Applied Mathematics

Helmut Schmidt Universität
Universität der Bundeswehr Hamburg

Angewandte Mathematik und Optimierung Schriftenreihe
Applied Mathematics and Optimization Series
AMOS # 46(2016)

Anke Stieber and Armin Fügenschuh

Variants in Modeling Time Aspects for the Multiple Traveling Salesmen Problem with Moving Targets

Herausgegeben von der
Professur für Angewandte Mathematik
Professor Dr. rer. nat. Armin Fügenschuh

Helmut-Schmidt-Universität / Universität der Bundeswehr Hamburg
Fachbereich Maschinenbau
Holstenhofweg 85
D-22043 Hamburg

Telefon: +49 (0)40 6541 3540
Fax: +49 (0)40 6541 3672

e-mail: appliedmath@hsu-hh.de
URL: http://www.hsu-hh.de/am

Angewandte Mathematik und Optimierung Schriftenreihe (AMOS), ISSN-Print 2199-1928
Angewandte Mathematik und Optimierung Schriftenreihe (AMOS), ISSN-Internet 2199-1936

Variants in Modeling Time Aspects for the Multiple Traveling Salesmen Problem with Moving Targets

Anke Stieber · Armin Fügenschuh

Abstract The multiple traveling salesmen problem with moving targets (MTSPMT) is a generalization of the classical traveling salesmen problem (TSP), where the targets (cities or objects) are moving over time. Additionally, for each target a visibility time window is given. The task is to find routes for several salesmen so that each target is reached exactly once within its visibility time window and the sum of all traveled distances of all salesmen is minimal. Applications of the described problem class can be found, e.g., in supply logistics and in the defense sector. Such instances can be very large in the number of variables and therefore time consuming to solve numerically. In handling the time aspect in different ways we present four distinct modeling approaches for the MTSPMT. First we present a time-discrete formulation embedded in a time-expanded network. The second model uses big-M constraints to arrange the time in a continuous way, which leads to a quadratic programming problem. The other two models comprise a time-relaxation approach where time-feasibility is ensured by solving a number of subprograms. For modeling of the subprograms to reintegrate time, again two possibilities are available: time discretization and applying big-M constraints. The solution procedure to solve the time-relaxed variants is proposed and computational results for randomly generated test instances to compare all different modeling approaches are presented. For problem instances with a fine discretization the time-relaxed variant with continuous times is the most promising one with respect to computational running times.

A. Stieber*
Tel.: +49-40-6541-2755
E-mail: anke.stieber@hsu-hh.de

A. Fügenschuh*
Tel.: +49-40-6541-3540
E-mail: fuegenschuh@hsu-hh.de

*Helmut Schmidt University / University of the Federal Armed Forces Hamburg
Holstenhofweg 85, 22043 Hamburg, Germany

Keywords dynamic traveling salesmen problem · moving targets · time-relaxation · integer linear programming · second-order cone programming · modeling

1 Introduction

In the classical TSP the targets (cities, points) are static entities and only one salesman is considered to traverse each target exactly once. A static TSP dataset consists of a certain number of targets and no targets are added or removed when processing, i.e., when the salesman traverses the cities. The location of all cities are fixed and there is no time aspect regarding the traversal of the dataset.

The multiple traveling salesmen problem with moving targets (MTSPMT) deals with the case that cities or targets are assigned to several salesmen. There is a visibility time window for each target, which defines the time when the target is added and the time when the target is removed again. In addition, the targets are not fixed, instead they vary their respective local positions over time, meaning the targets are moving on certain trajectories with a certain variable speed function. All salesmen start their tour from an initial depot. Each target must be visited once by exactly one salesman within its visibility time window. The objective is to minimize the total traveled distances of all salesmen. If we consider only one salesman and all cities are fixed and visible over the whole time horizon, we obtain the classical traveling salesman problem, which is NP-hard, see Garey and Johnson [6]. Thus, the MTSPMT as a generalization of the classical TSP is NP-hard, too.

The MTSPMT as a dynamical generalization of the TSP is more suitable to a wide range of real-world problems. A possible application of the MTSPMT can be found in the defense sector. An area, e.g., an airport or a military base, must be protected from incoming rocket, artillery or mortar fire (RAM). With a battery of laser guns deployed over or near the protected area the decision has to be made, which available laser to select for the countermeasure. Here, laser guns correspond to the salesmen and the incoming fire are the targets, where each target has a certain visibility time window and its local position is changing over time. In fact, this application is an online optimization problem, but it can be solved 'offline" with a moving horizon approach. For a detailed description of the application we refer to Stieber et al. [18].

Helvig et al. [8] adressed the Moving-Target TSP, which is the MTSPMT restricted to one salesman. As possible applications they mention 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 pursuer are considered and each pursuer must return to the origin for resupply after intercepting each target.

Many practical application such as routing and scheduling of vehicles have a dynamic component inherent. The time aspect is considered in traversing from customer to customer (or target to target). A related formulation can be found in the Stochastic Vehicle Routing Problem (SVRP) described by Gendreau et al. [7], where some problem characteristics are stochastic. As an example which is related to our problem see the m-TSP with Stochastic Travel Times (m-TSPST). This problem considers m vehicles to serve a number of customers. In this variant the

length of the arcs vary, that means travel times between nodes are time dependent, e.g., due to congestions on roads. Often the vehicles are associated with basic costs in order to minimize the number of used vehicles in the solution.

We want to investigate the time aspect of the MTSPMT. One way of modeling is the use of discrete time steps and embedding a multi-commodity flow problem in a time-expanded network. In case the precision and therefore the number of used time steps is increased, the size (measured in number of variables and constraints) of those problem instances can grow dramatically. We want to tackle this difficulty by presenting different modeling formulations regarding the time aspect. Instead of applying discrete time steps we introduce continuous time variables and solve instances by means of second-order cone programming (SOCP), see Lobo et al. [14] for an overview on SOCP. Another variant is to relax the time aspect completely, solve the resulting small problem and construct a time-feasible solution from it by introducing only parts of the time-discrete restrictions. Not every time-relaxed solution can result in a time-feasible solution. To this end, nearly all time-relaxed solutions have to be exploited. Therefor we embed the solution and construction procedure into a branch-and-bound framework to find a global optimal time-feasible solution. Instead of basing the construction process on the time discretization we are also able to use continuous time variables, which gives us the last modeling variant.

To test and compare all variants we used randomly generated problem instances with varying numbers of targets and salesmen. Computational tests with different discretization precisions are conducted.

The remainder of this article is organized as follows. In Section 2 we provide a survey of the relevant literature. Two model variants which incorporate the time as a discrete and a continuous concept are presented in Section 3. The relaxation of time and its solution method is addressed in Section 4 and 5. Computational results are presented in Section 6 and afterwards we conclude.

2 Literature

Our contribution addresses a generalization of the classical traveling salesman problem (TSP) by considering more than one salesman and moving targets with time windows. For a survey on the classical TSP we refer to Lawler et al. [12] or Reinelt [17].

There are a number of articles in the literature that concern dynamical TSP generalizations. However, representative articles often relate to TSP generalizations with a number of problem restrictions. For example the traveling salesman problem with moving targets (TSPMT) and several variants of this problem are addressed in Helvig et al. [8,9]. Restrictions are made for target speed, geometric behavior of the system and problem size (only few targets), thus the proposed algorithms are not applicable to the general case of TSPMT. Variants of TSPMT and TSPMT with resupply, where salesmen must return to the origin after intercepting a target are also addressed by Jiang et al. [10], Jindal et al. [11] and Liu [13].

