Stieber, Anke; Fügenschuh, Armin

Article — Published Version

Dealing with time in the multiple traveling salespersons problem with moving targets

Central European Journal of Operations Research

Provided in Cooperation with:
Springer Nature

Suggested Citation: Stieber, Anke; Fügenschuh, Armin (2020) : Dealing with time in the multiple traveling salespersons problem with moving targets, Central European Journal of Operations Research, ISSN 1613-9178, Springer, Berlin, Heidelberg, Vol. 30, Iss. 3, pp. 991-1017, https://doi.org/10.1007/s10100-020-00712-7

This Version is available at:
https://hdl.handle.net/10419/288402

Central European Journal of Operations Research (2022) 30:991–1017
https://doi.org/10.1007/s10100-020-00712-7

Dealing with time in the multiple traveling salespersons problem with moving targets

Anke Stieber[1] · **Armin Fügenschuh**[2]

Accepted: 6 October 2020 / Published online: 16 October 2020

Abstract
The multiple traveling salespersons problem with moving targets is a generalization of the classical traveling salespersons problem, where the targets (nodes or objects) are moving over time. Additionally, for each target a visibility time window is given. The task is to find routes for several salespersons so that each target is reached exactly once within its visibility time window and the sum of all traveled distances of all salespersons is minimal. We present different modeling formulations for this TSP variant. The time requirements are modeled differently in each approach. Our goal is to examine what formulation is most suitable in terms of runtime to solve the multiple traveling salespersons problem with moving targets with exact methods. Computational experiments are carried out on randomly generated test instances to compare the different modeling approaches. The results for large-scale instances show, that the best way to model time requirements is to directly insert them into a formulation with discrete time steps.

Keywords Dynamic traveling salespersons problem · Moving targets · Time-relaxation · Integer linear programming · Second-order cone programming

1 Introduction

This research deals with a dynamic variant of the traveling salesperson problem (TSP), where the targets or nodes are not fixed. This TSP generalization is called multiple traveling salespersons problem with moving targets (MTSPMT). A possible application of the MTSPMT can be found in the defense sector. An area, e.g., an airport or

✉ Anke Stieber
anke.stieber@hsu-hh.de

Armin Fügenschuh
fuegenschuh@b-tu.de

[1] Helmut Schmidt University, Hamburg, Germany

[2] Brandenburg University of Technology Cottbus-Senftenberg, Cottbus, Germany

a military base, must be protected from incoming hostile rocket, artillery, or mortar (RAM) fire. Simultaneous attacks from different firing positions are considered. There is a radar system detecting targets and estimating their trajectories. With a battery of laser guns deployed within or nearby the protected area decisions have to be taken, which available laser gun to select for the countermeasures to protect the area. The selected laser gun then aims at the incoming target for a certain period of time to destroy it. We assume that each target can be destroyed by firing the same period of time regardless how far away it is. This period of time is set to one unit. Generally, the closest laser gun is assigned to a target, where "close" does not refer to the physical distance between laser gun and target, but to the angle the laser gun needs to traverse for aiming at the target. The closest laser gun is used to prefer small angles over large angles, because in case of failure (target is not destroyed) an adjustment of the laser gun is then more likely to succeed. The goal is to destroy all incoming RAM. Here, a laser gun corresponds to a salesperson and the incoming firing entities are moving targets. Each target has got a visibility time window and a salesman can only intercept a target within its respective time window. This window starts at that moment at which the target is radar-detected, its trajectory is computed and reachability by all laser guns is guaranteed. The visibility time window ends at the latest point in time where a destruction of the target is possible (before impact).

Targets move continuously on their trajectories with a certain speed value. The number of laser guns is not fixed to one, thus, we are dealing with multiple salespersons. As for the classical TSP each target must be visited once by exactly one salesperson. The objective is to minimize all traversed distances of all laserguns. Usually, the optimization goal for TSP is minimizing travel times, however in the described application the area is protected by laserguns, which are centered in the middle or equally spread over the area. The concept of catching a target as early as possible could require a lasergun to traverse a long way to intercept that target as early as possible and the lasergun has to traverse the same way back. To save this traveling time and be ready for other targets we minimize the distances. Moreover, it is important to safely destroy a target than to adjust the laser gun for a second try in case of a miss. Destruction of a target is more likely conducted when the laser traverses small angels. In our test instances we assume a 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 a 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 taken immediately. It can be solved "offline" and new data is then integrated into the offline algorithm at runtime by a moving horizon approach. For a detailed description of the application we refer to Stieber et al. (2014).

Obviously, if we restrict the number of salesman to one, fix the position of each target to a certain point in space and extend all visibility windows to the whole considered time horizon, we obtain the classical TSP, which is NP-hard, see Garey and Johnson (1979). Thus, the MTSPMT as a generalization of the classical TSP is NP-hard, too. For a survey on the TSP we refer to Lawler et al. (1985) or Reinelt (1994). Helvig et al. (1998) addressed the Moving-Target TSP, which is the MTSPMT restricted to one salesperson. Possible applications for the Moving-Target TSP are a supply ship, that resupplies patrolling boats or an airplane that must intercept a number of mobile

ground units. The authors 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.

The MTSPMT is similar to the Vehicle Routing Problem (VRP) with time-dependent traveling times. However, in the MTSPMT, there is no capacity restriction imposed and due to the moving targets, we have changing traveling times and distances between any pair of targets. This means 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 two 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) as well as minimizing the traveled distance opposed to minimizing the traveled time.

In this research paper we investigate the time aspect of the MTSPMT in modeling. Based on different ways of integrating time into the model formulation, four different variants are presented and compared regarding performance. Two model approaches have already been published by Stieber et al. (2014), the time-discrete model TD and the time-continuous model TC. The latter one concerning continuous times, is the most precise model variant. Having a time discretization the objective function value of TD cannot be less than the objective of TC. However, the price for the preciseness of continuous time modeling usually is a higher computational burden. In addition, we generate time-free models. For these models we relax the time requirements completely and use subprograms to check and create time feasibility. These feasibility checkers are either based on discrete time steps or on a continuous time formulation. The two resulting models are called *time-free model with time-discrete feasibility checking* (TFTD) and *time-free model with time-continuous feasibility checking* (TFTC). The computational complexity for feasibility checking in subproblems is less complex than for TD respectively TC. The idea behind the time-free variants is to investigate if the computational complexity for TFTD and TFTC can be reduced compared to the models "with" time. In the discrete case we assume, that this could be true for a sufficient fine discretization.

The contributions of this research paper are in detail:

1. The TC model from Stieber et al. (2014) is tightened, such that the distance coefficients are only dependent on the arrival times of the salespersons (departure time is equal to the former arrival time).
2. We reformulate the TC model as a second-order cone program (SOCP), which can be solved by any standard MILP solver. This procedure requires some assumptions on the target movements.

3. The time-free modeling approach is proposed, which is a two-stage variant. In the first stage time is completely relaxed. A time-feasible solution is computed in a second stage by sub-problems integrating not all but necessary time restrictions. In case of a discrete feasibility checker we obtain the TFTD model and in the continuous case we have the TFTC model, respectively.
4. The optimal solutions of the two-stage models are computed using a branch-and-bound framework. We present an algorithm, that uses a callback function (an advanced feature of CPLEX).
5. Computational experiments are carried out using randomly generated and feasible problem instances with a varying number of targets, salespersons, and time steps. Usually, trajectories can have arbitrary shapes, however since the continuous approaches can only handle linear trajectories to be solved by the standard mixed-integer linear programming (MILP) solver CPLEX, we restrict our instances to straight lines.

The aim of this research is to find the best formulation to solve MTSPMT instances with regard to performance (in terms of CPU time). The remainder of this article is organized as follows. In Sect. 2 we provide a survey of the relevant literature. The two model variants TD and TC, which incorporate the time as a discrete and a continuous concept are recalled and extended in Sect. 3. The time-free approach (TFTD and TFTC) and its solution method are addressed in Sects. 4 and 5. The computational experiments are presented in Sect. 6 and we draw conclusions in Sect. 7.

