Brandenburg
University of Technology
Cottbus - Senftenberg

The Multiple Traveling Salesperson Problem with Moving Targets

Anke Stieber

Cottbus Mathematical Preprints
COMP # 26(2022)

The Multiple Traveling Salesperson Problem with Moving Targets

Von der Fakultät 1 - MINT - Mathematik, Informatik, Physik,
Elektro- und Informationstechnik
der Brandenburgischen Technischen Universität Cottbus–Senftenberg
genehmigte Dissertation
zur Erlangung des akademischen Grades eines

Dr. rer. nat.

vorgelegt von

Anke Stieber geb. Reuter

geboren am 15.08.1979 in Pasewalk

Vorsitzende: Prof. Dr. rer. nat. habil. Sabine Pickenhain
Gutachter: Prof. Dr. rer. nat. habil. Armin Fügenschuh
Gutachter: Prof. Dr. rer. nat. habil. Ekkehard Köhler
Gutachter: Prof. Dr. rer. nat. habil. Markus Bause
Tag der mündlichen Prüfung: 13. September 2022

Zusammenfassung

Bei militärischen Operationen und Auslandseinsätzen ist es notwendig, die verwendete Infrastruktur sowie militärische Objekte, wie beispielsweise Militärflugplätze oder Feldlager vor feindlichem Beschuss durch Raketen, Artillerie oder Mörsergranaten zu schützen. Derartige Angriffe können aus verschiedenen Richtungen kommen. Laserwaffen sind eine Möglichkeit diese Bedrohungen abzuwehren. Durch den gerichteten Beschuss können feindliche Geschosse (Ziele) in Sekundenschnelle neutralisiert werden. Daher werden mehrere Laserwaffen um oder auf dem zu schützenden Gebiet verteilt. Zur Abwehr eines Zieles wird derjenige Laser ausgewählt, der die Zerstörung des feindlichen Zieles mit geringster Bewegung der Strahlführungseinheit des Lasers ermöglicht. Ziel ist es, den Schaden der Angriffe zu minimieren. Demzufolge sollen möglichst alle Ziele zerstört werden und vorzugsweise mit den kleinstmöglichen Bewegungen der Laser. Diese Anwendung ist in der Professur für Mess- und Informationstechnik an der Helmut-Schmidt-Universität/Universität der Bundeswehr Hamburg untersucht worden, siehe dazu auch Knapp und Rothe [Kna12]. In der Veröffentlichung wird eine vollständige Enumeration aller Möglichkeiten beschrieben und daraus eine Zuordnung der Laserwaffen zu den Zielen bestimmt.

Die vorliegende Arbeit widmet sich dem Problem der Zuordnung mit Mitteln der Optimierung. Hierzu nutzen wir eine Beschreibung als Multiple-Traveling-Salesperson-Problem mit sich bewegenden Zielen, wobei die Laser als Verfolger (salespersons) fungieren, welche die sich bewegenden Ziele besuchen/neutralisieren müssen. Die Ziele bewegen sich kontinuierlich über der Zeit auf sogenannten Trajektorien (Flugbahnen). Zu jedem Ziel gibt es ein Zeitfenster, in dem das Ziel sichtbar ist und innerhalb dessen es besucht werden muss. Die Aufgabe besteht darin, Routen für die Verfolger zu finden, so dass jedes Ziel genau einmal besucht wird und die Summe der zurückgelegten Entfernungen minimal ist.

Zu dem hier vorliegenden Problem gibt es nur begrenzte Beiträge in der Literatur. Oft ist in den Forschungarbeiten die Bewegung der Ziele eingeschränkt oder nicht kontinuierlich, und Testinstanzen werden mit heuristischen Verfahren gelöst. Die Herausforderung in dieser Arbeit besteht darin, zu untersuchen, ob exakte Verfahren in der Lage sind reale Instanzen adäquat schnell zu lösen. Dazu betrachten wir das Problem von zwei Seiten, offline und online. Bei der Offline-Betrachtung liegt die gesamte Information der Ziele (Trajektorien und Zeitfenster) zu Beginn der Optimierung vor. Im realen Online-Fall hingegen werden diese Daten erst nach und nach mit Erscheinen der jeweiligen Ziele verfügbar.

Ein zentraler Aspekt dieser Arbeit ist es, ein geeignetes Model zur Beschreibung des Multiple-Traveling-Salesperson-Problems mit sich bewegenden Zielen zu entwickeln. Wir stellen fünf verschiedene Modellansätze vor, wobei der Zeitaspekt auf verschiedene Arten modelliert wird. Zum einen unterscheiden wir in zeitdiskretisierte und zeitkontinuierliche Modelle und zusätzlich unterteilen wir diese in einstufige und zweistufige Modelle. Während die einstufigen Modelle die gesamte Information in einem einzigen Mixed-Integer-Linear-Programm zusammenfassen, werden bei den zweistufigen Modellen in einer ersten Stufe die Zeitrestriktionen vollständig relaxiert, und anschließend durch Unterprogramme (zweite Stufe) wieder hergestellt. Wir entwickeln ein exaktes Lösungsverfahren für die zweistufigen Modelle, das die Callback-Funktionalität von cplex nutzt. Als fünftes Modell verwenden wir einen Set-Partitioning-Ansatz.

Bei der Abwehr von Geschossen handelt es sich um ein Echtzeit-Problem, das bedeutet, dass Lösungsalgorithmen einen bestimmten zeitlichen Rahmen für die Generierung von Lösungen einhalten müssen. Ein Ziel dieser Arbeit ist es, zu untersuchen welches Modell hinsichtlich der Laufzeit am besten performt. Dafür werden zufällig generierte Testinstanzen mit 6 bis 20 Zielen, 1 bis 6 Verfolgern und 3 verschiedenen Zeitdiskretisierungen erstellt.

Durchgeführte Tests mit linearen Trajektorien zeigen, dass bei Instanzen mit 1 oder 2 Verfolgern, bis zu 8 Zielen und der höchsten Diskretisierung das zeitdiskretisierte Zweistufen-

modell die besten Laufzeiten erreicht. Mit dem zeitdiskretisierten Einstufenmodell lassen sich sogar Instanzen mit bis zu 10 Zielen innerhalb von 3 Sekunden lösen. Die Modelle mit kontinuierlichen Zeitvariablen haben insgesamt schlechtere Laufzeiten. Die vorgestellten Varianten sind jedoch nicht auf lineare Trajektorien beschränkt. Wir untersuchen den Einsatz von nichtlinearen Flugbahnen mit den beiden Einstufenmodellen. Das zeitdiskretisierte Modell erreicht Laufzeiten von unter 3 Sekunden, während das zeitkontinuierliche Modell nicht einmal in der Lage ist, kleine Instanzen in angemessener Zeit zu lösen.

Anschließend testen wir mit dem zeitdiskretisierten Einstufenmodell einen realen Einsatz anhand des Online-Multiple-Traveling-Salesperson-Problems mit sich bewegenden Zielen. Hierbei bleiben die Verfolger solange auf ihren aktuellen Routen, bis ein oder mehrere neue Ziele erscheinen. An dieser Stelle muss entschieden werden, wie die aktuellen Routen gemäß der neuen Information angepasst werden. Hier kommen bekannte Verfahren wie REPLAN und IGNORE zum Einsatz. Für unsere Onlineexperimente verwenden wir Testinstanzen mit 8 bis 20 Zielen und nutzen zusätzlich sehr kurze Trajektorien. Dadurch können nicht alle Ziele besucht werden, wodurch sich jedoch die eingesetzten Verfahren besser beurteilen lassen. Für vier Verfolger, die gleichmäßig verteilt starten, lassen sich mehr als 76% der Testinstanzen lösen ohne ein Ziel auszulassen. Dieser Wert kann noch auf über 78% erhöht werden, in dem wir die Summe der frühestmöglichen Ankunftzeiten an den Zielen minimieren.

Ein weiterer wichtiger Teil dieser Arbeit ist unser Beitrag zur Kompetitivitätsanalyse, einer Methode zur Bewertung von Online-Algorithmen. Dazu beschränken wir uns auf einen Verfolger und betrachten das Online-Traveling-Salesperson-Problem mit sich bewegenden Zielen auf der reellen Zahlengeraden. Hier werden keine Zeitfenster betrachtet, die Ziele bewegen sich solange bis sie besucht werden. Zu diesem Problem beweisen wir eine untere Schranke für den Kompetitivitätsfaktor. Außerdem liefern wir einen Online-Algorithmus und beweisen seinen Kompetitivitätsfaktor. Anschließend vergleichen wir unseren Algorithmus mit einem Online-Algorithmus, den wir für das Online-Traveling-Salesperson-Problem mit sich bewegenden Zielen auf der reellen Zahlengeraden adaptiert haben und der auf AUSIELLO et al. [Aus01] basiert. Hierfür beweisen wir ebenfalls den Kompetitivitätsfaktor. Beide Kompetitivitätsfaktoren sind abhängig vom Geschwindigkeitsverhältnis von Verfolger und Zielen und je nach Geschwindigkeitsverhältnis ist der Kompetitivitätsfaktor des einen oder anderen Online-Algorithmus besser. Für unsere verwendeten Online-Instanzen ergibt sich damit ein analytisch bestimmter Kompetitivitätsfaktor von 3,47 und experimentell erhalten wir 1,55.

Zusammenfassend halten wir fest, dass die Art der Modellierung des Multiple-Traveling-Salesperson-Problems mit sich bewegenden Zielen die Laufzeit der Optimierung beeinflusst. Basierend auf den durchgeführten Tests, erscheint das zeitdiskretisierte Einstufenmodell als am geeignetsten für einen realen Einsatz. Die erzielten theoretischen Resultate zum Online-Traveling-Salesperson-Problem mit sich bewegenden Zielen auf der reellen Zahlengeraden stellen neue Ergebnisse in diesem Bereich dar.

Abstract

Military installations and objects in out-of-area missions, e.g., an air base or a field camp, must be protected from incoming hostile rockets, artillery or mortar fire. Simultaneous attacks from different firing positions are considered. Lasers as directed energy weapons are able to destroy those targets within seconds. Thus, several lasers are deployed over or nearby the protective area to be selected for the countermeasure. Generally, the laser is assigned to a target, that applies the smallest movement of its direction unit to aim at it. The goal is to minimize the damage and thus, to destroy all incoming targets, preferably with the smallest possible movements of the direction units. This application has been investigated at the professorship of Measurement and Information Technology at the Helmut Schmidt University/University of the Federal Armed Forces Hamburg, see Knapp and Rothe [Kna12]. Here, the decision for the countermeasure is based on complete enumeration.

We focus on the assignment of lasers to targets by means of optimization. We model the problem as a multiple traveling salesperson problem with moving targets, where the salespersons correspond to the lasers. The targets move over time on continuous trajectories. Additionally, each target is given a visibility time window. The task is to find routes for multiple salespersons such that each target is reached exactly once within its visibility time window and the sum of all traveled distances of all salespersons is minimal.

There is few literature about a generalization with moving targets and multiple salespersons. Often research articles focus on targets, whose movement is restricted or based on a stochastic process. Due to the high computational burden mainly heuristic approaches are used to solve test instances. We investigate if exact methods are able to solve real-world instances in reasonable time. On that account, we address the problem from two sides, offline and online. Considering the offline case, all input data is given in advance. In the online case, input information is revealed step-by-step during executing the algorithm. The information about trajectories and time windows is available from the moment the targets get visible.

One essential aspect studied in this work is to find an appropriate model formulation to solve instances of the multiple traveling salesperson problem with moving targets in reasonable time. We present five different modeling approaches, where the time requirements are handled in different ways. The approaches are classified in time-discrete and time-continuous models and further in one-stage and two-stage models. While for a one-stage model all information is included in a single mixed integer program, we formulate the problem without time constraints and ensure time feasibility by solving a number of subproblems for a two-stage model. We present a solution procedure based on the Callback functionality of cplex to solve the two-stage models. The fifth model applies a set partitioning approach.

From the mathematical point of view, we are faced with a real-time problem – an algorithm has to provide a solution within prescribed time requirements. Our goal is to examine what formulation is most suitable in terms of runtime to solve the multiple traveling salesperson problem with moving targets with exact solution methods. Due to the lack of real input data, we randomly generate test instances with 6 to 20 targets, 1 to 6 salespersons and 3 different discretization levels.

Computational tests with linear trajectories show that for instances with 1 or 2 salespersons, up to 8 targets, and with the finest discretization level the time-discrete two-stage model performs best. The time-discrete one-stage model can even solve instances up to 10 targets within 3 seconds. Models with a continuous time handling produce poor runtimes. However, the proposed models are not restricted to linear trajectories. We analyze the handling of non-linear trajectories with both one-stage models. Computational experiments show that the time-continuous model cannot even solve small test instances with non-linear

trajectories in reasonable time. The runtimes for the time-discrete model are below 3 seconds for the coarse discretization level.

Finally, we investigate the real case and address the online multiple traveling salesperson problem with moving targets with the time-discrete one-stage model. Here, salespersons follow current tours until one or more new targets arrive. This new target information has to be incorporated into the current tours. Therefor, we use the two familiar strategies REPLAN and IGNORE. We carry out experiments on randomly generated test instances with 8 to 20 targets, medium discretization level, and very short trajectory lengths. Short trajectories lead to instances, where it is often not possible to visit all targets, however, that way we get distinguishable and descriptive results for the applied strategies. The computational results show that for 4 salespersons equally distributed over the considered space over 76% of all instances can be solved without any misses. This value can slightly be increased to over 78% by an objective function that minimizes the sum of earliest interceptions.

Another important aspect of this work is our contribution to competitive analysis, a method to evaluate the quality of online algorithms. Here, we restrict the problem considered so far to one salesperson and address the online moving targets traveling salesperson problem on the real line. There are no time windows. At first, we prove a lower bound for the competitive ratio regarding this problem. Then, we develop an online algorithm and present its competitive ratio with the corresponding proof. For the static case, we show that this ratio is tight. Furthermore, we compare our algorithm to an online algorithm based on AUSIELLO et al. [Aus01] and adapted to moving targets. Therefor, we also prove a competitive ratio. The decision which algorithm has a better competitive ratio depends on the speed ratio of salespersons and targets. Applying the speed ratio of our test instances we obtain an analytical competitive ratio of 3.47, and a practical one of 1.55.

Finally, we can conclude, that the choice of modeling has an influence on the solution time of real-world instances of the multiple traveling salesperson problem with moving targets. Due to the experiments carried out in this work the time-discrete one-stage model is our model of choice. The theoretical results obtained for the online moving target traveling salesperson problem on the real line are new in this research area.

Acknowledgments

First of all I want to thank Prof. Dr. Armin Fügenschuh for supervising this thesis and for his professional advice throughout this research. I am also thankful to Prof. Dr. Markus Bause for his constant encouragement and for providing infrastructure during the last months of this thesis. Furthermore, I would like to thank the rest of my thesis committee: Prof. Dr. Sabine Pickenhain and Prof. Dr. Ekkehard Köhler.

In particular, I thank my husband for the support he gave me and for the motivating discussions when I was stuck in providing the long proofs. I am also thankful to our two children for their tremendous patience, sorry, that I had very little time for you during writing. I would also like to give special thanks to my parents for their constant support and for always being there when we need you.

Finally, I am grateful to all the people with whom I have worked at the Helmut Schmidt University, especially Liana and Marina for the enjoyable office atmosphere.

Credits

Main text parts and computational results of Chapter 1, Chapter 3 (3.1–3.3), Chapter 4 (4.1) and Chapter 5 (5.1, 5.2) have already been published in Central European Journal of Operations Research, see [Sti20]. Furthermore, the context and results regarding the time-discrete model in Section 3.6 and Section 5.4 have been published in the Operations Research Proceedings 2017, see [Sti18]. The time-discrete one-stage model provided in Section 3.1 is based on the model presented in [Füg14]. This preliminary version has been developed by Armin Fügenschuh.

Contents

1 The Multiple Traveling Salespersons Problem with Moving Targets

The classical traveling salesperson problem (TSP) is a combinatorial optimization problem. The task for a salesperson starting from a depot is to visit a number of fixed targets (cities, points) each exactly once and afterwards to end the tour at the depot. The TSP is an ordering problem. The first mathematical problem definition of the TSP was formulated 1930 in Vienna by Karl Menger [Men30]:

> Wir bezeichnen als Botenproblem (weil diese Frage in der Praxis von jedem Postboten, übrigens auch von vielen Reisenden zu lösen ist) die Aufgabe, für endlich viele Punkte, deren paarweise Abstände bekannt sind, den kürzesten die Punkte verbindenden Weg zu finden. Dieses Problem ist natürlich stets durch endlich viele Versuche lösbar. Regeln, welche die Anzahl der Versuche unter die Anzahl der Permutationen der gegebenen Punkte herunterdrücken würden, sind nicht bekannt. Die Regel, man solle vom Ausgangspunkt erst zum nächstgelegenen Punkt, dann zu dem diesem nächstgelegenen Punkt gehen usw., liefert im allgemeinen nicht den kürzesten Weg. (Karl Menger, 1930)

While it is relatively easy to understand the problem and to find a good solution, it is computationally difficult to find a proven optimal solution. In 1972 Richard Karp showed, that the Hamiltonian cycle problem is NP-complete [Kar72], which implies the NP-hardness of the TSP, see also Garey and Johnson [Gar79]. For a survey on TSP we refer to Lawler et al. [Law85] or Reinelt [Rei94].

There are different variants of the TSP. This research focuses on a dynamic variant with multiple salespersons and time windows. The dynamic behavior arises from the targets, which are not fixed as in the classical case. The targets move continuously on trajectories in a certain space. All targets have got a determined speed value and each one is assigned a visibility time window. Thus, a salesperson can only intercept a target within its respective time interval. This dynamic TSP generalization is called multiple traveling salespersons problem with moving targets (MTSPMT).

1.1 Application

The MTSPMT as a dynamical generalization of the TSP is suitable for real-world problems. A possible application of the MTSPMT can be found in the defense sector. Military installations and objects in out-of-area missions, e.g., an air base or a field camp, must be protected from incoming hostile rockets, artillery or mortar (RAM) fire. Simultaneous attacks from different firing positions can be considered. The flight time of RAM is approximately 30 seconds, whereas the first 5-10 seconds are needed to radar-detect the threat by the surveillance radar and estimate its trajectory. Based on the estimated impact point a decision must be taken whether it needs to be destroyed. Lasers as directed energy weapons are capable of destroying a RAM target within seconds. With a battery of lasers deployed over or nearby the protective area decisions have to be taken, which available laser to select for the countermeasure. The selected laser then aims at the incoming target in the air for a certain period of time to destroy it. The further away a target is, the longer a laser has to fire at the target to safely destroy it. For reasons of simplification we assume that all targets can be destroyed with the same amount of energy, each laser fires the same period of time, and this period of time is set to one unit. Every laser has its own limited energy storage, e.g., a capacitor or a battery. It is assumed, that a laser intercepts only one RAM target at a certain point of time and dead zones of the laser weapons are not considered.

Generally, the closest laser is assigned to a target, where “close” does not refer to the physical distance, but to the angle the laser needs to traverse for aiming at the RAM target.

Furthermore, due to safety requirements, a laser should not shoot across the protected area. However, this rule will be neglected, if the destruction of the RAM target is impossible otherwise. The goal is to minimize the damage of the field camp and thus, to destroy all incoming targets, preferably with the smallest possible movements of the lasers. The closest laser is used instead of the laser destroying the RAM in minimum time, to prefer small angle movements over large angle movements. Since, in case of a failure (target could not be destroyed) an adjustment of the laser is then more likely to succeed. The goal is to destroy all incoming RAM threats.

This application was investigated at the professorship of Measurement and Information Technology at the Helmut Schmidt University/University of the Federal Armed Forces Hamburg, see Knapp and Rothe [Kna12]. The authors present ideas of analysing targets according to their level of threat and provide a schematic system to engage targets by lasers. Iterating all possibilities between lasers and targets is proposed to decide which laser to take for the engagement of a certain target.

1.2 Problem Description

The application described above is modeled as an MTSPMT. In this context, the lasers correspond to the salespersons and the incoming RAM entities are the moving vertices, which we call moving targets. Every moving target has a certain visibility time window. The time window starts at that moment at which the target is radar-detected, its trajectory is computed and reachability by all lasers is guaranteed. The visibility time window ends at the latest point in time where a destruction of the target is possible, which is before impact. The trajectory of a target can be described by a function over time. It is assumed, that every target has the same constant speed value. Every salesperson starts its tour from an initial depot, which can be located in the center of the considered space. Alternatively, the depots may be evenly distributed over the area or the salespersons start from an arbitrary position (e.g., the last used positions). Accordingly, it is not required for the salespersons to finish their tours at the depot. We assume, that all salespersons have the same maximum speed value.

Every target must be visited once by exactly one salesperson within its visibility time window. The objective function is to minimize all traveled distances of all salespersons. Usually, the optimization goal for a dynamic TSP is to minimize travel times. However, the concept of intercepting a target as early as possible can cause a laser to traverse a long distance to catch that target earliest possible. To save traveling time and be ready for next upcomming targets we minimize the distances. Moreover, it is most important to safely destroy a target than to adjust for a second shot in case of a miss. Destruction of a target is more likely conducted when lasers traverse small angles. In our test instances we assume destruction of a target in the first try. The restriction that each target has to be destroyed is above all and will be modeled as the demand constraints.

From the mathematical point of view, this application is an online optimization problem, where the complete data of the problem instance is not given in advance and a decision has to be made immediately. In our study we initially solve it as an offline problem. Having developed a solution algorithm for the offline problem, it can be adapted to solve the online problem by a moving horizon approach, that is, the data is integrated to the (offline) algorithm at runtime, which most likely leads to a (partial) reversion of the current solution. Practically speaking, the targets, that are visible first, are optimized and the salespersons start their tours to intercept these targets accordingly, until new targets arise. When new targets arise, the algorithm has to decide how to insert the new data to the already constructed solution. One possibility is to finish unfinished tours and to optimally integrate

the new targets afterwards. Another possibility is to update current tours directly when new targets become visible, which may lead to aborting current tours and heading for other targets instead.

The first part of this thesis deals with the MTSPMT as an offline problem. In the second part (Section 5.5 and Chapter 6) the online variant, different versions of updating the current solution, and theoretical results regarding competitive analysis are addressed.

1.3 Related Problems and Applications

If we restrict the number of salespersons to one, the position of all cities to be fixed over time and extend all visibility windows to the whole time horizon, we obtain the classical TSP. Here, related problems and applications of the MTSPMT and TSP are addressed. The first problem is the Moving-Target TSP, which is the MTSPMT restricted to one salesperson. It was addressed by Helvig et al. [Hel98], who also mentioned some possible applications for the Moving-Target TSP: a supply ship, that resupplies patrolling boats or an airplane that must intercept a number of mobile ground units. They also addressed the Multi-Pursuer Moving-Target TSP with Resupply, where multiple pursuers are considered and each pursuer must return to the origin for resupply after intercepting a target.

Many practical applications such as routing and scheduling of vehicles have a time-dynamic component inherent. In the last two decades logistic distributions were faced with an increasing traffic load, traffic congestion and uncertainty in travel times caused by bad weather conditions or other random incidences. In this context Vehicle Routing Problems (VRPs) with time-dependent traveling times became more and more important. Here, the VRP with time-dependent traveling times and time windows is very similar to the MTSPMT, because traveling times for the MTSPMT are also time-dependent, due to the movement of the targets. However, the movement of a target does not only influence the length of a certain arc and thus the travel time to a certain target, but to all other targets as well. Since all targets are moving simultaneously and constantly the length of an arc has two degrees of freedom, i.e., the length of an arc is determined by the time the arc is entered and the time the arc is left. Both these times exactly correspond to two spatial positions of the incident targets.

Thus, the MTSPMT has varying distances and varying travel times between 2 targets and both of these do not correspond to each other since waiting is permitted. Assuming a target approaching a salesperson, the salesperson moves the smallest distance to the trajectory of the target with maximum speed and possibly waits there to catch the target, this results in a smaller average speed of the salesperson. The other case is, when the target is heading away from the salesperson, then it is better to catch the target as early as possible and without waiting, here the average speed is the maximum speed. The change in distances between targets is different compared to time-dependent VRP (TDVRP). For the MTSPMT minimizing the travel time is different from minimizing the distance traveled.

1.4 Literature

This thesis addresses a generalization of the classical traveling salesperson problem by considering multiple salespersons and moving targets with time windows. For a survey on the classical TSP we refer to Reinelt [Rei94].

There are a number of articles in the literature that deal with dynamical TSP generalizations. However, the TSP literature concerning moving targets is less developed [Eng13; Hel03; Jia05]. The first articles addressing the traveling salesperson problem with moving

targets (TSPMT) are Helvig et al. [Hel98; Hel03]. The authors present an exact and an approximation algorithm for the one-dimensional case, where targets and salespersons move on a straight line. The targets have unique speeds and all move simultaneously right from the beginning. The exact algorithm is based on dynamic programming and performs in $\mathcal{O}(n^2)$, where n is the number of targets. Other specific variants of the TSPMT with restrictions for target speed, movement and the number of targets are also addressed, e.g., the targets move towards the origin (starting point of the salesperson), never reach the origin, or the TSPMT with resupply, where the pursuer must return to the origin after intercepting a target. The proposed algorithms are not applicable to the general case of the TSPMT. Specific variants of the TSPMT and the TSPMT with resupply are also addressed by Jiang et al. [Jia05], Jindal et al. [Jin11], and Englot et al. [Eng13]. Solution approaches are mainly heuristics such as genetic algorithms.

A very recent article by Hassoun et al. [Has20] considers the TSPMT variant, where the targets do not start their movement simultaneously, instead the targets enter the system over time. Thus, each target cannot be intercepted earlier than its release times. The authors study the problem on the real line with the assumption, that all targets move with the same constant speed. The authors provide a $\mathcal{O}(n^5)$ time algorithm based on dynamic programming, where n is the number of targets. However, this variant is different from the one dimensional case in Helvig et al. [Hel03].

Some articles address a stochastic variant of the dynamic traveling salesperson problem. Here, locations of the targets change over time caused by stochastic processes. For example, problem instances in Ahrens [Ahr15] were generated from static TSP datasets originated from a published and standardized library. The movement of each target is modeled by a Gaussian-distributed random distance vector that is added to its location. Time is integrated in a way that the movement from one target to another can be done in one time step and the targets localized in the 2-dimensional space move at each time step. The instances were solved by heuristics in an online calculation. The author examined the applicability of standard (static) TSP solvers to the dynamic instances. Computational experiments were carried out with the Tour Construction Framework which combines global and local heuristics and the TSP tour construction heuristic “nearest neighbor”.

Another TSP variant is the generalized TSP (GTSP) also known as set TSP or group TSP. Here, the targets are organized in subsets (groups, clusters) and at least one element per subset has to be visited. In case exactly one element per subset has to be visited the problem is called equality GTSP (E-GTSP). For an asymmetrical E-GTSP the costs of anti-parallel arcs do not have to be equal. We assume, that the MTSPMT is modeled with discrete time steps. Then, we obtain a copy of each target for each time step in the respective time window. Let us consider the copies of each target as a subset (or cluster). To this end, we have a set of clusters and the task is still that each cluster has to be visited once by exactly one salesperson. In case of considering one salesperson, this gives us an E-GTSP. Moreover, the discrete target copies describe the discretized trajectory of the target. Any such trajectory position is characterized by the target number and the time step. Considering two positions of different targets as two nodes then there is an arc between these nodes, if and only if a salesperson is able to travel the Euclidean distance of the nodes within the time difference. That means, the salesperson is not allowed to exceed its maximum speed value. An instance of a MTSPMT generally contains no anti-parallel arc due to the progression of time. However, this can be interpreted as anti-parallel arcs with infinite costs. In this context an instance of the TSPMT can be formulated as an instance of the asymmetrical E-GTSP.

The asymmetrical GTSP and E-GTSP have been investigated in Laporte et al. [Lap87]. Noon and Bean [Noo93] showed, that any problem that can be modeled as a GTSP,

can be transformed into an asymmetrical TSP. The equivalent problem with multiple salespersons is the generalized vehicle routing problem (GVRP), see Ghiani and Improta [Ghi00]. For the symmetrical case there is a contribution by Sundar and Rathinam [Sun16]. The authors addressed the generalized multiple depot TSP (GMDTSP) and provide a polyhedral study for this problem class. They presented a branch-and-bound approach that was realized by the callback functionality of ibm ilog cplex Optimizer (short cplex). Computational experiments were carried out for 14 to 105 targets (which correspond to the number of target copies in our notation) and a maximum number of 21 clusters (which correspond to the number of targets in our notation).

In Picard and Queyranne [Pic78] a similar problem is considered: the time dependent traveling salesperson problem (TDTSP). Here, a complete graph is considered and the cost values (or the travel times) depend on the position of the target in the tour. Thus, there are number of targets many discrete time steps and only a discrete number of cost values. The authors use shortest paths in a multipartite network for the solution with branch-and-bound and relaxation methods. Abeledo et al. [Abe13] is based on the formulation given by Picard and Queyranne [Pic78] and provides a study of the TDTSP polytope. The authors present several facet defining inequalities and perform computational experiments with their branch-and-cut-and-price algorithm.

A generalization of the TDTSP is the time dependent vehicle routing problem (TDVRP). Here, more than one salesperson (vehicle) is considered and capacity restrictions as well as time windows are imposed. Malandraki and Daskin first formulated the TDVRP [Mal92] as a mixed-integer linear program. They used step functions to model the travel times depending on the distance between the nodes and the time of the day. Furthermore, a nearest-neighbor heuristic was presented. Due to the computational complexity of TDVRP scientific contributions mainly focus on heuristic approaches. Some exemplary articles follow.

Ichoua et al. [Ich03] integrated the time dependency in the travel speed instead of in the travel time. The travel speeds were modeled by step-functions of the time of the day, leading to piece-wise linear functions of the travel times. They report on a tabu search algorithm, that was adapted to the time-dependent model and experiments were conducted in a static and dynamic environment.

With regard to better reliability in scheduling and routing problems, Fleischmann et al. [Fle04] used time-varying continuous travel times and retrieved the information from a traffic information system that was tested in the city of Berlin. Computational results were reported for several VRP heuristics.

Haghani and Jung [Hag05] and Jung and Haghani [Jun01] also applied continuous travel time functions and gave a MILP formulation of the TDVRP with discrete time steps. A genetic algorithm was presented and the results were compared with an exact solution method for small and mid-sized instances and for bigger instances a lower-bound procedure was used. Instances with 5 to 30 nodes and 10, 15 and 30 time steps were considered.

Another article deals with a more realistic modeling of the time-dependent travel times. Mancini [Man14] models the travel times by a polynomial instead of linear functions, computational results were carried out by a heuristic method.

Very few literature is published concerning exact methods, some of them are e.g., Albiach et al. [Alb08] and Soler et al. [Sol09]. Based on the work of Noon and Bean [Noo93] they provide a theoretical work of transforming an instance of the asymmetric TDTSP with time windows or the TDVRP with time windows into an instance of the Asymmetric TSP (ATSP) or VRP (AVRP) respectively. Soler et al. [Sol09] is a generalization of the first one, because it deals with multiple salespersons. The conversions are carried out by transforming

the underlying graph into an instance of the GTSP or the GVRP respectively, and then into the ATSP and the AVRP. Albiach et al. [Alb08] also performed computational experiments using an exact algorithm for the Mixed General Routing Problem. Instances with up to 222 vertices and one salesperson were considered, where the number of arcs is between 5 and 20% of the arcs of a complete graph.

Van Woensel et al. [Van07] introduced a new approach to model potential traffic congestion. This model is based on a queueing approach to traffic flows. Computational experiments were carried out for small instances with 10 cities using explicit enumeration and for up to 100 cities with an ant colony heuristic.

In summary there are only few research contributions to the MTSPMT. Articles on the MTSPMT or the TSPMT mainly consider specific problems or impose restrictions on the movement of the targets. The MTSPMT in general is closely related to the time dependent TSP and VRP; the MTSPMT with discrete time steps is especially similar to the E-GTSP with multiple depots. The challenge of the MTSPMT is the continuous movement of all targets. An arc distance is not only dependent on the time the arc is entered but also on the time the arc is left. Both these times are related to the positions of the incident targets.

Since solving time-dependent problems is very time-consuming, most research articles do not concentrate on exact procedures. In our research we confront different modeling approaches and examine their computation times on randomly generated test instances. At first, we concentrate on an offline approach for solving MTSPMT instances to global optimality. Then, we use the most efficient model for an online consideration and compare online and offline results.

1.5 Contribution

In this thesis we investigate the time aspect of the MTSPMT in modeling when minimizing the traveled distances. Based on different ways of integrating time into the model formulation, five different model variants are presented and compared regarding performance. Two model approaches have already been published by Stieber et al. [Sti14], that is the time-discrete (TD) model and the time-continuous (TC) model. In addition, we generate time-free models. For these models we completely relax the time requirements in a first stage and use subprograms to check and create time feasibility in a second stage. These feasibility checker are based on discrete time steps on one hand and on the other hand we use a continuous time formulation such that not all but necessary time restrictions are integrated. These two resulting modeling variants are called time-free model with time-discrete feasibility checking (TFTD) and time-free model with time-continuous feasibility checking (TFTC). The optimal solutions of the two-stage models (TFTD and TFTC) are computed using a branch-and-bound framework. We present an algorithm, that makes use of a callback function (an advanced feature of cplex). As the last approach we embed the described models in a set partitioning approach. Computational experiments are carried out using randomly generated test instances with a varying number of targets, salespersons and time steps. The aim is to find a formulation to solve MTSPMT instances with the best performance in terms of CPU time.

In the second part of this thesis we use the obtained results to simulate a real-world application by considering the online case of the MTSPMT. That means, that the information of the targets is not given in advance instead it is revealed step-by-step and thus, a re-optimization is needed, which most likely cause a reversion and update of the current solution. Furthermore, we focus on online algorithms and methods to measure their

performance. The main part deals with the online TSPMT on the real line. Here, only one salesperson is considered and the targets are moving on the real line. Visibility time windows are replaced by release times. We develop an online algorithm for this problem and prove its competitive ratio, which is a measure for the quality of online algorithms. Furthermore, we compare our results to an online algorithm from the literature, which we adapt to moving targets. For this algorithm we also prove its competitive ratio. The decision which algorithm has a better competitive ratio depends on the speed ratio of salespersons and targets.

1.6 Structure

The remainder of this thesis is organized as follows. The basic definitions and concepts used in this work are given in Chapter 2. We present the different model formulations for the MTSPMT in Chapter 3 as well as some model adaptations for certain situations. Chapter 4 focuses on the solution procedures for the different models. Conducted experiments and obtained results are described in Chapter 5. Main topics of Chapter 6 are competitive analysis and the online MTSPMT with variants. Finally, concluding remarks in Chapter 7 complete this thesis.

2 Basic Definitions and Concepts

In this chapter we introduce the main concepts of linear, integer, and mixed integer programming as well as second-order cone programming. We provide the basics in graph and network theory and introduce selected concepts of computational complexity theory. Finally, an overview of interval arithmetic is given together with a concept of interval propagation adapted to a sequence of moving vertices.

A concise presentation is provided, which is not meant to be self-supporting. We assume basic knowledge in combinatorial optimization, linear and (mixed) integer programming as well as graph and network theory. Regarding combinatorial optimization and linear programming we recommend the books Grötschel, Lovász, and Schrijver [Grö88], Cook, Cunningham, Pulleyblank, and Schrijver [Coo97], Papadimitriou and Steiglitz [Pap17], Korte and Vygen [Kor00], and Chvátal [Chv83]. Standard references for linear and integer programming as well as cutting planes are Nemhauser and Wolsey [Nem88] and Schrijver [Sch98]. For complexity theory and the definition of $\mathcal{NP}$-hard problems we refer to Garey and Johnson [Gar79]. An introduction to convex optimization and especially to second order cone programming can be found in Boyd and Vandenberghe [Boy04].

2.1 Basic Definitions

We denote by $\mathbb{R}$, $\mathbb{Q}$, and $\mathbb{Z}$ the sets of real, rational, and integral numbers, respectively. The exclusion of negative numbers is indicated by $\mathbb{R}_+$, $\mathbb{Q}_+$ and $\mathbb{Z}_+$. Whole numbers are defined by $\mathbb{N} \coloneqq \mathbb{Z}_+ \backslash \{0\}$. Given a set E, we denote by 2^E the *power set* of E, which is the set of all subsets of E, including the empty set and E itself.

Any given vector $c \in \mathbb{R}^n$, $n \in \mathbb{N}$, is regarded as a column vector, the corresponding row vector is c^T. Given a matrix $A \in \mathbb{R}^{m \times n}$, we use $\{1, \dots, m\}$ to indicate the set of row indices of A and $\{1, \dots, n\}$ to indicate the set of column indices, respectively. Furthermore, the submatrix A_{IJ} of A with $I \subseteq \{1, \dots, m\}$ and $J \subseteq \{1, \dots, n\}$ is generated from A by removing all rows of the index set $\{1, \dots, m\} \backslash I$ and all columns of the index set $\{1, \dots, n\} \backslash J$. If we only restrict the column index set, we write $A_{\cdot J}$ and if we restrict the row index set $A_{I \cdot}$, respectively. If it is clear from the context, we omit the point and simply write A_I or A_J.

A *metric space* M is an ordered pair $M = (X, d)$ where X is a nonempty set and $d : X \times X \to \mathbb{R}_+$ is a metric or distance function on X, such that for any $x, y, z \in X$ the following conditions are satisfies:

(i) $d(x, y) = 0 \iff x = y$
(ii) $d(x, y) = d(y, x)$
(iii) $d(x, y) \leqslant d(x, z) + d(z, y)$.

A set $S \subseteq \mathbb{R}^n$ is *convex*, if for any two elements $x, y \in S$ the line segment $\{\lambda x + (1-\lambda)y : 0 \leqslant \lambda \leqslant 1\}$ is contained in S.

A *combinatorial optimization problem* is defined as follows. Let E be a finite set, $\mathcal{I}$ be a subset of 2^E, where each element of $\mathcal{I}$ is denoted as a feasible set or a feasible solution. In addition, there is a function $c : E \to \mathbb{R}_+$ such that for any set $F \subseteq E$ the value of the function is defined as $c(F) := \sum_{e \in F} c(e)$. The task is to find a set $I^* \in \mathcal{I}$ with $c(I^*)$ is minimal (maximal), that is, $c(I^*) \leqslant c(J)$ for all $J \in \mathcal{I}$ ($c(I^*) \geqslant c(J)$ for all $J \in \mathcal{I}$).

Given $m, n \in \mathbb{N}$, $b \in \mathbb{R}^m, c \in \mathbb{R}^n$ and a matrix $A \in \mathbb{R}^{m \times n}$. The task to find a vector $x \in \mathbb{R}^n$, which has the smallest (greatest) possible value $c^T x$ among all vectors holding $Ax \leqslant b$ is called *linear program* (LP). We write shortly

$$\begin{aligned} \min \quad & c^T x \\ \text{subject to} \quad & Ax \leqslant b \\ & x \in \mathbb{R}^n, \end{aligned} \qquad \begin{aligned} \max \quad & c^T x \\ \text{subject to} \quad & Ax \leqslant b \\ & x \in \mathbb{R}^n. \end{aligned}$$

We call min $c^T x$ the *objective function* (or *cost function*) and $Ax \leqslant b$ the *constraints* of the LP. Any vector $x \in \mathbb{R}^n$, that meets all constraints is a *feasible solution* of the LP. The set of all feasible solutions is given by

$$P := \{x \in \mathbb{R}^n : Ax \leqslant b\}.$$

The coefficients of x are called *variables*.

Considering an LP with the additional restriction $x \in \mathbb{Z}^n$, then we speak of an *integer program* (IP). In case only a part of the variables has to be integral it is called *mixed integer program* (MIP). This problem is of the form

$$\begin{aligned} \min \quad & c^T x + h^T y \\ \text{subject to} \quad & Ax + Gy \leqslant b \\ & x \geqslant 0 \\ & y \geqslant 0 \\ & x \in \mathbb{Z}^n \\ & y \in \mathbb{R}^p \end{aligned} \tag{1}$$

where the matrix $A \in \mathbb{Q}^{m \times n}$, the matrix $G \in \mathbb{Q}^{m \times p}$ and the vectors $c \in \mathbb{Q}^n$, $h \in \mathbb{Q}^p$ and $b \in \mathbb{Q}^m$. IPs and MIPs in this thesis are solved with the Simplex algorithm (see Section 2.2), thus, rational data is used. In addition, we can also assume integral data, since rational equations or inequalities can be scaled appropriately to obtain integral data with the same set of feasible solutions. The set of all feasible solutions to (1) is

$$S := \{(x, y) \in \mathbb{Z}^n_+ \times \mathbb{R}^p_+ : Ax + Gy \leqslant b\} \tag{2}$$

and called *mixed integer set*. A *linear relaxation* of this mixed integer set, which discards the integrality requirement is denoted by

$$P^0 := \{(x, y) \in \mathbb{R}^n_+ \times \mathbb{R}^p_+ : Ax + Gy \leqslant b\}, \tag{3}$$

referring to a relaxation of the solution space. Then, a *linear programming relaxation* of (1), referring to a relaxation of the problem, is

$$\min\{c^T x + h^T y : (x, y) \in P^0\}. \tag{4}$$

2.2 The Simplex Algorithm

The simplex algorithm is a popular algorithm for solving linear programs and was developed by George Dantzig in 1947, see [Dan90] for its history of origins. The idea of this method is to move on the polyhedron underlying a linear program from vertex to vertex along edges until an optimal vertex is found. This idea is formalized in the following and

starts with some definitions. The two references Grötschel [Grö09] and Papadimitriou and Steiglitz [Pap17] were used for this section.

