \setminus \{i,j,p,q\}$.

Vergleicht man die Literale der beiden Klauseln in (3.6) paarweise, so fällt auf, dass sich das $n$te Literal der ersten Klausel jeweils dichotomisch zum $n$ten Literal der zweiten Klausel verhält. Gegeben die zugrunde liegende $TSPTW$-Instanz ist lösbar, können besagte Paare, bis auf das Letzte, also niemals gleichzeitig erfüllt sein. Jede andere Paarung führt indes zu einer Eliminations-Regel. Gilt beispielsweise sowohl $k \prec i$ als auch $j \prec k$, so kann die Kante (i,j) in analoger Argumentationsweise zu der im Fall von (3.5) aus A gelöscht werden. Gilt indes sowohl $k \prec i$ als auch $R_k + \theta_{k,i} + \theta_{i,j} > D_j$, so muss zwar in einer zulässigen Lösung k vor i besucht werden, dies kann jedoch offenbar nicht geschehen, falls nach i sodann j besucht (also Kante (i,j) verwendet) wird. Insbesondere kann es erst recht nicht geschehen, wenn zwischen k und i noch weitere Knoten besucht werden, da dies die Überschreitung der *Deadline* D_j höchstens vergrößern würde. Folgerichtig muss j vor i besucht werden, um die Präzedenz $k \prec i$ einhalten zu können. Daher kann die Kante (i,j) in keiner zulässigen Tour enthalten sein und darf somit aus A gelöscht werden.

Die Argumentationen bezüglich der restlichen Literalen-Paare gestalten sich analog zu diesen Ausführungen.

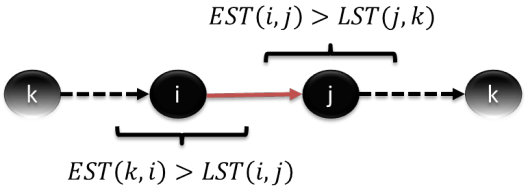

Abbildung 3.1.: Schematische Verbildlichung der Kantenlöschungs-Regel aus (3.7) (In Anlehnung an *Viergutz* [4])

Die Kantenlöschungs-Regel in (3.7) begründet sich indes wie folgt: Es sei die Kante (i,j) bereits in die Tour aufgenommen. Knoten $k \in V$ kann sodann nicht vorher besucht werden, da $EST(k,i) > LST(i,j)$, was bedeutet, dass i von k aus in keinem Fall früh genug erreicht werden kann, um rechtzeitig nach j aufzubrechen. Da jedoch ebenfalls $EST(i,j) > LST(j,k)$ gilt, was bedeutet, dass der Knoten k nicht nach der Kante (i,j) besucht werden kann, da die frühestmögliche Ankunftszeit von i bei j nicht ausreicht, um j rechtzeitig in Richtung k zu verlassen, kann Knoten k in diesem Fall folglich gar nicht besucht werden, womit keine Tour, welche Kante (i,j) enthält, alle Knoten $i \in V$ besuchen und somit zulässig sein kann. Daher darf Kante (i,j) aus A gelöscht werden. Tatsächlich kann (3.7) als eine Hilfsparameter-basierte Verstärkung des Falles von (3.6), in welchem je das letzte Literal der beiden Klauseln aufgegriffen ist, betrachtet werden. Somit legt diese Erkenntnis eine generelle Forcierung von (3.6) nahe, indem sie entsprechend mit (3.7) zu der folgenden, neuen Kantenlöschungs-Regel kombiniert wird:

$$(k \prec i \vee k \prec j \vee EST(i,j) > LST(j,k)) \wedge (i \prec k \vee j \prec k \vee EST(k,i) > LST(i,j)) \quad (3.8)$$

Gemäß den obigen Ausführungen ist leicht einzusehen, dass (3.8) gültig ist und insbe-

sondere die Kantenlöschungs-Regeln (3.6) sowie (3.7) in dem Sinne dominiert, dass sie stets mindestens genau so viele Kanten löschen wird.

In Anlehnung an die Illustration in *Viergutz* [4] sei in Abbildung 3.1 die Grundlage der Kantenlöschungs-Regel aus (3.7) noch einmal schematisch verbildlicht.

3.3. Zeitfensterstraffung

In einem dritten Schritt des Preprocessings werden schließlich, unter anderem basierend auf den in Abschnitt 3.1 errechneten Knotenpräzedenzen, die Zeitfenster gestrafft.

Hierzu seien zunächst die beiden Hilfs-Sets

$$\widehat{\pi}(i) := \{k \in V : (k,i) \in A \ \wedge \ k \not\succ i\} \quad \forall \ i \in V \setminus \{p\} \tag{3.9}$$

$$\widehat{\sigma}(i) := \{j \in V : (i,j) \in A \ \wedge \ j \not\prec i\} \quad \forall \ i \in V \setminus \{q\} \tag{3.10}$$

definiert. Hierbei soll $\widehat{\pi}(i)$ als die Menge aller möglichen *direkten* Vorgänger von Knoten i bezeichnet werden, wobei Knoten k gemäß (3.9) genau dann einen möglichen Vorgänger von i darstellt, wenn er kein notwendiger Nachfolger von i ist und die Kante (k,i) existiert. Symmetrischer Weise soll $\widehat{\sigma}(i)$ als die Menge aller möglichen Nachfolger von Knoten i bezeichnet werden, wobei gemäß (3.10) Knoten j genau dann einen möglichen Nachfolger von i darstellt, wenn er kein notwendiger Vorgänger von i ist.

Sodann formulieren sich die vier Zeitfensterstraffungs-Regeln

$$R_i \longleftarrow \max\{R_i, \min_{k \in \widehat{\pi}(i)} \{R_k + \theta_{k,i}\}\} \tag{3.11}$$

$$R_i \longleftarrow \max\{R_i, \min\{D_i, \min_{j \in \widehat{\sigma}(i)} \{R_j - \theta_{i,j}\}\}\} \tag{3.12}$$

$$D_i \longleftarrow \min\{D_i, \max\{R_i, \max_{k \in \widehat{\pi}(i)} \{D_k + \theta_{k,i}\}\}\} \tag{3.13}$$

$$D_i \longleftarrow \min\{D_i, \max_{j \in \widehat{\pi}(i)} \{D_j - \theta_{i,j}\}\}. \tag{3.14}$$

Die Regel (3.11) kann wie folgt begründet werden: Der Ausdruck $\min_{k \in \widehat{\pi}(i)}\{R_k + \theta_{k,i}\}$ liefert, betrachtet über alle Vorgänger k welche für den Knoten i in Frage kommen, die insgesamt früheste Ankunftszeit $R_k + \theta_{k,i}$ bei i, welche sich in einer zulässigen Tour ergeben kann. Liegt diese über der *Releasetime* R_i von i, ist selbige somit obsolet und kann entsprechend auf $\min_{k \in \widehat{\pi}(i)}\{R_i + \theta_{k,i}\}$ erhöht werden. In symmetrischer Argumentationsweise hierzu begründet sich die Regel (3.14).

In Abbildung 3.2 seien die Zeitfensterstraffungs-Regeln (3.11) und (3.14) noch einmal schematisch verbildlicht.

Die Regel (3.13) indes kann durch die folgende Überlegung legitimiert werden: Der Ausdruck $\max_{k \in \widehat{\pi}(i)}\{D_k + \theta_{k,i}\}$ liefert, betrachtet über alle Vorgänger k die für den Knoten i in Frage kommen, die insgesamt späteste Ankunftszeit $D_k + \theta_{k,i}$ bei i, welche sich in einer zulässigen Tour ergeben kann. Liegt diese über- oder auf der *Releasetime* R_i und zugleich unter der *Deadline* D_i, so ist sie zugleich die spätestmögliche Abfahrtszeit, welche sich bei i einstellen kann. Das alte D_i ist somit obsolet und kann entsprechend auf $\max_{k \in \widehat{\pi}(i)}\{D_k + \theta_{k,i}\}$ reduziert werden. In symmetrischer Argumentationsweise hierzu begründet sich die Regel (3.12), wobei selbige davon ausgeht, dass an keinem Knoten eine (aus einer Ankunft vor Beginn des Zeitfensters resultierende) Wartezeit entsteht, was o.B.d.A. möglich ist, da sich Wartezeiten gegenüber der Optimalität sowie der Zulässigkeit einer Lösung offensichtlich neutral verhalten.

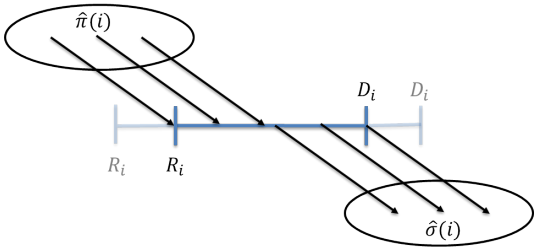

Abbildung 3.2.: Schematische Verbildlichung der Zeitfensterstraffungs-Regeln (3.11) und (3.14)

3.4. Iteratives Preprocessing

Augenscheinlich weisen die in den Abschnitten 3.1, 3.2 und 3.3 vorgestellten Preprocessing-Elemente erhebliche Synergien auf: Sind die Zeitfenster gestrafft, können schärfere Präzedenzen, ESTs und LSTs berechnet werden. Schärfere Präzedenzen, ESTs und LSTs begünstigen eine effektivere Kantenlöschung. Sind mehr Kanten gelöscht, können wiederum die Zeitfenster besser gestrafft werden, und so weiter.

Daher bietet es sich an, sämtliche Preprocessing-Routinen mehrmals zu wiederholen, bis ein gewisses Abbruchskriterium eintritt, beispielsweise eine ausbleibende weitere Straffung der Zeitfenster im letzten Iterations-Durchgang.[1]

In Abbildung 3.3 sei der Ablauf des allgemeinen, iterativen Preprocessings noch einmal anhand eines Flussdiagramms verbildlicht, wobei ΔTW_i für jeden Knoten $i \in V$ als die absolute Änderung der Zeitfensterweite $|W_i|$, von der vorhergehenden auf die aktuelle Preprocessing-Iteration, definiert sei.

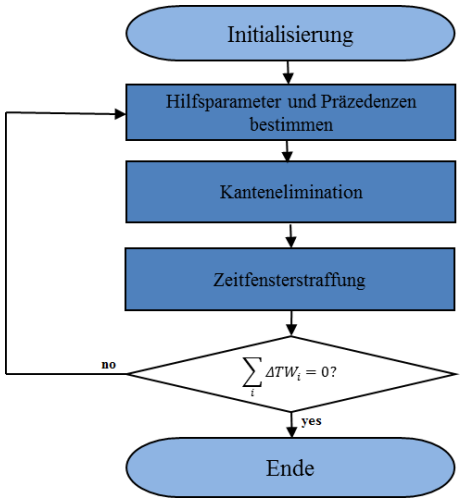

Abbildung 3.3.: Verbildlichung des iterativen, allgemeinen Preprocessings anhand eines Flussdiagramms

Um eine visuelle Impression der großen Bedeutung des Prepocessings für die Modellvereinfachung zu liefern, seien, im Rahmen eines Vorgriffs auf den Experimentalteil in Abschnitt 5, in Abbildung 3.4 die Adjazenz-Situationen der $n100w20.001$-Instanz[2] von

[1] eigene Experimente auf den Instanzen von *Dumas et al.* [28] verzeichneten hierbei beispielsweise im Durchschnitt meist zwischen 5 und 10 Iterationen

[2] für nähere Erläuterungen hierzu siehe Abschnitt 5

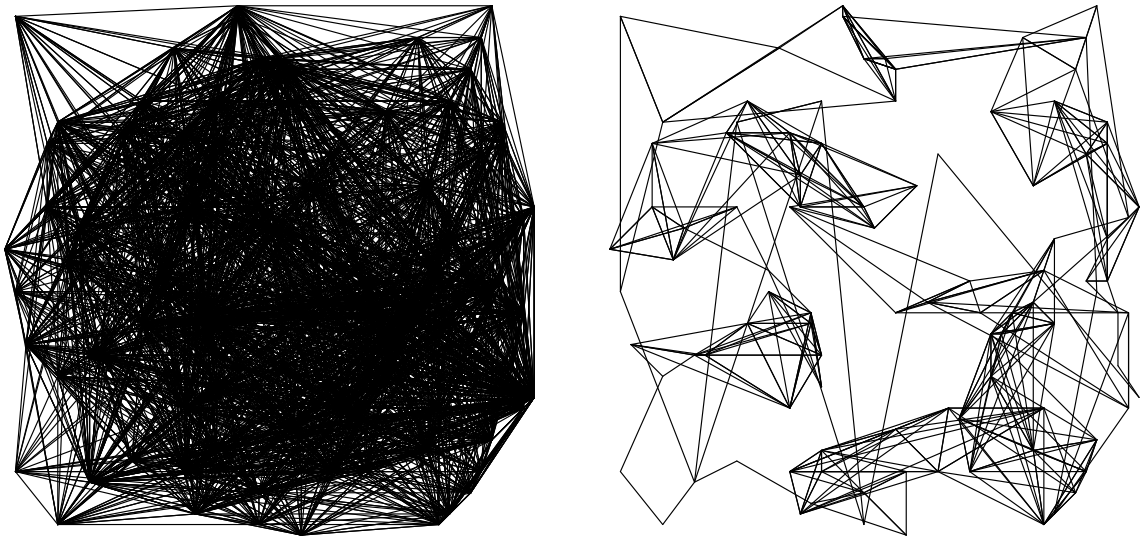

Abbildung 3.4.: Visualisierte Adjazenz-Situation der $n100w20.001$-Instanz von *Dumas et al.* [28] vor- (links) und nach (rechts) Preprocessing

Dumas et al. [28] vor- (links) und nach (rechts) der Anwendung des oben beschriebenen, iterativen Preprocessing-Verfahrens abgebildet, wobei aus Gründen der Übersicht auf die ausdrückliche Darstellung der Knoten verzichtet wurde, und eine Verbindungskante (i, j) bereits dann (als schwarze Verbindungslinie) angedeutet ist, wenn in mindestens einer der beiden Richtungen (i, j) und (j, i) eine Adjazenz besteht. Ferner soll für beide Graphen bereits die triviale Einschränkung gelten, dass eine Adjazenz in der Richtung (i, j) nur dann besteht, wenn $R_i + \theta_{i,j} \leq D_j$ gilt, es also prinzipiell möglich ist, den Knoten j von Knoten i aus vor der *Deadline* von j zu erreichen. Offensichtlich ist der Kontrast dennoch frappierend.

Ausführlichere Resultate und insbesondere Zahlen zum Preprocessing werden in Abschnitt 5 vorgestellt.

4. Generieren der Branch-And-Cut-Time Bucket Formulation

In diesem Kapitel wird die *Branch-And-Cut-Time Bucket Formulation*, namentlich die Einbettung der *TSPTW-TBR* in ein spezielles Branch-And-Cut-Framework (im Folgenden auch als *TSPTW-TBR* -Branch-And-Cut-Framework bezeichnet), in ihren Komponenten, sowie den einzelnen Schritten ihrer Konstruktion, vorgestellt. Insbesondere wird die *TSPTW-TBR* durch die Ergänzung des besagten Frameworks gleichwertig mit der *TSPTW-TBF* .

4.1. Erstellen der Time Bucket Relaxation

Im folgenden seien die von *Dash et al.* [5] eingeführten Elemente der grundlegenden Generierung einer "günstigen" *TSPTW-TBR*, sowie das problemspezifische Branch-And-Cut-Framework, in welches selbige einfließt um in die Branch-And-Cut-*Time Bucket Formulation* überführt werden zu können, erläutert. Überdies werden mögliche Modulierungen und Erweiterungen vorgestellt.

4.1.1. Der Bucket-Graph

Der Bucket-Graph $G' = \{\mathfrak{B}, A_{\mathfrak{B}}\}$ ist ein Hilfsgraph, den *Dash et al.* [5] unter anderem einführen, um die *TSPTW-TBR* einem ergänzenden, Bucket-spezifischen Preprocessing zu unterziehen und selbige somit, in der Regel empfindlich, weiter zu verkleinern. Hierbei stellen die Gesamtheit der Buckets die Knoten, und die Optionen des Reisens von einer Bucket $b_i \in B_i$ zu einer anderen Bucket $b_j \in B_j$, mit $i \in V \setminus \{q\}$ und $j \in V \setminus \{p\}$, die Kanten des in Rede stehenden Graphen dar.

Zur formalen Definition des Bucketgraphen sei zunächst das Hilfs-Set

$$N_i(k, b) = \{\beta \in B_i : b \in \mathcal{I}_k(i, \beta)\}, \tag{4.1}$$

wobei $N_i(k, b) = NULL$ wenn $r_b + \theta_{k,i} > D_i$, deklariert. Gegeben eine Startbucket $b \in B_k$ bei Knoten k und ein Verlassen dieser Bucket (zum Zeitpunkt r_b) in Richtung des Knotens i, gibt das Set $N_i(k, b)$ gemäß Definition (4.1) nun also die eindeutig determinierte Bucket $b' \in B_i$ wieder, welche in diesem Fall erreicht werden würde, gegeben eine solche Bucket existiert. Präziser sind sodann $\mathfrak{B} = \bigcup_{i \in V} B_i$ und $A_{\mathfrak{B}} = \bigcup_{(i,j) \in A} \{(b, b') : b \in B_i, b' \in B_j, b' = N_j(i, b)\}$.

Überdies notiere, *Dash et al.* [5] folgend, $B(S) = \{b \in \mathfrak{B} : b \in B_i \; mit \; i \in S\}$ für eine gegebene Knotenmenge $S \subseteq V$ die Menge aller Buckets, welche den Knoten in S zugeordnet werden können. Außerdem notiere $V(B) = \{i \in V : B_i \subseteq B\}$ für eine gegebene Bucket-Menge $B \subseteq \mathfrak{B}$ die Menge aller Knoten, welche sämtliche ihrer Buckets in B enthalten haben. Es bezeichne zudem, angelehnt an den Fall der entsprechenden Knoten-Operatoren, für zwei Bucket-Mengen $B, B' \subset \mathfrak{B}$ der Operator $\delta(B, B')$ die Menge aller Bucket-Graph-Kanten, welche in B starten und in B' enden, formal ist also $\delta(B, B') = \{(b, b') \in A_{\mathfrak{B}} : b \in B, b' \in B'\}$. Schlüssiger Weise seien ferner

$\delta^+(B) = \delta(B, \mathfrak{B} \setminus B)$ und $\delta^-(B) = \delta(\mathfrak{B} \setminus B, B)$. Zudem gelte für ein Bucketset $B \subseteq \mathfrak{B}$, dass $\overline{B} = \mathfrak{B} \setminus B$.

4.1.2. Das Initiale Bucket-Set

Nun stellt sich zunächst die Frage, wie ein initiales Set von Buckets, und somit ein initialer Bucket-Graph, zu bilden sind. *Dash et al.* [5] liefern hierzu den folgenden Vorschlag:

Wie bereits in Abschnitt 2.4.3.1 angesprochen, besteht die Möglichkeit, dass in einem Zeitfenster W_i eines Knotens $i \in V$ Zeitslots $t \in W_i$ existieren, welche in keiner zulässigen Lösung vom Salesman bereist werden können. Ein solcher Zeitslot sowie Aneinanderreihungen entsprechender Zeitslots sollen im Folgenden auch als "Löcher" bezeichnet werden. Während das Identifizieren sämtlicher Löcher einer *TSPTW*-Instanz $\mathcal{NP}$-schwer ist[1], so lautet indes eine hinreichende Bedingung für das Auffinden einer Teilmenge der in Rede stehenden Zeitslots $t \in W_i$ bei einem Knoten $i \in V \setminus \{p\}$, basierend auf der oben eingeführten Notation, zweifelsohne

$$\bigcup_{k \in \widehat{\pi}(i)} I_k(i,t) = \emptyset, \tag{4.2}$$

da, gegeben das Set $I_k(i,t)$ enthält, aggregiert über alle in Frage kommenden Vorgänger k von i, keinen einzigen Zeitslot, von welchem aus i zum Zeitslot t erreicht werden könnte, der Zeitslot t in einer zulässigen Tour folgerichtig niemals in Anspruch genommen werden wird.

Basierend auf eben diesem Kriterium werden nun für alle $i \in V \setminus \{p, q\}$ einer gegebenen *TSPTW*-Instanz deren Löcher identifiziert. Sodann werden die initialen Buckets der Knoten um diese identifizierten Löcher herum, und gemäß den in Abschnitt 2.4.3.1 aufgeführten Anforderungen, gebildet, indem alle aufeinanderfolgenden Zeitslots eines Knotens, welche nicht durch Löcher unterbrochen sind, zu einer Bucket zusammen gefasst werden. Korrespondierend dazu wird, gemäß seiner Definition in 4.1.1, der Bucket-Graph erstellt.

4.1.3. Die Bucket-Triangle-Inequality

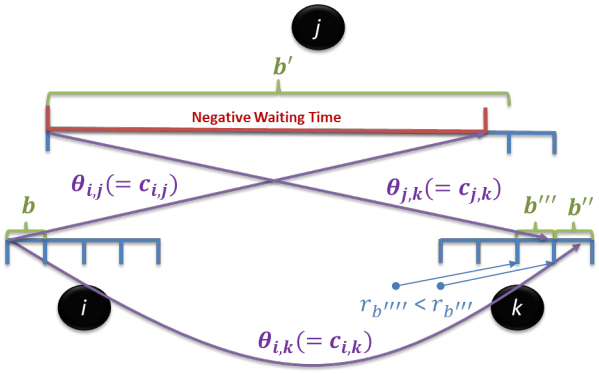

Abbildung 4.1.: Verbildlichung einer möglichen *BTI*-Verletzung

Bevor ein gültiges *Bucket-Preprocessing* durchgeführt werden kann, muss zunächst eine spezielle, metrische Eigenschaft des Bucket-Graphen gewährleistet werden. *Dash et al.* [5]

[1] siehe hierzu [5]

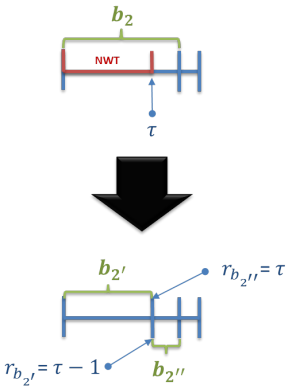

Abbildung 4.2.: Verbildlichung der Behebung einer *BTI*-Verletzung

führen hierzu den Begriff der sogenannten "*Bucket-Triangle-Inequality*" (*BTI*) ein.

Gegeben ein Bucket-Graph $G' = \{\mathfrak{B}, A_{\mathfrak{B}}\}$ kann folgende Problematik auftreten: Seien $i, j, k \in V$ drei unterschiedliche Knoten des ursprünglichen Graphen G und $b \in B_i, b' \in B_j$ sowie $b'', b''' \in B_k$ vier Buckets dieser Knoten, so können drei Kanten (b, b'), (b', b'''),$(b, b'') \in A_{\mathfrak{B}}$ des Bucket-Graphen existieren, während zugleich $r_{b'''} < r_{b''}$ gilt. Dies bedeutet, dass einem Salesman, welcher von i nach k reist und dabei einen Bogen über j schlägt, gegeben er startet in i bei Bucket b, eine Ankunftsbucket b''' bei k zugeordnet würde, welche strikt früher ist, als Bucket b'' bei k (namentlich ist $r_{b'''} < r_{b''}$), die der Salesman erreicht hätte, wenn er in der selben Ausgangssituation direkt von i nach k gereist wäre. Somit könnte er im ersten Fall den Knoten k früher verlassen, obwohl er hierbei einen Umweg in Kauf genommen hätte und damit, im Vergleich zu Fall 2, höchstens später hätte ankommen und weiterreisen dürfen (da die $\theta_{i,j}$ gemäß Annahme metrisch sind).

Diese Verletzung kann auftreten, wenn die *negative waiting time* bei Knoten j sehr groß ist und hierdurch der Zeitgewinn $r_b + \theta_{i,j} - r_{b'}$ den zeitlichen Distanzunterschied $\theta_{i,j} + \theta_{j,k} - \theta_{i,k}$ überkompensiert. In diesem Fall soll im Folgenden auch von einer *Bucket-Triangle-Inequality*-Verletzung oder kurz *BTI*-Verletzung, gesprochen werden. In Abbildung 4.1 sei eine solche mögliche *BTI*-Verletzung, angelehnt an das oben beschriebene Szenario, noch einmal veranschaulicht.

Um sicher zu stellen, dass sämtliche Kanten des Bucket-Graphen der *Bucket-Triangle-Inequality* gehorchen, werden zunächst, basierend auf der formalen Definition, nach einander alle *BTI*-Verletzungen in G' identifiziert und sodann wie folgt behoben: Seien die an der *Bucket-Triangle-Inequality*-Verletzung beteiligten Knoten und Buckets definiert wie im obigen Fall, so wird die Verursacher-Bucket b' in zwei neue Buckets b'_1 und b'_2 aufgeteilt, welche ihrerseits die Situation der entstehenden *negative waiting time* eliminieren. Hierzu wird, mit $\tau = r_b + \theta_{i,j}$ als Ankunftszeitpunkt von i bei j, die Bucket $b' = [r_{b'}, d_{b'}]$ in die Buckets $b'_1 = [r_{b'}, \tau - 1]$ und $b'_2 = [\tau, d_{b'}]$ gesplittet. Gegeben ein Starten von Bucket b bei i, wird als Ankunftsbucket bei j nun b'_2 zugeordnet werden, womit die *negative waiting time* $r_b + \theta_{i,j} - r_{b'}$ auf $r_b + \theta_{i,j} - r_{b'_2} = 0$ reduziert, und die *BTI*-Verletzung folglich beseitigt ist. In Abbildung 4.2 soll der gerade beschriebene Teilungsprozess noch einmal visualisiert werden.

Es bleibt auszuführen, dass durch das Beseitigen einer *BTI*-Verletzung nicht selten eine neue entsteht[2], was es nötig machen kann, sehr oft über das oben beschriebene Vorgehen

[2] wie die Experimente im Rahmen dieser Arbeit gezeigt haben

zu iterieren, bis die Kanten in G' zur Gänze der *Bucket-Triangle-Inequality* unterliegen (dieser Gesamtprozess soll von nun an auch als "*BTI-Cleaning*"bezeichnet werden). Da *jeder* dieser besagten Iterations-Schritte des *BTI-Cleanings* hinsichtlich seiner Laufzeitkomplexität, weil jedes Bucket-Graph-Knotentripel geprüft werden muss, augenscheinlich in $O(n^3(|b_{AVG}|)^3)$ liegt, wobei n die Anzahl der Knoten und $|b_{AVG}|$ die (mit jedem Durchgang wachsende) durchschnittliche Anzahl an Buckets pro Knoten bezeichnen soll, kann der komplette Prozess des *BTI-Cleanings* folglich sehr zeitkonsumierend ausfallen.

Eine Sicherstellung der *BTI*-Gültigkeit ist trotz des mit ihr verbundenen Rechenaufwandes unabdingbar, da sie nicht nur eine notwendige Voraussetzung für die Anwendbarkeit des Buckte-spezifischen Preprocessings darstellt, sondern ebenfalls eine erhebliche Verstärkung der Dualen Schranken generiert, was diverse Experimente von *Dash et al.* [5] belegen und überdies intuitiv einleuchtend ist, nicht zuletzt da das Beseitigen einer *BTI*-Verletzung stets mit der gezielten Eliminierung einer (potentiell großen) *negative waiting time* einhergeht.

4.1.4. Das Bucket-Preprocessing

Ziel des von *Dash et al.* [5] vorgestellten Bucket-Preprocessings ist es, in analoger Vorgehensweise zu dem in Abschnitt 3.2 beschriebenen Verfahren des allgemeinen Preprocessings, basierend auf gegebenen Präzedenzkenntnissen möglichst viele Kanten des Bucket-Graphen zu löschen.

4.1.4.1. Bucket-Knoten-Präzedenzen

Hierzu seien die in Abschnitt 3.1 eingeführten Präzedenznotationen zunächst auf Bucket-Knoten-Relationen erweitert:

Gegeben sei ein Knoten $i \in V$ welcher, gesetzt den Fall, dass Knoten $k \in V \setminus \{i\}$ zur Bucket $b \in B_k$ erreicht wird, in jeder zulässigen Tour *nach* diesem Knoten k besucht werden muss, so spricht man davon, dass die Bucket b den Knoten i preceeded. Man schreibt auch $b \prec i$. Symmetrischer Weise sei ein Knoten $i \in V$ gegeben, welcher, gesetzt der Fall, dass Knoten $j \in V \setminus \{i\}$ zur Bucket $b \in B_j$ erreicht wird, in jeder zulässigen Tour *vor* diesem Knoten j besucht werden muss. So spricht man davon, dass der Knoten i die Bucket b preceeded. Man schreibt auch $i \prec b$.

Initiale Präzedenzerkenntnisse über die Bucket-Knoten-Relationen lassen sich, zum Teil angelehnt an das Vorgehen in 3.1, wie folgt ableiten:

Zunächst wird $b \prec i$ für alle $b \in B_k$ gesetzt, wenn $k \prec i$. Diese Deklaration ist einzusehen, da gegeben k preceeded den Knoten i, erst recht jede einzelne Bucket $b \in B_k$ bei k den Knoten i preceeden muss. Auf symmetrische Argumentation hin wird $i \prec b$ für alle $b \in B_j$ gesetzt, gegeben $j \succ i$. Überdies gilt, für $i, j, k \in V$ und $b \in B_i$:

$$j \prec b \quad wenn \quad r_b + \theta_{i,j} > D_j \tag{4.3}$$

$$b \prec j \quad wenn \quad R_j + \theta_{j,i} > d_b \tag{4.4}$$

$$b \prec k \quad wenn \quad b \prec j \wedge j \prec k \tag{4.5}$$

$$j \prec b \quad wenn \quad j \prec k \wedge k \prec b \tag{4.6}$$

Gilt $r_b + \theta_{i,j} > D_j$, kann nach einem Verlassen der Bucket b Knoten j offenbar in keinem Fall mehr pünktlich erreicht werden, gegeben, er wird nach Bucket b besucht. Folgerichtig muss er vor Bucket b besucht werden. Eben dies besagt (4.3). Symmetrischer Weise legitimiert sich (4.4).

Insbesondere besitzen die von (4.3) und (4.4) implizierten Schlussfolgerungen nur dann Gültigkeit, wenn alle Kanten-Relationen in $A_{\mathfrak{B}}$ der *Bucket-Triangle-Inequality* gehorchen. Ansonsten könnte beispielsweise im Fall von Zuweisung (4.3) nicht ausgeschlossen werden, dass sich ein dritter Knoten $k \in V \setminus \{i, j\}$ mit einer Bucket $b' \in B_k$ findet, über welche der Weg von i nach j abgekürzt werden könnte, sodass der Zielknoten j doch noch vor seiner *Deadline* D_j erreicht würde. Eine analoge Argumentation verlangt die Gültigkeit sämtlicher *Bucket-Triangle-Inequalities* für die Anwendung von Deklaration (4.4).

(4.5) und (4.6) basieren auf derselben logischen Implikationsstruktur wie die Zuweisung (3.4).

Über die Deklarationen (4.3) bis (4.6) wird sodann iteriert, bis keine neuen Präzedenzerkenntnisse mehr gefunden werden.

4.1.4.2. Bucket-Graph-Reduktion

Sind die Präzedenzerkennisse gesammelt, so kann der Bucket-Graph G' mittels der folgenden Kanteneliminationsregeln, welche ebenfalls eine perfekte Analogie zu denen des allgemeinen Preprocessings aufweisen, verkleinert werden:

Wenn $b \prec j$ für irgendeine Bucket $b \in B_i$ und zwei Knoten $i, j \in V$, so können alle Bucket-Graph-Kanten $(b', b) \in A_{\mathfrak{B}}$ mit $b' \in B_j$ aus G' gelöscht werden, da sie sämtlich ein Bereisen des Knotens j *vor* der Bucket b implizieren. Symmetrischer Weise, wenn $j \prec b$ für irgendeine Bucket $b \in B_i$ und zwei Knoten $i, j \in V$, so sind sämtliche Bucket-Graph-Kanten $(b, b') \in A_{\mathfrak{B}}$ mit $b' \in B_j$ aus G' löschbar.