2 Related work

This research article deals with a generalization of the classical TSP by considering multiple salespersons and moving targets with time windows. For a survey on the classical TSP we refer to Reinelt (1994). 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 (Englot et al. 2013; Helvig et al. 1998, 2003; Jiang et al. 2005; Jindal et al. 2011). The first articles addressing the traveling salesperson problem with moving targets (TSPMT) are Helvig et al. (1998, 2003). The authors present an exact and an approximation algorithm for the one-dimensional case, where targets and salespersons move on a straight line. Other specific variants of the TSPMT with restrictions on target speed, movement, and number of targets are also addressed, e.g., the targets move towards the origin (starting point of the salesperson) or never reach the origin. 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. (2005), Jindal et al. (2011) and Englot et al. (2013). Solution approaches are mainly heuristics such as genetic algorithms.

Asahiro (2006) investigated the ball collecting problem (BCP), where n balls move in the Euclidean plane and k robots have to collect them. Each ball starts at a certain location and moves with constant speed in a certain direction. The robots start from the depot and move on straight track-lines, which go through the depot. The robots able

to move in both directions have to collect all balls. Different scenarios with respect to computational complexity are studied such as: Can k robots collect all n balls?

Time-dependent traveling salesperson problems (TDTSP) considered in Abeledo et al. (2013), Ahrens (2015), Albiach et al. (2008), Picard and Queyranne (1978) and time-dependent vehicle routing problems (TDVRP) addressed in Fleischmann et al. (2004), Haghani and Jung (2005), Ichoua et al. (2003), Jung and Haghani (2001), Malandraki and Daskin (1992) are very similar to the MTSPMT. The main difference to the TDTSP and TDVRP is, that the travel times in the MTSPMT vary continuously in the whole considered network instead of usually in a part of the network (e.g., congestion) or at a certain period of time (rush hour traffic). Furthermore, for the MTSPMT not only travel times change over time, but also Euclidean distances between any two targets. This is due to the continuous movement of the targets. Finally, VRP deal with restrictions on capacities of the vehicles, which is not the case for the MTSPMT. For a comprehensive literature overview of TDTSP and TDVRP we refer to Gendreau et al. (2015). In the following we provide some articles in more detail.

Ahrens (2015) addresses a stochastic variant of the dynamic traveling salesperson problem. Problem instances were generated from static TSP datasets, where the movement of each target is modeled by a Gaussian-distributed random distance vector that is added to its location in the 2-d space 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 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".

There is another TSP variant, the equality generalized TSP (E-GTSP), which is very similar to the MTSPMT. Assuming discrete time steps for the MTSPMT, we obtain a copy of each target for each time step in the respective time window. All copies of a target are called cluster. To this end, we have a set of clusters and the task is to visit each cluster exactly once. Arcs between target copies of two different clusters are realized, when a salesperson starting from the first target copy (at the respective time step) is able to reach the second target copy with at most maximum speed. Obviously, there are no anti-parallel arcs. In this context an instance of the TSPMT (MTSPMT with one salesperson) can be formulated as an instance of the asymmetrical E-GTSP. With the restriction that at least one element per cluster has to be visited we obtain the generalized TSP (GTSP). This problem is also known as set TSP or group TSP. The asymmetrical GTSP and E-GTSP have been investigated in Laporte et al. (1987). Noon and Bean (1993) showed, that any problem that can be modeled as a GTSP can be transformed into an asymmetrical TSP. The equivalent problem with more than one salesperson is the generalized vehicle routing problem (GVRP), see Ghiani and Improta (2000). For the symmetrical case there is a recent contribution by Sundar and Rathinam (2016). The authors addressed the generalized multiple depot TSP (GMDTSP) and provide a polyhedral study for this problem class. Presenting a branch-and-bound approach that was also realized by callbacks of 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).

Picard and Queyranne (1978) address the TDTSP. They consider a complete graph and travel times depending on the position of the target in the tour. Thus, the number of targets is equal to the number of discrete time steps and the number of travel time values is also discrete. The authors use shortest paths in a multipartite network for the solution with branch-and-bound and relaxation methods. Abeledo et al. (2013) is based on the formulation given by Picard and Queyranne (1978) and provides a study of the TDTSP polytope and computational experiments with their branch-and-cut-and-price algorithm.

In the case of multiple salespersons as well as capacity restrictions and time windows the TDVRP is concerned. Due to the computational complexity of TDVRP scientific contributions mainly focus on heuristic approaches (Fleischmann et al. 2004; Haghani and Jung 2005; Ichoua et al. 2003; Jung and Haghani 2001; Mancini 2014). Very few literature is published concerning exact methods, some of them are e.g., Albiach et al. (2008) and Soler et al. (2009). Based on the work of Noon and Bean (1993) 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. (2009) 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 generalized TSP or VRP respectively, and then into the ATSP and the AVRP. Albiach et al. (2008) 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.

Haghani and Jung (2005); Jung and Haghani (2001) report on heuristic and exact solution methods for small and mid-sized instances and for bigger instances a lower-bound procedure was used. Here, instances with 5 to 30 nodes and 10, 15 and 30 times steps were considered. Woensel et al. (2007) introduced a new approach to model potential traffic congestion. This model is based on a queuing approach to traffic flows. Computational experiments were carried out for small instances with 10 nodes using explicit enumeration and for up to 100 nodes with an ant colony heuristic.

Another similar problem is the VRP with roaming delivery locations (VRPRDL), see Reyes et al. (2017). This article focuses on a delivery to a customers car (trunk), which can be at different locations to different non-overlapping time intervals. For the MTSPMT that would mean, that the trajectories are cut in non-overlapping pieces and only certain pieces are considered.

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. Since solving time-dependent problems is very time-consuming, exact solution methods can handle only small and mid-sized instances. The challenge of the MTSPMT is the continuous movement of all targets, resulting in varying travel times and distances. In our research we confront different modeling approaches and examine their computation times on randomly generated test instances. Here, we concentrate on an offline approach for solving MTSPMT instances to global optimality. In real-world online applications this can be used as a subroutine within a moving horizon approach.

3 Mathematical models with time

Some basic notation is introduced 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. All salespersons start their tour at the same depot location o. Let $\mathcal{V} := \{1, \ldots, n\}$ be a set of nodes (targets or 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 nodes are moving. Thus, the distance for salesperson k traveling from node i to node 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

$$c_{i,j,k}(s,t) := \infty \quad \text{if and only if } s \notin [\underline{t}_i, \overline{t}_i] \text{ or } t \notin [\underline{t}_j, \overline{t}_j] \text{ or } (t-s)\overline{v} < \left\| p^j(t) - p^i(s) \right\|_2, \tag{1}$$

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. The arrival time of any salesperson at a target is equal to his departure time at the same target, because waiting times are included in the traveling times. The depot o has the entire time horizon as visibility window. 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

The time-discrete MILP formulation has already been addressed in Stieber et al. (2014). For the convenience of the reader it is concisely presented in the sequel. The model consists of a multi-commodity flow formulation embedded in a time expanded network. For an explanation of time expanded networks see Ford and Fulkerson (1958). 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. Let m be an integer number. The step size is defined by $\Delta := T/m$. Then the set of all time steps is $\mathcal{T} := \{0, \ldots, m\}$. With $\tilde{\mathcal{A}}$ we denote the set of arcs in the time-expanded network. An arc is only considered between different layers: the arc $(i, p, j, q) \in \tilde{\mathcal{A}}$, $(i, j) \in \mathcal{A}$, $p, q \in \mathcal{T}$ connects target i in layer p with target j in layer q. That means a salesperson departs in i at time step p and arrives in j at time step q. For each salesperson the arrival time at any node in $\mathcal{V}$ is equal to the departure time in the same node, since the waiting time is included in the traveling time. Having discrete time steps $p, q \in \mathcal{T}$ we are able to evaluate the distance function c for arcs at these times

$$c^{p,q}_{i,j,k} := c_{i,j,k}(p\Delta, q\Delta).$$

We introduce a family of binary decision variables $x^{p,q}_{i,j,k} \in \{0,1\}$. Here, $x^{p,q}_{i,j,k} = 1$ represents the decision of sending salesperson k from i to j, departing at time step p in i and arriving in j at time step q. The objective function is to minimize the total traveled distances of all salespersons:

$$\sum_{k\in\mathcal{W}} \sum_{(i,p,j,q)\in\tilde{\mathcal{A}}} c^{p,q}_{i,j,k}\, x^{p,q}_{i,j,k} \quad \to \quad \min\,. \tag{2}$$