Definition 2.1. Given $m, n \in \mathbb{N}$, $b \in \mathbb{R}^m, c \in \mathbb{R}^n$ and a matrix $A \in \mathbb{R}^{m \times n}$. A linear program in the form of (5) is said to be in standard form,

$$\begin{aligned} \min \quad & c^T x \\ \text{subject to} \quad & Ax = b \\ & x \geqslant 0. \end{aligned} \tag{5}$$

The underlying polyhedron of (5) is

$$P^{=}(A, b) \coloneqq \{x \in \mathbb{R}^n : Ax = b, x \geqslant 0\}. \tag{6}$$

In general, a subset $P \subseteq \mathbb{R}^n$ is called a polyhedron if there is an $m \in \mathbb{Z}_+$, a matrix $A \in \mathbb{R}^{m \times n}$ and a vector $b \in \mathbb{R}^m$, such that

$$P = P(A, b) \coloneqq \{x \in \mathbb{R}^n : Ax \leqslant b\}. \tag{7}$$

We note, that the description (6) can be transformed into an description (7) by writing the equation $Ax = b$ as $Ax \leqslant b$ and $-Ax \leqslant -b$ as well as $x \geqslant 0$ as $-x \leqslant 0$.

A subset $G \subseteq \mathbb{R}^n$ is called *hyperplane*, if there is an $a \in \mathbb{R}^n \backslash \{0\}$ and $\beta \in \mathbb{R}$, such that

$$G = \{x \in \mathbb{R}^n : a^T x = \beta\}.$$

Then, the corresponding *halfspace* $H \subseteq \mathbb{R}^n$ is defined as

$$H = \{x \in \mathbb{R}^n : a^T x \leqslant \beta\}. \tag{8}$$

Obviously halfspaces are polyhedra and convex sets. Thus, every polyhedron $P \neq \mathbb{R}^n$ is a finite intersection of halfspaces and thus, convex. A bounded polyhedron is called a *polytope*.

Let $P \subseteq \mathbb{R}^n$ be a polyhedron of the form (7). Then, a subset $F \subseteq P$ is called a *face* of P, if there is an inequality $c^T x \leqslant \gamma$, which is valid for P and holds

$$F = P \cap \{x : c^T x = \gamma\}.$$

If there is an $x \in \mathbb{R}^n$ and $F = \{x\}$, we call F *vertex* of P. Additionally, also x is called vertex of P.

In the following we consider $B = (p_1, \ldots, p_m) \in \{1, \ldots, n\}^m$ and $N = (q_1, \ldots, q_{n-m}) \in \{1, \ldots, n\}^{n-m}$. Any index contained in B cannot be in N and vice versa. We use B and N as row vectors and additionally as sets, in case the order of the indices are irrelevant. The following should hold for B and N: $B \cap N = \varnothing$, $B \cup N = \{1, \ldots, n\}$ and $q_i < q_j$ when $i < j$.

In the following Definition 2.2 the basic notation for linear programming is given.

Definition 2.2. Given a system of equations $Ax = b$, where $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$ and $\operatorname{rank}(A) = m$. The index vectors B and N are given as defined above.

(i) If $A_{.B}$ is a regular matrix, then $A_{.B}$ is called *basis* of A and $A_{.N}$ is called *nonbasis* of A. The *basic solution* corresponding to $A_{.B}$ is a vector $x \in \mathbb{R}^n$ with $x_N = 0$ and $x_B = A_{.B}^{-1} b$.

(ii) Let $A_{.B}$ be a basis of A and the corresponding basic solution $x \in P^{=}(A,b)$, then $A_{.B}$ is called *feasible* and x is called *basic feasible solution*. In this case $x_B \geqslant 0$.

(iii) A basic feasible solution x, that corresponds to the basis $A_{.B}$ is called *nondegenerate* if and only if $x_B = A_{.B}^{-1}b > 0$. In the other case x is *degenerate*.

Then, a first connection to the geometry of linear programs is built. Let P be a polyhedron $P = P^{=}(A,b) \subseteq \mathbb{R}^n$ with $\operatorname{rank}(A) = m < n$ and $x \in P$. Then, the following statements are equivalent.

(i) The point x is a vertex of P.

(ii) The vector x is a basic feasible solution, such that there is a basis $A_{.B}$ of A with $x_B = A_{.B}^{-1}b \geqslant 0$ and $x_N = 0$.

A vertex of $P^{=}(A,b)$ is uniquely connected to a basic feasible solution of $Ax = b, x \geqslant 0$. However, a basic feasible solution x can correspond to more than one basis. In this case x is degenerate. Reasons for degenerations are

- a redundant variable,
- a redundant inequality, or
- geometric reasons.

Degeneration caused by one of the first two reasons result from a redundant description of the polyhedron and can be fixed by deleting the corresponding variable or inequality. However, there are polyhedrons P where the degeneration is inherent in the geometry. See for example the top vertex of a pyramid over a square in $\mathbb{R}^3$, here, more than three hyperplanes meet at that vertex and thus cause a degeneration. An approach to deal with general degeneration is the ε-perturbation method, where the LP is expanded with a small perturbation. However, this approach is complicated in practice and thus not applied very often.

The idea of the simplex algorithm is to start from a basic feasible solution and determine a new basic feasible solution with a better objective function value. This step is called *pivoting* and changes the current basis and thus the current basic feasible solution.

Given an LP in standard form (2.1) with $\operatorname{rank}(A) = m$ and basis $A_{.B}$ of A. Then, $\hat{x}$ fulfills the system $Ax = b$ is equivalent to $\hat{x}_B = A_{.B}^{-1}b - A_{.B}^{-1}A_{.N}\hat{x}_N$. This term can be used to write the objective function value as

$$c^T\hat{x} = c_B^T A_{.B}^{-1}b + (c_N^T - c_B^T A_{.B}^{-1}A_{.N})\hat{x}_N,$$

where the term in parenthesis is called *relative cost* of $\hat{x}$. Having this, the *optimality criterion* of the simplex algorithm is described as follows: Given an LP in standard form (2.1) and a feasible basis $A_{.B}$ of A, then, if the relative cost holds

$$\bar{c}^T = c_N^T - c_B^T A_{.B}^{-1}A_{.N} \geqslant 0,$$

the corresponding basic feasible solution x, with $x_B = A_{.B}^{-1}b$ and $x_N = 0$ is optimal.

Finally, the pivoting step is formalized. Given an LP in standard form (2.1), a feasible basis $A_{.B}$ of A and the corresponding basic feasible solution x. We denote $\bar{A} := A_{.B}^{-1}A_{.N}$, $\bar{b} := A_{.B}^{-1}b$, and the relative cost $\bar{c}^T := c_N^T - c_B^T A_{.B}^{-1}A_{.N}$. Let $q_s \in N$ be an index such that $\bar{c}_s < 0$, then the following hold

(i) If $\bar{A}_{.s} \leqslant 0$, then the LP is unbounded.

(ii) If $\bar{A}_{\cdot s} \nleqslant 0$, then $A_{\cdot B'}$ determined by

$$\begin{aligned} \lambda_0 &:= \min_{i=1,\dots,m} \left\{ \frac{\bar{b}_i}{\bar{a}_{is}} : \bar{a}_{is} > 0 \right\}, \\ r &\in \left\{ i \in \{1,\dots,m\} : \frac{\bar{b}_i}{\bar{a}_{is}} = \lambda_0 \right\}, \\ B' &= (p_1,\dots,p_{r-1},q_s,p_{r+1},\dots,p_m), \end{aligned}$$

is a feasible basis with the corresponding basic feasible solution x', such that $c^T x' \leqslant c^T x$. In case $A_{\cdot B}$ is not degenerate, we have $c^T x' < c^T x$.

Note, that $A_{\cdot B'}^{-1}$ can be obtained from $A_{\cdot B}^{-1}$ by elementary row operations. Formally, this means $A_{\cdot B'}^{-1} = EA_{\cdot B}^{-1}$, where E is the $(m \times m)$ elementary matrix, that comprises the necessary elementary row operations and is defined as:

$$E = \begin{pmatrix} 1 & & & \eta_1 & & & \\ & \ddots & & \vdots & & & \\ & & 1 & \eta_{r-1} & & & \\ & & & \eta_r & & & \\ & & & \eta_{r+1} & 1 & & \\ & & & \vdots & & \ddots & \\ & & & \eta_m & & & 1 \end{pmatrix},$$

with

$$\begin{aligned} \eta_r &:= \frac{1}{\bar{a}_{rs}} \\ \eta_i &:= \frac{\bar{a}_{is}}{\bar{a}_{rs}} \quad i \in \{1,\dots,m\} \backslash \{r\}. \end{aligned}$$

The matrix element $\bar{a}_{rs}$ is called *pivot element*.

Putting everything together, the basic version of the simplex algorithm can be realized, see Algorithm 1 for a formal description. For an LP with a nonempty underlying polyhedron and only nondegenerate bases, the simplex algorithm terminates after a finite number of steps. Each basis has a better objective function value than the one before and the polyhedron has a finite amount of vertices, thus the sequence of bases is finite. At the last generated basis the optimum is found or unboundedness is determined.

To complete the simplex algorithm we formalize *phase I*, which gets an initial feasible basis and the corresponding basic feasible solution. For the given system of equations $Ax = b,\ x \geqslant 0$, we multiply every row $i \in \{1,\dots,m\}$, where $b_i < 0$ by -1, this does not change the underlying polyhedron. Hence, we can assume $b \geqslant 0$ in the following. Phase I introduces *artificial variables* y to the LP (5) to constitute a basic feasible solution. That means, the following auxiliary program is considered. We set $D := (A,\mathbb{I})$, where $\mathbb{I}$ is the identity matrix, then the auxiliary program is

$$\begin{aligned} \max \quad & \mathbb{1}^T Ax \\ \text{s.t.} \quad & D \begin{pmatrix} x \\ y \end{pmatrix} = b \\ & x, y \geqslant 0. \end{aligned} \tag{9}$$

Obviously, $B = (n+1,\dots,n+m)$ is a feasible basis of D and we can directly apply the

basic simplex version with the aim to obtain a feasible basis to the original problem (5). This phase I simplex algorithm is given by Algorithm 2. Having a feasible basis for (5) the basic simplex version is called a second time. This time the original objective function is used. Thus, the basic simplex version is called *phase II*. Then, the final two-phase simplex algorithm is given by Algorithm 2 and Algorithm 1. Even though the two-phase simplex is very efficient, it is not a polynomial time algorithm, since there are polyhedra with an exponential number of vertices in the problem size and all of them have to be enumerated. Nevertheless, KHACHIYAN proved that linear programming is in class $\mathcal{P}$ (see Section 2.7) by introducing the *ellipsoid method* in 1979, see KHACHIYAN [Kha79].

Algorithm 1: The basic version of the simplex algorithm.

Input : $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$ with $m < n$, $\operatorname{rank}(A) = m$, column index vectors $B = (p_1, \ldots, p_m) \in \{1, \ldots, n\}^m$, such that $A_{.B}$ is a feasible basis of A and $N = (q_1, \ldots, q_{n-m})$ with $B \cap N = \varnothing$, $B \cup N = \{1, \ldots, n\}$.

Output : An optimal solution of the LP (5) if it exists or the statement, that the problem is unbounded.

1 **if** *it has not been initialized* **then**
2 compute $A_{.B}^{-1}$,
3 $\bar{A} = A_{.B}^{-1} A_{.N}$,
4 $\bar{b} = A_{.B}^{-1} b$,
5 $\bar{c}^T = c_N^T - c_B^T A_{.B}^{-1} A_{.N}$,
6 $c_0 = c_B^T \bar{b}$;
7 opt $:= false$, unbounded $:= false$;
8 **while** $\neg$opt *and* $\neg$unbounded **do**
9 **if** $\bar{c}_i \geqslant 0 \; \forall i = 1, \ldots, n - m$ **then**
10 opt $:= true$;
11 print x with $x_B = \bar{b}, x_N = 0$ and $z_0 = c^T x = c_B^T \bar{b}$;
12 STOP;
13 **else**
`//Determination of a pivot column`
14 choose any $s \in \{1, \ldots, n - m\}$ with $\bar{c}_s < 0$;
15 **if** $\bar{a}_{is} \leqslant 0 \; \forall i \in \{1, \ldots, m\}$ **then**
16 unbounded $:= true$;
17 STOP;
18 **else**
19 find $\lambda_0 := \min\left\{ \frac{\bar{b}_i}{\bar{a}_{is}} : \bar{a}_{is} > 0, i = 1, \ldots, m \right\} = \frac{\bar{b}_r}{\bar{a}_{rs}}$;
`//pivot on` $\bar{a}_{rs}$
20 $B' := (p_1, \ldots, p_{r-1}, q_s, p_{r+1}, \ldots, p_m)$;
21 $N' := (q_1, \ldots, q_{s-1}, p_r, q_{s+1}, \ldots, q_{n-m})$;
22 $A_{.B'}^{-1} := E A_{.B}^{-1}$;
23 update $\bar{A}$, $\bar{b}$, $\bar{c}$, and c_0;

Two final notes on the simplex algorithm. Firstly, there are different variants of how to choose the column to enter the basis and how to resolve ties in line 19 of Algorithm 1, which determines the row and thus the variable to leave the basis. For those pivot selection strategies we refer to [Grö09] and [Pap17].

Secondly, in case of degeneration a sequence of degenerate bases can repeat during execution of the simplex algorithm. This process is called *cycling*. Even if degeneration occasionally occurs in practice it usually does not cause cycling, thus anticycling strategies are often not implemented [Grö09].

Algorithm 2: The phase I simplex algorithm.

Input : $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, $b \geqslant 0$.

Output :

(a) $P^=(A, b) = \varnothing$ or

(b) $P^=(A, b) = \{x\}$, or

(c) Return $I \subseteq \{1, \ldots, m\}$, $B = (p_1, \ldots, p_k)$, where it holds: With $A' := A_{I\cdot}$, $b' := b_I$, we have $P^=(A', b') \neq \varnothing$, $\operatorname{rank}(A') = |I| = k$, $k < n$, $P^=(A', b') = P^=(A, b)$ and A'_B is a feasible basis of A'.

```
1  introduce an artificial basis as described in (9);
2  call Simplex Algorithm 1 with (9);
3  if 1^T Ax < 1^T b then
4  |  print case (a);
5  |  STOP;
6  else
   |  //1^T Ax = 1^T b
7  |  if B ∩ {n+1,...,n+m} = ∅ then
8  |  |  if N ∩ {1,...,n} = ∅ then
9  |  |  |  print case (b);                                  /* m = n */
10 |  |  |  STOP;
11 |  |  else
12 |  |  |  I = {1,...,m};
13 |  |  |  print case (c);                                  /* m < n, k = m */
14 |  |  |  STOP;
15 |  else
   |  |  //basis contains artificial variables
16 |  |  if artificial variables can be driven out of the basis then
17 |  |  |  drive out the artificial variables and goto line 7;
18 |  |  else
19 |  |  |  B ∩ {n+1,...,n+m} = {p_1,...,p_t};
20 |  |  |  omit the rows D_1·,...,D_t·;
21 |  |  |  I = {t+1,...,m};
22 |  |  |  B = {p_{t+1},...,p_m};
23 |  |  |  k := m − t;
24 |  |  |  if k = n then
25 |  |  |  |  print case (b);
26 |  |  |  |  STOP;
27 |  |  |  else
28 |  |  |  |  print case (c) STOP;
```

2.3 Branch-and-Cut

The two algorithmic principles for solving MIPs in practice are currently the LP based *branch-and-bound* and the *cutting plane method*. The idea of the branch-and-bound method goes back to LAND and DOIG [Lan60]. A few years later DAKIN [Dak65] proposed an algorithm adapted from this idea. The branch-and-bound method is a systematic enumeration of candidate solutions following the divide-and-conquer paradigm. The *branching step* splits the problem and the solution space into smaller sub-problems, which can be solved more efficiently. By recursively applying this step to the sub-problems a *branching tree* (or *search tree*) is created. Each node corresponds to a created sub-problem, while the root node stands for the original (LP relaxed) problem. To deal with fractional variables when solving a MIP of the form (1) we basically use *variable branching* of the form

$$P_1 := P^0 \cap \{(x, y) : x_j \leqslant z\} \text{ and } P_2 := P^0 \cap \{(x, y) : x_j \geqslant z + 1\},$$

where P_1 and P_2 denote the linear relaxation of the solution spaces of the new created sub-problems, $j \in \{1, \ldots, n\}$ and $z \in \mathbb{Z}$. The corresponding integer sets to P_1 and P_2 are denoted by S_1 and S_2. Note, that we have $S_1 \cup S_2 = S$, where S is the integer set corresponding to P^0 (the solution space of the parent node). For further branching strategies we refer to CONFORTI et al. [Con14].

Before enumerating the candidate solutions of a branch, the branch is checked against estimated lower and upper bounds on the optimal solution. In case the branch cannot produce a better solution than the best one found so far, the whole branch is discarded. This step is called *bounding step* and prevents a complete enumeration of the search tree. Upper bounds are given by integral feasible solutions of sub-problems or provided by heuristics. Let (x_i, y_i) be an optimal solution of a sub-problem and z_i its objective function value with $i \in \{1,2\}$. If x_i is integral ($(x_i, y_i) \in S_i$), (x_i, y_i) is an optimal solution of the corresponding MIP of the sub-problem and a feasible solution to the original MIP. Since $S_i \subseteq S$, if follows

$$\min\{c^T x + h^T y : (x, y) \in S\} \leqslant z_i.$$

Lower bounds are provided by optimal solutions of LP relaxations. For the root node, which represents the LP relaxation of the original MIP, let (x^0, y^0) be the optimal solution, then we have

$$c^T x^0 + h^T y^0 \leqslant \min\{c^T x + h^T y : (x, y) \in S\}.$$

Thus, for any other node, if the current LP objective value is not smaller than the global upper bound found so far, neither the current node with its solution sub-space nor any possible child node to the current node can contain an optimal solution. Thus, the node in the branching tree can be pruned.

There are choices in the branch-and-bound method, that are left open, as branching strategy, node selection strategy, heuristics to produce good upper bounds and the formulation aspect to get good lower bounds. The formulation aspect deals with formulating the MIP in such a way that the objective value of the MIP and the one of its LP relaxation are in close proximity and hence, produce tight lower bounds. Since $\min\{c^T x + h^T y : (x, y) \in S\} = \min\{c^T x + h^T y : (x, y) \in \text{conv}(S)\}$, where $\text{conv}(S)$ denotes the convex hull of the mixed integer set S, it is obvious, that there is an LP with the same objective function value as the given MIP. Thus, an LP solver can produce the optimal solution to the given MIP, when applied to the linear description of $\text{conv}(S)$. Usually, this description is not

given. For more details on the mentioned aspects we refer to CONFORTI et al. [Con14].

One way to produce tight lower bounds is by generating *cutting planes* or simply *cuts*. A cutting plane is an inequality $\alpha^T x + \gamma^T y \geqslant \beta$ that is satisfied by every point in the mixed integer set S, but there are points $(\hat{x}, \hat{y})$ in the linear relaxation P^0 such that $\alpha^T \hat{x} + \gamma^T \hat{y} < \beta$. We say that the cutting plane is *separating* $(\hat{x}, \hat{y})$ from P^0. Thus, a cutting plane can be used to separate an infeasible fractional portion from the LP relaxation of a MIP. We define

$$P^1 := P^0 \cap \{(x, y) : \alpha^T x + \gamma^T y \geqslant \beta\}.$$

Since $S \subseteq P^1 \subset P^0$, the LP relaxation of MIP (1) based on P^1 is stronger than the LP relaxation (4), in the sense that its objective value is at least as good a lower bound as the objective value of (4).

The cutting plane method dates back to the 1950s, where DANTZIG, FULKERSON and JOHNSON [Dan54] discussed the loop conditions of a 49-city traveling salesperson problem instance, which are nowadays known as the subtour elimination constraints. Without knowing this work, GOMORY developed the first all-purpose cutting plane algorithm using fractional cuts [Gom58], the birth of the famous GOMORY fractional cuts (short Gomory cuts). He proved that his algorithm converges after a finite number of steps in case of a pure integer program with rational data.

The tightness of the lower bound is important for pruning the branching tree. Hence, the generation of cutting planes is combined with the branch-and-bound approach to tighten the lower bounds of the sub-problems. This leads to the *branch-and-cut* method. In the 1980s there was the beginning of combining branch-and-bound and cutting planes to branch-and-cut, see for example PADBERG and RINALDI [Pad87] and PADBERG and RINALDI [Pad91]. Nowadays it is a state-of-the-art method for solving mixed integer programs and implemented in modern commercial MIP solvers such as CPLEX [IBM17], GUROBI [Gur21], XPRESS [FIC21] and MINOS, as well as academic and open source codes such as SCIP [Zus21] and GLPK [GNU21]. In practice, not only one cut, but a bundle of cuts is added in rounds to the LP relaxation. Depending on their validity, cuts can be added globally to all nodes of the branch-and-bound tree or locally to a specific node and its descendants to tighten their LP relaxation. Over the years, several classes of general cutting planes have been developed in addition to Gomory cuts, as well as cuts that only serve for a specific problem class e.g., subtour elimination constraints. For more details we refer to overviews given by MARCHAND et al. [Mar02] and CORNUÉJOLS [Cor08].

When implementing a branch-and-cut algorithm, the following issues need special attention:

- Preprocessing (tighten and simplifying the formulation of the root LP before diving into the branch-and-cut, can also be applied to sub-problems)
- Heuristics (finding good upper bounds)
- Cutting plane generation (improving the lower bounds)
- Branching (there are different branching strategies and different ways to choose a variable or variable set for branching)
- Node selection (strategies can rely on the bound or on the position of the node in the tree)

It is not only the question of which procedure to choose concerning one of the above issues, but also which of the above issues to address first, in order to produce an efficient branch-and-bound algorithm. Suppose for example (x^0, y^0) is the optimal solution of the LP relaxation of a given MIP. In case x^0 contains fractional variables, it is not easy to

decide what is the best next step in the algorithm, e.g., generating cuts or branching, and on which variable? Clearly, all listed components are connected in any way and implementing an efficient branch-and-cut algorithm is a challenging task. For further details we again refer to Conforti et al. [Con14].

2.4 Second-Order Cone Programming

A second-order cone program is an optimization problem with linear constraints and the restriction, that the solution is contained in a *cone*, the *second-order cone*, which is also called *Lorentz* or *ice-cream cone*. The norm, that is associated with the second-order cone is the Euclidean norm.

The unit second-order cone of dimension k is defined as:

$$\mathcal{C}_k = \{(x,t) \in \mathbb{R}^k : x \in \mathbb{R}^{k-1}, t \in \mathbb{R}, \ \|x\|_2 \leqslant t\}. \tag{10}$$

Figure 1 visualizes the second-order cone in $\mathbb{R}^3$. The *second-order cone program* (SOCP) is defined as:

$$\begin{aligned} &\min\ f^T x \\ &\text{subject to}\ \|A_i x + b_i\|_2 \leqslant c_i^T x + d_i, \quad i = 1, \ldots, m \\ &\qquad\qquad\quad Fx = g, \end{aligned} \tag{11}$$

where $x \in \mathbb{R}^n$ is the optimization variable, $f \in \mathbb{R}^n$, $A_i \in \mathbb{R}^{n_i \times n}$, $b_i \in \mathbb{R}^{n_i}$, $c_i \in \mathbb{R}^n$, $d_i \in \mathbb{R}$, $F \in \mathbb{R}^{p \times n}$, and $g \in \mathbb{R}^p$. The constraint

$$\|A_i x + b_i\|_2 \leqslant c_i^T x + d_i$$

is called *second-order cone constraint of dimension* n_i and is an affine mapping of the unit second-order cone (10). The SOCP (11) is a convex optimization problem, since the objective function is convex and the constraints define a convex set of feasible solutions. SOCP can be solved very efficiently by *interior-point methods* also referred to as *barrier methods*. The optimization software CPLEX is used in this thesis. In CPLEX a SOCP has to be formulated in the following form (extracted from the CPLEX example ilosocpex1.cpp [IBM17])

$$\begin{aligned} &\min\ c_1^T x_1 + \cdots + c_r^T x_r \\ &\text{subject to}\ A_1 x_1 + \cdots + A_r x_r = b \\ &\qquad\qquad\quad x_i \in \mathcal{C}_{n_i}, \quad i = 1, \ldots, r, \end{aligned} \tag{12}$$

where x_i is a vector of length n_i ($x_i = (x_i[1], \ldots, x_i[n_i])$). The last constraint has to be written as

$$\begin{aligned} -x_i[1]^2 + x_i[2]^2 + \cdots + x_i[n_i]^2 &\leqslant 0, \\ x_i[1] &\geqslant 0, \\ x_i[2], \ldots, x_i[n_i] \text{ free variables.} & \end{aligned}$$

This SOCP is solved by the barrier optimizer of CPLEX, see Section 2.5 for more details on barrier methods.

For more information about SOCP and convex optimization, we refer to Boyd and Vandenberghe [Boy04], Lobo et al. [Lob98], and Nemirovski [Nem05].

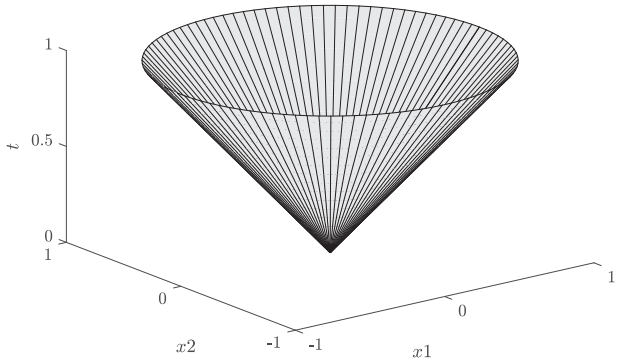

Figure 1: The boundary of the second-order cone in $\mathbb{R}^3$.

2.5 Interior Point Methods

Interior point methods (also called *barrier methods*) are a class of algorithms to solve linear and nonlinear convex optimization problems.

In 1984 KARMARKAR developed a new method to solve linear programs called *Karmarkar's algorithm* [Kar84]. This algorithm runs in polynomial time and is also efficient in practice. The idea is to traverse the interior of the underlying polyhedron to reach the optimal solution. Karmarkar's algorithm stimulated the development of interior point methods in the 90s. The resulting algorithms are competitive with the simplex algorithm [Grö09].

Interior point methods can solve convex optimization problems with twice differentiable objective and constraint functions of the following form:

$$\begin{aligned} \min \quad & f_0(x) \\ \text{subject to} \quad & f_i(x) \leqslant 0, \quad i = 1, \ldots, m, \\ & Ax = b, \end{aligned} \tag{13}$$

where $f_0, \ldots, f_m : \mathbb{R}^n \to \mathbb{R}$ are convex functions, and $A \in \mathbb{R}^{p \times n}$ with $\operatorname{rank} A = p < n$. We assume, that there is an optimal solution x^* with the corresponding objective function value p^*.

Some problems, which do not fulfill the form (13) and the assumption of differentiability, can be reformulated in the required form. Other convex optimization problems, such as SOCPs have to be handled by extensions of the barrier methods to problems with generalized inequalities. Generalized inequalities provide a paritial order in the $\mathbb{R}^n$ based on a cone. Given a proper cone $\mathcal{K}$ in $\mathbb{R}^n$ and $x, y \in \mathbb{R}^n$, the partial order is defined as $x \preceq_{\mathcal{K}} y$ if and only if $y - x \in \mathcal{K}$.

According to BOYD and VANDENBERGHE [Boy04], we present the idea of the barrier method for solving SOCPs of the following form:

$$\begin{aligned} \min \quad & f^T x \\ \text{subject to} \quad & \|A_i x + b_i\|_2 \leqslant c_i^T x + d_i, \quad i = 1, \ldots, m, \end{aligned} \tag{14}$$

where $A_i \in \mathbb{R}^{n_i \times n}$, $f \in \mathbb{R}^n$, $b_i \in \mathbb{R}^{n_i}$, $c_i \in \mathbb{R}^n$, and $d_i \in \mathbb{R}$.

At first a generalization of the logarithm function, which applies to the second-order cone, is needed. Having this, a logarithmic barrier function for (14) is defined as well as the central path, which leads through the interior of the cone to the optimal solution. The central path results from solving a sequence of unconstrained (or linear constrained in case

of (13)) minimization problems using *Newton's method*, see [Boy04] Section 9.5. A simple version of the iterative barrier method is given by Algorithm 3.

Given a proper cone $\mathcal{K} \subseteq \mathbb{R}^q$, we denote by $\operatorname{int} \mathcal{K}$ its interior. Then the function $\psi : \mathbb{R}^q \to \mathbb{R}$ is a *generalized logarithm* for $\mathcal{K}$ if ψ is concave, closed, twice continuously differentiable, its domain $\operatorname{dom} \psi = \operatorname{int} \mathcal{K}$, and $\nabla^2 \psi(y) \prec 0$, $y \in \operatorname{int} \mathcal{K}$. Additionally, there is a constant $\theta > 0$, such that for all $y \succ_{\mathcal{K}} 0$, and all $s > 0$ the equation $\psi(sy) = \psi(y) + \theta \log s$ holds. For the second-order cone in $\mathbb{R}^{n+1}$ we use the following generalized logarithm:

$$\psi(y) \coloneqq \log \left(y_{n+1}^2 - \sum_{i=1}^{n} y_i^2 \right).$$

It behaves like a logarithm along any ray in the cone.

Then, the *logarithmic barrier function* is defined as

$$\phi(x) \coloneqq - \sum_{i=1}^{m} \psi_i(-f_i(x)), \text{ with domain } \operatorname{dom} \phi = \{x : f_i(x) \prec_{\mathcal{K}_i} 0,\ i = 1, \ldots, m\}.$$

Thus, the corresponding logarithmic barrier function for (14) is given by

$$\phi(x) \coloneqq - \sum_{i=1}^{m} \log((c_i^T x + d_i)^2 - \|A_i x + b_i\|_2^2), \tag{15}$$

with the domain $\operatorname{dom} \phi = \{x : \|A_i x + b_i\|_2 < c_i^T x + d_i,\ i = 1, \ldots, m\}$.

Then, the minimization of $t f^T x + \phi(x)$ is an approximation of (14):

$$\min \quad t f^T x - \sum_{i=1}^{m} \log((c_i^T x + d_i)^2 - \|A_i x + b_i\|_2^2). \tag{16}$$

The accuracy of the approximation depends on t, that means the quality of the approximation improves as t grows. We can use Newton's method to solve (16). The solutions $x^*(t)$, for $t \geqslant 0$ are called *central points* assuming the minimum exists and is unique. The central points form the *central path*, which leads to the optimal solution of (14). Points on the central path are characterized by the *optimality condition* $t f + \nabla \phi(x^*(t)) = 0$, with

$$\nabla \phi(x) = -2 \sum_{i=1}^{m} \frac{1}{(c_i^T x + d_i)^2 - \|A_i x + b_i\|_2^2} \left((c_i^T x + d_i) c_i - A_i^T (A_i x + b_i) \right).$$

Geometrically speaking, the optimality condition is a tanget to the contour line of ϕ at the point $x^*(t)$. Then it follows, that the point

$$z_i^*(t) = -\frac{2}{t \alpha_i} (A_i x^*(t) + b_i), \quad w_i^*(t) = \frac{2}{t \alpha_i} (c_i^T x^*(t) + d_i), \quad i = 1, \ldots, m,$$

with $\alpha_i = (c^T x^*(t) + d_i)^2 - \|A_i x^*(t) + b_i\|_2^2$, is strictly feasible in the dual (regarding (14))

problem:

$$\begin{aligned} \max \ & -\sum_{i=1}^{m}(b_i^T z_i + d_i w_i) \\ \text{subject to} \ & \sum_{i=1}^{m}(A_i^T z_i + c_i w_i) = f \\ & \|z_i\|_2 \leqslant w_i, \quad i = 1, \ldots, m. \end{aligned}$$

In particular, the duality gap (difference between primal und dual objective function value) can be computed as

$$\sum_{i=1}^{m}\left((A_i x^*(t) + b_i)^T z_i^*(t) + (c_i^T x^*(t) + d_i) w_i^*(t)\right) = \frac{2m}{t}. \tag{17}$$

This result confirms, that $x^*(t)$ converges to an optimal point as $t \to \infty$. The iterative barrier algorithm is described in Algorithm 3.

Algorithm 3: A basic version of the barrier method.

Input : A SOCP of the form (14), $x \in \mathbb{R}^n$ strictly feasible, $t := t^{(0)} > 0$, $\mu > 1$, a tolerance $\varepsilon > 0$.

```
1  repeat
       //Centering step
2      Compute x*(t) by minimizing t f^T x̂ + φ(x̂), starting at x;
       //Update
3      x := x*(t);
       //Stopping criterion
4      if 2m/t < ε then
5          print x;
6          STOP;
7      else
8          continue;
       //Increase t
9      t := μ t;
10 until;
```

The barrier method is an iterative algorithm, where the parameter t is increased in every iteration, since simply setting $t = \frac{2m}{\varepsilon}$ (see duality gap (17)) for a given accuracy ε and solving the unconstrained problem

$$\min \ (\frac{2m}{\varepsilon}) f^T \hat{x} + \phi(\hat{x}),$$

using Newton's method, only works well for small problems, good starting points x and ε not too small. Thus, a sequence of unconstrained minimization problems is solved (see line 2) in the way that the last solution point found serves as the starting point for the next problem (see line 3). The values of t increase (see line 9) until $t > \frac{2m}{\varepsilon}$ (see line 4), which guarantees that the solution found is ε-suboptimal.

The parameter μ is the factor by which t is increased in each iteration. The choice of μ is a trade-off in the number of barrier iterations (line 2) and in the number of Newton steps

(number of Newton iterations for solving one unconstrained minimization problem). Small values of μ (i.e., near one) result in many barrier iterations and only few Newton steps per iteration, because in this case x is a very good starting point. For large values we have the opposite situation. However, for values of μ from around 3 to 100 the two effects nearly cancel and the total number of Newton steps is more or less constant. Thus, values from around 10 to 20 seem to work well in practice.

The choice of $t^{(0)}$ (the initial value of t) is another important issue. If $t^{(0)}$ is too large the first barrier iteration will require to many Newton iterations. If $t^{(0)}$ is too small, this will result in extra barrier iterations and possibly too many Newton steps in the first iteration. An appropriate choice is to choose $t^{(0)}$ such that $\frac{2m}{t^{(0)}}$ is of the same order as $f^T x^{(0)} - p^*$, since $\frac{2m}{t^{(0)}}$ is the duality gap in the first barrier iteration.

At the beginning, the barrier algorithm requires a strictly feasible starting point $x^{(0)}$. In case such a point is not known, there is a *basic phase I method*, that computes the required point (or detects infeasibility). The subsequent barrier method is called *phase II*. For the basic phase I method a set of inequalities and equalities are considered:

$$f_i(x) \leqslant 0,\ i = 1, \ldots, m \quad \text{and} \quad Ax = b,\ x \in \mathbb{R}^n, \tag{18}$$

where $f_i : \mathbb{R}^n \to \mathbb{R}$ are convex functions with continuous second derivatives. To find a strictly feasible solution to (18) the following auxilary problem is solved:

$$\begin{aligned} \min\ & s \\ \text{subject to}\ & f_i(x) \leqslant s, \quad i = 1, \ldots, m \\ & Ax = b \\ & x \in \mathbb{R}^n,\ s \in \mathbb{R}. \end{aligned} \tag{19}$$

We assume, that we can find a solution $x^{(0)}$, with $Ax^{(0)} = b$ and $x^{(0)} \in \operatorname{dom} f_1 \cap \cdots \cap f_m$ (e.g., with a solver for linear equations). For s we can choose any value larger than $\max_{i=1,\ldots,m} f_i(x^{(0)})$. Thus, we obtain a strictly feasible starting point $(x^{(0)}, s)$ for (19) and can apply the barrier method as phase I. Then, we distinguish between the following three cases. Let $\bar{p}^*$ be the optimal value.

1. $\bar{p}^* < 0$: For a feasible solution (x, s), with $s < 0$ we have, that x is strictly feasible ($f_i(x) < 0$) for (18). Thus, there is no need for a high accuracy in optimization, as soon as $s < 0$ the optimization procedure can be aborted and the corresponding x serves as the required starting point $x^{(0)}$.
2. $\bar{p}^* > 0$: Here, (18) is infeasible. Again, the optimization can be terminated, when a dual feasible point with positive objective function value is found, which proves $\bar{p}^* > 0$.
3. $\bar{p}^* = 0$: In case, the minimum is achieved in x^* and $s^* = 0$, then (18) is feasible, but not strictly feasible. Otherwise, (18) is infeasible. However, in practice the optimization procedure terminates with the conclusion $|\bar{p}^*| < \varepsilon$, for some small positive ε. Thus, we have either feasibility ($f_i(x) \leqslant -\varepsilon$) or infeasibility ($f_i(x) \leqslant \varepsilon$) of (18).

There are variations on the basic phase I method and there are other phase I methods as well. For more information about phase I methods, interior point methods in general and convex optimization we refer to Boyd and Vandenberghe [Boy04]. This reference also includes the polynomial worst case complexity bound, see [Boy04], Section 11.5.

2.6 The Basics of Graph Theory

The basic concepts of graph theory are provided according to GRÖTSCHEL et al. [Grö93]. An *undirected graph* G is defined as an ordered tuple $G = (V, E)$, where V is a finite nonempty set of *vertices* (*nodes*) and E is a finite nonempty family of unordered pairs from V, which we call *edges*. Given an edge e, we call $i, j \in V$ the *endpoints* of e and write $e = i, j$ instead of $e = \{i, j\}$. The edge is said to be *incident* to its endpoints. We assume that the two endpoints of an edge are distinct. Two vertices that are joined by an edge are called *adjacent* or *neighbors*. The set of edges having a vertex $v \in V$ as one of their endpoints is denoted as $\delta(v)$. Thus, $|\delta(v)|$ is the *degree* of $v \in V$. Two edges are called *parallel* if they have the same endpoints. A graph that contains no parallel edges is called *simple*. We only consider simple graphs.

The *order* of G is the number of vertices $|V|$ and the size of G is the number of edges $|E|$. A simple graph is called *complete* if every two of its vertices are endpoints of an edge. The complete graph of order n is denoted by K_n.

If W is a set of vertices in $G = (V, E)$, then the subgraph of G induced by a vertex set $W \subseteq V$ is called $G[W]$. The subgraph $G[W]$ contains only vertices from W and edges, where both endpoints are in W.

A *directed graph* or *digraph* is a graph where the edges have orientations. A digraph $D = (V, A)$ consists of a finite nonempty set V of *vertices* (*nodes*) and a finite nonempty family A of *arcs*. Every arc a consists of an ordered pair $a = (i, j)$ of vertices called *endpoints*; i is the *starting endpoint* (or *tail*) and j is the *terminal endpoint* (or *head*). For an arc $a = (i, j)$ we also say that *a leaves i* and *enters j* and that i is a *predecessor* of j and that j is a *successor* of i. *Parallel arcs* share the same head and tail. Two arcs are *anti-parallel* if they have the same endpoints, but a different direction (head and tail are interchanged). We do not allow parallel arcs in digraphs.

Given a digraph $D = (V, A)$, then the (undirected) graph $G = (V, E)$ having an edge i, j where D has an arc (i, j) or (j, i) is called the *underlying graph* of D. For $v \in V$ the set of arcs having v as starting (terminal) endpoint is denoted as $\delta^+(v)$ ($\delta^-(v)$). With $\delta(v) := \delta^+(v) \cup \delta^-(v)$ the numbers $|\delta^+(v)|$, $|\delta^-(v)|$, and $|\delta(v)|$ are called the *outdegree*, *indegree*, and *degree* of v, respectively.

Walks, Paths, Tours, Cycles

Given a graph or digraph, a *walk* (*directed walk* or *diwalk*) W is a finite sequence of vertices of the following form $W = v_0, v_1, v_2, \ldots v_k$, $k \geqslant 0$, such that for each v_i, $i = 1, 2, \ldots, k$ there is an edge (arc) $e_i = v_{i-1}, v_i$ ($a_i = (v_{i-1}, v_i)$). The vertices v_0 and v_k are called *origin* and *destination*, respectively. The length of the walk is k.

A walk (diwalk) in which all vertices are distinct is called a *path* (*directed path* or *dipath*). Given a walk (diwalk) W, the origin $s \in V$, and the destination $t \in V$ of W, then W is called s, t-walk ((s, t)-diwalk). A *closed* walk (diwalk) has identical origin and destination and nonzero length. A closed path (dipath) is called a *cycle* (*directed cycle* or *dicycle*, when there is no confusion we also call it a cycle).

A walk (diwalk) that traverses every edge (arc) of a graph (digraph) exactly once is said to be an *Eulerian path* (allowing for revisiting vertices). A closed Eulerian path is an *Eulerian cycle* or *Eulerian tour*.

A path (dipath) that visits every vertex of a graph (digraph) exactly once is called a *Hamiltonian path*. The closed version is a *Hamiltonian cycle* or *dicycle*, which is often

called *Hamiltonian tour*. A graph (digraph) that contains a Hamiltonian cycle (dicycle) is *Hamiltonian*. The Hamiltonian path problem and the Hamiltonian cycle problem are problems of determining whether a given graph or digraph contains a Hamiltonian path or cycle.

Further references for graph theory are for instance KORTE and VYGEN [Kor00] and DIESTEL [Die17].

The Basics of Network Flow Theory

A *transportation network* (or simply *network*) is a digraph with additional properties. Here, the arcs carry some kind of flow like water, gas, electricity, data, et cetera. To model such situations a network N is defined as the quadruple $N = (G, s, t, u)$, where $G = (V, A)$ is a digraph, $s, t \in V$ are two fixed vertices and $u : A \to \mathbb{R}_+$ is a capacity function on G. The vertex s is called *source*, at this node flow can enter the network. The vertex t is called *target* or *sink*, at this node flow can leave the network. A *flow* is a function $f : A \to \mathbb{R}_+$, that fulfills the following conditions.