Überdies, gegeben zwei Buckets $b \in B_i$ und $b' \in B_j$, mit $i, j \in V$, und ein dritter Knoten $k \in V \setminus \{i, j\}$, kann die Kante (b, b') aus G' gelöscht werden, wenn

$$b \prec k \wedge k \prec b' \tag{4.7}$$

$$(k \prec b \vee k \prec b' \vee r_b + \theta_{i,j} + \theta_{j,k} > D_k) \wedge (b \prec k \vee b' \prec k \vee R_k + \theta_{k,i} + \theta_{i,j} > D_j) \tag{4.8}$$

Die Regel (4.7) begründet sich analog zu der Regel (3.5), Regel (4.8) indes ist die bucketspezifische Subsumtion von Regel (3.6) und legitimiert sich dementsprechend.

4.1.5. Die Bucket-Refinement-Heuristik

Offensichtlich wächst die Größe der *TSPTW-TBR* (ceteris paribus, insbesondere unter Vernachlässigung der Effekte des *Bucket-Preprocessings*) quadratisch in der Gesamtanzahl $|B|$ der modellierten Buckets, bedingt durch eine Vermehrung der Spalten getrieben durch die z_i^b- und $y_{i,j}^b$-Entscheidungs-Variablen, sowie einer Vermehrung der Zeilen getrieben durch die Restriktionen (2.16) und (2.17). Jedoch ist ebenfalls unmittelbar einzusehen, dass eine *TSPTW-TBR* mit einer größeren Anzahl zugrunde liegender (und somit im Durchschnitt kleinerer) Buckets weniger anfällig für die in Abschnitt 2.4.3.3 beschriebene Problematik der Subtouren und der Unzulässigen Pfade ist, da sich mit steigender Bucket-Anzahl (namentlich $|\mathfrak{B}|$) und sinkender durchschnittlicher Bucket-Größe (namentlich $\sum_{b \in |\mathfrak{B}|} \frac{d_b - r_b}{|\mathfrak{B}|}$) eine zunehmende qualitative Konvergenz gegen die *TSPTW-TIF* einstellt. Insbesondere bedeutet dies ebenfalls, dass eine feinere Bucket-Struktur (ceteris paribus) sowohl im B&C-Wurzelknoten, als auch im Verlauf des Branch-And-Cut-Verfahrens tendenziell bessere Duale Schranken liefern sollte. Ist ein initiales Bucket-Set ermittelt und dem *BTI-Cleaning* , sowie dem *Bucket-Preprocessing* unterzogen worden, gilt es somit, die erstellte Bucket-Unterteilung möglichst "geschickt" zu verfeinern.

Es ergibt sich die Frage, wie eine günstige Bucket-Verfeinerungs-Methodik zu gestalten ist, da, wie oben angesprochen, jede zusätzliche Bucket die resultierende ILP-Formulierung empfindlich vergrößert und ihre Generierung somit vermieden werden sollte, falls sie keinen hinreichenden Beitrag zu einer Verbesserung des Modells hinsichtlich Stärke der LP-Relaxation und Verhalten im Branch-And-Cut-Baum verspricht. *Dash et al.* [5] liefern hierzu den folgenden, heuristischen Vorschlag:

Zunächst wird die zu dem Initialen Bucket-Set und -Graphen korrespondierende *TSPTW-TBR* erstellt. Insbesondere gilt nun somit $\beta \in \mathcal{I}_k(i,b) \Leftrightarrow (\beta, b) \in G'$ (was zwar in *Dash et al.* an keiner Stelle ausdrücklich erwähnt wird, aber, ihren Erläuterungen folgend, schlüssig erscheint), wodurch, wie oben bereits diskutiert, in aller Regel eine Verkleinerung des Sets $\mathcal{I}_k(i,b)$, und damit der Modellformulierung, erzielt wird. Sodann wird die zugehörige LP-Relaxation des resultierenden *TSPTW-TBR*-ILPs gelöst (konkret wird also die Nebenbedingung (2.19) ersetzt durch $x_{i,j},\ y^b_{i,j},\ z^b_i \in [0,1]\ \forall\ i \in V, \forall\ (i,j) \in A, \forall\ b \in B_i$). Sei $(\tilde{x}, \tilde{y}, \tilde{z})$ die daraus resultierende, potentiell fraktionale Belegung der Entscheidungsvariablen. Für diese kann nun insbesondere ein Set $\tilde{B} = \{b \in \mathfrak{B} : \tilde{z}^i_b > 0, b \in B_i, i \in V\}$ fraktional aktiver Buckets identifiziert werden.

Nur diejenigen Buckets $b \in \tilde{B}$, die nicht bereits auf die Größe eines einzelnen Zeitslots reduziert sind, für die also zusätzlich $r_b \neq d_b$ gilt, werden weiter geteilt, und zwar in je zwei neue Buckets b^1 und b^2, so dass $b^1 = [r_b, \tau - 1]$ und $b^2 = [\tau, d_b]$ mit einem entsprechenden Trennzeitpunkt $\tau \in [r_b + 1, d_b]$. Diese Selektion scheint plausibel, da nicht einzusehen ist, warum eine Bucket, welche für die aktuelle LP-Lösung, und somit die resultierende Duale Schranke, unmittelbar völlig irrelevant ist (namentlich eine Bucket $b' \notin \tilde{B}$) die zugehörige LP-Relaxation verstärken sollte, wenn sie verfeinert wird.

Ein möglichst günstiger Trennzeitpunkt τ für die zu teilende Bucket b soll nun wie folgt gewählt werden: Basierend auf der Erkenntnis, dass große *negative waiting times*, sowohl in Hinsicht auf potentielle Subtour-Bildungen, als auch auf Unzulässige Pfade, die ILP-Formulierung empfindlich schwächen, sollen die aus der Teilung resultierenden, neuen Buckets so beschaffen sein, dass die durchschnittlich bei ihnen entstehenden *negative waiting times* in Summe minimiert werden. Formal gestalten *Dash et al.* [5] diesen Ansatz, indem sie die fraktionalen $\tilde{z}^i_b$ zunächst in ihre Zeitslotkomponenten $\tilde{z}^i_t$ zerlegen:

$$\tilde{z}^i_t = \sum_{(k,b') \in \delta^-_{bkt \to t}(i,t)} \tilde{y}^{b'}_{k,i} \quad \forall\ t = r_b + 1, ..., d_b \tag{4.9}$$

Hierbei sei der Hilfsoperator $\delta^-_{bkt \to t}(i,t)$ so definiert, dass er alle eindeutig determinierten Knoten-Bucket-Tupel (k, b') liefert, welche als Vorgängerpunkte resultieren, gegeben Knoten i wird zum Zeitpunkt t erreicht und es wurde von Knoten k direkt nach i gereist. Formal ist also $(k, b') \in \delta^-_{bkt \to t}(i,t)$ genau dann, wenn $b' \in B_k, (b', b) \in A_{\mathfrak{B}}$ und $r_{b'} + \theta_{k,i} = t$.

Der verbleibende Fall $t = r_b$ wird über die Gegensumme ermittelt: Folglich ist $\tilde{z}^i_t = z^b_i - \sum_{t'=r_b+1}^{d_b} \tilde{z}^t_i$ für $t = r_b$.

Die so berechneten $\tilde{z}^i_t$ können nun als fraktionale Aktivitätsniveaus der Zeitslots interpretiert, und somit als Gewichtungsfaktoren eingesetzt werden, um je Bucket $b \in \tilde{B}$, die effektiv von der Lösung implizierte *negative waiting time* (gemittelt) zu beziffern. Konkret werden die *negative waiting times* $t - r_b$, die jedem einzelnen Zeitslot $t \in [r_b, d_b]$ zugeordnet werden können, mit ihrem jeweiligen Aktivitätsniveau $\tilde{z}^i_t$ multipliziert und sodann aggregiert. Formal ist die fraktional gemittelte *negative waiting time* bei Bucket $b \in B_i$

also

$$\widetilde{NWT}_b = \sum_{t=r_b}^{d_b} (t - r_b)\tilde{z}_i^t. \tag{4.10}$$

Gegeben die Bucket $b = [r_b, d_b]$ wird an der Stelle τ in die Buckets $b^1 = [r_b, \tau - 1]$ und $b^2 = [\tau, d_b]$ gesplittet, so sind die zu der aktuellen Lösung korrespondierenden *negative waiting times* an den beiden neuen Buckets, das obigen Schema subsumierend, also wie folgt zu berechnen:

$$\widetilde{NWT}_{b^1} = \sum_{t=r_b}^{\tau-1} (t - r_b)\tilde{z}_i^t \tag{4.11}$$

$$\widetilde{NWT}_{b^2} = \sum_{t=\tau}^{d_b} (t - \tau)\tilde{z}_i^t \tag{4.12}$$

Ein günstiger Trennzeitpunkt τ für Bucket b ist nun gemäß der ursprünglichen Intention jener, welcher die Summe der beiden neuen NWTs, namentlich die Hilfsfunktion

$$\alpha(b, \tau) = \sum_{t=r_b}^{\tau-1} (t - r_b)\tilde{z}_i^t + \sum_{t=\tau}^{d_b} (t - \tau)\tilde{z}_i^t, \tag{4.13}$$

minimiert. Da der Werte-Raum über τ diskret ist, kann der gesuchte Trennzeitpunkt in $O(d_b - r_b)$ ermittelt werden.

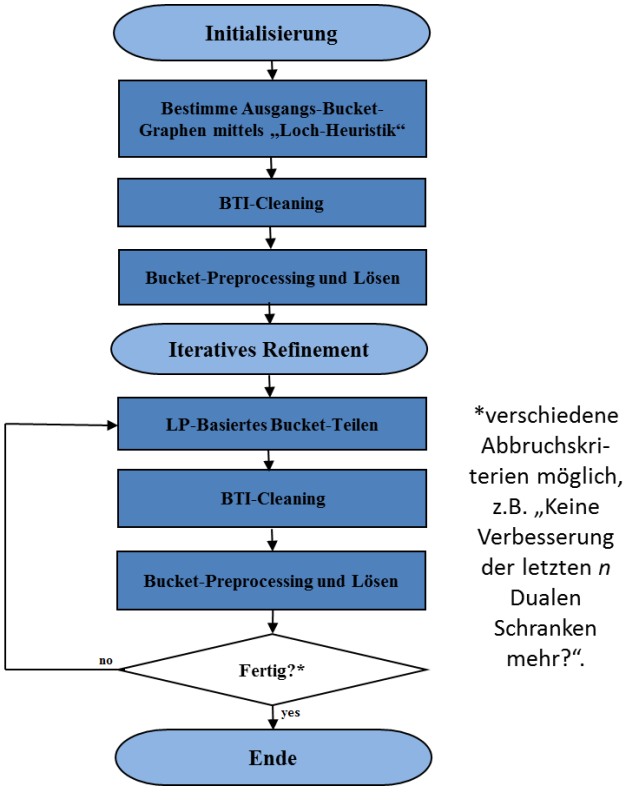

Abbildung 4.3.: Verbildlichung der iterativen Bucket-Refinement-Heuristik

Über das in diesem Abschnitt beschriebene Bucket-Teilungs-Vorgehen wird sodann iteriert, bis ein bestimmtes Abbruchkriterium erfüllt ist, beispielsweise eine ausbleibende Verbesserung der Dualen Schranke (namentlich des aktuellen optimalen LP-Zielfunktionswertes) seit den letzten 5 Durchgängen.

In Abbildung 4.3 sei die so entstehende, gesamte Prozedur der iterativen Bucket-Refinement-Heuristik noch einmal anhand eines Flussdiagramms veranschaulicht und zusammengefasst.

Die Effizienz dieser Heuristik, gemessen über ihren Erfolg bei dem Versuch, die resultierende *TSPTW-TBR* "so groß wie nötig, aber so klein wie möglich" zu halten, ist offenbar stark darauf angewiesen, dass bereits die Lösung der initialen *TSPTW-TBR* ein guter "Schätzer" für die Struktur der tatsächlichen, optimalen Lösung ihrer zugrunde liegenden TSPTW-Intsanz ist, da sich die Bucket-Teilung jeweils vollständig an einer NWT-Minimierung der aktuellen LP-Lösung orientiert. Weicht selbige in ihrer Struktur aber erheblich vom tatsächlich gesuchten, ganzzahligen Optimum ab, werden viele Buckets umsonst gesplittet und das Modell somit unnötig vergrößert. Dieser Effekt wird forciert durch die Tatsache, dass durch unnötige Aufsplittung von Buckets, ebenfalls neue (bezogen auf die tatsächlich gesuchte Lösung unnötige) *BTI*-Verletzungen entstehen, welche ihres Zeichens im Rahmen des *BTI-Cleanings* zu weiteren, überflüssigen Bucket-Vermehrungen führen. Es sei allerdings angemerkt, dass *Dash et al.* [5] auf den von ihnen verwendeten Testinstanzen diesbezüglich keine nennenswerten Komplikationen beobachten, und dass die beschriebene Bucket-Teilungs-Heuristik (bei ähnlicher *TSPTW-TBR*-Größe) insbesondere signifikant bessere Hebungen der Dualen Schranken generiert, als die Anwendung vergleichsweise trivialerer Teilungs-Ansätze wie beispielsweise das gleichmäßige zerlegen aller Zeitfenster in eine bestimmte Anzahl n_b von Buckets pro Knoten.

4.2. TBR-Branch-And-Cut-Framework

Ist die finale *TSPTW-TBR* erstellt, übergeben *Dash et al.* diese einem Branch-And-Cut-Framework, welches in jedem Knoten sogenannte *Bucket Sequential Ordering Polytope Inequalities* (siehe hierzu Abschnitt 4.2.1) sowie spezielle *infeasible path-Inequalities* (siehe hierzu Abschnitt 4.2.3) separiert und Incumbents, welche keine Zulässigkeit für die zugrunde liegende TSPTW-Instanz aufweisen, verwirft. Überdies wird mittels einer problemspezifischen Primalheuristik in jedem Knoten des B&C-Baumes versucht, basierend auf der aktuellen LP-Variablenbelegung, eine für die zugrunde liegende TSPTW-Instanz zulässige, ganzzahlige Lösung zu konstruieren (siehe hierzu Abschnitt 4.2.4). Diese Branch-And-Cut-Framework-unterstützte *TSPTW-TBR* ist nun effektiv gleichwertig mit der *TSPTW-TBF*.

Im Folgenden seien die in Rede stehenden Elemente des *TSPTW-TBR*-B&C-Frameworks sowie denkbare neue Erweiterungen der selbigen vorgestellt.

4.2.1. Bucket Sequential Ordering Polytope Inequalities

Zur Elimination von Subtouren im B&C-Framework entwickeln *Dash et al.* [5] eine Bucket-spezifische Variante der von *Balas et al.* [26] vorgestellten *Sequential Ordering Polytope-Inequalities* (*SOP-Inequalities*) für das *asymmetrische precedence-constrained TSP*[3].[4] Ein Generelles Zurückgreifen auf diese Restriktions-Klasse ist, angesichts der Tatsache, dass

[3] siehe hierzu [26]

[4] hierbei ist das *Sequential Ordering Polytope* gerade jener Lösungsraum, welcher durch das *asymmetrische precedence-constrained TSP* beschrieben wird. Siehe hierzu auch [26]

das TSPTW wie in Abschnitt 3.1 bereits ausgenutzt wird, a priori eine große Menge an Knoten-Präzedenzen implizieren kann, sinnvoll.[5] Technisch gesehen ist dieser Ansatz durch die Tatsache legitimiert, dass der Lösungsraum des *TSPTW-Polytops* im Lösungsraum des *Sequential Ordering Polytops* enthalten ist und somit die Chance besteht, mittels einer facettendefinierenden *Inequality* für das *Sequential Ordering Polytope* ebenfalls eine Facette des *TSPTW-Polytops* freizulegen.[6]

Im Folgenden seien die besagten *SOP-Inequalities* vorgestellt, um aus ihnen sodann die in das B&C-Framework einfließenden *Bucket-SOP-Inequalities* abzuleiten.

4.2.1.1. π- und σ-Cuts

Dash et al. [5] folgend sei einführend zunächst die Knoten-Präzedenznotation auf Sets von Knoten erweitert:

Gegeben zwei Knotenmengen $S, S' \in V$, so gilt $S \prec S'$, wenn *jeder* Knoten in S *jeden* Knoten in S' preceeded. Überdies sei für eine Knotenmenge $S \subseteq V$ das Hilfsset $\pi(S) = \{i \in V : i \prec j \; für \; irgendein \; j \in S\}$ die Menge aller Knoten i, welche *mindestens einen* Knoten j in S preceeden. Insbesondere kann i selbst in S enthalten sein. Ferner kann ein solches $i \in S \cap \pi(S)$ in einer zulässigen Tour niemals der letzte Knoten aus der Menge S sein, welcher von dieser Tour besucht wird, da mindestens ein Knoten $j \in S$ existiert, der ein obligatorischer Nachfolger von i ist. Symmetrischer Weise sei das Hilfsset $\sigma(S) = \{i \in V : i \succ j \; für \; irgendein \; j \in S\}$ die Menge aller Knoten i, welche *mindestens einen* Knoten j in S succeeden. Insbesondere kann i selbst in S enthalten sein. Überdies kann ein solches $i \in S \cap \sigma(S)$ in einer zulässigen Tour niemals der erste Knoten aus der Menge S sein, welcher von dieser Tour besucht wird, da mindestens ein Knoten $j \in S$ existiert, der ein obligatorischer Vorgänger von i ist.

Schließlich seien für eine Knotenmenge $S \subseteq V$ die Operatoren

$$\delta_\pi(S) = \delta(S \setminus \pi(S), \overline{S} \setminus \pi(S)) \tag{4.14}$$

$$\delta_\sigma(\overline{S}) = \delta(\overline{S} \setminus \sigma(S), S \setminus \sigma(S)) \tag{4.15}$$

definiert. Namentlich liefert $\delta_\pi(S)$ die Menge aller Kanten, welche aus der Menge S herausgehen und dabei keines ihrer beiden Enden in $\pi(S)$ haben. Somit ist es die Menge $\delta^+(S)$, bereinigt um alle Kanten welche, basierend auf den Präzedenzkenntnissen und der obigen Argumentation, nicht die letzten Kanten einer zulässigen Tour sein können, die aus der Knotenteilmenge S herausgehen. Analog hierzu erläutert sich der Operator $\delta_\sigma(\overline{S})$.

Sodann, gegeben ein Knotenset $S \subseteq V \setminus \{q\}$, definieren sich die ersten beiden *SOP-Inequalities* wie folgt:

$$(\pi - Inequalities) \qquad \sum_{(i,j) \in \delta_\pi(S)} x_{i,j} \geq 1 \tag{4.16}$$

$$(\sigma - Inequalities) \qquad \sum_{(i,j) \in \delta_\sigma(\overline{S})} x_{i,j} \geq 1 \tag{4.17}$$

Eine zulässige *TSPTW*-Tour ist, wie in Abschnitt 2.1.2 bereits behandelt, ein Hamiltonweg von p nach q durch G. Somit muss jede Knotenteilmenge $S \subseteq V \setminus \{q\}$ in einer zulässigen Tour mindestens einmal verlassen werden. Insbesondere muss in jeder zulässigen Tour eine Kante $(i, j) \in A$ mit $i \in S$ und $j \in \overline{S}$ existieren, welche die letzte Kante darstellt,

[5] vergleiche hierzu auch die entsprechenden Argumentationen in [5]
[6] siehe hierzu beispielsweise [25]

die aus S herausführt, sodass also die Resttour in $\overline{S}$ verbleibt. Die Menge aller Kanten, welche, basierend auf den Präzedenzkenntnissen, hierfür in Frage kommen, liefert, wie oben erörtert, gerade der Operator $\delta_\pi(S)$. Folglich ist die Restriktion (4.16) gültig für die *TSPTW-BMF* und damit auch für die *TSPTW-TBR*. In analoger Weise erläutert sich die Gültigkeit der Restriktion (4.17) für die *TSPTW-BMF* und damit auch für die *TSPTW-TBR*.

Da den Restriktionen (2.21), namentlich $\sum_{(i,j)\in\delta^+(S)} x_{i,j} \geq 1$, in der Summation der Operator $\delta^+(S)$ zu Grunde liegt und für jedes $S \subset V$ klarerweise sowohl $\delta_\pi(S) \subseteq \delta^+(S)$ als auch $\delta_\sigma(\overline{S}) \subseteq \delta^+(\overline{S})$ gilt, ist einzusehen, dass die Restriktionen (4.16) und (4.17) stets eine höchstens kleinere linke Seite als die entsprechenden Restriktionen in (2.21) aufweisen, während in allen Fällen die rechte Seite gleich groß ist, womit die π- und σ-*Inequalities* die klassischen SECs für das *TSPTW* dominieren, wobei die Rede von der Dominanz einer *Constraint A* über eine andere *Constraint B* immer dann sein soll, wenn die *Constraint A* nachweislich stets mindestens genau so viel Anteil des integral unzulässigen Lösungsraumes verbietet (beziehungsweise "abschneidet"), wie die *Constraint B*.

Überdies konnten *Balas et al.* [32] zeigen, dass die π- und σ-*Inequalities* unter gewissen Voraussetzungen sogar facettendefinierend für das *Sequential Ordering Polytope* sind.

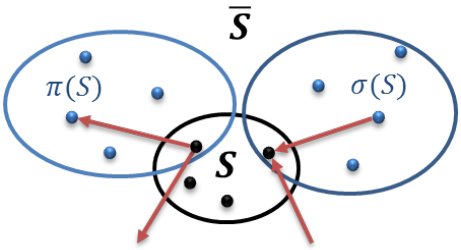

Abbildung 4.4.: Schematische Verbildlichung der Mengen-Konstellationen und ausgeschlossenen Kanten-Typen eines π- und eines σ-Cuts

In Abbildung 4.4 seien die Mengen-Konstellationen eines π- und eines σ-Cuts der Menge S, so wie je zwei rot gekennzeichnete, über den Operator $\delta_\pi(S)$ beziehungsweise $\delta_\sigma(\overline{S})$ in der *Inequality* ausgeschlossene Kanten-Typen des Schnittes $\delta^+(S)$ beziehungsweise $\delta^+(\overline{S})$, schematisch skizziert.

4.2.1.2. (π,σ)-Cuts

Die Klasse der *SOP-Inequalties* lässt sich nun wie folgt um eine dritte Restriktions-Familie erweitern:

Für jedes geordnete set $P = (v_1, ..., v_h)$ von Knoten in V sei $\theta(P) = \sum_{i=1}^{h-1} \theta_{v_i,v_{i+1}}$ als die Netto-Gesamtreisezeit, ohne Berücksichtigung von Zeitfenstereffekten, des zu diesem Set korrespondierenden Teil-Pfades definiert. Sodann, gegeben zwei disjunkte Knotenmengen X, Y mit $X \prec Y$ sei

$$Z = \{k \in V \setminus (X \cup Y) : \exists\ i \in X, j \in Y\ \ mit\ \ R_i + \theta(i,k,j) > D_j\}, \tag{4.18}$$

womit der Teil-Pfad $P = (i, k, j)$, wenn $k \in Z$, niemals in einer zulässigen Tour enthalten sein kann.

Lemma 2. Die Erfülltheit der Ungleichung $R_{v_1} + \sum_{(v_i,v_j)\in P} \theta_{v_i,v_j} > D_h$ ist eine Hinreichende Bedingung für die Unzulässigkeit des (Teil-)Pfades $P = (v_1, v_2, ..., v_h)$.

Beweis. R_{v_1} ist die generell frühestmögliche Startzeit bei Knoten v_1 und die Summe $\sum_{(v_i,v_j)\in P} \theta_{v_i,v_j}$ eine untere Schranke auf die Reisedauer entlang des Pfades P. Durch die Berücksichtigung der Zwischenknoten-Zeitfenster und den daraus gegebenenfalls resultierenden *Deadline*-Überschreitungen wird der Pfad erst recht unzulässig, aus den daraus gegebenenfalls resultierenden Wartezeiten indes wird die tatsächliche Reisezeit höchstens größer, woraus das Lemma folgt. □

Ferner sei

$$Q = \{(u,v) \in A : \exists\ i \in X, j \in Y\ \ mit\ \ R_i + \theta(i,u,v,j) > D_j\}, \tag{4.19}$$

womit der Teil-Pfad $P = (i,u,v,j)$, namentlich ein Reisen von i nach j über die Kante (u,v), wenn $(u,v) \in Q$, gemäß Lemma 2 ebenfalls niemals in einer zulässigen Tour enthalten sein kann.

Schließlich sei $W = \pi(X) \cup \sigma(Y) \cup Z$. So definieren sich für jedes $S \subset V$, sodass $X \subseteq S$ und $Y \subseteq \overline{S}$, die sogenannten (π,σ)-*Inequalities* wie folgt:

$$((\pi,\sigma) - Inequalities) \qquad \sum_{(i,j)\in\delta(S\setminus W,\overline{S}\setminus W)\setminus Q} x_{i,j} \geq 1 \tag{4.20}$$

Einmal mehr gilt, dass in einer zulässigen *TSPTW*-Tour die Menge S mindestens einmal verlassen werden muss, und genau einmal, über eine Kante $(i,j) \in A$ mit $i \in S$ und $j \in \overline{S}$, zum letzten Mal zu verlassen ist. Diese Kante (i,j) darf gemäß den Erläuterungen zu (4.19) nicht in Q enthalten sein, da sonst mindestens ein Knoten j' in Menge Y und somit auch in Menge $\overline{S}$ existiert, der per Voraussetzung noch zu besuchen ist, jedoch nicht mehr rechtzeitig erreicht werden könnte. Auf analoge Argumentation hin darf die Kante (i,j) ihr Ende nicht in Z haben. Erst recht darf sie somit auch nicht ihren Anfang in Z haben. Überdies darf sie, korrespondierend zu den Argumentationen bezüglich der π- und σ-Inequalities, weder ihren Anfang, noch ihr Ende in den Mengen $\pi(X)$ und $\sigma(Y)$ haben. Daher ist die Restriktion (4.20) gültig für die *TSPTW-BMF* und somit auch für die *TSPTW-TBR*.

Im Gegensatz zu den π- und σ-*Inequalities* sind für die (π,σ)-*Inequalities* allerdings keine Einschränkungen bekannt, unter welchen selbige facettendefinierend für das *Sequential Ordering Polytope* sind.[7]

In Abbildung 4.5 seien die Mengen-Konstellationen eines (π,σ)-Cuts noch einmal schematisch verbildlicht. Auf eine qualitative Andeutung ausgeschlossener Kanten der Menge $\delta^+(S)$ ist zu Gunsten der Übersicht verzichtet.

4.2.1.3. Simple π_b- und Simple σ_b-Cuts

Basierend auf den in Abschnitt 4.2.1.1 eingeführten π- und σ-Cuts definieren *Dash et al.* [5] sodann die sogenannten π_b- und σ_b-Cuts.

Zunächst wird hierzu die Bucket-Knoten-Präzedenznotation ebenfalls auf Sets von Buckets

[7] interessanter Weise spielen die Bucket-spezifischen (π,σ)-*Inequalities* dieser Erkenntnis zum Trotz jedoch, laut den Resultaten von *Dash et al.* [5], im Rahmen der Schließung des Integralen GAPs eine deutlich größere Rolle, als die Bucket-spezifischen π- und die Bucket-spezifischen σ-*Inequalities*. Dies mag unter anderem darauf zurückzuführen sein, dass die (π,σ)-*Inequalities* die Zusammenhangs-Struktur der Lösung als Ganzes "agressiver" beeinflussen, als die π- und die σ-*Inequalities*, was im *TBR*-B&C-Framework ebenfalls eine entscheidende "Triebfeder" für die Verbesserung der Dualen Schranken und das schnellere Auffinden zulässiger ganzzahliger Lösungen darstellt

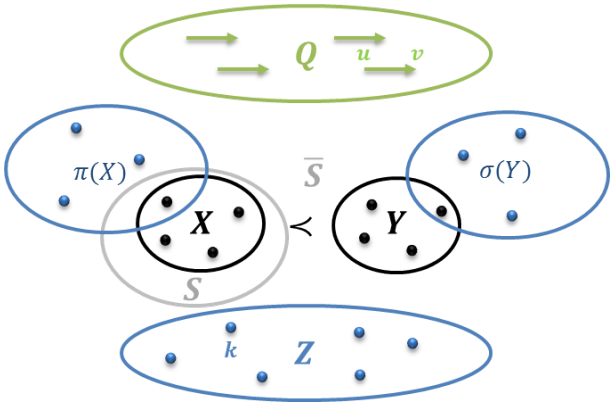

Abbildung 4.5.: Schematische Verbildlichung der Mengen-Konstellationen eines (π, σ)-Cuts

erweitert: Sei $B \subseteq \mathfrak{B}$ eine Menge an Buckets. So definiert sich der zum Operator $\pi(S)$ aus Abschnitt 4.2.1.1 korrespondierende Bucket-Operator $\pi(B)$ als $\pi(B) = \{b \in \mathfrak{B} : b \prec i \ mit \ i \in V \ und \ B_i \in B\}$. Konkret gibt er also die Menge aller Buckets $b \in \mathfrak{B}$ wieder, welche mindestens einen Knoten $i \in V$ preceeden, dessen Buckets seines Zeichens sämtlich in B enthalten sind. Symmetrischer Weise hierzu definiert sich der Bucket-Operator $\sigma(B)$ als $\sigma(B) = \{b \in \mathfrak{B} : i \prec b \ mit \ i \in V \ und \ B_i \in B\}$.

Analog zu den beiden Operatoren $\delta_\pi(S)$ und $\delta_\sigma(\overline{S})$ aus Abschnitt 4.2.1.1 definieren sich schließlich

$$\delta_\pi(B) = \delta(B \setminus \pi(B), \overline{B} \setminus \pi(B)) \tag{4.21}$$

$$\delta_\sigma(\overline{B}) = \delta(\overline{B} \setminus \sigma(B), B \setminus \sigma(B)). \tag{4.22}$$

Namentlich liefert $\delta_\pi(B)$ die Menge aller Bucket-Graph-Kanten, welche aus der Menge B herausgehen und dabei keines ihrer beiden Enden in $\pi(B)$ haben. Somit ist es, in Analogie zu Abschnitt 4.2.1.1, die Menge $\delta^+(B)$, bereinigt um alle Bucket-Kanten, welche, basierend auf den Bucket-Knoten-Präzedenzkenntnissen und der obigen Argumentation, in einer zulässigen Tour nicht die letzten Kanten sein können, welche die Menge B verlassen. Entsprechend hierzu erläutert sich der Operator $\delta_\sigma(B)$.

Definition Sei $B \subseteq \mathfrak{B}$, so dass mindestens ein Knoten alle seine Buckets in B enthalten hat und weder die Buckets des Startknotens p noch die des Endknotens q in B liegen, formal gelte also $B_t \subseteq B$ für irgendein $t \in V$ und $B_p \cup B_q \subseteq \overline{B}$. So definieren sich die ersten beiden *Bucket-SOP-Inequalities* wie folgt:

$$(\pi_b - Inequalities) \qquad \sum_{(i,j,b) \in \mu(\delta_\pi(B))} y^b_{i,j} \geq 1 \tag{4.23}$$

$$(\sigma_b - Inequalities) \qquad \sum_{(i,j,b) \in \mu(\delta_\sigma(\overline{B}))} y^b_{i,j} \geq 1 \tag{4.24}$$

Hierbei sei der Operator $\mu((b, b'))$ so beschaffen, dass er die eindeutige Abbildung von einer Kante $(b, b') \in A_\mathfrak{B}$ des Bucket-Graphen auf das zugehörige Tupel (i, j, b) einer Entscheidungs-Variablen $y^b_{i,j}$ aus der *TSPTW-TBR* liefert. Formal ist also $\mu : A_\mathfrak{B} \to V \times V \times B$ mit $\mu(b, b') = (i, j, b)$ genau dann wenn $b \in B_i$, $b' \in B_j$ und $b' = N_j(i, b)$.

In Anlehnung an *Dash et al.* [5] sei im Folgenden gezeigt, dass die π_b- beziehungsweise σ_b-*Inequalities* gültig für die *TSPTW-BMF* sowie die *TSPTW-TBR* sind und die π- beziehungsweise σ-*Inequalities* dominieren.