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

$$\sum_{k\in\mathcal{W}} \sum_{(i,p,q):(i,p,j,q)\in\tilde{\mathcal{A}}} x^{p,q}_{i,j,k} = 1, \quad \forall\, j \in \mathcal{V}. \tag{3}$$

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

$$\sum_{(j,p,q):(o,p,j,q)\in\tilde{\mathcal{A}}} x^{p,q}_{o,j,k} \le 1, \quad \forall\, k \in \mathcal{W}. \tag{4}$$

The following flow constraints ensure the feasibility of time. Conservation is ensured at each node of the salesperson tour except for the last one, this can be regarded as the sink of the flow:

$$\sum_{(i,p):(i,p,j,q)\in\tilde{\mathcal{A}}} x^{p,q}_{i,j,k} \ge \sum_{(i,p):(j,q,i,p)\in\tilde{\mathcal{A}}} x^{q,p}_{j,i,k}, \quad \forall\, j \in \mathcal{V}, q \in \mathcal{T}, k \in \mathcal{W}. \tag{5}$$

Summing up, we solve the following optimization problem:

$$\min\{(2) \mid (3), (4), (5), x \in \{0,1\}^{\tilde{\mathcal{A}}\times\mathcal{W}}\}. \tag{6}$$

The restrictions of the visibility time windows are embedded in the arcs, thus, arcs violating an involved time window constraint have infinite length. Usually TSP have to incorporate subtour elimination constraints. In fact, this point 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 (6) is not restricted to special shapes of the target trajectories. It can handle any trajectory, see for example Stieber and Fügenschuh (2018). Likewise, there is no need to restrict the speed function of the targets, the model is able to deal with varying target speeds. Obviously, an adverse effect of the above formulation is the increased problem size in case of a higher number of time steps (better accuracy), leading to a higher computational burden.

3.2 A time-continuous model

The salespersons may intercept the targets at any point of their trajectories, thus, the time constraints have to be modelled by continuous variables. As opposed to the discrete model, the decision whether a salesperson is able to reach the destination target of a direct link with its maximum speed has to be integrated into the model formulation. This is realized by applying the big-M method (similar to the Miller et al. (1960) (MTZ) constraints for the TSP) containing continuous time variables and time restrictions. An obvious consequence of this approach is the fact, that we obtain an objective function value 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). Additionally, continuous time variables $t_{i,k} \in \mathbb{R}$ are defined to describe the arrival time of salesperson k in node i. Here, the set of arcs is $\mathcal{A}$ with the arc length defined by (1). Now, we are able to formulate the continuous model.

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

$$\sum_{k\in\mathcal{W}} \sum_{(i,j)\in\mathcal{A}} c_{i,j,k}(t_{i,k}, t_{j,k})\, x_{i,j,k} \quad \rightarrow \quad \min\,. \tag{7}$$

Each node j must be visited once by exactly one salesperson:

$$\sum_{k\in\mathcal{W}} \sum_{i:(i,j)\in\mathcal{A}} x_{i,j,k} = 1, \quad \forall\, j \in \mathcal{V}. \tag{8}$$

Each salesperson k can only start once from the depot:

$$\sum_{j\in\mathcal{V}} x_{o,j,k} \leq 1, \quad \forall\, k \in \mathcal{W}. \tag{9}$$

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

$$\sum_{i:(i,j)\in\mathcal{A}} x_{i,j,k} \geq \sum_{i:(j,i)\in\mathcal{A}} x_{j,i,k}, \quad \forall\, j \in \mathcal{V}, k \in \mathcal{W}. \tag{10}$$

The following MTZ-type 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

$$t_{i,k} + \frac{c_{i,j,k}(t_{i,k}, t_{j,k})}{\overline{v}} \leq t_{j,k} + T \cdot \left(1 - x_{i,j,k}\right), \quad \forall\, (i,\, j) \in \mathcal{A}, k \in \mathcal{W}. \tag{11}$$

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

$$\underline{t}_j \le t_{j,k} \le \overline{t}_j, \quad \forall\, j \in \mathcal{V}_o, k \in \mathcal{W}. \tag{12}$$

Summarized, we aim to solve the following optimization problem:

$$\min\{(7) \mid (8), (9), (10), (11), (12), x \in \{0,1\}^{\mathcal{A}\times\mathcal{W}}, t \in \mathbb{R}^{\mathcal{V}_o\times\mathcal{W}}\}. \tag{13}$$

Basically, this model is the same as the time continuous model in Stieber et al. (2014). The only changes made are some modeling refinements, e.g., departure and arrival time variables are replaced by arrival variables because possible waiting is included, constraints (9) are only considered for depot node o and bounds for the visibility time windows are added.

The presented time-continuous formulation of the MTSPMT is based on an arbitrary nonlinear continuous function $c_{i,j,k}$ for the distance between two distinct moving nodes. In order 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 for all targets. Then $c_{i,j,k}$ represents the Euclidean distance between two points on two straight lines. With this, the above presented optimization problem (13) can be handled as a second order cone program (SOCP), see Boyd and Vandenberghe (2004) for an overview. The constraints (11) 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 Euclidean distance $c_{i,j,k}(t_{i,k}, t_{j,k})$; $a_{i,j,k}$ for the sum

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

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

$$\sum_{k\in\mathcal{W}} \sum_{(i,j)\in\mathcal{A}} a_{i,j,k} \quad \to \quad \min. \tag{14}$$

The constraints (8), (9), (10) and (12) remain unchanged. The constraints (11), 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 Eqs. (15)–(18). We define $\Delta x_i = \overline{x}_i - \underline{x}_i$, $\Delta y_i = \overline{y}_i - \underline{y}_i$ and $\Delta t_i = \overline{t}_i - \underline{t}_i$.

$$c^x_{i,j,k} - \left(\left(\underline{x}_j + t_{j,k}\frac{\Delta x_j}{\Delta t_j} - \underline{t}_j\frac{\Delta x_j}{\Delta t_j}\right) - \left(\underline{x}_i + t_{i,k}\frac{\Delta x_i}{\Delta t_i} - \underline{t}_i\frac{\Delta x_i}{\Delta t_i}\right)\right) = 0, \tag{15}$$
$$\forall\,(i,j) \in \mathcal{A}, i \ne o, k \in \mathcal{W}.$$

$$c^x_{o,j,k} - \left(\left(\underline{x}_j + t_{j,k} \frac{\Delta x_j}{\Delta t_j} - \underline{t}_j \frac{\Delta x_j}{\Delta t_j} \right) - o^x \right) = 0, \quad \forall\, j \in \mathcal{V}, k \in \mathcal{W}. \tag{16}$$

$$c^y_{i,j,k} - \left(\left(\underline{y}_j + t_{j,k} \frac{\Delta y_j}{\Delta t_j} - \underline{t}_j \frac{\Delta y_j}{\Delta t_j} \right) - \left(\underline{y}_i + t_{i,k} \frac{\Delta y_i}{\Delta t_i} - \underline{t}_i \frac{\Delta y_i}{\Delta t_i} \right) \right) = 0, \quad \forall\, (i,j) \in \mathcal{A}, i \neq o, k \in \mathcal{W}. \tag{17}$$

$$c^y_{o,j,k} - \left(\left(\underline{y}_j + t_{j,k} \frac{\Delta y_j}{\Delta t_j} - \underline{t}_j \frac{\Delta y_j}{\Delta t_j} \right) - o^y \right) = 0, \quad \forall\, j \in \mathcal{V}, k \in \mathcal{W}. \tag{18}$$

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

$$a_{i,j,k} \leq \overline{v} \left(t_{j,k} - t_{i,k} + T\,(1 - x_{i,j,k}) \right), \quad \forall\, (i,j) \in \mathcal{A}, k \in \mathcal{W}. \tag{19}$$