(i) Capacity constraints: For every arc the amount of flow on that arc cannot exceed the given capacity:

$$f(e) \leqslant u(e), \quad \forall e \in A.$$

(ii) Flow conservation constraints: For every vertex $v \in V$, except for s, t, the amount of flow that enters v has to be equal to the amount that leaves v:

$$\sum_{e \in \delta^-(v)} f(e) = \sum_{e \in \delta^+(v)} f(e), \quad \forall v \in V \backslash \{s, t\}.$$

This flow is called an s, t-flow and the *total value* of f denoted by $|f|$ is defined as the net flow into the sink:

$$|f| = \sum_{e \in \delta^-(t)} f(e) - \sum_{e \in \delta^+(t)} f(e).$$

So far we considered a flow of a single *commodity*. However, real-world problems often involve *multi-commodity flows*, that are flows of multiple, often differentiated commodities, that are simultaneously shipped through the same resource network. Given a network $N = (G,u)$ with a digraph $G = (V, A)$ and a capacity function $u : A \to \mathbb{R}_+$. There are k commodities, defined by triples (s_i, t_i, d_i), $i = 1, \ldots, k$, where s_i and t_i are the source and the sink of the i-th commodity and d_i is its demand. For each commodity there is a flow $f_i : A \to \{0, 1\}$, that has to satisfy the following constraints.

(i) Capacity constraints: For every arc the sum of all flows routed over that arc cannot exceed its capacity:

$$\sum_{i=1}^{k} f_i(e) \cdot d_i \leqslant u(e), \quad \forall e \in A.$$

(ii) Flow conservation constraints on intermediate nodes: For every commodity the amount of flow entering an intermediate (regarding the i-th commodity) node $v \in V$

is the same that leaves v:

$$\sum_{e \in \delta^+(v)} f_i(e) - \sum_{e \in \delta^-(v)} f_i(e) = 0, \quad \forall v \in V \backslash \{s_i, t_i\},\, i = 1, \ldots, k.$$

(iii) Flow conservation at the source: Each flow starts at its source node:

$$\sum_{v \in V} f_i(s_i, v) - \sum_{w \in V} f_i(w, s_i) = 1, \quad i = 1, \ldots, k.$$

(iv) Flow conservation at the target: Each flow ends at its target node:

$$\sum_{v \in V} f_i(v, t_i) - \sum_{w \in V} f_i(t_i, w) = 1, \quad i = 1, \ldots, k.$$

In case there is a cost function $c : A \to \mathbb{R}_+$ the additional objective function

$$\min \sum_{e \in A} \left(c(e) \sum_{i=1}^{k} f_i(e)\, d_i \right)$$

leads to the *minimum cost multi-commodity flow problem*. In the definitions above we used real-valued data, however, solving the problem and thus, describing it on the computer, requires the data to be rational.

There are applications, where multi-commodity flow problems are extended by time-dynamic components, e.g., time windows, time dependent arcs or flow variation over time. Then, the problem including the underlying network is dynamic, see for example time windows - a vertex can only be visited in a certain time interval. Considering the underlying network this can be seen as vertices that enter the network at a specific time and then leave the network at a specific time. To deal with the new temporal dimension classic static network flow models are no longer satisfactory, instead the concept of *network flows over time* (also called *dynamic flows*) is addressed, see Skutella [Sku09] for an overview. In the following, we describe the *time-expanded network* as a tool from the dynamic flows concept. The idea of a time-expanded network is that the dynamic flow problem can be reduced to a static one. We provide the definition from [Sku09] and then present a version for moving vertices, that is suitable for the application described in Section 1.1.

Definition 2.3. Given a network $N = (G, u, c)$ with a digraph $G = (V, A)$, capacities u, non-negative integral transit times τ, and arc costs c. Let $T \in \mathbb{N}$ be the time horizon, then the corresponding time-expanded network $N^T = (G^T, u^T, c^T)$ with $G^T = (V^T, A^T)$ is defined as follows. For every vertex $v \in V$ a number of T copies are generated: $v_0, v_1, \ldots, v_{T-1}$, thus,

$$V^T := \{v_\theta : v \in V,\, \theta = 0, 1, \ldots, T-1\}.$$

For every arc $e = (v, w) \in A$, we create $T-\tau_e$ copies: $e_0, e_1, \ldots, e_{T-1-\tau_e}$, where $e_\theta = (v_\theta, w_{\theta+\tau_e})$ with capacity $u^T_{e_\theta} := u_e$ and cost $c_{e_\theta} := c_e$. Additionally, A^T contains *holdover arcs* $(v_\theta, v_{\theta+1})$ for all $v \in V$ and $\theta = 0, 1, \ldots, T-2$. Holdover arcs get infinite capacity and zero cost. Finally, A^T is summarized as

$$\begin{aligned} A^T := &\{e_\theta = (v_\theta, w_{\theta+\tau_e}) : e = (v, w) \in A,\ \theta = 0, 1, \ldots, T-1-\tau_e\} \\ &\cup \{(v_\theta, v_{\theta+1}) : v \in V,\, \theta = 0, 1, \ldots, T-2\}. \end{aligned}$$

Further references about network flows are Ford and Fulkerson [For62], Ahuja et al. [Ahu93] and Grötschel et al. [Grö93].

2.7 Computational Complexity Theory

This section addresses the complexity of problems and describes the classes $\mathcal{P}$ and $\mathcal{NP}$. It is based on the specifications provided by GRÖTSCHEL et al. [Grö93].

To formalize the *size* of a problem instance an *encoding scheme* is needed. An encoding scheme represents each instance of a *problem* and each solution by a string of symbols. The instance string is also called *input* and the solution string *output*, respectively. The *encoding length* or *input size* of a problem instance is defined as the length of the input string. Most of the standard encoding schemes are equivalent regarding our purposes. Thus, to present general results about time complexity it is not necessary to decide for a particular encoding scheme.

Given an encoding scheme and an algorithmic model (e.g., Turing machine), then the *time complexity function* $f : \mathbb{N} \to \mathbb{N}$ of an algorithm describes the maximum time $f(n)$ needed to solve any instance of a problem with an input size of at most $n \in \mathbb{N}$. A *polynomial time algorithm* is defined as an algorithm, that satisfies $f(n) \leqslant p(n)$ for all $n \in \mathbb{N}$, and some polynomial p.

The big-$\mathcal{O}$ notation is used to classify algorithms. It describes how the run time of an algorithm grows as the input size of the problem grows. More formally, given two time complexity functions f, g, then $f(n)$ is $\mathcal{O}(g(n))$, if there are positive integer constants c and N such that

$$f(n) \leqslant c\, g(n) \quad \text{for all } n \geqslant N.$$

In order to decide if a problem is *hard* or not, it is convenient to only analyze *decision problems*. A decision problem is a problem that can be defined as a yes-no question of the input values. The Hamiltonian cycle problem is a decision problem. The class of all decision problems that can be solved by a *deterministic polynomial time algorithm* is called the class $\mathcal{P}$. A deterministic algorithm always generates the same output and the underlying machine performs the same sequence of states when invoked with the same input. The class $\mathcal{NP}$ contains all decision problems that can be solved by a *non-deterministic polynomial time algorithm*. These are algorithms, which allow *guesses* and the correctness of the guess can be verified in polynomial time. Obviously, the class $\mathcal{P}$ is contained in $\mathcal{NP}$, since if a problem can be solved in polynomial time, then its solution can also be verified in polynomial time, simply by solving the problem.

The hardest problems in $\mathcal{NP}$ are called $\mathcal{NP}$-complete problems. The Hamiltonian cycle problem is $\mathcal{NP}$-complete, see GAREY and JOHNSON [Gar79]. Formally, a decision problem is $\mathcal{NP}$-complete if it is in $\mathcal{NP}$ and if every other problem in $\mathcal{NP}$ can be *polynomially transformed* to it. Given two decision problems P and P' and a fixed encoding scheme, then a polynomial transformation is an algorithm that generates an encoded instance of P' from an encoded instance of P, such that: For every instance p of P, the answer to p is 'yes' if and only if the answer to p' is 'yes', where $p' \in P'$ is the transformation of p. It can be seen, that if there is a polynomial algorithm, that can solve P', then also P can be solved in polynomial time by transforming any instance of P to an instance of P' and applying its polynomial solution algorithm. Thus, if any $\mathcal{NP}$-complete problem can be solved in polynomial time, then all $\mathcal{NP}$-complete problems can be solved in polynomial time, which will conclude to $\mathcal{P} = \mathcal{NP}$. However, there is no known algorithm to solve an $\mathcal{NP}$-complete problem in polynomial time. The question about the existence of such an algorithm is called the $\mathcal{P}$ *versus* $\mathcal{NP}$ *problem*. This is still one of the major open problems at the boundary of mathematics and computer sciences.

Usually, optimization problems are not decision problems, however, they can be associated with decision problems in a natural way. Assume for instance a maximization (minimization) linear program. For the associated decision problem an additional input $Q \in \mathbb{Q}$ is introduced and one asks if there is a feasible solution with an objective function value of at least (at most) Q. We call a problem $\mathcal{NP}$-hard, if the associated decision problem is $\mathcal{NP}$-complete.

2.8 Interval Arithmetic

Interval arithmetic is a technique to deal with numerical errors. According to HAYES [Hay03] it does not improve the accuracy of a calculation, but it provides a certificate of accuracy for every result. A result of an ordinary computation is a single number, a point on the real line lying at some unknown distance from the true answer. An interval computation provides upper and lower bounds of the solution, which are guaranteed to enclose the exact true answer. Basic arithmetic operations can be performed on intervals. Let $\circ$ be any of the operations $+$, $-$, $\times$, and $\div$, then the corresponding interval operation for two intervals $[\underline{x}, \overline{x}]$ and $[\underline{y}, \overline{y}]$ is defined as follows:

$$[\underline{x}, \overline{x}] \circ [\underline{y}, \overline{y}] = [\min(\underline{x} \circ \underline{y}, \underline{x} \circ \overline{y}, \overline{x} \circ \underline{y}, \overline{x} \circ \overline{y}), \max(\underline{x} \circ \underline{y}, \underline{x} \circ \overline{y}, \overline{x} \circ \underline{y}, \overline{x} \circ \overline{y})].$$

A further reference to interval arithmetic is JAULIN et al. [Jau01].

As a feasibility checking method embedded in a branch-and-bound framework we use *interval constraint propagation* (ICP), which is a subarea of interval arithmetic. The basic idea of ICP is described according to JAULIN et al. [Jau02]. Consider the variables a, d, and t with their prior feasible domains (intervals)

$$\begin{aligned} a &\in [\underline{a}, \overline{a}] = [1, 6], \\ d &\in [\underline{d}, \overline{d}] = [1, 5], \\ t &\in [\underline{t}, \overline{t}] = [3, 8]. \end{aligned}$$

Assume that all three variables are linked by the constraint $a = d + t$. More formally this constraint is defined as

$$ADD = \{(a, d, t) \in \mathbb{R}^3 : a = d + t\}.$$

ADD is the contraction of the prior feasible intervals for the three variables by removing inconsistent values. Thus, the contracted feasible domains for the variables can be obtained by

$$\begin{aligned} [a] &:= [a] \cap ([d] + [t]) = [1, 6] \cap ([1, 5] + [3, 8]) = [4, 6], \\ [d] &:= [d] \cap ([a] - [t]) = [1, 5] \cap ([1, 6] - [3, 8]) = [1, 3], \\ [t] &:= [t] \cap ([a] - [d]) = [3, 8] \cap ([1, 6] - [1, 5]) = [3, 5]. \end{aligned}$$

Other constraints can be defined as well, e.g., $MULT : z = x \times y$ or $EXP : y = \exp(x)$.

Assume two fixed targets T_1 and T_2 with domain intervals (visibility window) $[\underline{t}_1, \overline{t}_1]$ and $[\underline{t}_2, \overline{t}_2]$, respectively and a salesperson, that is moving from T_1 to T_2. Further assume the above described variable d being the departure time of the salesperson in T_1 and the variable a the arrival time in T_2. The variable t is the travel time from T_1 to T_2. The travel time includes moving from the position of T_1 to the position of T_2 plus possible waiting time, such that $a = d + t$. Then, the time feasibility can be checked by contracting the feasible domains of the variables. Given an ordered sequence of fixed targets, we can decide by recursively executing the proposed procedure, if there is a possible arrival interval at

the last target of the sequence, then the sequence is time feasible and thus, a feasible tour, otherwise not. Then, the time feasibility of a predefined sequence of targets can be checked by recursively applying the described steps.

3 Mathematical Models with Time

In this chapter we introduce five different model approaches and present two model adaptations for infeasible instances and non-linear trajectories. Some basic notation is provided to formulate the MTSPMT as a mixed-integer optimization problem. We assume a finite time horizon $[0, T]$. The operating space is a square in the $\mathbb{R}^2$ with a side length $S \in \mathbb{R}_+$, but the following models may also be extended to the $\mathbb{R}^3$. Let $\mathcal{W} := \{1, \ldots, w\}$ be a set of salespersons. To simplify matters, all salespersons start their tour at the same depot location o, but the model can easily be extended to different depot locations. Let $\mathcal{V} := \{1, \ldots, n\}$ be a set of targets (customers), then $\mathcal{V}_o := \mathcal{V} \cup \{o\}$ and $\mathcal{A} \subseteq \mathcal{V}_o \times \mathcal{V}$ be a set of arcs. The length of an arc depends on the time the arc is traversed and varies over time, since the targets are moving. Thus, the distance for salesperson k traveling from target i to target j starting at a time in i and arriving at a time in j is given by the function $c_{i,j,k} : [0, T] \times [0, T] \to \mathbb{R}_+ \cup \{\infty\}$. Since each target $i \in \mathcal{V}$ is assigned a visibility time window $[\underline{t}_i, \overline{t}_i]$, we have

$$c_{i,j,k}(s,t) := \begin{cases} \infty, & \text{if and only if } \; s \notin [\underline{t}_i, \overline{t}_i] \text{ or } t \notin [\underline{t}_j, \overline{t}_j] \text{ or} \\ & (t-s)\,\overline{v} < \left\| p^j(t) - p^i(s) \right\|_2 \\ \left\| p^j(t) - p^i(s) \right\|_2, & \text{otherwise,} \end{cases} \tag{20}$$

where $p^i(s)$ and $p^j(t)$ are the respective locations of the targets at the times s and t and $\overline{v}$ is the maximum speed value of all salespersons. Waiting times are included in the traveling times, thus, the arrival time of a salesperson at a target is equal to its departure time at the same target. The visibility window of the depot o is the entire time horizon. All arcs with a finite length can be traveled with maximum speed $\overline{v}$ plus potential waiting time. The goal is to reach each target from $\mathcal{V}$ by exactly one salesperson such that the sum of all traveled distances of all salespersons is minimized.

3.1 A Time-Discrete Model

A time-discrete MILP formulation is presented in the sequel. The model consists of a multi-commodity flow formulation embedded in a time expanded network (see Definition 2.3, on page 24). Here, we have discretized the whole time horizon in time steps. For each time step there is a time layer with a copy of each target that is visible in this time step. The target copies are called snapshot targets. Let m be an integral number. The step size is defined by $\Delta := T/m$. Then the set of all time steps is $\mathcal{T} := \{0, \ldots, m\}$. Given a time step $\theta \in \mathcal{T}$, the corresponding time t_θ is given as $t_\theta = \Delta\theta$. For a vertex $v \in \mathcal{V}_o$ its visibility time window is given by an interval of time steps $[\underline{\tau}_v, \overline{\tau}_v] \subseteq \mathcal{T}$.

Then, the time-expanded network $\tilde{N} := (\tilde{\mathcal{V}}, \tilde{\mathcal{A}})$ is defined as follows. For every moving target v we create $\overline{\tau}_v - \underline{\tau}_v + 1$ many snapshot targets: $v_{\underline{\tau}_v}, v_{\underline{\tau}_v+1}, \ldots, v_{\overline{\tau}_v}$, thus,

$$\tilde{\mathcal{V}} := \{v_\theta : v \in \mathcal{V}_o,\, \theta = \underline{\tau}_v, \underline{\tau}_v{+}1, \ldots, \overline{\tau}_v\}.$$

The set of arcs in the time-expanded network is given by $\tilde{\mathcal{A}} \subseteq \mathcal{A} \times \mathcal{T}^2$. An arc $a = (i, j, \theta, \lambda)$ in the time-expanded network only exists, if the position of $j \in \mathcal{V}$ at time step λ can be reached from the position of $i \in \mathcal{V}_o$ at time step θ with a speed of at most $\overline{v}$, thus,

$$\tilde{A} := \{(i,j,\theta,\lambda) : (i,j) \in \mathcal{A},\; \theta \in [\underline{\tau}_i, \overline{\tau}_i],\; \lambda \in [\underline{\tau}_j, \overline{\tau}_j],\; \lambda \geqslant \theta, \tag{21}$$

$$\left\| p^i(t_\theta) - p^j(t_\lambda) \right\|_2 (t_\lambda - t_\theta)^{-1} \leqslant \overline{v}\}. \tag{22}$$

There are no capacity restrictions and no flow costs on the arcs. Additionally, no holdover arcs are considered, since waiting is allowed and included in the way, that waiting combined with traveling at maximum speed can be equal to not wait and travel with reduced speed. See Figure 2 for a visualization of a time-expanded network with six moving targets, for the sake of clearity only a part of the arcs set is shown.

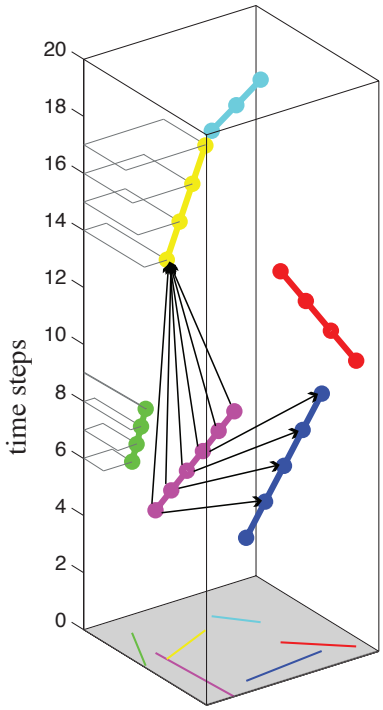

Figure 2: Visualization of six moving targets in a time-expanded network. The picture shows snapshot targets for different time steps and a part of the arcs set for the purple target.

For each salesperson the arrival time at any node in $\mathcal{V}$ is equal to the departure time at the same node, since waiting time is included in the traveling time. Having discrete time steps $\theta, \lambda \in \mathcal{T}$ we are able to evaluate the distance function for arcs at these times:

$$c_{i,j,k}^{\theta,\lambda} := c_{i,j,k}(t_\theta, t_\lambda).$$

Then, we introduce a family of binary decision variables $x_{i,j,k}^{\theta,\lambda} \in \{0,1\}$. Here, $x_{i,j,k}^{\theta,\lambda} = 1$ represents the decision of sending salesperson k from i to j, departing at time step θ in i and arriving in j at time step λ. The objective function is to minimize the total traveled distances of all salespersons:

$$\min \sum_{k \in \mathcal{W}} \sum_{(i,j,\theta,\lambda) \in \tilde{\mathcal{A}}} c_{i,j,k}^{\theta,\lambda} \, x_{i,j,k}^{\theta,\lambda}. \tag{23}$$

The demand constraint requires, that each node $j \in \mathcal{V}$ must be visited once by exactly one salesperson:

$$\sum_{k \in \mathcal{W}} \sum_{(i,\theta,\lambda):(i,j,\theta,\lambda) \in \tilde{\mathcal{A}}} x_{i,j,k}^{\theta,\lambda} = 1, \quad \forall\, j \in \mathcal{V}. \tag{24}$$

Each salesperson $k \in \mathcal{W}$ can only start once from the depot:

$$\sum_{(j,\theta,\lambda):(o,j,\theta,\lambda) \in \tilde{\mathcal{A}}} x_{o,j,k}^{\theta,\lambda} \leqslant 1, \quad \forall\, k \in \mathcal{W}. \tag{25}$$

The following flow constraints ensure the feasibility of time. Conservation is ensured at

each target of the salesperson tour except for the last one, this can be regarded as the sink of the flow:

$$\sum_{(i,\theta):(i,j,\theta,\lambda)\in\tilde{\mathcal{A}}} x^{\theta,\lambda}_{i,j,k} \geqslant \sum_{(i,\theta):(j,i,\lambda,\theta)\in\tilde{\mathcal{A}}} x^{\lambda,\theta}_{j,i,k}, \quad \forall\, j \in \mathcal{V}, \lambda \in \mathcal{T}, k \in \mathcal{W}. \tag{26}$$

Summing up, we solve the following optimization problem:

$$\min \{(23) \mid (24), (25), (26), x \in \{0,1\}^{\tilde{\mathcal{A}}\times\mathcal{W}}\}. \tag{27}$$

This formulation is called the time-discrete model or short TD model.

The restrictions of the visibility time windows are embedded in the modeling of the arcs. Usually TSP have to incorporate subtour elimination constraints. In fact, this is included by the inherent time dependency. Since time evolves, there is no cycle in the underlying time expanded network and consequently subtour elimination constraints are not needed. Furthermore, the presented model (27) is not restricted to special shapes of the target trajectories. It can handle an arbitrary non-linear trajectory, see for example Stieber and Fügenschuh [Sti18] and Section 5.4. Likewise, there is no need to restrict the speed of the targets, the model can be adapted to deal with varying target speeds. An adverse effect of the above formulation is the use of discrete time steps. In case a better accuracy of the objective function is needed, the level of discretization has to be increased, resulting in more layers of the time expanded graph and thus in more copies of the targets. As a consequence, the number of arcs and the number of variables and constraints will increase leading to a higher computational burden.

In case all salespersons are identical and start from the same depot, we have a symmetric problem. Considering the symmetric problem the proposed formulation can be simplified by removing the index k in (27) and replacing (25) by a single constraint ensuring the degree of the depot node to be at most $|\mathcal{W}|$:

$$\sum_{(j,\theta,\lambda):(o,j,\theta,\lambda)\in\tilde{\mathcal{A}}} x^{\theta,\lambda}_{o,j} \leqslant |\mathcal{W}|.$$

If energy consumption is an issue, we can restrict the number of targets, that can be intercepted within a certain period of time. We denote the maximum number of targets by L_k and the number of consecutive time steps by h_k, for $k \in \mathcal{W}$. The index k is used, when different kinds of lasers are used, with different battery capacities. Then, the following energy consumption constraints can be integrated in the model formulation:

$$\sum_{\lambda=u}^{u+h_k} \sum_{(i,j,\theta,\lambda)\in\tilde{\mathcal{A}}} x^{\theta,\lambda}_{i,j,k} \leqslant L_k, \quad \forall\, k \in \mathcal{W}, u \in \mathcal{T}, u \leqslant m - h_k.$$

Finally, we discuss a remark on the objective function. For the described reasons we minimize the sum of traveled distances as in the classical TSP. However, it is also possible to minimize the traveled time:

$$\min \sum_{k\in\mathcal{W}} \sum_{(i,j,\theta,\lambda)\in\tilde{\mathcal{A}}} (\lambda - \theta)\, x^{\theta,\lambda}_{i,j,k}.$$

Since a salesperson cannot exceed its maximum speed, we have $(\lambda-\theta) \geqslant \overline{v}^{-1}\, c^{\theta,\lambda}_{i,j,k}$. Thus, no matter which objective function is used, traveled time or traveled distance, both resulting integer programs are equal in the way that the same optimal solution is found. In an

online consideration, where the information about targets is not given in advance, another intuitive objective function is to intercept each target as early as possible. This case is addressed in Section 5.5.

3.2 A Time-Continuous Model

For this model variant the salespersons may intercept targets at any point of their trajectories, thus, the time restrictions have to be modeled by continuous variables. As opposed to the discrete model, the decision if a certain point on a trajectory can be reached from a point on another trajectory in the corresponding time interval, has to be modeled differently. Here, it is integrated into the model formulation. Recall, in the discrete case the length of a time-dependent arc and the decision if it is valid (finite arc length) is calculated in advance. For the time-continuous model, this is realized by applying big-M constraints (similar to the Miller-Tucker-Zemlin constraints for the TSP, see MILLER et al. [Mil60]). These constraints contain continuous time variables and time restrictions. Here this means, that a big-M constant is used to ensure an inequality with continuous time variables for the case, that the corresponding binary decision variable takes on the value of one. In case the decision variable is zero, the inequality is deactivated by the big-M constant. Hence, M needs to be sufficiently large. A simple example is

$$\begin{aligned} &x \leqslant 10^5 y \\ &x \geqslant 0 \\ &y \in \{0,1\}, \end{aligned}$$

where 10^5 is the big-M constant. An obvious consequence of this time-continuous approach is the fact, that we obtain an optimal feasible solution with best accuracy.

We introduce a family of binary decision variables $x_{i,j,k} \in \{0,1\}$, where $x_{i,j,k} = 1$ represents the decision of sending salesperson k from i to j (independently of the time). Moreover, continuous time variables $t_{i,k} \in \mathbb{R}$ are defined to describe the arrival time of salesperson k at target i. The set of arcs is $\mathcal{A} \subseteq \mathcal{V}_o \times \mathcal{V}$, and the length of an arc is defined by the function $c_{i,j,k}$, see (20). Now, we formulate the continuous model.

The objective function is to minimize the total traveled distances of all salespersons:

$$\min \sum_{k \in \mathcal{W}} \sum_{(i,j) \in \mathcal{A}} c_{i,j,k}(t_{i,k}, t_{j,k})\, x_{i,j,k}. \tag{28}$$

Each node $j \in \mathcal{V}$ must be visited once by exactly one salesperson:

$$\sum_{k \in \mathcal{W}} \sum_{i:(i,j) \in \mathcal{A}} x_{i,j,k} = 1, \quad \forall\, j \in \mathcal{V}. \tag{29}$$

Each salesperson $k \in \mathcal{W}$ can only start once from the depot:

$$\sum_{j \in \mathcal{V}} x_{o,j,k} \leqslant 1, \quad \forall\, k \in \mathcal{W}. \tag{30}$$

Flow conservation is ensured at each target of each salesperson tour except for the last one (sink):

$$\sum_{i:(i,j) \in \mathcal{A}} x_{i,j,k} \geqslant \sum_{i:(j,i) \in \mathcal{A}} x_{j,i,k}, \quad \forall\, j \in \mathcal{V}, k \in \mathcal{W}. \tag{31}$$

The following big-M constraints guarantee time-feasibility, that means, if salesperson k moves from i to j and arrives at $t_{i,k}$ in i, he cannot be earlier in j than $t_{i,k}$ plus the time he needs to travel from the position of i at $t_{i,k}$ to the position of j at $t_{j,k}$ using maximum speed $\overline{v}$. The time horizon T is the so called big-M constant in the big-M constraints

$$t_{i,k} + \frac{c_{i,j,k}(t_{i,k},t_{j,k})}{\overline{v}} \leqslant t_{j,k} + T \cdot (1 - x_{i,j,k}), \quad \forall\, (i,j) \in \mathcal{A}, k \in \mathcal{W}. \tag{32}$$

For the visibility time windows, the time variables have to satisfy the following bounds:

$$\underline{t}_j \leqslant t_{j,k} \leqslant \overline{t}_j, \quad \forall\, j \in \mathcal{V}_o, k \in \mathcal{W}. \tag{33}$$

Summarized, we aim to solve the following optimization problem:

$$\min \{\, (28) \mid (29), (30), (31), (32), (33),\, x \in \{0,1\}^{\mathcal{A}\times\mathcal{W}}, t \in \mathbb{R}^{\mathcal{V}_o\times\mathcal{W}}\}. \tag{34}$$

The presented formulation is called time-continuous model or short TC model of the MTSPMT. It is based on an arbitrary nonlinear continuous function $c_{i,j,k}$ for the distance between two points of distinct moving targets. In order to get a linear objective function and to apply a standard MILP solver such as CPLEX, we have to restrict the movement of the targets. To this end, we assume the trajectories to be straight lines and the speed of each target to be constant. To simplify matters, we use the same constant speed $\overline{v}_T$ for all targets throughout this thesis. Additionally, it is assumed that $\overline{v} \gg \overline{v}_T$, such that the salespersons are able to catch targets in reasonable short time.

Then, $c_{i,j,k}$ represents the Euclidean distance between two points on two straight lines. With this, the above presented optimization problem (34) can be handled as a SOCP. The constraints (32) define the cones and make the set of feasible solutions to be convex. In the sequel we present the adjusted SOCP formulation that we use for CPLEX.

We introduce the real auxiliary variables $c^x_{i,j,k}$ and $c^y_{i,j,k}$ for the x- and y-components of the directional vector, that goes from point $p^i(t_{i,k})$ to point $p^j(t_{j,k})$ with the Euclidean distance $c_{i,j,k}(t_{i,k},t_{j,k})$ and $a_{i,j,k}$ for the integral over the time horizon for departure and arrival time:

$$a_{i,j,k} = \int_{t_{i,k}\in[0,T]} \int_{t_{j,k}\in[0,T]} c_{i,j,k}(t_{i,k},t_{j,k})\, x_{i,j,k}.$$

Finally, $\overline{a}_{i,j,k}$ is used for the right hand side of the cone definition. Hence, we formulate the following SOCP. The objective function is

$$\min \sum_{k\in\mathcal{W}} \sum_{(i,j)\in\mathcal{A}} a_{i,j,k}. \tag{35}$$

The constraints (29), (30), (31) and (33) remain unchanged. The constraints (32), that contain quadratic terms, are transformed into the following family of auxiliary conditions, where the trajectory of a target is represented by the convex combination of its start $(\underline{x}_i,\underline{y}_i)$ and end point $(\overline{x}_i,\overline{y}_i)$, see equations (36)–(39). We define $\Delta x_i = \overline{x}_i - \underline{x}_i$, $\Delta y_i = \overline{y}_i - \underline{y}_i$ and $\Delta t_i = \overline{t}_i - \underline{t}_i$.

$$c^x_{i,j,k} - \left(\left(\underline{x}_j + t_{j,k}\frac{\Delta x_j}{\Delta t_j} - \underline{t}_j\frac{\Delta x_j}{\Delta t_j}\right) - \left(\underline{x}_i + t_{i,k}\frac{\Delta x_i}{\Delta t_i} - \underline{t}_i\frac{\Delta x_i}{\Delta t_i}\right)\right) = 0, \tag{36}$$

$$\forall\, (i,j) \in \mathcal{A}, i \neq o, k \in \mathcal{W}.$$

$$c^x_{o,j,k} - \left(\left(\underline{x}_j + t_{j,k} \frac{\Delta x_j}{\Delta t_j} - \underline{t}_j \frac{\Delta x_j}{\Delta t_j} \right) - o^x \right) = 0, \quad \forall\, j \in \mathcal{V}, k \in \mathcal{W}. \tag{37}$$

$$c^y_{i,j,k} - \left(\left(\underline{y}_j + t_{j,k} \frac{\Delta y_j}{\Delta t_j} - \underline{t}_j \frac{\Delta y_j}{\Delta t_j} \right) - \left(\underline{y}_i + t_{i,k} \frac{\Delta y_i}{\Delta t_i} - \underline{t}_i \frac{\Delta y_i}{\Delta t_i} \right) \right) = 0, \tag{38}$$
$$\forall\, (i,j) \in \mathcal{A}, i \neq o, k \in \mathcal{W}.$$

$$c^y_{o,j,k} - \left(\left(\underline{y}_j + t_{j,k} \frac{\Delta y_j}{\Delta t_j} - \underline{t}_j \frac{\Delta y_j}{\Delta t_j} \right) - o^y \right) = 0, \quad \forall\, j \in \mathcal{V}, k \in \mathcal{W}. \tag{39}$$

The following constraints describe the condition of the uniform movement of the targets:

$$a_{i,j,k} \leqslant \overline{v} \left(t_{j,k} - t_{i,k} + T \left(1 - x_{i,j,k} \right) \right), \quad \forall\, (i,j) \in \mathcal{A}, k \in \mathcal{W}. \tag{40}$$

The next conditions are needed to formulate the cone constraints:

$$\overline{a}_{i,j,k} = a_{i,j,k} + R \left(1 - x_{i,j,k} \right), \quad \forall\, (i,j) \in \mathcal{A}, k \in \mathcal{W}, \tag{41}$$

where R is the longest possible distance. For a square the diagonal can be used $R = \lceil \sqrt{2} S \rceil$ (with S being the edge length).
Finally, the cone constraints are given as:

$$(c^x_{i,j,k})^2 + (c^y_{i,j,k})^2 \leqslant (\overline{a}_{i,j,k})^2, \quad \forall\, (i,j) \in \mathcal{A}, k \in \mathcal{W}. \tag{42}$$

Summarized, the transformed SOCP reads the following:

$$\begin{aligned} \min \{\, (35) \mid\ & (29), (30), (31), (33), (36), (37), \\ & (38), (39), (40), (41), (42), \\ & x \in \{0,1\}^{\mathcal{A} \times \mathcal{W}}, t \in \mathbb{R}^{\mathcal{V}_o \times \mathcal{W}}, c^x, c^y, a, \overline{a} \in \mathbb{R}^{\mathcal{A} \times \mathcal{W}} \}. \end{aligned} \tag{43}$$

Having this TC model formulation, a salesperson is able to intercept a moving target at any point on its trajectory. In general, the objective function value of (43) is less than or equal to the objective function value of (27) for the same instance. However, the assumptions we had to made are severe, the model is only applicable to linear trajectories with constant speeds. In contrast to this, the time-discrete model can handle trajectories of any shape and speed. Moreover, model formulations based on big-M constraints usually have a weak linear programming relaxation and thus, more nodes in the branch-and-bound tree have to be examined, slowing down the solution process Codato and Fischetti [Cod04]. Additionally, due to the big-M constants numerical instabilities can occur in the solution procedure if the constants are not tight enough. In (43) there are two of such constants, see constraints (40) and (41). Obviously, a comparison of both modeling approaches, discrete and continuous, is difficult because of the different time modeling. The consequence for a runtime comparison of these formulations is, that our test instances have to be based on linear trajectories with a constant speed for all targets.

3.3 Time Relaxations

For the next model formulations we implement the time restrictions in a different way. We focus on the so-called *time-free model*. This means, in a first phase we relax the time completely and later in a second phase we reintegrate parts of the time restrictions. The

technique was first introduced by FÜGENSCHUH et al. [Füg13]. The authors successfully applied the method to the scheduling and routing of planes and tourist travel requests during fly-in safaris, which essentially is a VRP with time windows and pickup and delivery.

First of all, the discrete case is considered and a projection of (27) is performed from the time-discrete variable space $\tilde{\mathcal{A}} \times \mathcal{W}$ to $\mathcal{A} \times \mathcal{W}$ (i.e., from $\mathcal{V}_o \times \mathcal{V} \times \mathcal{T} \times \mathcal{T} \times \mathcal{W}$ to $\mathcal{V}_0 \times \mathcal{V} \times \mathcal{W}$). The time-free counterpart of the variables $x^{p,q}_{i,j,k}$ is simply $x_{i,j,k}$. While reducing all time-discrete arcs between two different targets to only one time-free arc, we have to attach an appropriate length for this arc. To preserve the set of feasible solutions, the minimum length over all time-discrete arcs between 2 targets is used as the required length for the new time-free arc. That means, given a salesperson k and 2 different targets $i \in \mathcal{V}_o$ and $j \in \mathcal{V}$, $i \neq j$ the distance $c_{i,j,k}$ is defined as

$$c_{i,j,k} = \min \{c^{\theta,\lambda}_{i,j,k} \mid \theta, \lambda \in \mathcal{T}\}. \tag{44}$$

With this, the time-discrete model (27) reduces to the following model in the time-free space:

$$\begin{aligned}
\min \quad & \sum_{k \in \mathcal{W}} \sum_{(i,j) \in \mathcal{A}} c_{i,j,k}\, x_{i,j,k} && && (45)\\
\text{s.t.} \quad & \sum_{k \in \mathcal{W}} \sum_{i:(i,j) \in \mathcal{A}} x_{i,j,k} = 1, && \forall\, j \in \mathcal{V} \\
& \sum_{j:(o,j) \in \mathcal{A}} x_{o,j,k} \leqslant 1, && \forall\, k \in \mathcal{W} \\
& \sum_{i:(i,j) \in \mathcal{A}} x_{i,j,k} - \sum_{i:(j,i) \in \mathcal{A}} x_{j,i,k} \geqslant 0, && \forall\, j \in \mathcal{V}, k \in \mathcal{W} \\
& x \in \{0,1\}^{\mathcal{A} \times \mathcal{W}}.
\end{aligned}$$

This is a classical multi-commodity flow problem, which is easy to solve by standard MILP solvers. Obviously, the optimal value of the time-free model (45) is a lower bound to (27), because the distance coefficients are computed by minimization over time-discrete arcs. However, the reconstruction of a time-feasible solution from a time-free solution is not straightforward and not every time-free solution yields a time-feasible solution. For this purpose we have to test every time-free solution, that we encounter during the solution process, for time-feasibility. This examination is embedded in a branch-and-bound framework in order to benefit from pruning nodes whose lower bounds exceed the current best solution value.

Given an optimal solution of (45), then, we generate feasible times at which the salespersons intercept the targets according to the time-free solution. Assuming the objective function value of the constructed time-feasible solution is equal to the objective function value of the time-free solution, then the constructed solution is proven global optimal for (27). However, this rarely happens. It is likely, that a time-free solution is infeasible with respect to the time constraints. Thus, besides the optimal time-free solution, we also have to investigate the other feasible time-free solutions in a branch-and-bound process. That means (45) serves as the master problem in the branch-and-bound framework. For any feasible solution of the master problem, we try to construct a feasible counterpart with respect to the given time constraints. If the objective function value of this time-feasible solution is better than previously found ones, it is stored. Then, the current time-free solution is treated as infeasible and cut off in the branch-and-bound process to prevent a repetition. Obviously, the generated cuts should also take into account all salesperson permutations in case the

salespersons are homogeneous. If there is no time-feasible counterpart for a time-free solution, we also have to cut off the time-free solution as well. In this way we are able to check all time-free solutions. Additionally, the branch-and-bound tree can be pruned by exploiting lower bounds. This solution method is realized using the callback functionality of CPLEX. Analyzing the time-free solution before the construction phase also leads to an advantage in processing. For this we refer to the Chapter 4. For now, we concentrate on the construction of a time-feasible solution from a time-free solution.

Given a time-free solution, we call the sequence of targets, that is defined by all decision variables that are set to one, a *pretour*. Note, a pretour may be disconnected and there may be salespersons with an empty pretour. A salesperson is called active, if its pretour is not empty. The pretours of all active salespersons are tested for feasibility. A pretour is called feasible if it starts at the depot o and is a Hamiltonian path in the induced subgraph. For each feasible pretour a time-feasible tour is required.

A Time Relaxation with Discrete Time Feasibility Checking

Assuming we have a non-empty pretour. The construction of a time-feasible tour is done by setting up a checking sub-MILP. For the sub-MILP we consider all time restrictions, that are connected with the pretour. In detail this means, we include only those time-dependent arcs, that have a time-free counterpart in the given pretour (the corresponding solution variable is nonzero). In case the checking sub-MILP provides for each active salesperson a time-feasible tour, a feasible solution to (27) is found. The best time-feasible solution is stored.

The time-feasibility checking MILP is set up as a minimum-cost flow problem from a source to a sink for each active salesperson separately. Given an active salesperson k, then its pretour defines the sequence of targets, k has to visit. Let us assume n_k is the number of targets k has to visit and $(v_1, v_2, \ldots, v_{n_k})$ is the ordered sequence. Additionally, depot position o is considered as the source of the flow and we extend the sequence by a node d, which serves as the sink. Thus, $\mathcal{P}^k = \{o = v_0, v_1, v_2, \ldots, v_{n_k}, v_{n_k+1} = d\}$ denotes the sequence of targets (pretour) considered for the minimum-cost flow problem of k.

Since we have to consider only those time-expanded arcs, that correspond to an arc of the pretour, the checking MILP includes all arcs, that go from depot position o to discrete positions of v_1, from discrete positions of v_1 to discrete positions of v_2, and so on. Additionally, artificial time-discrete arcs from discrete positions of v_{n_k} to the sink d have to be generated. That means, for each arc, that enters v_{n_k} at time step θ an arc is created, that leaves v_{n_k} at time step θ and enters d at $\theta + 1$. The distance of all arcs between v_{n_k} and d is set to zero. We denote this set of arcs for salesperson k as $\mathcal{A}^k$. According to the time-discrete model (27), we introduce binary decision variables $x^{\theta,\lambda}_{v_i,v_{i+1}}$ describing the decision of sending salesperson k from target v_i to its successor v_{i+1} leaving at time step θ and arriving at time step λ. The time-feasibility checking MILP for an active salesperson k is formulated as follows:

$$\min \sum_{i=0}^{n^k} \sum_{(\theta,\lambda):(v_i,v_{i+1},\theta,\lambda)\in\mathcal{A}^k} c^{\theta,\lambda}_{v_i,v_{i+1}}\, x^{\theta,\lambda}_{v_i,v_{i+1}} \tag{46}$$

$$\text{s.t.} \sum_{(\theta,\lambda):(o,v_1,\theta,\lambda)\in\mathcal{A}^k} x^{\theta,\lambda}_{o,v_1} = 1$$

$$\sum_{(\theta,\lambda):(v_{n_k},d,\theta,\lambda)\in\mathcal{A}^k} x^{\theta,\lambda}_{v_{n_k},d} = 1$$

$$\sum_{\theta:(v_{j-1},v_j,\theta,\lambda)\in\mathcal{A}^k} x^{\theta,\lambda}_{v_{j-1},v_j} - \sum_{\theta:(v_j,v_{j+1},\lambda,\theta)\in\mathcal{A}^k} x^{\lambda,\theta}_{v_j,v_{j+1}} = 0, \quad \forall\, j \in \{1,\dots,n_k\},\ \lambda \in \mathcal{T}$$

$$x \in \{0,1\}^{\mathcal{A}^k}.$$

The optimization problem (46) aims to find the shortest path from o to d. This kind of optimization problem can be solved in polynomial time by, e.g., Dijkstra's algorithm [Dij59]. If the checking MILP (46) results in a time-feasible tour for every active salesperson, we have a total time-feasible solution for (27). If for any of the active salespersons, the checking MILP is infeasible, then it is not possible to construct a time-feasible tour out of its pretour. In this case, the construction process is aborted and the current time-free solution (with all its pretours) is cut off.