Satz 3. Die π_b-*Inequalities* und die σ_b-*Inequalities* sind gültig für die *TSPTW-BMF* und somit für die *TSPTW-Time Bucket Relaxation.*

Beweis. Der Beweis gestaltet sich, nach einer Subsumtion des Argumentationsraumes von G auf G', absolut analog zu den Ausführungen aus Abschnitt 4.2.1.1 betreffend die Gültigkeit der π- und σ-*Inequalities* für die *TSPTW-TBR* :

Offenbar korrespondiert eine gültige *TSPTW*-Tour mit einem Pfad durch den Bucket-Graphen G', welcher bei einer Bucket in B_p startet, sodann von jedem Knoten $i \in V \setminus \{p, q\}$ je genau eine Bucket $b \in B_i$ besucht und schließlich in einer Bucket von B_q endet.

Somit muss das Bucket-Set B, da $B_q \in \overline{B}$, in einer zulässigen Lösung mindestens einmal in Richtung $\overline{B}$ verlassen werden, um die Tour zu schließen. Sei, für zwei Knoten $i \in V(B)$ und $j \in V(\overline{B})$, $(b, b') \in \mathfrak{B}$, mit $b \in B_i$ und $b' \in B_j$, die letzte Bucket-Graph-Kante, welche von B nach $\overline{B}$ führt. So kann nicht gelten, dass $b \in \pi(B)$, da in diesem Fall mindestens ein Knoten $k \in V(B)$ existieren müsste, von welchem Knoten i ein obligatorischer Vorgänger ist und der somit noch besucht würden müsste. Dies ist aber ein Widerspruch zu der Annahme, dass (b, b') die letzte Kante ist, welche B und somit die Knotenmenge $V(B)$ verlässt. Erst recht kann daher ebenfalls nicht gelten, dass $b' \in \pi(B)$. Genau diesen Argumentationsinhalten genügt, wie oben ausgeführt, der Operator $\delta_\pi(B)$, weswegen die π_b-*Inequalities* gültig für die *TSPTW-BMF* und somit für die *TSPTW-TBR* sind.

Symmetrischer Weise muss das Bucket-Set B, da $B_p \in \overline{B}$, in einer zulässigen Lösung mindestens einmal von $\overline{B}$ aus besucht werden, um die Tour zu vervollständigen. Sei, für zwei Knoten $k \in V(\overline{B})$ und $i \in V(B)$, $(b, b') \in \mathfrak{B}$, mit $b \in B_k$ und $b' \in B_i$, die erste Kante, welche von $\overline{B}$ nach B führt. So kann nicht gelten, dass $b' \in \sigma(B)$, da in diesem Fall mindestens ein Knoten $j \in V(B)$ existieren müsste, von welchem Knoten i ein obligatorischer Nachfolger ist und der somit schon besucht worden sein muss. Dies ist aber ein Widerspruch zu der Annahme, dass (b, b') die erste Kante ist, welche $\overline{B}$ in Richtung B, und somit die Knotenmenge $V(B)$, verlässt. Erst recht kann daher ebenfalls nicht gelten, dass $b \in \sigma(B)$. Genau diesen Argumentationsinhalten genügt, wie oben ausgeführt, der Operator $\delta_\sigma(B)$, weswegen die σ_b-*Inequalities* gültig für die *TSPTW-BMF* und somit für die *TSPTW-TBR* sind. □

Satz 4. Die π_b-*Inequalities* dominieren die π-*Inequalities* und die σ_b-*Inequalities* dominieren die σ-*Inequalities.*

Beweis. Es sei eine π-*Inequality* mit dem zugrunde liegenden Knoten-Set $S \subseteq V \setminus \{p, q\}$ betrachtet. Sei $X = \delta(B(S) \setminus B(\pi(S)), B(\overline{S}) \setminus B(\pi(S)))$ das zum Kanten-Set $\delta_\pi(S)$ korrespondierende Bucket-Kanten-Set. Damit lässt sich die linke Seite der π-*Inequality* wie folgt umschreiben:

$$\sum_{(i,j)\in\delta_\pi(S)} x_{i,j} = \sum_{(i,j)\in\delta_\pi(S)} \sum_{b\in B_i} y^b_{i,j} = \sum_{(i,j,b)\in\mu(X)} y^b_{i,j} \tag{4.25}$$

Für die zu dem Knoten-Set S korrespondierende π_b-*Inequality* mit dem folglich zugrunde liegenden Bucket-Set $B(S)$ gilt indes $\delta_\pi(B(S)) = \delta(B(S) \setminus \pi(B(S)), B(\overline{S}) \setminus \pi(B(S)))$. Sei

$Y = \delta_\pi(B(S))$ Damit lässt sich die linke Seite der π_b-*Inequality* schreiben als:

$$\sum_{(i,j,b)\in\mu(\delta_\pi(B(S)))} y^b_{i,j} = \sum_{(i,j,b)\in\mu(Y)} y^b_{i,j} \tag{4.26}$$

Da die Menge an Buckets, welche alle den Knoten zuzuordnen sind, die mindestens einen Knoten in S preceeden, klarerweise eine Teilmenge sämtlicher Buckets ist, welche mindestens einen Knoten in S preceeden, gilt $B(\pi(S)) \subseteq \pi(B(S))$ und somit $X \supseteq Y$. Dies bedeutet, dass

$$\sum_{(i,j,b)\in\mu(X)} y^b_{i,j} \geq \sum_{(i,j,b)\in\mu(Y)} y^b_{i,j}, \tag{4.27}$$

woraus folgt, dass die linke Seite einer π_b-*Inequality* stets größer oder gleich der linken Seite der korrespondierenden π-*Inequality* ist, während die rechte Seite in beiden Fällen immer den Wert 1 aufweist. Somit ist die Dominanz der π_b-*Inequaliies* über die π-*Inequaliies* gezeigt.

In völlig symmetrischer Argumentationsweise hierzu beweist sich die Dominanz der σ_b-*inequalities* über die σ-*Inequalities*. □

Offensichtlich kann, in Gegenrichtung zu der obigen Verfahrensweise, jeder π_b- und σ_b-*Inequality* für ihr entsprechendes Bucket-Set $B \subseteq \mathfrak{B} \setminus \{B_p, B_q\}$ ebenfalls ein zugrunde liegendes Knoten-Set $S \subseteq V$ zugeordnet werden, welches genau alle die Knoten enthält, deren Buckets sämtlich in B enthalten sind. Gemäß der eingeführten Notation ist dieses Set gerade $V(B)$. Gilt nun $B(V(B)) = B$, gibt also mit anderen Worten die Vereinigungsmenge über alle Buckets, welche dem Knoten-Set $V(B)$ zugeordnet werden können, die Bucket-Menge B zur Gänze wieder, so sprechen *Dash et al.* [5] auch von einer "*simple* π_b-*Inequality*"(beziehungsweise "einfachen π_b-*Inequality*") beziehungsweise einer "*simple* σ_b-*Inequality*"(beziehungsweise "einfachen σ_b-*Inequality*").

Separations-Heuristik Zur Separation ausgewählter "*simple* $\pi_b - Cuts$" im Rahmen des *TSPTW-TBR*-Branch-And-Cut-Frameworks entwickeln *Dash et al.* [5], basierend auf den Ausführungen von *Ascheuer et al.* [13] zur Separation von π-*Cuts*, die folgende Heuristik:

Sei $A^S_{\mathfrak{B}} = \{(b, b') \in A_{\mathfrak{B}} : b \notin \pi(B(S)), b' \notin \pi(B(S))\}$ mit $S \subset V$ und $q \notin S$ eine Bucket-Kanten-Hilfsmenge, welche, analog zum Operator $\delta_\pi(B(S))$, um sämtliche Bucket-Kanten bereinigt ist, die ihren Anfang und/oder ihr Ende in $\pi(B(S))$ haben.

Gegeben eine *TSPTW-TBF*-LP-Lösung (x^*, y^*, z^*) lässt sich mit Hilfe dieser Kantenmenge ein "Projektions-Parameter" $\tilde{x}_{i,j}$ wie folgt bestimmen:

$$\tilde{x}_{i,j} = \sum_{(i,j,b)\in\mu(A^S_{\mathfrak{B}})} (y^b_{i,j})^*. \tag{4.28}$$

Da gemäß Gleichung (2.18) gilt, dass $x_{i,j} = \sum_{b\in B_i} y^b_{i,j}$, spiegelt $\tilde{x}_{i,j}$ somit die Werte-Belegung der Entscheidungsvariablen $(x_{i,j})^*$, adjustiert an die Bedingungen des Parameters $\delta_\pi(B(S))$, wider. Somit gilt nun für die linke Seite einer zu der Knoten-Menge S korrespondierenden einfachen π_b-*Inequality*, ausgewertet für den Vektor y^* der in Rede

stehenden LP-Lösung

$$\sum_{(i,j,b)\in\mu(\delta_\pi(B(S)))} (y^b_{i,j})^* = \sum_{(i,j)\in S} \sum_{b:(i,j,b)\in\mu(A^S_{\mathfrak{B}})} (y^b_{i,j})^* = \sum_{(i,j)\in\delta(S)} \tilde{x}_{i,j}. \tag{4.29}$$

Es resultiert die indirekte Evaluationsmöglichkeit der linken Seite einer einfachen π_b-*Inequality* mit dem zugrunde liegenden Bucket-Set $B(S)$ über den Ausdruck

$$\sum_{(i,j)\in\delta(S)} \tilde{x}_{i,j},$$

womit selbige π_b-*Inequality* von der Lösung (x^*, y^*, z^*) genau dann verletzt wird, wenn die Werte von $\tilde{x}_{i,j}$ der Ungleichung $\sum_{(i,j)\in\delta(S)} \tilde{x}_{i,j} \geq 1$, also namentlich der entsprechenden SEC wie sie in (2.21) definiert ist, bezogen auf die Werte $\tilde{x}_{i,j}$, nicht genügen. Überdies gilt, da klarerweise $A^S_{\mathfrak{B}} \subseteq A_{\mathfrak{B}}$ und somit $\tilde{x}_{i,j} \leq (x_{i,j})^*$, dass, gegeben $(x_{i,j})^*$ verletzt die in Rede stehende SEC, $\tilde{x}_{i,j}$ selbige erst recht verletzt.

Hieraus ergibt sich die Möglichkeit, die diskutierte, einfache π_b-*Inequality* mittels der Werte von $\tilde{x}_{i,j}$, auf die gleiche Weise wie eine klassische SEC zu separieren, namentlich also unter der Ausnutzung des *Max-Flow-Min-Cut-Theorems*[8], indem maximale Flüsse in $G = (V, A)$ bestimmt werden, wobei die Kantenkapazitäten zu $\tilde{x}_{i,j}$ zu wählen sind[9].

Im Rahmen dessen ist nach *Dash et al.* [5] folgendes zu beachten: Sei $S' \supseteq S$ mit $q \notin S'$ eine Obermenge von S. Es gilt somit $\pi(B(S)) \subseteq \pi(B(S'))$, wodurch für S' die Auswertung des entsprechenden Hilfsausdrucks $\sum_{(i,j)\in\delta(S')} \tilde{x}_{i,j}$ lediglich eine obere Schranke für die tatsächliche linke Seite der π_b-*Inequality* welche S' zuzuordnen ist, namentlich

$$\sum_{(i,j,b)\in\mu(\delta_\pi(B(S')))} (y^b_{i,j})^*, \tag{4.30}$$

liefert. Somit ist zwar auch in diesem Fall die in Rede stehende π_b-*Inequality* verletzt, wenn $\tilde{x}_{i,j}$ der korrespondierenden SEC nicht genügt, die Umkehrung gilt jedoch klarerweise nicht. Auf diese Problematik wird im Folgenden noch zurück zu kommen sein.

Die konkrete Separationsroutine gestalten *Dash et al.* [5] sodann analog zu den folgenden Ausführungen: Gegeben, wie oben, die Lösung (x^*, y^*, z^*), werden für jeden Knoten $v \in V \setminus \{p, q\}$ das Set $S := v$ gesetzt und die folgenden Schritte ausgeführt:

Step 1 Berechne $\tilde{x}_{i,j} = \sum_{(i,j,b)\in\mu(A^S_{\mathfrak{B}})} (y^b_{i,j})^*$ mit $A^S_{\mathfrak{B}}$ wie oben definiert.

Step 2 Berechne mittels des Hilfs-LPs *Maximum-Flow* den maximalen Fluss z^0 von *Quelle* $:= S$ nach *Senke* $:= \{q\}$ durch $G = (V, A)$ mit den Kantenkapazitäten $u_{i,j} := \tilde{x}_{i,j}$. Sei $\delta(S')$ der resultierende minimale Schnitt. Für $z^0 < 1$ ist die zu S' korrespondierende *simple* π_b-*Inequality* folglich verletzt und die entsprechende Schnittebene

$$\sum_{(i,j,b)\in\mu(\delta_\pi(B(S')))} y^b_{i,j} \geq 1$$

dem Cut-Pool hinzu zu fügen.

Step 3 Ist $z^0 \geq 1$, aber $\pi(B(S)) \neq \pi(B(S'))$, steht indes, auf Grund der oben angesprochenen Problematik, noch nicht fest, dass die zu S' korrespondierende einfache

[8] siehe hierzu beispielsweise [51]
[9] vgl. hierzu beispielsweise [5]

π_b-*Inequality* unverletzt ist, weil die Summe $\sum_{(i,j)\in\delta(S')} \tilde{x}_{i,j}$, da gemäß Voraussetzung von *Step 3* $\pi(B(S')) \supset \pi(B(S))$ gilt, eine Überschätzung der tatsächlichen linken Seite darstellt. Um dies zu beheben sind die Hilfskantenmenge $A^S_{\mathfrak{B}} := A^{S'}_{\mathfrak{B}}$, und somit auch der Hilfs-Paramter $\tilde{x}_{i,j} := \sum_{(i,j,b)\in\mu(A^{S'}_{\mathfrak{B}})} (y^b_{i,j})^*$ entsprechend an S' anzugleichen. Sodann ist $S := S'$ zu setzen und zu *Step 2* zurück zu kehren.

Im Rahmen dessen kann das Hilfs-LP *Maximum-Flow*, angelehnt an das aus der Literatur bekannte, klassische Maximum-Fluss-LP[10], wie folgt gestaltet werden:

Hilfs-LP *Maximum-Flow*

$$\begin{aligned} \max \quad & z^0 \\ \text{s.t.} \quad & z^0 = \sum_{(i,j)\in\delta^+(Quelle)\cap A^{Flow}} f_{i,j} \qquad (4.31) \end{aligned}$$

$$\sum_{k\in V^-(i)\cap A^{Flow}} f_{k,i} = \sum_{j\in V^+(i)\cap A^{Flow}} f_{i,j} \quad \forall\ i \in V \setminus \{Quelle \cup Senke\} \qquad (4.32)$$

$$f_{i,j} \leq u_{i,j} \qquad \forall\ (i,j) \in A^{Flow} \qquad (4.33)$$

mit $z^0 \in \mathbb{Q}^{\geq 0}, f_{i,j} \in \mathbb{Q}^{\geq 0}\ \forall\ (i,j) \in A$. Hierbei ist z^0 die Hilfsvariable, in welcher der Zielfunktionswert gespeichert wird, die somit also den Betrag des maximalen Flusses und, gemäß des Max-Flow-Min-Cut-Theorems, folglich auch den Wert des minimalen Schnittes im aktuell betrachteten Träger-Graphen wiedergibt. A^{Flow} indes stellt die aktuelle Träger-Kantenmenge $A^{Flow} = \{(i,j) \in A : u_{i,j} > 0\}$ des Flussgraphen dar, welche dazu dient, eine dünnere Besetztheit der zugrunde liegenden Modell-Matrix, und somit eine Verkleinerung des LPs, zu erzielen. Die Entscheidungsvariable $f_{i,j}$ gibt sodann die Flussmenge entlang der Kante $(i,j) \in A^{Flow}$ an. Die Kantenmenge A^{Flow}, die Sets *Quelle* und *Senke* sowie die Kantenkapazitäts-Parameter $u_{i,j}$ sind in jeder Flussberechnung, wie oben angegeben, neu zu deklarieren.

Korrespondierend zum Max-Flow-Min-Cut Theorem, liefert das Duale Problem zum Hilfs-LP *Maximum-Flow* gerade das entsprechende LP des gesuchten minimalen Schnittes. Nach der Berechnung von z^0 ist eine Bestimmung der Cut-Knotenmenge S' daher wie folgt möglich:

Seien

$$SP_i$$

die Schattenpreise der jeweiligen Restriktionen in (4.32), so ist

$$S' = \{i \in V : SP_i = 1\}.$$

Hervorzuheben ist, dass aufgrund einer potentiell fraktionalen Natur der Kapazitätsschranken $u_{i,j}$ die Lösung des primalen LPs klarerweise fraktional sein kann. Da jedoch die zugrunde liegende Modell-Matrix *total unimodular*[11] und die *right hand site* des Dualen LPs, als transponierter Kosten(-einheits-)vektor des Hilfs-LPs *Maximum-Flow*, stets

[10] vgl. hierzu beispielsweise [7]

[11] da es sich um ein (gerichtetes) Fluss-Problem handelt, folgt diese Tatsache unmittelbar aus dem *Satz von Heller und Tompkins* [52]

integral sind, wird das Duale LP seines Zeichens immer integral lösen und dies insbesondere in polynomiell beschränkter Zeit. Da die Schattenpreise der primalen Restriktionen den Dualen Entscheidungsvariablen entsprechen, folgt hieraus sodann, dass stets $SP_i \in \{0, 1\}$.

Es sei angemerkt, dass *Dash et al.* [5] in ihrer Beschreibung keine konkrete Aussage über die Art und Weise machen, auf welche in Step 2 der jeweilige minimale Schnitt ermittelt wird. Alternativ zum *Fluss-LP*-Ansatz wären beispielsweise (unter zusätzlicher Beschleunigung durch sogenannte "Shrinking"-Techniken[12]) ebenfalls die Verwendung des *Edmonds-Karp-Algorithmus'*[13], oder des *Minimum-Cut-Algorithmus' von Nagamochi, Ono und Ibaraki* [30] denkbar. Auch wenn letzterer mit $O(nm + n^2 log(n))$ insbesondere die geringste, aktuell bekannte Laufzeit-Komplexität für die Ermittlung minimaler Schnitte aufweist[14], ist die Wahl der LP-Variante aufgrund spezieller Paradigmen der Entwicklungsumgebung `GAMS` , in welcher die Implementationen im Rahmen dieser Arbeit erfolgen, im Hinblick auf die benötigte Laufzeit jedoch vorzuziehen.

In jedem Fall gestaltet sich die Separationsroutine ebenfalls sowohl unter Rückgriff auf den *Fluss-LP*- als auch den *Edmonds-Karp-Algorithmus*-Ansatz, effizient.[15] [16]

Für die Separation ausgewählter, einfacher σ_b-*Cuts* liefern *Dash et al.* [5] keine ausdrückliche Beschreibung, verbleiben jedoch mit dem Hinweis, dass sich selbige sehr ähnlich zu der Separations-Heuristik von ausgewählten, einfachen π_b-*Cuts* gestaltet.

Dies lässt darauf schließen, dass das folgende Vorgehen äquivalent zu den Separations-Maßnahmen von *Dash et al.* [5] bezüglich σ_b-*Cuts* sein sollte:

Sei nunmehr $A^S_{\mathfrak{B}} = \{(b, b') \in A_{\mathfrak{B}} : b \notin \sigma(B(S)), b' \notin \sigma(B(S))\}$. Gegeben, wie oben, die Lösung (x^*, y^*, z^*), werden für jeden Knoten $v \in V \setminus \{p, q\}$ das Set $S := v$ gesetzt und die folgenden Schritte ausgeführt:

Step 1 Berechne $\tilde{x}_{i,j} = \sum_{(i,j,b)\in\mu(A^S_{\mathfrak{B}})}(y^b_{i,j})^*$ mit $A^S_{\mathfrak{B}}$ wie oben definiert.

Step 2 Berechne mittels des Hilfs-LPs *Maximum-Flow* den maximalen Fluss z^0 von *Quelle* $:= S$ nach *Senke* $:= \{p\}$ durch $G = (V, A)$ mit den Kantenkapazitäten $u_{j,i} := \tilde{x}_{i,j}$. Sei $\delta(S')$ der resultierende minimale Schnitt. Für $z^0 < 1$ ist die zu S' korrespondierende *simple* σ_b-*Inequality* folglich verletzt und die entsprechende Schnittebene

$$\sum_{(i,j,b)\in\mu(\delta_\sigma(\overline{B(S')}))} y^b_{i,j} \geq 1$$

dem Cut-Pool hinzu zu fügen.

Step 3 Ist $z^0 \geq 1$, aber $\sigma(B(S)) \neq \sigma(B(S'))$, steht indes, auf Grund der oben angesprochenen Problematik, noch nicht fest, dass die zu S' korrespondierende *simple* σ_b-*Inequality* unverletzt ist. Dementsprechend ist die Hilfskantenmenge $A^S_{\mathfrak{B}} := A^{S'}_{\mathfrak{B}}$, und somit auch der Hilfs-Paramter $\tilde{x}_{i,j} := \sum_{(i,j,b)\in\mu(A^{S'}_{\mathfrak{B}})}(y^b_{i,j})^*$ entsprechend an S' anzugleichen. Sodann ist $S := S'$ zu setzen und zu *Step 2* zurück zu kehren.

Insbesondere werden hier nun die Kantenkapazitäten "gespiegelt" deklariert und als

[12] für Erläuterungen zu verschiedenen "*Shrinking*"-Techniken im Rahmen von Separationsroutinen siehe beispielsweise[38]
[13] siehe hierzu [48]
[14] vgl. hierzu beispielsweise [9]
[15] zur Effizienz des Lösens von LPs siehe beispielsweise [42]
[16] zur Effizienz des *Edmonds-Karp-Algorithmus'* siehe beispielsweise [48]

Senke ist jeweils der Startknoten p zu wählen.

Es handelt sich bei den beschriebenen Vorgehensweisen um Heuristiken, da auf diese Weise nicht alle verletzten π_b- und σ_b-*Inequalties* einer Lösung ermittelt werden. Jedoch separiert sich eine, aus praktischen Gesichtspunkten ausreichende, Teilmenge. Insbesondere werden zu jeder verletzten klassischen SEC die entsprechenden π_b- und σ_b-*Cuts* gefunden.

Ein Stärke der beschriebenen Separationstechniken ist, wie *Dash et al.* [5] hervorheben, dass sie zwar Schnittebenen generieren, welche auf den $y^b_{i,j}$-Variablen beruhen, diese jedoch im Raum der $x_{i,j}$-Variablen ermitteln, und somit, da in aller Regel klarerweise $|V| + |A| \ll |\mathfrak{B}| + |A_{\mathfrak{B}}|$ gilt, auf deutlich kleineren Graphen agieren müssen.

Bei der Separation nicht-einfacher π_b- und σ_b-*Inequalties* ist es hingegen unvermeidlich, sich bei der Separation auf G' zu bewegen, wodurch ein erheblich größerer Rechenaufwand generiert wird, welcher durch die zusätzliche Verbesserung der Dualen Schranken, laut experimenteller Ergebnisse von *Dash et al.* [5], nicht aufgewogen werden kann. Überdies verkompliziert die Ergänzung des Cut-Pools um nicht-einfache *Bucket-SOP-Inequalties* ebenfalls die Lösbarkeit der Folgeknoten-LPs in einem umfangreicheren Maße, als es die entsprechende Verbesserung der Dualen-Schranken legitimieren könnte. Daher sehen *Dash et al.* [5] von der Separation nicht-einfacher π_b- und σ_b-*Inequalties* in ihrem finalen *TSPTW-TBR*-Branch-And-Cut-Framework ab.

4.2.1.4. Simple (π_b,σ_b)-Cuts

Basierend auf den in Abschnitt 4.2.1.2 eingeführten (π,σ)-Cuts definieren *Dash et al.* [5] sodann die sogenannten (π_b,σ_b)-Cuts.

Seien X, Y zwei disjunkte Knoten-Sets mit $X \prec Y$ und die Sets Z und Q so definiert wie in Abschnitt 4.2.1.2. So sind, entsprechend den Sets Q und W, die Mengen

$$\tilde{Q} = \{(b, b') \in A_{\mathfrak{B}} : (i, j) \in Q, b \in B_i, b' \in B_j\} \tag{4.34}$$

$$\tilde{W} = \pi(B(X)) \cup \sigma(B(Y)) \cup B(Z) \tag{4.35}$$

im weitesten Sinne deren Bucket-Projektionen.

Definition Für ein Bucket-Set $B \subseteq \mathfrak{B}$ mit $X \subseteq V(B)$ und $Y \subseteq V(\overline{B})$ lautet die zur (π,σ)-*Inequaltity* korrespondierende (π_b,σ_b)-*Inequaltity* dann

$$((\pi_b, \sigma_b) - Inequalities) \qquad \sum_{(i,j,b) \in \mu(\delta(B \setminus \tilde{W}, \overline{B} \setminus \tilde{W}) \setminus \tilde{Q})} y^b_{i,j} \geq 1. \tag{4.36}$$

Ihre Gültigkeit für die *TSPTW-BMF* und somit die *TSPTW-TBR* belegt sich, nach Subsumtion des Argumentationsraumes von G auf G', absolut analog zu den Ausführungen aus Abschnitt 4.2.1.2 betreffend die Gültigkeit der (π, σ)-*Inequalities* für die *TSPTW-BMF* und somit die *TSPTW-TBR*.

In Analogie zu Abschnitt 4.2.1.3, bezeichnen *Dash et al.* [5], wenn $B(V(B)) = B$, die resultierende Restriktion auch als "*simple* (π_b, σ_b)-*Inequality*"(beziehungsweise "einfache (π_b, σ_b)-*Inequality*").

Separations-Heuristik Basierend auf der Vorgehensweise von *Ascheuer et al.* [13] bei der Separation von (π,σ)-Cuts, entwickeln *Dash et al.* [5] die folgende Separations-Heuristik für (π_b,σ_b)-*Inequalities*:

Sie betrachten zu Abschnitt 4.2.1.4 korrespondierende Sets X, Y der Form $X = \{u\}$ und $Y = \{w\}$ mit $u, w \in V$ und $u \prec w$. Da gezeigt werden kann, dass, gegeben $u, v, w \in V$ mit $u \prec v \prec w$, alle (π_b,σ_b)-*Inequalities*, welche aus den Basis-Sets $X = \{u\}$ und $Y = \{w\}$ resultieren, redundant zu jenen sind, welche sich aus $X = \{u\}, Y = \{v\}$ und $X = \{v\}, Y = \{w\}$ ergeben[17], wäre es vergeudete Rechenkapazität, Tupel $X = \{u\}, Y = \{w\}$ zu Grunde zu legen, für die ein dritter Knoten $v \in V$ existiert, so dass die Präzedenz zwischen u und w unmittelbar aus der Präzedenz-Kette $u \prec v \prec w$ folgt. Daher schließen *Dash et al.* [5] besagte Paare u, w in der Separations-Routine aus.

Sodann definiert sich für ein konkretes Tupel $X = \{u\}, Y = \{w\}$ welches die obigen Kriterien erfüllt, analog zum Vorgehen im Fall der π_b- und σ_b-*Inequalities*, ein Hilfskanten-Set $A_{\mathfrak{B}}^{u,w}$ mit $A_{\mathfrak{B}}^{u,w} = \{(b, b') \in A_{\mathfrak{B}} : b, b' \notin \tilde{W}, (b, b') \notin \tilde{Q}\}$. Somit repräsentiert $A_{\mathfrak{B}}^{u,w}$ die Menge aller Kanten in $A_{\mathfrak{B}}$, adjustiert an die Bedingungen der Summations-Kanten-Menge $\delta(B \setminus \tilde{W}, \overline{B} \setminus \tilde{W}) \setminus \tilde{Q}$ für die Basis-Sets $X = \{u\}, Y = \{w\}$. Daraufhin ermittelt sich der Hilfs-Parameter $\tilde{x}_{i,j}$ in diesem Fall zu

$$\tilde{x}_{i,j} = \sum_{(i,j,b) \in \mu(A_{\mathfrak{B}}^{u,w})} (y_{i,j}^b)^*. \tag{4.37}$$

Gegeben eine *TSPTW-TBR*-LP Lösung (x^*, y^*, z^*), gilt nun also, mit dem zu X gehörigen Bucket-Set $B = B(X)$, für die linke Seite der entsprechenden einfachen (π_b, σ_b)-*Inequality*, ausgewertet für den Vektor y^*:

$$\sum_{(i,j,b) \in \mu(\delta(B \setminus \tilde{W}, \overline{B} \setminus \tilde{W}) \setminus \tilde{Q})} (y_{i,j}^b)^* = \sum_{(i,j) \in X} \sum_{b:(i,j,b) \in \mu(A_{\mathfrak{B}}^{u,w})} (y_{i,j}^b)^* = \sum_{(i,j) \in \delta(X)} \tilde{x}_{i,j}. \tag{4.38}$$

woraus die Möglichkeit resultiert, die linke Seite einer (π_b, σ_b)-*Inequality* einmal mehr über einen maximalen Fluss in G, mit Kantenkapazitäten $\tilde{x}_{i,j}$, zu bestimmen. Präziser entspricht für jedes Knoten-Set S mit $u \in S$ und $w \in \overline{S}$ die linke Seite der zugehörigen einfachen (π_b, σ_b)-*Inequality* dem Gewicht des Schnittes $\delta(S)$ in G, wobei für jede Kante $(i, j) \in A$ der Parameter $\tilde{x}_{i,j}$ deren entsprechende Kantenkapazität (beziehungsweise Kantengewichtung) darstellt.

Die konkrete Seperationsroutine gestalten *Dash et al.* [5] sodann wie folgt: Gegeben, wie oben, die Lösung (x^*, y^*, z^*), werden für jedes Knotenpaar u, w für welches bekannt ist, dass $u \prec w$ und nicht $u \prec v \prec w$ für irgend ein $v \in V \setminus \{u, w\}$, die entsprechenden Sets $X = \{u\}, Y = \{v\}$ deklariert und die folgenden Schritte ausgeführt:

Step 1 Berechne $\tilde{x}_{i,j} = \sum_{(i,j,b) \in \mu(A_{\mathfrak{B}}^{u,w})} (y_{i,j}^b)^*$ mit $A_{\mathfrak{B}}^{u,w}$ wie oben definiert.

Step 2 Berechne mittels des Hilfs-LPs *Maximum-Flow* den maximalen Fluss z^0 von *Quelle* := X nach *Senke* := Y durch $G = (V, A)$ mit den Kantenkapazitäten $u_{i,j} := \tilde{x}_{i,j}$. Sei $\delta(S')$ der resultierende, minimale Schnitt. Für $z^0 < 1$ ist die zu S' korrespondierende, einfache (π_b, σ_b)-*Inequality* folglich verletzt und die entsprechende

[17] siehe hierzu [5]

Schnittebene

$$\sum_{(i,j,b)\in\mu(\delta(B(S')\setminus\tilde{W},\overline{B(S')}\setminus\tilde{W})\setminus\tilde{Q})} y^b_{i,j} \geq 1$$

dem Cut-Pool hinzu zu fügen.