Another application is described by Menezes et al. [15]. Several customers (people, objects or vehicles) move through a system, where each customer may have multiple demands for a resource while traveling. Mobile servers meet the

demand and provide for the customers. The authors assume that the customers are moving on structured paths through the system. One mobile server can intercept the customers multiple times to provide for their demand. Here the objective is not serving all points and minimizing the total traveled distance, but to maximize utility or profit. Examples of such a system might be people waiting in a line in a theme park or at an attraction. The authors do not solve practical problem instances, but structure and analyze the problem and answer the question how long should the server stay at any one location.

Some articles address the dynamic traveling salesman problem, which is very similar to our problem description. Here, locations of targets change over time and targets may be added or deleted, but generally only one salesman is considered. Problem instances were mainly solved by heuristics as an online approach like Ahrens did in [1]. He created dynamic TSP instances by using static TSP datasets from a published and standardized library. 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. This movement is modeled by a Gaussian-distributed random distance that is added to the static points from the original dataset in each time step. 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".

In contrast to most of the mentioned articles we address the MTSPMT, a TSP with several salesmen and moving targets with a visibility time interval corresponding to adding and removing targets. In this research we concentrate on an offline approach for solving MTSPMT instances to global optimality. In real-world online applications this can be used by applying a moving horizon approach. Since the time inherent in the dynamic variants of the TSP makes instances difficult in managing, we examine different ways of how to model time in these problems. Furthermore exact solution methods with regard to the different model variants are discussed.

3 Two Mathematical Models

Here, we introduce some basic notation to formulate the MTSPMT as an optimization problem. Let $\mathcal{W} = \{1, \dots, w\}$ be a set of salesmen. All salesmen start their respective tour at the same location h. We assume a finite time horizon $[0, T]$. Let $\mathcal{V} = \{1, \dots, n\}$ be a set of nodes (targets, cities or customers) and $\mathcal{A} \subseteq \mathcal{V} \times \mathcal{V}$ be a set of arcs (roads). 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 salesman k traveling from node i to node j starting at time s in i and arriving at time t in j is given by the function $c_{i,j,k} : [0, T] \times [0, T] \to \mathbb{R}_+ \cup \{\infty\}$. Each target $i \in \mathcal{V}$ is assigned a time window $[t_i^S, t_i^E]$. Since the targets are only visible within this certain time window we have $c_{i,j,k}(s, t) = \infty$ if s or t is outside this window. A maximum speed value v^{max} is assigned to the salesmen, that is the fastest speed at which they can traverse an arc in the underlying network. The goal is to reach each target from $\mathcal{V}$ by exactly one salesman from $\mathcal{W}$ such that the sum of all traveled distances of all salesmen is minimal.

3.1 Time-Discrete Model

We have already addressed the time-discrete mixed-integer linear programming (MILP) formulation in [18]. For the convenience of the reader it is presented here in a concise way. We put a multi-commodity flow formulation in a time-expanded network. For an explanation of time-expanded networks see Ford and Fulkerson [4]. Therefor, we introduce a discretization of time. Let m be an integer number. The step size is defined by $\Delta := T/m$. Then the set of all time steps is denoted by $\mathcal{T} := \{0, \ldots, m\}$. Regarding different time steps, we have to take into account different arcs between any pair of nodes i and j. $\mathcal{A}^{TD} \subseteq (\mathcal{V} \times \mathcal{T}) \times (\mathcal{V} \times \mathcal{T})$ is the set of arcs, where an arc $(i, s, j, t) \in \mathcal{A}^{TD}$ now depends on the departure time s and arrival time t, when traveling from node i to j. For each salesman 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, thus, salesmen do not necessarily use their maximum speed. Having discrete points of time $p, q \in \mathcal{T}$ we are able to evaluate the distance function c for arcs at these points $c^{p,q}_{i,j,k} := c_{i,j,k}(p, q)$.

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 salesman 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 salesmen:

$$\sum_{k \in \mathcal{W}} \sum_{(i,p,j,q) \in \mathcal{A}^{TD}} c^{p,q}_{i,j,k} x^{p,q}_{i,j,k} \quad \rightarrow \quad \min\,. \tag{1}$$

The demand constraint requires, that each node j must be visited once by exactly one salesman:

$$\sum_{k \in \mathcal{W}} \sum_{i \in \mathcal{V}} \sum_{(p,q):(i,p,j,q) \in \mathcal{A}^{TD}} x^{p,q}_{i,j,k} = 1, \quad \forall\, j \in \mathcal{V}. \tag{2}$$

Each salesman k can do at most one trip at any time step p:

$$\sum_{(i,p,j,q) \in \mathcal{A}^{TD}} x^{p,q}_{i,j,k} \leq 1, \quad \forall\, k \in \mathcal{W}, p \in \mathcal{T}. \tag{3}$$

The following flow conservation constraints ensure the feasibility of time, where each node j at which salesman k ends his tour can be regarded as the sink of the flow:

$$\sum_{(i,p):(i,p,j,q) \in \mathcal{A}^{TD}} x^{p,q}_{i,j,k} \geq \sum_{(i,p):(j,q,i,p) \in \mathcal{A}^{TD}} x^{q,p}_{j,i,k}, \quad \forall\, j \in \mathcal{V}, q \in \mathcal{T}, k \in \mathcal{W}. \tag{4}$$

Summing up, we solve the following optimization problem:

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

The presented model (5) is not restricted to special shapes of target trajectories. It can handle any trajectory. Likewise, there is no need to restrict the speed function of the targets, the model is able to deal with varying target speeds. A serious drawback of this model is the use of a time discretization. In case there is a need for a far better accuracy, the level of discretization has to be increased, resulting in more time steps per target. Thus, the number of arcs grows and thereby the number of variables and constraints of the model, which leads to a higher computational burden.

3.2 Time-Continuous Model

Another way of modeling the time aspect in the MTSPMT is to apply the big-M method (Miller-Tucker-Zemlin [16] constraints) instead of embedding the multi-commodity flow structure in a time-expanded network. In this case the time is modeled by continuous time variables and time restriction constraints.

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 salesman k from i to j (independent of the time). Additionally, time-continuous variables $t_{i,k} \in \mathbb{R}$ are defined to describe the arrival time of salesman k in node i. Now, we are able to formulate the continuous model.

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

$$\sum_{k \in \mathcal{W}} \sum_{(i,j) \in \mathcal{A}} c_{i,j,k}(t_{i,k}, t_{j,k}) x_{i,j,k} \quad \rightarrow \quad \min\,. \tag{6}$$

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

$$\sum_{k \in \mathcal{W}} \sum_{i:(i,j) \in \mathcal{A}} x_{i,j,k} = 1, \quad \forall\, j \in \mathcal{V}. \tag{7}$$

Each salesman k can do at most one trip:

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

The following constraints ensure flow conservation:

$$\sum_{(i):(i,j) \in \mathcal{A}} x_{i,j,k} \geq \sum_{(i):(j,i) \in \mathcal{A}} x_{j,i,k}, \quad \forall\, j \in \mathcal{V}, k \in \mathcal{W}. \tag{9}$$

The Miller-Tucker-Zemlin constraints guarantee time-feasibility, that means, if salesman k goes 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 $t_{j,k}$ when using maximum speed. The time horizon T is the so called big-M constant in the following big-M constraints

$$t_{i,k} + \frac{c_{i,j,k}(t_{i,k}, t_{j,k})}{v^{max}} \leq t_{j,k} + T \cdot \left(1 - x_{i,j,k}\right), \quad \forall\, (i,j) \in \mathcal{A}, k \in \mathcal{W}. \tag{10}$$

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

$$t_j^S \leq t_{j,k} \leq t_j^E, \quad \forall\, j \in \mathcal{V}, k \in \mathcal{W}. \tag{11}$$

Summarized, we aim to solve the following optimization problem:

$$\min\{(6) \mid (7), (8), (9), (10), (11), x \in \{0,1\}^{\mathcal{A} \times \mathcal{W}}, t \in \mathbb{R}^{\mathcal{V} \times \mathcal{W}}\}. \tag{12}$$

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 nodes. In order to apply a standard MILP solver such as IBM ILOG CPLEX or Gurobi, 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. Here,

we use the same constant speed for all targets. Then $c_{i,j,k}$ represents the Euclidean distance between two points on a straight line. With this, the above presented optimization problem (12) can be handled as a second order cone program (SOCP). To solve instances based on formulation (12) with a standard MILP solver, the model has to be adjusted to the rules of the respective solver to formulate SOCPs (e.g., by introducing auxiliary variables for CPLEX). The constraints (10) define the cones and make the set of feasible solutions to be convex.

