Combinatorial Lower Bounds for the Generalized Traveling Salesman Problem with Neighborhoods

Jindřiška Deckerová*, Petr Váňa, Jan Faigl

Faculty of Electrical Engineering,Czech Technical University in Prague, Technická 2, 166 27, Prague, Czech Republic

Abstract

In this paper, we study the Generalized Traveling Salesman Problem with Neighborhoods (GTSPN), a variant of the Traveling Salesman Problem (TSP), where the goal is to find the shortest path visiting each of the given neighborhood sets represented as a set of convex regions. The GTSPN is motivated by the sequencing problem of robotic manipulators, where an operation can be achieved from multiple locations, such as the visual inspection that can be performed from several possible views. The GTSPN formulation allows for exploiting continuous optimization to find the most suitable locations for the inspection, yielding possible solution cost reduction. Moreover, instances with overlapping high-dimensional convex regions further allow modeling neighborhood sets with complex shapes. We propose a novel approach to determine the first lower bounds to the studied GTSPN by employing the Branch-and-Bound (BB) method and the Mixed-Integer Second-Order Cone Programming (MISOCP) model for particular BB subproblems. In addition, the proposed method allows for solving the GTSPN to optima. The developed lower bound determination is further exploited in the empirical evaluation of existing heuristic approaches to the GTSPN and assesses the solution quality using the relative optimality gap. Regarding the presented results, the proposed BB-based approach provides tight lower bounds and solutions with up to 20 % optimality gap for the GTSPN instances with less than 15 neighborhood sets for the given limited computational time. Furthermore, the presented results support that the proposed approach is suitable for solving high-dimensional instances of the GTSPN that can be found in inspection tasks with robotic manipulators.

Keywords: Combinatorial Optimization, Generalized Traveling Salesman Problem with Neighborhoods, Lower bounds

*Corresponding author

Email addresses: deckejin@fel.cvut.cz (Jindřiška Deckerová), vanapet1@fel.cvut.cz (Petr Váňa), faiglj@fel.cvut.cz (Jan Faigl)

1. Introduction

The Traveling Salesman Problem (TSP) is a well-known combinatorial optimization problem with a wide range of approaches proposed over the past decades (Gutin and Punnen, 2007). The TSP stands to determine the most cost-efficient path for visiting a given set of locations; hence, an optimal sequence of visits to the locations is to be found. The sequence can be utilized in robotics as the robot sequencing problem (Suárez-Ruiz et al., 2018) or routing problem with aerial vehicles (Oberlin et al., 2010).

The TSP with Neighborhoods (TSPN) is a practical generalization of the TSP motivated by tasks where it is sufficient to visit a region (neighborhood) instead of the exact location. The additional degrees of freedom allow for saving the total travel cost by determining the suitable locations for visits to the neighborhoods. The problem formulation with neighborhoods is suitable for surveillance missions, data collection planning (Faigl, 2019; Yuan et al., 2007), robotic environment monitoring (Dunbabin and Marques, 2012), and also tasks for robotic manipulators (Gentilini, 2012; Vicencio et al., 2014). The TSPN combines the combinatorial optimization of finding the optimal sequence of visits with continuous optimization to determine the locations of visits to the neighborhoods. The continuous optimization can be addressed by discretizing the neighborhoods, the constant factor approaches (Dumitrescu and Mitchell, 2003), or approximation schemata (Arkin and Hassin, 1994; de Berg et al., 2005; Elbassioni et al., 2009). However, these approaches are parameter sensitive.

The exact approaches to the TSPN, such as (Gentilini et al., 2013) based on the Mixed-Integer Nonlinear Programming (MINLP) formulation, are relatively complex and computationally demanding. Therefore, various heuristics have been proposed to provide time-efficient solvers to the TSPN. Yang et al. (2018) propose a fast double-loop hybrid algorithm based on the particle swarm optimization and genetic algorithm. The unsupervised learning of the self-organizing map has been deployed to address the continuous optimization of determining the suitable locations for visits to the neighborhoods of the non-Euclidean TSPN in the polygonal domain (Faigl et al., 2013). The method is further improved by Faigl (2018) as the Growing Self-Organizing Array (GSOA), a general heuristic for routing problems with neighborhoods employed in the solution of data collection planning (Faigl, 2019) and also the herein addressed Generalized TSPN (GTSPN) (Faigl et al., 2019).