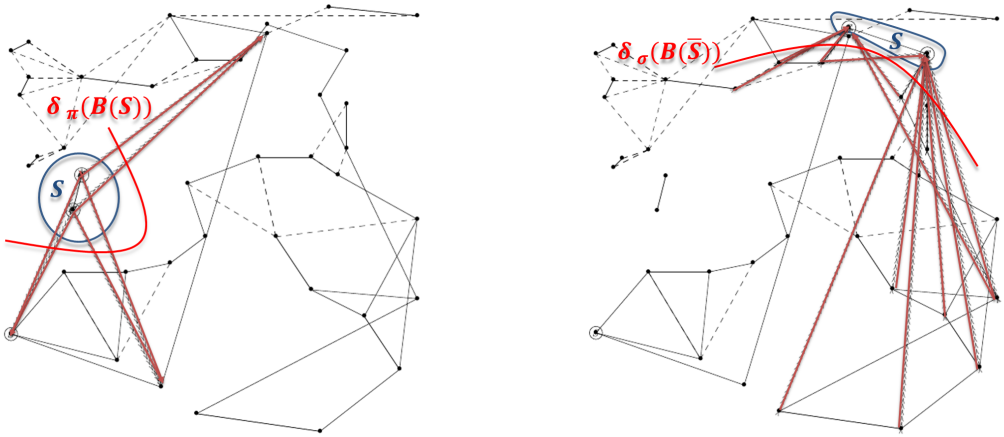

Abbildung 4.6.: Verbildlichungen je eines ausgewählten π_b- (links) und σ_b-*Cuts* (rechts) separiert auf der $n40w80001$-Experimentalinstanz von *Dumas et al.* [28]

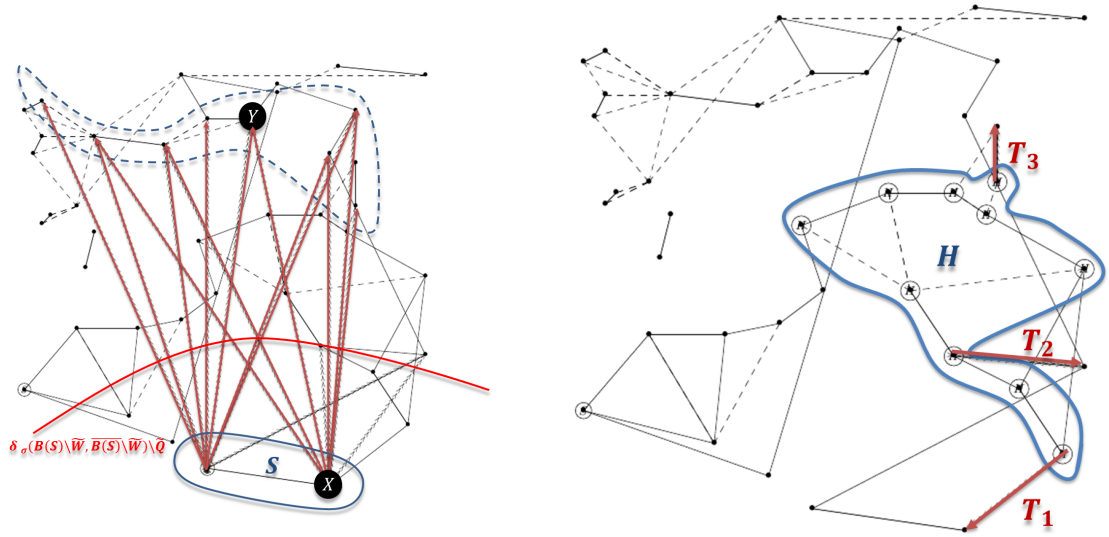

Abbildung 4.7.: Verbildlichungen je eines ausgewählten (π_b, σ_b)- (links) und *Bucket Strengthened 2-Matching-Cuts* (rechts) separiert auf der $n40w80.001$-Experimentalinstanz von *Dumas et al.* [28]

Nach der Berechnung von z^0 bestimmt sich die Knotenmenge S' einmal mehr zu

$$S' = \{i \in V : SP_i = 1\},$$

mit den SP_i als Schattenpreisen der jeweiligen Restriktionen in (4.32).

Von der Separation nicht-einfacher (π,σ)-*Inequalities* sehen *Dash et al.* [5] in ihrem finalen *TSPTW-TBR*-Branch-And-Cut-Framework, aus den selben Gründen wie im Fall der nicht-einfachen π_b- und σ_b-*Inequaltities*, ab.

Im Rahmen eines Vorgriffs auf den Experimentalteil dieser Arbeit, in Abschnitt 5, seien in den Abbildungen 4.6 und 4.7 je eine π_b- eine σ_b- sowie eine (π,σ)-*Inequaltity*, separiert auf der $n40w80.001$-Instanz von *Dumas et al.* [28], illustriert. Hierbei sind die jeweiligen (gerichteten) Cut-Kanten als rote Pfeile hervorgehoben und die zugehörigen Cut-Knoten blau umkreist. Kanten $(i, j) \in A$ welche in der jeweils betrachteten LP-Lösung integral gewählt sind (also die Variablen-Belegung $x_{i,j} = 1$ aufweisen) sind als durchgehende schwarze Verbindungslinien zwischen den Knoten dargestellt. Fraktional gewählte Kanten (also jene, welche eine Variablen-Belegung $0 < x_{i,j} < 1$ aufweisen) sind hingegen als gestrichelte schwarze Verbindungslinien illustriert. Auf das Einzeichnen in der Lösung inaktiver Kanten ist zugunsten der Übersicht verzichtet.

4.2.2. Neue Präzedenz-basierte Bucket-Inequalities

In Ergänzung zu den von *Dash et al.* [5] verwendeten Schnittebenen ist beispielsweise eine Bucket-spezifische Variante der in *Ascheuer* [25] vorgestellten, sogenannten "*Strengthened 2-Matching-Constraints*" denkbar. Selbige basieren ihrerseits auf den klassischen "*2-Matching-Constraints*", welche Sonderfälle der von *Grötschel und Padberg* [46] eingeführten "*Comb-Inequalities*" darstellen, und im Rahmen ihrer Verstärkung, ebenso wie die in Abschnitt 4.2.1 vorgestellten *SOP-Inequalities*, auf vorliegende Präzedenzkenntnisse zwischen den Knoten zurückgreifen.

Im Folgenden seien die *2-Matching-Inequalities* sowie die *Strengthened 2-Matching-Inequalities* präsentiert, um aus selbigen sodann die *Bucket Strengthened 2-Matching-Inequalities* abzuleiten.

4.2.2.1. 2-Matching-Inequalities

Es notiere für ein Knoten-Set $S \subseteq V$ der Operator $A(S)$ die Menge aller Kanten, welche *in* S verlaufen. Formal ist also $A(S) = \{(i, j) \in A : i, j \in S\}$.

Seien $H, T_1, T_2, ..., T_k \subset V \setminus \{p, q\}$, mit $k \geq 3$ und ungerade oder $k = 1 \wedge |H| \geq 4$, eine Sammlung von Knotenmengen (namentlich eine "$Handle$" und mehrere "$Teeth$"), welche den folgenden Bedingungen genügen: (1.) $|H \cap T_l| = 1$ für alle $l = 1, ..., k$, (2.) $|T_l \setminus H| = 1$ für alle $l = 1, ..., k$ und (3.) $|T_l \cap T_m| = \emptyset$ für $1 \leq l < m \leq k$. Ferner sei $T = \bigcup_{l=1,...,k} T_l$.

So definiert sich die entsprechende *2-Matching-Inequality* (*2M-Inequality*) als

$$\sum_{(i,j):i,j \in H} x_{i,j} + \sum_{l=1}^{k} \sum_{(i,j) \in A(T_l)} x_{i,j} \leq |H| + \frac{k-1}{2}, \tag{4.39}$$

wobei diese Form sowohl mit *symmetrischen* als auch *asymmetrischen* Instanzen kompatibel ist, da sie in ihren Parametern an keiner Stelle ausdrücklich auf Kanten- sondern lediglich auf Knotenmengen zurückgreift.

Der Name dieser Restriktions-Familie fußt auf der Tatsache, dass sie das gesamte Polytop des sogenannten *2-Matching-Problems* (*2MP*), namentlich der Problematik des Auffindens einer (kostenminimalen) Kantenmenge A' in einem (Kosten-gewichteten) Graphen

$G = (V, A)$, sodass jeder Knoten $i \in V$ mit genau zwei dieser ausgewählten Kanten inzidiert, beschreibt, wie *Edmonds* [50] zeigen konnte.

Aufgrund der Tatsache, dass das *2MP* augenscheinlich eine Relaxation des TSP, und somit erst recht des gerichteten (also asymmetrischen) TSP[18] sowie des $TSPTW$ darstellt, scheint im behandelten Rahmen ein Zurückgreifen auf *2-Matching-Inequalities*, beziehungsweise Präzedenz-verstärkte Varianten derselbigen, sinnvoll. Überdies fördern die besagten Cuts, aufgrund ihrer "Knotenmengen-übergreifenden Form", potentiell die Zusammenhangs-Struktur der Lösung als Ganzes, womit sich, den Argumentationen in Abschnitt 4.2.1.4 folgend, ein ähnlich positiver Effekt für das Lösungsverhalten der *TSPTW-TBR* einstellen sollte, wie es im Fall der (π_b, σ_b)-Cuts zu beobachten ist.

In Anlehnung an die gängigen Beweise aus der *TSP*-Literatur[19] sei im Folgenden gezeigt, dass die Ungleichung (4.39), und somit die zugehörige Restriktions-Familie der *2-Matching-Inequalities*, gültig für das *TSPTW* und somit für die *TSPTW-BMF* sowie die *TSPTW-TBR* sind.

Satz 5. Die *2-Matching-Inequality* (4.39) ist gültig für das *TSPTW* und somit für die *TSPTW-BMF* sowie die *TSPTW-TBR*.

Beweis. Es sei der Operator $\delta(v)$ so definiert, dass er alle Kanten $(i, j) \in A$ wiedergibt, welche den Knoten v berühren. Präziser ist also $\delta(v) = \{(i, j) \in A : (i, j) \in \delta^-(v) \vee (i, j) \in \delta^+(v)\}$. Ferner notiere $A^{Teeth}(T) := \bigcup_{l=1,...,k} A(T_l)$.

Offenbar gilt für alle Knoten $v \in H$, da gemäß Voraussetzung $H \in V \setminus \{p, q\}$ und jeder dieser Knoten in einer gültigen Tour genau einmal besucht- und genau einmal verlassen wird, dass $\sum_{(i,j)\in\delta(v)} x_{i,j} = 2$, also

$$\sum_{v\in H} \sum_{(i,j)\in\delta(v)} x_{i,j} = 2|H|, \tag{4.40}$$

Ferner gilt für alle Kanten $(i, j) \in \delta(H, \overline{H}) \setminus A^{Teeth}(T)$, auf Grund der nicht-Negativitäts-Bedingungen, dass $x_{i,j} \geq 0$, also ist

$$- \sum_{(i,j)\in\delta(H,\overline{H})\setminus A^{Teeth}(T)} x_{i,j} \leq 0 \tag{4.41}$$

Außerdem ist für alle Kanten $(i, j) \in A^{Teeth}(T)$ auf Grund des binären Wertebereiches von x offenbar $x_{i,j} \leq 1$, und wegen des Ausschlusses von Subtouren innerhalb der Teeth ferner $\sum_{(i,j)\in A(T_l)} x_{i,j} \leq 1$ für alle $l = 1, ..., k$, womit

$$\sum_{l=1}^{k} \sum_{(i,j)\in A(T_l)} x_{i,j} \leq 1. \tag{4.42}$$

Addieren von (4.40),(4.41) und (4.42) sowie Durchmultiplizieren mit dem Faktor $\frac{1}{2}$ ergibt

$$\frac{1}{2} \sum_{v\in H} \sum_{(i,j)\in\delta(v)} x_{i,j} - \frac{1}{2} \sum_{(i,j)\in\delta(H,\overline{H})\setminus A^{Teeth}(T)} x_{i,j} + \frac{1}{2} \sum_{l=1}^{k} \sum_{(i,j)\in A(T_l)} x_{i,j} \leq |H| + \frac{1}{2}k \tag{4.43}$$

[18] tatsächlich konnte *Fischetti* [36] sogar zeigen, dass die *2-Matching-Inequalities* (für $n \neq 6$ mit n als Anzahl der Knoten) facettendefinierend für das *asymmetrische TSP-Polytop* sind

[19] siehe hierzu beispielsweise [24]

Da im Rahmen des ersten Summanden der linken Seite alle Kanten (i,j), welche komplett in H verlaufen, für die also $i,j \in H$ gilt, je doppelt gezählt werden, nämlich einmal über $\delta(i)$ und einmal über $\delta(j)$, ferner hingegen alle Kanten (i,j), welche aus H herausführen, namentlich die Kanten $(i,j) \in A^{Teeth}(T)$ und die Kanten $(i,j) \in \delta(H,\overline{H}) \setminus A^{Teeth}(T)$, je einzeln gezählt werden, nämlich nur über $\delta(i)$, gilt die Zerlegung

$$\sum_{v \in H} \sum_{(i,j) \in \delta(v)} x_{i,j} = 2 \sum_{(i,j):i,j \in H} x_{i,j} + \sum_{(i,j) \in \delta(H,\overline{H}) \setminus A^{Teeth}(T)} x_{i,j} + \sum_{(i,j) \in A^{Teeth}(T)} x_{i,j}. \quad (4.44)$$

Einsetzen von (4.44) in (4.43) und Vereinfachen liefert

$$\sum_{(i,j):i,j \in H} x_{i,j} + \sum_{l=1}^{k} \sum_{(i,j) \in A(T_l)} x_{i,j} \leq |H| + \frac{1}{2}k \quad (4.45)$$

Da gemäß Voraussetzung $x_{i,j} \in \{0,1\}$ gilt und somit die linke Seite klarerweise immer ganzzahlig ist, kann der Summand $\frac{1}{2}k$ der rechten Seite abgerundet werden. Da überdies gemäß Voraussetzung k ungerade ist, gilt ferner $\lfloor \frac{1}{2}k \rfloor = \frac{1}{2}k - \frac{1}{2}$, womit sich die 2-Matching-Inequality aus (4.39) ergibt. □

4.2.2.2. Strengthened-2-Matching-Inequalities

Die Deklarationen aus Abschnitt 4.2.2.1 sollen weiterhin gelten.

Es sei überdies $S := H \cup T$, $s_i := H \cap T_i$, $t_i := T_i \setminus H$ (insbesondere gilt gemäß Voraussetzung jeweils $|s_i| = |t_i| = 1$). Aus Gründen der Übersicht seien die linke und die rechte Seite der *2-Matching-Inequality* ferner verkürzt notiert mit $\sum_{LS_{2M}} x_{i,j} := \sum_{(i,j):i,j \in H} x_{i,j} + \sum_{l=1}^{k} \sum_{(i,j) \in A(T_l)} x_{i,j}$ und $RS_{2M} := |H| + \frac{1}{2}(k-1)$. So definiert *Ascheuer* die folgenden, sogenannten *Strengthened 2-Matching-Inequalities* (*S2M-Inequalities*):

$$\sum_{LS_{2M}} x_{i,j} + \sum_{i=1}^{k} \sum_{(i,j) \in \delta(\overline{S} \cap \sigma(t_i), s_i)} x_{i,j} + \sum_{i=1}^{k} \sum_{(i,j) \in \delta(s_i, \overline{S} \cap \pi(t_i))} x_{i,j} \leq RS_{2M} \quad (4.46)$$

$$\sum_{LS_{2M}} x_{i,j} + \sum_{i=1}^{k} \sum_{(i,j) \in \delta((H \setminus T) \cap \sigma(s_i), t_i)} x_{i,j} + \sum_{i=1}^{k} \sum_{(i,j) \in \delta(t_i, (H \setminus T) \cap \pi(s_i))} x_{i,j} \leq RS_{2M} \quad (4.47)$$

In Anlehnung an *Ascheuer* [25] sei im Folgenden die Gültigkeit der *Stregthened 2-Matching-Inequalities* für das *TSPTW* und somit für die *TSPTW-BMF* sowie die *TSPTW-TBR* gezeigt.

Satz 6. Die *Strengthened 2-Matching-Inequalities* (4.46) und (4.47) sind gültig für das *TSPTW* und somit für die *TSPTW-BMF* sowie die *TSPTW-TBR*.

Beweis. Wenn in Ungleichung (4.46) keine der, von den beiden zusätzlichen Summanden inkarnierten, präzedenzbehafteten Kanten verwendet wird, also

$$\sum_{i=1}^{k} \sum_{(i,j) \in \delta(\overline{S} \cap \sigma(t_i), s_i)} x_{i,j} + \sum_{i=1}^{k} \sum_{(i,j) \in \delta(s_i, \overline{S} \cap \pi(t_i))} x_{i,j} = 0$$

gilt, reduziert sie sich auf die ursprüngliche *2-Matching-Inequality* (4.39).

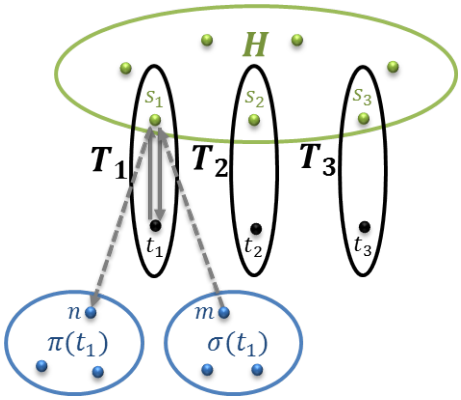

Abbildung 4.8.: Schematische Verbildlichung der Mengenkonstellationen einer *Strengthened 2-Matching-Inequality* in (4.46) mit $k = 3$ (In Anlehnung an *Ascheuer* [25])

Es gilt ferner für jedes $i = 1, ..., k$, dass jeweils nur entweder eine Kante aus der Menge $\delta(\overline{S} \cap \sigma(t_i), s_i)$, oder der Menge $\delta(s_i, \overline{S} \cap \pi(t_i))$ oder der Menge $A(T_i)$ ausgewählt werden kann, da sich sonst implizit eine Präzedenzbedingung verletzt sähe (konkret sind die entsprechenden beiden Pfad-Mengen offenbar $P_{\sigma\_verletzt} = (m, s_i, t_i)$ mit $m \in \overline{S} \cap \sigma(t_i)$ und $P_{\pi\_verletzt} = (t_i, s_i, n)$ mit $n \in \overline{S} \cap \pi(t_i)$), und/oder einer der Knoten in T_i zweimal besucht oder zweimal verlassen, und/oder sich in T_i eine Subtour schließen würde. Abbildung 4.8 soll diesen Sachverhalt für den Fall $k = 3$ noch einmal illustrierend verdeutlichen (wobei die zusätzlichen ("verstärkenden") Kanten der *Strengthened 2-Matching-Inequality* lediglich für den Knoten t_1, in gestrichelter Form und schematisch für nur je einen Knoten aus $\overline{S} \cap \pi(t_i)$ beziehungsweise $\overline{S} \cap \sigma(t_i)$, angedeutet werden, und die Kanten $A(T_i)$ ebenfalls nur für $i = 1$ abgebildet sind. Überdies wird aus Vereinfachungsgründen $S \cap \pi(t_i) = S \cap \sigma(t_i) = \emptyset$ angenommen).

Formal gilt also

$$\sum_{(i,j)\in\delta(\overline{S}\cap\sigma(t_i),s_i)} x_{i,j} + \sum_{(i,j)\in\delta(s_i,\overline{S}\cap\pi(t_i))} x_{i,j} + \sum_{(i,j)\in A(T_i)} x_{i,j} \leq 1,$$

woraus, bei Auswahl einer Kante eines der linksseitigen Summanden, ein komplementäres Verhalten der zwei anderen linksseitigen Summanden, und somit die Gültigkeit der *Strengthened 2-Matching-Inequality* aus (4.46) für das *TSPTW* sowie die *TSPTW-BMF* und die *TSPTW-TBR* folgen.

In absoluter Analogie hierzu beweist sich die Gültigkeit der *Strengthened 2-Matching-Inequality* aus (4.47) für das *TSPTW* und somit die *TSPTW-BMF* sowie die *TSPTW-TBR*. □

Sehr leicht lässt sich überdies zeigen, dass die *Strengthened 2-Matching-Inequalities* (4.46) und (4.47) die *2-Matching-Inequalities* dominieren.

Satz 7. Die *Strengthened 2-Matching-Inequalities* (4.46) und (4.47) dominieren die *2-Matching-Inequalities*.

Beweis. In beiden Fällen der *Strengthened 2-Matching-Inequalities* werden auf der linken Seite der ursprünglichen *2-Matching-Inequality* je zwei Summanden addiert die, wegen

$x_{i,j} \in \{0,1\}$, in keinem Fall negativ werden. Hierdurch wird die linke Seite somit höchstens größer, während die rechte Seite unverändert ist. Die Behauptung folgt. □

4.2.2.3. Bucket Strengthened 2-Matching-Inequalities

Diese *Strengthened 2-Matching-Inequalities* lassen sich nun zu den folgenden, Bucketsinvolvierenden Ungleichungen verstärken, welche als "*Bucket Strengthened 2-Matching-Inequalities*" bezeichnet werden sollen:

Definition Es gelten weiterhin die Deklarationen aus den Abschnitten 4.2.2.1 und 4.2.2.2.

Aus Gründen der Übersichtlichkeit seien ferner $H_T := H \setminus T$, $B_{s_i} := B(s_i)$, $B_{t_i} := B(t_i)$, sowie

$$\begin{aligned}
\delta(B_{\overline{S}\cap\sigma^{t_i}}, B_{s_i}) &:= \delta(B(\overline{S}) \cap \sigma(B(t_i)), B_{s_i}) \\
\delta(B_{s_i}, B_{\overline{S}\cap\pi^{t_i}}) &:= \delta(B_{s_i}, B(\overline{S}) \cap \pi(B(t_i))) \\
\delta(B_{H_T\cap\sigma^{s_i}}, B_{t_i}) &:= \delta(B(H_T) \cap \sigma(B(s_i)), B_{t_i}) \\
\delta(B_{t_i}, B_{H_T\cap\pi^{s_i}}) &:= \delta(B_{t_i}, B(H_T) \cap \pi(B(s_i))).
\end{aligned}$$

wobei im Rahmen der letzten vier Zuweisungen in einem weitesten Sinne die Bucket-Extensionen der zugrunde liegenden Summen-Operatoren aus den *Strengthened 2-Matching-Inequalities* notiert sind. Sodann seien die zugehörigen *Bucket Strengthened 2-Matching-Inequalities* (*BS2M-Inequalities*) wie folgt definiert:

$$\sum_{LS_{2M}} x_{i,j} + \sum_{i=1}^{k} \sum_{(i,j,b)\in\mu(\delta(B_{\overline{S}\cap\sigma^{t_i}}, B_{s_i}))} y^b_{i,j} + \sum_{i=1}^{k} \sum_{(i,j,b)\in\mu(\delta(B_{s_i}, B_{\overline{S}\cap\pi^{t_i}}))} y^b_{i,j} \quad \leq RS_{2M} \quad (4.48)$$

$$\sum_{LS_{2M}} x_{i,j} + \sum_{i=1}^{k} \sum_{(i,j,b)\in\mu(\delta(B_{H_T\cap\sigma^{s_i}}, B_{t_i}))} y^b_{i,j} + \sum_{i=1}^{k} \sum_{(i,j,b)\in\mu(\delta(B_{t_i}, B_{H_T\cap\pi^{s_i}}))} y^b_{i,j} \leq RS_{2M}. \quad (4.49)$$

Im Folgenden sei gezeigt, dass die *Bucket Stregthened 2-Matching-Inequalities* gültig für das *TSPTW* und somit für die *TSPTW-BMF* sowie die *TSPTW-TBR* sind, und ferner insbesondere die entsprechenden *Stregthened 2-Matching-Inequalities* dominieren.

Satz 8. Die *Bucket Strengthened 2-Matching-Inequalities* (4.48) und (4.49) sind gültig für das *TSPTW* und somit für die *TSPTW-BMF* sowie die *TSPTW-TBR*.

Beweis. Nach Subsumtion des Argumentations-Raumes von G auf G' völlig analog zu dem Beweis von Satz 7. □

Satz 9. Die *Bucket Strengthened 2-Matching-Inequalities* aus (4.48) dominieren die *Strengthened 2-Matching-Inequalities* aus (4.46) und die *Bucket Strengthened 2-Matching-Inequalities* aus (4.49) dominieren die *Strengthened 2-Matching-Inequalities* aus (4.47).

Beweis. Bei Subsumtion der *Stregthened 2-Matching-Inequality* aus (4.46) auf den Raum des Bucket-Graphen G' folgt (mit (2.18)) für die zweiten beiden Summanden der linken Seite

$$\sum_{i=1}^{k} \sum_{(i,j)\in\delta(\overline{S}\cap\sigma(t_i), s_i)} x_{i,j} = \sum_{i=1}^{k} \sum_{(i,j,b)\in\mu(\delta(B(\overline{S}\cap\sigma(t_i)), B(s_i)))} y^b_{i,j}$$

und

$$\sum_{i=1}^{k} \sum_{(i,j)\in\delta(s_i,\overline{S}\cap\pi(t_i))} x_{i,j} = \sum_{i=1}^{k} \sum_{(i,j,b)\in\mu(\delta(B(s_i),B(\overline{S}\cap\pi(t_i))))} y_{i,j}^{b}.$$

Da klarerweise die Menge aller Buckets, welche den Knoten zugeordnet werden können, die den Knoten t_i succeeden, eine Teilmenge sämtlicher Buckets darstellt, die den Knoten t_i succeden, also $\sigma(B(t_i)) \supseteq B(\sigma(t_i))$ ist, gilt

$$\delta(B(\overline{S}) \cap \sigma(B(t_i)), B(s_i)) \supseteq \delta(B(\overline{S} \cap \sigma(t_i)), B(s_i)).$$

Aus völlig symmetrischer Argumentation folgt, dass $\pi(B(t_i)) \supseteq B(\pi(t_i)))$ und somit

$$\delta(B_{s_i}, B(\overline{S}) \cap \pi(B(t_i)))) \supseteq \delta(B(s_i), B(\overline{S} \cap \pi(t_i))).$$

Daher gilt

$$\sum_{i=1}^{k} \sum_{(i,j,b)\in\mu(\delta(B(\overline{S})\cap\sigma(B(t_i)),B(s_i)))} y_{i,j}^{b} \geq \sum_{i=1}^{k} \sum_{(i,j,b)\in\mu(\delta(B(\overline{S}\cap\sigma(t_i)),B(s_i)))} y_{i,j}^{b}$$

sowie

$$\sum_{i=1}^{k} \sum_{(i,j,b)\in\mu(\delta(B_{s_i},B(\overline{S})\cap\pi(B(t_i))))} y_{i,j}^{b} \geq \sum_{i=1}^{k} \sum_{(i,j,b)\in\mu(\delta(B(\overline{S}\cap\sigma(t_i)),B(s_i)))} y_{i,j}^{b},$$

womit die linke Seite der *Bucket Stregthened 2-Matching-Inequality* aus (4.48) stest größer oder gleich der linken Seite der entsprechenden *Stregthened 2-Matching-Inequality* aus (4.46) ist, während die rechten Seiten immer identisch sind. Folglich dominieren die *Bucket Stregthened 2-Matching-Inequalities* aus (4.48) die *Stregthened 2-Matching-Inequalities* aus (4.46).

In völliger Analogie hierzu beweist sich die Dominanz der *Bucket Stregthened 2-Matching-Inequalities* aus (4.49) über die *Stregthened 2-Matching-Inequalities* aus (4.47). □

Ferner folgt aus der Beweisführung von Satz 8 (beziehungsweise Satz 7) unmittelbar, dass einer verletzten (*symmetrischen*) *2-Matching-Inequality* aus (4.39) (mittels Ergänzung der entsprechenden beiden Summanden) stets zwei verletzte *Bucket Strengthened 2-Matching-Inequalities* aus (4.48) und (4.49) zugeordnet werden können, die Gegenimplikation gilt jedoch klarerweise nicht. Um diese Tatsache mittels einer Distinguierung hervorzuheben, seien *Bucket Strengthened 2-Matching-Inequalities* aus (4.48) und (4.49), deren zugrunde liegende *2-Matching-Inequality* (4.39) *nicht* verletzt ist, auch als "schwach verletzt" bezeichnet, während im Fall einer verletzten (*symmetrischen*) *2-Matching-Inequality* (4.39) die erst recht verletzten, zuzuordnenden *Bucket Strengthened 2-Matching-Inequalities* aus (4.48) und (4.49) entsprechend als "stark verletzt" kategorisiert werden sollen.

Separations-Heuristik Aufbauend auf den Ausführungen und dem Algorithmus von *Araóz et al.* [65] zur Ermittlung verletzter *2-Matching-Inequalities* auf *symmetrischen TSP*-Instanzen, sei im Folgenden eine mögliche Separations-Heuristik für (*asymmetrische*) *Bucket Strengthened 2-Matching-Inequalities* entwickelt:

Angelehnt an die Erkenntnisse von *Padberg und Rao* [44], betrachten *Araóz et al.* [65] die klassischen *2-Matching-Inequalities* in der folgenden, alternativen Formulierungsvariante, welche einen günstigen Ausgangspunkt für das Anwenden Fluss-orientierter Separationsroutinen bietet:

$$\sum_{(i,j)\in\delta^+(H)\setminus A^+(T)} x_{i,j} \geq \sum_{(i,j)\in A^+(T)} x_{i,j} - k + 1, \quad \forall\ A^+(T) \subseteq \delta(H), \tag{4.50}$$

mit $H \subset V$, $T = \bigcup_{l=1,...,k} T_l \subset V$ wie oben definiert und k ungerade, wobei die Notation, entsprechend den Konventionen dieser Arbeit, moduliert ist. Insbesondere sind sowohl Mengen, Operatoren als auch Folgeargumentationen einmal mehr subsumiert auf den hier vorliegenden Fall des *asymmetrischen TSP* beziehungsweise *-TSPTW*.[20] In diesem Sinne notiere $A^+(T)$ die Menge aller Kanten in T, welche aus der Handle H herausführen und Teeth darstellen. Präziser sei also $A^+(T) = \{(i,j) \in \delta(H,\overline{H}) : (i,j) \in A(T_l)\ \textit{für irgendein}\ l = 1,...,k\}$.

Ein Umstellen von (4.50) liefert die Form

$$\sum_{(i,j)\in\delta^+(H)\setminus A^+(T)} x_{i,j} + \sum_{(i,j)\in A^+(T)} (1 - x_{i,j}) \geq 1, \quad \forall\ A^+(T) \subseteq \delta(H). \tag{4.51}$$

Basierend hierauf lässt sich das Separationsproblem auf ein *Maximum-Fluss*-gestütztes Auffinden von zwei Sets $H^* \subset V, T^* = \{T_1^*, ..., T_k^*\} \subset V$ mit k ungerade, reduzieren, welche die Ungleichung (4.51) beziehungsweise (4.50), verletzen:

Gegeben eine Lösung (x^*, y^*, z^*) der *TSPTW-TBR*, so notiere, in Anlehnung an *Araóz et al.* [65], $G_{x^*} = (V, A_{x^*})$ einen Träger-Graphen mit $A_{x^*} = \{(i,j) \in A : x^*_{i,j} > 0 \vee x^*_{j,i} > 0\}$, wobei jeder Kante in A_{x^*} eine (zum Ausmaß der entsprechenden Entscheidungsvariablen-Fraktionalität korrespondierende) Kapazität $h_{i,j} = h_{j,i} = \min\{x^*_{i,j}, 1 - x^*_{i,j}\}$ zugeordnet wird (insbesondere seien die asymmetrischen Belegungen der Variablen also zu einer symmetrischen Kapazitätskonstellation der Kanten hin gedoppelt). Basierend hierauf erfolgt die Partitionierung $A_{x^*} = A^< \cup A^>$, wobei eine Kante $(i,j) \in A_{x^*}$ genau dann dem Set $A^<$ zugeordnet wird, wenn ihre Kapazität $h_{i,j}$ durch die tatsächliche Variablenbelegung $x^*_{i,j}$ gegeben ist. Komplementärer Weise wird $(i,j) \in A_{x^*}$ genau dann dem Set $A^>$ zugeordnet, wenn $h_{i,j} = 1 - x^*_{i,j}$ ist. Kanten $(i,j) \in A_{x^*}$ mit dem indifferenten Kapazitätswert $h_{i,j} = 0.5$ werden beliebig zugeordnet.