Following the above given assumptions also for formulation (5) it turned out, that the time-continuous formulation has a major advantage. It does not concentrate on discrete time steps, thus, a salesman is able to reach a moving node at any possible point of its trajectory. In all cases, the precision of the optimal solution of (12) is equal or better than the optimal solution of (5). The objective function value of (12) serves as a lower bound for the objective function value of (5) under the given trajectory assumptions. However, model formulations based on big-M constraints usually have a weak linear programming relaxation and thus, more nodes have to be examined, which slows down the solution process [2].

4 Time-Relaxation

Assume we raise the number of time steps m to a fine discretization for the time-discrete model, the resulting MILPs get extremely large and standard MILP solvers do not find an optimal solution in reasonably short time. In the following we use a method to tackle such problems by relaxing the time aspect in the model. This technique was first introduced by Fügenschuh et al. [5]. They successfully applied the method for scheduling and routing a small number of planes for fly-in safaris.

First of all, we perform a projection of (5) from the time-discrete variable space $\mathcal{A}^{TD} \times \mathcal{W}$ to $\mathcal{A} \times \mathcal{W}$ (i.e. from $(\mathcal{V} \times \mathcal{T}) \times (\mathcal{V} \times \mathcal{T}) \times \mathcal{W}$ to $\mathcal{V} \times \mathcal{V} \times \mathcal{W}$). The time-free counterpart of variables $x^{p,q}_{i,j,k}$ is simply $x_{i,j,k}$. The distance coefficients $c_{i,j,k}$ are formed by minimizing the distance over all time-expanded arcs from i to j, $c_{i,j,k} = \min\{c^{p,q}_{i,j,k} \mid p, q \in \mathcal{T}\}$. With this, we have the following model in the time-free space:

$$\min \sum_{k \in \mathcal{W}} \sum_{(i,j) \in \mathcal{A}} c_{i,j,k} x_{i,j,k} \tag{13}$$

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

$$\sum_{j:(h,j) \in \mathcal{A}} x_{h,j,k} \leq 1, \quad \forall\, k \in \mathcal{W}$$

$$\sum_{i:(i,j) \in \mathcal{A}} x_{i,j,k} - \sum_{i:(j,i) \in \mathcal{A}} x_{j,i,k} \geq 0, \quad \forall\, j \in \mathcal{V}, k \in \mathcal{W}.$$

This is a classical multi-commodity flow problem, which is easy to solve by standard MILP solvers. The optimal solution of the time-relaxed (or time-free) model (13) serves as a lower bound to (5), because the distance coefficients are computed as the minimum over all arc distances between two nodes. However, the reconstruction of a time-feasible solution from a time-free solution is not straightforward and not

every time-free solution yields in a time-feasible solution. For this purpose we have to run through nearly every time-free solution and try to create time-feasibility. We embed this construction within a branch-and-bound framework.

In the branch-and-bound framework the time-free model (13) serves as the master problem. Given an optimal solution of the master problem, feasible times at which salesmen reach the nodes, have to be constructed from it. In case the objective function value of the constructed time-feasible solution is equal to the objective function value of the time-free solution, we are done and the constructed solution is proven global optimal for (5). However, this rarely happens. Therefore, the master problem is embedded in a branch-and-bound framework in order to obtain all possible time-free solutions. That means the master problem is treated as infeasible. For any of the time-free solutions, we try to construct a feasible counterpart with respect to the time constraints. If the time-feasible solution has the best objective function value found so far, it is stored. The time-free solution is then cut off in the branch-and-bound process. This can additionally be done for all salesmen permutations in order to prevent a repetition due to symmetries. If for a given time-free solution no time-feasible solution exists, we have to cut off the time-free solution as well. This method is realized by using the callback functionality of CPLEX. Analyzing the time-free solution before trying to construct time-feasibility leads to a speed up in processing. For this we refer to the Section 5. For now, we want to concentrate on the construction of a time-feasible solution from a time-free solution.

A tour in a time-free solution may be not connected, thus we call it a pretour. Assume we have a time-free solution, which is given through a bunch of pretours (not more than the number of salesmen). Then, a salesman is called active, if and only if there is an arc $(i, j) \in \mathcal{A}$ where the corresponding binary decision variable is nonzero in the given time-free solution. Pretours for all active salesmen are extracted from the solution. For each of these pretours a time-feasible tour is required.

4.1 Discrete time-feasibility checking

For a given time-free solution we construct a time-feasible solution by setting up a checking sub-MILP for each of the time-free salesman tours. The sub-MILP makes use of formulation (5) restricted to those arcs that correspond to binary variables, that are nonzero in the given time-free solution. It has to be checked for each salesman if a feasible tour exists that can be projected to the corresponding part of the time-free solution. If the checking MILP results in a feasible tour for all salesmen a total time-feasible tour is found.

The time-feasibility checking MILP is set up as a minimum flow problem from a source to a sink. For salesman s we are able to extract the sequence of targets s has to visit from the binary variables x in the time-free solution. Let us assume n_s is the number of targets s has to intercept and $(v_1, v_2, \ldots, v_{n_s})$ is the sequence extracted from the x variables of the time-free solution. Additionally home position h is considered as the source and we extend our network by a node t, which serves as the sink. Thus, $\mathcal{V}^s = \{h, v_1, v_2, \ldots, v_{n_s}, t\}$ denotes the sequence of targets for the flow.

From the set of all time-expanded arcs we now have to consider only those arcs where the corresponding x variable is non-zero in the time-free incumbent. That means, the set of arcs our checking MILP is based on, consists of all time-discrete arcs departing from home position h and arriving in v_1, from v_1 to v_2 and so on, until from v_{n_s-1} to v_{n_s}. Additionally, we have to introduce artificial time-discrete arcs from v_{n_s} to sink t. That means, for each arc, that arrives in v_{n_s} an arc is introduced departing at the time it arrived in v_{n_s} and arriving in t one time step later than the latest possible arrival time of s in v_{n_s}. The distance of all arcs between v_{n_s} and t is zero. We denote the set of all arcs by $\mathcal{A}^s$.

According to the time-discrete model, we introduce binary decision variables $x^{p,q}_{i,succ(i)}$ describing the decision of sending salesman s from target i to its successor $succ(i)$ starting at time step p and arriving at time step q. The corresponding predecessor of i is denoted by $pred(i)$. Then, the time-feasibility checking MILP for salesman s is formulated as follows:

$$\min \quad \sum_{(i,p,succ(i),q)\in\mathcal{A}^s} c^{p,q}_{i,succ(i),s} x^{p,q}_{i,succ(i)}, \tag{14}$$

$$\text{s.t.} \sum_{p:(pred(j),p,j,q)\in\mathcal{A}^s} x^{p,q}_{pred(j),j} = \begin{cases} -1, & \text{if } j = h, \\ 1, & \text{if } j = t, \\ \sum_{p:(j,q,succ(j),p)\in\mathcal{A}^s} x^{q,p}_{j,succ(j)}, & \text{otherwise.} \end{cases}$$

The optimization problem (14) aims to find the shortest flow from h to t. This kind of optimization problem can be solved in polynomial time by, e.g., Dijkstra's algorithm [3]. In case the checking MILP (14) results in a feasible tour for each salesman, we are able to construct a time-feasible solution for (5) by combining all salesman tours. If any of the salesman solutions is infeasible the construction process is aborted.