The next conditions are needed to formulate the cone constraints:

$$\overline{a}_{i,j,k} = a_{i,j,k} + R\,(1 - x_{i,j,k}), \quad \forall\, (i,j) \in \mathcal{A}, k \in \mathcal{W}, \tag{20}$$

where $R := \lceil \sqrt{2} S \rceil$ ($\sqrt{2}S$ is the diagonal of the square).
Finally, the cone constraints are given as:

$$(c^x_{i,j,k})^2 + (c^y_{i,j,k})^2 \leq (\overline{a}_{i,j,k})^2, \quad \forall\, (i,j) \in \mathcal{A}, k \in \mathcal{W}. \tag{21}$$

Summarized, the transformed SOCP reads the following:

$$\begin{aligned} &\min\{(14) \mid (8), (9), (10), (12), (15), (16), (17), (18), (19), (20), (21) \\ &\quad x \in \{0,1\}^{\mathcal{A}\times\mathcal{W}}, t \in \mathbb{R}^{\mathcal{V}_o\times\mathcal{W}},\ c^x,\ c^y,\ a,\ \overline{a} \in \mathbb{R}^{\mathcal{A}\times\mathcal{W}}\}. \end{aligned} \tag{22}$$

Having this model formulation, a salesperson is able to intercept a moving target at any possible point on its trajectory. In general, the objective function value of (22) is less than or equal to the objective function value of (6) for the same instance. However, the assumptions we had to make 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 have to be examined in the branch and bound tree, which slows down the solution process Codato and Fischetti (2004). Additionally, due to the big-M constants numerical instabilities can occur in the solution procedure if the constants are not tight enough. In (22) there are two of such constants, see (19) and (20). Obviously, a comparison of both modeling approaches (discrete and continuous) is difficult, because of the different characteristics.

4 Time relaxations

For the next model formulations we concentrate on the time aspect in a different way namely on a time-free model. This means relaxing the time completely and later reintegrating parts of the time restrictions. The technique was first introduced by Fügenschuh et al. (2013). They successfully applied the method to the scheduling and routing of planes and tourist travel requests during fly-in safaris, which essentially is a VRPTW with pickup and delivery.

First of all, we perform a projection of (6) from the time-discrete variable space $\tilde{\mathcal{A}} \times \mathcal{W}$ to $\mathcal{A} \times \mathcal{W}$ (i.e., from $(\mathcal{V}_o \times \mathcal{T}) \times (\mathcal{V} \times \mathcal{T}) \times \mathcal{W}$ to $\mathcal{V}_0 \times \mathcal{V} \times \mathcal{W}$). The time-free counterpart of 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 arc, we have to find a suitable length for this new arc to preserve the optimal solution. Thus, the minimum length of all time-discrete arcs between two targets is used as the length for the new arc. That means, for any salesperson k and any two nodes i and j the distance coefficient $c_{i,j,k}$ is taken as

$$c_{i,j,k} = \min\{c^{p,q}_{i,j,k} \mid p, q \in \mathcal{T}\}. \tag{23}$$

With this, we have the following model in the time-free space:

$$\begin{aligned}
\sum_{k\in\mathcal{W}} \sum_{(i,j)\in\mathcal{A}} c_{i,j,k}\, x_{i,j,k} \quad &\to \quad \min \\
\text{s.t.} \sum_{k\in\mathcal{W}} \sum_{i:(i,j)\in\mathcal{A}} x_{i,j,k} \;&=\; 1, \quad \forall\, j \in \mathcal{V} \\
\sum_{j:(o,j)\in\mathcal{A}} x_{o,j,k} \;&\le\; 1, \quad \forall\, k \in \mathcal{W} \\
\sum_{i:(i,j)\in\mathcal{A}} x_{i,j,k} - \sum_{i:(j,i)\in\mathcal{A}} x_{j,i,k} \;&\ge\; 0, \quad \forall\, j \in \mathcal{V}, k \in \mathcal{W} \\
x \;&\in\; \{0,1\}^{\mathcal{A}\times\mathcal{W}}.
\end{aligned} \tag{24}$$

This is a classical multi-commodity flow problem, which is easy to solve by standard MILP solvers. The optimal solution of the time-free model (24) is a lower bound to (6), because the distance coefficients are computed by minimization. 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 examine every time-free solution, that we encounter during the solution process for time-feasibility. The examination is embedded in a branch-and-bound framework in order to prune nodes whose lower bounds exceed the current best solution value.

Given an optimal solution of (24), feasible times at which salespersons reach the targets have to be constructed from it. 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 (6). However, this rarely happens. It is more likely, that a time-free solution is infeasible with respect to the time constraints. Thus, apart from an optimal time-free solution, we have to

investigate also further feasible time-free solutions in the branch-and-bound process. That means (24) serves as the master problem in the branch-and-bound framework. For any 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 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. Obviously, the generated cuts should take into account all possible salespersons permutations in order to prevent a repetition of the same time-free solution due to salesperson symmetries. In case that 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 Sect. 5. For now, we concentrate on the construction of a time-feasible solution from a time-free solution.

Given a time-free solution, according to the variables set to one, we obtain a set of arcs for each salesperson and call it a pretour. Note, a pretour may be disconnected and the pretour of some salespersons can be empty. A salesperson is called active, if its pretour is not empty. The pretours of all active salespersons are extracted from the solution. A pretour is called feasible if it is a Hamiltonian path in the induced sub-graph starting at the depot o. For each feasible pretour a time-feasible tour is required.

4.1 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 the time restrictions of the salesperson, who belongs to the pretour. In more detail, we include only those time-dependent arcs, that have a time-free counterpart in the given pretour meaning the corresponding solution variable is nonzero. In the case that the checking sub-MILP for each active salesperson results in a time-feasible tour an overall time-feasible tour is found. The best overall time-feasible tour solves (6).

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. For an active salesperson k 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 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, $\mathcal{V}^k = \{o = v_0, v_1, v_2, \ldots, v_{n_k}, v_{n_k+1} = d\}$ denotes the sequence of nodes considered for the minimum-cost flow problem of salesperson k.

Since we have to consider only those time-expanded arcs, that correspond to an arc of the pretour, the checking MILP consists of all arcs, that go from depot position o to distinct positions of v_1 and from distinct positions of v_1 to distinct positions of v_2 and so on. Additionally, we have to introduce artificial time-discrete arcs from distinct positions of v_{n_k} to the sink d. That means, for each arc, that ends in v_{n_k} at time step p an arc is introduced from (v_{n_k}, p) to $(d, p + 1)$. The distance of all arcs between

v_{n_k} and d is zero. We denote this set of arcs for salesperson k by $\mathcal{A}^k$. According to the time-discrete model, we introduce binary decision variables $x^{p,q}_{v_i,v_{i+1}}$ describing the decision of sending salesperson k from target v_i to its successor v_{i+1} starting at time step p and arriving at time step q. Then, the time-feasibility checking MILP for an active salesperson k is formulated as follows:

$$\begin{aligned}
\sum_{(v_i,p,v_{i+1},q)\in\mathcal{A}^k} c^{p,q}_{v_i,v_{i+1}}\, x^{p,q}_{v_i,v_{i+1}} \quad &\to \quad \min \\
\text{s.t.} \sum_{(p,q):(o,p,v_1,q)\in\mathcal{A}^k} x^{p,q}_{o,v_1} \quad &= \quad 1 \\
\sum_{(p,q):(v_{n_k},p,d,q)\in\mathcal{A}^k} x^{p,q}_{v_{n_k},d} \quad &= \quad 1 \\
\sum_{p:(v_{j-1},p,v_j,q)\in\mathcal{A}^k} x^{p,q}_{v_{j-1},v_j} - \sum_{p:(v_j,q,v_{j+1},p)\in\mathcal{A}^k} x^{q,p}_{v_j,v_{j+1}} \quad &= \quad 0,\ \forall j \in \{1,\dots,n_k\}, q \in \mathcal{T} \\
x \quad &\in \quad \{0,1\}^{\mathcal{A}^k}. \qquad (25)
\end{aligned}$$

The optimization problem (25) 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 Dijkstra (1959). In case the checking MILP (25) results in a time-feasible tour for each active salesperson, we are able to construct a total time-feasible solution for (6) by combining all salesperson tours. If the checking MILP results in an infeasible solution for any of the active salespersons the construction process is aborted for the corresponding time-free solution.