Überdies seien, *Araóz et al.* [65] folgend, die Knoten-(Mengen-)Eigenschaften "$odd^>$" und "$even^>$" eingeführt. Hierbei ist ein Knoten $v \in V$ genau dann $odd^>$, wenn $\lfloor\frac{1}{2}|\delta(v) \cap A^>|\rfloor \neq \frac{1}{2}|\delta(v) \cap A^>|$ gilt, wenn also die Anzahl aller Kanten in der Hilfs-Kantenmenge $A^>$, welche v berühren, ungerade ist. Gilt hingegen $\lfloor\frac{1}{2}|\delta(v) \cap A^>|\rfloor = \frac{1}{2}|\delta(v) \cap A^>|$, ist die in Rede stehende Kantenanzahl also gerade, so soll der Knoten v entsprechender Weise als $even^>$ bezeichnet werden. Im Rahmen einer Erweiterung dieser Nomenklatur auf Sets von Knoten sei eine Knotenmenge $S \subset V$ genau dann $odd^>$, wenn $\lfloor\frac{1}{2}|\{v \in S : v\ \textit{ist}\ odd^>\}|\rfloor \neq \frac{1}{2}|\{v \in S : v\ \textit{ist}\ odd^>\}|$ gilt, wenn also die Anzahl aller Knoten in S, welche die Eigenschaft $odd^>$ haben, ungerade ist. Im Gegenfall $\lfloor\frac{1}{2}|\{v \in S : v\ \textit{ist}\ odd^>\}|\rfloor = \frac{1}{2}|\{v \in S : v\ \textit{ist}\ odd^>\}|$ ist die Knotenmenge S entsprechend $even^>$.

Des Weiteren seien für einen gegebene Schnitt $\delta^+(S')$ (mit $S' \subset V$) die beiden "Grenz"-Kanten $(i,j)_1$ und $(i,j)_2$ je frei gewählt, so dass $x^*_{(i,j)_1} = \min\{x^*_{i,j} : (i,j) \in \delta^+(S') \cap A^>\}$ (womit $(i,j)_1$ also eine "fraktionalste"Kante aus der Menge $A^>$ im Cut $\delta^+(S')$ darstellt)

[20] diese konkrete Übertragung ist deswegen ohne Weiteres möglich, da die *2-Matching-Constraints* zu den *symmetrischen ATSP-Inequalities* gehören. Siehe hierzu beispielsweise [25]

und $x^*_{(i,j)_2} = \max\{x^*_{i,j} : (i,j) \in \delta^+(S') \setminus A^>\}$ (womit $(i,j)_2$ also eine "fraktionalste"Kante im Cut $\delta^+(S')$ ist, welche nicht in der Menge $A^>$ liegt) gilt.

Aus Gründen der Übersicht notiere im Folgenden $x^{\blacktriangleright}_{(i,j)_1} := x^*_{(i,j)_1} - (1 - x^*_{(i,j)_1})$ und $x^{\blacktriangleleft}_{(i,j)_2} := (1 - x^*_{(i,j)_2}) - x^*_{(i,j)_2}$. *Araóz et al.* [65] konnten zeigen, dass

Crt.1 eine *symmetrische 2-Matching-Inequality*, welche einem Knoten-Set $S' \subset V$ mit S' $odd^>$ zuzuordnen ist, genau dann verletzt ist, wenn

$$\sum_{(i,j)\in\delta^+(S')} h_{i,j} < 1,$$

wobei für $Handle$ und $Teeth$

$$\begin{aligned} Handle &:= S', \\ Teeth &:= \delta^+(S') \cup A^>, \end{aligned}$$

gilt.

Crt.2 eine *symmetrische 2-Matching-Inequality*, welche einem Knoten-Set $S' \subset V$ mit S' $even^>$ zuzuordnen ist, genau dann verletzt ist, wenn

$$\sum_{(i,j)\in\delta^+(S')} h_{i,j} + \min\{x^{\blacktriangleright}_{(i,j)_1}, x^{\blacktriangleleft}_{(i,j)_2}\} < 1$$

wobei für $Handle$ und $Teeth$

$$\begin{aligned} Handle &:= S', \\ Teeth &:= \begin{cases} (\delta^+(S') \cup A^>) \setminus \{(i,j)_1\}, & \text{falls } x^{\blacktriangleright}_{(i,j)_1} < x^{\blacktriangleleft}_{(i,j)_2} \\ (\delta^+(S') \cup A^>) \cup \{(i,j)_2\}, & \text{falls } x^{\blacktriangleright}_{(i,j)_1} \geq x^{\blacktriangleleft}_{(i,j)_2} \end{cases} \end{aligned}$$

gilt.

Hierbei werden die Teeth der *symmetrischen 2-Matching-Inequalities*, im Gegensatz zum bisher betrachteten (*asymmetrischen*) Fall, nicht über Knoten-, sondern über ausdrückliche Kanten-Mengen beschrieben. *Symmetrische 2-Matching-Inequalities* dieser Form können jedoch, wie oben bereits angesprochen, o.B.d.A. in die für *aysmmetrische 2-Matching-Inequalities* taugliche Formalisierung aus Gleichung (4.39) überführt werden. Konkret kann dies geschehen, indem die jeweiligen Start- und Endknoten der Teeth-Kanten extrahiert und den entsprechenden Knoten-Mengen T_i mit $i = 1, ..., k$ zugewiesen werden. Praktisch gesehen werden die Teeth-Kanten somit schlichtweg "gedoppelt".

Ferner lassen sich, wie oben bereits diskutiert, aus einer verletzten (*symmetrischen*) *2-Matching-Inequality* stets zwei verletzte (*asymmetrische*) *Bucket Strengthened 2-Matching-Inequalities* ableiten. Somit lässt sich die in Abbildung 4.9 dargestellte Separations-Heuristik für *Bucket Strengthened 2-Matching-Inequalities* formulieren. Ihre Kernstruktur fußt auf der Routine, welche *Araóz et al.* [65] für das Aufspüren verletzter *symmetrischer 2-Matching-Inequalities* beschreiben.

Die Subroutine "AddBSMCutsFor" übersetzt jeweils die gefundenen Elemente der verletzten *symmetrischen 2-Matching-Inequality* in die beiden, korrespondierenden *asymmetrischen Bucket Strengthened 2-Matching-Inequalities*. Es sei angemerkt, dass im Rahmen des vorliegenden Algorithmus', im Gegensatz zur formalen Handhabung, auch im *asymmetrischen* Fall die Teeth der Inequalities über Kantenmengen kodiert werden, um die

Implementierung in der verwendeten Entwicklungsumgebung zu vereinfachen.

Offenbar ist der Algorithmus in Abbildung 4.9, wegen Crt.1 und Crt.2, sogar eine exakte Separationsroutine für sämtliche verletzten, *symmetrischen 2-Matching-Inequalities*[21], und damit auch für sämtliche stark verletzten *Bucket Strengthened 2-Matching-Inequalities*[22], indes aber klarerweise nicht für alle verletzten (*asymmetrischen*) *Bucket Strengthened 2-Matching-Inequalities*, und somit als Heuristik zu bezeichnen.

procedure FindBSMCuts
  for all $i \in V \setminus \{p, q\}$ **do**
    for all $j \in V \setminus \{i, p, q\}$ und nicht $j \prec i$ und nicht $i \succ j$ **do**
      Berechne den maximalen Fluss z^0 von i nach j durch $G_{x^*} = (V, A_{x^*})$ mit den Kantenkapazitäten $h_{i,j}$.
      Es sei $\delta^+(S')$ der resultierende minimale Schnitt.
      if S' ist $odd^>$ und $z^0 < 1$ und $(|\delta^+(S') \cup A^>| \geq 3 \vee |S'| \geq 4)$ **then**
        AddBSMCutsFor(S',$\delta^+(S') \cup A^>$)
      if S' ist $even^>$ und $z^0 + \min\{x^{\blacktriangleright}_{(i,j)_1}, x^{\blacktriangleleft}_{(i,j)_2}\} < 1$ **then**
        if $x^{\blacktriangleright}_{(i,j)_1} < x^{\blacktriangleleft}_{(i,j)_2}$ und $(|(\delta^+(S') \cup A^>) \setminus \{(i,j)_1\}| \geq 4 \vee |S'| \geq 4)$ **then**
          AddBSMCutsFor(S',$(\delta^+(S') \cup A^>) \setminus \{(i,j)_1\}$)
        if $x^{\blacktriangleright}_{(i,j)_1} \geq x^{\blacktriangleleft}_{(i,j)_2}$ **then**
          AddBSMCutsFor(S',$(\delta^+(S') \cup A^>) \cup \{(i,j)_2\}$)

procedure AddBSMCutsFor($H \subset V$,$T^e_{sym} \subset A$)
  Füge die *Bucket Strengthened 2-Matching-Inequalities* aus (4.48) und (4.49), für die Handle H und die Teeth

$$T^e_{asym} = \{(i,j) \in A : (i,j) \in T^e_{sym} \ \vee \ (j,i) \in T^e_{sym}\},$$

  dem Cut-Pool hinzu.

Abbildung 4.9.: Möglicher Pseudocode zur Beschreibung der Separations-Heuristik für *Bucket Strengthened 2-Matching-Inequalities*

Zu kommentieren bleibt, dass der "Basis"-Algorithmus auch *Inequalities* mit $k = 1$ separiert (nämlich genau dann, wenn S' $odd^>$ mit $|\delta^+(S') \cup A^>| = 1$, oder S' $even^>$ mit $|\delta^+(S') \cup A^>| = 2$ ist und $z^0 + \min\{x^{\blacktriangleright}_{(i,j)_1<1}, x^{\blacktriangleleft}_{(i,j)_2}\} < 1$ mit $x^{\blacktriangleright}_{(i,j)_1} < x^{\blacktriangleleft}_{(i,j)_2}$ gilt), welche, falls $|H| < 4$, nach der in dieser Arbeit angeführten Definition keine *2-Matching-Constraints* darstellen. Selbige werden daher mittels je einer zusätzlichen If-Abfrage ausgeschlossen.

Überdies sei angemerkt, dass die von *Araóz et al.* [65] vorgeschlagene Basisroutine auf die Konstruktion des sogenannten *Minimum-Cut-Trees*, namentlich einen Baum, bestehend aus sämtlichen Kanten die jeweils den kostenminimalen Schnitten aller Knotenpaare in G^* zugeordnet werden können, zurück greift, was eine erhebliche Akzeleration des Algorithmus' bewirkt. Aufgrund spezieller Paradigmen der Entwicklungsumgebung `GAMS` , in welcher die Implementationen im Rahmen dieser Arbeit erfolgen, sei im Hinblick auf die benötigte Laufzeit jedoch die im Pseudocode aus Abbildung 4.9 beschriebene Implementierungsvariante vorgezogen.

[21] welche keine Präzedenzen konterkarieren
[22] welche keine Präzedenzen konterkarieren

Im Rahmen eines Vorgriffs auf den Experimentalteil in Abschnitt 5 sei in Abbildung 4.7 (rechts) ein *Bucket Strengthened 2-Matching-Cut*, separiert auf der $n40w80.001$-Instanz von *Dumas et al.* [28], illustriert. Hierbei sind einmal mehr die jeweiligen (gerichteten) Cut-Kanten als rote Pfeile hervorgehoben und die zugehörigen Cut-Knoten blau umkreist. Zugunsten der Übersicht ist auf jedwedes Andeuten der präzedenzverstärkenden Bucket-Graph-Kanten, sowie der in Gegenrichtung gedoppelten Teeth-Kanten verzichtet.

4.2.3. Infeasible Path-Inequalities

Wie in Abschnitt 2.4.3.3 bereits diskutiert, kann im Rahmen der *TSPTW-TBR* ebenfalls das Problem auftreten, dass sich Unzulässige Pfade einstellen. Sei $P = (v_1, v_2, ..., v_h)$ ein gerichteter, elementarer Pfad in G. So gilt für die Anzahl der Kanten, welche mit $|P|$ notiert werden soll, klarerweise $|P| = h - 1$. Es ist bekannt, dass das Entscheidungsproblem, ob ein Pfad in einer fraktionalen Lösung die Eigenschaft aufweist, im Sinne der Definition aus Abschnitt 4.2.3, $infeasible$ zu sein, $\mathcal{NP}$-vollständig ist[23]. Jedoch lassen sich einige, ebenfalls auf fraktionalen Lösungen gültige, hinreichende Bedingungen für Unzulässige Pfade formulieren, von welchen eine bereits im Rahmen des Lemma 2 eingeführt wurde. Konkret wurde gezeigt, dass P insbesondere *infeasible* ist, falls

$$R_{v_1} + \sum_{(v_i,v_j)\in P} \theta_{v_i,v_j} > D_h \tag{4.52}$$

gilt. Also enthält eine fraktionale Lösung, welche (Teil-)Pfade aufweist, die gemäß Kriterium (4.52) unzulässig sind, in jedem Fall Unzulässige Pfade, die Gegenimplikation gilt jedoch nicht. Im Fall einer integralen, zyklenfreien Lösung, namentlich bei Vorliegen eines Hamiltonpfades von p nach q durch G, ist die Gegenimplikation hingegen augenscheinlich gültig. Selbiger ist somit genau dann frei von Unzulässigen Pfaden, wenn (4.52) für sämtliche (Teil-)Pfade und insbesondere den Hamilton-Pfad als Ganzes, erfüllt ist. Dies bedeutet für eine Prüfroutine, welche potentielle Incumbents kontrolliert, dass das Kriterium (4.52), in Verbindung mit der Sicherstellung von Subtourfreiheit, hinreichend ist. Selbiges wird in der praktischen Umsetzung noch auszunutzen sein.

Im Rahmen einer Erweiterung des Kriteriums (4.52) kann überdies leicht gezeigt werden, dass P ebenfalls *infeasible* ist, falls sich ein Knoten v_t findet, welcher nicht im Pfad enthalten ist, und der, wenn er an den Anfang oder das Ende von P gesetzt wird, den resultierenden Pfad (zum Beispiel laut Kriterium (4.52)) *infeasible* werden lässt.

Lemma 3. Existiert für einen Pfad $P = (v_1, v_2, ..., v_h)$ mindestens ein Knoten v_t, welcher nicht im Pfad enthalten ist, so dass sowohl $P' = \{v_t, v_1, v_2, ..., v_h\}$ als auch $P'' = \{v_1, v_2, ..., v_h, v_t\}$ gemäß Kriterium (4.52) Unzulässige Pfade sind, so ist P ebenfalls ein Unzulässiger Pfad.

Beweis. Laut Voraussetzung gilt sowohl

$$R_{v_t} + \theta_{v_t,v_1} + \sum_{(v_i,v_j)\in P} \theta_{v_i,v_j} > D_h$$

als auch

$$R_{v_1} + \sum_{(v_i,v_j)\in P} \theta_{v_i,v_j} + \theta_{v_h,v_t} > D_t.$$

[23] siehe hierzu [5]

Somit kann, gegeben Pfad P wird bereist, der Knoten v_t weder vor P noch nach P besucht werden, da in beiden Fällen eine *Deadline* -Überschreitung resultiert. Folglich kann Knoten v_t gar nicht bereist werden, womit eine Tour, welche P enthält, niemals vollständig und somit niemals zulässig sein kann. Das Lemma folgt. □

Diese Tatsache wird bei der Einführung der *Bucket Tournament-Constraints* implizit von Relevanz sein.

4.2.3.1. Tournament-Cuts

Wie in Abschnitt 4.2.3 ebenfalls bereits behandelt, lässt sich für jeden Unzulässigen Pfad P eine sogenannte *infeasible path-Constraint* der Form

$$\sum_{i=1}^{h-1} x_{v_i,v_{i+1}} \leq h-2 \tag{4.53}$$

formulieren.

Definition Da klarerweise ein Pfad P_T mit $|P_T| = h-1$, der aus Kanten von P kombiniert mit einzelnen Kanten $(i,j) \notin P$ besteht, welche zwar in einem Knoten i des Pfades starten und in einem Konten j des Pfades enden, dabei jedoch in Richtung des Pfades einen oder mehrere Knoten aus P überspringen, ebenfalls kein zulässiger Pfad im Sinne des *TSPTW* sein kann, falls diese Kanten ergänzt werden, andererseits der Pfad höchstens kürzer wird, falls sie Kanten aus P ersetzen, können die in Rede stehenden Restriktionen zu den sogenannten "*Tournament-Contstraints*"(*TCs*)

$$\sum_{(i,j)\in T(P)} x_{i,j} \leq |P| - 1 \quad \forall \ P \in \Pi, \tag{4.54}$$

mit $T(P) = \{(i,j) \in A : (i,j) = (v_i, v_j) \ mit \ 1 \leq i < j \leq h\}$ und Π wie in Abschnitt 4.2.3 definiert, verstärkt werden.

Kann überdies gezeigt werden, dass sogar jeder mögliche Pfad, welcher aus den Knoten in P gebildet werden kann, unzulässig ist, so lassen sich die Tournament-Constraints verstärken zu

$$\sum_{(i,j)\in T'(P)} x_{i,j} \leq |P| - 1 \quad \forall \ P \in \Pi, \tag{4.55}$$

mit $T'(P) = \{(i,j) \in A : (i,j) = (v_i, v_j) \ mit \ 1 \leq i,j \leq h\}$.

Separations-Heuristik *Ascheuer et al.* [13] folgend, separieren *Dash et al.* [5] die *Tournament-Constraints* im Rahmen ihres *TSPTW-TBR*-Branch-And-Cut-Frameworks mittels der folgenden Enumerations-Heuristik:

Gegeben eine Lösung (x^*, y^*, z^*) der *TSPTW-TBF*, so spannt die Heuristik für jeden Wurzelknoten $v \in V \setminus \{p,q\}$ einen Baum auf, dessen Blättern $l \in V \setminus \{v\}$ je ein elementarer Pfad von l nach v durch G zugeordnet werden kann. Konkret lässt sich das Kern-Schema der Heuristik durch die rekursive Funktion beschreiben, welche in Abbildung 4.10 mittels Pseudocode dargestellt ist.

procedure ExtendEnumTreeFromNode($l \in V, P = (v_1, ..., v_h), h \in \mathbb{Z}$)
  if $\sum_{(i,j)\in T(P)} x_{i,j} \leq h - 2$ **then**
    return *STOPP*
  if $\sum_{(i,j)\in T(P)} x_{i,j} > h - 2$ und P gemäß (4.52) *infeasible* ist **then**
    Füge die zu P gehörende, verletzte *Tournament-Inequality* dem Cut-Pool hinzu.
    Prüfe vorher, ob eine Verstärkung im Sinne der Ungleichung 4.55 möglich ist.
    return *STOPP*
  if $\sum_{(i,j)\in T(P)} x_{i,j} > h - 2$ und P gemäß (4.52) nicht *infeasible* ist **then**
    for each $w \in V \setminus \{P\}$ mit $x^*_{w,l} > 0$ **do**
      $P' \leftarrow P$
      Setze w an den Anfang von P
      return ExtendEnumTreeFromNode($w, P, h + 1$)
      $P \leftarrow P'$

Abbildung 4.10.: Möglicher Pseudocode zur Beschreibung des Kern-Algorihtmus der *Tornament Constraint*-Separation

if $\sum_{(i,j)\in T(P)} x_{i,j} > h - 2$ und P gemäß (4.52) nicht *infeasible* ist **then**
  if Entsprechende *Bucket Tournament-Constraint* verletzt **then**
    Ergänze den Cut-Pool um die verletzte *Bucket Tournament-Constraint*

$$\sum_{v\in V\setminus S} \sum_{b\in L_v} y^b_{v,v_1} + \sum_{i\in S:(i,l)\in A} x_{i,l} + \sum_{(i,j)\in T(P)} x_{i,j} \leq |P|$$

  for each $w \in V \setminus \{P\}$ mit $x^*_{w,l} > 0$ **do**
    $P' \leftarrow P$
    Setze w an den Anfang von P
    return ExtendEnumTreeFromNode($w, P, h + 1$)
    $P \leftarrow P'$

Abbildung 4.11.: Modulierung der dritten If-Abfrage zur Separation von *Bucket Tournament-Constraints*

Inhaltlich erläutert, spannt die Heuristik den Enumerations-Baum also (vom jeweiligen Wurzelknoten v aus) im Rahmen einer Tiefensuche auf, wobei sich die Pfade, welche den jeweils aktuellen Blättern zugeordnet werden können, entlang der fraktional aktiven Kanten (i, j) (also jenen mit $x^*_{i,j} > 0$), von hinten nach vorne entwickeln.

Konkret wird für jedes neue Blatt l der entsprechende Pfad $P = (l, ..., v)$ daraufhin geprüft, ob seine zugehörige *Tournament-Constraint* erfüllt ist. Ist dies der Fall, so wird der Baum von diesem Blatt aus nicht weiter verzweigt, da ab hier, weil $x \leq 1$ und der Ausdruck $h - 2$ mit jedem neuen Knoten um den Wert 1 wächst, die *Tournament-Constraint* aller Söhne erst recht erfüllt sein wird.

Verletzt P die zugehörige *Tournament-Constraint* hingegen, und ist der Pfad P zusätzlich gemäß dem in Lemma 2 behandelten Kriterium (4.52) *infeasible*, so wird die entsprechende *Tournament-Constraint*

$$\sum_{(i,j)\in T(P)} x_{i,j} \leq |P| - 1$$

(beziehungsweise die verstärkte Version, im Sinne der Ungleichung 4.55) dem Cut-Pool

hinzugefügt. Sodann wird auch in diesem Fall der Baum vom aktuellen Blatt aus nicht weiter verzweigt, da gezeigt werden kann, dass jede weitere, in diesem Teilbaum resultierende *Tournament-Constraint* von der bereits separierten *Tournament-Constraint* dominiert wird.[24]

Verletzt P die zugehörige *Tournament-Constraint* indes, und ist der Pfad P zusätzlich gemäß dem in Lemma 2 behandelten Kriterium (4.52) nicht *infeasible*, so wird der Baum um jede vom aktuellen Blatt l ausgehende fraktional aktive Kante (i, j), welche noch nicht im Baum ist, erweitert. Für jeden auf diese Wiese entstehenden Sohn $w \in V$ wird der Ursprungspfad $P = (l, ..., v)$ um den Knoten w so ergänzt, dass für den jeweils resultierenden Pfad $P^w_{neu} = \{w, l, ..., v\}$ gilt.

Insbesondere implizieren die Ausführungen vom Beginn dieses Abschnittes, dass die beschriebene Separations-Heuristik im Fall einer integralen, zyklenfreien Lösung sogar zu einem exakten Separations-Algorithmus wird, da sie hier offensichtlich sämtliche Pfade, welche gemäß des Kriteriums (4.52) *infeasible* sind, aufspürt, falls selbige existieren.

Dash et al. [5] konstatieren in Anlehnung an *Ascheuer et al.* [13], dass die Laufzeit der beschriebenen Heuristik stets polynomiell beschränkt ist, da für jede fraktionale Lösung die Anzahl an Pfaden, welche die *Tournament-Constraint* verletzen können, in $O(n^k)$ liegt. Auch wenn für die Behauptung der polynomiellen Laufzeit kein streng formaler Beweis angeführt wird, so beobachten *Dash et al.* [5] in ihren Experimenten jedoch bestätigender Weise, dass die Heuristik in allen Fällen nach wenigen Rechenschritten schließt.[25]

4.2.3.2. Bucket Tournament-Cuts

Es ist evident, dass sich der Mechanismus der *Tournament-Constraint* mittels einer verfeinerten Distinguierung der Knoten-Startzeiten, basierend auf aktiven Buckets, erheblich verstärken lässt. *Dash et al.* [5] setzen dies wie folgt um:

Sei $P = (v_1, v_2, ..., v_h)$ ein beliebiger, elementarer Pfad durch G, welcher (nach dem Kriterium (4.52)) nicht zwangsläufig unzulässig sein muss. Betrachtet man nun sämtliche, potentiellen Startbuckets $b_l \in B_{v_1} = \{b_1, ..., b_k\}$ des ersten Pfad-Knotens v_1, so sei $b_t \in B_{v_1}$ die erste Bucket, ab welcher der Pfad P nach Subsumtion auf das Kriterium (4.52) *infeasible* ist, namentlich also $r_{b_t} + \sum_{(v_i,v_j)\in P} \theta_{v_i,v_j} > D_{v_h}$ gilt[26]. Folgerichtig kann der Knoten v_1 in einer zulässigen Tour, welche den Pfad P enthält, zu keiner der Buckets $\{b_t, b_{t+1}, ..., b_h\}$ verlassen werden, da jede Startbucket nach b_t den resultierenden Weg erst recht unzulässig werden lässt. Es notiere $L = \{b_t, b_{t+1}, ..., b_h\}$ die Menge eben all dieser unzulässigen Buckets.

Basierend hierauf lässt sich die zugehörige *Tournament-Constraint* sodann wie folgt verstärken:

$$\sum_{b\in L} z^b_{v_1} + \sum_{(i,j)\in T(P)} x_{i,j} \leq |P| \tag{4.56}$$

Offenbar wird die Restriktion (4.56), falls Knoten v_1 zu einer der zulässigen Startbuckets verlassen wird (also $z^b_{v_1} = 0$ für alle $b \in L = \{b_t, b_{t+1}, ..., b_h\}$ gilt), zu $\sum_{(i,j)\in T(P)} x_{i,j} \leq |P|$

[24] siehe hierzu [5]

[25] die im Rahmen dieser Arbeit durchgeführten Versuche untermauern diesen Sachverhalt

[26] für den Sonderfall eines generell Unzulässigen Pfades im Sinne des Kriteriums (4.52) gilt augenscheinlich $b_t = b_1$, da $r_{b^i_1} = R_{v_i}$ für alle Knoten $v_i \in V$

und somit, wie zu wünschen ist, redundant, da diese Ungleichung keine Kante aus P verbietet. Wird als Startbucket hingegen eine unzulässige Bucket gewählt (gilt also $z^b_{v_1} = 1$ für genau ein $b \in L = \{b_t, b_{t+1}, ..., b_h\}$) so wird die entsprechende Restriktion (4.56) zu $1 + \sum_{(i,j)\in T(P)} x_{i,j} \leq |P|$ also $\sum_{(i,j)\in T(P)} x_{i,j} \leq |P| - 1$ und nimmt somit, wie zu wünschen ist, die Form der korrespondierenden, generellen *Tournament-Constraint* an, wodurch in diesem Fall die Verwendung mindestens einer Kante (i, j) aus P ausdrücklich verboten wird.

Definition Diese Restriktion lässt sich folgendermaßen weiter verschärfen: Sei $S = \{v_1, v_2, ..., v_h\}$ die Menge aller Knoten in P, und für irgendein $v \in V \setminus S$ das Set L_v eine Menge von Buckets bei Knoten v, die in einer zulässigen Tour nicht gewählt werden dürfen, gegeben die Tour enthält den Pfad P sowie die Kante (v, v_1). Präziser ist $L_v = \{b \in B_v : r_b + \theta_{v,v_1} + \sum_{(v_i,v_j)\in P} \theta_{v_i,v_j} > D_{v_h}\}$. Sodann definieren *Dash et al.* [5] die entsprechende, sogenannte *Bucket Tournament-Constraint (BTC)* zu

$$\sum_{v\in V\setminus S} \sum_{b\in L_v} y^b_{v,v_1} + \sum_{i\in S:(i,v_1)\in A} x_{i,v_1} + \sum_{(i,j)\in T(P)} x_{i,j} \leq |P|. \tag{4.57}$$

Angelehnt an *Dash et al.* [5] sei im Folgenden die Gültigkeit der *BTC* für die *TSPTW-TBF*, sowie ihre Dominanz über die Restriktions-Familie aus (4.56) gezeigt.

Satz 10. Gegeben ein elementarer Pfad $P = (v_1, ..., v_h)$, so ist die entsprechende *Bucket Tournament-Constraint* gültig für die *TSPTW-TBF* .

Beweis. Gegeben

$$\sum_{(i,j)\in T(P)} x_{i,j} = |P|,$$

ist der Pfad P also vollständig in der gewählten Tour enthalten, so muss der Summand $\sum_{v\in V\setminus S} \sum_{b\in L_v} y^b_{v,v_1}$ den Wert 0 annehmen, um die Ungleichung nicht zu verletzen. Dies ist auch so zu wünschen, da ein Wert von 1 implizieren würde, dass eine der Buckets in L_v vor dem Pfad P besucht wird, wodurch die *Deadline* D_{v_h} des Knotens v_h gemäß Voraussetzung überschritten würde. Überdies muss zur Erfüllung der Ungleichung auch der Summand $\sum_{i\in S:(i,v_1)\in A} x_{i,v_1}$ den Wert 0 annehmen. Dies ist ebenfalls so zu wünschen, da ein Wert von 1 implizieren würde, dass von genau einem Knoten in $S \setminus \{v_1\}$ aus, zurück zu Knoten v_1 eine Subtour geschlossen würde, was in einer zulässigen Lösung ebenfalls nicht der Fall sein darf.

Ist hingegen

$$\sum_{(i,j)\in T(P)} x_{i,j} < |P|,$$

wird der Pfad also nicht vollständig gewählt, so ist zu wünschen, dass die Restriktion (4.57) redundant wird, da in diesem Fall nicht mehr zwingend zu verbieten ist, dass eine Bucket $b \in L_v$ oder eine Kante (i, v_1), mit $i \in S$, gewählt wird. Sei konkret

$$\sum_{(i,j)\in T(P)} x_{i,j} + c = |P|,$$

mit $c \geq 1$, so wird die Restriktion (4.57) zu

$$\sum_{v \in V \setminus S} \sum_{b \in L_v} y^b_{v,v_1} + \sum_{i \in S:(i,v_1) \in A} x_{i,v_1} \leq c.$$

Da sich der zweite Summand der linken Seite, wegen (2.18), schreiben lässt als

$$\sum_{i \in S:(i,v_1) \in A} x_{i,v_1} = \sum_{i \in S:(i,v_1) \in A} \sum_{b \in B_v} y^b_{i,v_1}$$

und sodann gilt, dass

$$\sum_{v \in V \setminus S} \sum_{b \in L_v} y^b_{v,v_1} + \sum_{i \in S:(i,v_1) \in A} \sum_{b \in B_v} y^b_{i,v_1} \leq \sum_{i \in V:(i,v_1) \in A} \sum_{b \in B_v} y^b_{i,v_1} = \sum_{i \in \delta^-(v_1)} x_{i,v_1},$$

wobei das Kleiner-Gleich-Verhältnis wegen $L_v \subseteq B_v$ und der Tatsache, dass nach Voraussetzung $(v, v_1) \in A$ ist, resultiert, und das Gleichheits-Verhältnis abermals wegen (2.18) und der Tatsache, dass per Definition $\delta^-(v_1)$ gerade die Menge $\{i \in V : (i, v_1) \in A\}$ ist, gilt, stellt sich in diesem Fall somit eine Redundanz zur *TSPTW-BMF* -Ungleichung (2.6) ein, welche klarerweise gültig für jede *TSPTW*-Formulierung und somit auch für die *TSPTW-TBF* ist.

□

Satz 11. Gegeben ein elementarer Pfad $P = (v_1, ..., v_h)$, so dominiert die entsprechende *Bucket Tournament-Constraint* die Restriktion (4.56).

Beweis. Für die ersten beiden Summanden der linken Seite von *Bucket Tournament-Constraint* (4.57) gilt offenbar

$$\sum_{v \in V \setminus S} \sum_{b \in L_v} y^b_{v,v_1} + \sum_{i \in S:(i,v_1) \in A} x_{i,v_1} \geq \sum_{v \in V \setminus \{v_1\}} \sum_{b \in L_v} y^b_{v,v_1},$$

da wegen (2.18) stets $\sum_{i \in S:(i,v_1) \in A} x_{i,v_1} \geq \sum_{v \in S} \sum_{b \in L_v} y^b_{v,v_1}$ ist. Überdies gilt per Definition, verfolgt man eine Bucket $b \in L$ für einen Knoten $v \in V \setminus \{v_1\}$ über die zugehörige Kante $(b', b_l) \in A_{\mathfrak{B}}$ zurück zu ihrer Vorgänger-Bucket $b' \in B_v$ bei Knoten v, dass selbige ihres Zeichens in L_v liegen muss. Daher ist

$$L \subseteq \bigcup_{v \in V \setminus \{v_1\}} \{b \in B_{v_1} : (b', b) \in A_{\mathfrak{B}} \ \textit{für irgendein} \ b' \in L_v\}$$

weswegen sodann, in Verbindung mit (2.17), gilt

$$\sum_{v \in V \setminus \{v_1\}} \sum_{b \in L_v} y^b_{v,v_1} \geq \sum_{b \in L} z^b_{v_1},$$

was bedeutet, dass

$$\sum_{v \in V \setminus S} \sum_{b \in L_v} y^b_{v,v_1} + \sum_{i \in S:(i,v_1) \in A} x_{i,v_1} + \sum_{(i,j) \in T(P)} x_{i,j} \geq \sum_{b \in L} z^b_{v_1} + \sum_{(i,j) \in T(P)} x_{i,j},$$

also die linke Seite von (4.57) immer größer oder gleich der linken Seite von (4.56) ist, woraus die Behauptung folgt. □

Separations-Heuristik Die *Bucket Tournament-Constraint* separieren *Dash et al.* im Rahmen der Enumerations-Heuristik aus Abbildung 4.10, indem sie die dritte If-Abfrage wie in Abbildung 4.11 dargestellt modulieren.