4.2 Continuous time-feasibility checking

According to our time-continuous model (12), there is another possibility to formulate the feasibility checking sub-MILP. In case the trajectories of all targets are straight lines with constant speed, the checking MILP can be modeled as a quadratic program without discrete time steps:

$$\min \quad c_{h,v_1}(t_h,t_1)\, x^{t_h,t_1}_{h,v_1} + \sum_{i=v_1}^{v_{n_s-1}} c^{t_i,t_{succ(i)}}_{i,succ(i)}\, x^{t_i,t_s ucc(i)}_{i,succ(i)}, \tag{15}$$

$$\text{s.t.} \quad c^{t_i,t_{succ(i)}}_{i,succ(i)} \le v^{\text{MAX}} \cdot (t_{succ(i)} - t_i), \quad \forall\, i = h, v_1, \ldots, v_{n_s},$$

$$t^S_i \le t_i \le t^E_i, \quad \forall\, i = 1, \ldots, n_s.$$

Here, $c_{i,j}(t_i,t_j)$ again denotes the Euclidean distance between position of node i at time step t_i and node j at time step t_j. In the objective function all traveled distances are summed up, while in the restriction time-feasibility is checked. That means, the travel time s needs to go from i to $succ(i)$ has to be at least that high, that the used speed of traveling is at most v^{MAX}, the maximum speed. The bounds restriction ensures that each node is reached within its specified visibility window.

5 Implementational Details

To solve the time-free problem and to construct feasible times for the salesman tours, we embed the time-relaxed problem (13) within a branch-and-bound framework. The model (13) is called master problem. To check the solutions of the master problem and to produce the best time-feasible solution we used the callback utilities of CPLEX. We implemented an instance of the LazyConstraintCallback and an instance of the BranchCallback.

Each time a candidate feasible solution of the master problem is found at a node in the branch-and-bound tree CPLEX' LazyConstraintCallback is invoked. This is a user written callback to solve mixed-integer linear programs (MILP) while applying constraints in a "lazy" fashion, i.e., only when they are violated.

The LazyConstraintCallback performs a checking of the candidate feasible solution (master problem) and a possible construction of the feasible times according to (14) and (15). Algorithm 1 presents an overview of the callback. A more detailed description of single steps is given in the following.

In line 2-3 of the algorithm the branching rule is given. If the lower bound at a node is greater or equal to the best found objective function value of an incumbent, we will exit this callback and prune the current node. This is done by CPLEX' BranchCallback, which is invoked afterwards. Besides pruning subtrees the BranchCallback performs branching the same way CPLEX is suggesting it.

Since (13) is a multi-commodity flow formulation, any solution presents a feasible flow but not necessarily a feasible salesman tour due to the lack of subtour elimination constraints. In line 4-6 of Algorithm 1 we extract the solution for every salesman and test whether one of these candidate tours contains a cycle. If a cycle $\mathcal{C} \subset \mathcal{A}$ is found, it will be cut off by the following constraints:

$$\sum_{(i,j)\in\mathcal{C}} x_{i,j,k} \leq |\mathcal{C}| - 1, \quad \forall k \in \mathcal{W}. \tag{16}$$

Before setting up a MILP to solve the time-feasibility checking, we are able to exploit the visibility windows of the sequential targets of a candidate salesman tour to get information about time-feasibility. In line 7-9 a prechecking on time-feasibility is performed by means of interval propagation. To check the tour of an active salesman we start at the first node of this tour. It can be reached over the whole visibility window from the salesman home position. Then, the visibility window defines the departure interval to travel to the second node. This has to be adjusted to the arrival interval of the second node, which is related to the visibility window of the second node. This procedure is visualized in an example shown in Figure 1. Here, the arrival interval of node v_1 is the discrete interval $[2, 5]$. This interval reduces to the departure interval $[2, 3]$ because with a later departure the salesman cannot reach the following node v_2 within its visibility window of $[0, 4]$. Then, the corresponding arrival interval of v_2 is $[3, 4]$. The departure interval of v_2 has to be within or equal to $[3, 4]$. It has to be checked with the visibility window of the next node and so on. In case there is an infeasible interval, we have the information that the corresponding salesman tour is time-infeasible. Thus, the solution can be rejected. This is done by adding a global cut to the master problem.

Algorithm 1: CPLEX LazyConstraintCallback: Check time-free solution and if exists construct solution with times.

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

#Exploit bounds
if *current objective function value* $curr\_obj\_val \geq best\_obj\_val$ **then**
    return;
#Cycle detection
if *x solution of an active salesman s contains a cycle* **then**
    add cut for all salesmen;
    return;
#Time-feasibility checking by interval propagation
if *x solution of an active salesman s can be identified as time infeasible* **then**
    add global cut for all salesmen;
    return;
#Checking
initialize current solution $curr\_sol \leftarrow \emptyset$;
initialize $all\_salesmen\_feasible \leftarrow true$;
initialize objective function value for time-feasible solution $tour\_val \leftarrow 0$;
for *each* $s \in \mathcal{W}$ **do**
    if *s is not active* **then**
        continue;
    if $all\_salesmen\_feasible \neq true$ **then**
        break;
    if *time-free x solution part for s is already in solution pool* **then**
        get time-feasible solution tour t for s;
        if *t is infeasible* **then**
            $all\_salesmen\_feasible \leftarrow$ false;
    else
        set up MILP to compute time-feasible tour t for s;
        solve MILP;
        if *MILP is infeasible* **then**
            $all\_salesmen\_feasible \leftarrow$ false;
            add cut to cut off this x solution part for all salesmen;
        else
            $\#t$ is valid tour of s;
            $tour\_val \leftarrow$ add objective function value of t;
            $curr\_sol = curr\_sol \cup t$;
        add time-free x solution of s and t to solution pool;
if $all\_salesmen\_feasible = true$ **then**
    add cut to prevent solution to be repeated by other salesmen;
    if $tour\_val < best\_obj\_val$ **then**
        best_obj_val = tour_val;
        save curr_sol;
        return;

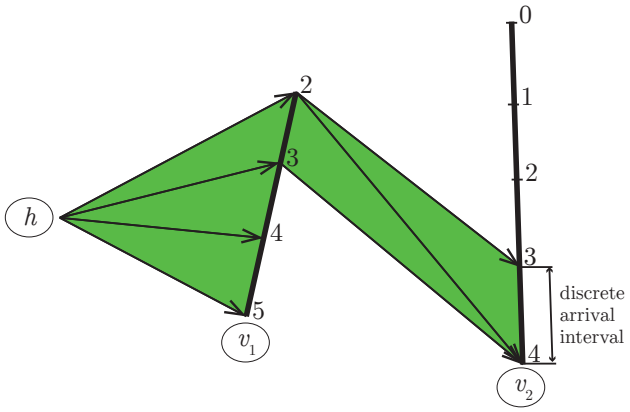

Fig. 1 Interval propagation. This figure presents the home position h of all salesman and a given salesman tour consisting of the sequence h, v_1, v_2. Each target is visualized by its trajectory and the corresponding discrete time steps, which are given by numbers.