In summary, the model (45) with distances defined in (44) and the sub-MILP formulation (46) is called the time-free model with a time-discrete feasibility checking or short TFTD model. Just as the TD model, the TFTD model can be applied to non-linear shaped trajectories.

A Time Relaxation with Continuous Time Feasibility Checking

According to our time-continuous model (34), a time-free master problem can also be combined with a continuous time-feasibility checking sub-MILP. For this variant we cannot use (45) as the master problem, since its distance coefficients $c_{i,j,k}$ in the objective function are dependent on a discretization and on time-discrete arcs, see (44). Here, these coefficients are not valid. We have to replace the minimal time-expanded arclength by the real minimum length between 2 trajectories. Thus, the distance coefficients $c_{i,j,k}$ are computed as follows: For a given trajectory j we compute the time interval $[t^i_1, t^i_2]$ for trajectory i, from which j can be reached:

$$t^i_1 := \min\{t \in [\underline{t}_i, \overline{t}_i] : \left\|p^j(q) - p^i(t)\right\|_2 \leqslant \overline{v}(q-t),\ q \geqslant t,\ q \in [\underline{t}_j, \overline{t}_j]\},$$

$$t^i_2 := \max\{t \in [\underline{t}_i, \overline{t}_i] : \left\|p^j(q) - p^i(t)\right\|_2 \leqslant \overline{v}(q-t),\ q \geqslant t,\ q \in [\underline{t}_j, \overline{t}_j]\}.$$

Then we compute the minimum distance between the trajectory of i reduced to the interval $I := [t^i_1, t^i_2]$ and trajectory of j with $J := [\underline{t}_j, \overline{t}_j]$:

$$c_{i,j,k} = \begin{cases} \infty, & \text{if } I = \varnothing \\ \min\{c_{i,j,k}(t,q) : t \in I, q \in J\}, & \text{otherwise.} \end{cases} \tag{47}$$

Then the time-relaxed master model is the model (45) combined with the distance coefficients defined in (47). Having this, we can formulate the feasibility checking sub-MILP for the time-continuous case. As for the continuous model (43), the assumption of linear trajectories and constant target speed is made. Then, the checking sub-MILP for a feasible pretour of an active salesperson k can be modeled as a quadratic program. We again use continuous time variables t_{v_i} to define the arrival time of k in v_i. For a given pretour sequence $\{o = v_0, v_1, \dots, v_{n_k}\}$, we obtain the checking MILP as follows:

$$\min \quad \sum_{i=0}^{n_k-1} c_{v_i,v_{i+1}}(t_{v_i},t_{v_{i+1}})\, x_{v_i,v_{i+1}} \tag{48}$$

$$\text{s.t.} \qquad c_{v_i,v_{i+1}}(t_{v_i},t_{v_{i+1}}) \leqslant \overline{v}\,(t_{v_{i+1}} - t_{v_i}), \qquad \forall\, i = 0,1,\dots,n_k-1,$$

$$t_{v_i} \in [\underline{t}_i, \overline{t}_i], \qquad \forall\, i = 1,\dots,n_k,$$

$$x \in \{0,1\}^{\mathcal{V}_o \times \mathcal{V}}.$$

Here, the function $c_{v_i,v_{i+1}}(t_{v_i},t_{v_{i+1}})$ again denotes the Euclidean distance between the position of target v_i at time step t_{v_i} and the position of target v_{i+1} at time step $t_{v_{i+1}}$, see (20). In the objective function all traveled distances are summed up, while in the restrictions time-feasibility is checked. That means, the travel speed, that k uses to traverse a distance of $c_{v_i,v_{i+1}}(t_{v_i},t_{v_{i+1}})$ in a time difference of $(t_{v_{i+1}} - t_{v_i})$, has to be at most $\overline{v}$, the maximum speed. The bounds restriction ensures that each target is reached within its specified visibility window.

The model (45) with continuous distances (47) and the feasibility checking MILP (48) is called time-free model with a time-continuous feasibility checking or in short TFTC model. In contrast to the proposed TC model (34), the TFTC does not contain any big-M constraints.

3.4 A Set Partitioning Approach

This paragraph focuses on a set partitioning approach. Here, we consider a sequence S of all possible subsets of n targets, we have $|S| = 2^{|\mathcal{V}|} - 1 = 2^n - 1$. Then, let I be the index set $I = \{1, 2, \ldots, 2^n - 1\}$ such that s_i is the i-th subset. We define $a_i, i \in I$ as the incidence vector of s_i. The incidence vectors build up a matrix $A \coloneqq (a_i)_{i \in I}$. We assume, that every subset provides a valid tour with a finite length. Thus, we can assume, that for every s_i there is a tour with minimal length among all valid tours. This minimal tour length is called $d_i \in \mathbb{R}_+, i \in I$. The task is to find at most $w \in \mathcal{W}$ of these tours, such that each target is in exactly one tour and the aggregated lengths is minimized. The optimization problem is called set partitioning problem (SPP) and defined as

$$\begin{aligned} \min \quad & d^T x && (49)\\ \text{s.t.} \quad & A\,x = 1, \\ & \sum_{i \in I} x_i \leqslant w \\ & x_i \in \{0,1\} \quad \forall\, i \in I, \end{aligned}$$

where the dimension of matrix A is $n \times 2^n - 1$. The set partitioning constraints $Ax = 1$ ensure, that each target is reached by a salesperson, where $x_i = 1$ if and only if the optimal tour of subset $A_{(.,i)}$ is chosen. The inequality in (49) constitutes that at most w many salespersons are required.

In order to solve (49) we have to compute each of the $2^n - 1$ many optimal tours for all possible subsets of V in advance. Hence, for each subset we have to solve a MTSPMT with one salesperson, which is a TSPMT. For solving the TSPMT any of the before mentioned model variant can be applied, i.e., TD, TFTD, TC and TFTC. Once all optimal tours have been computed, we can easily solve (49), we can even solve it several times with different values of w.

3.5 Adaptations to the Formulations Regarding Infeasibility

So far, the described models find an optimal solution for feasible instances of MTSPMT. Here, we also focus on a model, that can handle infeasible instances. In case it is not possible to intercept all targets within their visibility time windows, due to the limited

amount of salespersons, we speak of infeasible instances. Instead of an output that just states: "infeasible", we are interested in a solution that incorporates as many targets as possible for interception and minimizes the traveled distances for all salespersons.

To adapt the models TD and TC we introduce a penalty constant $D \in \mathbb{R}_+$ with a value of the longest possible distance of the underlying space. For a squared space it is the length of its diagonal. Furthermore, binary slack variables $s_i, i \in \mathcal{V}$ are used to model the targets, that are not intercepted. Then, TD model (27) is adjusted in the following way. The objective function is extended by a penalty term for each missing target. The demand constraints of each target is also adjusted by the corresponding slack variable in case it is not intercepted. Finally, the domain definition of the slack variables is given:

$$\min \sum_{k\in\mathcal{W}} \sum_{(i,j,\theta,\lambda)\in\tilde{\mathcal{A}}} c^{\theta,\lambda}_{i,j,k} x^{\theta,\lambda}_{i,j,k} + \boldsymbol{D \sum_{j\in\mathcal{V}} s_j}$$

$$\sum_{k\in\mathcal{W}} \sum_{(i,\theta,\lambda):(i,j,\theta,\lambda)\in\tilde{\mathcal{A}}} x^{\theta,\lambda}_{i,j,k} + \boldsymbol{s_j} = 1, \quad \forall\, j \in \mathcal{V}$$

$$\boldsymbol{s_j \in \{0,1\}^{\mathcal{V}}}\ .$$

All other constraints remain unchanged.

The same modifications can be applied to the time-continuous model TC:

$$\min \sum_{k\in\mathcal{W}} \sum_{(i,j)\in\mathcal{A}} c_{i,j,k}(t_{i,k}, t_{j,k}) x_{i,j,k} + \boldsymbol{D \sum_{j\in\mathcal{V}} s_j}$$

$$\sum_{k\in\mathcal{W}} \sum_{i:(i,j)\in\mathcal{A}} x_{i,j,k} + \boldsymbol{s_j} = 1, \quad \forall\, j \in \mathcal{V}$$

$$\boldsymbol{s_j \in \{0,1\}^{\mathcal{V}}}\ .$$

3.6 Adaptations to the Formulations Regarding Non-Linear Trajectories

This section focuses on non-linear continuous trajectories. For time-discrete models the handling is straightforward, the non-linear trajectories are discretized by time and every time step is associated with a point in the Euclidean space. The set of feasible arcs of the time-expanded network is computed accordingly to (21) and in advance. Then, the model TD (27) can be applied to the non-linear data, see for example Stieber and Fügenschuh [Sti18].

Concerning continuous models non-linear trajectories can only be addressed when the non-linear trajectories are approximated by piece-wise linear functions. A piece-wise linear approximation is a method to fit a curve (here a trajectory) by piece-wise linear segments (line segments). The approximation can be produced by sampling the curve and interpolating linearly between the obtained sampling points.

To apply the time continuous model TC to piece-wise linear trajectories some adjustments are needed. We introduce additional variables $d_{i,j,k}$ to model distances between the targets $i \in \mathcal{V}_o$ and $j \in \mathcal{V}$. Furthermore, $d^x_{j,k}$ and $d^y_{j,k}$ denote the x- and y-coordinate of the position of target j at time $t_{j,k}$. We also introduce cost variables, which are called $c_{i,j,k}$, for $(i,j) \in \mathcal{A}$ and $k \in \mathcal{W}$. Then, the objective function reads as follows:

$$\min \sum_{k\in\mathcal{W}} \sum_{(i,j)\in\mathcal{A}} c_{i,j,k}. \tag{50}$$

The cost variables are connected to the traveled distances by a big-M formulation:

$$c_{i,j,k} = d_{i,j,k} - (1 - x_{i,j,k}) \cdot R, \quad \forall\, (i,j) \in \mathcal{A}, k \in \mathcal{W}. \tag{51}$$

Having these constraints, the objective function serves as an upper bound on the distances, that are traveled by the salespersons. Here, the big-M constant R is again the longest possible distance among all pairs of nodes, for a squared area it is the length of its diagonal.

The time-free part of the TC model remains unchanged, that are the constraints (29), (30) and (31). The same holds for the visibility time window constraints (33) and the time-feasibility constraints . We repeat the constraints for the sake of completeness:

$$\sum_{k \in \mathcal{W}} \sum_{i:(i,j) \in \mathcal{A}} x_{i,j,k} = 1, \qquad \forall\, j \in \mathcal{V}, \tag{52}$$

$$\sum_{j \in \mathcal{V}} x_{o,j,k} \leqslant 1, \qquad \forall\, k \in \mathcal{W}, \tag{53}$$

$$\sum_{i:(i,j) \in \mathcal{A}} x_{i,j,k} \geqslant \sum_{i:(j,i) \in \mathcal{A}} x_{j,i,k}, \qquad \forall\, j \in \mathcal{V}, k \in \mathcal{W}, \tag{54}$$

$$\underline{t}_j \leqslant t_{j,k} \leqslant \bar{t}_j, \qquad \forall\, j \in \mathcal{V}_o, k \in \mathcal{W}. \tag{55}$$

The trajectories of the targets are given by piece-wise linear functions (PWL^x,PWL^y : $[0,T] \to \mathbb{R}$), which are parametrized by the time. Then, the x- and y-coordinates of the position of target $j \in \mathcal{V}$ are given by:

$$d^x_{j,k} = PWL^x(t_{j,k}), \quad d^y_{j,k} = PWL^y(t_{j,k}), \qquad \forall\, j \in \mathcal{V}, k \in \mathcal{W}. \tag{56}$$

The following quadratic constraints connect the trajectory positions with the Euclidean distance $d_{i,j,k}$:

$$d^2_{i,j,k} \geqslant (d^x_{j,k} - d^x_{i,k})^2 + (d^y_{j,k} - d^y_{i,k})^2, \quad \forall\, (i,j) \in \mathcal{A}, k \in \mathcal{W}. \tag{57}$$

Finally, the time-feasibility constraints remain unchanged, but technically, the distance functions $c_{i,j,k}(t_{i,k},t_{j,k})$ in (32) are replaced by the cost variables $c_{i,j,k}$. The meaning of both entities are the same, however in this case we are faced with linear constraints compared to (32):

$$c_{i,j,k} \leqslant \overline{v}\,(t_{j,k} - t_{i,k}) + T \cdot (1 - x_{i,j,k})\,, \quad \forall\, (i,j) \in \mathcal{A}, k \in \mathcal{W}. \tag{58}$$

Summarized, we aim to solve the following optimization problem:

$$\begin{aligned} &\min\{\,(50) \mid (51) - (58),\\ &\qquad x \in \{0,1\}^{\mathcal{A} \times \mathcal{W}}, t \in \mathbb{R}_+^{\mathcal{V}_o \times \mathcal{W}}, d^x, d^y \in \mathbb{R}_+^{\mathcal{V} \times \mathcal{W}}, c, d \in \mathbb{R}_+^{\mathcal{A} \times \mathcal{W}}\}, \end{aligned} \tag{59}$$

which is a SOCP, where the non-linear trajectories are approximated by piece-wise linear functions. The solver CPLEX, that we use for our test instances provides algorithmic structures to simply define piece-wise linear functions.

3.7 Summary

The MTSPMT should be treated as an online optimization problem, that is, the targets are not known before the optimization starts, instead they occur afterwards and often step

by step. Still, a fast routine to solve the offline variant could serve as the backbone of an online solver with a moving horizon approach. Here new data is integrated into the offline algorithm at run-time, and a fast offline algorithm can be used to get a tentative decision to the online problem that is re-optimized anytime new targets emerge.

To find fast offline algorithms we investigate different model approaches. The main focus is on the time aspect and the different model formulations can be characterized accordingly. The presented model approaches are the TD model, where the targets are embedded in a time-expanded network, the TC model, which includes the check of feasibility of the arcs in the model, the two-phase time-free models with discrete and continuous time feasibility checking and a set partitioning model, where the best tours of all possible subsets of targets have to be computed in advance. In the end adaptations for the models are provided, which deal with infeasible instances and for the time-discrete models with non-linear trajectories of the targets. The generated test instances live in the $\mathbb{R}^2$, nevertheless all presented models are not restricted to the two-dimensional space, they can also be applied to the $\mathbb{R}^3$.

4 Implementational Details

The presented models in Chapter 3 are implemented by the programming language C++ using Eclipse as an integrated development environment. The optimization software CPLEX is used to model and solve randomly generated problem instances. More precisely, the Concert Technology provided by CPLEX is integrated, which includes an application programming interface for C++ programs.

While problem instances modeled as TD and TC can be solved directly by invoking the solver, the solution procedure for the time-free models TFTD and TFTC is a lot more complex. A branch-and-bound framework is used to solve the time-relaxed master problem, then, for a feasible solution of the master problem a time feasibility checking sub-MILP is constructed. This solution procedure is described in more detail in Section 4.1. The course of action for the set partitioning approach is discussed in Section 4.2.

4.1 Solution Procedures for the Time-Relaxed Models

This section focuses on the time-relaxed master problem (45). While in the discrete case, the distance coefficients $c_{i,j,k}$ are calculated as the minimal arclength between 2 targets as given in (44), in the continuous case the distance coefficients for a pair of targets, that can be intercepted sequentially by a salesperson, are computed by minimizing the distance function over two continuous time intervals as stated in (47).

The model (45) (with either (44) or (47)) is called master problem and the solution procedure is embedded in a branch-and-bound framework. To check the solutions of the master problem and to produce time-feasible solutions we use the callback utilities of CPLEX. We implement an instance of the BranchCallback and an instance of the LazyConstraintCallback. The latter one is a user-written callback to solve mixed-integer linear programs. Each time a candidate feasible solution of the master problem (called pretour) is found at a node in the branch-and-bound tree the LazyConstraintCallback is invoked and violated constraints are applied. Those constraints are applied in a "lazy" fashion, i.e., only if they are violated, what makes the name of the callback.

In the LazyConstraintCallback a validation check of the pretour is performed as well as the construction of feasible times according to the sub-MILPs (46) and (48). The callback algorithm is presented in Algorithm 4. A more detailed description of individual steps is given in the sequel. In lines 1-3 the objective function value of the current solution of the master problem is checked. The objective function value of the master problem serves as a lower bound for the MTSPMT. If this lower bound at the current node is greater or equal to the best objective function value "with time" found so far, we cannot improve. Thus, the callback terminates and the current node is pruned in the BranchCallback, which is invoked afterwards. Since the objective function value of the master problem is checked against the current best objective function value of the problem "with time", that means two different problems are considered, thus, pruning is not executed automatically. The BranchCallback is only used for pruning, branching is performed the default way of CPLEX.

The lines 4-11 perform prechecks to the pretours in order to sort out infeasible ones. The first precheck is a cycle detection, which is described in Section 35, the second precheck is a time-feasibility check using interval propagation of the visibility time windows, see Section 35 for more details. After prechecking is executed one of the MILPs (46) or (48) is set up to compute the corresponding tours with times.

Each pretour and its counterpart "with time" are stored in a solution pool in order to prevent setting up and solving the same sub-MILP more than once for repeating pretours

Algorithm 4: LazyConstraintCallback: Pretour validation check and construction of times.

Data: Time-relaxed solution variables x, best objective function value found so far *best_obj_val*.
Result: If exist, time-feasible tours for all salespersons.

```
1  #Bounds exploitation
2  if current objective function value curr_obj_val ⩾ best_obj_val then
3      return;
4  #Cycle detection
5  if the pretour of an active salesperson s contains a cycle then
6      add a global cut for all salespersons;
7      return;
8  #Validation check by interval propagation
9  if the pretour of an active salesperson s is identified as time-infeasible then
10     add a global cut for all salespersons;
11     return;
12 #Construction of feasible times
13 initialize current solution curr_tour ← ∅; all_salespersons_feasible ← true;
14 initialize objective function value for time-feasible solution tour_val ← 0;
15 for each active salesperson s ∈ 𝒲 do
16     if all_salespersons_feasible ≠ true then
17         break;
18     if pretour of s is already in solution pool then
19         get time-feasible tour r for s from solution pool;
20         if r is infeasible then
21             all_salespersons_feasible ← false;
22     else
23         set up MILP to compute time-feasible tour r for s;
24         solve MILP;
25         #r is a time-feasible tour of s;
26         tour_val ← objective function value of MILP;
27         curr_tour = curr_tour ∪ r;
28         add pretour of s and time-feasible tour r to solution pool;
29 if all_salespersons_feasible = true then
30     add global cut to prevent solution to be repeated by another salespersons permutation;
31     if tour_val < best_obj_val then
32         best_obj_val = tour_val;
33         save curr_tour;
34         return;
35 return;
```

in different solutions. This is realized in the lines 19 and 28 of Algorithm 4. To this end, we use a single-threaded optimization instead of a multi-threaded one. In multi-threaded mode it is only allowed to access local data, that is data only related to the current node in the branch-and-bound tree. However, a solution pool contains data from different nodes and thus, cannot satisfy the restriction of local data. Another reason for the single-threaded mode is, that CPLEX is not deterministic in the order of callback invocation. For multiple runs of the same instance with the same parameter setting on the same platform CPLEX cannot guarantee the same order of callback invocations, this may lead to a non-deterministic variability in running times.

A time-feasible solution is found, when there is a time-feasible tour for each active salesman. This solution has to be returned to the master problem to possibly replace the best objective function value found so far and its corresponding solution. Then a global cut is added to the master problem in order to prevent the repetition of the current time-free solution by another salesperson permutation.

Summing up, we have the time-discrete model (TD) (27), the time-continuous model (TC) (34), the time-free master (45) with time-discrete feasibility checking (TFTD) (46) and the time-free master (45) with time-continuous feasibility checking (TFTC) (48). TFTD and TFTC can be solved using Algorithm 4.

Pretour Checking: Cycle Detection

Since (45) is a multi-commodity flow formulation, any solution presents a feasible flow but not necessarily a feasible tour due to the lack of subtour elimination constraints in the master problem. In lines 4-7 of Algorithm 4 we extract the pretours of all active salespersons and test whether one of these pretours contains a cycle. If a cycle $\mathcal{C} \subset \mathcal{A}$ is found, it will be cut off by the following constraints:

$$\sum_{(i,j)\in\mathcal{C}} x_{i,j,k} \leqslant |\mathcal{C}| - 1, \quad \forall\, k \in \mathcal{W}. \tag{60}$$

We have also extended this cut by all arcs within the cycle. We define the targets of the cycle as $\mathcal{V}(\mathcal{C})$, then the extended cut reads as follows

$$\sum_{(i,j)\in\mathcal{A}:i,j\in\mathcal{V}(\mathcal{C})} x_{i,j,k} \leqslant |\mathcal{C}| - 1, \quad \forall\, k \in \mathcal{W}.$$

Experiments showed, that this extended cut could not improve the runtimes compared to the cut (60). There are other cut formulations as well. We tested a variant, that considers the arcs between the cycle targets and the remaining targets. We define the remaining targets as $\overline{\mathcal{V}}(\mathcal{C}) := \mathcal{V}_o \backslash \mathcal{V}(\mathcal{C})$. Then, the following cut was considered

$$\sum_{(i,j)\in\tilde{\mathcal{A}}} x_{i,j,k} \geqslant 1, \quad \forall\, k \in \mathcal{W},$$

where $\tilde{\mathcal{A}} := \mathcal{A} \cap \big(\{(i,j) : i \in \overline{\mathcal{V}}(\mathcal{C}), j \in \mathcal{V}(\mathcal{C})\} \cup \{(i,j) : i \in \mathcal{V}(\mathcal{C}), j \in \overline{\mathcal{V}}(\mathcal{C})\}\big)$. However, this cut also did not have any positive effect on the runtime compared to (60). Thus, the cut defined in (60) is used.

Pretour Checking: Interval Propagation

Here, a time-feasibility check for a pretour is addressed. This method of checking exploits the visibility time windows of a pretour to get the information whether a corresponding time-feasible tour exists or not. It is located in the lines 8-11 of Algorithm 4. The visibility time windows (intervals) are propagated sequentially from one target of a pretour to its successor. For moving targets the interval propagation to test time-feasibility is a lot more complicated than for fixed targets as stated in Section 2.8. Since the distance that has to be traveled is dependent on the position of the targets at the departure and the arrival time.

At first, we address the case of discrete time steps. Given a pretour $v_1, v_2, \ldots, v_q$ of an active salesman, we start at the first node v_1. Here we check, if there are time steps in the

visibility time window of v_1, which can be used to arrive from the depot o and to leave for v_2. All those time steps are combined by intersecting both, the possible *arrival* and the possible *departure interval* in v_1. This generated interval serves as the new visibility time window for v_1 and it is propagated to the following node the same way. Note, that for the depot node o the whole time horizon is considered as visibility time window, thus, the whole visibility time window of v_1 can be used as the arrival interval. With respect to the visibility time window of v_2 we compute a possible departure interval for v_1 with feasible time steps for a departure to v_2. The next step is to generate a cut set from both interval, the arrival interval and the departure interval of v_1. This procedure is continued to the last target of the pretour. The case that an intersection interval at a certain target of the pretour is empty, indicates that the pretour, we started with, is not time-feasible and the interval propagation procedure is aborted. The case, that a non-empty arrival interval is found for the last target of the pretour indicates, that this pretour is time-feasible. An example of the first steps of this procedure is visualized in Figure 3. The arrival interval of target v_1 is the discrete interval $[2, 5]$, which is the whole visibility time interval of v_1. The interval to leave for v_2 with a speed of at most $\overline{v}$ is $[2, 3]$ and the corresponding arrival interval of v_2 is $[3, 4]$. If the salesperson leaves v_1 later than at time step 3, target v_2 cannot be reached within its visibility window of $[0, 4]$. An earlier departure is not possible due to the visibility time interval of v_1. Then the intersection of the arrival interval $[2, 5]$ and the departure interval of $[2, 3]$ in v_1 results in the new interval $[2, 3]$. This procedure of time interval propagation has to be continued to the following nodes of the current pretour and their visibility time windows.

In case there is an empty cut set, we have an infeasible interval and thus, an infeasible pretour and also the corresponding time-free solution is infeasible, which is rejected then. The time-free solution is cut off by adding the following global cut to the master problem. Let $o = v_0, \ldots, v_{n_s}$ be an infeasible pretour of the current solution. Then, $\mathcal{P} \subset \mathcal{A}$ denotes the set of consecutive target pairs of the pretour. We formulate the following constraint for every salesperson to cut off this pretour:

$$\sum_{(i,j)\in\mathcal{P}} x_{i,j,k} \leqslant |\mathcal{P}| - 1, \quad \forall\, k \in \mathcal{W}. \tag{61}$$

For any pair of anti-parallel arcs (i,j) and (j,i) in the time-free master problem, we know, that only one of these arcs can be traversed by a salesperson. If both arcs were traveled, then there would be a target visited twice, which is against the demand constraint. Hence, we have

$$x_{i,j,k} + x_{j,i,k} \leqslant 1, \quad \forall\, k \in \mathcal{W}. \tag{62}$$

Lifting $|P| - 1$ anti-parallel arcs into the cut (61), it extends to:

$$\sum_{(i,j)\in\mathcal{P}} x_{i,j,k} + \sum_{(i,j)\in\mathcal{P}\setminus(v_{n_s-1},v_{n_s})} x_{j,i,k} \leqslant |\mathcal{P}| - 1, \quad \forall k \in \mathcal{W}. \tag{63}$$

Then, we address the case of continuous times. Interval propagation for the time-feasibility checking with continuous times is done in a similar way as for the discrete case. In general, the resulting arrival and departure intervals are slightly larger, see Figure 4, due to an exact calculation of the travel time. Figure 4 presents a pretour consisting of the target sequence o, v_1, v_2, where o is the depot. Each target is visualized by its trajectory and corresponding discrete time steps (black numbers). However, a salesperson can intercept a target at any time on the visualized trajectory. Here, the arrival and departure time is not rounded to the next time step, its exact value is computed using the Euclidean distance

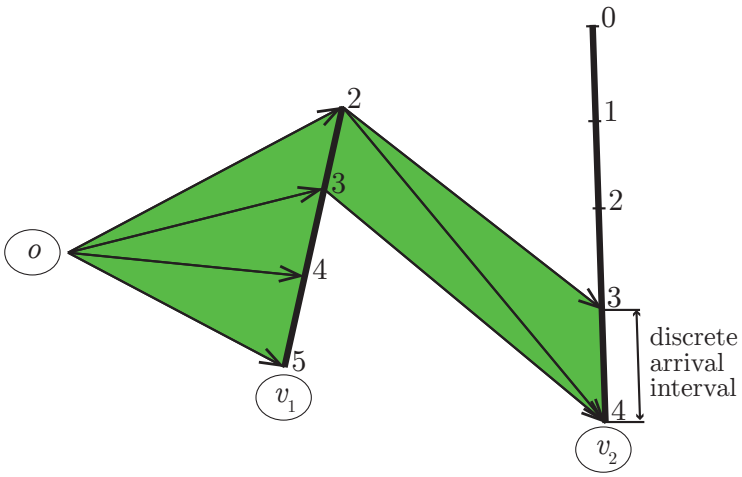

Figure 3: Interval propagation for discrete time steps. This figure presents the depot position o and the first 2 targets of a given pretour: o, v_1, v_2. Each target is visualized by its trajectory and the corresponding discrete time steps, which are indicated by numbers. The green area describes the discrete arrival and departure interval between the 3 consecutive targets.

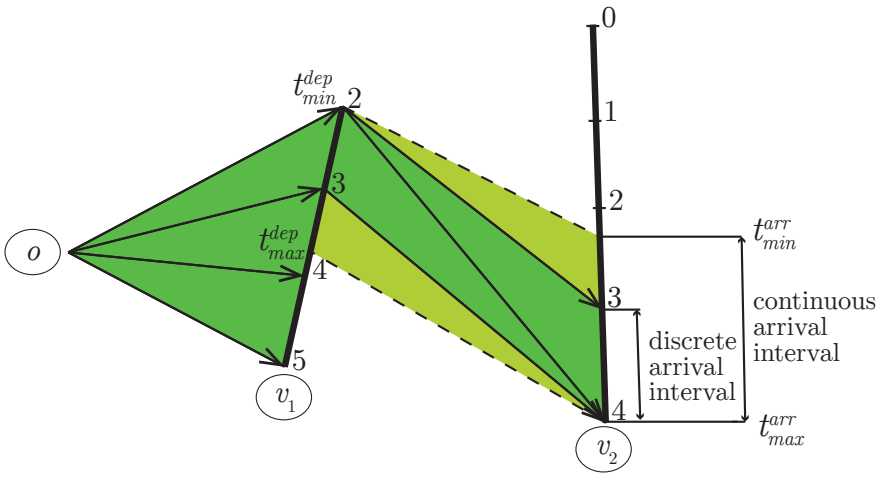

Figure 4: Interval propagation for continuous times. The dark green area describes the discrete departure and arrival interval between consecutive targets. The light green area is an extension of the departure and arrival interval when continuous times are considered.

between the corresponding positions of the targets. With this a salesperson is able to arrive earlier and to depart later compared to the case of discrete time steps.

The computation of the exact arrival interval depends on the departure interval of the predecessor target and is obtained by finding earliest and latest arrival. An example is visualized in Figure 4. Here, the arrival interval of v_1, which is called $[a_{v_1}]$, is the whole visibility time window of v_1, since the predecessor node (depot o) has the whole time horizon as departure interval. Hence, the arrival interval of v_1 is

$$[a_{v_1}] := [\underline{t}_{v_1}, \bar{t}_{v_1}] = [2, 5].$$

For the departure interval of a target the visibility time window of the successor target has to be considered, see for instance v_1 in Figure 4. Here, the departure interval $[d_{v_1}] = [t^{dep}_{min}, t^{dep}_{max}]$ has to be identified based on the visibility time window of v_2, which is $[\underline{t}_{v_2}, \bar{t}_{v_2}] = [0, 4]$ and the previously determined arrival interval $[a_{v_1}]$. The departure interval $[d_{v_1}]$ is

computed as follows:

$$[d_{v_1}] := [\,\underline{t}_{v_1},\, \max\{z \in [a_{v_1}] : \|p^{v_2}(\tilde{z}) - p^{v_1}(z)\|_2 \leqslant \overline{v}\,(\tilde{z} - z), \tilde{z} \in [\underline{t}_{v_2}, \overline{t}_{v_2}]\}\,]. \tag{64}$$

In case the maximum in (64) does not exists, we have $[d_{v_1}] = \varnothing$. Then, the sequence of targets o, v_1, v_2 is not time feasible. This means, that a salesperson is not able to travel from the depot to v_1 and then to v_2 and visit every target within its visibility time window. If the resulting interval $[d_{v_1}]$ is feasible, it serves as the new visibility time window of v_1 for an arrival in v_2. The arrival interval of v_2 is called $[a_{v_2}] = [t^{arr}_{min}, t^{arr}_{max}]$ and is calculated as

$$[a_{v_2}] := [\,\min\{\tilde{z} \in [\underline{t}_{v_2}, \overline{t}_{v_2}] : \|p^{v_2}(\tilde{z}) - p^{v_1}(z)\|_2 \leqslant \overline{v}\,(\tilde{z} - z), z \in [d_{v_1}]\},\, \overline{t}_{v_2}\,]. \tag{65}$$

If the minimum in (65) does not exist, then $[a_{v_2}] = \varnothing$ and time-feasibility is not ensured for a travel sequence of o, v_1, v_2.

Given a predefined sequence of moving targets (pretour), this procedure is continued to the first detection of an empty interval (time-infeasibility) or to a non-empty arrival interval at the last target of the sequence (time-feasibility).

The optimization problems stated in (64) and (65) can be handled by squaring the constraint and thus formulating a SOCP to get the correct departure or arrival interval.

If for a pretour time-infeasibility is detected by interval propagation, then, the corresponding pretour is cut off for any salesperson permutation. In the other case the pretour is time-feasible and in the following step the MILP (48) is set up to compute the corresponding times.

Finally, every pair of pretour and corresponding tour "with times" is stored in a *solution pool* in order to prevent setting up and solving the same sub-MILP more than once for pretours, re-occurring in other master solutions. The solution pool is used in the discrete and in the continuous case. It is realized in Algorithm 4 in the lines 18, 19, and 28.

4.2 Solution Procedure for the Set-Partitioning Approach

Each subset of n targets is binary coded, which results in $2^n - 1$ many different subsets. Then, for each subset the corresponding TSPMT (with one salesperson) is modeled via the TD formulation and solved. For each subset its objective function value and its runtime is stored. Having this information, the different subsets are combined optimally according to the SPP formulation (49), to ensure, that every target is in exactly one subset and thus, visited exactly once. The number of how many subsets are combined at most corresponds to the number of salespersons, since each subset is a tour of one salesperson. Once, all subsets are solved as TSPMT, the SPP can be solved several times with different numbers of salespersons.

To formulate the TSPMT any of the proposed model formulations from Chapter 3 can be applied. However, with respect to computation times, we decided to only use models with discretized time steps (TD and TFTD).

5 Computational Results

The presented models have either a discrete or a continuous handling of time. Generally, it depends on the application or on the computational complexity which approach to choose. Due to the specific characteristics of discrete and continuous models as the handling of time and the handling of arcs with the decision if a salesperson can move the arc length (distance) with a speed of at most $\overline{v}$, a mutual comparison is not easy. Another issue is the shape of the trajectories. The time-continuous models are restricted to straight lines and constant target speed to be solvable, while the time-discrete models are not. Furthermore, there is a difference in accuracy between a discrete and continuous handling of time. Obviously, the objective function value of an instance modeled with continuous times is always less than or equal to the objective function value of the same instance modeled with discrete times.

Due to the described reasons, we compare the runtimes of the two discrete modeling approaches and the runtimes of the two continuous modeling approaches. We use instances with randomly generated linear target trajectories for all modeling approaches and a constant speed value for all targets.

5.1 Instance Generation

We use a set of randomly generated test instances. A test instance is specified by a number of salespersons, a number of moving targets and a step size as the accuracy of discretization. The operating space is a square of size 500 length units and the trajectories are created with random lengths between 100 and 400 length units. For reasons of clarity and simplicity in visualization of the instances we prohibit any pairwise intersections of the trajectories. The targets are assigned a constant speed value of 32 length units per time unit, while the salespersons can travel at most 200 length units in a time unit. In Stieber et al. [Sti14] we observed, that instances have a higher complexity, if the difference between target speed and salesperson speed is high. Here, the number of possible tours of the salespersons rises when the speed difference increases.

Obviously, a power of two for the target speed is required to be able to create finer time-discretizations of the trajectories by introducing new time steps right in the middle of two existing ones. Following this, we create three different levels of time discretization. In particular, for the first discretization level, called D32, a step size of 32 length units is used for the trajectories. Then, the same instances are generated with a two times finer discretization (D16). Here, the step size between two consecutive time steps is 16 length units and for the four times finer discretization (D8) time steps are included every 8 length units. Obviously, the size of the instances in terms of number of variables and constraints is increased with a higher number of time steps.

For the comparison of runtimes the generated test instances are solvable instances, that means no target will reach its upper time limit before being visited by a salesman. This is achieved by assigning the visibility time windows to the targets in such a way that one salesperson is able to intercept all targets one after another. In all the instances salespersons start their tour at the depot position o, an initial position located in the center of the operating space. In total, instances are created, where the number of targets is 6, 8, 10, 12, 14, 16, 18 and 20, the number of salespersons is 1, 2, 3, 4, 5 and 6 and the discretization levels are D32, D16 and D8. An example of an instance with 12 targets, 2 salespersons starting from the depot in the middle and medium discretization level D16 is visualized in Figure 5. The visibility time windows are given by the numbers at the end points of the trajectories.

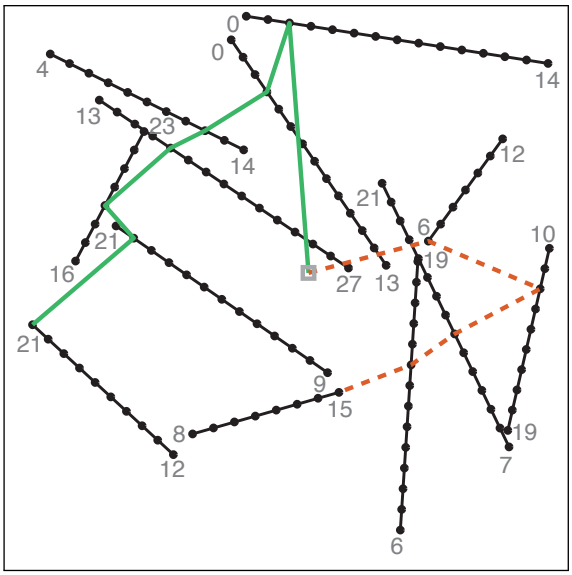

Figure 5: An instance of the MTSPMT with 12 targets, 2 salespersons, and medium discretization level (D16). Trajectories are visualized by black lines, the numbers at the endpoints of the trajectories correspond to their visibility time windows. The solid green line segments and the dashed red line segments describe the tours of the 2 salespersons.

The computational experiments were carried out on an Apple Mac Pro computer running the MacOS 10.12.6 operating system with an Intel Xeon E5 running at 3.5 GHz on 6 cores, 12 MB L3 cache, and 128 GB 1066 MHz DDR3 RAM. The version of CPLEX we used was 12.10 [IBM17]. Our aim is to evaluate the presented models with respect to their computational times for solving the generated instances. The runtime of an instance is defined by the time required by CPLEX to compute the global proven optimal solution, including the time needed for the callbacks.

5.2 Runtime Comparison

All optimization problems are solved with CPLEX. While the solution procedure of the time-free models TFTD and TFTC is customized by a LazyConstraintCallback and a BranchCallback of CPLEX, the instances modeled with TD and TC are solved without callbacks and directly by CPLEX' MILP or Barrier algorithms. The CPLEX parameters used for the optimization of the generated instances are listed in Table 1. For the time-free models, the node heuristic (HeurFreq) is turned off in order to save runtime, otherwise CPLEX would permanently check time-free solutions that are usually infeasible. Furthermore, we set the MIPEmphasis parameter to moving best bounds for the time-free models. For TD this setting would extremely slow down the computation, thus, to be fair we leave the MIPEmphasis parameter at its default value. Moreover, for the time-free master models, the cuts created by CPLEX are also turned off, since our LazyConstraintCallback is producing cuts, when checking pretours. Since CPLEX' callbacks are not compatible with dynamic search, it is turned off for all branch-and-bound models. For several reasons we cannot use parallel optimization. It is not compatible with the solution pool, since it makes our callbacks non-deterministic. Another reason is, that CPLEX starts several callbacks concurrently and even if the solution is already found, optimization terminates after finishing all callbacks and their synchronization. Furthermore, CPLEX cannot guarantee the same order of callback invocation for multiple runs of the same instance and with the same parameter setting, in this sense parallel optimization may lead to slightly different

runtimes. We use the sequential optimization mode instead. For each instance a time limit of one hour is imposed for its solution. All other CPLEX parameters are left at their default values.

In order to compare runtimes with a time limit, we compute a comparable score *sc*. It takes into account the runtime, which is at most 3,600 sec and the remaining gap

$$gap := \frac{|bestbound - bestinteger|}{1e{-}10 + |bestinteger|},$$

where *bestbound* is the objective function value of the best node remaining in the branch-and-bound tree and *bestinteger* is the best integer objective function value found so far. The value of *gap* is at most 1. The score is defined as

$$sc = \frac{runtime}{3600} + gap. \tag{66}$$

Obviously, we have $sc \in [0, 2]$. In case the score is less than 1, the optimization has finished within an hour and the gap is 0. In case the score is above 1, the optimization has aborted with a gap equal to $(sc - 1) \cdot 100\%$.

In the first experimental set we fix the number of salespersons to a value of 3, the discretization level to D16 and vary the number of targets from 6 to 20. For each number of targets 21 instances are randomly generated. The runtimes and gaps as well as the arithmetic means are visualized in Figure 6. The results for the models TD and TFTD are shown in the left picture, while the values for TC and TFTC are displayed in the right picture. Note, that for reasons of better comparability the TD and TC values are shifted to the left by 0.2 on the horizontal axis and the TFTD and TFTC values are shifted to the right by 0.2 of the respective target number. Technically, the values belong to one single number of targets, however without shifting the values would overlap, which would lead to a worse readability and comparability. We also use this kind of shifting in the following visualizations.

The problem addressed here is a real-time problem. This means the time to produce a solution is restricted to a prescribed limit. Due to the lack of real input data, we set this real-time limit to 3 sec. The obtained results show that with model TD instances with up to 10 targets can be solved within the real-time limit of 3 sec. Furthermore, the results over all four models show, that for 6 to 20 targets, the best averaged scores are obtained with the TD model. Concerning the averaged runtimes of the other three models TFTD, TC and TFTC only instances with 6 targets can be solved in less than 3 sec. However, there are six instances, that can be solved faster with TFTD than with TD. From an amount of 10 targets onward the runtimes of TFTD increases significantly, from 14 targets and above no instance can be solved within the time limit of 3,600 sec. Regarding the continuous cases, TC and TFTC are very similar up to 10 targets, but above 10 targets, where most of the instances cannot be solved within the time limit, TFTC has got much better gaps than TC (only half of it).

For the second experimental set we fix the number of targets to 10, the discretization level to D16, and the number of salespersons varies from 1 to 6. The respective runtimes and gap values of all four models are visualized in Figure 7. The best scores over all salespersons are also obtained for the TD model. The runtime of TFTD is sensitive to the number of salespersons, the higher the number of salespersons, the more instances cannot be solved within the time limit. A similar behavior can be observed for TC and TFTC, while the arithmetic mean of the scores again is smaller for TFTC than for TC regarding 4 salespersons and more.

Table 1: CPLEX parameter settings.