4.2 A time relaxation with continuous time feasibility checking

According to our time-continuous model (13), there is also a time-relaxed variant with a continuous time-feasibility checking sub-MILP. For this variant we cannot use (24) directly, since its distance coefficients $c_{i,j,k}$ in the objective function are dependent on a discretization and on the time-discrete arcs, see (23). In a continuous model, these coefficients are not valid. Here, we have to replace the minimum time-expanded arc length by the real minimum length between any two trajectories. Thus, the distance coefficients $c_{i,j,k}$ are computed as follows: For an endpoint p^j_e of trajectory j we seek for the time interval $[t^i_1, t^i_2]$ for i, such that

$$t^i_1 = \inf\{t \in [\underline{t}_i, \overline{t}_i] \mid k \text{ can travel from } p^i(t) \text{ to } p^j_e \text{ with speed } \overline{v}\}$$

and

$$t^i_2 = \sup\{t \in [\underline{t}_i, \overline{t}_i] \mid k \text{ can travel from } p^i(t) \text{ to } p^j_e \text{ with speed } \overline{v}\},$$

where $p^i(t)$ is the position of i at time t. Let I be the union of the intervals for both endpoints of j and $J = [\underline{t}_j, \overline{t}_j]$. Then we compute the overall minimum distance

between the trajectory of i reduced to the interval I and trajectory of j:

$$c_{i,j,k} = \min \{c_{i,j,k}(s,t) \mid s \in I,\ t \in J\}. \tag{26}$$

Then the continuous time-relaxed model is (24) with (26). Having this, we can formulate the feasibility checking sub-MILP for the continuous case. As for the continuous model (22) with the assumption of linear trajectories and constant target speed, the checking sub-MILP for an active salesperson k can be modeled as a quadratic program. Here, we again use continuous time variables t_{v_i} to define the arrival time in v_i. With the target sequence $\{o = v_0, v_1, \ldots, v_{n_k}\}$ and the corresponding set of arcs $\mathcal{A}^k$, we obtain the checking MILP for k as follows:

$$\begin{aligned}
&\sum_{i=0}^{n_k-1} c_{v_i,v_{i+1}}(t_{v_i}, t_{v_{i+1}})\, x_{v_i,v_{i+1}} \rightarrow \min \\
&\qquad \text{s.t.} \quad c_{v_i,v_{i+1}}(t_{v_i}, t_{v_{i+1}}) \leq \overline{v}\,(t_{v_{i+1}} - t_{v_i}), \quad \forall\, i = 0, 1, \ldots, n_k - 1, \\
&\qquad\qquad t_{v_i} \in [\underline{t}_i, \overline{t}_i], \quad \forall\, i = 1, \ldots, n_k \\
&\qquad\qquad x \in \{0,1\}^{\mathcal{A}^k}.
\end{aligned} \tag{27}$$

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 $t_{v_{i+1}}$. 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 needs to traverse an arc with length $c_{v_i,v_{i+1}}(t_{v_i}, t_{v_{i+1}})$ in a time difference of $(t_{v_{i+1}} - t_{v_i})$, has to be at most $\overline{v}$, the maximum speed. In contrast to the proposed TC model (13), the time-free model with the continuous checking MILP (TFTC) does not contain any big-M constant.

5 Implementational details

In this section we address the time-free problem (24). In case of a discretization of time, (23) is needed for the calculation of the arc lengths. In the continuous case we use (26) instead. The model (24) (with either (23) or (26)) 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 the best time-feasible solution we use the callback utilities of CPLEX. We implement an instance of the BranchCallback and 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 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.

In the LazyConstraintCallback we include a validation check of the feasible solution of the master problem and the construction of feasible times according to (25) and (27). Our callback algorithm is presented in Algorithm 1. A more detailed description of individual steps is given in the sequel.

Algorithm 1: LazyConstraintCallback: Pretour validation check and time-feasibility construction.

Data: Time-relaxed x variables, best objective function value found so far $best\_obj\_val$
Result: If exists 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      for all salespersons add a global cut;
7      return;
8  #Validation check by interval propagation
9  if the pretour of an active salesperson s is identified as time-infeasible then
10     for all salespersons add a global cut;
11     return;
12 #Time-Feasibility Check
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 line 2–3 the objective function value of the master problem is compared to the best objective function value found so far. This value serves as a lower bound. If the current value cannot beat the best time-feasible objective function value found so far, the callback will quit. Afterwards, the current node is pruned in the BranchCallback, which is subsequently invoked. The task of the BranchCallback is pruning. Branching decisions are left to the default way of CPLEX.

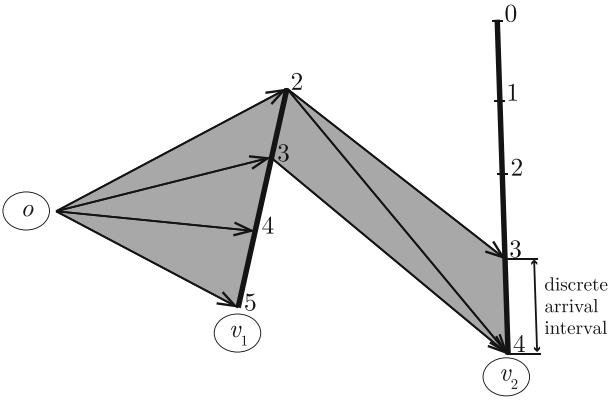

Fig. 1 Interval propagation. This figure presents the depot position o and a given salesperson tour consisting of the sequence o, v_1, v_2. Each target is visualized by its trajectory and the corresponding discrete time steps, which are given by numbers. The grey area describes the discrete *arrival* and *departure interval* between consecutive nodes

Since the master problem is a multi-commodity flow problem, each solution represents a feasible flow but not necessarily a feasible tour, due to the lack of subtour elimination constraints. This leads to line 5–7 in order to check each active pretour against cycles. In case a cycle $\mathcal{C} \subset \mathcal{A}$ is found, it will be cut off by the following constraints:

$$\sum_{(i,j)\in\mathcal{C}} x_{i,j,k} \leq |\mathcal{C}| - 1, \quad \forall k \in \mathcal{W}. \tag{28}$$

The last validation check performed is to exploit the visibility time windows along the pretour and test if there are any feasible times to arrive at the last node of the pretour, see line 9–11. This interval propagation is visualized in Fig. 1. Let us consider any target v of the pretour. Then we investigate if there are time steps in the visibility window of v, which can be used to arrive from its predecessor (starting in its time window) and to leave for its successor (arriving in its time window). The resulting time interval serves as the new visibility window for v and it is propagated to the succeeding target the same way. Since for the depot node o the whole time horizon is considered as visibility window, the whole visibility window for v_1 can be used as its *arrival interval*. Regarding the visibility window of v_2 we obtain a possible *departure interval* for v_1 with feasible time steps to depart for v_2. The next step is to generate a cut set from both intervals of v_1. This procedure is continued to the last node of the pretour. If at the last node or any node before the intersection interval is empty, this indicates that the pretour we started with is not time-feasible.