The TSPN becomes the Generalized TSP (GTSP) if the neighborhoods are formed by discrete locations, such as sampling the continuous neighborhoods into finite discrete sets. The GTSP is to determine the most cost-efficient tour that visits all the given sets, where a set is considered visited if the tour visits at least one location (vertex) of the set. The GTSP can be transformed into the TSP using the transformation by Noon and Bean (1993) at the cost of increased instance size. Alternatively, meta-heuristics can be utilized, such as the Ant Colony System (Pintea et al., 2007), genetic algorithms (Silberholz and Golden, 2007), or local search methods (Karapetyan and Gutin, 2012). Besides, the available GLKH (Helsgaun, 2015) and GLNS (Smith and Imeson, 2017) solvers are reported to provide optimal or near-optimal solutions of the GTSP instances.

Although existing combinatorial solvers can be used to solve the GTSP instances, even the optimal solution of such discrete instances of the TSPN would only be approximate solutions of the original TSPN with continuous neighborhoods (Faigl, 2018). Therefore, finding solutions for the TSPN with improved solution costs is an open task. However, neighborhoods can generally be of complex shapes that might be challenging to model and exploit. Therefore, modeling a neighborhood as a set of possibly overlapping regions, each of relatively simple shapes, can describe arbitrarily complex neighborhoods. Yet, it is sufficient to visit at least one such region. Hence, we get the Generalized TSPN (GTSPN) as a combination of the TSPN and GTSP.

In the studied GTSPN, the neighborhood is a neighborhood set represented as a set of convex, possibly overlapping regions; see Figure 1. The formulation of the GTSPN is motivated to combine benefits and overcome limitations of the individual TSPN and GTSP formulations. The GTSPN allows for exploiting clustered or non-convex neighborhoods found in the Multi-goal Path Planning (Alatartsev et al., 2015) to solve inspection tasks with robotic manipulators (Vicencio et al., 2014), which can have 3D or even 7D regions exploiting the manipulator's degrees of freedom.

So far, the GTSPN has been addressed (to the best of the authors' knowledge) only by three approaches. The Hybrid Random-Key Genetic Algorithm (HRKGA) for the TSPN (Gentilini, 2012) has been employed in the solution of the GTSPN by Vicencio et al. (2014) with novel chromosome coding procedure and crossover operators. Each chromosome consists of a vector corresponding to the particular locations within the respective neighborhood, where an index gives the respective neighborhood within the set, and a random fractional number corresponding to the order of visit to the neighborhood set. The populations of chromosomes are created by uniform crossover operator, arithmetic average crossover operator, and mutations. The HRKGA iteratively creates random populations that correspond to the solution of the GTSPN instance. Besides, improvement heuristics are used to enhance the method's performance.

The second approach is based on adapting the GSOA (Faigl, 2018) to the GTSPN (Faigl et al., 2019). The GSOA is an array of nodes iteratively adapted toward the neighborhood sets. The nodes encode the locations of visits to the respective neighborhoods, and the order of nodes in the array defines the sequence of visits to