4.2.4. Primalheuristiken

In jedem Knoten des Branch-And-Cut-Baumes versuchen *Dash et al.* [5] zudem, basierend auf den entsprechenden LP-Variablen-Belegungen, mittels einer Primalheuristik eine integrale, zulässige Lösung zu konstruieren, namentlich eine gültige *TSPTW*-Tour zu schließen.

4.2.4.1. TBR-spezifische Primalheuristik

Gegeben eine aktuelle LP-Lösung (x^*, y^*, z^*), so beginnt die Heuristik beim Startknoten $p \in V$ und kann sodann im Pseudocode mittels der rekursiven Funktion beschrieben werden, welche die Abbildung 4.12 zeigt, wobei $x^m_{i,j}$ den Schattenpreis, beziehungsweise die reduzierten Kosten bezeichnen soll, welche der entsprechenden Variablen in der aktuellen Lösung zugeordnet werden können, P den Pfad beschreibt, welcher bereits konstruiert ist, i den letzten Knoten auf diesem aktuellen Pfad darstellt und t_i die entsprechende Besuchszeit des Knotens i enthält.

procedure SELECTNEXTNEIGHBOURFROM($i \in V, t_i \in \mathbb{Z}, P = (v_1, ..., v_h)$)
  $\tilde{c}_{i,j} \leftarrow x^m_{i,j}$ für alle $j \in V \setminus \{i\}$
  $N \leftarrow \{j \in V \setminus \{\{q\} \cup P\} : \tilde{c}_{i,j} = 0, t_i + \theta_{i,j} \leq D_j\}$
  if $N = \emptyset$ **then**
    return $ABBRUCH$
  $j \leftarrow argmin_{k \in N}\{D_k - \theta_{i,k}\}$
  $x_{i,j} \leftarrow 1$
  $P \leftarrow P \cup \{v_j\}$
  $t_j \leftarrow \max\{t_i + \theta_{i,j}, R_j\}$
  if Alle Knoten außer q werden von P bereist **then**
    $x_{j,q} \leftarrow 1$
    return $FERTIG$
  if SELECTNEXTNEIGHBOURFROM(j, t_j, P) = ABBRUCH **then**
    return $ABBRUCH$
  else
    return $FERTIG$

Abbildung 4.12.: Möglicher Pseudocode einer rekursiven Beschreibung des Kern-Algorithmus' der von *Dash et al.* [5] verwendeten Primalheuristik

Offenbar wird vom aktuellen Knoten i aus der Pfad wie folgt erweitert: Zunächst werden alle Nachbarn j von i sondiert, welche, gegeben die Abreisezeit t_i, noch vor ihrer *Deadline* erreicht werden können, für welche also gilt, dass $t_i + \theta_{i,j} \leq D_j$. Aus den resultierenden Verbindungskanten $(i, j) \in A$ kommen sodann jene in die engere Auswahl, deren jeweilige Variable $x_{i,j}$ in der aktuellen LP-Lösung einen Schattenpreis von $x^m_{i,j} = 0$ aufweist. Da die Größe $x^m_{i,j}$ den Wert wiedergibt, um den der entsprechende Zielfunktions-Koeffizient $c_{i,j}$ (im Fall eines Minimierungs-Problems) sinken müsste, damit die zugehörige Variable in die Simplex-Basis aufgenommen werden sollte, verheißt ein $x^m_{i,j} = 0$ folglich, dass eine Auswahl der Kante (i, j) im Hinblick auf die aktuelle LP-Lösung vorteilhaft, beziehungsweise bereits erfolgt ist, womit diese Selektions-Einschränkung sinnvoll scheint.

Sei mit $N \leftarrow \{j \in V \setminus \{q \cup P\} : x^m_{i,j} = 0, t_i + \theta_{i,j} \leq D_j\}$, wie im Pseudocode, das so resultierende Set der, laut Heuristik, final in Frage kommenden Folgeknoten notiert. Ist N für den aktuellen Knoten leer, so ist es nicht mehr möglich, das Konstruieren der zulässigen Tour zu beenden und die Heuristik bricht ab. Ist N hingegen nicht leer, so wird aus allen $j \in N$ der Knoten als effektiver Nachfolger von i ausgewählt, welcher den geringsten Puffer bezüglich der spätestmöglichen Abreisezeit von i nach j, beziehungsweise die kleinste, gleich-gewichtete Summe aus *Deadline* und negierter Reisezeit, namentlich $D_j - \theta_{i,j}$, aufweist.

Sind alle Knoten außer q im konstruierten Pfad P enthalten, so wird die gefundene, zulässige *TSPTW*-Tour entsprechend geschlossen und die Heuristik bricht mit Erfolgsmeldung ab.

Den heuristischen Mechanismus so zu konstruieren, dass er unbesuchten Knoten, welchen auf Grund ihrer Distanz und *Deadline* eine besondere Dringlichkeit zukommt, Priorität einräumt, scheint plausibel. Allerdings sind auch ambitioniertere Selektionskriterien denkbar.

4.2.4.2. Neue TBR-spezifische Primalheuristik

Sowohl die Resultate in Abschnitt 5 als auch die experimentellen Ergebnisse von *Dash et al.* [5] deuten darauf hin, dass die Effektivität des *TBR*-B&C-Frameworks erheblich von "guten" integralen Schranken in frühen Knoten des B&C-Baumes (und den daraus resultierenden Möglichkeiten des Boundings) profitiert. Daher scheint es legitim, komplexere Primalheuristiken in Betracht zu ziehen, da die potentiell erhöhten Rechenzeiten mit großer Wahrscheinlichkeit durch den entstehenden Mehrwert für das Lösungsverhalten legitimiert sind.

Experimente im Rahmen dieser Arbeit haben gezeigt, dass die von *Dash et al.* [5] verwendete Primalheuristik in vielen Fällen insbesondere an "mangelnder Voraussicht" krankt. So werden häufig Knoten als Nachfolger gewählt, welche augenscheinliche "Sackgassen" darstellen, da von selbigen aus ihres Zeichens, gemäß den verwendeten Suchkriterien, keine weiteren Nachfolger mehr in Frage kommen. Basierend auf dieser Erkenntnis scheint beispielsweise eine an die (für das *Springerproblem* entwickelte) *Wahrnsdorf-Regel*[27] angelehnte Erweiterung der Heuristik sinnvoll. Selbige orientiert sich bei der Bewertung eines potentiellen nächsten Feldes auf dem Schachbrett an der Anzahl der von dort aus für den Springer möglichen Folgefelder. Sodann wird eines der Felder als das nächste gewählt, welche im Rahmen dieser Kalkulation den geringsten Wert aufweisen, folglich also als "pressierend" einzustufen sind. Der Pseudocode einer denkbaren Variation dieser Strategie, zur Verstärkung der ursprünglichen Primalheuristik, ist in Abbildung 4.13 dargestellt.

Augenscheinlich werden nach wie vor zunächst alle diejenigen noch nicht im Pfad befindlichen Nachbarn j von i als potentielle Nachfolger eingestuft, also in der Menge N gespeichert, für welche $(t_i + \theta_{i,j} \leq D_j) \wedge (x^m_{i,j} = 0)$ gilt.

Sodann soll jedoch zusätzlich für jeden dieser Knoten $j \in N$ die Menge $PotSuc(j)$ der wiederum von diesem Knoten aus in Frage kommenden Nachfolger j' bestimmt werden, welche noch nicht im Pfad enthalten sind. Selbige müssen nun konsequenterweise der Bedingung $(\max\{t_i + \theta_{i,j}, a_j\} + \theta_{j,j'} \leq D_{j'}) \wedge (x^m_{j,j'} = 0)$ genügen. Insbesondere wird mittels des Maximierungs-Operators bei dem Test auf eine potentielle Überschreitung der *Dead-*

[27] für Erläuterungen zum Begriff des *Springerproblems* und zum Begriff der *Warnsdorf-Regel* siehe beispielsweise [29] und [67]

procedure SELECTNEXTNEIGHBOURFROMNEW($i \in V, t_i \in \mathbb{Z}, P = (v_1, ..., v_h)$)
  $\tilde{c}_{i,j} \leftarrow x^m_{i,j}$ für alle $j \in V \setminus \{i\}$
  $N \leftarrow \{j \in V \setminus \{\{q\} \cup P\} : \tilde{c}_{i,j} = 0, t_i + \theta_{i,j} \leq D_j\}$
  if $N = \emptyset$ **then**
    return $ABBRUCH$
  for all $j \in N$ **do**
    $PotSuc(j) \leftarrow \{j' \in V \setminus \{P\} : \tilde{c}_{j,j'} = 0, \max\{t_i + \theta_{i,j}, R_j\} + \theta_{j,j'} \leq D_{j'}\}$
    $Puffer(j) \leftarrow \sum_{j' \in PotSuc(j)} [D_{j'} - (\max\{t_i + \theta_{i,j}, R_j\} + \theta_{j,j'})]$
  $j \leftarrow argmin_{k \in N : |PotSuc(k)| > 0}\{\alpha * (D_k - \theta_{i,k}) + (1 - \alpha) * Puffer(k)/|PotSuc(k)|\}$
  $x_{i,j} \leftarrow 1$
  $P \leftarrow P \cup \{v_j\}$
  $t_j \leftarrow \max\{t_i + \theta_{i,j}, R_j\}$
  if Alle Knoten außer q werden von P bereist **then**
    $x_{j,q} \leftarrow 1$
    return $FERTIG$
  if SELECTNEXTNEIGHBOURFROM(j, t_j, P) = ABBRUCH **then**
    return $ABBRUCH$
  else
    return $FERTIG$

Abbildung 4.13.: Möglicher Pseudocode einer rekursiven Beschreibung des Kern-Algorithmus' der neuen Primalheuristik

line des Knotens j' eine mögliche, Zeitfenster-bedingte Wartezeit bei j in die Kalkulation mit einbezogen. Überdies wird für jeden Knoten $j \in N$ der zu der Menge $PotSuc(j)$ korrespondierende, kumulierte zeitliche Puffer $\sum_{j' \in PotSuc(j)} [D_{j'} - (\max\{t_i + \theta_{i,j}, a_j\} + \theta_{j,j'})]$ berechnet.

Nun soll jener Knoten $k \in N$ als tatsächlicher Nachfolger gewählt werden, welcher die minimale, (mittels $\alpha \in [0, 1]$) gewichtete Summe aus dem unmittelbaren Puffer $D_k - \theta_{i,k}$ und dem jeweils durchschnittlichen Folgeknoten-Gesamtpuffer $Puffer(k)/|PotSuc(k)|$ aufweist, wobei die entscheidende Änderung darin besteht, dass potentielle Nachfolger k mit

$$|PotSuc(k)| = 0$$

aus der Betrachtung ausgeschlossen sind, womit das eingangs beschriebene, primäre Problem der ursprünglichen Primalheuristik eliminiert ist.

In gezielten Experimenten bezüglich des Gewichtungsfaktors hat sich eine Wahl von $\alpha = 0.95$ als günstig erweisen, womit es offenbar zweckdienlich ist, dem Einbezug zeitlicher Puffer der unmittelbaren Nachfolger eine deutlich größere Rolle zukommen zu lassen, als dem der jeweiligen, durchschnittlichen Folgepuffer, was den Kern-Mechanismus der von *Dash et al.* [5] vorgeschlagenen Heuristik wiederum kreditiert.

5. Experimente

In diesem Kapitel werden die verschiedenen (Haupt-)Experimente, welche im Rahmen dieser Arbeit getätigt wurden, beschrieben und deren Resultate zusammengefasst.

5.1. GAMS 23.8 und die BCH-Facility

In diesem Abschnitt sei die Entwicklungsumgebung `GAMS 23.8` [58], in welcher die Experimente umgesetzt sind, kurz vorgestellt.

Das "`General Algebrahic Modeling System`" (kurz: "`GAMS`") ist eine algebraische Modellierungssprache für mathematische Optimierungsprobleme.[1] Neben den "herkömmlichen" Basiselementen einer Programmiersprache, wie beispielsweise den Möglichkeiten des Konstruierens bedingter Abfragen und Anweisungs-Schleifen, vereinfacht `GAMS` das Formulieren Gemischt-Linearer-Optimierungs-Programme und im Rahmen des Lösungsverfahrens derselbigen die Kommunikation mit gängigen MIP-Solvern wie beispielsweise CPLEX [61], AMPL [62] oder SCIP [63]. Überdies bietet `GAMS` seit der Version `21.3` die Erweiterung der sogenannten `BCH-Facility`, welche dem User (beispielsweise im Fall der Anwendung von CPLEX) eine dynamische Interaktion mit dem Solver während des B&C-Algorithmus' ermöglicht, und im Rahmen dessen die Option bietet, sowohl im `GAMS`-Code programmierte Separations- beziehungsweise Cutting-Routinen als auch im `GAMS`-Code programmierte Primalheuristiken unter der Deklaration verschiedener Parametereinstellungen, in das Branch-And-Cut-Framework einzubinden.

Es sei angemerkt, dass `GAMS`, neben seinen Vorteilen, spezielle Paradigmen birgt, welche insbesondere Schleifendurchläufe, die in Programmiersprachen wie `C++` über Array-Indizes erfolgen würden, stattdessen auf sogenannten "`Sets`" ausführt, was die Laufzeit Loop-lastiger[2] Quelltexte (welche insbesondere bei der Implementation der komplexen, "mengenspezifischen" *Bucket-SOP-Inequalitites*, *Infeasible Path-Inequalities* und *BS2M-Inequalities* unvermeidlich sind) erheblich verlangsamt. Überdies ist hervorzuheben, dass `GAMS` keine Rekursion unterstützt und daher sämtliche, in den Pseudocdes rekursiv präsentierten Funktionen stattdessen "straight forward" implementiert sind.

5.2. Verwendete Instanzen

Im Folgenden seien die verwendeten Experimental-Instanzen kurz eingeführt und ihren entsprechenden, literarischen Ursprüngen zugeordnet.

[1] vgl. hierzu [55]

[2] "Loop-lastig" in dem Sinne, dass Iterationen über große `Set`-Tupel, gepaart mit sehr spezifischen Abfragen bezüglich der jeweiligen Werteausprägungen von Nöten sind, wie beispielsweise bei jeder Neuermittlung, sowie jedem Auslesen der Hilfskantenmengen $A^S_{\mathfrak{B}}$ beziehungsweise $A^{u,w}_{\mathfrak{B}}$ (in `GAMS` je ein Sub-`Set` über insgesamt vier `Sets` - zweimal das zugrunde liegende Knoten- und zweimal das zugrunde liegende Bucket-`Set`) im Rahmen der Separationen von *Bucket-SOP-Inequalities*

5.2.1. Dumas et al.-Instanzen

Die "*Dumas et al.*-Instanzen" führen *Dumas, Desrosiers und Gelinas* [28] im Rahmen ihres Artikels zu einem exakten Lösungsalgorithmus für das $TSPTW$ein. Die in dieser Arbeit verwendeten Dateien der besagten Sammlung weisen Städteanzahlen von $n = 20$ bis $n = 100$ und durchschnittliche Zeitfensterweiten von $w = 60$ bis $w = 100$ auf (die Hilfs-Knoten $p, q \in V$ jeweils ausgeschlossen).

Sämtliche Daten, namentlich die Koordinaten sowie die Zeitfenster der einzelnen Kunden, sind zufallsgeneriert. Die implizite Angabe der Knoten-Distanzen mittels Kunden-Koordinaten birgt den Vorteil, dass es möglich ist, sämtliche errechneten Lösungen und Zwischenresultate adäquat zu visualisieren, wie es bereits im Rahmen diverser Abbildungs-Vorgriffe dieser Arbeit ausgenutzt wurde[3]. Die schlussendliche Distanzmatrix $c(= \theta)$ soll sodann mittels der (gerundeten) euklidischen Abstände zwischen den Knoten bestimmt werden. Hierbei ist zu beachten, dass die Distanzen nach dem Rundungs-Vorgang nicht mehr zwingend der Dreiecksungleichung unterliegen müssen (dies kann dann passieren, wenn der Rundungseffekt das Ursprungsverhältnis der Schenkelkanten $(i, k), (k, j)$ zu der direkten Verbindungskante (i, j) überkompensiert, wie es beispielsweise für die Parameterkonstellation $c_{i,k} = 1.4, c_{k,j} = 1.4, c_{i,j} = 2.6$ der Fall ist, welche gerundet zu $c_{i,k} = 1, c_{k,j} = 1, c_{i,j} = 3$ wird). Um die besagte Metrik wieder zu gewährleisten, soll die in *Dumas et al.* [28] vorgeschlagene Hilfs-Routine Anwendung finden, welche mittels des Pseudocodes aus Abbildung 5.1 beschrieben werden kann.

```
procedure RestoreTriangleInequality
    Okay := 0
    while Okay = 0 do
        Okay := 1
        for each i ∈ V \ {q} do
            for each j ∈ V \ {i, p} do
                for each k ∈ V \ {i, j, p, q} do
                    if c_{i,j} < c_{i,k} + c_{k,j} then
                        c_{i,j} := c_{i,k} + c_{k,j}
                        Okay := 0
```

Abbildung 5.1.: Möglicher Pseudocode der Hilfsroutine von *Dumas et al.* [28] zur Wiederherstellung der Dreiecksungleichungs-Gültigkeit in der Distanzmatrix

Offenbar wird so lange über sämtliche Knoten-Tripel (welche der Definition aus Abschnitt 2.1.2 genügen - zu diesem Zeitpunkt sind auf den in Rede stehenden Instanzen grundsätzlich alle Knotenpaare adjazent, außer jene mit p als End- und/oder q als Start-Knoten) iteriert, und eine eventuelle Dreiecksungleichungs-Verletzung mittels einer "Verlängerung" der jeweiligen direkten Verbindungskante (i, j) behoben, bis im letzten Iterations-Durchlauf über sämtliche Knoten-Tripel keine Verletzung mehr aufgespürt wurde.

Abbildung 5.2.1 zeigt beispielhaft die $n20w100.001$-Instanz von *Dumas et al.* [28] im *txt*-Format. Hierbei sind die Dateinamen so angelegt, dass die Ziffer hinter "n" die Anzahl der Städte (ohne die Hilfsknoten $p, q \in V$) und die Ziffer hinter w die entsprechende, durchschnittliche Zeitfensterweite wiedergibt. Hinter dem Punkt findet sich die Instanz-Nummer bezogen auf die entsprechende Instanz-Klasse (wobei unter eine Instanz-Klasse alle Instanzen fallen, welche die gleiche Städteanzahl und die gleiche (durchschnittliche)

[3] besagte Abbildungen sind mittels der `GAMS-Put-Facility` in Verbindung mit LaTeX [59] erstellt

Zeitfensterweite aufweisen).

```
!! n20w100.001    96.50 363

CUST NO.   XCOORD.   YCOORD.    DEMAND   READY TIME   DUE DATE   SERVICE TIME

    1      13.00      43.00       0.00        0.00     405.00       0.00
    2      35.00      27.00       0.00        0.00     133.00       0.00
    3      21.00      34.00       0.00        0.00     117.00       0.00
    4      20.00       9.00       0.00      156.00     260.00       0.00
    5      35.00      31.00       0.00        0.00     158.00       0.00
    6      38.00      23.00       0.00        0.00      90.00       0.00
    7      28.00      20.00       0.00       69.00     153.00       0.00
    8       6.00       1.00       0.00      221.00     280.00       0.00
    9      31.00      29.00       0.00        0.00      83.00       0.00
   10      11.00      40.00       0.00        0.00      33.00       0.00
   11       9.00      28.00       0.00       14.00     124.00       0.00
   12      19.00      35.00       0.00        0.00      25.00       0.00
   13      37.00      14.00       0.00      198.00     330.00       0.00
   14       6.00       5.00       0.00      116.00     185.00       0.00
   15       2.00      37.00       0.00       81.00     182.00       0.00
   16      30.00      50.00       0.00       44.00     126.00       0.00
   17      31.00      37.00       0.00       31.00     156.00       0.00
   18      24.00       2.00       0.00      134.00     271.00       0.00
   19       8.00      22.00       0.00      147.00     213.00       0.00
   20      38.00      31.00       0.00      243.00     377.00       0.00
   21      48.00      24.00       0.00      270.00     358.00       0.00
  999       0.00       0.00       0.00        0.00       0.00       0.00
```

Abbildung 5.2.: Abbildung der $n20w100.001$-Instanz von *Dumas et al.* [28] im "*txt*"-Format

```
16
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
70 0 81 80 70 93 85 85 86 88 94 90 85 80 84 86
46 46 0 57 56 69 60 67 61 65 70 65 67 57 61 64
35 46 47 0 45 58 51 53 53 55 59 56 53 37 47 50
35 45 46 45 0 58 50 53 51 53 59 55 53 45 49 51
66 90 90 89 89 0 92 89 93 93 97 94 89 89 84 81
42 54 54 57 56 58 0 63 53 61 63 61 63 57 63 65
53 65 65 68 67 69 63 0 64 72 74 72 74 68 74 76
56 80 80 81 80 68 77 80 0 74 66 74 80 81 83 84
55 70 69 73 71 69 65 76 55 0 73 74 76 73 78 79
48 71 69 71 71 48 63 71 62 60 0 60 71 71 73 74
55 79 78 79 79 67 75 79 73 74 55 0 79 79 82 83
53 65 65 68 67 69 63 74 64 72 74 72 0 68 74 76
34 45 46 37 44 57 50 52 52 54 58 55 52 0 46 49
46 62 64 60 61 71 69 61 69 70 73 70 61 60 0 56
51 66 67 63 65 76 74 72 74 74 78 75 72 63 51 0
0         5800
0         600
76        676
808       1408
845       1445
895       1495
1136      1736
1403      2003
1426      2026
1556      2156
1594      2194
1707      2307
1626      2226
1602      2202
2236      2836
2300      2900
```

Abbildung 5.3.: Abbildung der $rgb17$-Instanz von *Ascheuer* [25] im "*tw*"-Format

Im Rahmen der Experimente ist von jeder Instanz-Klasse je die erste Datei verwendet. Bei den Angaben der Instanz-Namen in den Tabellen entfällt die entsprechende dreistellige Instanznummer .001 aus Gründen der Platzersparnis.

5.2.2. Ascheuer-Instanzen

Die "*Ascheuer*-Instanzen" präsentiert *Ascheuer* [25][4] im Rahmen seiner Dissertation "Hamiltonian Path Problems in the On-line Optimization of Flexible Manufacturing Systems". Die in dieser Arbeit verwendeten Dateien der besagten Sammlung weisen Städteanzahlen von $n = 17$ bis $n = 41$ und durchschnittliche Zeitfensterweiten von (ca.) $w = 600$ auf (die Hilfs-Knoten $p, q \in V$ jeweils ausgeschlossen).

Die Daten, namentlich Distanzmatrix $c(= \theta)$ sowie Zeitfenster, sind aus dem Praxis-Problem eines Stapel-Kranes abgeleitet, welcher diverse Transportaufgaben in einem "single aisled automatic storage system" zu erfüllen hat und hierbei bestimmte Abfolgen, beziehungsweise Zeitfenster, berücksichtigen muss. Die Distanzmatrix ist von vornherein metrisch, unterliegt also in sämtlichen Kanten-Tupeln im Sinne von Abschnitt 2.1.2 der Dreiecksungleichung.

Abbildung 5.2.1 zeigt beispielhaft die $rgb17.tw$-Instanz von *Ascheuer* [25] im *txt*-Format. Hierbei sind die Dateinamen so angelegt, dass die Ziffer hinter "rgb" die Anzahl der Städte (in der Regel inklusive Hilfsknoten $p, q \in V$) und ein eventueller Buchstabe hinter dieser Ziffer die entsprechende Instanz-Nummerierung bezogen auf die jeweilige Instanz-Klasse wiedergibt.

5.3. Versuchs-Setup

In diesem Abschnitt sei das Versuch-Setup vorgestellt und in seinem Aufbau kurz diskutiert.

Präparation der Instanzen Im Rahmen der Experimente dieses Abschnittes sind die Instanzen grundlegend wie folgt präpariert:

Auf den *Dumas et al.*-Instanzen wird die jeweils erste Stadt (namentlich jene in der obersten Zeile) mit sämtlichen ihrer Eigenschaften gedoppelt und je einmal als Hilfsknoten $p \in V$, sowie einmal als Hilfsknoten $q \in V$ deklariert. Sodann wird die Distanzmatrix, wie in Abschnitt 5.2.1 beschrieben, ermittelt (wobei für jedes Knotenpaar $i, j \in V, i \neq q, j \neq p$ je eine "Hin-" und eine "Rück-Kante" mit $c_{i,j} = c_{j,i}$ kreiert werden). Auf den *Ascheuer*-Instanzen indes wird die Distanzmatrix um je eine Null-Zeile und eine Null-Spalte erweitert. Sodann werden die erste Zeile- beziehungsweise Spalte dem Hilfsknoten $p \in V$, und die letzte Zeile- beziehungsweise Spalte dem Hilfsknoten $q \in V$ zugeordnet.[5]

Nachdem die Instanzen dem in Kapitel 3 beschriebenen, allgemeinen Preprocessing unterzogen wurden, werden sie der in Abschnitt 4.1 vorgestellten Iterativen Bucket-Refinement-Heusristik übergeben. Im Rahmen dessen wird über sämtliche *Dumas et al.*-Instanzen je 10 mal, über sämtliche *Ascheuer*-Instanzen je 25 mal iteriert. In allen Fällen stellt sich eine Stagnation des Zielfunktionswertes der resultierenden *TSPTW-TBR* -Relaxation (namentlich der Dualen Schranke) lange vor der letzten Iteration ein. Es sei angemerkt, dass die beschriebenen Präparations-Prozesse einmal mehr Routinen darstellen,

[4] in der angelehnten Publikation von *Ascheuer et al.* [13] werden sie erneut aufgegriffen

[5] Im Rahmen des Einlesens und der Manipulation entsprechender Instanz-Dateien kommen ebenfalls die Programme *MS Visual Basic 6.0* [60] sowie *MS Excel 2010* [57] zum Einsatz

welche sich in der Entwicklungsumgebung `GAMS` verhältnismäßig zeitkonsumierend gestalten. Daher werden sie in den folgenden Experimentalauswertungen als "vorab gegeben" behandelt und nicht in den Angaben der CPU-Zeiten berücksichtigt.

Selektion der Instanzen Die Selektion der *Dumas et al.*-Instanzen ist getroffen, um das Spektrum größerer Städtezahlen bei "kleinen"[6] Zeitfenstern, sowie steigender Städteanzahl, bei in ähnlichem Maße sinkender Zeitfenstergröße, abzudecken. Es sei anmerkt, dass die entsprechenden Zeitfenster, trotz ihres relativ geringem Umfangs, die jeweils größten Weiten aufweisen, welche auf den *Dumas et al.*-Instanzen zu den entsprechenden Städteanzahlen zu finden sind. Die Selektion der *Ascheuer*-Instanzen ist indes getroffen, um das Spektrum von steigenden Städteanzahlen bei konstant "großen"[7] Zeitfensterweiten abzudecken.

Es sei ergänzend angemerkt, dass die Auswahl der Dateien $n20w100.txt$ von *Duma et al.*, sowie $rgb17.tw$ von *Ascheuer* im Hinblick auf experimentelle Untersuchungen strenggenommen obsolet ist, da die beiden entsprechenden Instanzen, aufgrund ihrer sehr geringen Städteanzahl, ungeachtet der Zeitfensterweiten, "zu einfach" sind, als dass sie differenzierbare Reaktionen auf verschiedene Modulationen des Lösungsvorgehens aufweisen würden. Sie sind lediglich im Sinne eines vollständigen Benchmarkings aufgeführt und sollen fortan aus Unterscheidungsgründen auch als die "*trivialen* Instanzen" bezeichnet werden, während im Fall aller anderen Experimentaldateien von den "*nicht-trivialen* Instanzen" die Rede sein soll.

Überdies wurden im Rahmen dieser Arbeit auch auf anderen als den aufgeführten Instanzen, sondierende Experimente durchgeführt. Diesen Versuchsreihen entstammen beispielsweise die diversen Vorgriffs-Abbildungen aus den vorangegangenen Kapiteln.

Umsetzung des B&C-Frameworks Die konkrete Umsetzung des "Basis"-*TSPTW-TBR* - B&C-Frameworks wird sodann wie folgt gestaltet, wobei im Rahmen dessen auf den Solver GAMS/CPLEX [61] in Verbindung mit der `BCH-Facility` zurückgegriffen wird und, *Dash et al.* [5] folgend, alle Default-CPLEX-Cuts sowie Default-CPLEX-Primalheuristiken, deaktiviert sind:

Aus Rücksicht auf die Attribute der Entwicklungsumgebung `GAMS` in Verbindung mit der `BCH-Facility`, wird die Separation der in Abschnitt 4.2.1 beschriebenen *Bucket-SOP-Inequalities* sowie der in Abschnitt 4.2.3 beschriebenen *Infeasible Path-Inequalitites*[8] jeden dritten B&C-Knoten (statt jeden ersten B&C-Knoten) vorgenommen. Die in Abschnitt 4.2.4.1 beschriebene Primalheurstik wird indes für jeden B&C-Knoten aufgerufen. Die Kontrollroutine, welche einen neuen Incumbent auf seine Zulässigkeit bezüglich der zugrunde liegenden *TSPTW*-Instanz untersucht, stellt zunächst die Subtourfreiheit der Lösung sicher. Ist selbige gegeben, liegt also ein Hamiltonweg vor, so wird überprüft, ob die vom besagten Weg induzierten Startzeiten $Startzeit_i$ in den Zeitfenstern W_i der entsprechenden Knoten $i \in V \setminus \{q\}$ liegen. Ist dies der Fall, so wird der Incumbent akzeptiert. Ansonsten wird er verworfen. Im Rahmen dessen werden auf unzulässigen Incumbents *keine* Cuts separiert und an den Cut-Pool weitergegeben, da dies in der wünschenswerten sowie zeit-effizienten Form nicht von der `BCH-Facility` unterstützt wird (konkret lässt sich die Separationsroutine zwar, im Rahmen eines außerordentlichen Aufrufes, gezielt für Incumbents verwenden, die entsprechenden Separationen fallen jedoch (im Gegensatz zu

[6] Vgl. hierzu beispielsweise [54]

[7] Tatsächlich zählen die Zeitfenster der *Ascheuer*-Instanzen, im Bereich der exakten TSPTW-Algorithmen, bis heute zu den größten in der entsprechenden, gängigen Literatur. Vgl. hierzu beispielsweise [54]

[8] ohne die Verstärkung

der Kontrollroutine) auch auf integralen Instanzen sehr zeitkonsumierend aus, während das Weitergeben der ermittelten Schnitte an den Cut-Pool erst im Rahmen des nächsten, regulären Separtionsroutinenaufrufes möglich ist. Dies ist dann im Hinblick auf die Gesamtlaufzeit höchst problematisch, wenn sich mehrere hundert unzulässige Incumbents hintereinander aufreihen, wie es insbesondere auf größeren *Ascheuer*-Instanzen häufig der Fall sein kann).[9]

Es sei angemerkt, dass die CPU-Zeiten instanzübergreifend nicht ohne Weiteres vergleichbar sind, da im Rahmen der Experimente zwei verschiedene Rechenmaschinen zum Einsatz kamen (namentlich ein *Intel(R) Core(TM) i5, 2.53 GHz* und ein *Intel(R) Core(TM)2 Quad, 2.40 GHz*). Die Resultate innerhalb einer Instanz sind jedoch stets auf dem selben Terminal errechnet und somit gegenüberstellbar.