In the given example the *arrival interval* of node v_1 is the discrete interval $[2, 5]$. The *departure interval* to leave for v_2 with a speed of at most $\overline{v}$ is $[2, 3]$ with a possible arrival in v_2 in $[3, 4]$. A departure after 3 in v_1 cannot reach v_2 within its visibility window of $[0, 4]$. An earlier departure is not possible due to the visibility window of v_1. Thus, the intersection of $[2, 5]$ and $[2, 3]$ leads to $[2, 3]$, which is considered

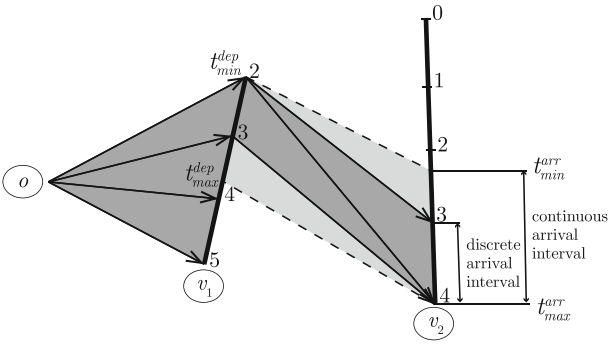

Fig. 2 Interval propagation. This figure presents the depot position o and a given salesperson tour consisting of the sequence o, v_1, v_2. Each target is visualized by its trajectory and the corresponding discrete time steps, which are given by black numbers. The grey area describes the discrete departure and arrival interval between consecutive nodes. The light grey area is the extended departure and arrival interval when considering continuous times

as the new visibility window for v_1. The procedure of time interval propagation has to be continued to the following targets, but it is important to use the updated time window for the predecessor node. In case there is an empty cut set at any target of the pretour, we have an infeasible interval. This means, there is no time-feasible tour for the current pretour and we can reject the current time-free solution. This is done by adding the following global cut to the master problem. For an infeasible pretour $o = v_0$ to v_{n_s}, let $\mathcal{P} \subset \mathcal{A}$ be the corresponding sequence of arcs, then, we formulate the following constraint for each salesperson to cut off this pretour:

$$\sum_{(i,j)\in\mathcal{P}} x_{i,j,k} \leq |\mathcal{P}| - 1, \quad \forall k \in \mathcal{W}. \tag{29}$$

Note, that for any pair of anti-parallel arcs (i, j) and (j, i) we have

$$x_{i,j,k} + x_{j,i,k} \leq 1, \quad \forall k \in \mathcal{W}. \tag{30}$$

This means, for a variable in (29), which is bounded by 1, we can add the variable of the anti-parallel arc at no cost (30). Thus, lifting $|P| - 1$ anti-parallel arcs to the cut (29), leads to:

$$\sum_{(i,j)\in\mathcal{P}} x_{i,j,k} + \sum_{(i,j)\in\mathcal{P}\setminus(v_{n_s-1},v_{n_s})} x_{j,i,k} \leq |\mathcal{P}| - 1, \quad \forall k \in \mathcal{W}. \tag{31}$$

Interval propagation for the continuous case is done in a similar way. In general, the resulting arrival and departure intervals are slightly larger, see Fig. 2.

The travel time is not rounded to the next time step, its exact value is computed using maximum salesperson speed and the Euclidean distance between the corresponding positions of the targets. With this a salesperson is able to arrive earlier and to depart later compared to the case with discrete time steps. The computation of the new time window

of a target is again an intersection of arrival and the departure interval. The exact *arrival interval* depends on the previously updated time interval of the predecessor node and the departure interval depends on the visibility window of the successor. In Fig. 2 the *arrival interval* of v_1 is the whole visibility window, since the predecessor node is the depot. For the *departure interval* of v_1 the values t_{min}^{dep}, t_{max}^{dep}, t_{min}^{arr} and t_{max}^{arr} have to be identified. Obviously, we have $t_{min}^{dep} = \underline{t}_{v_1}$ and $t_{max}^{arr} = \overline{t}_{v_2}$. Assuming a constant maximum speed we take the equation of uniform movement to compute t_{max}^{dep}, which is the latest possible departure in v_1:

$$\overline{v}\,(\overline{t}_{v_2} - t_{max}^{dep}) = \left\| p_{arr} - p_{dep} \right\|_2, \tag{32}$$

where p_{arr} and p_{dep} are the positions of the targets at the times $\overline{t}_{v_2}$ and t_{max}^{dep} respectively. Since the right hand side of the motion equation is depended on t_{max}^{dep} we have to square both sides and replace p_{arr} and p_{dep} by their trajectory parameterization. This leads to a quadratic equation, from which t_{max}^{dep} can be obtained. An intersection of both intervals of v_1, which are [2, 5] and $[t_{min}^{dep}, t_{max}^{dep}]$) gives us the new time interval of feasible time steps. The next step is to calculate the *arrival interval* $[t_{min}^{arr}, t_{max}^{arr}]$ of v_2 from the recently computed time interval of v_1. Here, the missing t_{min}^{arr} is again calculated by the equation of uniform movement, see (32). This interval is the arrival interval of v_2 when we move to the next nodes for interval propagation. The intervals $[t_{min}^{dep}, t_{max}^{dep}]$ and $[t_{min}^{arr}, t_{max}^{arr}]$ have to be computed accordingly for v_2 and v_3. This procedure is continued to the last node of the pretour. If at the last node or any node before an empty intersection interval has been detected, there is no time-feasible salesperson tour and the corresponding pretour is cut off for any salespersons permutation. In the feasible case the MILP (25) or (27) is set up to compute the corresponding times.

Each computed pretour and its counterpart with time are stored in a solution pool in order to prevent setting up and solving the same sub-MILP again and again for pretours occurring more than once. This is realized in Algorithm 1 in the lines 18–19 and 32. The existence of a solution pool forces us to use a single-threaded optimization instead of a multi-threaded one. The reason is, that in parallel mode it is not allowed to access data, which is not local. Another reason against parallel mode is, that CPLEX is not deterministic due to a different order of callback invocations for multiple runs of the same instance with the same parameter setting on the same platform. This would lead to an undesired, non-deterministic variability in the runtimes.

Finally, a time-feasible solution is found, when there is a time-feasible tour for each active salesperson. This solution is returned to the master problem and the best objective function value found so far is saved. Then a global cut is added to the master problem in order to prevent a repetition of the current solution by other salesperson permutations. Summing up, we have the time-discrete model (6) (TD), the time-continuous model (13) (TC), the time-free model (24) with time-discrete feasibility checking (25) according to Algorithm 1 (TFTD) and the time-free model (24) with time-continuous feasibility checking (27) according to Algorithm 1 (TFTC).

6 Computational experiments

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 a mutual comparison is not easy. Consider for example the shape of the trajectories. The time-continuous models are restricted to straight lines to be solvable, while the time-discrete models are not. Another point is the difference in objective function values between discrete and continuous approaches. Obviously the objective function value of the continuous approaches is always less or equal than the objective function value of the discrete models, but with this the computational burden will also be higher. Thus, we perform a runtime comparison of the discrete approaches and one of the continuous approaches. As a basis we concentrate on randomly generated linear target trajectories with constant speed for all our modeling variants.

6.1 Instance generation

We use a set of randomly generated test instances. A test instance is specified by number of salespersons, number of moving targets and distance of time steps. 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. If two hostile RAM meet in the air by chance, they would be deflected or destroy each other before their destinations are reached. Since such a szenario is very unlikely and due to visualization clarity we do not support such a situation in the trajectory generation by prohibiting any pairwise intersections. This is not an assumption to the models, all four variants can cope with trajectory intersections. 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 one time unit. In Stieber et al. (2014) we observed, that instances have a higher complexity, if the difference between target speed and salespersons speed is high. In this case the number of possible tours of the salespersons rises with an increased speed difference. 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 3 different levels of time discretization. In particular, for the first discretization level called D32 a time step is introduced to the trajectories every 32 length units (target speed). 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 4 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.

The current research deals with solvable test instances, that means no target will reach its upper time limit before being visited by a salesperson. 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 instances salespersons start their tours at the depot position o, an initial position located in the center of the

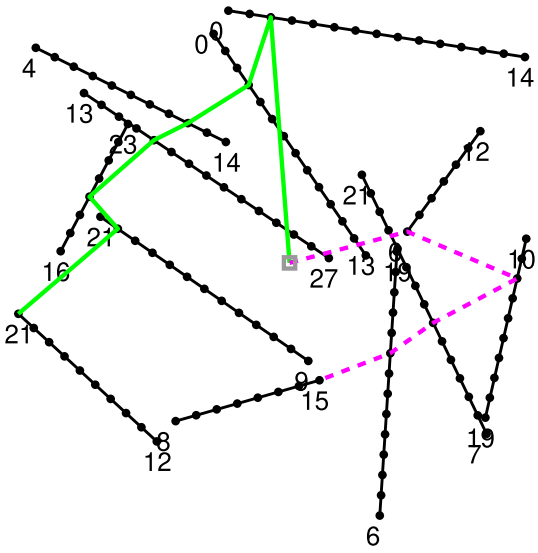

Fig. 3 An instance of the MTSPMT with 12 targets 2 salespersons and medium discretization level (D16)

operating space. Of course all models can easily be adapted to the case of multiple depot positions. In total, instances with number of targets of 6, 8, 10, 12, 14, 16, 18 and 20, with number of salespersons of 1, 2, 3, 4, 5 and 6 and with discretization levels D32, D16 and D8 are created. An instance as an example with 12 targets, 2 salespersons starting from the depot in the middle and medium discretization level D16 is visualized in Fig. 3. The visibility time window is given by the numbers at the end points of the trajectories.