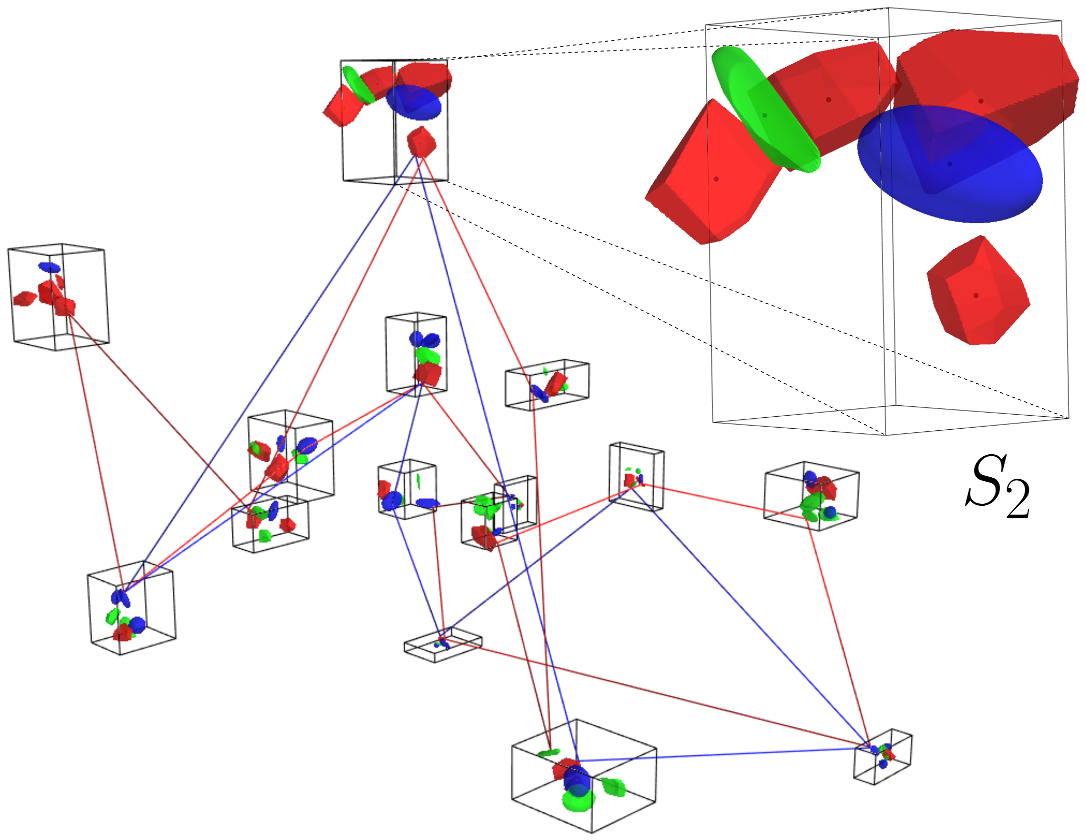


Figure 1: An instance of the GTSPN with 3D regions, the feasible solution depicted in red and an infeasible solution in blue that does not cover all neighborhood sets and, thus, yields lower bound. Besides, a detail of the neighborhood set S_2 that consists of six regions $Q_{2,k}$, $1 \leq k \leq m_2$ is shown.

the neighborhood sets. Similarly to the HRKGA, the GSOA is a randomized method since the nodes are iteratively added into the array in a random order to avoid local minima.

Finally, Faigl et al. (2019) propose a decoupled approach referred to as the Centroid-GTSP$^+$ that is based on the solution of the GTSP using centroids of the neighborhoods to determine the sequence of visits to the neighborhoods and the respective sets. The solution is then improved by the post-processing procedure using local optimization, where locations of visits are locally adjusted within the corresponding neighborhoods to decrease the solution cost.

Regarding the existing related approaches on the lower bounds, the TSPN with disk-shaped neighborhoods is of particular interest. The problem is called the Close Enough TSP (CETSP) in the literature, and it has been introduced by Gulczynski et al. (2006) as a suitable formulation to plan to collect utility measurements remotely. The CETSP is extensively studied in (Mennell, 2009), where the author proposes several heuristics based on the Steiner Zones and transformation to the GTSP, together with a set of benchmark instances. Mennell (2009) further proposes a fast approach to determine the lower bound values on the optimal solutions of the CETSP by simplifying the problem; however, the obtained lower bounds are relatively poor. By solving the CETSP exactly, Behdani and Smith (2014) determine tight lower and upper bounds with a gap up to 1 % on many instances using various Mixed-Integer Programming (MIP) models based on the partitioning of the locations.