Models	CPLEX parameter	Parameter value
TD, TC, TFTD and TFTC master:	EpGap	0.0
	WorkMem	12288.0
	Param::Threads	1
	Param::TimeLimit	3600
TFTD and TFTC master:	HeurFreq	-1
	MIPEmphasis	3
	CutsFactor	1.0
TFTD subMILP:	EpGap	0.0
TFTC subMILP:	EpGap	0.0

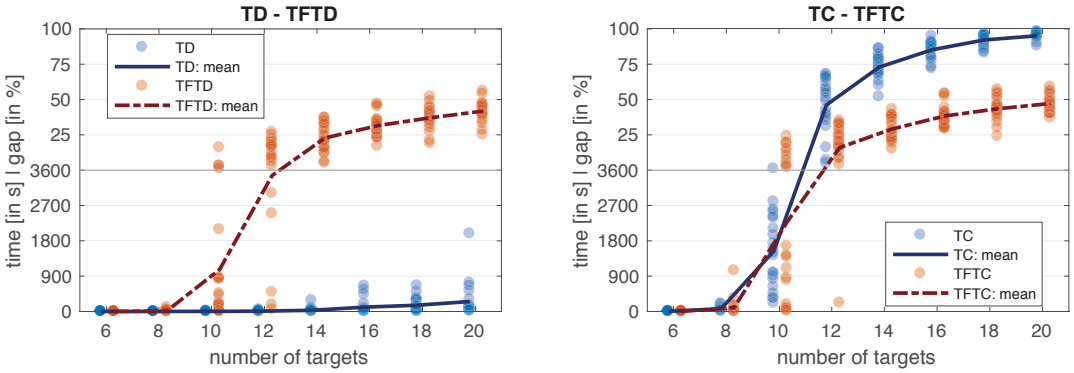

Figure 6: Visualization of the runtimes and gaps of the models TD and TFTD as well as TC and TFTC for instances with an increasing number of targets, 3 salespersons, and a discretization level D16. For instances, that are solved within the time limit of 3,600 sec, the corresponding colored points indicate the exact runtime on the vertical axis. In the other cases, the colored points indicate the gap after a processing time of 3,600 sec. The mean score values of the models are shown as solid and dashed lines.

In the third experimental set the number of targets is fixed to 10 and the number of salespersons to 3, while the discretization level varies from D32 (low) via D16 (medium) to D8 (high). Here, only discrete models are considered, because the other ones are based on continuous time variables. The runtimes and gap values for the models TD and TFTD are visualized in the left picture of Figure 8, the right picture is a zoomed visualization of the runtimes for TD. One can see, that the runtimes for TD increases for a higher number of time steps. This is opposed to the values of TFTD, where the mean is nearly the same for all three discretization levels. For D8 there are 3 out of 21 instances, that can be solved faster with TFTD than with TD.

In the last setting the time limit is set to the real-time limit of 3 sec. The scores adapted to the new time limit are computed for a varying number of salespersons. Here, we set the number of targets to 8 and use the high discretization level D8. The number of salespersons varies from 1 to 6. The obtained results are visualized in Figure 9. Considering TD and TFTD, we observe, that for 1 and 2 salespersons the arithmetic mean of the scores is better for TFTD than for TD. This behavior changes when the number of targets is 3 and more, here, the arithmetic mean of the scores is better for TD. In the continuous case TC has a lower mean for 1 salesperson, for 2 and more salespersons the mean for TFTC is

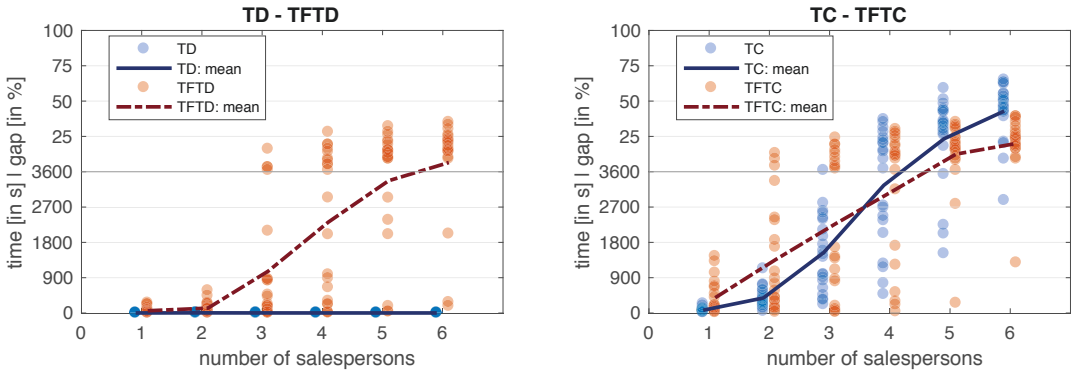

Figure 7: Visualization of the runtimes and gaps of the models TD and TFTD as well as TC and TFTC for instances with an increasing number of salespersons, 10 targets, and a discretization level D16. For instances, that are solved within the time limit of 3600 sec, the corresponding colored points indicate the exact runtime on the vertical axis. In the other cases, the colored points indicate the gap after a processing time of 3600 sec. The mean score values of the models are shown as solid and dashed lines.

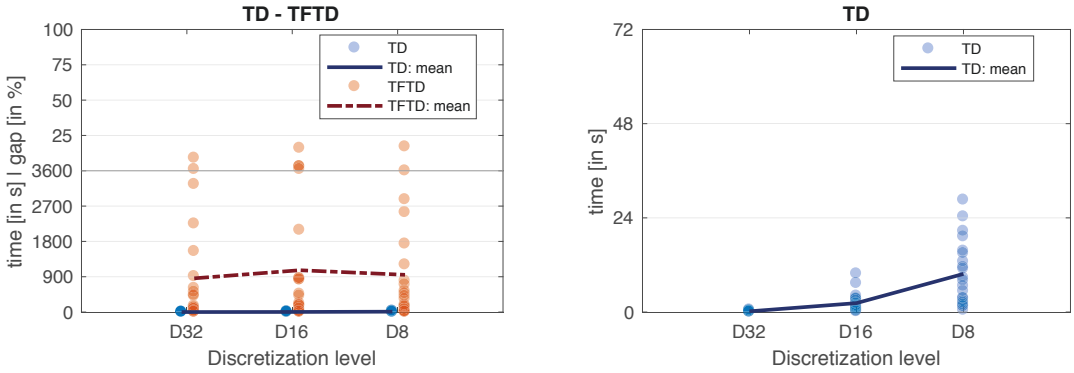

Figure 8: Visualization of the runtimes and gaps of the models TD and TFTD for instances with an increasing discretization levels, 10 targets, and 3 salespersons. For instances, that are solved within the time limit of 3600 sec, the corresponding colored points indicate the exact runtime on the vertical axis. In the other cases, the colored points indicate the gap after a processing time of 3600 sec. The mean score values of the models are shown as solid and dashed lines. The right graphic only shows the runtime on the vertical axis.

better. For 3 and more salespersons TC is not able to solve any of the generated instances to optimality within 3 sec, where TFTC is able to solve some instances to optimality for 3 and 4 salespersons. The averaged scores and runtimes of all four experimental settings are reported in Table 5 and Table 6 in Apendix A.1 on page 101.

Concluding, the results suggest that the continuous problems are more difficult to solve than the discrete ones. Comparing TD and TFTD, the time-free variant can outperform TD only for instances with a small number of targets combined with a small number of salespersons and for the highest discretization. In the continuous case the time-free variant is better than TC for large instances with multiple salespersons. Here, the dual bounds for TFTC are usually better than for TC, which also gives a better gap and thus a better score. The gap value can be obtained from the score by taking $gap = (sc - 1) \cdot 100\%$.

5.3 Computational Results of the Set Partitioning Approach

Here, we have to compute an optimal tour for every possible target subset and one salesperson (TSPMT). We call this step *subset tour generation*. Each of the presented models TD, TFTD, TC and TFTC can be applied to the subset tour generation. Once all subset tours are optimized, their tour lengths are given by the corresponding objective function values. The tour lengths are used as cost coefficients and for a certain number of salesperson the SPP model (49) is solved to obtain the best combination of subset tours.

Each model TD, TFTD, TC, and TFTC is used to compute all optimal subset tours. The reported runtime values include the runtime to generate all optimal tours to all subsets as well as the solution of (49). Since the runtime for optimizing the SPP is negligible, we obey the time limit of 3,600 sec to the subset tour generation. That means, when the time limit is reached, generation is aborted. Note that this may lead to an incomplete set of subset tours, which then, may result in a worse objective function value or an infeasible SPP (49). The CPLEX parameters we use for solving (49) are given in Table 2.

Table 2: CPLEX parameter settings.

Models	CPLEX parameter	Parameter value
Set partitioning model:	EpGap	0.0
	WorkMem	12288.0
	Param::Threads	1
	Param::MIP::Strategy::File	2

For the experiments we create instances with 6, 8, 10 and 12 targets. The number of subsets grows exponentially with the number of targets, thus, instances with at most 12 targets are created. For every target number five different randomly generated instances are produced. For the comparison of the runtimes, the number of salespersons is irrelevant, because solving (49) is very fast and can thus, be repeated for different numbers of salespersons. The highest difference in runtime resulting from solving (49) for different numbers of salespersons over all created instances is at most 0.07 sec. Thus, we only report the results for 2 salespersons . The modeling as a set partitioning (SP) approach requires to solve a

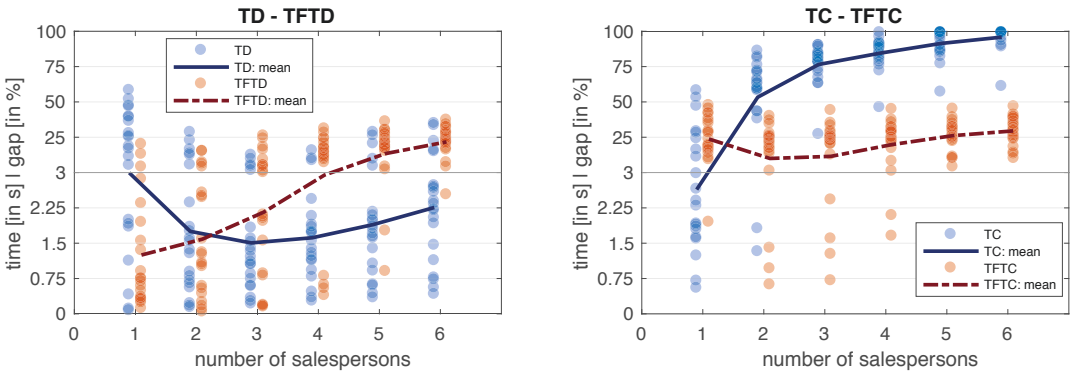

Figure 9: Visualization of the runtimes and gaps of the models TD and TFTD as well as TC and TFTC for a time limit of 3 sec and an increasing number of salespersons. For instances, that are solved within the time limit of 3 sec, the corresponding colored points indicate the exact runtime on the vertical axis. In the other cases, the colored points indicate the gap after a processing time of 3 sec. The mean score values of the models are shown as solid and dashed lines.

sequence of independent optimization problems, thus, there is no final overall gap value. That means, in case the optimal result cannot be computed within the time limit of 3,600 sec, there is no gap value provided and hence, we compare the plain runtimes instead of the scores calculated by runtime and gap.

The arithmetic means of the runtimes for the SP approach with all four models over all discretization levels are reported in Table 7 in Apendix A.1. A comparison of the runtimes between the SP approach with the TD model (TD SP) and with the TFTD model (TFTD SP) is shown in the left picture of Figure 10. The SP approach together with the continuous models (TC SP and TFTC SP) are displayed in the right picture. In both pictures the medium discretization level (D16) is used and the best performance is obtained with TFTD SP. This is similar regarding the other discretization levels. In the right picture it can be seen, that up to 10 targets TFTC SP has lower runtimes than TC SP, however, this changes for 12 targets.

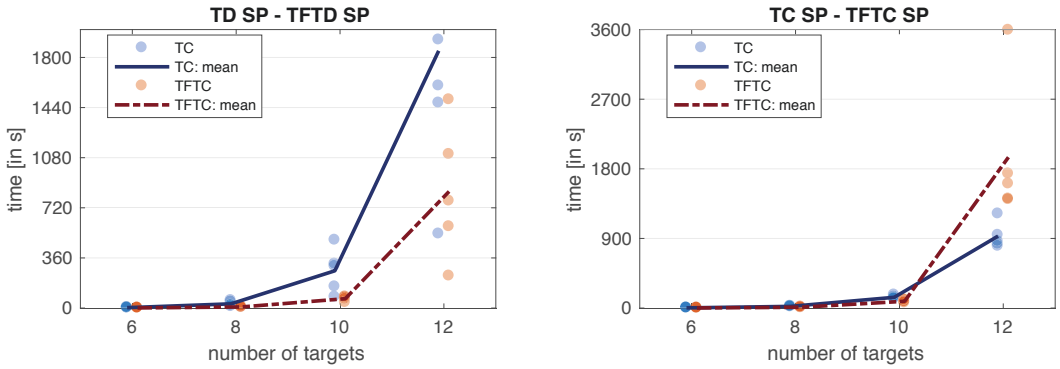

Figure 10: Visualization of runtimes and corresponding arithmetic means for an increasing number of targets modeled with the SP approach. Instances with 6, 8, 10, and 12 targets are considered and the medium discretization level D16 is used. For each number of targets five randomly generated instances are used. The left picture shows results for the models TD with SP and TFTD with SP and the right picture for TC with SP and TFTC with SP, respectively. For a better visualization the results for TD with SP and TC with SP are shifted to the left by a small distance and for TFTD with SP and TFTC with SP to the right, nonetheless they belong to the indicated number of targets given on the horizontal axis.

All generated instances can be solved within the time limit with all four models, except for the instances with 12 targets and discretization level D8. Here, the subset tour generation for TD SP cannot be finished within the time limit of 3,600 sec for 4 out of 5 instances. In each case an incomplete set of subset tours leads to an infeasible SPP (49). The infeasibility is caused, because the incomplete subset tour generation has not considered the last target in any subset within the time limit, which causes an infeasible demand constraint in the SPP 49.

The objective function values obtained in the experiments for D16 are visualized in Figure 11. Since the objective function values for TD SP and TFTD SP are equal as well as for TC SP and TFTC SP, thus, only the values for TD SP and TC SP are shown. The graphic shows, how the values behave, when the number of salespersons is increased. As long as it is advantageous in aggregated tour lengths to apply a new salesperson from the center of the considered space, the objective function value decreases. However, from 4 to 6 salespersons this improvement is very low. Thus, a number of 4 salespersons seems as a good choice.

Comparing the SP results with the results obtained by plain TD, TFTD, TC and TFTC (see Figure 6 in Section 5.2), we can conclude, that the average runtime for TD is better than TD SP. However, this is not the case for TFTD. Since TFTD is fast for 1 salesperson,

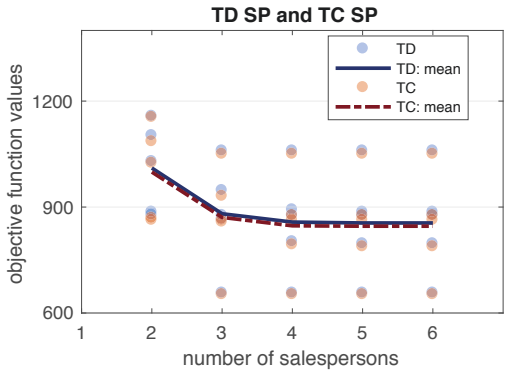

Figure 11: Visualization of the objective function values for the models TD with SP and TC with SP when the number of salespersons increases. We use 2 to 6 salespersons and the discretization level D16 for the time-discrete model TD.

the averaged runtimes for TFTD SP, where the number of salespersons is 1 in the subset tour generation, are much better than for TFTD. This is similar for TC and TFTC, both perform better in the set partitioning approach.

However, TFTD SP cannot outperform TD. Moreover, the SP approach is not applicable for real-time use, due to possible infeasibilities of the SPP, when the subset tour generation cannot be processed completely.

5.4 Computational Results for Non-Linear Trajectories

For the non-linear trajectories we use polynomial functions and trigonometric functions and a combination of both by sum and product. We create 16 non-linear trajectories. Then, instances for 4, 6, 8, 10, 12, 14, and 16 targets are created in a way, that we start with 4 trajectories and gradually add 2 more until we have 16. See Figure 12 for the visualization of all 16 trajectories with a medium discretization. For all generated trajectories, the time steps are distributed in a way that all instances are solvable instances. We use an equidistant sampling of the time steps to apply a time-discrete model variant.

The computations are performed on a single thread, the CPLEX parameter for the MIP gap is set to 0.0. All other CPLEX parameters are used with their default values. In the experiments 2, 4, and 6 salespersons are used and the discretization level varies from D32 to D8. The aim of this examination is to show, that time-discrete models are capable to handle instances with non-linear trajectories. We choose the TD model as a time-discrete model for our experiments, because it is able to cope with instances with a high number of targets and the solution times are reasonable low. All generated non-linear instances are modeled with TD and solved with CPLEX. The obtained runtimes and objective function values are reported in Table 8 in Appendix A.1. The runtime values show, that with TD modeling, the instances can be solved in a reasonable time. For D32 the longest runtime over all instances is below 2.5 sec (this is achieved for 16 non-linear trajectories and 2 salespersons). For the finer discretization levels runtimes up to 409.59 sec are achieved, however, the instance with 16 targets and 6 salespersons can be solved for D16 in 4.21 sec. Moreover, the values show, that instances with a high number of targets and a low number of salespersons are more difficult so solve (in terms of runtimes) than with a high number of salespersons. Considering D16 and D8 and instances with 10 targets and more, the highest runtimes are obtained with 2 salespersons. An optimal solution of the instance with 16 non-linear trajectories in D16 is visualized in Figure 13 with an objective function value of 994.689.

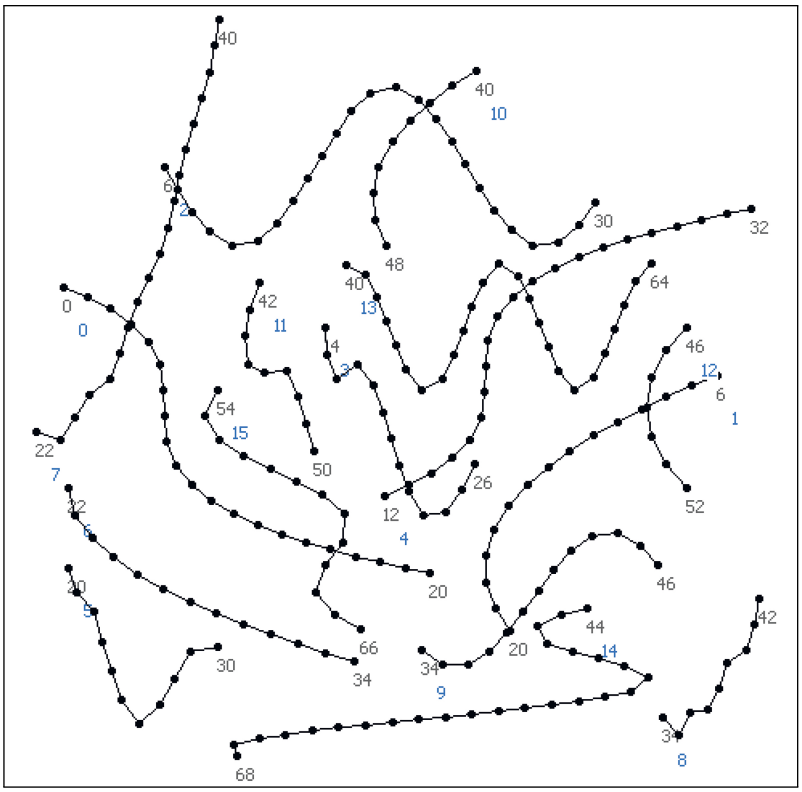

Figure 12: Visualization of an instance with 16 non-linear trajectories. The number of the trajectory is given in blue and the visibility time windows of the trajectories are given in grey numbers. The visualized instance has a discretization level of D16.

Next to discrete models also continuous models can be applied to instances with non-linear trajectories. However in this case, the trajectories have to be approximated by piece-wise linear segments. We use an adaptive sampling and create instances with the same trajectories as for the TD model. The Figure 14 visualizes an instance with 6 targets and 4 salespersons and its optimal solution. The optimal tours of the salespersons are shown by red lines, here, only three tours are needed for the optimal solution.

For the experiments the time-continuous model TC is used. The CPLEX parameter reported in Table 1 on page 50 for TC are used. Instances with 4 and 6 targets and 2 and 4 salespersons are solved with a time limit of 3,600 sec. The results are reported in Table 9 and clearly show, that the computations with the TC model are much more harder and thus need longer runtimes than with the TD model. Only small instances with 4 targets and 2 salespersons can be optimized within the time limit and the runtimes lie in the range of 3 to 35 sec. All other instances exceed the given time limit of 3,600 sec. Although we applied a handover of a MIP start computed with the time-discrete modeling of TD as an attempt to reduce the runtime. The decision variables of the TD solution is given to CPLEX as a MIP start, when optimizing an instance with the TC model. Usually, applying a MIP start reduces the reported gap value. Whether a committed MIP start can be used as an incumbent is reported in the last column of Table 9.

The applicability of the TC model to non-linear trajectories is modest. Another attempt to reduce the problem complexity would be to coarsen the approximation of piece-wise linear segments. However, in our application, this means that the laser would aim at a target with displacement, which might not result in a safe destruction of the threat. Depending on the application parameters as what is an acceptable displacement for the

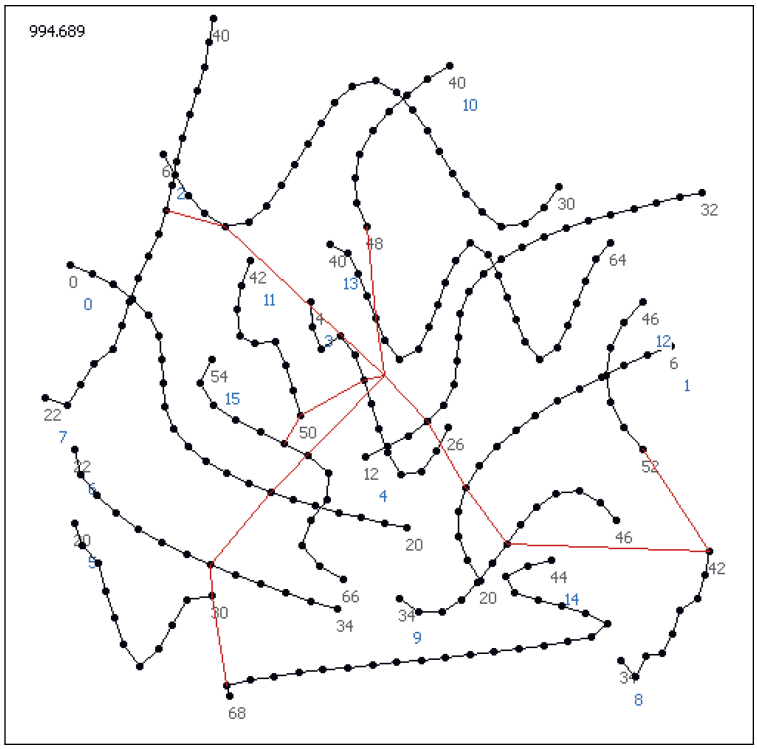

Figure 13: Visualization of an instance with 16 non-linear trajectories and optimal tours (red lines) for 6 salespersons starting in the center. The results are based on the TD model and a discretization level of D16.

laser, the approximation can be computed. However, based on the reported runtimes and the underlying displacement accuracy the TC model is not applicable for real-world instances. The TD model is only applicable with discretization level D32 and some instances also with D16 regarding a possible runtime of at most 3 sec for real-world scenarios. Note, that in the time-discrete case an interception is only possible at the time steps, which coincide with the exact trajectory, while in the time-continuous case an interception is also possible at any interpolated point between two points, where consecutive line segments join.

5.5 Computational Results of the Online MTSPMT

In this section, we regard the MTSPMT as an online problem with a moving horizon approach. That means information about targets is not given in advance, it is revealed step by step. The point of time, where information (trajectory and visibility time window) about a target becomes available, is the moment the target gets visible and thus, enters the considered space. In this online consideration an optimal solution computed at a time t, is based on information about all targets, that are revealed up to time t. When new targets arrive (at a later point in time), a decision has to be taken, how to integrate the new information at runtime. As a consequence, current tours may be canceled in favor of new ones resulting from a re-optimization with the new input data. Thus, the considered time horizon is moving along the time line such that it starts at the point in time, when the (re-)optimization is performed and it end at the last visibility time window endpoint of all not yet visited targets.

In the context of our application (see Section 1.1) a fast optimization algorithm is needed. According to the obtained results for the different modeling approaches in Section 5.2 (see

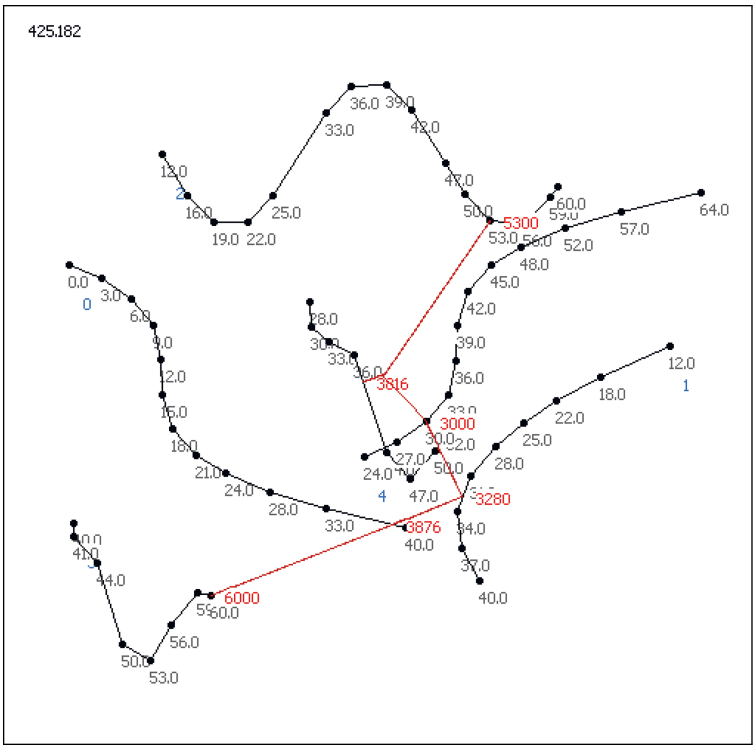

Figure 14: Visualization of an instance with 6 non-linear trajectories, where the target number is given in blue and the endpoints of the piece-wise linear segments are given by black points and grey numbers. A salesperson number of 4 is used, but only 3 salespersons are sufficient and their optimal tours (red lines) start in the center.

also Table 5), the time-discrete model TD combined with the MILP solver CPLEX is used to solve the MTSPMT in an online consideration with a moving horizon, because it is a fast variant for small (< 10 targets) and mid-sized (10 to 16 targets) instances and the fastest variant for large (> 16 targets) instances.

In an optimization step only targets that are visible and have not yet been visited are considered. An important difference to all previously computed instances, is that the salespersons start from different locations in each optimization step. At the point of time, where one or more new targets get visible and a new optimization has to be processed, the current local position of each salesperson is used as their starting (depot) location in this optimization step. After deciding how to integrate the new optimal solution to the current tours of the salespersons, the salespersons move along the updated tours, and thus, their starting locations for the next optimization step change, except for those, who do not have to move. This case, where all the salespersons do not start from the same depot is called *asymmetric*.

Another issue in this context is, that we are faced with infeasible instances. However, if it is not possible at an optimization step to intercept all visible not yet visited targets, we want as the first goal the salespersons to intercept as many as possible. Then, the second goal is to intercept this number of targets by minimizing the total traveled distances. Both these goals have been addressed in Section 3.5, where adaptations to the TD model regarding infeasible instances are presented. We use this adapted TD model as the model for the optimization steps in the moving horizon approach. For the asymmetrical case the model is called *asymmetric adapted TD model* (aaTD).

Given a point in time, where one or more targets are revealed, there are different ways of updating the current tours: either by a heuristic approach or by re-optimization and

integrating the new optimal solution into the current tours. Regarding the latter one, a natural approach for the design of online algorithms is the REPLAN strategy. This strategy is more a general principle than an algorithm for this particular problem, see GRÖTSCHEL et al. [Grö01a; Grö01b].

Strategy REPLAN: Follow the current tours. Whenever a new target becomes available, a re-optimization (aaTD model) is performed based on the current positions of the salespersons and including all not yet intercepted targets. Then, all salespersons follow the new tours.

For salespersons, who are moving towards a target, this may mean, that they have to immediately abort their current tour, turn, and head to another target. At any point in time REPLAN tries to be as close as possible to the global offline optimum. We also want to present another strategy, which is similar to the REPLAN strategy. It is called IGNORE strategy and presented in [Grö01b] for problems as the online TSP: a salesperson completes its current tour, if it is performing one, and upcoming targets are temporarily ignored and stored in a buffer for re-optimization afterwards. An adapted version for the online MTSPMT is given in the following.

Strategy IGNORE: Whenever a new target becomes available, all salespersons follow their current tours until a salesperson reaches a target, the upcoming targets are temporarily ignored and stored in a buffer. At that moment, where one of the salespersons reaches a target, re-optimization starts with all not yet visited targets and based on the current positions of the salespersons.

In case a new target becomes visible and there is a non-active salesperson (with no tour) or a salesperson, who just arrived at a target re-optimization starts right away. Then, the IGNORE strategy acts like the REPLAN strategy.

We use both promising strategies REPLAN and IGNORE for our computational experiments. Randomly generated instances with 8 to 20 targets, 2 and 4 salespersons, and medium discretization level D16 are created. The parameters for TD reported in Table 1 are used for the computations. We want to examine the performance of the online strategies in terms of how many targets could not be intercepted at all, which is called the number of misses. To obtain distinguishable and descriptive results we use very short trajectory lengths: 50-150 length units. Hence, with very short trajectories it is challenging to intercept all targets.

For instances with 2 salespersons (lasers), the center of the area $[250, 250]$ is used as the depot for both salespersons. Having 4 salespersons, regarding the application of protecting an area we assume the salespersons to be spread over the whole area and thus, we divide the area in four equal squares and put a salesperson in each center of the squares. Hence, the following positions are given as the four different depots of the salespersons: $[125, 125]$, $[375, 125]$, $[125, 375]$, and $[375, 375]$.

The results of the computations are visualized in Figure 15 for 2 salespersons and Figure 16 for 4 salespersons. The values show that nearly 12% of all instances with 2 salespersons can be solved without missing any targets with the REPLAN strategy, only slightly over 8% can be solved with the IGNORE strategy, respectively. However, using 4 salespersons the REPLAN strategy can solve over 76% of the instances without missing any targets and the IGNORE strategy over 73%. While for instances with 4 salespersons the number of misses is very similar for both online strategies, the REPLAN strategy performs better than IGNORE regarding instances with 2 salespersons. The reason is, that with 4 salespersons IGNORE often acts like REPLAN. The complete results are listed in Table 10 in Apendix A.2.

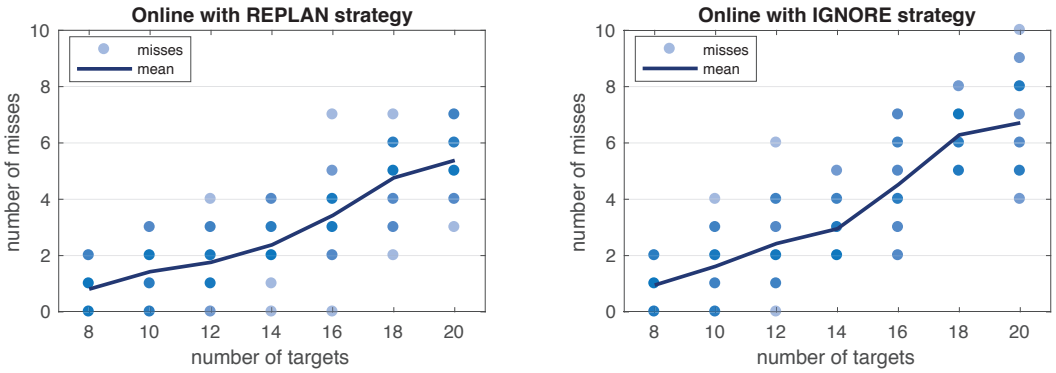

Figure 15: Visualization of the number of misses and its arithmetic mean for the online strategies REPLAN (left) and IGNORE (right). The arithmetic mean is calculated over 21 randomly generated instances with 2 salespersons and discretization level D16. The optimization problem is modeled with the TD approach. The moving horizon approach is used to simulate the online consideration of the MTSPMT.

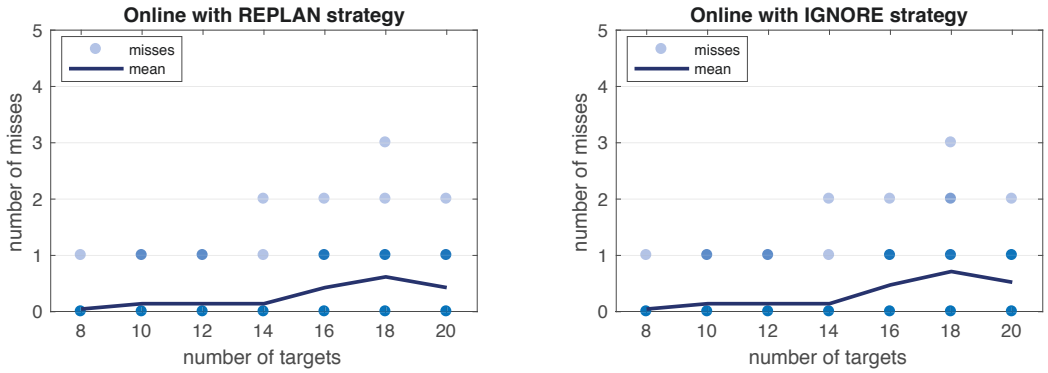

Figure 16: Visualization of the number of misses and the arithmetic mean for the online strategies REPLAN (left) and IGNORE (right). The arithmetic mean is calculated over 21 randomly generated instances with 4 salespersons and discretization level D16. The optimization problem is modeled with the TD approach. The moving horizon approach is used to simulate the online consideration of the MTSPMT.

For a comparison of the objective function values between the online strategy REPLAN/IGNORE and the offline optimization, we use all instances with 2 salespersons, where the number of misses is zero. In case of misses in the online computation the objective function values are not comparable to the offline ones even if the penalty term is neglected. For the REPLAN strategy we have 17 instances and for the IGNORE strategy 12 instances, respectively. These instances are additionally optimized in an offline fashion, where all input information is given in advance. Then, the ratio of online objective function value and offline objective function value is computed. For REPLAN we obtain a worst ratio of 1.55 and an averaged ratio of 1.19 (both rounded to the second position after decimal point). For IGNORE we have a worst ratio of 1.55 and an averaged ratio of 1.20 (both rounded to the second position after decimal point). All computed objective function values are visualized in Figure 17, left for the REPLAN strategy and right for the IGNORE strategy. For a better visualization, the instances are arranged according to the offline objective function value from low to high. All computational results are reported in Table 11 and Table 12 in the Apendix A.2.

Concerning the averaged and worst case ratio of the online/offline objective function

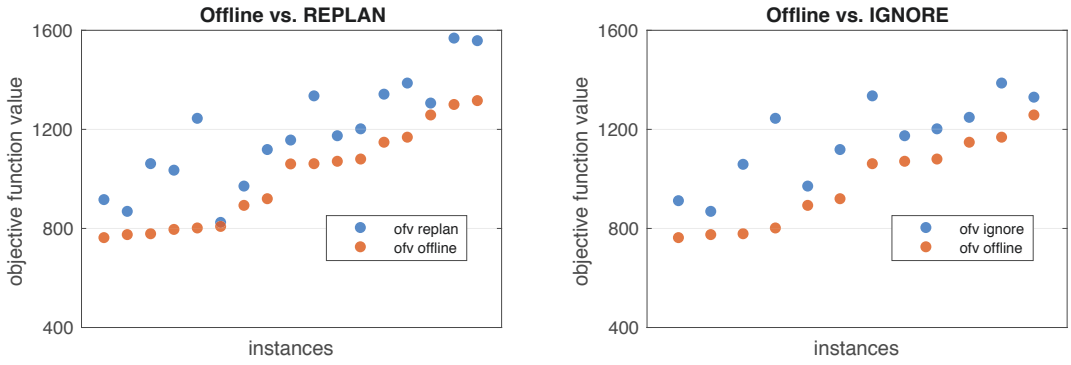

Figure 17: Visualization of the objective function values (ofv) for instances optimized in an offline consideration and in an online consideration with the REPLAN strategy (left) and the IGNORE strategy (right). The instances with 8-16 targets for REPLAN and with 8-12 targets for IGNORE, 2 salespersons, and discretization level D16 can be solved without any misses. The optimization problems are modeled with the TD approach.

values the quality of the strategies REPLAN and IGNORE seems to be equal. A theoretical overview about online optimization and the methods of how to measure the quality of online strategies is given in Section 6.

Online MTSPMT with Min-Time

For the last computational experiments we investigate the influence on the number of misses when the objective function is changed from minimize total traveled distances, which we call *min-dist* in the following, to minimize earliest possible interception, which is called *min-time*, respectively. Here, also the TD model (27) is used for modeling, however, the objective function is replaced by the min-time objective function given as

$$\min \sum_{k \in \mathcal{W}} \sum_{(i,j,\theta,\lambda) \in \tilde{\mathcal{A}}} \lambda \, x_{i,j,k}^{\theta,\lambda}. \tag{67}$$

Thus, we minimize the sum of all earliest possible interceptions. Each time a re-optimization is performed, the TD model 27 with the objective function (67) is used and the current positions of the salespersons serve as their depots for this particular re-optimization step.

Here, we used the same test instances with 8 to 20 targets and 2 and 4 salespersons as before. Again, for instances with 2 salespersons both pursuers start from the center of the protected area (depots). In the other case, the depots of 4 salespersons are located at the positions $[125, 125]$, $[375, 125]$, $[125, 375]$, and $[375, 375]$, note that we consider a square with 500 length units. For each target number we have 21 test instances and optimization is performed with the online strategies REPLAN and IGNORE. The resulting numbers of misses are visualized in Figure 18 for 2 salespersons and in Figure 19 for 4 salespersons.

The computational results show that nearly 15% of all instances with 2 salespersons can be solved without missing any targets with the REPLAN strategy, over 12% can be solved with the IGNORE strategy, respectively. However, using 4 salespersons the REPLAN and the IGNORE strategy can solve over 78% of the instances without missing any targets. For 2 salespersons, the averaged values of the misses for the IGNORE strategy are slightly higher than for the REPLAN strategy, while with 4 salespersons the arithmetic means of the misses are equal in the exact same manner for both online strategies. The computational results are reported in Table 13 in Apendix A.2.

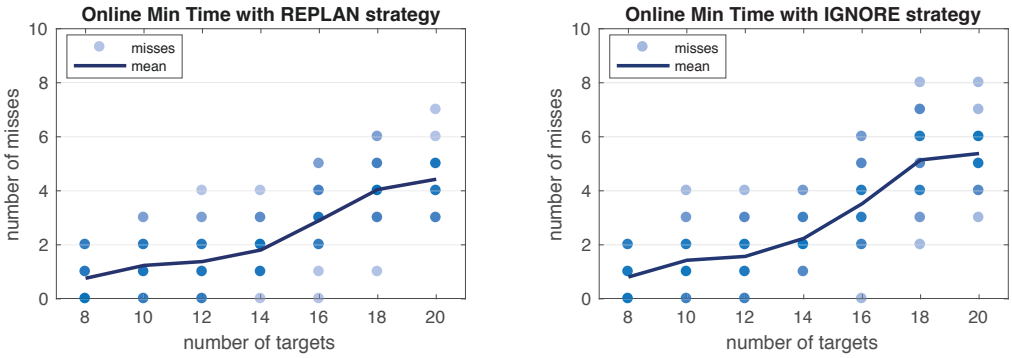

Figure 18: Visualization of the number of misses and the corresponding arithmetic mean for the online strategies REPLAN (left) and IGNORE (right) with 2 salespersons. The arithmetic mean is calculated over 21 randomly generated instances with 2 salespersons and discretization level D16. The optimization problem is modeled with the TD approach and the min-time objective function.

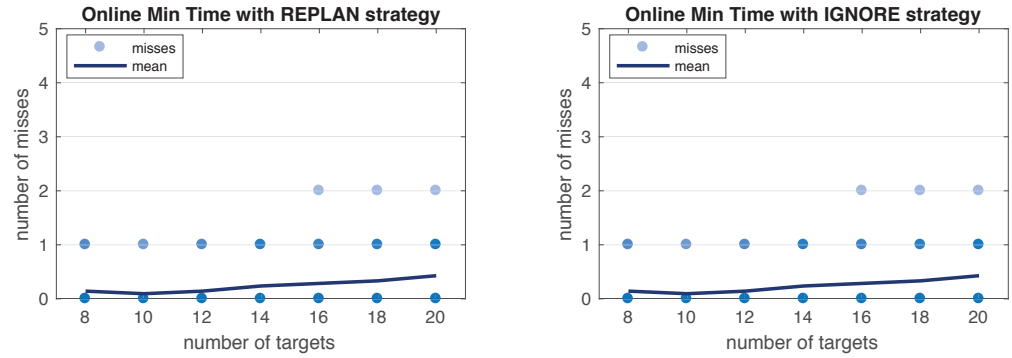

Figure 19: Visualization of the number of misses and the corresponding arithmetic mean for the online strategies REPLAN (left) and IGNORE (right) with 4 salespersons. The arithmetic mean is calculated over 21 randomly generated instances with 4 salespersons and discretization level D16. The optimization problem is modeled with the TD approach and the min-time objective function.

If we compare the number of misses with the results obtained with the objective function min-dist (see Figure 15 and Figure 16), we can conclude, that there are less misses with objective function min-time. While for a small number of targets (8 to 12) the results are very similar, for more than 12 targets, the min-time objective function causes a smaller arithmetic mean of misses than the min-dist objective function no matter what online strategy is used. In a real world online situation it is advantageous to use the min-time objective function to catch as most targets as possible. Obviously, if the traveled distance of all lasers is an issue the min-dist objective function shall be used.