Therefore, consider an infeasible sequence of arcs $\mathcal{P} \subset \mathcal{A}$ from node r to node s, we can formulate the following constraints for each salesman to cut off this solution:

$$\sum_{(i,j)\in\mathcal{P}} x_{i,j,k} \leq |\mathcal{P}| - 1, \quad \forall k \in \mathcal{W}. \tag{17}$$

For a pair of anti-parallel arcs (i, j) and (j, i) we have

$$x_{i,j,k} + x_{j,i,k} \leq 1, \quad \forall k \in \mathcal{W}. \tag{18}$$

Thus, lifting $|P| - 1$ backward arcs into the cut extends the applied constraints to:

$$\sum_{(i,j)\in\mathcal{P}} x_{i,j,k} + \sum_{(i,j)\in\mathcal{P}\setminus(pred(s),s)} x_{j,i,k} \leq |\mathcal{P}| - 1, \quad \forall k \in \mathcal{W}. \tag{19}$$

Interval propagation for the time-feasibility checking with continuous times is done in a similar way. In general the resulting arrival interval is slightly larger, see Figure 2, due to an exact calculation of the travel time. The travel time is not rounded to the next time step, its exact value is computed from the maximum salesman speed and the Euclidean distance between the corresponding positions of the targets. With this a salesman is able to arrive earlier and to depart later compared to the time-discrete case. Computation of the exact departure and arrival interval consists of identifying the values $t^{dep}_{min}, t^{dep}_{max}, t^{arr}_{min}$ and t^{arr}_{max}, as visualized in Figure 2. As an initialization we take the whole visibility window $[t^S, t^E]$ as the departure interval and therefore compute the possible arrival interval for the arrival node. Since only a constant maximum speed is considered we take the equation of motion to compute t^{arr}_{min}, which has to be within its visibility window:

$$v^{max} \cdot (t^{arr}_{min} - t^S) = \left\| p_{arr} - p_{dep} \right\|_2, \tag{20}$$

where p_{arr} and p_{dep} are the positions of the nodes at the times t^{arr}_{min} and t^S respectively. Since the right hand side of the motion equation is also depended on

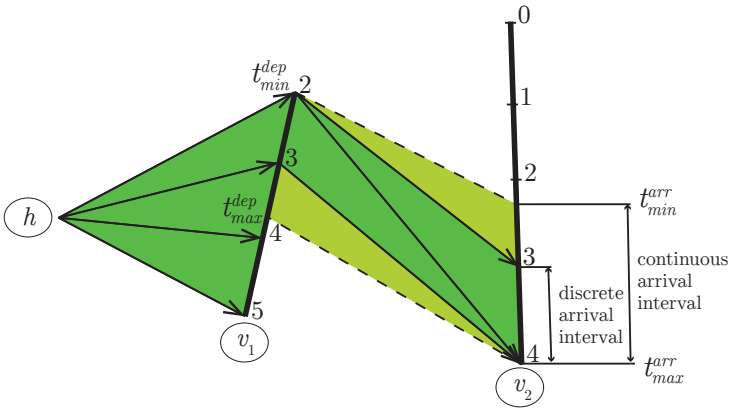

Fig. 2 Interval propagation. Visualized is the discrete departure and arrival interval between consecutive nodes and the extended departure and arrival interval between consecutive nodes when continuous times are considered. This figure presents the home position h of all salesman and a given salesman tour consisting of the sequence h, v_1, v_2. Each target is visualized by its trajectory and the corresponding discrete time steps, which are given by numbers. The variables t^{dep}_{min} and t^{dep}_{max} define the exact departure interval and t^{arr}_{min} and t^{arr}_{max} define the exact arrival interval.

t^{arr}_{min} we have to square both sides and replace p_{arr} and p_{dep} by their trajectory descriptions. This leads to a quadratic equation and t^{arr}_{min} can be obtained. t^{arr}_{max} is set to t^E of the arrival node, because waiting is permitted for salesman s.

The next step is to calculate the right departure interval based on the recently computed arrival interval in the same way. For the next pair of consecutive nodes an intersection of the departure interval and the arrival interval of the former pair has to be taken. In case an empty intersection has been detected the salesman tour is time-infeasible.

As mentioned earlier CPLEX' BranchCallback is responsible for pruning nodes or subtrees. The decision on which variable branching is performed adheres to CPLEX, but the BranchCallback supports a way to give information to the new generated nodes. We use this technique to give an adjusted set of arcs to the new nodes. This set is then further adjusted according to CPLEX' branching strategy. That means when branching is performed we get new information about the solution in this subtree in the case the branch variable is set to 1. In the other case, where the branch variable is set to 0 there is no additional information. More precisely, assume the branch variable that is set to 1 puts node w after node v for salesman s. Then, we have the information that all existing time-expanded arcs $\{(i,p,w,q) \in \mathcal{A}^{TD} \mid i \in \mathcal{V} \setminus_{\{v\}}\}$ that end in w can be removed for the solution procedure of the current node being processed. This information is passed down in the branch-and-bound tree and further adjusted in the corresponding subtree at nodes with branch variables, which are set to 1. Based on the local and adjusted set of time-expanded arcs, we can recompute the distance coefficients $c_{i,j,k}$ for the time-relaxed model (13). In case the distance coefficients change, we introduce a new constraint to tighten the time-free model. This gives us an extended solution procedure. Additionally, time-feasibility checking by interval

Table 1 Instance settings.

	salesmen				
targets	1	2	3	4	6
4	×	×	-	×	-
6	×	×	×	-	×
8	×	×	-	×	-

propagation can be performed on the local time-expanded set of arcs after global time-feasibility checking. In case of time-infeasibility a local cut is added to the current subtree. In any other case a time-feasible solution can be obtained because of valid departure and arrival time intervals. This solution is then calculated by setting up and solving the checking MILP.

Finally, when a time-feasible solution is found there is a time-feasible tour for each active salesman. This solution has to be returned manually to the master problem and to the best solution found so far. Infeasible solutions are cut off by lazy constraints.

Summing up, we have the time-discrete model (5) (TD for short), the time-continuous model (12) (TC), the time-free model (13) with time-discrete feasibility checking (14) according to Algorithm 1 (TF-TD), the time-free model (13) with time-continuous feasibility checking (15) according to Algorithm 1 (TF-TC) and the time-free model (13) with discrete time-feasibility checking according to Algorithm 1 extended by local information (TF-TDloc).

6 Computational Results

We applied our methods described above to a set of randomly generated test instances. A test instance is generated by specifying the number of salesmen and the number of moving targets. For each of those pairs we created 25 instances, where the trajectory generation with more targets are based on the trajectories for fewer targets. That means an instance with 6 targets contains the same 4 trajectories as in the instance with 4 targets (plus 2 new ones). We generated our test instances and checked afterwards if they are really solvable by a primitive check. This check was simply a test if the number of targets is divisible by the number of salesmen without remainder. The positive outcome provided an allocation of targets to salesmen. With the allocation we tested if each salesman can intercept its part of the targets one after another in the middle of the respective trajectories. If it is not possible our trajectory generation is repeated. Due to this checking procedure, the target and salesmen pairs we used can be found in Table 1. The operating space is a square of size 500 length units. Due to reasons of visualization the trajectories of the targets are generated to have random lengths between 100 and 400 length units. The targets are assumed to have a constant traveling speed of 16 length units per time step. That leads to the normal or single (1) precision of time steps. The instances are then expanded by a higher discretization of time steps. We used a double (2) and a fourfold (4) precision for our tests. More precisely for two consecutive time steps we introduced a new one in the middle for double precision. Taking double precision and again introducing new time steps in the middle of two consecutive time steps leads to fourfold precision.

Table 2 CPLEX parameter settings.

model	CPLEX parameter	parameter value
TD	EpGap	0.0
	MIPEmphasis	3
TC	EpGap	0.0
	MIPEmphasis	3
	BarQCPEpComp	1e-8
TF-TD	EpGap	0.0
	MIPEmphasis	3
	HeurFreq	-1
TF-TD subMILP[1]	EpGap	0.0
TF-TC	EpGap	0.0
	MIPEmphasis	3
	HeurFreq	-1
TF-TC subMILP	EpGap	0.0
	BarQCPEpComp	1e-8

The salesmen have a maximum traveling speed of 200 length units per time step. Their actual traveling speed can be any value less or equal to the maximum speed in order to permit waiting. In all instances salesmen start their tours at the "home position", an initial position in the center of the operating space.