The bounds by Behdani and Smith (2014) are improved in (Carrabs et al., 2017b), where the authors utilize a graph reduction algorithm to reduce the size of the problem and apply a new discretization schema to partition the continuous neighborhoods. The approach has been developed by Carrabs et al. (2017a) using heuristic discretization schema combined with the Second-Order Cone Programming (SOCP) to improve the bounds of the solution cost of the CETSP instances. The approach is further developed in (Carrabs et al., 2020) using novel discretization schema and the SOCP in the adaptive approach to estimate the lower and upper bounds. The adaptive approach specifically improves the results on the instances by Mennell (2009) with a high overlap ratio. Besides, Coutinho et al. (2016) propose the exact Branch-and-Bound (BB) method with the SOCP to solve the CETSP. The method provides optimal solutions for all instances in (Behdani and Smith, 2014) and several instances by Mennell (2009). For the remaining instances, the method finds improved lower bounds.

In the presented work, we study the GTSPN motivated by the advancements in determining the lower bound values of the CETSP by Coutinho et al. (2016). Hence, we propose a novel BB-based method as the first solver to determine the lower bounds of the GTSPN instances. The lower bound values established by the proposed BB method are utilized to empirically evaluate the solution quality of the existing solvers for the GTSPN: HRKGA (Vicencio et al., 2014), GSOA, and Centroid-GTSP$^+$ by Faigl et al. (2019). Based on the reported empirical evaluation results, the proposed method represents a viable approach for assessing the quality of found solutions. The main contributions of the paper are considered as follows.

- Novel formulation of the GTSPN as the Mixed-Integer Second-Order Cone Programming (MISOCP) suitable to represent the GTSPN for a given BB subproblem.

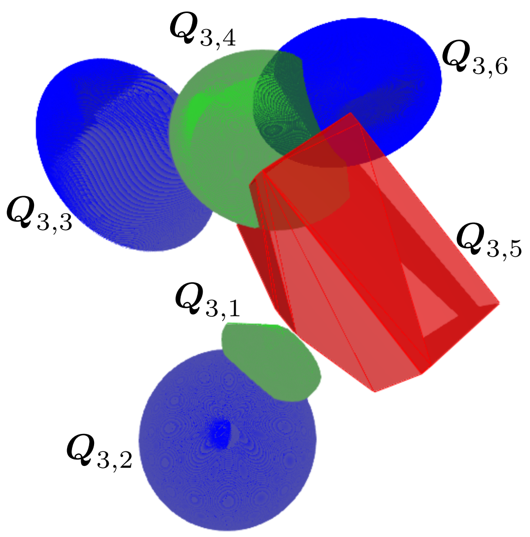
- Novel BB-based method that provides the first lower bounds on the GTSPN instances, including instances with high-dimensional regions motivated by robot manipulator tasks.
- Tight lower bounds on the GTSPN instances with at most 20 neighborhood sets, showing existing heuristics provide solutions with the relative gap up to 20 %.
- Optimal solutions of the evaluated GTSPN instances with up to 12 neighborhood sets.

The rest of the paper is organized as follows. The GTSPN is formally defined in Section 2 with the description of the neighborhood sets. In Section 3, we present an overview of the BB approach employed in determining the lower bound values. The computational results are reported in Section 4. Finally, the conclusion and final remarks are summarized in Section 5.

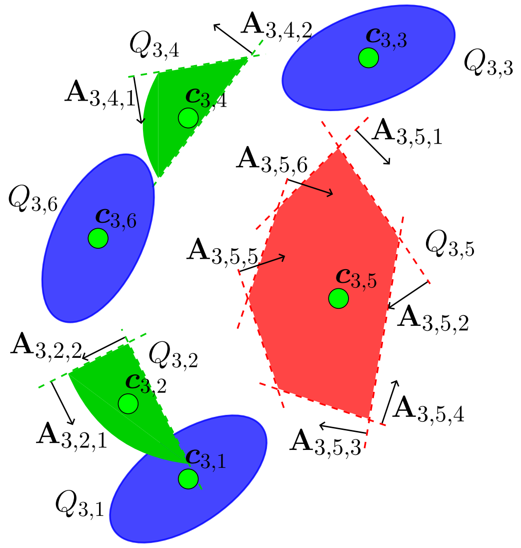
2. Problem Statement

The studied Generalized Traveling Salesman Problem with Neighborhoods (GTSPN) is a variant of the TSPN with the neighborhoods represented as neighborhood sets consisting of convex regions that may overlap. The neighborhood sets are futher referred to as the *region sets*. The GTSPN stands to determine the shortest path visiting each of the region sets, where a region set is considered visited if at least one of its regions is visited. The presented formulation of the GTSPN follows (Faigl et al., 2019) that is slightly modified to fit the proposed BB-based solver.