5.4. Resultate

In diesem Abschnitt seien die Resultate der verschiedenen Experimente präsentiert.

Tabelle 5.1.: Erläuternde Übersicht der Spaltenköpfe

Spaltenkopf	**Beschreibung**
Instanz	Name der Instanz
$\lvert\overline{V}\rvert$	Anzahl der Knoten aus $V \setminus \{p, q\}$
$\lvert\overline{W}\rvert$	Durchschnittliche Zeitfensterweite der Knoten aus $V \setminus \{p, q\}$
OPT	Zielfunktionswert des Optimums
Cutted lpRB	Duale Schranke im Wurzelknoten nach Cuts
Nds 1.Int	Anzahl der B&C-Knoten bis zum ersten zulässigen Incumbent
rGAB 1.Int	Tatsächliches, relatives OPT-GAP des ersten zulässigen Incumbents
Nds OPT	Anzahl der B&C-Knoten bis zum optimalen (zulässigen) Incumbent
Nds total	Gesamtanzahl der B&C-Knoten
CPUt total	Insgesamt benötigte CPU-Zeit
Nds AVG	Durchschnittlich je B&C-Knoten konstruierte Tourlänge
Nds Var	Varianz der konstruierten Tourlängen

Tabelle 5.1 gibt einen Übersicht über ausgewählte Spaltenköpfe der folgenden Tabellen und liefert jeweils eine kurze Erläuterung.

5.4.1. Auswirkungen der Ergänzung von BS2M-Cuts

Aufbauend auf den Resultaten sondierender Experimente bezüglich der Zeit-Effizienz verschiedener Schnitt-Frequenzen und Tupel-Selektionen, werden die *BS2M-Cuts* wie folgt in das B&C-Framework integriert: Im Rahmen jedes Aufrufes der Cutting-Routinen werden ebenfalls die *BS2M-Cuts*, wie in Abbildung 4.9 beschrieben, separiert, jedoch lediglich alle 3 Aufrufe für sämtliche Knoten-Tupel (i, j). In den verbleibenden zwei Aufrufen werden,

[9] Es ist nicht auszuschließen, dass eine Adjustierung der beschriebenen Abweichungen hin zu dem exakten, von *Dash et al.* [5] verwendeten B&C-Framework, stellenweise zu Veränderungen, beziehungsweise Relativierungen der im Folgenden präsentierten Ergebnisse führen würde, es ist jedoch einzusehen, dass das verwendete *TSPTW-TBR*-B&C-Framework dennoch das Abschätzen adäquater Tendenzen gestatten sollte.

einem heuristischen Vereinfachungs-Vorschlag von *Araóz et al.* [65] folgend, ausschließlich jene Knoten als Quellen deklariert, welche die Eigenschaft $odd^{>}$ aufweisen.

Die Performance des so resultierenden, *BS2M*-Cuts-unterstützen *TSPTW-TBR* -B&C-Frameworks wird sodann der des "Basis"-*TSPTW-TBR* -B&C-Frameworks gegenübergestellt.

Ergebnisse In Tabelle 5.4.1 sind die Resultate des beschriebenen Vergleichs, auf den verwendeten Test-Instanzen, aufgeführt.

Für alle Instanz-Tupel mit und ohne *BS2M*-Cuts werden jeweils von links nach rechts die folgenden Daten berichtet: Der Name der Instanz (im Fall der *BS2M*-Cuts-Ergänzungen erweitert um den Index M), die Städteanzahl der betrachteten Instanz (wobei die Hilfsknoten $p, q \in V$ jeweils ausgenommen sind), die durchschnittliche Zeitfensterweite dieser Städte (aus Platzgründen bis auf die erste Vorkommastelle gerundet), der Zielfunktionswert der optimalen (bezogen auf die zugrunde liegende *TSPTW*-Instanz zulässigen) Lösung der betrachteten Instanz, der LP-Zielfunktionswert im Wurzelknoten nach Abschluss sämtlicher Separations-Durchgänge sowie Primalheuristiken (namentlich die LP-Root-Bound nach Cuts), die Anzahl der *bearbeiteten* B&C-Knoten bis zum ersten (bezogen auf die zugrunde liegende *TSPTW*-Instanz zulässigen) Incumbent, das tatsächliche relative Optimalitäts-GAP dieses Incumbents in Prozent (berechnet zu $1 - ZF_{Inc}/OPT$, mit ZF_{Inc} als Zielfunktionswert des in Rede stehenden Incumbents), die Anzahl der *bearbeiteten* B&C-Knoten bis zu dem Incumbent, welcher dem gesuchten (bezogen auf die zugrunde liegende *TSPTW*-Instanz zulässigen) Optimum entspricht, die Anzahl der insgesamt benötigten B&C-Knoten sowie die insgesamt benötigte CPU-Zeit.

Tabelle 5.2.: Experimentalergebnisse zur Ergänzung von BS2M-Cuts

Instanz	$\lvert\overline{V}\rvert$	$\lvert\overline{W}\rvert$	*OPT*	**Cutted lpRB**	**Nds 1.Int**	**rGAP 1.Int**	**Nds OPT**	**Nds total**	**CPUt total**
n20w100	20	100	237.00	237.00	0	0.00%	0	0	50
n20w100$_M$	20	100	237.00	237.00	0	0.00%	0	0	95
n40w100	40	100	429.00	414.67	15	0.46%	41	46	8193
n40w100$_M$	40	100	429.00	414.44	16	0.23%	24	29	3739
n80w80	80	80	624.00	624.00	2	0.00%	2	3	5343
n80w80$_M$	80	80	624.00	623.56	1	0.32%	2	3	8116
n100w60	100	60	655.00	654.58	3	0.00%	3	4	13891
n100w60$_M$	100	60	655.00	652.93	6	0.00%	6	13	33343
rgb017	15	600	847.00	846.00	1	0.24%	2	3	86
rgb017$_M$	15	600	847.00	846.00	1	0.00%	1	2	161
rgb031a	31	600	1817.00	1813.77	1	0.76%	6	14	698
rgb031a$_M$	31	600	1817.00	1814.64	0	0.49%	3	6	1140
rgb034a	34	606	2169.00	2168.00	83	0.00%	83	84	15040
rgb034a$_M$	34	606	2169.00	2168.00	2	0.00%	2	4	2010
rgb041a	41	606	2547.00	2535.48	34	1.51%	1261	1581	23911
rgb041a$_M$	41	606	2547.00	2535.51	23	0.20%	466	697	9808

Eine dramatische Verbesserung des Lösungsverhaltens mittels Ergänzung der *BS2M*-Cuts auf den größeren betrachteten *Ascheuer*-Instanzen ist offensichtlich: Die Instanz $rgb041a$ schließt nach 697 statt 1581 Knoten und dies in weniger als der Hälfte der oh-

ne *BS2M*-Cuts benötigten Zeit, die Instanz $rgb034a$ sogar nach 4 statt 84 Knoten in weniger als einem Siebtel der ohne *BS2M*-Cuts benötigten Zeit. Getrieben wird dieser Performance-Unterschied aber augenscheinlich weniger durch eine Verbesserung der Dualen Root-Bounds als viel mehr durch eine generelle, positive Beeinflussung der LP-Strukturen und Dualen Schranken in den fortlaufenden Söhnen des B&C-Baumes. So findet sich der erste zulässige $rgb041a$-Incumbent im Fall der *BS2M*-Cut-Unterstützung bereits nach 23 statt nach 34 Knoten und weist zudem ein deutlich besseres, reales Optimalitäts-GAP, namentlich 0.20% statt 1.51%, auf. Der erste zulässige $rgb034a$-Incumbent resultiert gar nach 2 statt nach 83 Knoten (wobei es sich hier in beiden Fällen bereits um das gesuchte Optimum handelt - namentlich ist das reale GAP je 0.00%). Zudem kann auf den Ascheuer-Instnazen das gefundene, tatsächliche Optimum im Fall der ergänzten *BS2M*-Cuts jeweils nach deutlich weniger Knoten als selbiges identifiziert werden, was als Indikator für die positiven Auswirkungen der *BS2M*-Schnitte auf die Dualen Schranken im Verlauf des Branch-And-Cut-Frameworks betrachtet werden kann.

Auf den kleineren *Ascheuer*-Instanzen deutet sich jedoch auch an, dass die Mehrwert-Entfaltung von *BS2M*-Cuts auf eine tendenziell höhere Städteanzahl angewiesen zu sein scheint. So schließt die Instanz $rgb031a$ bei *BS2M*-Cut-Ergänzung zwar ebenfalls nach deutlich weniger Knoten, die hierfür benötigte totale CPU Zeit ist jedoch höher, da der Zeitgewinn im B&C-Baum den zeitlichen Mehraufwand für die Separation der *BS2M*-Cuts nicht mehr überkompensieren kann. Gleiches gilt auf der $rgb017$-Instanz.

Intuitiv einzusehen ist, dass eine generelle Wahrscheinlichkeit für die Verletzung von $2M$-Cuts, und somit auch von *BS2M*-Cuts, aufgrund ihrer komplexeren, geometrischen Struktur, durch umfangreichere Zeitfensterweiten (und somit tendenziell größere NWTs) erhöht, durch kleinere Zeitfensterweiten hingegen verringert werden sollte. In der Tat deuten die Resultate auf den *Dumas et al.*-Instanzen ihres Zeichens an, dass die Mehrwert-Entfaltung von *BS2M*-Cuts ebenfalls durch sinkende Zeitfenstergrößen erheblich gehemmt wird. Vielmehr scheint die *BS2M*-Cut-Ergänzung auf manchen dieser Instanzen die Struktur der B&C-Baum-LPs sogar nachgerade "zu stören". So werden die LP-Root-Bounds unter Ergänzung der *BS2M*-Cuts stellenweise geringfügig verschlechtert (dass es prinzipiell möglich ist, einen schlechteren LP-Wert zu erhalten, obwohl zusätzliche Cuts die Schranke intuitiverweise höchstens verbessern sollten, liegt daran, dass bereits im Wurzelknoten durchschnittlich 5 bis 6 Separations-Iterationen durchgeführt werden, womit sich die LP-Strukturen, und somit deren optimale Zielfunktionswerte, nach dem ersten Separationsdurchgang in verschiedene "Richtungen" entwickeln können). Ebenfalls der positive Effekt auf die Anzahl benötigter Knoten ist geschmälert, im Fall der $n100w60$ Instanz sogar ins Gegenteil umgekehrt, womit auch die benötigte totale CPU-Zeit durch die Ergänzung von *BS2M*-Cuts erhöht ist. Eine Ausnahme bildet die $n40w100$-Instanz, welche mit *BS2M*-Cuts nach 29 statt 46 Knoten in etwas weniger als der Hälfte der benötigten Zeit schließt. Dies wiederum deutet bei einem direkten Vergleich mit der $n100w60$-Instanz darauf hin, dass eine verhältnismäßig größere Zeitfensterweite (namentlich eine Zeitfensterweite von $|\overline{W}| = 100$ statt einer Zeitfensterweite von $|\overline{W}| = 60$) eine wichtigere Voraussetzung für ein effizientes Arbeiten der *BS2M*-Cuts darstellt, als eine im ähnlichen Maße vergrößerte Städteanzahl (namentlich eine Städteanzahl von $|\overline{V}| = 100$ statt einer Städteanzahl von $|\overline{V}| = 60$).

Um die prinzipielle Mächtigkeit der *TSPTW-TBF*, beziehungsweise Branch-And-Cut-Framework gestützten *TSPTW-TBR* zu verdeutlichen, sei abschließend ein kurzes Benchmarking eingeworfen (wobei pauschal die in Abschnitt 2 beschriebenen Preprocessing-routinen vorausgesetzt sind): Während die (Basis-) *TSPTW-TBF* beispielsweise auf der

n100w60.001 Instanz nach insgesamt 4 Knoten nachweislich mit dem gesuchten Optimum schließt, bewegt sich auf der gleichen Instanz im Fall einer *Big-M Formulation* die Anzahl benötigter Knoten im Millionenbereich (konkret wird der optimale Incumbent nach einhalbmillionen Knoten gefunden, dessen Belegung mittels der Dualen Schranke benötigt jedoch ein Vielfaches mehr an weiteren Branching-Verzweigungen). Ferner sieht sich diese Diskrepanz ceteris paribus sowohl durch ein Ansteigen der Städteanzahl als auch eine Vergrößerung der Zeitfensterweiten erheblich verstärkt, wie sondierende Experimente im Rahmen dieser Arbeit gezeigt haben.

Konklusion Es ist nicht auszuschließen, dass eine höhere Frequenz der Cut-Aufrufe (namentlich ein Aufrufen der Cut-Routinen in jedem, satt nur in jedem dritten B&C-Knoten), sowie eine Ergänzung der *Infeasible-Path-Constraint*-Separationsroutine um die im Rahmen von Gleichung (4.55) beschrieben Verstärkungs-Maßnahmen, die auf umfangreicheren Instanzen zu beobachtende Überlegenheit der *BS2M*-Cuts-Ergänzung ein Stück weit relativieren. Die beobachteten Resultate sind dennoch bezeichnend und legen die Vermutung nahe, dass die *BS2M*-Cuts auf Instanzen mit größeren Zeitfenstern und erhöhter Städteanzahl eine erhebliche Bereicherung für das B&C-Framework darstellen und somit in diesen Fällen integriert werden sollten.[10]

5.4.2. Auswirkungen der neuen Primalheuristik

In diesem Abschnitt werden zwei Versionen des *TSPTW-TBR*-B&C-Frameworks gegenübergestellt, wobei die erste Version in jedem B&C-Knoten die in Abbildung 4.13 beschriebene, "neue Primalheuristik" verwendet, während die zweite Version in jedem B&C-Knoten auf die in Abbildung 4.12 beschriebene, "alte Primalheuristik" zurückgreift.

Auf eine Aufführung der CPU-Zeiten ist verzichtet, da sich der zeitliche Mehraufwand, welcher für die erweiterte Primalheuristik zu verzeichnen ist, stets lediglich im Millisekunden-Bereich bewegt, und daher in Anbetracht der durchweg verhältnismäßig niedrigen B&C-Baum-Knotenzahlen vernachlässigt werden kann.

Ergebnisse In Tabelle 5.4.2 sind die Resultate des beschriebenen Vergleichs aufgeführt (wobei die Instanz-Namen in den entsprechenden Zeilen zu den Resultaten der neuen Primalheuristik mit dem Index N versehen sind).

Die ersten drei Spalten-Köpfe von links sowie der erste Spalten-Kopf von rechts sind identisch mit den entsprechenden Pendants in Tabelle 5.4.1. In der Spalte "Nds AVG" ist sodann die durchschnittliche Tourlänge (gemessen über die Städteanzahl) angegeben, welche die jeweilige Heuristik pro B&C-Knoten in der Lage war zu konstruieren. In der Spalte "Nds VAR" findet sich die entsprechende Varianz der Städteanzahl je Knoten, bezogen auf den besagten Durchschnittswert. Die folgenden drei Spalten beziehen sich nur auf jene Branch-And-Cut-Knoten, bis zu welchen die B&C-Frameworks der beiden Heuristiken jeweils einen identischen Verlauf aufweisen (konkret sind dies auf allen Instanzen außer $rgb031a$ sämtliche Knoten des entsprechenden B&C-Baumes): Spalte "HN>H" gibt im Rahmen dessen den prozentualen Anteil der B&C-Knoten an, in welchen die neue Primalheuristik eine längere Tour konstruieren konnte als die alte Primalheuristik. Dementsprechend berichten die Spalte "HN=H" über den prozentualen Anteil der Gleichstände und die Spalte "HN<H" über den prozentualen Anteil der B&C-Knoten, in welchen die konstruierte Tour der neuen Heuristik kürzer ausfiel, als die der alten Heuristik. Hierbei

[10] je nach Entwicklungsumgebung überdies in der Laufzeit-verbesserten Version der *Minimum-Cut-Tree*-Implementierung

ist in Klammern jeweils das durchschnittliche Ausmaß der resultierenden Städteanzahl-Diskrepanz pro B&C-Knoten angegeben. Die Spalte "1.Sl Cnstr" gibt an, ob der erste zulässige Incumbent von keiner der beiden Heuristiken konstruiert wurde (tie), von beiden Heuristiken (und somit zwangsläufig gleichzeitig) konstruiert wurde (tie), oder von der neuen Heuristik konstruiert wurde aber von der alten nicht (N).[11]

Tabelle 5.3.: Experimentalergebnisse zur neuen Primalheuristik

Instanz	$\lvert\overline{V}\rvert$	$\lvert\overline{W}\rvert$	**Nds Avg**	**Nds Var**	**rel Nds HN>H**	**rel Nds HN=H**	**rel Nds HN<H**	**1.Sl Cstr**	**Nds total**
n20w100	20	100	18.50	0.50	0.00%	100.00%	0.00%	tie	0
n20w100$_N$	20	100	18.50	0.50	(.)	(0.00)	(.)	tie	0
n40w100	40	100	27.25	68.25	72.22%	22.22%	5.56%	tie	46
n40w100$_N$	40	100	31.81	22.62	(6.42)	(0.00)	(1.50)	tie	46
n80w80	80	80	71.00	220.89	10.00%	90.00%	0.00%	tie	3
n80w80$_N$	80	80	75.20	51.96	(42.00)	(0.00)	(.)	tie	3
n100w60	100	60	79.55	838.87	20.00%	60.00%	20.00%	tie	4
n100w60$_N$	100	60	85.36	256.65	(33.00)	(0.00)	(1.00)	tie	4
rgb017	15	600	14.50	5.39	10.00%	90.00%	0.00%	tie	3
rgb017$_N$	15	600	14.70	2.90	(16.25)	(0.00)	(.)	tie	3
rgb031a	31	600	28.30	16.68	66.67%	33.33%	0.00%	N	14
rgb031a$_N$	31	600	30.38	14.55	(4.00)	(0.00)	(.)	N	3
rgb034a	34	606	25.49	42.95	38.16%	42.11%	19.74%	tie	84
rgb034a$_N$	34	606	27.68	49.53	(6.86)	(0.00)	(1.93)	tie	84
rgb041a	41	606	31.88	104.88	28.42%	55.48%	16.10%	tie	1581
rgb041a$_N$	41	606	36.18	51.21	(16.31)	(0.00)	(2.11)	tie	1581

Die Überlegenheit der neuen Primalheuristik ist offensichtlich: So erschließt sie auf sämtlichen *nicht-trivialen* Instanzen durchschnittlich deutlich mehr Städte je B&C-Baum-Knoten und dies, bis auf eine Ausnahme[12], durchweg mit einer zuverlässigeren Konstanz (namentlich einer geringeren Varianz). Der Grund für letztere Beobachtung ist klarerweise hauptsächlich in einer Vermeidung der "Ausfälle" zu sehen, welche im Fall der ursprünglichen Primalheuristik stets entstehen, wenn ein augenscheinlicher "Sackgassen-Knoten" angesteuert wird.

Zudem verzeichnet die neue Primalheuristik in den seltenen Fällen, in welchen sie auf konkreten B&C-Baum-Knoten unterliegt ($HN < H$) im Schnitt jeweils lediglich einen geringen, einstelligen Rückstand an erschlossenen Städten (konkret bewegt sich die besagte Durchschnittsgröße zwischen den Werten 1.00 und 2.11), während sie in den deutlich häufigeren Fällen einer Überlegenheit ($HN > N$) im Schnitt stets einen großen einstelligen, teilweise sogar einen zweistelligen Vorsprung aufzuweisen vermag (konkret bewegt sich die besagte Durchschnittsgröße zwischen den Werten 4.00 und 42.00).

Überdies ist die neue Primalheurisitik auf der $rgb031a$-Instanz in der Lage, an einem sehr führen Punkt des Branch-And-Cut-Algorithmus' eine zulässige, ganzzahlige Lösung zu konstruieren, welche von der ursprünglichen Primalheuristik nicht gefunden werden kann, womit sie in diesem Fall den gesamten Lösungsverlauf entschieden zu beeinflussen

[11] Die für die Angaben in dieser Tabelle benötigten Zusatzinformationen sind mittels der `GAMS-Put-Facility` aus dem Verlauf des Branch-And-Cut-Algorithmus' extrahiert

[12] namentlich der Instanz $rgb34a$

vermag. Konkret schließt das B&C-Framework, welches auf die erweiterte Primalheuristik zurückgreift, hier bereits nach 3 statt nach 14 Knoten.

Konklusion Jedoch ist für eine effektive Verbesserung des Lösungsverhaltens der B&C-Framework-gestützten *TSPTW-TBR* lediglich der letzte, in Abschnitt 5.4.2 aufgeführte Sachverhalt entscheidend. Daher weisen die Resultate insgesamt (zusammen mit dem geringen zeitliche Mehraufwand für die erweiterte Heuristik) darauf hin, dass ein Inbetrachtziehen noch ambitionierterer Primalheuristiken opportun sein könnte. Namentlich wäre ein Ergänzen des betrachteten Erweiterungsvorschlags um einen dosierten Einsatz von Backtracking[13] in Kombination mit einer Variante der Tabu-Suche[14] (wobei die Tabuliste beispielsweise aus Kanten $(i,j) \in A$ aufgebaut sein könnte, welche ab bestimmten Knotenbesuchs-Zeitpunkten t_i nicht mehr gewählt werden sollten, da selbiges in einer vergangenen Iteration sofort, oder einige Knoten später, zu einer unzulässigen Lösung geführt hat) denkbar. Überdies könnte eine, an den Ansatz von *Gendrau et al.* [18] angelehnte, *TSPTW*-spezifische Insertion-Technik in die Primalheuristik eingebettet werden.

5.4.3. Auswirkungen einer Bucket-spezifischen Branching-Strategie

Im folgenden sollen die Auswirkungen einer Bucket-spezifischen Branching-Regel auf das Lösungsverhalten der B&C-Framework-gestützten *TSPTW-TBR* untersucht werden.

Die für ein Bucket-orientiertes Branching grundsätzlich in Frage kommenden Entscheidungsvariablen sind augenscheinlich die z_i^b mit $i \in V, b \in W_i$ sowie die $y_{i,j}^b$ mit $i,j \in V, b \in W_i$. Potentielle Maxime einer Branchingentscheidung kann beispielsweise sein, ein möglichst großes Ausmaß an erzwungenen, eindeutigen Fixierungen der Variablen im betrachteten LP zu generieren, um auf diese Weise zwei deutlich von einander abgegrenzte B&C-Knoten-Söhne zu erhalten. Unter diesem Gesichtspunkt sind die $y_{i,j}^b$ vorzuziehen, da sich ihre Werte nicht aus denen anderer Entscheidungsvariablen zusammensetzen, sondern ihnen im Modell eine ausschließlich determinierende Rolle zukommt. So werden (wegen (2.16) und (2.18)) beispielsweise bei einem Branching auf der Variablen $y_{i_1,j_1}^{b_1}$ zugleich ebenfalls verbindliche (und insbesondere integrale) Entscheidungen für die Variable $z_{i_1}^{b_1}$ sowie die Variable x_{i_1,j_1} getroffen, während im Fall eines Branchings auf der Variablen $z_{i_1}^{b_1}$ der Solver einen großen Spielraum fraktionaler Gestaltung bezüglich der über (2.16) und (2.18) zuzuordnenden Variablen $x_{i,j}$ *s.t.* $(i,j) \in \delta^+(i_1)$ sowie $y_{i_1,j}^{b_1}$ *s.t.* $(b_1, i_1, j) \in \mu(b_1, b')$ *mit* $b' \in W_j$ behält.

Orientiert man sich ferner an dem Ziel, mittels des Branchings möglichst baldig eine für die zugrunde liegende TSPTW-Instanz zulässige, integrale Lösung zu finden, so scheint überdies, in Anlehnung an den Mechanismus der Primalheursistik, das Einführen von (mittels zeitlicher Puffer induzierten) Prioritäten innerhalb der $y_{i,j}^b$ sinnvoll. Dieser Idee folgend sollte somit jener (noch nicht gebranchten) Variablen $y_{i,j}^b$ die höchste Branchingpriorität zugestanden werden, welche die niedrigste Differenz $D_j - (r_b + \theta_{i,j})$ aufweist, namentlich also den geringsten verbleibenden zeitlichen Spielraum für die TSPTW-Tour impliziert.

Ergebnisse Die Auswirkungen der daraus resultierenden, Bucket-spezifischen Branching-Strategie sind in Tabelle 5.4.3 zusammengefasst (wobei die Instanz-Namen in den entsprechenden Zeilen mit dem Index B versehen sind), und den jeweiligen Resultaten ohne Bucket-spezifisches Branching (im Folgenden auch als "Default-Branching" bezeichnet)

[13] für eine Erläuterung zum generellen Begriff der Backtracking-Methode siehe beispielsweise [12]
[14] für eine Erläuterung zum generellen Begriff der Tabu-Suche siehe beispielsweise [21]

gegenüber gestellt. Hierbei ist auf eine Betrachtung der $n20w100$ Instanz verzichtet, da selbige bereits im Wurzelknoten optimal löst und somit keine Branching-Entscheidungen erfährt.

Sämtliche Spaltenköpfe sind identisch mit den entsprechenden Pendants in Tabelle 5.4.1, wobei die Rubrik "Cutted lpRB" ausgelassen ist. Da sich die Branching-Regel in ihrem Mechanimus an die Primalheuristik anlehnt, ist die Frage nach eventuellen Synergien zwischen diesen beiden B&C-Framework-Elementen interessant. Daher sind überdies in den Spalten "Nds 1.Int" sowie "Nds OPT" die eingetragenen Knoten-Anzahlen jeweils mit dem Hoch-Index "+" versehen, falls der entsprechende Incumbent durch die Primalheuristik ermittelt wurde.

Tabelle 5.4.: Experimentalergebnisse zu einer Bucket-spezifischen Branching-Strategie

Instanz	$\lvert\overline{V}\rvert$	$\lvert\overline{W}\rvert$	*OPT*	**Nds 1.Int**	**rGAP 1.Int**	**Nds OPT**	**Nds total**	**CPUt total**
n40w100	40	100	429.00	15	0.46%	41	46	8193
n40w100$_B$	40	100	429.00	16	0.23%	17	43	6162
n80w80	80	80	624.00	2	0.00%	2	3	5343
n80w80$_B$	80	80	624.00	8	0.16%	9	10	18513
n100w60	100	60	655.00	3^+	0.00%	3^+	4	13891
n100w60$_B$	100	60	655.00	2^+	0.30%	21	22	22415
rgb017	15	600	847.00	1	0.24%	2	3	86
rgb017$_B$	15	600	847.00	1	0.24%	2	3	96
rgb031a	31	600	1817.00	1^+	0.76%	6	14	698
rgb031a$_B$	31	600	1817.00	2	0.22%	5^+	11	577
rgb034a	34	606	2169.00	83^+	0.00%	83^+	84	15040
rgb034a$_B$	34	606	2169.00	3^+	0.78%	7	13	2475
rgb041a	41	606	2547.00	34	1.51%	1261	1581	23911
rgb041a$_B$	41	606	2547.00	25	0.08%	37	9990	24895

Auf sämtlichen *nicht-trivialen* Instanzen mit verhältnismäßig geringer Städteanzahl (namentlich mit $\lvert\overline{V}\rvert \leq 40$) kann, ungeachtet der Zeitfenstergröße, eine klare Performance-Verbesserung erzielt werden: So sind in den besagten Fällen durchweg weniger B&C-Knoten, und somit auch weniger CPU-Zeit, vonnöten, bis das Optimum gefunden und bestätigt ist. Hierbei werden sowohl der erste zulässige Incumbent, als auch der optimale Imcumbent, stets nach deutlich weniger Knoten als im Fall des Default-Branchings gefunden, was dafür spricht, dass der aggressive, problemspezifische Branching-Mechanismus seine zugrunde liegende Intention auf Instanzen dieser Größe erfolgreich umzusetzen vermag.

Auf den Instanzen mit höheren Städteanzahlen (namentlich $\lvert\overline{V}\rvert > 40$) werden jedoch ebenfalls zwei Schwachpunkte der betrachteten Branching-Regel deutlich: Zunächst ist auf der $rgba041a$ zwar ebenfalls zu beobachten, dass der erste zulässige Incumbent, sowie das gesuchte Optimum mittels des aggressiven, problemspezifischen Branchings deutlich früher gefunden werden, dies kann jedoch nicht in einen letztendlichen Zeitvorteil umgemünzt werden, da die Bestätigung des Optimalwertes ein dramatisches Vielfaches an weiteren B&C-Knoten benötigt. Konkret ist das Optimum nach 37 statt 1261 Knoten gefunden, der B&C-Algorithmus schließt jedoch erst nach 9990 statt nach 1581 Knoten. Dieser Performance-Unterschied dürfte der Tatsache geschuldet sein, dass die neue Branching-Strategie, im Gegensatz zum Default-Branching, die potentielle Verbesserung

der Dualen Schranken entlang des B&C-Baumes in ihrer Entscheidungsfindung gänzlich vernachlässigt. Die Güte der Dual Bounds im B&C-Baum ist im Fall von großen Zeitfensterweiten und erhöhten Städteanzahlen jedoch ein entscheidender Faktor für ein erfolgreiches Arbeiten des B&C-Frameworks, wie bereits im Rahmen von anderen Experimenten dieses Abschnittes 5 zu beobachten war. Auf Instanzen mit kleineren Zeitfenstern aber einer großen Städteanzahl (namentlich die Instanzen $n80w80$ und $n100w60$) ist indes zu beobachten, dass die neue Branching-Strategie (im völligen Gegensatz zu ihrer Performance auf den restlichen Instanzen) für das Auffinden des ersten zulässigen Incumbents sowie des gesuchte Optimums, jeweils deutlich mehr B&C-Knoten benötigt, als das Default-Branching. Dies kann darauf zurückzuführen sein, dass die Anzahl der Bucket-Graph-Kanten und somit die Anzahl der $y^b_{i,j}$-Variablen, in der Anzahl der Städte tendenziell quadratisch wächst, womit das vereinzelte Branchen auf ausgewählten $y^b_{i,j}$-Variablen ab Größenordnungen von $|\overline{V}| \geq 80$, im Sinne der ursprünglichen Intention, keinen hinreichend effektiven Einschlag mehr auf das Lösungsverhalten, beziehungsweise das zügige Konstruieren zulässiger Touren, zu erzielen vermag.

Konklusion Da auf Instanzen mit verhältnismäßig geringer Städteanzahl klare Performance-Verbesserungen erzielt werden können, auf größeren Instanzen indes jedoch sowohl die Beschränkung auf die $y^b_{i,j}$-Variablen als auch das außer Acht lassen der Dualen Schranken zu einem klaren Nachteil für das Lösungsverhalten gereichen, wäre beispielsweise eine Hybrid-Branching-Strategie denkbar, welche ebenfalls die z^b_i-Variablen (die ihres Zeichens ein lineares Wachstum in der Städteanzahl aufweisen) mit einbezieht und überdies auf eine komplexere Bewertungsfunktion bezüglich der Branching-Prioritäten zurück greift, welche ebenfalls "Augenmerk" auf die potentielle Verbesserung der Dualen Schranken legt.

5.4.4. Auswirkungen einer modulierten Preprocessing-Routine

Im Folgenden seien die Auswirkungen eines gezielten Auslassens der Zeitfensterstraffungs-Regeln im Rahmen des allgemeinen Preprocessings, sowohl auf die Bildung der *TSPTW-TBR* als auch auf das Lösungsverhalten des *TSPTW-TBR* -Branch-And-Cut-Frameworks, untersucht.

Ergebnisse Die Tabelle in Abbildung 5.4.4 zeigt zunächst die Auswirkungen der verschiedenen Preprocessing-Routinen auf die Instanz-Eigenschaften sowie diverse Attribute der resultierenden *TSPTW-TBR*.

Die ersten drei Spaltenköpfe sind identisch mit den entsprechenden Pendants in Tabelle 5.4.1. Überdies wird in der Spalte "$|\pi|$" angegeben, wie viele (nicht redundante) Präzedenzen auf der Instanz ermittelt werden konnten. Spalte "Ral $|A|$ Del" beziffert sodann den prozentualen Anteil der ursprünglichen Kanten, welche im Rahmen des Preprocessings gelöscht wurden.