Our proposed methods are not restricted to the two dimensional space. They can also be applied to the n-dimensional space, $n \in \mathbb{N}$. Furthermore our time-discrete models are not restricted to linear trajectories, it is also possible to handle non-linear trajectories. In order to compare all models with each other we only used linear trajectories and a constant traveling speed of the targets.

The time-expanded model (5) was solved with the MILP solver CPLEX using a start heuristic described in [18]. The CPLEX parameters we used are listed in Table 2 all other parameters were used with their default values.

The time-continuous model (12) could be implemented in CPLEX as a second order cone program due to the assumption of linear trajectories and constant speed of the moving targets. In this case it is a convex optimization problem with constraints that formulate second order cones. We also used CPLEX to solve such problems. We set the parameter for the convergence tolerance for quadratically constrained problems (BarQCPEpComp) to `1e-8`. All parameters we set differently from their default values can be extracted from Table 2.

The time-relaxed models were also solved by CPLEX. We turned off the parameter for the node heuristic (HeurFreq) in order to save running time. The node heuristics would permanently check time free solutions that are usually infeasible and furthermore these solutions would arise repeatedly. For TF-TC we used the same convergence tolerance parameter (BarQCPEpComp) as in its counterpart with time. We refer to Table 2 for the complete list of applied CPLEX parameters. To compare computational running times of all five models we aggregated the running times of each test set by using the geometric mean of the 25 instances in each test set. The values are given in Table 3. The first three columns specify the test set, where the first column (nbt) denotes the number of targets, the second column (nbs) denotes the number of salesmen and the third column describes the precision of the time steps. Column 4 to 8 contain the geometric means of the computational running times of the proposed five models. The missing values in the last row result from a computer memory that was not sufficient for solving

Table 3 Geometric mean of the running times of all proposed models.

instance							
nbt	nbs	precision	TD	TC	TF-TD	TF-TC	TF-TDloc
4	1	1	0.2665	0.0963	0.0472	0.0457	0.0473
4	1	2	8.9292	0.0963	0.0761	0.0457	0.0767
4	1	4	89.9859	0.0963	0.1871	0.0457	0.1939
4	2	1	2.1852	0.1371	0.0553	0.0601	0.0549
4	2	2	40.0755	0.1371	0.0894	0.0601	0.0899
4	2	4	170.9560	0.1371	0.2143	0.0601	0.2195
4	4	1	6.6612	0.3301	0.0934	0.0692	0.0936
4	4	2	68.2254	0.3301	0.1975	0.0692	0.1989
4	4	4	269.6514	0.3301	0.5855	0.0692	0.5905
6	1	1	1.5033	0.1917	0.3848	0.2720	0.4007
6	1	2	41.9211	0.1917	1.0185	0.2720	1.0822
6	1	4	563.2793	0.1917	3.4645	0.2720	3.6497
6	2	1	16.3594	0.9682	0.6293	0.4074	0.6743
6	2	2	147.5708	0.9682	1.5951	0.4074	1.7370
6	2	4	738.3345	0.9682	5.3395	0.4074	5.8295
6	3	1	29.9652	3.1040	0.8601	0.3496	0.9173
6	3	2	177.8745	3.1040	2.1990	0.3496	2.3505
6	3	4	675.9556	3.1040	7.2594	0.3496	7.7560
6	6	1	75.3395	59.9248	10.2121	3.2947	10.2333
6	6	2	308.3957	59.9248	20.7388	3.2947	20.8952
6	6	4	1197.5080	59.9248	59.3193	3.2947	59.8541
8	1	1	3.6934	0.6386	3.0288	1.9126	3.3066
8	1	2	116.3597	0.6386	9.2450	1.9126	10.2484
8	1	4	1460.2306	0.6386	32.2851	1.9126	36.0268
8	2	1	56.5407	23.8182	8.9582	3.5514	9.9173
8	2	2	441.6773	23.8182	25.4224	3.5514	28.5381
8	2	4	2373.6045	23.8182	94.4134	3.5514	106.7477
8	4	1	122.5020	837.2738	92.1096	32.4143	96.5363
8	4	2	510.0498	837.2738	179.6741	32.4143	185.6987
8	4	4	2549.0757	837.2738	–	32.4143	–

some of the used instances. Our computational experiments were carried out on an Apple Mac mini computer running the MacOS 10.9.5 operating system with an Intel Core i7 running at 2.6 GHz on 4 cores, 6 MB L3 cache, and 16 GB 1600 MHz DDR3 RAM. The version of CPLEX we used was 12.5.1.

For the test set with 4 targets and 2 salesmen we put the problem sizes and the non-aggregated running times of all 25 instances in Table 4. The same values for the test set with 6 targets and 6 salesmen can be found in Table 5. In both tables problem sizes are compared for single precision and fourfold precision, where the respective first column (nbv) denotes the number of variables, the respective second column (nbc) is the number of constraints and the third column (nbn) represents the number of non-zeros in the corresponding MILP. Columns 7 to 11 show computational running times of the instances with fourfold precision. The last row in Table 4 and Table 5 contains the aggregated running times for all 25 instances.

Looking at the values of the TD model in column 4 of Table 3, it is obvious that the values strongly correlate with the precision of time steps, but not with the number of used salesmen. The TD model is embedded in a time-expanded graph, which grows drastically in number of arcs when applying instances with a high number of time steps (high precision). Since the TC model is independent of time

Table 4 Problem sizes and exact running times of all proposed models for the 25 instances with target number 4 and number of salesmen 2.

precision 1			precision 4			running times for precision 4				
nbv	nbc	nbn	nbv	nbc	nbn	TD	TC	TF-TD	TF-TC	TF-TDloc
1314	124	5174	16988	454	67636	122.55	0.15	0.06	0.03	0.06
2114	158	8356	28492	590	113582	190.59	0.20	0.80	0.13	0.86
2496	174	9872	34198	654	136356	255.22	0.16	0.29	0.06	0.29
2880	174	11396	40194	654	160302	444.16	0.16	0.94	0.14	0.99
1478	134	5826	19752	494	78684	98.20	0.10	0.17	0.05	0.18
3022	186	11964	42252	702	168526	365.98	0.14	1.00	0.11	1.06
1564	138	6168	20556	510	81884	146.07	0.13	0.12	0.03	0.12
2022	164	7988	27618	614	110082	117.06	0.16	0.49	0.08	0.50
2768	182	10952	38744	686	154514	260.89	0.12	0.40	0.07	0.39
3132	186	12404	43996	702	175496	266.39	0.15	0.08	0.02	0.08
2116	160	8362	28754	598	114616	180.87	0.11	0.09	0.04	0.09
1028	118	4038	13056	430	51948	85.21	0.11	0.07	0.04	0.06
3078	178	12188	43038	670	171666	440.82	0.13	0.53	0.09	0.52
1278	128	5030	16498	470	65682	111.91	0.09	0.05	0.03	0.05
1888	154	7454	25432	566	101350	143.18	0.12	0.68	0.16	0.72
1726	148	6818	22862	550	91108	116.33	0.15	0.69	0.16	0.71
2066	158	8158	28456	590	113422	213.58	0.13	0.14	0.04	0.13
1138	116	4472	14662	422	58350	105.04	0.13	0.13	0.05	0.14
2326	170	9200	31706	638	126414	137.81	0.17	1.11	0.15	1.10
980	114	3848	12592	414	50094	68.34	0.15	0.12	0.05	0.12
1926	152	7604	26280	574	104740	182.70	0.11	0.27	0.10	0.27
2244	154	8868	30954	574	123400	241.26	0.25	0.07	0.04	0.07
3714	202	14718	52928	766	211174	397.71	0.17	1.10	0.10	1.19
1570	148	6190	20710	550	82494	123.66	0.13	0.04	0.02	0.05
1162	124	4570	15178	454	60414	101.22	0.10	0.03	0.03	0.03
					$\sum$	4916.75	3.52	9.45	1.83	9.78

steps, the model is solved once. For reasons of better comparisons we take this one geometric mean value for all three different precision values, as can be seen in Table 3. However, the values of the TC model strongly correlate with the number of salesmen. That means a high number of salesmen makes the problem more difficult. This effect becomes particularly apparent at instances with 8 targets. Running times are increased by a factor of 10^3 when considering salesman number from 1 to 4.