6.2 Computational results

The generated instances are embedded in the 2-d space, nevertheless the proposed models and methods are not restricted to the 2-d space. Furthermore our time-discrete models TD and TFTD are not restricted to linear trajectories, it is also possible to handle non-linear trajectories. The TD model and nonlinear trajectories are addressed in Stieber and Fügenschuh (2018).

All generated instances are solved with CPLEX. While the solution procedure of the time-free models TFTD and TFTC is customized by LazyConstraintCallbacks and BranchCallbacks of CPLEX, the instances modeled with TD and TC are solved without callbacks and directly by CPLEX' MILP and 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

Table 1 CPLEX parameter settings

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

the 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, in this sense parallel optimization may lead to different runtimes. We use sequential optimization mode instead. A time limit of one hour is enforced to our experiments. All other CPLEX parameters are left at their default values.

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 [7]. 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, CPLEX requires to compute the global proven optimal solution, including the time needed for the callbacks. 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 3600 s and the remaining gap, which is at most 1. It is computed as $sc = \frac{runtime}{3600} + gap$. The score takes values between 0 and 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 $(sc-1)\cdot 100$ percent.

In the first experimental set we fix the number of salesperson (3) and the discretization level (D16) and only vary the number of targets between 6 and 20. For each number of targets 21 instances are randomly generated. The scores of all 21 instances and the related arithmetic mean for model TD and TFTD is reported in Fig. 4. In case an instance is solved to proven optimality the score only consists of the runtime part of the score (gap $= 0$) and is located in the lower half of the figure. In the other case the gap is positive and the score value is in the upper half of the figure at the right gap value. Note, that for reasons of better comparability the TD score values are

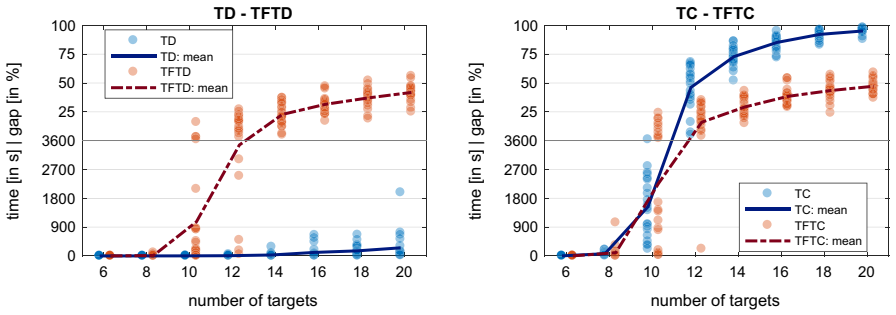

Fig. 4 Visualization of the scores (runtime and gap) depending on the number of targets

shifted to the left (by 0.2) and the TFTD score values are shifted to the right (by 0.2) of the respective target number. Technically, the scores of TD and TFTD belong to one single number of targets and would overlap in this case, which would lead to a worse readability and comparability. The same is done for TC and TFTC and in the following visualizations.

For the TD model most of the instances can be solved within 900 s, there is one instance, that needs 1980 s. The TFTD model is only fast for small instances with 6 and 8 targets. The score values of TD and TFTD are close to each other in this target range. Here are even 6 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 3600 s. 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).

The next experimental set fixes the number of targets (10) and the discretization level (D16). We vary the number of salespersons from 1 to 6. The respective score values of all four models are reported in Fig. 5. The scores of TD are low and close to each other regardless of the number of salespersons. 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 is again smaller for TFTC than for TC regarding 4 salespersons and more.

In the third experimental set the number of targets (10) and the number of salespersons (3) is fixed, while the discretization level varies from D32 (coarse) via D16 (medium) to D8 (fine). Here, only discrete models are considered, because the other ones are based on continuous time variables. The scores for TD and TFTD are reported in the left graphic of Fig. 6, the right graphic is a zoomed visualization of the runtimes for TD. One can see, that the runtimes for TD increases with a higher number of time steps. Unlike TFTC, where the mean is nearly the same for all three discretization levels. For D8 there are three out of 21 instances, that can be solved faster with TFTD than with TD.

As the last experiments we want to investigate the runtime of the proposed models for a time limit of 3 s. This is important, since from the mathematical point of view

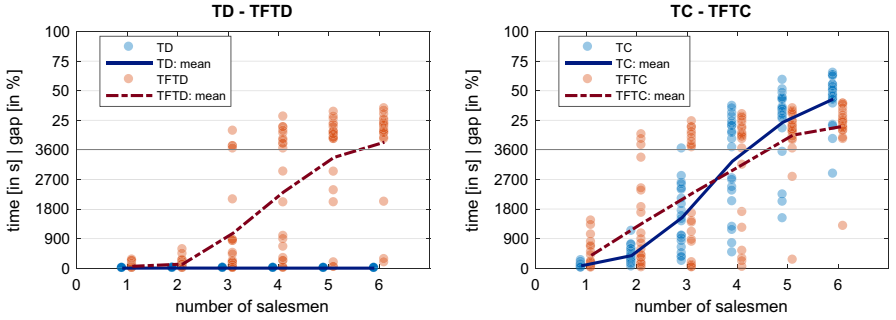

Fig. 5 Visualization of the scores (runtime and gap) depending on the number of salespersons

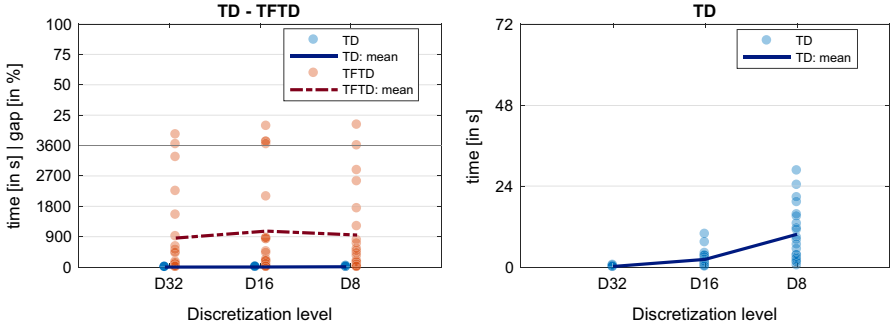

Fig. 6 Visualization of the scores (runtime and gap) depending on the discretization level

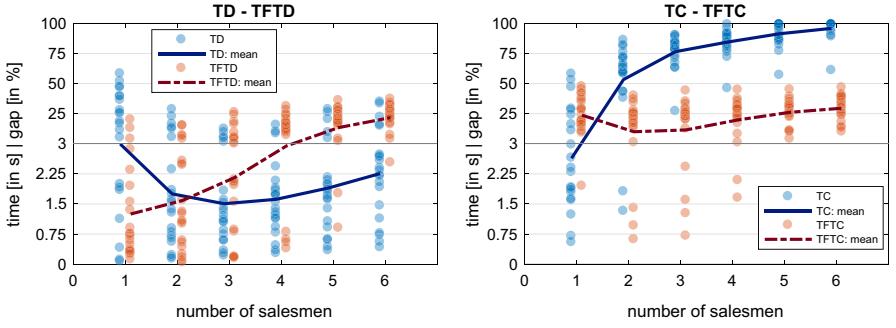

Fig. 7 Visualization of the scores (runtime and gap) depending on the number of salespersons with a time limit of 3 s

the application is an online problem. Here, we set the number of targets to 8 and the discretization level to D8 and vary the number of salespersons from 1 to 6. The results are reported in Fig. 7. For the discrete models we observe, that for one and two salespersons the mean of TFTD is better (runtime is lower) than the mean of TD. This behavior changes when the number of targets is three and greater, there the mean of TD is better. In the continuous case TC is only better (lower runtime) for one salesperson, for two and more salesperson the mean of TFTC is better than the mean

of TC. With the number of salespersons of three and more TC is not able to solve any of the generated instances to optimality within 3 s, where TFTC is able to solve some instances to optimality for three and four salespersons.