Ist die *TSPTW-TBR* generiert[15], so lässt sich eine Bucket-induzierte Zeitfenstergröße $W^{bkt}_i = [R^{bkt}_i, D^{bkt}_i]$ des Knotens i bestimmen, wobei R^{bkt}_i die *Releasetime* der ersten Bucket von i und D^{bkt}_i die *Deadline* der letzten Bucket von i darstellen. Der Durchschnitt dieser Zeitfenstergrößen, gebildet über alle Knoten $i \in V \setminus \{p, q\}$, ist in der Spalte "$|\overline{W}|$ *Rmg*" wiedergegeben. Spalte "$|\pi_b \cup \sigma_b|$" gibt Auskunft über die addierte Anzahl der Bucket-Knoten- und Knoten-Bucket-Präzendenzen, welche auf der entsprechenden

[15] wobei im Rahmen der Bucket-Bildung, im Fall der ausgelassenen Zeitfensterstraffung, die Forderungen $r_{b^i_1} = R_i$, $d_{b^i_L} = D_i$ ignoriert sind, gegeben am Anfang und/oder am Ende des entsprechenden Zeitfensters finden sich Löcher

TSPTW-TBR -Instanz ermittelt werden konnten. "$|A_{\mathfrak{B}}|$ Rmg" gibt die Anzahl der nach finaler Bucket-Graph-Reduktion verbleibenden Bucket-Graph-Kanten an. "$|A_{\mathfrak{B}}|$ Init" Beziffert indes die ursprüngliche Menge an Bucket-Graph-Kanten zu Beginn der letzte Iteration im Rahmen der iterativen Bucket-Refinement-Heuristik. "lpRB" gibt schließlich den Zielfunktionswert der optimalen LP-Relaxations-Lösung der in Rede stehenden *TSPTW-TBR* (ohne Cuts) wieder.

Tabelle 5.5.: Vergleichszahlen zum Preprocessing mit und ohne Zeitfensterstraffung

Instanz	$\lvert\overline{V}\rvert$	$\lvert\overline{W}\rvert$	$\lvert\pi\rvert$	**Rel $\lvert A\rvert$ Deltd**	$\lvert\overline{W}\rvert$ **Rmg**	$\lvert\pi_b \cup \sigma_b\rvert$	$\lvert A_{\mathfrak{B}}\rvert$ **Rmg**	$\lvert A_{\mathfrak{B}}\rvert$ **Init**	**Init lpRB**
n20w100	20	100	163	58.47%	89.45	11076	5242	6630	232.00
n20w100$_R$	20	100	155	57.85%	96.50	11626	5375	7155	230.53
n40w100	40	100	603	61.79%	97.10	83034	26570	37040	370.11
n40w100$_R$	40	100	600	61.50%	99.70	85265	27760	38438	368.78
n80w80	80	80	2777	79.67%	72.20	289768	29498	59743	597.14
n80w80$_R$	80	80	2772	79.61%	74.18	293108	30176	60542	597.13
n100w60	100	60	4438	81.11%	57.98	413290	49173	79991	625.98
n100w60$_R$	100	60	4435	81.02%	59.83	420676	50642	81403	625.72
rgb017	15	600	98	57.79%	599.47	5000	2739	2939	842.75
rgb017$_R$	15	600	98	57.79%	599.47	5000	2739	2939	842.75
rgb031a	31	600	400	63.91%	600.00	37738	11072	13343	1775.69
rgb031a$_R$	31	600	400	63.91%	600.00	37738	11072	13343	1775.69
rgb034a	34	606	456	58.56%	602.77	79700	31185	37194	2164.46
rgb034a$_R$	34	606	456	58.56%	603.65	78938	30920	36954	2164.00
rgb041a	41	606	698	65.71%	600.78	106689	28423	35278	2533.68
rgb041a$_R$	41	606	698	65.71%	600.81	106689	28423	35278	2533.68

Generell ist zunächst, insbesondere auf den *Dash et al.* Instanzen, welche höhere Städtezahlen und zugleich geringere Zeitfensterweiten aufweisen, die große Bedeutung des Preprocessings für die Modellvereinfachung zu erkennen. So werden beispielsweise auf der n100w60 Instanz im Rahmen des unberührten Preprocessings über 81.11% der Kanten gelöscht und die durchschnittliche Zeitfensterweite kann von 60 auf 57.98 reduziert werden, was hochgerechnet einer Löschung von insgesamt $2.02 * 100 = 202$ Zeitslots entspricht. Die Resultate bezüglich der gelöschten Kanten auf den Isntanzen n80w80 und n40w100 Instanzen gestalten sich ähnlich bezeichnend, wobei in dieser Hinsicht eine tendenzielle Abnahme der Preprocessing-Effektivität mit steigender Zeitfensterweite zu beobachten ist. Diese Tendenz bestätigt sich auf den *Dash et al.*-Instanzen, welche durchweg Zeitfensterweiten von 600 oder größer verzeichnen.

Ebenfalls die Auswirkungen der ausgelassenen Zeitfensterstraffung auf die Eigenschaften der resultierenden *TSPTW-TBR* hinsichtlich der Unterschiede in Bucketgraph-Größe (ausgelassene Zeitfensterstraffung führt zu tendenziell größeren Bucket-Graphen), Bestimmbarkeit von Präzedenzen (dass im Fall ohne Zeitfensterstraffung absolut betrachtet in der Regel mehr Präzedenzen ermittelt werden, liegt an den größeren, jeweils zugrunde liegenden Bucktegraphen) und Güte der Dualen Schranken im Vergleich zu denen der *TSPTW-TBR*, welche aus dem unberührten Preprocessing hervorgeht, sind im Fall der *Dumas et al.*-Instanzen (namentlich im Fall verhältnismäßig kleiner Zeitfenster) erheblicher als im Fall der *Ascheuer*-Instanzen. Auf den ersten Blick mag es paradox erscheinen,

dass bei Instanzen größerer Zeitfensterweiten die Zeitfensterstraffung scheinbar eine geringere Rolle für die Bildung der *TSPTW-TBR* spielt. Dies liegt jedoch darin begründet, dass gerade aufgrund der größeren Zeitfensterausmaße weniger direkte (Knoten-Knoten-)Präzedenzen ermittelt, und daher bei den *Releasetime*- und *Deadline*-Verschiebungen eine tendenziell geringere Anzahl an indizierenden Kanten ausgeschlossen werden können, womit die Effektivität der Zeitfensterstraffungs-Regeln in diesen Fällen erheblich geschwächt ist. Die geringen Nuancen, um welche die Zeitfenster hier eingedämmt werden, können jedoch interessanter Weise für den Verlauf des *TSPTW-TBR* -Branch-And-Cut Frameworks unter gewissen Umständen eine erhebliche Rolle spielen, wie sich im Folgenden zeigen wird.

Tabelle 5.6.: Experimentalergebnisse zu verschiedenen Preprocessing-Routinen

Instanz	$\lvert\overline{V}\rvert$	$\lvert\overline{W}\rvert$	*OPT*	**Nds 1.Int**	**rGAP 1.Int**	**Nds OPT**	**Nds total**	**CPUt total**
n20w100	20	100	237.00	0	0.00%	0	0	50
n20w100$_R$	20	100	237.00	0	0.00%	0	0	47.24
n40w100	40	100	429.00	15	0.46%	41	46	8193
n40w100$_R$	40	100	429.00	55	7.54%	164	174	35518
n80w80	80	80	624.00	2	0.00%	2	3	5343
n80w80$_R$	80	80	624.00	2	0.00%	2	3	5357
n100w60	100	60	655.00	3	0.00%	3	4	13891
n100w60$_R$	100	60	655.00	6	0.00%	6	17	22675
rgb017	15	600	847.00	1	0.24%	2	3	86
rgb017$_R$	15	600	847.00	1	0.24%	2	3	89
rgb031a	31	600	1817.00	1	0.76%	6	14	698
rgb031a$_R$	31	600	1817.00	1	0.76%	6	14	712
rgb034a	34	606	2169.00	83	0.00%	83	84	15040
rgb034a$_R$	34	606	2169.00	0	0.00%	0	0	400
rgb041a	41	606	2547.00	34	1.51%	1261	1581	23911
rgb041a$_R$	41	606	2547.00	34	1.51%	1261	1581	23969

Die Tatsache, dass auch im Fall ohne Zeitfensterstraffung die meisten Bucket-induzierten Zeitfenstergrößen im Durchschnitt von der jeweils ursprünglichen durchschnittlichen Zeitfenstergröße abweichen, ist auf den Effekt zurückzuführen, dass bei ausbleibenden Zeitfensterstraffungs-Regeln die initiale Bucket-Bildung eventuelle Löcher an den Rändern der Zeitfenster eliminiert und somit einen Teil des Zeitfensterstraffungs-Mechanismus' abdeckt.

Pauschal ist, insbesondere im Fall der *Dumas et al.*-Instanzen[16] und ungeachtet des Kern-Experimentes, bemerkenswert, wie viele zusätzliche Bucket-Graph-Kanten jeweils mittels des Bucket-spezifischen Preprocessings eliminiert werden können. Hierbei sei bedacht, dass die Größe "$|A_{\mathfrak{B}}|$ Init" bereits sämtliche Kantenlöschungseffekte enthält, welche im Rahmen des allgemeine Preprocesssing erzielt werden konnten.

Die Tabelle 5.4.4 gibt sodann die Auswirkungen der modulierten Preprocessing-Routine auf des Lösungsverhalten der resultierenden B&C-Framework-gestützten *TSPTW-TBR*, jeweils gegenüber gestellt dem Lösungsverhalten der B&C-Framework-gestützten *TSPTW-TBR*, welche aus dem unberührten Preprocessing hervorgeht, wieder. Sämtliche Spal-

[16] was einmal mehr auf die verhältnismäßig kleinen Zeitfenster zurückzuführen sein könnte

tenköpfe sind identisch mit den entsprechenden Pendants in Tabelle 5.4.3.

Die tendenziell negativen Auswirkungen der ausgelassenen Zeitfensterstraffung auf das *TSPTW-TBR* -B&C-Framework-Lösungsverhalten sind im Fall der *Dumas et al.*-Instanzen offensichtlich: So schließt die $n40w100$-Instanz nach 174 statt 46 Knoten in einer entsprechend deutlich schlechteren CPU-Zeit. Im Rahmen dessen wird der erste Incumbent nach 55 statt nach 15 Knoten gefunden und weist zudem ein deutlich schlechteres, reales Optimalitäts-GAP auf (namentlich 7.54% statt 0.46%). Ebenfalls das gesuchte Optimum findet sich erst deutlich später als im Fall der Zeitfensterstraffungs-gestärkten *TSPTW-TBR* (namentlich nach 164 statt nach 41 Knoten). Im Fall der $n100w60$-Instanz gestalten sich die negativen Auswirkungen auf das Lösungsverhalten ähnlich prägnant.

Auf den *Dash et al.* -Instanzen hingegen sind etwaige Auswirkungen, korrespondierend zu den geringeren Diskrepanzen beim Preprocessing, kaum ausgeprägt. Tatsächlich ist die einzige nennenswerte Abweichung sogar eine positive. So schließt die $rgb034a$-Instanz ohne gestraffte Zeitfenster bereits im Wurzelknoten statt nach 84 Knoten. Es sei allerdings angemerkt, dass der frühe, optimale Incumbent, welcher diesen drastischen Performanceunterschied ermöglicht, von der Primalheuristik ermittelt wird. Zudem zeigen die Resultate aus Tabelle 5.4.4, dass $rgb034a$ die einzige Instanz ist, welche (nach Bucket-Preprocessing) im Fall von ausgelassener Zeitfensterstraffung einen kleineren (und somit tendenziell "besseren") Bucket-Graphen aufweist, als im Fall mit vorangegangener Zeitfensterstraffung[17]. Daher kann dieses Resultat als "glücklicher Zufall" und somit als "Ausreißer" betrachtet werden.

Generell suggeriert die Durchwachsenheit der Ergebnis-Unterschiede über das gesamte Spektrum der Instanzen sehr deutlich die extreme Volatilität, welche die B&C-Framework-gestützten *TSPTW-TBR* im Rahmen ihres Lösungsverhaltens gegenüber den vorbereitenden Maßnahmen an den Tag legt.

Konklusion Zusammenfassend lässt sich festhalten, dass die B&C-Framework-gestützten *TSPTW-TBR*, je nach Instanz-Eigenschaften, sowohl in ihrer Generierung als auch ihrem Lösungsverhalten im B&C-Framework extrem neuralgisch auf Modulationen im Preprocessing reagieren kann. Dies legt nahe, dass die Anwendung eines ambitionierten Preprocessings (mit gegebenenfalls mehr Elementen als den in dieser Arbeit vorgestellten) so sinnvoll wie wichtig ist, um ein möglichst großes Ausmaß dieser potentiellen Volatilität von vornherein einzudämmen. Forciert wird diese Erkenntnis durch die Tatsache, dass das in dieser Abreit vorgestellte Preprocsseing bezüglich seiner benötigten Laufzeit (auf *nicht-trivialen* Instanzen) nur einen vernachlässigbar kleinen Bruchteil (in der Regel liegt die für das Preprocessing aufgewendete CPU-Zeit im zweistelligen Sekundenbereich) der insgesamt benötigten Lösungs-Zeit darstellt.

[17] dass diese kontraintuitive Tatsache überhaupt möglich ist, könnte auf das komplexe Zusammenspiel von *Bucket-Refinement-Heuristik*, *BTI-Cleaning* und *Bucket-Preprocessing* im Rahmen der iterativen *TSPTW-TBR* -Bildung zurückzuführen sein

6. Zusammenfassung und kritische Würdigung

In dieser Arbeit wurden das *TSPTW*, sowie drei verschiedene ILP-Formulierungen für selbiges, vorgestellt und untersucht. Insbesondere wurde hierbei auf die *TSPTW-TBF*, beziehungsweise die Branch-And-Cut-Framework unterstützte *TSPTW-TBR* eingegangen. Im Rahmen dessen wurden sowohl die von *Dash et al.* vorgestellten Elemente des besagten Lösungsansatzes präsentiert, als auch auf denkbare Erweiterungen, beziehungsweise Modulationen, eingegangen. Im Rahmen des Vorschlags einer möglichen Ergänzung der Separationsroutinen um sogenannte *Bucket Strengthenend 2-Matching Inequalitites* wiesen die Resultate der entsprechenden Experimente auf ein erhebliches Mehrwertpotential dieser Cuts hin. Ebenfalls im Rahmen der Betrachtung einer erweiterten Primalheuristik deuten die experimentellen Resultate an, dass in dieser Hinsicht ein ambitionierterer Ansatz, als der von *Dash et al.* vorgestellte, opportun ist. Ein Vorschlag für eine Bucket-spezifische Branching-Strategie konnte sich hingegen im Rahmen der Experimente nicht als vorteilhaft durchsetzen, da er unter anderem zu wenig "Augenmerk" auf die Verbesserung der Dualen Schranken entlang des B&C-Baumes legt und somit auf komplexeren Instanzen dem Default-Branching deutlich unterlegen ist. Experimente im Rahmen der Modulation von Preprocessingroutinen wiederum belegten die große Relevanz, welche dem Preprocessing im Rahmen der B&C-Framework-unterstützten *TSPTW-TBR* zukommt und deuteten darauf hin, dass hier ebenfalls die Anwendung ambitionierterer, wenn auch zeitaufwändigerer Methoden, opportun sein könnte.

Sowohl die Resultate der Experimente mit einem (gegenüber dem Original von *Dash et al.* [5] leicht modulierten) *TSPTW-TBR*-B&C-Framework, inklusive des eingeworfenen Benchmarkings bezogen auf die *TSPTW-BMF*, sowie die Ergebnisse von *Dash et al.* [5] selbst, unterstreichen die Mächtigkeit des Time Bucket-Ansatzes. Während er in der Präparationsphase (namentlich der Phase der *iterativen Bucket-Refinement-Heurstisk*), je nach Instanz, im Vergleich zu anderen (I)LP-basierten Ansätzen als relativ zeitkonsumierend einzustufen ist (insbesondere in Entwicklungsumgebungen wie `GAMS`), so ist seine Überlegenheit im Rahmen des anschließenden Branch-And-Cut-Lösungsverlaufs, sowohl im Vergleich zur *TSPTW-BMF* [1], als auch im Vergleich zu Ansätzen wie der, auf den Resultaten von *Ascheuer* [25] aufbauenden, B&C-Framework-ergänzten *TSPTW-TIF* von *Ascheuer et al.* [13][2] evident. Zwar ist im Rahmen dessen die Bearbeitung der einzelnen B&C-Knoten, insbesondere aufgrund der diversen, ambitionierten Separationsroutinen (einmal mehr besonders in Entwicklungsumgebungen wie `GAMS`) welche stets aufgerufen werden, beispielsweise im Vergleich zu einem nicht-Cut-unterstützten Ansatz der *TSPTW-BMF* , als verhältnismäßig zeitintensiv einzustufen, die starken Dualen Schranken sowie das generell günstige Verhalten der *TSPTW-TBR* -LP-Relaxationen im B&C-Baum können diesen Tatbestand jedoch (insbesondere auf "größeren" Instanzen) weit überkompensieren. Die Überlegenheit gegenüber der B&C-Framework-ergänzten *TSPTW-*

[1] wie sondierende Experimente im Rahmen dieser Arbeit gezeigt haben. Vgl. hierzu auch den Einwurf in Abschnitt 5.4.1

[2] vgl. hierzu die Ergebnisse von *Dash et al.* [5]

TIF von *Ascheuer et al.* [13] indes, stellt sich primär aufgrund einer vergleichsweise deutlich kompakteren Struktur der *TSPTW-TBF*, sowie der Anwendbarkeit nachweislich dominierender Bucket-spezifischer Cuts, ein.[3]

Überdies bietet die grundlegende Konstruktion des Bucket-Ansatzes von *Dash et al.* [5] viele Angriffspunkte für effiziente Erweiterungen und Verfeinerungen, wie es bereits im Rahmen dieser Arbeit, bei der Konstruktion neuer Bucket-spezifischer Cuts sowie einer ambitionierteren Primalheuristik, ausgenutzt wurde. Über die vorgestellten Extensionen hinaus ist, dem Ausblick von *Dash et al.* [5] folgend, zur Umgehung der Trade-off-Problematik "Güte der LP-Relaxation vs. LP-Größe (beziehungsweise Bucket-Anzahl)", sowie der Problematik des Erstellens gänzlich "überflüssiger" Buckets, eine Erweiterung des Ansatzes um Branch-And-Price-Techniken denkbar. Im Rahmen dessen könnte initiierend mit einem geringen Set an Buckets (und somit wenigen, Bucket-spezifischen Variablen) begonnen, und die zusätzlich benötigten Variablen mittels eines in das B&C-Framework eingebetteten Pricings, gezielt im Verlauf der Lösungsfindung hinzugefügt werden. Dies würde unter Umständen allerdings ebenfalls ein komplexes, dynamisches Anpassen der Bucket-spezifischen Schnitte im Cut-Pool an die jeweils neue Bucket-Struktur erfordern.

Zusammenfassend lässt sich festhalten, dass die (Preprocessing-unterstützte) *Time Bucket Formulation* von *Dash et al.* [5] zu den effektivsten und mächtigsten Ansätzen gezählt werden kann, welche zur Zeit in der Literatur bezüglich exakter Lösungsverfahren für das *TSPTW* zu finden sind[4], und überdies viele interessante Ansatzpunkte für verbessernde, beziehungsweise erweiternde Maßnahmen bietet.

[3] vgl. hierzu auch die Argumentationen von *Dash et al.* [5]

[4] vgl. hierzu unter anderem die in [54] aufgeführten Arbeiten, als auch die eigenen Aussagen von *Dash et al.* [5]

A. Literaturverzeichnis

Literaturverzeichnis

[1] M. López-Ibáñez, C. Blum. Beam-ACO for the travelling salesman problem with time windows. *Computers & Operations Research* **37**, S.1570-1583, 2010.

[2] R. F. da Silva, S. Urrutia. A General VNS heuristic for the traveling salesman problem with time windows. *Discrete Optimization* **7**, S.203-211, 2010.

[3] R.F. da Silva, S. Urrutia. The Traveling Salesman Problem with Time Windows (TSPTW) - Approaches & Additional Resources. http://homepages.dcc.ufmg.br/~rfsilva/tsptw/, 2010.

[4] C. Viergutz. Complex Scheduling Problems. http://www.informatik.uni-osnabrueck.de/kombalg/lehre/csp09/\csp09_folien_09.pdf, 2009.

[5] S. Dash, O. Günlük, A. Lodi, A. Tramontani. A Time Bucket Formulation for the TSP with Time Windows. http://researcher.watson.ibm.com/researcher/files/us-sanjeebd/\DGLT-tsptw.pdf, 2009.

[6] J. Albiach, J.M. Sanchis, D. Soler. An asymmetric TSP with time windows and with time-dependent travel times and costs: An exact solution through a graph transformation. *European Journal of Operational Research* **189**, S.789-802, 2008.

[7] A. Ramamoorthy, H. Sahota. EE 520: Topics in Communications: Network Coding, Lecture 12. http://www.ece.iastate.edu/~adityar/Teaching/EE520/lec12_notes.pdf, 2007.

[8] J.W. Ohlmann, B.W. Thomas. A compressed-annealing heuristic for the Travelling salesman problem with time windows. *INFORMS Journal on Computing* **19**, S.80-90, 2007.

[9] A. Lodi. Lecture 3: The Travelling Salesman Problem: Inequalities and Separation. http://www.dmf.unicatt.it/iniziative/scuola_geom/2004/Note/Lodi/Lodi_Lecture_3.pdf, 2004.

[10] J.-F. Cordeau, G. Desaulniers, J. Desrosiers, M.M. Solomon, F. Soumis. VRP with Time Windows. *In P. Toth, D. Vigo, eds. The Vehicle Routing Problem. SIAM Monographs on Discrete Mathematics and Applications*, S.157-194, 2002.

[11] F. Focacci, A. Lodi, M. Milano. A hybrid exact algorithm for the TSPTW. *INFORMS Journal on Computing* **14**, S.403-417, 2002.

[12] R. Sedgewick. Algorithmen. *Addison-Wesley, München*, 2002.

[13] N. Ascheuer, M. Fischetti, M. Grötschel. Solving the asymmetric travelling salesman problem with time windows by branch-and-cut. *Mathematical Programming, Ser.* **A 90**, S.475-506, 2001.

[14] V.V. Vazirani. Approximation Algorithms. *Springer, Berlin*, 2001.

[15] N. Ascheuer, M. Fischetti, M. Grötschel. A polyhedral study of the asymmetric travelling salesman problem with time windows. *Networks* **36**, S.69-79, 2000.

[16] R. W. Calvo. A new heuristic for the traveling salesman problem with time windows. *Transportation Science* **34**, S.113-124, 2000.

[17] G. Pesant, M. Gendreau, J.-Y. Potvin, J.M. Rousseau. On the flexibility of constraint programming models: from single to multiple time windows for the travelling salesman problem. *European Journal of Operational Research* **117**, S.253-263, 1999.

[18] M. Gendreau, A. Hertz, G. Laporte, M. Stan. A generalized insertion heuristic for the Travelling salesman problem with time windows. *Operations Research* **46** , S.330-335, 1998.

[19] G. Pesant, M. Gendreau, J.-Y. Potvin, J.-M. Rousseau. An exact constraint logic programming algorithm for the Travelling salesman problem with time windows. *Transportation Science* **32**, S.12-29, 1998.

[20] G.B. Dantzig, M.N. Thapar. Linear Programming 1:Introduction. *Springer, Berlin*, 1997.

[21] F. Glover, M. Laguna. Tabu Search. *Kluwer Academic Publishers*, 1997.

[22] A. Mingozzi, L. Bianco, S. Ricciardelli. Dynamic programming strategies for the travelling salesman problem with time windows and precedence constraints. *Operations Research* **45**, S.365-377, 1997.

[23] W.B. Carlton, J.W. Barnes. Solving the travelling salesman problem with time windows using tabu search. *IEE Transactions* **28**, S.617-629, 1996.

[24] L. Fleischer, 'E. Tardos. Separating Maximally Violated Comb Inequalities in Planar Graphs. *Technical Report No. 1150, School of Operations Research and Industrial Engineering, College of Engineering, Cornell University Ithaca, NY*, 1996.

[25] N. Ascheuer. Hamiltonian Path Problems in the On–line Optimization of Flexible Manufacturing Systems. *PhD thesis, Technische Universität Berlin, Berlin, Germany*, 1995.

[26] E. Balas, M. Fischetti, W.R. Pulleyblank. The precedence-constrained asymmetric Travelling salesman polytope. *Mathematical Programming, Ser.* **A 68**, S.241-265, 1995.

[27] J. Desrosiers, Y. Dumas, M.M. Solomon, F. Soumis. Time constrained routing and scheduling. *In M.O. Ball, T.L. Magnanti, C.L. Monma, G.L. Nemhauser, eds. Network Routing. Elsevier, Amsterdam, The Netherlands*, S.35-139, 1995.

[28] Y. Dumas, J. Desrosiers, E. Gelinas, M.M. Solomon. An optimal algorithm for the travelling salesman problem with time windows. *Operations Research* **43**, S.367-371, 1995.

[29] A. Conrad, T. Hindrichs, H. Morsy, I. Wegener. Solution of the Knight's Hamiltonian Path Problem on Chessboards. *Discrete Applied Mathematics, Volume* **50**, S.125-134, 1994.

[30] H. Nagamochi, T. Ono, T. Ibaraki. Implementing an efficient minimum capacity cut algorithm. *Mathematical Programming, vol.* **67**, S.325-341, 1994.

[31] A. Langevin, M. Desrochers, J. Desrosiers, S. Gélinas, F. Soumis. A two-commodity flow formulation for the Travelling salesman and makespan problems with time windows. *Networks* **23**, S.631-640, 1993.

[32] E. Balas, M. Fischetti, W. Pulleyblank. The precedence constrained asymmetric traveling salesman polytope. *Technical Report 15213, Carnegie Mellon University, Pittsburgh*, 1992.

[33] Y. Caseau, P. Koppstein. A rule-based approach to a time-constrained traveling salesman problem. *Proceedings of the 2nd International Symposium of Artificial Intelligence and Mathematics, Fort Lauderdale, FL.*, 1992.

[34] M. Gendreau, A.Hertz, G. Laporte. New insertion and postoptimization procedures for the travelling salesman problem. *Operations Research* **40**, S.1086-1094, 1992.

[35] J. Tsitsiklis. Special cases of the Travelling salesman and repairman problems with time windows. *Networks* **22**, 1992.

[36] M. Fischetti. Facets of the asymmetric traveling salesman polytope. *Mathematics of Operations Research*, **16**, S.42–56, 1991.

[37] M. Kubo, H. Kasugai. The precedence constrained travelling salesman problem. *Journal of the Operations Research Society of Japan* **34**, 1991.

[38] M. Padberg and G. Rinaldi. Facet identification for the symmetric Travelling salesman polytope. *Mathematical Programming*, **47**, S.219–257, 1990.

[39] J. Desrosiers, M. Sauve, E. Soumis. Lagrangian relaxation methods for solving the minimum fleet size multiple Travelling salesman problem with time windows. *Management Science* **34**, 1988.

[40] C.F. Daganzo, Modelling distribution problems with time windows, Parts I and II. *Transportation Science* **21**, 1987.

[41] M.W.P. Savelsbergh. Local search in routing problems with time windows. *Annals of Operations Research* **4**, S.285-305, 1985.

[42] N. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica, Nr.* **4**, S.373–395, 1984.

[43] E. Baker. An exact algorithm for the time constrained Travelling salesman problem. *Operations Research* **31**, 1983.

[44] M. Padberg, M.R. Rao. Odd minimum cut-sets and b-matchings. *Mathematics of Operations Research* **7**, S.67-80, 1982.

[45] N. Christofides, A. Mingozzi, P. Toth. State space relaxation procedures for the computation of bounds to routing problems. *Networks* **11**, S.145-164, 1981.

[46] M. Grötschel, M.W. Padberg. On the symmetric travelling salesman problem I: inequalities. *Mathematical Programming*, **16**, S.265-280, 1979.

[47] M.R. Garey, D.S. Johnson. Computers and Intractability : A Guide to the Theory of NP-Completeness. *Freeman, San Francisco*, 1979.

[48] J. Edmonds, R.M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the Association for Computing Machinery* **19**, S.248–264, 1972.

[49] S.A. Cook. The Complexity of Theorem Proving Procedures. *Annual ACM Symposium on Theory of Computing*, S.151-158, 1971.

[50] J. Edmonds. Maximum matching and a polyedron with 0,1-vertices. *Journal of Research of the National Bureau of Standards B69B*, S.125–130, 1965.

[51] P. Elias, A. Feinstein, C.E. Shannon. Note on maximum flow through a network. *IRE Transactions on Information Theory IT-* **2**, S.117—119, 1956.

[52] I. Heller, C.B. Tompkins. An Extension of a Theorem of Dantzig. In: H.W. Kuhn, A.W. Tucker. (eds.) Linear Inequalities and Related Systems. *Princeton University Press, Princeton, New Jersey*, S.247–254, 1956.

[53] Wikipedia. Travelling salesman problem. http://en.wikipedia.org/wiki/Travelling_salesman_problem, Stand: Juni, 2013.

[54] M. López-Ibáñez, C. Blum: Benchmark Instances for the Travelling Salesman Problem with Time Windows (TSPTW). http://iridia.ulb.ac.be/~manuel/tsptw-instances, Stand: Juni, 2013.

[55] Wikipedia. GAMS. http://de.wikipedia.org/wiki/General_Algebraic_Modeling_System, Stand: Juni, 2013.

[56] Concorde TSP Solver. http://www.tsp.gatech.edu/concorde.html, Stand: Juni, 2013.

[57] Microsoft (Office)-Homepage. http://office.microsoft.com/de-de, Stand: Juni, 2013.

[58] GAMS-Homepage. http://www.gams.com/, Stand: Juni, 2013.

[59] Latex-Homepage. http://www.latex-projekt.org/, Stand: Juni, 2013.

[60] Microsoft-Homepage. http://www.microsoft.com/, Stand: Juni, 2013.

[61] CPLEX/GAMS. http://www.gams.com/dd/docs/solvers/cplex.pdf, Stand: Juni, 2013.

[62] AMPL-Homepage. http://www.ampl.com/, Stand: Juni, 2013.

[63] SCIP-Homepage. http://www.SCIP.zib.de/, Stand: Juni, 2013.

[64] C. Kingsford. CMSC 451: SAT, Coloring, Hamiltonian Cycle, TSP. http://www.cs.cmu.edu/~ckingsf/bioinfo-lectures/sat.pdf, Stand: Juni, 2013.

[65] J. Aráoz, E. Fernández, O. Meza. A simple exact separation algorithm for 2-matching inequalities. http://www-eio.upc.es/~elena/Reports/DR200713.pdf, [o. Jahr].

[66] R. Bosch, M. Trick. Integer Programming. http://www.inf.ufpr.br/aurora/disciplinas/topicosia2/livros/search/integer.pdf, [o. Jahr].

[67] A. Conrad. Knight's Tour. http://www.axel-conrad.de/springer/springer.html#download, [o. Jahr].

[68] M. Gendreau, G. Laporte, M.M. Solomon. Single-vehicle routing and scheduling to minimize the number of missed deadlines. *Transportation Science*, forthcoming [o. Jahr].

B. Erklärung

Eidesstattliche Erklärung

Ich versichere hiermit, dass die vorliegende Masterarbeit selbstständig verfasst und keine weiteren als die angegebenen Hilfsmittel benutzt sowie die Stellen der Arbeit, die in anderen Werken dem Wortlaut oder dem Sinn nach entnommen sind, durch Angaben der Quellen sichtbar gemacht wurden.

Die eingereichte schriftliche Fassung der Arbeit entspricht der auf dem elektronischen Speichermedium.

Weiterhin versichere ich, dass diese Arbeit noch nicht als Abschlussarbeit an anderer Stelle vorgelegen hat.

.....
Datum und Unterschrift