6 Online MTSPMT

In this chapter we will give a deeper insight into the MTSPMT from the perspective of the real application, which describes an online situation. In contrast to the former offline point of view, where the entire input sequence of the targets is given beforehand, the information is now revealed over time, in an online fashion. Regarding the application of the MTSPMT, targets become known, when they are close enough in order to be detected by the radar system. In this sense multiple targets can occur to the radar detection simultaneously and in successive steps. We remind, that a defense decision has to be taken immediately, which means before complete information about the targets is available. In fact, it is necessary to produce a part of the solution with an *online algorithm* as soon as a new target becomes known. According to Grötschel et al. [Grö01b] an algorithm *runs online* "if it makes a decision (computes a partial solution) whenever a new piece of data requests an action".

A general online routing problem is given by a sequence of requests (points in a metric space combined with release times) and a server with the task of serving each request at their release time or later. Additional restrictions may be defined. In the notation of the MTSPMT, a salesperson corresponds to a server and a target to a request. We call the online variant of the MTSPMT the *online multiple traveling salespersons problem with moving targets* (OLMTSPMT).

For online problems there are several modeling approaches, which describe the way in which the input information becomes available to an online algorithm. The two most common online paradigms are the *sequence model* and the *time-stamp model*, see for example [Grö01b]. In the sequence model, a finite and ordered requests sequence is presented one-by-one to an online algorithm. The requests must be served in the order of their occurrence. When serving a request, an online algorithm does not have any knowledge about future requests, not even if there are future requests. An action made by the online algorithm of how to serve the request is irrevocable. Only after serving a request, the next request in the sequence becomes available. The well-known *k-server problem*, see for example Koutsoupias [Kou09], is an example of an online problem in the sequence model. Here, an online algorithm must control the movement of a set of k servers, located at points in a metric space, and handle requests in the order of their presentation also at points in a metric space. At revelation of a request, the algorithm has to decide which server to send immediately to the requested point with the goal to keep the total distance of all servers as small as possible.

In the time-stamp model requests are presented over time at certain *arrival* or *release times*. An online algorithm must determine an action at a time t based on all requests, that have been released up to time t and have not yet been served. Future requests are unknown as well as if there will be any. A request can arrive at any time. The difference to the sequence model is, that the online algorithm is allowed to wait, to postpone and to revoke decisions as long as they have not been executed. Time advances while decisions are made and executed. The time-stamp model is very natural for online routing and scheduling problems, because waiting and revoking decisions is allowed and requests can be served in an order different from the order of their release (Lipmann [Lip03]). The online machine scheduling problem, where jobs arriving over time are distributed to a number of machines, is formulated in the time-stamp model since concepts like postponing and execution in an order different to the release order are essential. Another example for the time-stamp model is the online traveling salesperson problem, which is discussed in more detail in Section 6.2.

6.1 Competitive Analysis

Competitive Analysis is the most widely accepted systematic to measure the performance of online algorithms. Alternative measures are addressed at the end of this section. The first theoretical investigations in this context started when Sleator and Tarjan [Sle85] suggested to compare an online algorithm with an optimal offline algorithm. In standard competitive analysis, see Borodin and El-Yaniv [Bor98], the *competitive ratio* is calculated as the worst case ratio over all possible sequences of requests by the objective function value of an online algorithm and the objective function value produced by an optimal offline algorithm using the same input sequence. Here, we stick to deterministic algorithms (actions of the algorithm are uniquely determined by the input of the instance).

Let ALG be a deterministic online algorithm and σ a sequence of requests. Denote by $Z^{ALG}(\sigma)$ the cost incurred by ALG on σ and denote by $Z^{OPT}(\sigma)$ the cost of an optimal offline algorithm OPT on σ. The optimal offline algorithm OPT knows the complete request sequence in advance and thus, can serve it with minimal cost.

Definition 6.1. Competitive Algorithm.
Let $\varrho \geqslant 1$ be a real number. The deterministic online algorithm ALG is said to be ϱ-competitive, if

$$Z^{ALG}(\sigma) \leqslant \varrho \, Z^{OPT}(\sigma) \tag{68}$$

for all request sequences σ. The *competitive ratio* of ALG is the infimum value over all such ϱ that ALG is ϱ-competitive.

Competitive analysis of online algorithms can also be interpreted as a two-player game between an *online player* and a malicious *offline adversary*, see for example [Bor98] and [Lip03]. The adversary generates a request sequence, that both players have to process. The online player uses an online algorithm to determine the actions for the requests presented step by step by the adversary. Here, we only concentrate on deterministic online algorithms. The adversary based on the knowledge of the online strategy used by the online player, generates the input so as to maximize the competitive ratio. At any point in time the online player only knows requests presented so far. After the last request is presented, the adversary has to process the same sequence of requests, however, he knows the complete sequence in advance and uses an optimal offline algorithm. The aim of the adversary is to increase the costs of the online player compared to his own costs.

Even though competitive analysis is a standard tool in measuring online algorithms, it has also been criticized as being too pessimistic. We should be careful to compare two online algorithms for the same problem only based on their competitive ratio. The competitive ratio is defined as a worst case ratio. An algorithm that performs well in practice is only judged on his bad performance on a typical worst case input sequence. Another reason why this measurement can be viewed as not realistic, is the enormous power of the adversary.

To overcome the pessimistic view of competitive analysis various extensions and alternatives have been proposed in the literature. Some of these concepts are introduced in the following. In *comparative analysis* the adversary is only allowed to choose from a restricted class of algorithms as in Koutsoupias and Papadimitriou [Kou94]. Blom et al. [Blo00] introduce the concept of a *fair adversary*, where the movement of the adversary is restricted. In this approach it is no longer possible for an adversary to move to a point, where it knows a request will appear without revealing it to the online player before reaching the point themselves. Thus, the movements of a fair adversary is restricted to the convex hull of former presented requests. The authors show that a fair adversary is weaker and thus, lower

competitive ratios for the online traveling salesperson problem can be obtained. Another approach is to give more power to the online player than to the adversary as in the concept of *resource augmentation*, see for example Kalyanasundaram and Pruhs [Kal00], Phillips et al. [Phi02], and Roughgarden [Rou20]. For example the online player is given more or faster processors than the adversary. Depending on the considered problem another resource can be the capacity of the server or the cache size. A mechanism to directly compare two online algorithms is the Relative Worst Order Ratio, see Boyar and Favrholdt [Boy07]. Here, a worst case request sequence is considered and the performances of the two algorithms on their worst permutation of this sequence is compared.

It could be proven, that all extensions and alternatives to competitive analysis are useful for some specific problem and obtain significant results. However, none of these alternatives could replace competitive analysis as the standard tool for evaluating and analysing online algorithms [Grö01b]. In this thesis, we use competitive analysis as the standard tool of measuring the performance of online algorithms.

6.2 The Online Traveling Salesperson Problem

The online traveling salesperson problem (OLTSP) is a problem, that is well known in online optimization and to which the time-stamp model is applied to. The requests (cities in the classical TSP) are presented as points in a metric space over time, the time a request pops up is called its release time. There is one server (salesperson in the classical TSP) starting from the origin and following a tour to visit the presented requests at a time after their release. While the server is moving, new requests may arrive and the tour of the server may be adjusted. After the last request is visited the server has to move to the depot or not. In case the server has to return to the depot the problem variant is called closed OLTSP or homing-OLTSP. In the other case there is no such requirement and the tour can have a free end, this variant is called open OLTSP or nomadic-OLTSP. Usually, the server has unit speed and the objective is to minimize the time until all requests have been served. In case of the closed variant it is the time when the server, after having served all requests, has returned to the origin. This objective function is called the *makespan*.

The OLTSP is well studied in the literature, see for example [Apr09; Aus01; Bje17; Lip03]. There is also research about generalizations of the OLTSP like flexible service with penalty, and deadlines [Gut06; Jai11; Wen15]. Gutiérrez et al. [Gut06] address the whack-a-mole computer game, where moles (requests) appear at certain locations and must be whacked (served) by a hammer before they disappear under ground. The goal is to serve as many requests as possible, while all requests have the same visibility time window. The problem is studied on the real line and on the uniform metric space. The OLTSP with service flexibility is the main focus in Jaillet and Lu [Jai11]. Here, the server can decide which request to serve and has to visit all accepted requests. The objective is to minimize the completion time to serve all accepted requests plus the sum of the penalties of all rejected requests. Wen et al. [Wen15] address the OLTSP with deadlines and service flexibility and different objectives. They show, that no deterministic or randomized online algorithm can achieve a constant competitive ratio for this problem on general metric spaces, not even on the half line. The competitive ratio is unbounded in the number of requests.

The OLMTSPMT is very similar to the OLTSP. With the following conversions an instance of the OLMTSPMT can be transformed into an instance of the open OLTSP. We set the speed of the targets to zero, extend all deadlines (second value of the visibility time intervals) to the considered time horizon, use only one salesperson, and minimize the makespan, thus, we obtain an instance of the open OLTSP.

Definition of the OLTSP

First, we define the considered problem in the time-stamp model. An instance of the *online traveling salesperson problem* is given by a metric space $M = (X, d)$, a special point $O \in X$ (origin), and a sequence of requests $\sigma = r_1, \ldots, r_n$ with $n \in \mathbb{N}$. Each request is a pair $r_i = (x_i, t_i)$, where $x_i \in X$ is the point in the metric space, where the request has to be served and $t_i \geqslant 0$ is the time at which the request is presented. We assume, that the sequence is ordered such that $t_i \leqslant t_j$, if $i \leqslant j$. A server, starting at the origin O at time 0, moves at unit speed and has to serve all requests, where each request cannot be served earlier than the time it is released. Such a tour of the server is called a feasible online/offline solution. In case of the open OLTSP the cost of the tour is the time the server served the last request. For the closed OLTSP the cost of the tour is the time the server has returned to the origin after having served the last request.

Online Algorithms

An online algorithm of the OLTSP has to decide on the behavior of the server at any moment t in time based on requests presented until t only. Future requests are unknown, the algorithm does not even know, if there are future requests. This can lead to a difficult situation in algorithm design for the closed variant. Assume, that the online server has served the last presented request, then the server can immediately return to the origin or wait for a certain period of time. If the server directly moves back to the origin, a new request may arrive and his way back to the origin was in vain. Otherwise, waiting a long time and then returning to the origin leads to an unnecessary expense. How long should a server wait?

The algorithm also determines the order in which the requests are to be served, this is opposed to problems, which are modeled in the sequence model (e.g., k-server problem). We note, that the objective of the OLTSP is to minimize the makespan, which is different from the objective in the classical TSP and in the MTSPMT, where the length of the tour (tours) are to be minimized. Generally the makespan can be computed from the traveled distance with unit speed plus the time the server was idle.

We confine ourselves to using deterministic online algorithms. An algorithm is said to be deterministic if its actions uniquely depend on the input. An online player that rather uses a randomized strategy can apply random moves, e.g., every time a decision has to be made, he can make a random choice out of a set of algorithms.

A deterministic online server can move at a speed ranging from zero to unit speed. He can change direction and speed at every time and every place. Especially, the server can be idle and wait while there are requests yet to be served. Blom et al. [Blo00] introduced a particular class of deterministic online algorithms for the OLTSP, which is called *zealous algorithms*. Intuitively, a zealous server, a server operated by a zealous algorithm, never waits, when there are yet unserved requests and he has to move directly to an unserved request without any detour.

The OLTSP on the Real Line

The OLTSP on the real line is not a trivial problem and comes up in one dimensional collection or delivery problems, as for example robotic welding/screwing/depositing material or in horizontal or vertical item delivery systems. The server is initially located at the origin. It can move either to the negative or to the positive direction or it can wait at its current position.

The OLTSP on the line has been studied over the last two decades [Apr09; Aus95; Aus01; Blo00; Jai11; Lip03; Wen15]. However, quite recently a tight analysis in terms of competitive ratio of the OLTSP on the line was presented by Bjelde et al. [Bje21]. These latest results and the former bounds found in the literature are summarized in Table 3.

Table 3: Results for the OLTSP.

OLTSP	open		closed	
on the line	lower bound	upper bound	lower bound	upper bound
latest results	2.04	2.04		1.64
former results	2	2.33	1.64	1.75

The former results of Table 3 were presented by Ausiello et al. [Aus95; Aus01]. For the closed variant the gap between lower and upper bound could be closed by providing an upper bound of 1.64 by Bjelde et al. [Bje21]. For the open OLTSP [Bje21] presented a lower and an upper bound of 2.04 and thus provided a tight analysis.

Note, that the bounds for the closed variant are smaller than the bounds for the open variant. The reason for this, is the additional information in the closed OLTSP for the server to finish its tour at the depot (origin). This information can be used in the way that certain requests can be left for serving to the server, when the server is moving to the origin. Then it can intercept all requests, that lay between server and origin on its return.

The lower bounds achieved on the real line for the open and closed OLTSP also apply to general metric spaces. For the open OLTSP Ausiello et al. [Aus01] present a 2.5-competitive algorithm and for the closed variant they introduce a 2-competitive algorithm. Jaillet and Wagner [Jai08] also address the closed OLTSP and present a 2-competitive online algorithm, where general precedence and capacity constraints can be added to the problem. Another 2-competitive online algorithm for the multiple closed OLTSP is presented. Both algorithm are generalizations from the Plan-At-Home algorithm [Aus01]. An online algorithm for the non-negative part of the line is proposed by Blom et al. [Blo00] and achieves a competitive ratio of 1.5. However, their main focus is on the presentation of a *fair adversary*. In this setting the optimal offline algorithm is not allowed to travel outside the convex hull of the requests known so far. The authors give a 1.28-competitive online algorithm for the non-negative real line and a fair setting.

6.3 The Online Moving Target Traveling Salesperson Problem

We generalize the OLTSP by assigning a constant speed to the requests, which are called targets in this context. The targets continuously move in a metric space and the server (pursuer) has to catch them in any order. Obviously, the maximum speed of the server needs to be greater than the speed of the targets, otherwise there is no chance for the pursuer to catch a target, that is heading away from the server. The server can change direction anytime and on any place. We call this problem the *online moving target traveling salesperson problem* (OLMTTSP).

The differences of this problem to the OLMTSPMT are, that we use only one server (instead of multiple) and the visibility of any target is only determined by its release time and the end of the considered time horizon, there is no deadline.

In the following, we concentrate on the OLMTTSP on the real line ($\mathbb{R}$). Similar to the theoretical study in terms of competitive ratio of the OLTSP on the line in Table 3, we

address the OLMTTSP on the line and provide some new results on competitive ratios. We provide a lower bound of $2 + \frac{1}{a-1}$ for the competitive ratio, where a is the ratio of the maximum speed of the server and the constant speed of the targets. Additionally, an algorithm for the OLMTTSP on the line is presented and another known algorithm for the OLTSP on the line is adapted to moving targets. The competitive ratios of both algorithms provide upper bounds for the OLMTTSP on the line. However, the quality of the bounds depends on the value of a. This means, that for $a \leqslant 15$ the bound from the new algorithm is best and for $a > 15$ the bound derived from the adjusted algorithm is best.

After a literature overview of the OLMTTSP on the line the definition of the OLMTTSP is given followed by the theoretical study of competitive analysis. The Moving Target TSP was already considered by Helvig et al. [Hel03], however, not as an online problem. Targets, present right from the beginning, move with different speed values. A server chasing the targets has to return to the origin afterwards. The authors present an exact algorithm, which runs in $O(n^2)$ time, where n is the number of targets.

The Model of the OLMTTSP on the Real Line

An instance of the OLMTTSP is given by the real line $\mathbb{R}$, a special point $O \in \mathbb{R}$ (origin), and a sequence of targets $\sigma = r_1, \ldots, r_n$ with $n \in \mathbb{N}$. Each target is a tuple $r_i = (x_i, t_i, d_i)$, where $x_i \in \mathbb{R}$ is the point, where the target appears, $t_i \geqslant 0$ is the time at which the target is presented and $d_i \in \{-1,1\}$ is the direction in which the target is moving on the line. Throughout, we refer to the negative direction (-1) as *left* or *left-hand side* and to the positive direction (1) as *right* or *right-hand side*. A server starting at the origin O at time 0 moves at most with unit speed and has to catch all targets, where each target cannot be intercepted earlier than the time it is released. The ratio between the maximum speed of the server and the constant speed of the targets is defined as a, where $a \in \mathbb{R}$ and $a > 1$. Thus, the constant speed of the targets is given by $1/a$. Having this, the pursuer has a greater maximum speed and is thus able to catch the targets. The objective is to minimize the makespan. The concept of the open and closed variant equally applies here as for the OLTSP. However, the open variant, where the server does not need to return to the origin is more related to the OLMTSPMT and thus, is solely used hereinafter.

A Lower Bound for the Open OLMTTSP on the Real Line

The first result provides a lower bound for the open OLMTTSP. We show, that no online algorithm can have a competitive ratio smaller than $2 + \frac{1}{a-1}$ on the real line, where the constant speed of the targets is $\frac{1}{a}$, $a > 1$. Provided is a sequence of targets, that also Ausiello et al. [Aus01] used for a lower bound of 2 for the open OLTSP on the real line.

Lemma 6.2. A server at position x moves with unit speed towards a target, that is at the same time at position y on the real line. The target is moving away from the server with a constant speed of $\frac{1}{a}$, $a > 1$. Then the target has moved a distance of $z = \frac{|y-x|}{a-1}$ at the moment the server intercepts the target.

Proof. Since server and target meet at the same time, we have, that the distance traveled by the server divided by his speed is equal to the distance traveled by the target divided by the speed of the target:

$$\frac{|y-x|+z}{1} = \frac{z}{\frac{1}{a}} \quad \Leftrightarrow \quad \frac{|y-x|}{a-1} = z.$$

□

Theorem 6.3. Any ρ-competitive online algorithm for the open OLMTTSP has $\rho \geqslant 2+\frac{1}{a-1}$. This lower bound is tight on the real line.

Proof. The completion times for the online server and the offline adversary are called Z^{OL} and Z^* respectively. Consider the problem on the real line with 0 as the origin. At time $t = 1$ a target r_1 is released, depending on the position of the server. If the server is on the negative side the target is presented at position 1 and moving away from the server $r_1 = (1, 1, 1)$. In the other case the target is given at position -1 and heading away from the server $r_1 = (-1, 1, -1)$. Without loss of generality we assume the first case. The server starts moving towards the target at time 1. Since the server is moving with unit speed, it is not earlier than at time 2 at the release position of the target. Then, to reach the interception point the server needs additionally at least a distance of $\frac{1}{a-1}$ (see Lemma 6.2). Thus, the total time the online server needs is at least $Z^{OL} \geqslant 2 + \frac{1}{a-1}$. When the server is at the origin at time 1, it is exactly $2 + \frac{1}{a-1}$. With the previous knowledge of all targets the optimal offline adversary completes at time $Z^* = 1$, exactly when the target is released. Thus,

$$\rho \geqslant \frac{Z^{OL}}{Z^*} \geqslant 2 + \frac{1}{a-1}.$$

Note, that if the release of the target at position 1 is earlier or later than 1 the competitive ratio is decreased. □

Consider the case for $a \to \infty$, then the targets do not move at all and the limit of our lower bound $\lim_{a\to\infty} \left(2 + \frac{1}{a-1}\right) = 2$ matches the lower bound proven by Ausiello et al. [Aus01]. Although, we restricted ourselves to a deterministic algorithm, the above theorem 6.3 also holds for randomized online algorithms.

The NEF Algorithm for the Open OLMTTSP on the Real Line

In this section we propose and analyze an intuitive algorithm for the open OLMTTSP on the real line. As we will see, the algorithm belongs to the class of zealous algorithms, the server keeps moving as long as there are yet unvisited targets. At any moment in time we denote by *leftmost extreme*, the target, whose current position is furthest away on the left side of the server and by *rightmost extreme*, the target, whose position is furthest away on the right side of the server. There are two things to note. Firstly, targets, that move towards the server are not complicated, the server could simply stay where he is to catch them. Secondly, the targets that are moving away from the server have to be caught first, otherwise the distance between server and target increases. Considering both rays of the line from the location of the server, the server has to move to the nearest extreme first.

The algorithm is called *Nearest Extreme First* (NEF) and performs as follows.

NEF: Let S^A be the set of unvisited targets, that are moving away from the server and S^T be the set of unvisited targets, that are moving towards the server. At the beginning, the online server is located at the origin 0. Since targets, that are moving towards the server, can be caught simply by doing nothing, the server moves to intercept targets traveling away from him first. Thus, the leftmost and rightmost extreme targets of S^A are determined. The server catches the *nearest extreme first*. The

nearest extreme is the extreme (either leftmost or rightmost one), whose current distance to the server is smallest. After intercepting the nearest extreme NEF moves to catch the other one. All targets in between (no matter what direction they move) are intercepted on the fly. If there are no more targets moving away ($S^A = \varnothing$), then the targets moving towards NEF have to be caught. This can simply be done by waiting. However, to reduce completion time the server first moves to the nearest extreme of S^T and afterwards to the other extreme. Each time a new target enters the system, it must be checked if the current tour has to be updated. Cases, which force an update of the tour:

a) The new target is moving away from the server and is an extreme target of S^A.
b) The new target is moving towards the server, $S^A = \varnothing$, and the new target is an extreme target of S^T.

In all other cases (target is moving away, but is not an extreme target; target is moving towards NEF and $S^A \neq \varnothing$; target is moving towards NEF, $S^A = \varnothing$ and target is not an extreme target) the server can continue his tour as before. Targets on his way can be caught on the fly.

This algorithm achieves a competitive ratio, that depends on the speed ratio of the server and the targets. The competitive ratio is shown in Theorem 6.5. Here, we consider 0 as the origin, where the online server and its adversary (offline server) are located at time $t = 0$. Both server move with unit speed while the targets move at a speed of $\frac{1}{a}$, $a > 1$. For convenience we use distances traveled by servers also as times and vice versa. The following Theorem 6.5 is related to Theorem 6.1 from Ausiello et al. [Aus01], which gives a 7/3 competitive ratio for the algorithm "serve Extreme Nearest to the Origin first (ENO)" for the open OLTSP on the line. This algorithm relates the extreme requests to the origin, as opposed to the NEF algorithm, which relates the extreme targets to the current position of the server. As it turns out for the static case of the open OLTSP on the real line, ENO is slightly better than NEF.

Before the presentation of Theorem 6.5, a relevant lemma is mentioned. It is needed to simplify the lower bound of the completion time of the offline adversary within the proof of the theorem.

Lemma 6.4. Let r_y be a target and s be any server that moves with at most unit speed. At any time $\tau > 0$, we denote the position of r_y by $p_y(\tau)$ and its distance to the server s by $y(\tau)$. If $|p_y(\tau)| \leqslant \tau$, we have $\tau \geqslant \dfrac{y(\tau)}{2}$.

Proof. For the server s starting in the origin at time 0 we have at time τ, that $|p^s(\tau)| \leqslant \tau$. Then, the worst case for the distance between server and target is, when both are on different sides of the origin. Thus, $y(\tau) \leqslant 2\tau$. □

Theorem 6.5. Algorithm NEF achieves a competitive ratio of

$$\frac{5}{2} + \frac{3a-1}{(a-1)^2} + \begin{cases} \dfrac{3a-1}{2(a-1)^2}, & \text{if } a \geqslant 2 \\ \dfrac{3a-1}{2(a-1)}, & \text{if } a < 2. \end{cases}$$

Proof. We assume that the last target is presented at time t. Let $p^{\mathrm{NEF}}(t)$ be the position of the online server at time t and S be the set of yet unvisited targets. Then, we consider the leftmost and rightmost target from $p^{\mathrm{NEF}}(t)$ in S. We assume, that both these targets are

moving away from the server, cases regarding targets, which move towards the server are addressed later. Without loss of generality we suppose the leftmost target is nearer to the server than the rightmost target and the rightmost one has a positive abscissa. Having this, the leftmost target, which is nearer to the server is called r_x and the rightmost target is called r_X, respectively. Let the positions of the leftmost and rightmost targets at time t be $p_x(t)$ and $p_X(t)$, with their distances from the online server as $x(t)$ and $X(t)$ respectively. Hence, $x(t) \leqslant X(t)$ and we have $p_X(t) > 0$. In addition, let $-L$ be the leftmost release position of the entire sequence of targets and R the rightmost release position, respectively. In case there is no target released on the left side of the origin, we set $L = 0$ and if there is no target released on the right side of the origin $R = 0$, respectively. Then, at time t the targets presented at $-L$ and R are at most $L + \frac{t}{a}$ and $R + \frac{t}{a}$ units away from the origin and all other targets are within this interval of $\left[-L - \frac{t}{a}, R + \frac{t}{a}\right]$ as well.

We denote the objective function value of the online server by Z^{NEF} and the optimal offline cost by Z^*. Obviously, the offline server cannot finish before the last request is presented, hence $Z^* \geqslant t$. Moreover, the following inequalities hold: $-L - \frac{t}{a} \leqslant p_x(t)$, $p_X(t) \leqslant R + \frac{t}{a}$, and $-L - \frac{t}{a} \leqslant p^{\mathrm{NEF}}(t) \leqslant R + \frac{t}{a}$. We estimate the competitive ratio Z^{NEF}/Z^* depending on the position $p^{\mathrm{NEF}}(t)$ of the NEF online server:

1. $p^{\mathrm{NEF}}(t) \leqslant p_x(t)$.
   In this case the leftmost not yet visited target is to the right of the NEF online server. Thus, to complete the task the server has to catch the rightmost target, which is located at $p_X(t)$, the other one is visited on the fly. The worst position the NEF server can be at time t in this case is $-L - \frac{t}{a}$, hence, the target to catch cannot exceed the position $p_X(t) + \dfrac{L + \frac{t}{a} + p_X(t)}{a-1}$ (see Lemma 6.2) at the time of interception. Thus, we have

   $$Z^{\mathrm{NEF}} \leqslant t + L + \frac{t}{a} + p_X(t) + \frac{L + \frac{t}{a} + p_X(t)}{a-1}.$$

   Applying $p_X(t) \leqslant R + \frac{t}{a}$ and the triangle inequality we have

   $$Z^{\mathrm{NEF}} \leqslant t + L + R + \frac{2t}{a} + \frac{L + R + \frac{2t}{a}}{a-1}.$$

   There are two lower bounds for the offline server $Z^* \geqslant t$ and $Z^* \geqslant L + R$. Having this, the competitive ratio can be obtained as

   $$\begin{aligned}\frac{Z^{\mathrm{NEF}}}{Z^*} &\leqslant \frac{t}{Z^*} + \frac{L+R}{Z^*} + \frac{2t}{aZ^*} + \frac{L+R}{(a-1)Z^*} + \frac{2t}{a(a-1)Z^*} \\ &\leqslant 1 + 1 + \frac{2}{a} + \frac{1}{a-1} + \frac{2}{a(a-1)} \\ &= 2 + \frac{3}{a-1}.\end{aligned}$$

   Note that with $a \to \infty$ we obtain the static case, where the targets do not move. Here, we have a competitive ratio of 2, when $a \to \infty$, which matches the result given by Ausiello et al. [Aus01] in this case.
2. $p_x(t) < p^{\mathrm{NEF}}(t)$
   We distinguish between two different cases: In the first one the optimal offline server visits r_X after r_x and in the second one it visits r_x after r_X.
   - $r_x \to r_X$

We assume that the offline server intercepts r_x in $p_x(d)$ at time d and this is the last visit at this position. Then, the adversary has to move from its current position $p_x(d)$ at least to the interception point with target r_X to finish. Consider the case, where $d \geqslant t$. We obtain a lower bound for the objective function value of the offline adversary for $d = t$. Since at time t the online server is located between $p_x(t)$ and $p_X(t)$ we can use the distances $x(t)$ and $X(t)$ to formulate the lower bound of the adversary:

$$Z^* \geqslant t + x(t) + X(t) + \frac{x(t) + X(t)}{a-1}. \tag{69}$$

At time t the online server is between $p_x(t)$ and $p_X(t)$ with the distances $x(t) \leqslant X(t)$. Thus, it has to catch r_x first with cost of $x(t) + \frac{x(t)}{a-1}$. Then, the server moves back, the return to the previous position costs again a time of $x(t) + \frac{x(t)}{a-1}$. The online server moves further to intercept r_X, which has moved in the meantime by $\frac{2x(t)}{a} + \frac{2x(t)}{a(a-1)}$. Then, the cost of intercepting r_X is:

$$X(t) + \frac{2x(t)}{a} + \frac{2x(t)}{a(a-1)} + \frac{X(t) + \frac{2x(t)}{a} + \frac{2x(t)}{a(a-1)}}{a-1}.$$

In total we have

$$\begin{aligned} Z^{\mathrm{NEF}} \leqslant\ & t + \underbrace{2x(t) + \frac{2x(t)}{a-1}}_{\text{catch } r_x \text{ and return}} + X(t) + \underbrace{\frac{2x(t)}{a} + \frac{2x(t)}{a(a-1)}}_{\substack{r_X \text{ moved, while NEF} \\ \text{caugth } r_x \text{ and returned}}} \\ & + \underbrace{\frac{X(t) + \frac{2x(t)}{a} + \frac{2x(t)}{a(a-1)}}{a-1}}_{\text{see Lemma 6.2}} \\ =\ & t + 2x(t) + X(t) + \frac{2x(t) + X(t)}{a-1} + \frac{2x(t)a}{(a-1)^2}. \end{aligned} \tag{70}$$

With the lower bound from (69) and $x(t) \leqslant X(t)$ we obtain for the competitive ratio

$$\frac{Z^{\mathrm{NEF}}}{Z^*} \leqslant \frac{t + 2x(t) + X(t) + \frac{2x(t) + X(t)}{a-1}}{t + x(t) + X(t) + \frac{x(t) + X(t)}{a-1}} + \frac{2x(t)\,a}{(a-1)^2\,Z^*}.$$

For Z^* we can use either the lower bound $Z^* \geqslant 2x(t)$ or $Z^* \geqslant \frac{2x(t)}{a-1}$, both results follow from (69) with $x(t) \leqslant X(t)$. For each bound we obtain a different result. These results are combined on the interval $a > 1$, to obtain the best competitive ration, which is

$$\frac{Z^{\mathrm{NEF}}}{Z^*} \leqslant 2 + \begin{cases} \dfrac{a}{(a-1)^2}, & \text{if } a \geqslant 2 \\ \dfrac{a}{a-1}, & \text{if } a < 2. \end{cases} \tag{71}$$

Now, we consider the case, where the last visit of position $p_x(d)$ is at $d < t$. It is not guaranteed, that r_X is visible at time d, but if it was visible its position would be at $p_X(d)$ in order to be at time t in $p_X(t)$. In this setting at time d there are three different positions for the online NEF server:
A) NEF is between $p_x(d)$ and $p_X(d)$ and has moved left at time t,
B) NEF is between $p_x(d)$ and $p_X(d)$ and has moved right at time t, and
C) NEF is to the right of $p_X(d)$ and has moved left at time t.
Note, that in case C) it is not possible for the online server to be at the right-hand side of $p^{\mathrm{NEF}}(d)$ at time t, because $X(t)$ would be less than $x(t)$, which is a contradiction to our assumption $x(t) \leqslant X(t)$. Furthermore, the NEF server cannot be left of $p_x(d)$ at time d, because it is to the right of $p_x(t)$ at time t, which means then, that r_x has already been visited at time t.
At time d the optimal offline adversary is located in $p_x(d)$ and has to catch r_X. Thus, it needs at least a time of

$$Z^* \geqslant d + x(d) + X(d) + \frac{x(d) + X(d)}{a-1} \tag{72}$$

to finish. Since the last target is presented at time t, the adversary cannot finish before t

$$Z^* \geqslant t. \tag{73}$$

Both these bounds hold for all coming cases.

A) $p_x(d) < p^{\mathrm{NEF}}(d) \leqslant p_X(d)$ and $p^{\mathrm{NEF}}(t) \leqslant p^{\mathrm{NEF}}(d)$
The online server is located between r_x and r_X at time d and has moved to the left at time t. This distance is called Δ^{le} and it is defined as:

$$\Delta^{le} := p^{\mathrm{NEF}}(d) - p^{\mathrm{NEF}}(t).$$

Obviously, $\Delta^{le} \geqslant 0$. Relations between the distances of targets and online server at times d and t can be formulated as follows:

$$x(t) = x(d) + \frac{t-d}{a} - \Delta^{le}, \qquad X(t) = X(d) + \frac{t-d}{a} + \Delta^{le}.$$

These relations transform the upper bound of Z^{NEF} provided in (70) to:

$$\begin{aligned} Z^{\mathrm{NEF}} \leqslant\ & t + 2x(d) + X(d) - \Delta^{le} + \frac{2x(d) + X(d) - \Delta^{le}}{a-1} \\ & + \frac{a\left(2x(d) - 2\Delta^{le}\right)}{(a-1)^2} + \frac{(t-d)\,(3a-1)}{(a-1)^2}. \end{aligned} \tag{74}$$

NEF is between r_x and r_X, but it can either be closer to r_x, then we have $x(d) \leqslant X(d)$ or closer to r_X, then we have $x(d) > X(d)$. Obviously, the cases have an effect on the lower bound of Z^*. At first, we consider $x(d) \leqslant X(d)$. Here, the lower bound in (72) can be transformed with $d \geqslant \frac{x(d)}{2}$ (see Lemma 6.4) into the following bounds

$$\begin{aligned} Z^* &\geqslant \frac{3}{2}x(d) + X(d) + \frac{x(d) + X(d)}{a-1}, \\ Z^* &\geqslant \frac{5}{2}x(d), \end{aligned} \tag{75}$$

$$Z^* \geqslant \frac{2x(d)}{a-1}.$$

Since $-\Delta^{le} \leqslant 0$, the upper bound for Z^{NEF} can be further evaluated to

$$\begin{aligned} Z^{\mathrm{NEF}} &\leqslant t + 2x(d) + X(d) + \frac{2x(d) + X(d)}{a-1} + \frac{a\,(2x(d))}{(a-1)^2} \\ &\quad + \frac{(t-d)\,(3a-1)}{(a-1)^2}. \end{aligned} \tag{76}$$

Then, the competitive ratio can be computed as

$$\begin{aligned} \frac{Z^{\mathrm{NEF}}}{Z^*} &\leqslant \frac{t}{t} + \frac{2x(d) + X(d) + \dfrac{2x(d) + X(d)}{a-1}}{\frac{3}{2}x(d) + X(d) + \dfrac{x(d) + X(d)}{a-1}} + \frac{a(2x(d))}{(a-1)^2\,Z^*} \\ &\quad + \frac{(t-d)(3a-1)}{t\,(a-1)^2} \\ &\leqslant 1 + 1 + \frac{\frac{1}{2}x(d)}{\frac{5}{2}x(d)} + \frac{x(d)}{(a-1)\,Z^*} + \frac{a\,2x(d)}{(a-1)^2\,Z^*} + \frac{3a-1}{(a-1)^2}. \end{aligned}$$

Here, we apply the different bounds from (75), depending on the bounds we obtain different results, which are best combined with regard to a:

$$\begin{aligned} \frac{Z^{\mathrm{NEF}}}{Z^*} &\leqslant 2 + \frac{1}{5} + \begin{cases} \dfrac{2}{5(a-1)}, & \text{if } a \geqslant \frac{9}{5} \\ \dfrac{1}{2}, & \text{if } a < \frac{9}{5} \end{cases} + \begin{cases} \dfrac{4a}{5(a-1)^2}, & \text{if } a \geqslant \frac{9}{5} \\ \dfrac{a}{a-1}, & \text{if } a < \frac{9}{5} \end{cases} \\ &\quad + \frac{3a-1}{(a-1)^2} \\ &= \frac{11}{5} + \frac{3a-1}{(a-1)^2} + \begin{cases} \dfrac{2(3a-1)}{5(a-1)^2}, & \text{if } a \geqslant \frac{9}{5} \\ \dfrac{3a-1}{2(a-1)}, & \text{if } a < \frac{9}{5}. \end{cases} \end{aligned} \tag{77}$$

In the case $x(d) > X(d)$, the online server is closer to r_X and has to move to the left at time t. The movement to the left goes at least to the center between r_x and r_X:

$$\Delta^{le} \geqslant \frac{x(d)}{2} - \frac{X(d)}{2}. \tag{78}$$

We apply this to the upper bound of Z^{NEF} in (74), thus, we have

$$\begin{aligned} Z^{\mathrm{NEF}} &\leqslant t + \frac{3}{2}x(d) + \frac{3}{2}X(d) + \frac{\frac{3}{2}x(d) + \frac{3}{2}X(d)}{a-1} + \frac{a\,(x(d) + X(d))}{(a-1)^2} \\ &\quad + \frac{(t-d)\,(3a-1)}{(a-1)^2}. \end{aligned}$$

In this case, we obtain slightly different lower bounds for Z^*:

$$Z^* \geqslant \frac{3}{2}x(d) + X(d) + \frac{x(d)+X(d)}{a{-}1},$$
$$Z^* \geqslant \frac{5}{2}X(d),$$
$$Z^* \geqslant \frac{x(d)+X(d)}{a{-}1}.$$

Having this, the competitive ratio is calculated as

$$\begin{aligned}
\frac{Z^{\mathrm{NEF}}}{Z^*} &\leqslant \frac{t}{t} + \frac{\frac{3}{2}x(d) + \frac{3}{2}X(d) + \frac{\frac{3}{2}x(d)+\frac{3}{2}X(d)}{a{-}1}}{\frac{3}{2}x(d) + X(d) + \frac{x(d)+X(d)}{a{-}1}} + \frac{a\,(x(d)+X(d))}{(a{-}1)^2 Z^*} \\
&\quad + \frac{(t{-}d)\,(3a{-}1)}{t\,(a{-}1)^2} \\
&\leqslant 1 + 1 + \frac{\frac{1}{2}X(d)}{\frac{5}{2}X(d)} + \frac{\frac{1}{2}x(d)+\frac{1}{2}X(d)}{(a{-}1)Z^*} + \frac{a\,(x(d)+X(d))}{(a{-}1)^2 Z^*} \\
&\quad + \frac{3a{-}1}{(a{-}1)^2} \\
&\leqslant 2 + \frac{1}{5} + \begin{cases} \dfrac{1}{2(a{-}1)}, & \text{if } a \geqslant 2 \\ \dfrac{1}{2}, & \text{if } a < 2 \end{cases} + \begin{cases} \dfrac{a}{(a{-}1)^2}, & \text{if } a \geqslant 2 \\ \dfrac{a}{a{-}1}, & \text{if } a < 2 \end{cases} \\
&\quad + \frac{3a{-}1}{(a{-}1)^2} \\
&= \frac{11}{5} + \frac{3a{-}1}{(a{-}1)^2} + \begin{cases} \dfrac{3a{-}1}{2(a{-}1)^2}, & \text{if } a \geqslant 2 \\ \dfrac{3a{-}1}{2(a{-}1)}, & \text{if } a < 2. \end{cases}
\end{aligned} \tag{79}$$

Concluding this case, we have that the result in (77) cannot be greater than the result in (79).