The time-free running times have a similar correlation behavior as their counterparts with time. For a comparison of time and time-free variants we put the running time values in graphical visualizations, see Figure 3 for the TD and TF-TD models and Figure 4 for the TC and TF-TC models. In the first picture of Figure 3 instances with 4 moving targets are considered. In the second picture instances with 6 moving targets and in the third picture instances with 8 moving targets are visualized. In all three pictures it is obvious, that the time-free variant (dashed lines) outperforms the variant with times (solid lines). This might indicate that for instances with 4, 6 and 8 targets the TF-TD variant usually runs faster than the TD variant. This behavior can also be seen when looking at the exact running times in Table 4 and Table 5. The values of column TF-TD are far below the running time values of column TD, even if the computational effort for 6 targets and 6 salesmen is much higher than for 4 targets and 2 salesmen. To solve all 25 instances with 4 targets and 2 salesmen TF-TD only needs about 0.2% of

Table 5 Problem sizes and exact running times of all proposed models for the 25 instances with target number 6 and number of salesmen 6.

precision 1			precision 4			running times for precision 4				
nbv	nbc	nbn	nbv	nbc	nbn	TD	TC	TF-TD	TF-TC	TF-TDloc
10338	516	40986	139008	1920	554610	901.23	57.70	8.73	0.25	8.70
11826	576	46920	160290	2160	639672	1154.05	29.66	23.66	0.69	28.47
17292	684	68694	240480	2592	960054	1705.57	43.53	28.58	0.50	29.77
20076	696	79782	283758	2640	1133016	2350.53	52.05	36.46	0.74	37.21
10902	540	43230	148518	2016	592608	1021.46	72.34	55.20	4.84	55.99
15708	648	62382	218814	2448	873498	1764.26	64.60	334.86	39.94	336.02
9900	528	39240	132816	1968	529848	855.83	47.18	21.85	0.84	21.56
11250	588	44616	152082	2208	606846	625.70	82.27	176.01	17.25	181.08
17340	690	68880	243378	2616	971658	1622.43	71.23	325.67	32.56	253.97
19446	702	77292	273636	2664	1092600	2070.53	77.44	285.50	15.82	291.88
12804	594	50814	175974	2232	702306	1275.71	54.24	281.25	40.80	307.38
9906	522	39264	133044	1944	530772	964.50	69.27	33.69	1.87	32.75
18018	666	71586	253482	2520	1012026	1977.32	39.92	44.58	1.24	47.08
11034	558	43758	150624	2088	601038	1094.28	76.71	198.38	19.86	208.19
11748	564	46602	160152	2112	639072	969.60	94.12	159.67	10.08	162.81
13926	612	55284	190332	2304	759696	1618.97	47.84	380.44	59.42	359.68
12546	588	49782	171036	2208	682584	1022.71	61.94	56.90	4.04	56.90
9372	492	37134	127062	1824	506910	1383.62	45.72	6.17	0.38	6.52
12420	612	49284	168354	2304	671838	842.11	59.39	82.20	5.40	68.30
6936	438	27450	89874	1608	358374	471.37	81.10	47.28	4.00	48.81
10272	546	40722	137934	2040	550326	831.70	73.37	28.09	2.42	33.22
14466	600	57420	200280	2256	799398	1427.90	77.25	7.32	0.18	7.63
19560	726	77730	276570	2760	1104288	1822.87	79.24	355.87	39.57	346.95
12456	600	49422	170994	2256	682428	1007.93	48.09	35.66	1.40	36.02
11766	552	46674	159666	2064	637140	1308.09	46.55	8.29	0.11	7.91
					$\sum$	32090.27	1552.73	3022.32	304.18	2974.80

the time TD needs to solve the instances (see the last row of Table 4). Regarding 6 targets and 6 salesmen TF-TD needs 10% of the TD time (see the last row of Table 5).

In Figure 4 the TC variant is compared with its time-free counterpart TF-TC. Since there is no time discretization and therefore no precision that can be raised to make the problem instances more difficult, we looked at the number of salesmen. A high number of salesmen also results in more difficult problem instances. The three pictures visualize the geometric mean of the running times for the different target numbers 4, 6 and 8. Running times increased over the number of used salesmen, but the time-free variant TF-TC has lower running times in all three diagrams when more than one salesman is considered. Looking at the aggregated running times in Table 4 and Table 5, we have that TF-TC only needs about half as long (50%) as TC to solve all 25 instances with 4 targets and 2 salesmen. For 6 targets and 6 salesmen TF-TC only needs about 20% of the time, that TC takes to solve all 25 instances.

The last column of Table 3 is the time-free variant TF-TD extended by local information about arcs with times from the current branch-and-bound tree. The aim was to exploit this information to create cutting planes in order to prune nodes in the branch-and-bound tree for an advantage in running times. Comparing the values of column 6 and 8, makes clear that the values are very similar, thus there

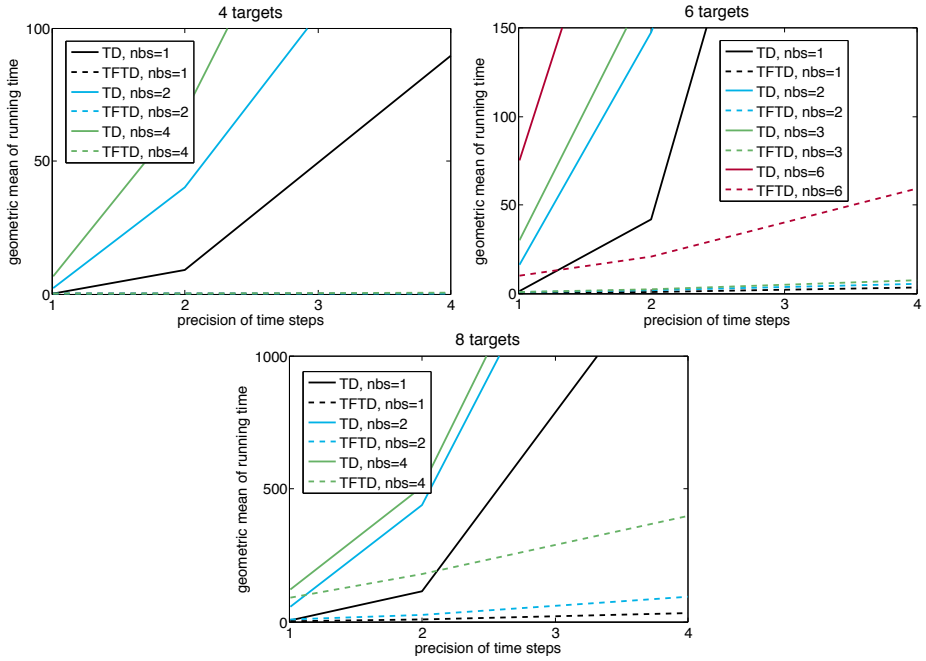

Fig. 3 Visualization of computational running times of the time-discrete model variants for target numbers 4, 6 and 8.

is no advantage visible through applying local information. For a target number of 6 and 8 the values of TF-TDloc are all worse than the values of TF-TD.