7 Conclusion

We addressed the MTSPMT, a dynamic variant of the TSP, where multiple salespersons are searching for their tours in a system with continuously moving targets. We presented four different model formulations, which can be separated by discrete and continuous time handling on one hand and on the other hand by the solution approach (direct or by time-free master plus subproblems). For the time-free variants we presented an exact branch-and-cut algorithm. Due to the different ways of modeling the variants have different characteristics. For instance, TD is sensitive to the level of discretization. Our assumption was, that TFTD might be faster than TD when the discretization is fine enough. It turned out, that the instance size also plays an important role. Thus, the assumption only holds for small instances. Best performance for larger instances is reached by TD. The continuous times variants are much more difficult, but gaps are smaller for TFTC than for TC when larger instances are concerned. Obviously, the time-free modeling variant (TFTC) has an advantage over the direct modeling (TC) so that it produces better bounds and thus better gaps. This is especially visible with a time limit of 3 s.

According to the defense application, TD would be the best choice to use in a moving horizon approach with an adequate discretization level. Since TFTD has the fastest runtimes for small instances with one or two salespersons, it might be suitable for the Moving-Target TSP, where a possible application is a supply ship, that resupplies patrolling boats. For future research we want to have a closer look on online optimization. We want to test our models and solution methods as subroutines in a moving horizon approach. Another point for future work will be to investigate non-solvable instances, that are instances, where it is not possible to reach all targets within their visibility time windows.

Acknowledgements We want to thank Johannes Schmidt for providing the MATLAB code, that we used to generate our graphics in Figs. 4, 5, 6 and 7.

Funding Open Access funding enabled and organized by Projekt DEAL.

References

Abeledo HG, Fukasawa R, Pessoa AA, Uchoa E (2013) The time dependent traveling salesman problem: polyhedra and algorithm. Math Program Comput 5:27–55

Ahrens B (2015) The tour construction framework for the dynamic travelling salesman problem. SouthEastCon 2015:1–8. https://doi.org/10.1109/SECON.2015.7132999

Albiach J, Sanchis JM, Soler D (2008) An asymmetric tsp with time windows and with time-dependent travel times and costs: an exact solution through a graph transformation. Eur J Oper Res 189(3):789–802

Asahiro Y, Horiyama T, Makino K, Ono H, Sakuma T, Yamashita M (2006) How to collect balls moving in the euclidean plane. Discrete Appl Math **154**(16) 2247–2262. https://doi.org/10.1016/j.dam.2006.04.020. Discrete Algorithms and Optimization, in Honor of Professor Toshihide Ibaraki at His Retirement from Kyoto University

Boyd S, Vandenberghe L (2004) Convex optimization. Cambridge University Press, Cambridge

Codato G, Fischetti M (2004) Combinatorial benders' cuts. In: Bienstock D, Nemhauser G (eds) Integer programming and combinatorial optimization, vol 3064. Lecture notes in computer science. Springer, Berlin, pp 178–195. https://doi.org/10.1007/978-3-540-25960-2_14

IBM ILOG CPLEX 12.10 Documentation available at https://www.ibm.com/support/knowledgecenter/SSSA5P_12.10.0/ilog.odms.cplex.help/cplex_KC_home.html (08/2020)

Dijkstra EW (1959) A note on two problems in connexion with graphs. Numer Math 1(1):269–271. https://doi.org/10.1007/bf01386390

Englot BJ, Sahai T, Cohen I (2013) Efficient tracking and pursuit of moving targets by heuristic solution of the traveling salesman problem. In: CDC, pp. 3433–3438. IEEE

Fleischmann B, Gietz M, Gnutzmann S (2004) Time-varying travel times in vehicle routing. Transp Sci 38(2):160–173

Ford LR, Fulkerson DR (1958) Constructing maximal dynamic flows from static flows. Oper Res 6(3):419–433

Fügenschuh A, Nemhauser G, Zeng Y (2013) Scheduling and routing of fly-in safari planes using a flow-over-flow model. In: Jünger M, Reinelt G (eds) Facets of combinatorial optimization. Springer, Berlin, pp 419–447

Garey MR, Johnson DS (1979) Computers and intractability: a guide to the theory of NP-completeness. W. H. Freeman and Company, New York

Gendreau M, Ghiani G, Guerriero E (2015) Time-dependent routing problems. Comput Oper Res 64(C):189–197. https://doi.org/10.1016/j.cor.2015.06.001

Ghiani G, Improta G (2000) An efficient transformation of the generalized vehicle routing problem. Eur J Oper Res 122:11–17

Haghani A, Jung S (2005) A dynamic vehicle routing problem with time-dependent travel times. Comput Oper Res 32(11):2959–2986

Helvig C, Robins G, Zelikovsky A (1998) Moving-target tsp and related problems. In: Bilardi APG, Italiano GF, Pucci G (eds.) Proceedings of the European symposium on algorithms, *Lecture notes in computer science*, vol. 1461, pp. 453–464. Springer, Berlin

Helvig C, Robins G, Zelikovsky A (2003) The moving-target traveling salesman problem. J Algorithms 49(1):153–174

Ichoua S, Gendreau M, Potvin JY (2003) Vehicle dispatching with time-dependent travel times. Eur J Oper Res 144(2):379–396

Jiang Q, Sarker R, Abbass H (2005) Tracking moving targets and the non-stationary traveling salesman problem. Complex Int 11:171–179

Jindal P, Kumar A, Kumar S (2011) Multiple target intercepting traveling salesman problem. Int J Comput Sci Technol 2(2):327–331

Jung S, Haghani A (2001) Genetic algorithm for the time-dependent vehicle routing problem. Transp Res Record: J Transp Res Board 1771:164–171

Laporte G, Mercure H, Nobert Y (1987) Generalized travelling salesman problem through n sets of nodes: the asymmetrical case. Discrete Appl Math 18(2):185–197

Lawler E, Lenstra J, Rinnooy A, Shmoys D (1985) The traveling salesman problem: a guided tour of combinatorial optimization. Wiley, Chichester, New York

Malandraki C, Daskin MS (1992) Time dependent vehicle routing problems: Formulations, properties and heuristic algorithms. Transp Sci 26(3):185–200

Mancini S (2014) Time dependent travel speed vehicle routing and scheduling on a real road network: the case of torino. Transp Res Procedia 3:433–441
Miller CE, Tucker AW, Zemlin RA (1960) Integer programming formulation of traveling salesman problems. J ACM (JACM) 7(4):326–329
Noon CE, Bean JC (1993) An efficient transformation of the generalized traveling salesman problem. Inf Syst Oper Res 31(1):39–44
Picard JC, Queyranne M (1978) The time-dependent traveling salesman problem and its application to the tardiness problem in one-machine scheduling. Oper Res 26(1):86–110
Reinelt G (1994) The traveling salesman: computational solutions for TSP applications. Springer, Berlin
Reyes D, Savelsbergh M, Toriello A (2017) Vehicle routing with roaming delivery locations. Transp Res Part C: Emerg Technol 80(C):71–91. https://doi.org/10.1016/j.trc.2017.04.003
Soler D, Albiach J, Martínez E (2009) A way to optimally solve a time-dependent vehicle routing problem with time windows. Oper Res Lett 37(1):37–42
Stieber A, Fügenschuh A (2018) The multiple traveling salesmen problem with moving targets and nonlinear trajectories. In: Kliewer N, Ehmke JF, Borndörfer R (eds) Operations research proceedings 2017. Springer International Publishing, Cham, pp 489–494
Stieber A, Fügenschuh A, Epp M, Knapp M, Rothe H (2014) The multiple traveling salesmen problem with moving targets. Optim Lett 9(8):1569–1583
Sundar K, Rathinam S (2016) Generalized multiple depot traveling salesmen problem— polyhedral study and exact algorithm. Comput Oper Res 70:39–55
Woensel TV, Kerbache L, Peremans H, Vandaele N (2007) A queueing framework for routing problems with time-dependent travel times. J Math Model Algorithms 6(1):151–173