B) $p_x(d) < p^{\mathrm{NEF}}(d) \leqslant p_X(d)$ and $p^{\mathrm{NEF}}(t) > p^{\mathrm{NEF}}(d)$
This case is similar to the one before, however, the position of the online server at time t is to the right of $p^{\mathrm{NEF}}(d)$. Hence, the distance Δ^{ri} is defined as:

$$\Delta^{ri} \coloneqq p^{\mathrm{NEF}}(d) - p^{\mathrm{NEF}}(t), \quad \Delta^{ri} \geqslant 0.$$

Obviously, at time d the server is closer to r_x than to r_X, which means $x(d) \leqslant X(d)$. Thus, the distance Δ^{ri} is bounded by the center between r_x and r_X:

$$\Delta^{ri} < \frac{X(d)}{2} - \frac{x(d)}{2}. \tag{80}$$

Finally, we need the relations between the target-server distances at time d

and t

$$x(t) = x(d) + \frac{t-d}{a} + \Delta^{ri}, \qquad X(t) = X(d) + \frac{t-d}{a} - \Delta^{ri}, \tag{81}$$

which are used with (70) to obtain the following upper bound for NEF:

$$\begin{aligned} Z^{\mathrm{NEF}} &\leqslant t + 2x(d) + X(d) + \Delta^{ri} + \frac{2x(d) + X(d) + \Delta^{ri}}{a-1} \\ &\quad + \frac{a\left(2x(d) + 2\Delta^{ri}\right)}{(a-1)^2} + \frac{(t-d)(3a-1)}{(a-1)^2} \\ &\quad \text{apply (80)} \\ &\leqslant t + \frac{3}{2}x(d) + \frac{3}{2}X(d) + \frac{\frac{3}{2}x(d) + \frac{3}{2}X(d)}{a-1} + \frac{a\left(x(d) + X(d)\right)}{(a-1)^2} \\ &\quad + \frac{(t-d)\,(3a-1)}{(a-1)^2} \end{aligned}$$

The offline adversary located in $p_x(d)$ has to intercept r_X. Next to the bounds (72) and (73) the following equations hold with $d \geqslant \frac{x(d)}{2}$ (Lemma 6.4):

$$\begin{aligned} &Z^* \geqslant \frac{3}{2}x(d) + X(d) + \frac{x(d) + X(d)}{a-1}, \\ &Z^* \geqslant x(d) + X(d) \quad \text{and} \quad Z^* \geqslant \frac{x(d) + X(d)}{a-1}. \end{aligned}$$

Having this, the competitive ratio is computed as

$$\begin{aligned} \frac{Z^{\mathrm{NEF}}}{Z^*} &\leqslant \frac{t}{t} + \frac{\frac{3}{2}x(d) + \frac{3}{2}X(d) + \frac{\frac{3}{2}x(d) + \frac{3}{2}X(d)}{a-1}}{\frac{3}{2}x(d) + X(d) + \frac{x(d) + X(d)}{a-1}} + \frac{a\left(x(d) + X(d)\right)}{(a-1)^2\, Z^*} \\ &\quad + \frac{(t-d)\,(3a-1)}{t\,(a-1)^2} \\ &\leqslant 2 + \frac{\frac{1}{2}X(d)}{X(d)} + \frac{\frac{1}{2}x(d) + \frac{1}{2}X(d)}{(a-1)\,Z^*} + \frac{a\left(x(d) + X(d)\right)}{(a-1)^2\,Z^*} + \frac{3a-1}{(a-1)^2} \\ &\leqslant \frac{5}{2} + \begin{cases} \dfrac{1}{2(a-1)}, & \text{if } a \geqslant 2 \\ \dfrac{1}{2}, & \text{if } a < 2. \end{cases} + \begin{cases} \dfrac{a}{(a-1)^2}, & \text{if } a \geqslant 2 \\ \dfrac{a}{a-1}, & \text{if } a < 2. \end{cases} + \frac{3a-1}{(a-1)^2} \\ &= \frac{5}{2} + \frac{3a-1}{(a-1)^2} + \begin{cases} \dfrac{3a-1}{2(a-1)^2}, & \text{if } a \geqslant 2 \\ \dfrac{3a-1}{2(a-1)}, & \text{if } a < 2. \end{cases} \end{aligned} \tag{82}$$

C) $p_X(d) < p^{\mathrm{NEF}}(d)$
This can only be the case, when r_X is not visible at time d and it must be that $p^{\mathrm{NEF}}(t) \leqslant p^{\mathrm{NEF}}(d)$, otherwise the assumption $x(t) \leqslant X(t)$ would be

violated. Hence we obtain the following distance definition and relations:

$$\Delta^{le} := p^{\mathrm{NEF}}(d) - p^{\mathrm{NEF}}(t), \quad \Delta^{le} \geqslant 0,$$

$$x(t) = x(d) + \frac{t-d}{a} - \Delta^{le}, \tag{83}$$

$$X(t) = \frac{t-d}{a} - X(d) + \Delta^{le}. \tag{84}$$

The online server has to move left within the time from d to t, at least to the center between $p_x(d)$ and $p_X(d)$:

$$\frac{x(d) + X(d)}{2} \leqslant \Delta^{le}. \tag{85}$$

We apply (83) and (84) to the upper bound of NEF (70) and obtain

$$\begin{aligned} Z^{\mathrm{NEF}} &\leqslant t + 2x(d) - X(d) - \Delta^{le} + \frac{3(t-d)}{a} \\ &\quad + \frac{2x(d) - X(d) - \Delta^{le} + \frac{3(t-d)}{a}}{a-1} + \frac{a\left(2x(d) - 2\Delta^{le} + \frac{2(t-d)}{a}\right)}{(a-1)^2} \\ &\quad \text{apply (85)} \\ &\leqslant t + \frac{3}{2}x(d) - \frac{3}{2}X(d) + \frac{\frac{3}{2}x(d) - \frac{3}{2}X(d)}{a-1} + \frac{a\,(x(d) - X(d))}{(a-1)^2} \\ &\quad + \frac{(t-d)\,(3a-1)}{(a-1)^2}. \end{aligned}$$

With $d \geqslant \frac{x(d)}{2}$ (see Lemma 6.4), the lower bounds of Z^* (72) and (73) are extended by

$$Z^* \geqslant \frac{3}{2}x(d) - X(d) + \frac{x(d) - X(d)}{a-1} \geqslant x(d) - X(d),$$

$$Z^* \geqslant \frac{x(d) - X(d)}{a-1}.$$

Then, the competitive ratio results in

$$\begin{aligned} \frac{Z^{\mathrm{NEF}}}{Z^*} &\leqslant \frac{t}{t} + \frac{\frac{3}{2}x(d) - \frac{3}{2}X(d) + \frac{\frac{3}{2}x(d) - \frac{3}{2}X(d)}{a-1}}{\frac{3}{2}x(d) - X(d) + \frac{x(d) - X(d)}{a-1}} + \frac{a\,(x(d) - X(d))}{Z^*(a-1)^2} \\ &\quad + \frac{(t-d)\,(3a-1)}{t\,(a-1)^2} \\ &\leqslant 2 + \frac{\frac{1}{2}x(d) - \frac{1}{2}X(d)}{(a-1)Z^*} + \frac{a\,(x(d) - X(d))}{(a-1)^2 Z^*} + \frac{3a-1}{(a-1)^2} \\ &\leqslant 2 + \begin{cases} \frac{1}{3(a-1)}, & \text{if } a \geqslant \frac{5}{3} \\ \frac{1}{2}, & \text{if } a < \frac{5}{3} \end{cases} + \begin{cases} \frac{2a}{3(a-1)^2}, & \text{if } a \geqslant \frac{5}{3} \\ \frac{a}{a-1}, & \text{if } a < \frac{5}{3} \end{cases} + \frac{3a-1}{(a-1)^2} \end{aligned}$$

$$= 2 + \frac{3a-1}{(a-1)^2} + \begin{cases} \frac{3a-1}{3(a-1)^2}, & \text{if } a \geqslant \frac{5}{3} \\ \frac{3a-1}{2(a-1)}, & \text{if } a < \frac{5}{3} \end{cases}. \tag{86}$$

- $r_X \to r_x$
  In this case the optimal offline server intercepts r_X first and then r_x. We assume, that the offline server visits r_X at time d in position $p_X(d)$ and this is the last visit at this position. Being at $p_X(d)$, the adversary has at least to intercept r_x to finish. At first the case $d \geqslant t$ is considered. However, in this case the lower bound for Z^* and the upper bound for Z^{NEF} is exactly the same as in the case $x \to X$ and the same competitive ratio as in (71) is achieved. Hence, the case $d < t$ is left for examination. Here, it is not guaranteed, that r_x is visible, due to this, we use again the virtual point $p_x(d)$, which is the location where r_x would be, if it was visible at time d and reaches $p_x(t)$ at time t. Then, there are four different cases for the position of the online server at time d:
  a) NEF is to the left of $p_x(d)$ and has moved left at time t,
  b) NEF is to the left of $p_x(d)$ and has moved right at time t,
  c) NEF is between $p_x(d)$ and $p_X(d)$ and has moved right at time t, and
  d) NEF is between $p_x(d)$ and $p_X(d)$ and has moved left at time t.

  a) $p^{\text{NEF}}(d) \leqslant p_x(d)$ and $p^{\text{NEF}}(t) \leqslant p^{\text{NEF}}(d)$
  Here, target r_x cannot be visible at time d, otherwise it would already be caught at time t. Thus, a possible scenario is, that NEF moves to the right (r_X) and between time d and t a target r_z is presented to the left of NEF. This target is nearer to NEF than r_X and forces the online server to move left. If $r_z = r_x$, NEF must not intercept it until time t. In the case $r_z \neq r_x$, NEF has caught r_z at time t or r_z is closer to NEF than r_x and can be caught on the way to r_x. The left movement of the NEF server is bounded by r_x to the left side. As an abbreviation we set

  $$\Delta^{le} := p^{\text{NEF}}(d) - p^{\text{NEF}}(t).$$

  The following constraints describe the distances at time d and t:

  $$x(t) = \frac{t-d}{a} - x(d) - \Delta^{le} \tag{87}$$

  $$X(t) = X(d) + \frac{t-d}{a} + \Delta^{le}. \tag{88}$$

  Obviously, we have

  $$\Delta^{le} < \frac{t-d}{a} - x(d), \tag{89}$$

  otherwise NEF would be at or to the left of r_x at time t, which is a contradiction to our assumption, that r_x is the leftmost target at time t. With the constraints (87) and (88) the upper bound for NEF (70) can be transformed to

  $$Z^{\text{NEF}} \leqslant t + X(d) - 2x(d) - \Delta^{le} + \frac{X(d) - 2x(d) - \Delta^{le}}{a-1} + \frac{a\left(-2x(d) - 2\Delta^{le}\right)}{(a-1)^2} + \frac{(t-d)\,(3a-1)}{(a-1)^2}.$$

Since $\Delta^{le} \geqslant 0$, $x(d) \geqslant 0$ and $a > 1$ we can increase the bound to

$$Z^{\mathrm{NEF}} \leqslant t + X(d) - x(d) + \frac{X(d) - x(d)}{a-1} + \frac{(t-d)\,(3a-1)}{(a-1)^2}.$$

As lower bounds for the offline adversary we have in this case:

$$\begin{aligned} Z^* &\geqslant d + X(d) - x(d) + \frac{X(d) - x(d)}{a-1} \\ &\geqslant X(d) - x(d) + \frac{X(d) - x(d)}{a-1} \quad \text{and} \\ Z^* &\geqslant t. \end{aligned}$$

Hence, the competitive ratio results in

$$\begin{aligned} \frac{Z^{\mathrm{NEF}}}{Z^*} &\leqslant \frac{t}{t} + \frac{X(d) - x(d) + \dfrac{X(d) - x(d)}{a-1}}{X(d) - x(d) + \dfrac{X(d) - x(d)}{a-1}} + \frac{(t-d)\,(3a-1)}{t\,(a-1)^2} \\ &= 2 + \frac{3a-1}{(a-1)^2}. \end{aligned} \tag{90}$$

b) $p^{\mathrm{NEF}}(d) \leqslant p_x(d)$ and $p^{\mathrm{NEF}}(t) > p^{\mathrm{NEF}}(d)$
This case is the same as the one before, except that the position of NEF at time t is to the right of its position at time d. At first, we set

$$\Delta^{ri} := p^{\mathrm{NEF}}(t) - p^{\mathrm{NEF}}(d)$$

and then we describe the necessary bounds:

$$x(d) \leqslant X(d),$$

$$x(t) = \frac{t-d}{a} - x(d) + \Delta^{ri}, \tag{91}$$

$$X(t) = X(d) + \frac{t-d}{a} - \Delta^{ri}, \tag{92}$$

$$\Delta^{ri} \leqslant \frac{X(d) + x(d)}{2} \tag{93}$$

$$d \geqslant \frac{X(d) - x(d)}{2}. \tag{94}$$

Constraint (93) results from the fact, that the furthest to the right NEF can move is to the center between $p_x(d)$ and $p_X(d)$ (which is equal to the center between $p_x(t)$ and $p_X(t)$), otherwise the assumption $x(t) \leqslant X(t)$ is hurt. The last constraint (94) follows from $\frac{X(d)}{2} \leqslant d$ stated in Lemma 6.4. Applying (91) and (92) to (70), we can write the upper bound for the NEF server as

$$\begin{aligned} Z^{\mathrm{NEF}} &\leqslant t + X(d) - 2x(d) + \Delta^{ri} + \frac{x(d) - 2x(d) + \Delta^{ri}}{a-1} \\ &\quad + \frac{2\Delta^{ri} - 2x(d)}{(a-1)^2} + \frac{(t-d)(3a-1)}{(a-1)^2} \\ &\quad \text{apply (93)} \end{aligned}$$

$$\leqslant t + \frac{3}{2}X(d) - \frac{3}{2}x(d) + \frac{\frac{3}{2}X(d) - \frac{3}{2}x(d)}{a{-}1} + \frac{a\,(X(d) - x(d))}{(a{-}1)^2} + \frac{(t{-}d)\,(3a{-}1)}{(a{-}1)^2}.$$

The lower bounds of the optimal offline server

$$Z^* \geqslant d + X(d) - x(d) + \frac{X(d) - x(d)}{a{-}1}, \quad Z^* \geqslant t,$$

can be transformed with (94) to

$$Z^* \geqslant \frac{3}{2}X(d) - \frac{3}{2}x(d) + \frac{X(d) - x(d)}{a{-}1} \geqslant \frac{3}{2}X(d) - \frac{3}{2}x(d),$$
$$Z^* \geqslant \frac{X(d) - x(d)}{a{-}1}.$$

Thus, we can compute the competitive ratio as

$$\begin{aligned}\frac{Z^{\mathrm{NEF}}}{Z^*} &\leqslant \frac{t}{t} + \frac{\frac{3}{2}X(d) - \frac{3}{2}x(d) + \frac{\frac{3}{2}X(d) - \frac{3}{2}x(d)}{a{-}1}}{\frac{3}{2}X(d) - \frac{3}{2}x(d) + \frac{X(d) - x(d)}{a{-}1}} + \frac{a\,(X(d) - x(d))}{(a{-}1)^2\,Z^*} \\ &\quad + \frac{(t{-}d)(3a{-}1)}{t\,(a{-}1)^2} \\ &\leqslant 2 + \frac{\frac{1}{2}X(d) - \frac{1}{2}x(d)}{(a{-}1)Z^*} + \begin{cases} \dfrac{2a}{3(a{-}1)^2}, & \text{if } a \geqslant \dfrac{5}{3} \\ \dfrac{a}{(a{-}1)}, & \text{if } a < \dfrac{5}{3} \end{cases} + \frac{3a{-}1}{(a{-}1)^2} \\ &\leqslant 2 + \frac{3a{-}1}{(a{-}1)^2} + \begin{cases} \dfrac{3a{-}1}{3(a{-}1)^2}, & \text{if } a \geqslant \dfrac{5}{3} \\ \dfrac{3a{-}1}{2(a{-}1)}, & \text{if } a < \dfrac{5}{3} \end{cases} \end{aligned} \tag{95}$$

This competitive ratio is equal to the one in case C), because it is the symmetrical case.

c) $p_x(d) < p^{\mathrm{NEF}}(d) < p_X(d)$ and $p^{\mathrm{NEF}}(t) > p^{\mathrm{NEF}}(d)$
Considering position and movement of the NEF server, this case is identical to the case B), where the optimal offline server is catching r_x first. Thus, we use the upper bound for NEF as computed in case B), which is

$$Z^{\mathrm{NEF}} \leqslant t + \frac{3}{2}x(d) + \frac{3}{2}X(d) + \frac{\frac{3}{2}x(d) + \frac{3}{2}X(d)}{a{-}1} + \frac{a\,(x(d) + X(d))}{(a{-}1)^2} + \frac{(t{-}d)\,(3a{-}1)}{(a{-}1)^2}.$$

The standard lower bound (72) for the movement of the offline server to catch r_x can be extended by $d \geqslant \frac{X(d)}{2}$ (see Lemma 6.4) and $x(d) \leqslant X(d)$ to

$$Z^* \geqslant x(d) + \frac{3}{2}X(d) + \frac{x(d) + X(d)}{a{-}1} \geqslant \frac{5}{2}x(d),$$

$$Z^* \geqslant x(d) + X(d), \text{ and}$$
$$Z^* \geqslant \frac{x(d) + X(d)}{a-1}.$$

Thus, the competitive ratio can be calculated as:

$$\begin{aligned}
\frac{Z^{\mathrm{NEF}}}{Z^*} &\leqslant \frac{t}{t} + \frac{\frac{3}{2}x(d) + \frac{3}{2}X(d) + \frac{\frac{3}{2}x(d) + \frac{3}{2}X(d)}{a-1}}{x + \frac{3}{2}X(d) + \frac{x(d)+X(d)}{a-1}} + \frac{a\,(x(d)+X(d))}{(a-1)^2 Z^*} \\
&\quad + \frac{(t-d)(3a-1)}{t(a-1)^2} \\
&\leqslant 2 + \frac{\frac{1}{2}x(d)}{\frac{5}{2}x(d)} + \frac{\frac{1}{2}x(d) + \frac{1}{2}X(d)}{(a-1)Z^*} + \frac{a\,(x(d)+X(d))}{(a-1)^2 Z^*} + \frac{3a-1}{(a-1)^2} \\
&\leqslant \frac{11}{5} + \frac{3a-1}{(a-1)^2} + \begin{cases} \dfrac{3a-1}{2(a-1)^2}, & \text{if } a \geqslant 2 \\ \dfrac{3a-1}{2(a-1)}, & \text{if } a < 2. \end{cases}
\end{aligned} \tag{96}$$

d) $p_x(d) < p^{\mathrm{NEF}}(d) < p_X(d)$ and $p^{\mathrm{NEF}}(t) \leqslant p^{\mathrm{NEF}}(d)$
This last case corresponds to case A). In the same manner as in case A) we have to differentiate between $x(d) \leqslant X(d)$ and $x(d) > X(d)$. In the first case, the upper bound for NEF (76) can be used here. Since $d \geqslant \frac{X(d)}{2} \geqslant \frac{x(d)}{2}$ the lower bounds for Z^* (75) can be used here as well. Thus, we obtain the same competitive ratio as in (77), which is recalled here

$$\frac{Z^{\mathrm{NEF}}}{Z^*} \leqslant \frac{11}{5} + \frac{3a-1}{(a-1)^2} + \begin{cases} \dfrac{2(3a-1)}{5(a-1)^2}, & \text{if } a \geqslant \frac{9}{5} \\ \dfrac{3a-1}{2(a-1)}, & \text{if } a < \frac{9}{5}. \end{cases} \tag{97}$$

In the second case $x(d) > X(d)$, a detailed look at the possible distances of NEF is needed. We provide the following familiar abbreviation

$$\Delta^{le} := p^{\mathrm{NEF}}(d) - p^{\mathrm{NEF}}(t)$$

as well as related bounds for the distances at time d and t

$$x(t) = x(d) + \frac{t-d}{a} - \Delta^{le}, \tag{98}$$
$$X(t) = X(d) + \frac{t-d}{a} + \Delta^{le}, \tag{99}$$
$$\Delta^{le} \geqslant \frac{x(d) - X(d)}{2}, \tag{100}$$
$$d \geqslant \frac{X(d)}{2}, \quad \text{(see Lemma 6.4).} \tag{101}$$

Constraint (100) results from the fact, that the movement to the left has to go at least to the center between r_x and r_X at time t, which is obviously

the same at time d. Then, we have as upper bound for NEF:

$$\begin{aligned}
Z^{\mathrm{NEF}} &\leqslant t + 2x(t) + X(t) + \frac{2x(t) + X(t)}{a{-}1} + \frac{2x(t)a}{(a{-}1)^2} \\
&\qquad \text{apply (98) and (99)} \\
&\leqslant t + 2x(d) + X(d) - \Delta^{le} + \frac{2x(d) + X(d) - \Delta^{le}}{a{-}1} \\
&\quad + \frac{a\left(2x(d) - 2\Delta^{le}\right)}{(a{-}1)^2} + \frac{(t{-}d)(3a{-}1)}{(a{-}1)^2} \\
&\qquad \text{apply (100)} \\
&\leqslant t + \frac{3}{2}x(d) + \frac{3}{2}X(d) + \frac{\frac{3}{2}x(d) + \frac{3}{2}X(d)}{a{-}1} + \frac{a\left(x(d) + X(d)\right)}{(a{-}1)^2} \\
&\quad + \frac{(t{-}d)(3a{-}1)}{(a{-}1)^2}.
\end{aligned}$$

With (101) and $x(d) > X(d)$ the bounds for the optimal offline server are given by (72), (73) and:

$$\begin{aligned}
Z^* &\geqslant x(d) + \frac{3}{2}X(d) + \frac{x(d) + X(d)}{a{-}1} \\
Z^* &\geqslant \frac{x(d) + X(d)}{a{-}1} \\
Z^* &\geqslant x(d) + X(d) \geqslant x(d).
\end{aligned}$$

The competitive ratio can be computed as

$$\begin{aligned}
\frac{Z^{\mathrm{NEF}}}{Z^*} &\leqslant \frac{t}{t} + \frac{\frac{3}{2}x(d){+}\frac{3}{2}X(d) + \frac{\frac{3}{2}x(d){+}\frac{3}{2}X(d)}{a{-}1}}{x(d){+}\frac{3}{2}X(d) + \frac{x(d){+}X(d)}{a{-}1}} + \frac{a\left(x(d){+}X(d)\right)}{(a{-}1)^2 Z^*} \\
&\quad + \frac{(t{-}d)(3a{-}1)}{t\,(a{-}1)^2} \\
&\leqslant 2 + \frac{\frac{1}{2}x(d)}{x(d)} + \frac{\frac{1}{2}x(d){+}\frac{1}{2}X(d)}{(a{-}1)Z^*} + \begin{cases} \dfrac{a}{(a{-}1)^2}, & \text{if } a \geqslant 2 \\ \dfrac{a}{a{-}1}, & \text{if } a < 2 \end{cases} \\
&\quad + \frac{3a{-}1}{(a{-}1)^2} \\
&\leqslant \frac{5}{2} + \frac{3a{-}1}{(a{-}1)^2} + \begin{cases} \dfrac{3a{-}1}{2(a{-}1)^2}, & \text{if } a \geqslant 2 \\ \dfrac{3a{-}1}{2(a{-}1)}, & \text{if } a < 2. \end{cases}
\end{aligned} \tag{102}$$

The competitive ratio in case B), which is given in (82) is equal to the competitive ratio of the second case in d) (102) and this ratio is the biggest value of all computed ratios in this proof.

So far only targets moving away from the online server have been considered. Now, we are addressing the case of targets moving towards the online server. At first, the distance

between target and server is shrinking while the server is waiting. Thus, the time to catch a coming target cannot be greater than the time to catch a target that is moving away from the server provided both targets have the same distance to the server. If the algorithm NEF does not distinguish between the movement directions of the targets and determines the leftmost and rightmost extreme targets of all target no matter of their moving direction, the competitive ratio of NEF cannot be more than what has been computed. Thus, for the worst case consideration only those targets are essential, that move away from the server. Assume the two cases: NEF how it is defined and NEF not distinguishing between moving direction of the targets. Then, it is obvious that in the first case the makespan is always smaller or equal to the latter case. However, the worst case consideration as is done with the competitive ratio is equal for both cases. Assume NEF at the origin and on one side a target is moving away from it and on the other side a target that is moving towards it. Then, smallest makespan is reached by first caching the target that is moving away and then the other. □

Note, that for $a \to \infty$ we have the static case (OLTSP), where the targets do not move over time. NEF can also be applied to the OLTSP on the real line, in this case the competitive ratio is

$$\lim_{a\to\infty} \left(\frac{5}{2} + \frac{3a-1}{(a-1)^2} + \begin{cases} \frac{3a-1}{2(a-1)^2}, & \text{if } a \geqslant 2 \\ \frac{3a-1}{2(a-1)}, & \text{if } a < 2. \end{cases} \right) = \frac{5}{2}.$$

In fact, for the OLTSP on the line the competitive ratio of NEF is slightly worse than the result of Ausiello et al. [Aus01] in Theorem 6.1, which is $\frac{7}{3}$. Moreover, the algorithm used in [Aus01], Theorem 6.1, can also be applied to the OLMTTSP on the real line. We provide the competitive ratio of ENO for the open OLMTTSP in Section 6.3.

Theorem 6.6. The ratio in Theorem 6.5 is tight.

Proof. We consider the static case, where the targets do not move. Here, we construct the following sequence of targets as visualized in Figure 20.

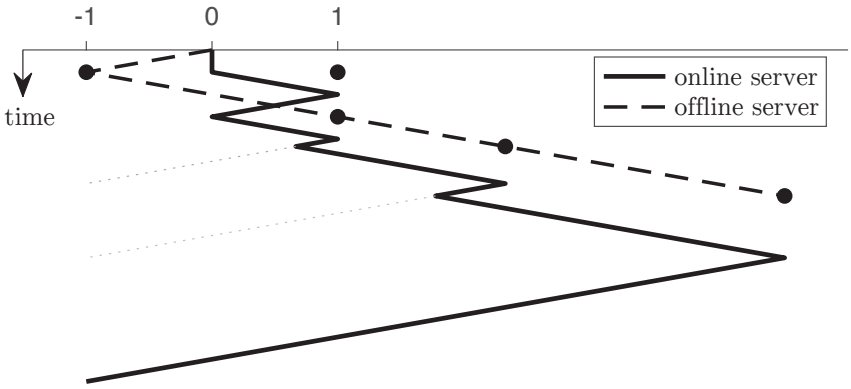

Figure 20: Worst case example for the open OLTSP on the real line.

At time $t = 1$ two targets in -1 and 1 are presented. NEF leaves the origin without loss of generality towards 1 and arrives in 1 at time $t = 2$. Then NEF turns round towards the left target. The offline server starts at time $t = 0$ and intercepts the left target at time $t = 1$, afterwards it moves to the right side to catch the other one at time $t = 3$. At $t = 3$, where NEF is at the origin (in the middle between left target and offline server), the adversary presents a new target at position 1, which is caught by the offline server at the

moment of release $t = 3$. Then, the adversary continues to move to the right side, while NEF moves right to catch the new target, which is equally distant as the left one and then turns round. Again, at the moment, where NEF is in the middle between the left target and the adversary, a new target at the position of the adversary is presented and so on.

Let us look at the objective function value for NEF and the offline adversary in each iteration (after a new target is presented). The number of iteration is called $n \in \mathbb{N}$. We start with $n = 0$ when 3 targets have been presented. Then, we have the next iteration when a new target is presented. For each iteration we compute the objective function value as if no more target would be revealed, see Table 4, where 'nbt' denotes the number of presented targets.

Table 4: Objective function values for iterations $n = 0, \ldots, 4$.

n	nbt	NEF		offline adversary	
0	3	6	$= 4 + 2$	3	$= 3$
1	4	$\frac{28}{3}$	$= 4 + 2 + 2\,\frac{5}{3}$	$\frac{13}{3}$	$= 3 + \frac{4}{3}$
2	5	$\frac{134}{9}$	$= 4 + 2 + 2\,\frac{5}{3} + 2\left(\frac{5}{3}\right)^2$	$\frac{59}{9}$	$= 3 + \frac{4}{3} + \frac{4}{3}\,\frac{5}{3}$
3	6	$\frac{652}{27}$	$= 4 + 2 + 2\,\frac{5}{3} + 2\left(\frac{5}{3}\right)^2 + 2\left(\frac{5}{3}\right)^3$	$\frac{277}{27}$	$= 3 + \frac{4}{3} + \frac{4}{3}\,\frac{5}{3} + \frac{4}{3}\left(\frac{5}{3}\right)^2$
4	7	$\frac{3206}{81}$	$= 4 + 2\sum_{i=0}^{4}\left(\frac{5}{3}\right)^i$	$\frac{1331}{81}$	$= 3 + \frac{4}{3}\sum_{i=1}^{4}\left(\frac{5}{3}\right)^{i-1}$

Thus, for iteration n we have

$$Z^{\mathrm{NEF}} = 4 + 2\sum_{i=0}^{n}\left(\frac{5}{3}\right)^i$$

and

$$Z^* = 3 + \frac{4}{3}\sum_{i=1}^{n}\left(\frac{5}{3}\right)^{i-1} = \frac{11}{5} + \frac{4}{5}\sum_{i=0}^{n}\left(\frac{5}{3}\right)^i.$$

Then, the competitive ratio can be computed as

$$\frac{Z^{\mathrm{NEF}}}{Z^*} = \frac{4 + 2\sum_{i=0}^{n}\left(\frac{5}{3}\right)^i}{\frac{11}{5} + \frac{4}{5}\sum_{i=0}^{n}\left(\frac{5}{3}\right)^i} = \frac{2\left(2 + \sum_{i=0}^{n}\left(\frac{5}{3}\right)^i\right)}{\frac{4}{5}\left(\frac{11}{4} + \sum_{i=0}^{n}\left(\frac{5}{3}\right)^i\right)} \xrightarrow[n\to\infty]{} \frac{5}{2}.$$

□

The ENO Algorithm for the Open OLMTTSP on the Real Line

The algorithm ENO (short for serve Extreme Nearest to the Origin first) has been defined in Ausiello et al. [Aus01] for the open OLTSP on the line. We adapt ENO to the case of targets moving over time and provide the competitive ratio with regard to the speed ratio a. The essential difference of ENO to NEF is that it serves the target with the least distance *to the origin* first.

ENO: At the beginning, the online server is located at the origin 0. Let S^A be the set of presented targets not yet visited, that are moving away from the server and S^T be the set of presented targets not yet visited, that are moving towards the server. The leftmost and rightmost extreme targets of S^A are determined and the server starts with the extreme, that is nearer to the origin and then catches the other extreme. If $S^A = \varnothing$, the online server intercepts the extreme target of S^T that is nearer to the origin and then the other. Each time a new target enters the system, it must be checked if the current tour has to be updated.

Theorem 6.7. Algorithm ENO achieves a competitive ratio of

$$\frac{7}{3} + \frac{20a - 4}{3(a-1)^2}.$$

Proof. We assume that time t is the time, where the last target is presented. Let $p^{\mathrm{ENO}}(t)$ be the position of the ENO server at time t and S be the set of not yet visited targets. Then, we assume, that the leftmost and rightmost targets from $p^{\mathrm{ENO}}(t)$ in S are moving away from the server, which is the worst case with the same argumentation as for NEF. Without loss of generality we suppose the leftmost target is nearer to the origin than the rightmost target and the rightmost one has been presented with positive abscissa. We call the leftmost target r_x and the rightmost target r_X, respectively. Let the positions at which r_x and r_X are presented be p_x and p_X and their positions at time t be $p_x(t)$ and $p_X(t)$. Then, we have $p_X(t) > 0$ and $|p_x(t)| \leqslant p_X(t)$. Let R be rightmost point, where a target moving in the positive direction was presented until time t and let $-L$ be either the leftmost revelation point of a target moving in the negative direction until time t or 0 in case the leftmost target was presented on the non-negative side of the line. With Z^* being the completion time of the optimal offline server and Z^{ENO} the completion time of ENO the following constraints hold:

$$\begin{aligned} Z^* &\geqslant t \\ -L - \frac{t}{a} &\leqslant p_x(t) \\ p_X(t) &\leqslant R + \frac{t}{a} \\ -L - \frac{t}{a} &\leqslant p^{\mathrm{ENO}}(t) \leqslant R + \frac{t}{a}. \end{aligned}$$

Three different cases are considered depending on the position of ENO $p^{\mathrm{ENO}}(t)$:

1. $-L - \frac{t}{a} \leqslant p^{\mathrm{ENO}}(t) \leqslant p_x(t)$
   This case is exactly the same as the case 1 for algorithm NEF on page 70, thus we simply recall the result for the competitive ratio

   $$\frac{Z^{\mathrm{ENO}}}{Z^*} \leqslant 2 + \frac{3}{a-1}. \tag{103}$$

2. $p_x(t) \leqslant p^{\mathrm{ENO}}(t) \leqslant |p_x(t)|$
   Note, that if $p_x(t) \geqslant 0$ this case reduces to $p^{\mathrm{ENO}}(t) \leqslant p_x(t)$, which is exactly the previous case. Thus, we consider $p_x(t) < 0$. The worst case for ENO is when $p^{\mathrm{ENO}}(t) = |p_x(t)|$. Then, ENO has to move to the origin, visit the leftmost extreme,

return to the origin and then catch the rightmost extreme. Thus, we obtain

$$Z^{\mathrm{ENO}} \leqslant t + \underbrace{3|p_x(t)| + \frac{4|p_x(t)|}{a-1}}_{\text{intercepting } r_x \text{ and back to the origin}} + \underbrace{p_X(t) + \frac{3|p_x(t)| + \frac{4|p_x(t)|}{a-1}}{a} + \frac{p_X(t) + \frac{3|p_x(t)| + \frac{4|p_x(t)|}{a-1}}{a}}{a-1}}_{\text{intercepting } r_X, \text{ which has moved during catching } r_x}$$

and in compact form:

$$Z^{\mathrm{ENO}} \leqslant t + 3|p_x(t)| + p_X(t) + \frac{4|p_x(t)| + p_X(t)}{a-1} + |p_x(t)|\frac{3a+1}{(a-1)^2}.$$

The offline server needs at least a completion time of

$$Z^* \geqslant 2|p_x| + R, \text{ if } p_x < 0 \text{ and}$$
$$Z^* \geqslant R, \text{ if } p_x \geqslant 0.$$

Note, that in the latter case we have $p_x \leqslant p_X \leqslant R$, it is not possible that $p_x > p_X$, because at time t either r_x or r_X is already visited or is not an extreme target that is moving away from ENO anymore, which is a contradiction in either case. At first, we consider $p_x < 0$. Here, we apply $|p_x(t)| \leqslant |p_x| + \frac{t}{a}$ and $p_X(t) \leqslant p_X + \frac{t}{a} \leqslant R + \frac{t}{a}$ and obtain for Z^{ENO}:

$$Z^{\mathrm{ENO}} \leqslant t + 3|p_x| + R + \frac{4|p_x| + R}{a-1} + |p_x|\left(\frac{3a+1}{(a-1)^2}\right) + \frac{4ta}{(a-1)^2}.$$

Thus, we have the following competitive ratio:

$$\begin{aligned}\frac{Z^{\mathrm{ENO}}}{Z^*} &\leqslant \frac{t}{t} + \frac{3|p_x| + R}{2|p_x| + R} + \frac{4|p_x| + R}{(2|p_x| + R)(a-1)} + \frac{|p_x|}{3|p_x|}\left(\frac{3a+1}{(a-1)^2}\right) + \frac{4ta}{(a-1)^2} \\ &\leqslant 1 + \frac{4}{3} + \frac{1}{a-1} + \frac{2}{3(a-1)} + \frac{3a+1}{3(a-1)^2} + \frac{4a}{(a-1)^2} \\ &= \frac{7}{3} + \frac{20a-4}{3(a-1)^2}. \end{aligned} \tag{104}$$

In the other case $p_x \geqslant 0$, we apply $|p_x(t)| \leqslant -p_x + \frac{t}{a}$ and also $p_X(t) \leqslant R + \frac{t}{a}$ to the upper bound of Z^{ENO}:

$$Z^{\mathrm{ENO}} \leqslant t + R - 3p_x + \frac{R - 4p_x}{a-1} - p_x\left(\frac{3a+1}{(a-1)^2}\right) + \frac{4ta}{(a-1)^2}.$$

Then, we have the competitive ratio

$$\begin{aligned}\frac{Z^{\mathrm{ENO}}}{Z^*} &\leqslant \frac{t}{t} + \frac{R - 3p_x}{R} + \frac{R - 4p_x}{R(a-1)} + \frac{4ta}{t(a-1)^2} \\ &\leqslant 1 + 1 + \frac{1}{a-1} + \frac{4a}{(a-1)^2} \\ &= 2 + \frac{5a-1}{(a-1)^2}. \end{aligned} \tag{105}$$

3. $|p_x(t)| < p^{\mathrm{ENO}}(t) \leqslant R+\frac{t}{a}$

We consider two different cases: a) in the optimal offline solution r_x is visited after R, b) in the optimal offline solution R is visited after r_x.

a) $R \to r_x$

Then, we have $Z^* \geqslant 2R-p_x$ and p_x can be on the right or on the left half-line. At time t, the worst case for ENO is, when it is close to the rightmost extreme $p_X(t)$ and $p_X(t) \leqslant R+\frac{t}{a}$. Then, ENO has to move to r_x and back to the rightmost extreme. The upper bound is

$$\begin{aligned} Z^{\mathrm{ENO}} &\leqslant t + 2R+\frac{2t}{a}-2p_x(t) + \frac{2R+\frac{2t}{a}-2p_x(t)}{a-1} + \frac{2R+\frac{2t}{a}-2p_x(t)}{a} \\ &\quad + \frac{4R+\frac{4t}{a}-4p_x(t)}{a(a-1)} + \frac{2R+\frac{2t}{a}-2p_x(t)}{a(a-1)^2} \\ &\leqslant t + 2R-2p_x(t) + \frac{2R-2p_x(t)}{a-1} + (2R-2p_x(t))\,\frac{a}{(a-1)^2} + \frac{2ta}{(a-1)^2}. \end{aligned}$$

Next, we apply $-p_x(t) \leqslant \frac{t}{a}-p_x$ (this holds no matter if $p_x(t)$ is negative or not). Thus, we have

$$Z^{\mathrm{ENO}} \leqslant t + 2R-2p_x + \frac{2R-2p_x}{a-1} + (2R-2p_x)\,\frac{a}{(a-1)^2} + \frac{4ta}{(a-1)^2}. \tag{106}$$

Then, the competitive ratio results in

$$\begin{aligned} \frac{Z^{\mathrm{ENO}}}{Z^*} &\leqslant \frac{t}{t} + \frac{2R-2p_x}{2R-p_x} + \frac{2R-2p_x}{(2R-p_x)(a-1)} + \frac{(2R-2p_x)a}{(2R-p_x)(a-1)^2} + \frac{4ta}{t(a-1)^2} \\ &\leqslant 1 + \frac{4}{3} + \frac{4}{3(a-1)} + \frac{4a}{3(a-1)^2} + \frac{4a}{(a-1)^2} \\ &= \frac{7}{3} + \frac{20a-4}{3(a-1)^2}. \end{aligned} \tag{107}$$

b) $r_x \to R$

The optimal offline server visits r_x at time d and it is the last time the server is at this position, note that $d \geqslant |p_x|$. Then, the offline adversary needs to travel at least from p_x to R. Thus, $Z^* \geqslant d + R - p_x$. For $d \geqslant t$ and with (106), we obtain

$$\begin{aligned} \frac{Z^{\mathrm{ENO}}}{Z^*} &\leqslant \frac{d + 2R-2p_x}{d + R-p_x} + \frac{2R-2p_x}{(R-p_x)(a-1)} + \frac{(2R-2p_x)a}{(R-p_x)(a-1)^2} + \frac{4da}{d(a-1)^2} \\ &\leqslant 2 + \frac{8a-2}{(a-1)^2} \end{aligned} \tag{108}$$

Otherwise, if $d < t$, then r_x is already presented at time d and the following claim holds:

Claim. At every time t', $d \leqslant t' \leqslant t$, we have $p^{\mathrm{ENO}}(t') \geqslant |p_x(t')|$.

Proof of the Claim. The target r_x is presented at time d or earlier. Suppose $p^{\mathrm{ENO}}(t') < p_x(t')$, then r_x is already visited at time t, because at time t ENO is to the right-hand side of r_x. This is a contradiction and proves the claim for $p_x(t') \geqslant 0$. For $p_x(t') < 0$, we suppose $p^{\mathrm{ENO}}(t') < |p_x(t')|$. Note, that r_x is the

leftmost unvisited target at time t and it must have been so at time d. Thus, to arrive at the right-hand side of $|p_x(t)|$ at time t, ENO should travel away from r_x. However, this is only possible, when the rightmost target at time t' is less or equal to $|p_x(t')|$, which contradicts to $|p_x(t)| < p^{\mathrm{ENO}}(t)$ as we are assuming in this case.

Now, the proof of Theorem 6.7 continues. Since ENO is to the right of $|p_x(d)|$ at time d and it is constantly moving left towards r_x from time d on at least until time t. Note, that any target, that is presented between r_x and ENO and is nearer to the origin than r_x does not force ENO to change its current tour, because r_x is the leftmost not yet served target from time d to time t. Thus, we can use (106) with the time d instead of time t as the upper bound for ENO. Then, the competitive ratio is

$$\begin{aligned}\frac{Z^{\mathrm{ENO}}}{Z^*} &\leqslant \frac{d+2R-2p_x}{d+R-p_x} + \frac{2R-2p_x}{(R-p_x)(a-1)} + \frac{(2R-2p_x)a}{(R-p_x)(a-1)^2} + \frac{4da}{d(a-1)^2} \\ &\leqslant 2 + \frac{8a-2}{(a-1)^2} \end{aligned} \tag{109}$$

Over all considered cases ENO has a competitive ratio of

$$\frac{7}{3} + \frac{20a-4}{3(a-1)^2}.$$

□

Note, that for $a \to \infty$ the competitive ratio reduces to $\frac{7}{3}$ for the static case (OLTSP). This matches the result, presented by Ausiello et al. [Aus01].

Discussion of the Results

The online algorithms NEF and ENO have different competitive ratios regarding the static case. ENO with $\frac{7}{3}$ is slightly better than NEF with $\frac{5}{2}$. However, in the case of catching moving targets the situation is more complex. The competitive ratios as functions of the speed ratio a of both algorithms are displayed in Figure 21. The result, which algorithm is better, depends on the speed ratio a. The visualization shows and this can also be verified analytically, that for $1 < a \leqslant 15$ the competitive ratio of NEF is better than that of ENO. However, for $a > 15$ the situation changes and the competitive ratio of ENO is better. Since the concept of competitive ratio is a worst case consideration, we cannot decide which algorithm is better in general or in practice. It is only possible to compare the worst case situations of both algorithm and therewith the competitive ratios. This means, that for fast targets ($a \leqslant 15$) NEF can handle its worst case better than ENO. Comparing both worst cases with slow targets ($a > 15$) ENO defeats NEF.

We can use the speed ratio, that is used in our test instances to calculate a corresponding competitive ratio. For all test instances a speed ratio of $a = 6.25$ was applied. With discretization level D16 the speed of the salespersons is 100 length units per time step and the speed of the targets is 16 length units per time step. Thus, we obtain a competitive ratio of 3.47.