Summarizing, for a test set defined by number of targets and number of salesmen we created 25 instances. We solved the instances for three different precision values (discretization) and then took the geometric mean of the 25 running times. Based on these values we observed that the TD variant is precision sensitive and the TC model with continuous times is very sensitive against the number of salesmen. Regarding difficult instances, that are instances with either a high precision value or a high number of salesmen, the time-free variants beat the variants with time. Since the range of application for the discrete model variants (TD, TF-TD) and the continuous model variants (TC, TF-TC) is different, it is not straightforward to make a comparison over all model variants. The discrete variants allow for nonlinear target trajectories and variable speed functions, while the continuous variants are restricted to linear trajectories and constant target speeds. Furthermore the discrete variants apply for problems based on time steps (e.g., time tables), where departure and arrival take place exactly at these time steps and not in between as in our continuous variants. Nonetheless we restricted our problem instances to linear trajectories and constant speed of each target, in order to solve them with all variants for a comparison of computational run-times. Regarding the difficult ones of our instances, the best choice with respect to running times is the TF-TC variant.

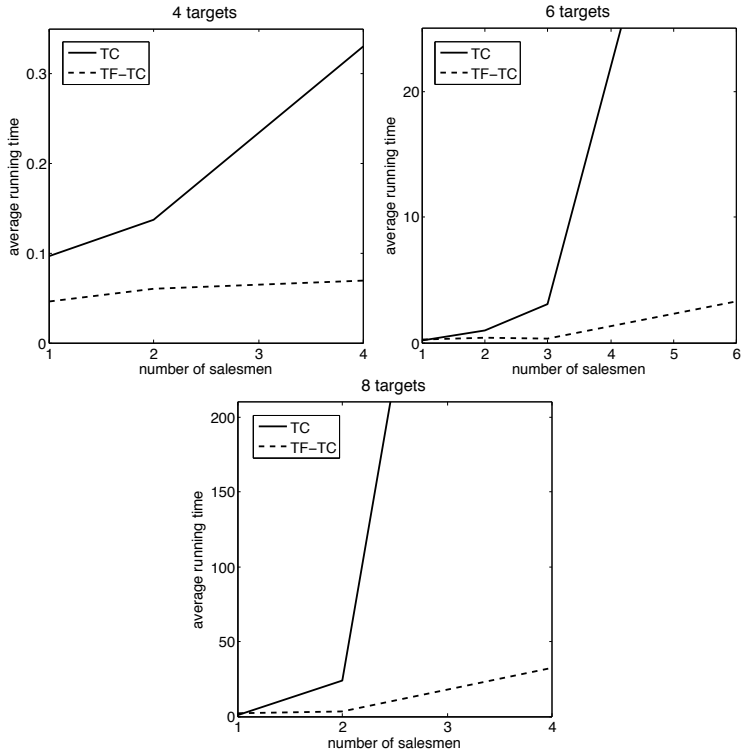

Fig. 4 Visualization of computational running times of the time-continuous model variants for target numbers 4, 6 and 8.

7 Conclusion

We addressed a dynamic variant of the TSP, where several salesmen are looking for their tours through a system with moving targets. The targets enter the system at a certain time and leave the system again, after a while. Intercepting is only possible during this visibility time interval. We presented two model formulation, the first (TD) is based on discrete time steps and the second one (TC) uses continuous times by applying big-M constraints. Due to the different modeling of time both variants have different characteristics inherent. The precision sensitive TD model requires very long computational times especially when the number of time steps is increased. While the TC model is independent of number of time steps, it correlates with the number of salesmen and therefore results in high running times.

Our main research focus is on another modeling variant. We completely relaxed the time aspect and solved the resulting simple time-free program. With some analysis of the time-free solutions and re-integrating parts of the time restrictions, global optimal salesman tours can be constructed. The re-integration of time can be done using discrete and continuous times. While the time-free variants follow a similar correlation behavior to their counterparts with time, for most of the considered instances they result in lower running times. Exploiting local information during branching does not result in an apparent effect for running times.

For continuous times the time-free variant is faster for most of our instances. Thus, having linear trajectories and constant speed TF-TC is the most promising variant with respect to computational running times. In case of nonlinear trajectories TF-TD would be the first choice.

8 Acknowledgements

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

References

1. Ahrens, B.: The tour construction framework for the dynamic travelling salesman problem. In: SoutheastCon 2015, pp. 1–8 (2015). DOI 10.1109/SECON.2015.7132999
2. Codato, G., Fischetti, M.: Combinatorial benders cuts. In: D. Bienstock, G. Nemhauser (eds.) Integer Programming and Combinatorial Optimization, *Lecture Notes in Computer Science*, vol. 3064, pp. 178–195. Springer Berlin Heidelberg (2004). DOI 10.1007/978-3-540-25960-2_14
3. Dijkstra, E.W.: A note on two problems in connexion with graphs. NUMERISCHE MATHEMATIK **1**(1), 269–271 (1959)
4. Ford, L.R., Fulkerson, D.R.: Constructing Maximal Dynamic Flows from Static Flows. Operations Research **6**(3), 419–433 (1958)
5. Fügenschuh, A., Nemhauser, G., Zeng, Y.: Scheduling and routing of fly-in safari planes using a flow-over-flow model. In: M. Jünger, G. Reinelt (eds.) Facets of Combinatorial Optimization, pp. 419–447. Springer Berlin Heidelberg (2013). DOI 10.1007/978-3-642-38189-8_17. URL `http://dx.doi.org/10.1007/978-3-642-38189-8_17`
6. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman and Company, New York (1979)
7. Gendreau, M., Laporte, G., Seguin, R.: Stochastic vehicle routing. European Journal of Operational Research **88**(1), 3–12 (1996)
8. Helvig, C., Robins, G., Zelikovsky, A.: Moving-Target TSP and Related Problems. In: A.P. G. Bilardi G. F. Italiano, G. Pucci (eds.) Proceedings of the European Symposium on Algorithms, *Lecture Notes in Computer Science*, vol. 1461, pp. 453 – 464. Springer Verlag, Berlin (1998)
9. Helvig, C., Robins, G., Zelikovsky, A.: The moving-target traveling salesman problem. Journal of Algorithms **49**(1), 153 – 174 (2003)
10. Jiang, Q., Sarker, R., Abbass, H.: Tracking moving targets and the non-stationary traveling salesman problem. Complexity International **11**, 171 – 179 (2005)
11. Jindal, P., Kumar, A., Kumar, S.: Multiple Target Intercepting Traveling Salesman Problem. International Journal on Computer Science and Technology **2**(2), 327 – 331 (2011)
12. Lawler, E., Lenstra, J., Rinnooy, A., Shmoys, D.: The Traveling Salesman Problem: a Guided Tour of Combinatorial Optimization. John Wiley and Sons, Chichester, New York (1985)
13. Liu, C.H.: The Moving-Target Traveling Salesman Problem with Resupply. Tech. rep., The National Chung Cheng University Library, http://ccur.lib.ccu.edu.tw/handle/987654321/7877 (2013)
14. Lobo, M.S., Vandenberghe, L., Boyd, S., Lebret, H.: Applications of second-order cone programming. Linear Algebra and its Applications **284**(13), 193–228 (1998)
15. Menezes, M.B., Ketzenberg, M., Oliva, R., Metters, R.: Service delivery to moving demand points using mobile servers. International Journal of Production Economics **168**(C), 158–166 (2015)
16. Miller, C.E., Tucker, A.W., Zemlin, R.A.: Integer programming formulation of traveling salesman problems. Journal of the ACM (JACM) **7**(4), 326–329 (1960)
17. Reinelt, G.: The Traveling Salesman: Computational Solutions for TSP Applications. Springer Verlag, Berlin (1994)
18. Stieber, A., Fügenschuh, A., Epp, M., Knapp, M., Rothe, H.: The multiple traveling salesmen problem with moving targets. Optimization Letters **9**(8), 1569–1583 (2014)