Comparing the competitive ratios of NEF and ENO with the lower bound of $2 + \frac{1}{a-1}$ (see Theorem 6.3), it is obvious that there is still a gap between lower and upper bounds for the open OLMTTSP on the real line. For the static case, which is the open OLTSP on the real line, the very recent literature contribution Bjelde et al. [Bje21] closed the

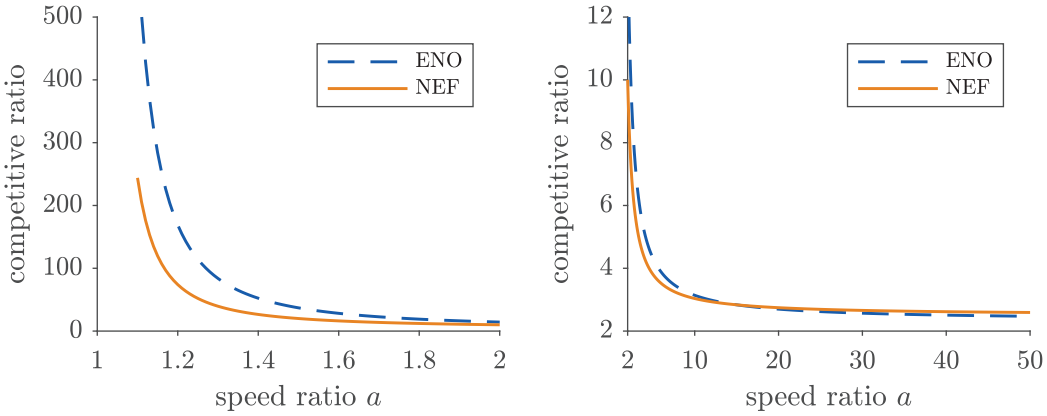

Figure 21: Comparison of the competitive ratios of NEF and ENO for the OLMTTSP on the line.

gap between the lower bound of 2 and the upper bound of 2.33 (see Table 3). The authors provided a lower bound of 2.04 which is the second-largest root of the polynomial $9\rho^4 - 18\rho^3 - 78\rho^2 + 210\rho - 107$. Throughout their article the assumption $t \geqslant |x|$ is made, for all requests $r = (x, t)$. This is not a restriction, since a server cannot serve a request earlier than its release time and an early release only supports the online algorithm. The proof of the lower bound is about an adversarial strategy, that forces the makespan of the online algorithm to be larger than the makespan of the optimal offline algorithm by a factor of at least 2.04. The idea is to present alternatingly leftmost and rightmost extreme requests until a pair of critical (with certain conditions) requests appear. Then, it is shown that additional requests can be released such that the resulting competitive ratio is at least 2.04. Waiting is allowed, however, only for a while to be competitive. Thus, different cases depending on the possible behavior of the online algorithm are analysed and the termination of the procedure is shown. Moreover, the authors also present an online algorithm with a competitive ratio of 2.04, which matches their lower bound. The strategy of the algorithm is to move to the origin and to wait there whenever possible, and to serve the extreme requests as late as possible that the competitive ratio of 2.04 is still guaranteed. The concept behind this, is to wait for more information and to delay the decision in which order to serve the extreme requests as long as possible. With this, at any time the ratio defined by the distance between server and origin and the time is bounded below 1 and this is important for the analysis of the algorithm. Depending on this bound, on distances of the extreme requests, and thus on possible waiting the decision is taken, which of the extreme requests to serve first. The proof is very technical and it analyses a comprehensive set of possible situations and cases.

To close the gap for the open OLMTTSP an approach similar to the one presented by Bjelde et al. [Bje21] might be possible. We suppose the beneficial concept of waiting is much more advanced in analysing with regard to mainly two points. Firstly, the targets are moving. Thus simply waiting increases the distance between server and target and thus increases the makespan bidirectionally. The second point is the ratio between maximal server speed and target speed: a. It might be necessary to change the strategy of the online algorithm based on a. Closing or improving the gap for the open OLMTTSP on the line is a challenge for future research.

7 Conclusion

At first, we have investigated the MTSPMT as an offline problem. The main difference to the classical TSP is, that the targets are moving continuously on trajectories, resulting in continuously varying travel times and distances between any two targets. Different modeling approaches have been provided and we have investigated the impact of the modeling variant on the optimization time for randomly generated test instances.

The best runtimes over nearly all instances could be obtained with the time-discrete model TD, where feasible arcs with travel lengths and times are modeled in a time-expanded network. However, the time-discrete model TFTD is the best choice for small instances (8 targets, 1 or 2 salespersons and discretization level D8). This could also be used by embedding the model into a set partitioning approach, since here, a sequence of TSPMT (1 salesperson) has to be solved.

Models, which use continuous time variables, have turned out to be harder to solve than discrete ones and thus, have required more computational time. One thing, that contributes to a higher complexity is, that the decision, whether an arc is feasible (a salesperson can use it with at most maximum speed) is inserted into the model. Comparing TC and TFTC, it can bee seen, that for small instances (up to 10 targets and up to 3 salespersons) better scores could be achieved with TC than with TFTC. This changes for instances with more than 10 targets or more than 3 salespersons. Usually, these instances could not be solved within the time limit with none of the time-continuous models. However, relative gaps are lower with TFTC than with TC for those instances.

Furthermore, the computational experiments have shown, that the discrete model TD can easily handle non-linear trajectories, while the continuous model TC not. For the TC model not only the computational burden resulting from the piece-wise linear approximation of the trajectories has been an issue, but also the displacement between the interpolated interception point and the real target position.

To process real-world test instances, we have adapted our models to handle different depot positions and infeasible instances. From the mathematical point of view, the MTSPMT is an online problem, thus, we have solved test instances in an online consideration. We have choosen TD for modeling, due to its offline result, the TD model was able to solve instances up to 10 targets within 3 sec in average. The two online strategies REPLAN and IGNORE have been adapted to multiple salespersons and moving targets and applied to our test instances. We have investigated the number of misses in intercepting the targets. Although, the IGNORE strategy has resulted in slightly more misses than the REPLAN strategy, we have computed a worst case ratio of the obtained online and offline objective function values of 1.55 for both strategies. This value can be seen as a practically obtained competitive ratio.

We also addressed the OLMTTSP on the real line without time windows from the theoretical point of view of competitive analysis. Here, we have proven a lower bound for the competitive ratio. Furthermore, we developed an online algorithm and adapted another one from the literature. We have proven competitive ratios for both online algorithms. The decision which algorithm has a better competitive ratio depends on the speed ratio of salespersons and targets. We have applied the speed ratio of our test instances and obtained an analytical competitive ratio of 3.47 for the OLMTTSP.

If we consider the OLMTTSP on the real line with all targets moving away from the server and apply the REPLAN strategy with the min-dist objective function as the online algorithm on the real line, then this online strategy behaves equally to the presented NEF online algorithm. On the real line the salesperson has to move with maximum speed no matter if

REPLAN or NEF is used, thus, for the competitive ratio it does not matter if we use min-dist or min-time as objective function. We can compare the practically obtained competitive ratio 1.55 and the analytically determined one 3.47. Obviously, the competitive ratio obtained in practice is much better than the theoretically obtained competitive ratio. Even though the competitive ratio obtained in practice is a worst case value and the number of instances is small, this result underscores the criticism about competitive analysis regarding its overly pessimistic evaluation.

Our theoretical results in competitive analysis obtained for the open OLMTTSP on the real line are new to this area. Closing or improving the gap is a challenge for future research.

8 References

[Abe13] Abeledo, H. G., R. Fukasawa, A. A. Pessoa, and E. Uchoa: 'The time dependent traveling salesman problem: polyhedra and algorithm'. *Mathematical Programming Computation* (2013), vol. 5: pp. 27–55 (cit. on p. 5).

[Ahr15] Ahrens, B.: 'The tour construction framework for the dynamic Travelling Salesman Problem'. *SoutheastCon 2015*. Apr. 2015: pp. 1–8 (cit. on p. 4).

[Ahu93] Ahuja, R. K., T. L. Magnanti, and J. B. Orlin: *Network Flows: Theory, Algorithms, and Applications*. Prentice hall, 1993 (cit. on p. 24).

[Alb08] Albiach, J., J. M. Sanchis, and 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* (2008), vol. 189(3): pp. 789–802 (cit. on pp. 5, 6).

[Apr09] Aprea, M., E. Feuerstein, G. Sadovoy, and A. S. de Loma: 'Discrete online TSP'. *Algorithmic Aspects in Information and Management*. Ed. by Goldberg, A. V. and Y. Zhou. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009: pp. 29–42 (cit. on pp. 64, 66).

[Aus95] Ausiello, G., E. Feuerstein, S. Leonardi, L. Stougie, and M. Talamo: 'Competitive algorithms for the on-line traveling salesman'. *Algorithms and Data Structures*. Ed. by Akl, S. G., F. Dehne, J.-R. Sack, and N. Santoro. Berlin, Heidelberg: Springer Berlin Heidelberg, 1995: pp. 206–217 (cit. on p. 66).

[Aus01] Ausiello, G., E. Feuerstein, S. Leonardi, L. Stougie, and M. Talamo: 'Algorithms for the on-line travelling salesman'. *Algorithmica* (2001), vol. 29(4): pp. 560–581 (cit. on pp. d, f, 64, 66–70, 82, 83, 87).

[Bje17] Bjelde, A., Y. Disser, J. Hackfeld, C. Hansknecht, M. Lipmann, J. Meissner, K. Schewior, M. Schlöter, and L. Stougie: 'Tight bounds for online TSP on the line'. *Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 2017: pp. 994–1005 (cit. on p. 64).

[Bje21] Bjelde, A., J. Hackfeld, Y. Disser, C. Hansknecht, M. Lipmann, J. Meissner, M. SchlÖter, K. Schewior, and L. Stougie: 'Tight Bounds for Online TSP on the Line'. *ACM Trans. Algorithms* (Dec. 2021), vol. 17(1) (cit. on pp. 66, 87, 88).

[Blo00] Blom, M., S. O. Krumke, W. de Paepe, and L. Stougie: 'The Online-TSP against Fair Adversaries'. *Algorithms and Complexity*. Ed. by Bongiovanni, G., R. Petreschi, and G. Gambosi. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000: pp. 137–149 (cit. on pp. 63, 65, 66).

[Bor98] Borodin, A. and R. El-Yaniv: *Online Computation and Competitive Analysis*. USA: Cambridge University Press, 1998 (cit. on p. 63).

[Boy07] Boyar, J. and L. M. Favrholdt: 'The Relative Worst Order Ratio for Online Algorithms'. *ACM Trans. Algorithms* (2007), vol. 3(2): pp. 1–19 (cit. on p. 64).

[Boy04] Boyd, S. and L. Vandenberghe: *Convex Optimization*. Cambridge University Press, 2004 (cit. on pp. 8, 17–19, 21).

[Chv83] Chvátal, V.: *Linear Programming*. W.H. Freeman and Company, New York, 1983 (cit. on p. 8).

[Cod04] Codato, G. and M. Fischetti: 'Combinatorial Benders' Cuts'. *Integer Programming and Combinatorial Optimization*. Ed. by Bienstock, D. and G. Nemhauser. Vol. 3064. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2004: pp. 178–195 (cit. on p. 33).

[Con14] Conforti, M., G. Cornuéjols, and G. Zambelli: *Integer Programming*. Switzerland: Springer International Publishing, 2014 (cit. on pp. 15–17).

[Coo97] Cook, W. J., W. H. Cunningham, W. R. Pulleyblank, and A. Schrijver: *Combinatorial Optimization*. A Wiley-Interscience publication. Wiley, 1997 (cit. on p. 8).

[Cor08] Cornuéjols, G.: 'Valid inequalities for mixed integer linear programs'. *Mathematical Programming* (2008), vol. 112(1): pp. 3–44 (cit. on p. 16).

[Dak65] Dakin, R. J.: 'A tree-search algorithm for mixed integer programming problems'. *The Computer Journal* (Jan. 1965), vol. 8(3): pp. 250–255 (cit. on p. 15).

[Dan54] Dantzig, G., R. Fulkerson, and S. Johnson: 'Solution of a Large-Scale Traveling-Salesman Problem'. *Journal of the Operations Research Society of America* (1954), vol. 2(4): pp. 393–410 (cit. on p. 16).

[Dan90] Dantzig, G. B.: 'Origins of the Simplex Method'. *A History of Scientific Computing*. New York, NY, USA: Association for Computing Machinery, 1990: pp. 141–151 (cit. on p. 9).

[Die17] Diestel, R.: *Graph Theory*. Springer, Berlin, Heidelberg, 2017 (cit. on p. 23).

[Dij59] Dijkstra, E. W.: 'A Note on Two Problems in Connexion with Graphs'. *NUMERISCHE MATHEMATIK* (1959), vol. 1(1): pp. 269–271 (cit. on p. 36).

[Eng13] Englot, B. J., T. Sahai, and I. Cohen: 'Efficient tracking and pursuit of moving targets by heuristic solution of the traveling salesman problem'. *CDC*. IEEE, 2013: pp. 3433–3438 (cit. on pp. 3, 4).

[FIC21] FICO: *FICO Xpress Optimization*. `https://www.fico.com`. July 2021 (cit. on p. 16).

[Fle04] Fleischmann, B., M. Gietz, and S. Gnutzmann: 'Time-Varying Travel Times in Vehicle Routing'. *Transportation Science* (2004), vol. 38(2): pp. 160–173 (cit. on p. 5).

[For62] Ford, L. R. and D. R. Fulkerson: *Flows in Networks*. Princeton University Press, 1962 (cit. on p. 24).

[Füg14] Fügenschuh, A., M. Knapp, and H. Rothe: *The Multiple Traveling Salesmen Problem with Moving Targets*. Tech. rep. 12. July 2014 (cit. on p. i).

[Füg13] Fügenschuh, A., G. Nemhauser, and Y. Zeng: 'Scheduling and Routing of Fly-in Safari Planes Using a Flow-over-Flow Model'. English. *Facets of Combinatorial Optimization*. Ed. by Jünger, M. and G. Reinelt. Springer Berlin Heidelberg, 2013: pp. 419–447 (cit. on p. 34).

[Gar79] Garey, M. R. and D. S. Johnson: *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, New York, 1979 (cit. on pp. 1, 8, 25).

[Ghi00] Ghiani, G. and G. Improta: 'An efficient transformation of the generalized vehicle routing problem'. *European Journal of Operational Research* (2000), vol. 122: pp. 11–17 (cit. on p. 5).

[GNU21] GNU project: *GLPK - GNU Linear Programming Kit*. `https://www.gnu.org/software/glpk/`. July 2021 (cit. on p. 16).

[Gom58] Gomory, R. E.: 'Outline of an algorithm for integer solutions to linear programs'. *Bulletin of the American Mathematical Society* (1958), vol. 64(5): pp. 275–278 (cit. on p. 16).

[Grö09] Grötschel, M.: *Lineare und Ganzzahlige Programmierung (Algorithmische Diskrete Mathematik II)*. Skriptum zur Vorlesung im WS 2009/2010, Institut für Mathematik, TU Berlin. 2009 (cit. on pp. 10, 13, 14, 18).

[Grö01a] Grötschel, M., S. Krumke, and J. Rambau: 'Online Optimization of Complex Transportation Systems'. English. *Online Optimization of Large Scale Systems*. Ed. by Grötschel, M., S. O. Krumke, and J. Rambau. Springer, Sept. 2001: pp. 705–729 (cit. on p. 58).

[Grö01b] Grötschel, M., S. Krumke, J. Rambau, T. Winter, and U. Zimmermann: 'Combinatorial Online Optimization in Real Time'. *Online Optimization of Large Scale Systems*. Ed. by Grötschel, M., S. Krumke, and J. Rambau. 2001: pp. 679–704 (cit. on pp. 58, 62, 64).

[Grö88] Grötschel, M., L. Lovász, and A. Schrijver: *Geometric Algorithms and Combinatorial Optimization*. Vol. 2. Springer, 1988 (cit. on p. 8).

[Grö93] Grötschel, M., L. Lovász, and A. Schrijver: *Geometric Algorithms and Combinatorial Optimization*. Vol. 2. Springer-Verlag Berlin Heidelberg, 1993 (cit. on pp. 22, 24, 25).

[Gur21] Gurobi Optimization: *Gurobi Optimizer*. `https://www.gurobi.com`. July 2021 (cit. on p. 16).

[Gut06] Gutiérrez, S., S. Krumke, N. Megow, and T. Vredeveld: 'How to whack moles'. *Theoretical Computer Science* (2006), vol. 361(2). Approximation and Online Algorithms: pp. 329–341 (cit. on p. 64).

[Hag05] Haghani, A. and S. Jung: 'A dynamic vehicle routing problem with time-dependent travel times'. *Computers & Operations Research* (2005), vol. 32(11): pp. 2959–2986 (cit. on p. 5).

[Has20] Hassoun, M., S. Shoval, E. Simchon, and L. Yedidsion: 'The single line moving target traveling salesman problem with release times'. *Annals of Operations Research* (June 2020), vol. 289(2): pp. 449–458 (cit. on p. 4).

[Hay03] Hayes, B.: 'A Lucid Interval'. *American Scientist* (2003), vol. 91: p. 484 (cit. on p. 26).

[Hel98] Helvig, C. S., G. Robins, and A. Zelikovsky: 'Moving-Target TSP and Related Problems'. *Proceedings of the European Symposium on Algorithms*. Ed. by G. Bilardi G. F. Italiano, A. P. and G. Pucci. Vol. 1461. Lecture Notes in Computer Science. Springer Verlag, Berlin, 1998: pp. 453–464 (cit. on pp. 3, 4).

[Hel03] Helvig, C. S., G. Robins, and A. Zelikovsky: 'The moving-target traveling salesman problem'. *Journal of Algorithms* (2003), vol. 49(1): pp. 153–174 (cit. on pp. 3, 4, 67).

[IBM17] IBM: *ILOG CPLEX Optimization Studio*. Documentation available at `https://www.ibm.com/support/knowledgecenter/SSSA5P_12.7.1/ilog.odms.studio.help/Optimization_Studio/topics/COS_home.html`. Sept. 2017 (cit. on pp. 16, 17, 48).

[Ich03] Ichoua, S., M. Gendreau, and J.-Y. Potvin: 'Vehicle dispatching with time-dependent travel times'. *European Journal of Operational Research* (2003), vol. 144(2): pp. 379–396 (cit. on p. 5).

[Jai11] Jaillet, P. and X. Lu: 'Online traveling salesman problems with service flexibility'. *Networks* (2011), vol. 58(2): pp. 137–146 (cit. on pp. 64, 66).

[Jai08] Jaillet, P. and M. R. Wagner: 'Generalized Online Routing: New Competitive Ratios, Resource Augmentation, and Asymptotic Analyses'. *Operations Research* (2008), vol. 56(3): pp. 745–757 (cit. on p. 66).

[Jau02] Jaulin, L., I. Braems, and E. Walter: 'Interval methods for nonlinear identification and robust control'. *In Proceedings of the 41st IEEE Conference on Decision and Control (CDC)*. 9-13 decembre 2002, Las Vegas, 2002 (cit. on p. 26).

[Jau01] Jaulin, L., M. Kieffer, O. Didrit, and E. Walter: *Applied Interval Analysis*. Springer-Verlag London, 2001 (cit. on p. 26).

[Jia05] Jiang, Q., R. Sarker, and H. Abbass: 'Tracking moving targets and the non-stationary traveling salesman problem'. *Complexity International* (2005), vol. 11: pp. 171–179 (cit. on pp. 3, 4).

[Jin11] Jindal, P., A. Kumar, and S. Kumar: 'Multiple Target Intercepting Traveling Salesman Problem'. *International Journal on Computer Science and Technology* (2011), vol. 2(2): pp. 327–331 (cit. on p. 4).

[Jun01] Jung, S. and A. Haghani: 'Genetic Algorithm for the Time-Dependent Vehicle Routing Problem'. *Transportation Research Record: Journal of the Transportation Research Board* (2001), vol. 1771: pp. 164–171 (cit. on p. 5).

[Kal00] Kalyanasundaram, B. and K. Pruhs: 'Speed is as Powerful as Clairvoyance'. *J. ACM* (2000), vol. 47(4): pp. 617–643 (cit. on p. 64).

[Kar84] Karmarkar, N.: 'A new polynomial-time algorithm for linear programming'. *Combinatorica* (1984), vol. 4(4): pp. 373–395 (cit. on p. 18).

[Kar72] Karp, R. M.: 'Reducibility Among Combinatorial Problems.' *Complexity of Computer Computations*. Ed. by Miller, R. E. and J. W. Thatcher. The IBM Research Symposia Series. Plenum Press, New York, 1972: pp. 85–103 (cit. on p. 1).

[Kha79] Khachiyan, L.: 'A Polynomial Algorithm in Linear Programming'. *Doklady Akademii Nauk SSSR* (1979), vol. 244: pp. 1093–1096 (cit. on p. 13).

[Kna12] Knapp, M. and H. Rothe: 'Concept for Simulating Engagement Strategies for C-RAM Systems using Laser Weapons'. *Proceedings of the DMMS*. 2012 (cit. on pp. c, e, 2).

[Kor00] Korte, B. and J. Vygen: *Combinatorial Optimization Theory and Algorithms*. Springer-Verlag Berlin Heidelberg, 2000 (cit. on pp. 8, 23).

[Kou94] Koutsoupias, E. and C. H. Papadimitriou: 'Beyond competitive analysis'. *Proceeding 35th Annual Symposium on Foundations of Computer Science*. Final version in Siam J. on Comp. Santa Fe, New Mexico, 1994: pp. 394–400 (cit. on p. 63).

[Kou09] Koutsoupias, E.: 'The k-server problem'. *Computer Science Review* (2009), vol. 3(2): pp. 105–118 (cit. on p. 62).

[Lan60] Land, A. H. and A. G. Doig: 'An Automatic Method of Solving Discrete Programming Problems'. *Econometrica* (1960), vol. 28(3): pp. 497–520 (cit. on p. 15).

[Lap87] Laporte, G., H. Mercure, and Y. Nobert: 'Generalized travelling salesman problem through n sets of nodes: the asymmetrical case'. *Discrete Applied Mathematics* (1987), vol. 18(2): pp. 185–197 (cit. on p. 4).

[Law85] Lawler, E. L., J. K. Lenstra, A. H. G. Rinnooy, and D. B. Shmoys: *The Traveling Salesman Problem: a Guided Tour of Combinatorial Optimization*. John Wiley and Sons, Chichester, New York, 1985 (cit. on p. 1).

[Lip03] Lipmann, M.: 'On-Line Routing'. English. PhD thesis. Department of Mathematics and Computer Science, 2003 (cit. on pp. 62–64, 66).

[Lob98] Lobo, M. S., L. Vandenberghe, S. Boyd, and H. Lebret: 'Applications of second-order cone programming'. *Linear Algebra and its Applications* (1998), vol. 284(1). International Linear Algebra Society (ILAS) Symposium on Fast Algorithms for Control, Signals and Image Processing: pp. 193–228 (cit. on p. 17).

[Mal92] Malandraki, C. and M. S. Daskin: 'Time Dependent Vehicle Routing Problems: Formulations, Properties and Heuristic Algorithms'. *Transportation Science* (1992), vol. 26(3): pp. 185–200 (cit. on p. 5).

[Man14] Mancini, S.: 'Time Dependent Travel Speed Vehicle Routing and Scheduling on a Real Road Network: The Case of Torino'. *Transportation Research Procedia* (2014), vol. 3: pp. 433–441 (cit. on p. 5).

[Mar02] Marchand, H., A. Martin, R. Weismantel, and L. Wolsey: 'Cutting planes in integer and mixed integer programming'. *Discrete Applied Mathematics* (2002), vol. 123(1): pp. 397–446 (cit. on p. 16).

[Men30] Menger, K.: *Vienna Colloquium*. `https://de.wikipedia.org/wiki/Problem_des_Handlungsreisenden`. 1930 (cit. on p. 1).

[Mil60] Miller, C. E., A. W. Tucker, and R. A. Zemlin: 'Integer Programming Formulation of Traveling Salesman Problems'. *Journal of the ACM (JACM)* (1960), vol. 7(4): pp. 326–329 (cit. on p. 31).

[Nem88] Nemhauser, G. and L. Wolsey: *Integer and Combinatorial Optimization*. John Wiley & Sons, 1988 (cit. on p. 8).

[Nem05] Nemirovski, A.: *Lectures on Modern Convex Optimization*. Department ISYE, Georgia Institute of Technology, Fall Semester 05. 2005 (cit. on p. 17).

[Noo93] Noon, C. E. and J. C. Bean: 'An Efficient Transformation Of The Generalized Traveling Salesman Problem'. *INFOR: Information Systems and Operational Research* (1993), vol. 31(1): pp. 39–44 (cit. on pp. 4, 5).

[Pad87] Padberg, M. and G. Rinaldi: 'Optimization of a 532-City Symmetric Traveling Salesman Problem by Branch and Cut'. *Oper. Res. Lett.* (1987), vol. 6(1): pp. 1–7 (cit. on p. 16).

[Pad91] Padberg, M. and G. Rinaldi: 'A Branch-and-Cut Algorithm for the Resolution of Large-Scale Symmetric Traveling Salesman Problems'. *SIAM Rev.* (1991), vol. 33(1): pp. 60–100 (cit. on p. 16).

[Pap17] Papadimitriou, C. H. and K. Steiglitz: *Combinatorial Optimization: Algorithms and Complexity*. Dover Books on Computer Science. Dover Publications, 2017 (cit. on pp. 8, 10, 13).

[Phi02] Phillips, C. A., C. Stein, E. Torng, and J. Wein: 'Optimal Time-Critical Scheduling via Resource Augmentation'. *Algorithmica* (2002), vol. (2): pp. 163–200 (cit. on p. 64).

[Pic78] Picard, J.-C. and M. Queyranne: 'The Time-Dependent Traveling Salesman Problem and Its Application to the Tardiness Problem in One-Machine Scheduling'. *Operations Research* (1978), vol. 26(1): pp. 86–110 (cit. on p. 5).

[Rei94] Reinelt, G.: *The Traveling Salesman: Computational Solutions for TSP Applications*. Springer Verlag, Berlin, 1994 (cit. on pp. 1, 3).

[Rou20] Roughgarden, T.: *Resource Augmentation*. 2020 (cit. on p. 64).

[Sch98] Schrijver, A.: *Theory of Linear and Integer Programming*. John Wiley & Sons, 1998 (cit. on p. 8).

[Sku09] Skutella, M.: 'An Introduction to Network Flows over Time'. *Research Trends in Combinatorial Optimization: Bonn 2008*. Ed. by Cook, W., L. Lovász, and J. Vygen. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009: pp. 451–482 (cit. on p. 24).

[Sle85] Sleator, D. D. and R. E. Tarjan: 'Amortized Efficiency of List Update and Paging Rules'. *Commun. ACM* (1985), vol. 28(2): pp. 202–208 (cit. on p. 63).

[Sol09] Soler, D., J. Albiach, and E. Martínez: 'A Way to Optimally Solve a Time-dependent Vehicle Routing Problem with Time Windows'. *Operations Research Letters* (Jan. 2009), vol. 37(1): pp. 37–42 (cit. on p. 5).

[Sti18] Stieber, A. and A. Fügenschuh: 'The Multiple Traveling Salesmen Problem with Moving Targets and Nonlinear Trajectories'. *Operations Research Proceedings 2017*. Ed. by Kliewer, N., J. F. Ehmke, and R. Borndörfer. Cham: Springer International Publishing, 2018: pp. 489–494 (cit. on pp. i, 30, 38).

[Sti20] Stieber, A. and A. Fügenschuh: 'Dealing with time in the multiple traveling salespersons problem with moving targets'. *Central European Journal of Operations Research* (2020), vol. (cit. on p. i).

[Sti14] Stieber, A., A. Fügenschuh, M. Epp, M. Knapp, and H. Rothe: 'The multiple traveling salesmen problem with moving targets'. *Optimization Letters* (2014), vol. 9(8): pp. 1569–1583 (cit. on pp. 6, 47).

[Sun16] Sundar, K. and S. Rathinam: 'Generalized multiple depot traveling salesmen problem — Polyhedral study and exact algorithm'. *Computers & Operations Research* (2016), vol. 70: pp. 39–55 (cit. on p. 5).

[Van07] Van Woensel, T., L. Kerbache, H. Peremans, and N. Vandaele: 'A Queueing Framework for Routing Problems with Time-dependent Travel Times'. *Journal of Mathematical Modelling and Algorithms* (2007), vol. 6(1): pp. 151–173 (cit. on p. 6).

[Wen15] Wen, X., Y. Xu, and H. Zhang: 'Online Traveling Salesman Problem with Deadlines and Service Flexibility'. *J. Comb. Optim.* (2015), vol. 30(3): pp. 545–562 (cit. on pp. 64, 66).

[Zus21] Zuse Institute Berlin: *SCIP - Solving Constraint Integer Programs*. `https://www.scipopt.org`. July 2021 (cit. on p. 16).

List of Figures

List of Tables

A Appendix

A.1 Computational Offline Results

Table 5: Arithmetic means of the scores for the models TD, TFTD, TC, and TFTC. Instances are characterized by the parameters 'nbt' (number of targets), 'nbs' (number of salespersons), 'dl' (discretization level), and 'tl' (time limit). The arithmetic means of the scores are taken over 21 different randomly generated instances and are reported in columns five to eight for the different models TD, TFTD, TC, and TFTC.

instance parameter				arithmetic mean of the score for			
nbt	**nbs**	**dl**	**tl**	**TD**	**TFTD**	**TC**	**TFTC**
6	3	D16	3,600	0.0000	0.0001	0.0007	0.0006
8	3	D16	3,600	0.0001	0.0035	0.0194	0.0275
10	1	D16	3,600	0.0003	0.0160	0.0198	0.1081
10	2	D16	3,600	0.0011	0.0345	0.1064	0.3770
10	3	D32	3,600	0.0001	0.2388	|	|
10	3	D16	3,600	0.0006	0.2960	0.4271	0.6320
10	3	D8	3,600	0.0027	0.2637	|	|
10	4	D16	3,600	0.0004	0.6436	0.9040	0.8756
10	5	D16	3,600	0.0004	0.9362	1.2330	1.1248
10	6	D16	3,600	0.0004	1.0649	1.4294	1.1998
12	3	D16	3,600	0.0020	0.9627	1.4619	1.1600
14	3	D16	3,600	0.0082	1.2276	1.7290	1.2919
16	3	D16	3,600	0.0306	1.3152	1.8519	1.3838
18	3	D16	3,600	0.0442	1.3710	1.9219	1.4348
20	3	D16	3,600	0.0704	1.4181	1.9514	1.4719
8	1	D8	3	0.9985	0.4162	0.8797	1.2375
8	2	D8	3	0.5838	0.5290	1.5333	1.0998
8	3	D8	3	0.5020	0.7187	1.7650	1.1147
8	4	D8	3	0.5385	0.9823	1.8446	1.1980
8	5	D8	3	0.6320	1.1341	1.9137	1.2613
8	6	D8	3	0.7526	1.2168	1.9585	1.2952

Table 6: Arithmetic means of the runtimes in [s] for the models TD, TFTD, TC, and TFTC. Instances are characterized by the parameters 'nbt' (number of targets), 'nbs' (number of salespersons), 'dl' (discretization level), and 'tl' (time limit). The arithmetic means of the runtimes are taken over 21 different randomly generated instances and are reported in columns five to eight for the different models TD, TFTD, TC, and TFTC.

instance parameter				arithmetic means of the runtimes for			
nbt	nbs	dl	tl	TD	TFTD	TC	TFTC
6	3	D16	3,600	0.11	0.23	2.40	2.12
8	3	D16	3,600	0.38	12.42	69.86	99.16
10	1	D16	3,600	1.22	57.52	71.15	389.04
10	2	D16	3,600	4.02	124.32	383.12	1312.78
10	3	D32	3,600	0.19	842.57	|	|
10	3	D16	3,600	2.28	1,026.25	1,535.85	2,076.31
10	3	D8	3,600	9.72	920.15	|	|
10	4	D16	3,600	1.61	2,104.25	2,813.97	2,739.07
10	5	D16	3,600	1.43	2,929.89	3,357.13	3,401.13
10	6	D16	3,600	1.50	3,202.84	3,565.23	3,489.64
12	3	D16	3,600	7.26	3,038.77	3,600.00	3,439.02
14	3	D16	3,600	29.43	3,600.02	3,600.00	3,600.04
16	3	D16	3,600	110.11	3,600.02	3,600.02	3,600.04
18	3	D16	3,600	159.07	3,600.01	3,600.01	3,600.02
20	3	D16	3,600	253.42	3,600.02	3,600.01	3,600.03
8	1	D8	3	2.37	1.20	2.26	2.97
8	2	D8	3	1.62	1.52	2.86	2.73
8	3	D8	3	1.46	2.00	3.00	2.72
8	4	D8	3	1.57	2.54	3.00	2.90
8	5	D8	3	1.79	2.84	3.00	3.01
8	6	D8	3	2.10	2.98	3.00	3.01

Table 7: Arithmetic means of the runtimes [s] for the set partitioning approach. The subset tour generation is computed using the models TD, TFTD, TC, and TFTC. Instances are characterized by the parameters 'nbt' (number of targets), 'nbs' (number of salespersons), 'dl' (discretization level), and 'tl' (time limit). The arithmetic means of the runtimes are reported in columns five to eight for the different models and are taken over five different instances. For the instances with 12 targets and D8, four out of five instances are infeasible in the column of TD, all other instances are solved to optimality.

instance parameter				arithmetic mean of the runtimes [s]			
nbt	nbs	dl	tl	TD	TFTD	TC	TFTC
6	2-6	D32	3,600	0.37	0.38	|	|
6	2-6	D16	3,600	1.39	0.45	2.48	1.25

continued on next page . . .

nbt	nbs	dl	tl	TD	TFTD	TC	TFTC
6	2-6	D8	3,600	5.37	0.81	|	|
8	2-6	D32	3,600	3.12	2.97	|	|
8	2-6	D16	3,600	15.17	3.99	21.56	10.28
8	2-6	D8	3,600	91.62	8.33	|	|
10	2-6	D32	3,600	21.92	21.94	|	|
10	2-6	D16	3,600	133.85	33.61	138.82	87.58
10	2-6	D8	3,600	1,279.67	78.43	|	|
12	2-6	D32	3,600	135.94	244.98	|	|
12	2-6	D16	3,600	923.87	418.68	930.03	1,949.92
12	2-6	D8	3,600	3,137.54	1,053.75	|	|

Table 8: Runtimes and objective function values for the TD model and non-linear trajectories. Instances are characterized by the parameters 'nbt' (number of targets), and 'nbs' (number of salespersons). The runtimes [s] for the model TD are reported in columns three to five for the different discretization levels. The objective function values for the TD model are given in columns six to eight for the different discretization levels.

instance param.		**runtimes [s]**			**objective function values**		
nbt	**nbs**	**D32**	**D16**	**D8**	**D32**	**D16**	**D8**
4	2	0.02	0.04	0.13	295.4	294.1	291.2
4	4	0.03	0.07	0.27	292.7	284.2	282.0
6	2	0.05	0.23	0.95	449.2	434.6	431.1
6	4	0.08	0.44	1.26	437.6	423.4	422.0
6	6	0.12	0.77	2.31	437.6	423.4	422.0
8	2	0.21	1.85	3.76	584.3	531.6	521.9
8	4	0.09	0.88	2.95	531.6	492.9	484.0
8	6	0.13	1.28	4.51	531.6	492.9	484.0
10	2	0.27	8.26	143.83	849.3	819.2	818.3
10	4	0.24	1.32	4.24	716.1	678.3	666.2
10	6	0.40	1.61	7.40	716.1	678.3	666.2
12	2	0.29	3.41	33.73	998.2	967.6	965.5
12	4	0.33	2.06	7.86	865.1	826.7	813.2
12	6	0.51	1.69	10.98	865.1	826.7	813.2
14	2	0.77	10.35	151.66	1,187.3	1,139.8	1,136.9
14	4	0.51	2.55	10.69	955.1	924.8	911.8
14	6	0.36	2.74	17.59	948.6	910.8	898.3
16	2	2.48	15.40	409.59	1,321.4	1,281.0	1,276.7
16	4	1.33	6.91	60.10	1,071.8	1,034.3	1,022.0
16	6	1.00	4.21	38.96	1,039.1	994.7	982.5

Table 9: Results for the TC model and non-linear trajectories. Instances are characterized by the parameters 'nbt' (number of targets), and 'nbs' (number of salespersons). The column 'dl' (discretization level) is related to the model TD, which is applied to provide a mip start to the continuous optimization. The columns four to seven report runtimes, objective function values (ofv), gaps, and whether a mip start could be used or not. Runtime values for the TD model are given in parenthesis. A time limit of 3,600 sec is used.

instance parameter			runtimes [s] and results			mip start
nbt	nbs	dl	runtime TC (TD)	ofv TC (TD)	gap TC	used
4	2	D32	34.76 (0.02)	292.7 (295.4)	0.00	yes
4	2	D16	3.22 (0.04)	292.7 (294.1)	0.00	no
4	2	D8	4.01 (0.13)	292.7 (291.2)	0.00	yes
4	4	D32	3,600.00 (0.03)	334.7 (292.7)	0.68	no
4	4	D16	3,600.00 (0.07)	340.7 (284.2)	0.76	no
4	4	D8	3,600.00 (0.27)	283.3 (282.0)	0.71	yes
6	2	D32	3,600.00 (0.05)	433.1 (449.2)	0.44	yes
6	2	D16	3,600.00 (0.22)	436.7 (434.6)	0.56	yes
6	2	D8	3,600.00 (0.96)	453.5 (431.1)	0.35	yes
6	4	D32	3,600.00 (0.08)	428.5 (437.6)	0.88	yes
6	4	D16	3,600.00 (0.44)	443.6 (423.4)	0.88	yes
6	4	D8	3,600.00 (1.26)	425.2 (422.0)	0.88	yes

A.2 Computational Online Results

Table 10: Averaged number of online instances without misses and the sum of all targets, that could not be intercepted for the online strategies REPLAN and IGNORE. Instances are characterized by the parameters 'nbt' (number of targets), 'nbs' (number of salespersons), and 'dl' (discretization level). For each number of targets 'nbt' 21 instances are generated. The columns with title 'nbiwm' contains the number of instances without any misses, while the columns with title 'nbm' show the sum of misses (targets that could not be intercepted) over all 21 instances.

instance parameter			replan		ignore	
nbt	nbs	dl	nbiwm	nbm	nbiwm	nbm
8	2	D16	8	17	6	20
10	2	D16	5	30	5	34
12	2	D16	2	37	1	51
14	2	D16	1	50	0	62
16	2	D16	1	72	0	95
18	2	D16	0	100	0	132
20	2	D16	0	113	0	141
8	4	D16	20	1	20	1

continued on next page ...

nbt	nbs	dl	nbiwm	nbm	nbiwm	nbm
10	4	D16	18	3	18	3
12	4	D16	18	3	18	3
14	4	D16	19	3	19	3
16	4	D16	13	9	12	10
18	4	D16	11	13	10	15
20	4	D16	13	9	11	11

Table 11: Objective function values online (REPLAN) and offline based on the same instances are listed. Instances are characterized by the parameters 'nbt' (number of targets), 'nbs' (number of salespersons), and 'dl' (discretization level). The online objective function values are given in column 'online' and the corresponding offline objective function values in column 'offline'. The ratio of both values (online divided by offline) is given in column 'ratio'.

— instance parameter —			— objective function values —		
nbt	**nbs**	**dl**	**online**	**offline**	**ratio**
8	2	D16	1,058.00	774.91	1,37
8	2	D16	1,170.66	1,067.48	1.10
8	2	D16	1,031.53	792.37	1.30
8	2	D16	1,331.65	1,057.79	1.26
8	2	D16	912.77	759.26	1.20
8	2	D16	1,198.52	1,076.56	1.11
8	2	D16	820.73	804.58	1.02
8	2	D16	967.37	889.50	1,09
10	2	D16	865.29	771.48	1.12
10	2	D16	1,383.33	1,164.77	1.19
10	2	D16	1,240.95	798.23	1.55
10	2	D16	1,302.55	1,254.52	1.04
10	2	D16	1,338.47	1,144.37	1.17
12	2	D16	1,114.89	916.38	1.22
12	2	D16	1,153.22	1,056.80	1.09
14	2	D16	1,554.49	1,312.45	1.18
16	2	D16	1,564.87	1,296.85	1.21

Table 12: Objective function values online (IGNORE) and offline based on the same instances are listed. Instances are characterized by the parameters 'nbt' (number of targets), 'nbs' (number of salespersons), and 'dl' (discretization level). The online objective function values are given in column 'online' and the corresponding offline objective function values in column 'offline'. The ratio of both values (online divided by offline) is given in column 'ratio'.

— instance parameter —			— objective function values —		
nbt	**nbs**	**dl**	**online**	**offline**	**ratio**
8	2	D16	1,055.22	774.91	1.36
8	2	D16	1,170.66	1,067.48	1.10
8	2	D16	1,331.65	1,057.79	1.26
8	2	D16	908.44	759.26	1.20
8	2	D16	1,198.52	1,076.56	1.11
8	2	D16	967.37	889.50	1.09
10	2	D16	865.29	771.48	1.12
10	2	D16	1,383.33	1,164.77	1.19
10	2	D16	1,240.95	798.23	1.55
10	2	D16	1,326.33	1,254.52	1.06
10	2	D16	1,244.90	1,144.37	1.09
12	2	D16	1,114.89	916.38	1.22

Table 13: Averaged number of online instances without misses and the sum of all targets, that could not be intercepted for the online strategies REPLAN and IGNORE. Instances are characterized by the parameters 'nbt' (number of targets), 'nbs' (number of salespersons), and 'dl' (discretization level). For each number of targets 'nbt' 21 instances are generated. The columns with title 'nbiwm' contains the number of instances without any misses, while the columns with title 'nbm' show the sum of misses (targets that could not be intercepted) over all 21 instances.

– instance parameter –			——— replan ———		——— ignore ———	
nbt	**nbs**	**dl**	**nbiwm**	**nbm**	**nbiwm**	**nbm**
8	2	D16	10	16	9	17
10	2	D16	5	26	4	30
12	2	D16	5	29	4	33
14	2	D16	1	38	0	47
16	2	D16	1	61	1	74
18	2	D16	0	85	0	108
20	2	D16	0	93	0	113
8	4	D16	18	3	18	3
10	4	D16	19	2	19	2
12	4	D16	18	3	18	3
14	4	D16	16	5	16	5
16	4	D16	16	6	16	6

continued on next page . . .

nbt	nbs	dl	nbiwm	nbm	nbiwm	nbm
18	4	D16	15	7	15	7
20	4	D16	13	9	13	9

IMPRESSUM

Brandenburgische Technische Universität Cottbus-Senftenberg
Fakultät 1 | MINT - Mathematik, Informatik, Physik, Elektro- und Informationstechnik
Institut für Mathematik
Platz der Deutschen Einheit 1
D-03046 Cottbus

Professur für Ingenieurmathematik und Numerik der Optimierung
Professor Dr. rer. nat. Armin Fügenschuh

E fuegenschuh@b-tu.de
T +49 (0)355 69 3127
F +49 (0)355 69 2307

Cottbus Mathematical Preprints (COMP), ISSN (Print) 2627-4019
Cottbus Mathematical Preprints (COMP), ISSN (Online) 2627-6100

www.b-tu.de/cottbus-mathematical-preprints
cottbus-mathematical-preprints@b-tu.de
doi.org/10.26127/btuopen-6110