Let the region sets be $\mathcal{S} = \{S_1, \ldots, S_n\}$, where each d-dimensional region set S_i consists of m_i convex regions $Q_{i,j}$ such that $S_i = \{Q_{i,j} \subset \mathbb{R}^d \,|\, j \in \{1 \ldots, m_i\}\} \subset \mathbb{R}^d$. The task is to determine the shortest path formed by the sequence of visits $\Sigma = (\sigma_1, \ldots, \sigma_n)$, $\sigma_i \in \mathbb{N}$, $1 \leq \sigma_i \leq n$ and $\sigma_i \neq \sigma_j$ for $i \neq j$, to the region sets together with the closed-loop path represented as a sequence $P = (\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n)$ of locations of visits to the sets, $\boldsymbol{p}_i \in \mathbb{R}^d$ and $\boldsymbol{p}_i \in Q_{\sigma_i,j}$ for some $1 \leq j \leq m_{\sigma_i}$, connected by straight line segments. Thus, each $\boldsymbol{p}_i$ is from at least one region of the corresponding region set S_i that can be expressed as (4). Note that only one location of visit is determined for overlapping regions within the set, but each set of regions has exactly one location of visits that might be identical with the location of visit to the other set of regions. The travel cost between two locations of visits $\boldsymbol{p}_i$ and $\boldsymbol{p}_j$ is denoted $\left\|\boldsymbol{p}_i - \boldsymbol{p}_j\right\|$ that stands for the Euclidean distance. The GTSPN is formally defined



(a) Detail of the neighborhood set in the 3D.



(b) Detail of the neighborhood set in the 2D.

Figure 2: Detail of the region set S_3 with the used notation. The set consists of six regions, $m_3 = 6$: ellipsoids are shown in blue, a polyhedron is in red, and hybrid regions are in green.

as Problem 2.1. An example of the GTSPN instance, its solution, and lower bound are depicted in Figure 1.

Problem 2.1 (Generalized Traveling Salesman Problem with Neighborhoods (GTSPN)).

$$\underset{\Sigma,P}{\text{minimize}} \quad \mathcal{L}(\Sigma, P) = \|\boldsymbol{p}_n - \boldsymbol{p}_1\| + \sum_{i=1}^{n-1} \left\|\boldsymbol{p}_i - \boldsymbol{p}_{i+1}\right\| \tag{1}$$

$$\text{subject to } \Sigma = (\sigma_1, \ldots, \sigma_n), \quad \sigma_i \in \{1, \ldots, n\},\ i \neq j : \sigma_i \neq \sigma_j \ \ \forall i, j \in \{1, \ldots, n\} \tag{2}$$

$$P = (\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n), \quad \boldsymbol{p}_i \in \mathbb{R}^d,\ \forall i \in \{1, \ldots, n\} \tag{3}$$

$$\boldsymbol{p}_i \in \bigcup_{j=1}^{m_{\sigma_i}} Q_{\sigma_i, j}\,, \quad \forall i \in \{1, \ldots, n\} \tag{4}$$

2.1. Representation of the Region Sets

Each $S_i \in \mathcal{S}$ from the region sets $\mathcal{S}$ is a finite set of convex, possibly overlapping regions $Q_{i,k}$, where $1 \leq k \leq m_i$. We follow sets representation proposed in (Vicencio et al., 2014), where regions are motivated by real-world robotics applications and approximated using linear and quadratic constraints. The regions are of three types: polyhedron regions, ellipsoid regions, and hybrid regions that combine the polyhedron and ellipsoid regions, see Figure 2. The particular constraints of the regions are as follows.

A region $Q_{i,j}$ is a polyhedron if it is defined by l linear constraints, each defining a half-space. Thus, all points $\boldsymbol{x} \in Q_{i,j}$ for $Q_{i,j} \subset \mathbb{R}^d$ must satisfy

$$\mathbf{A}_{i,j}\,\boldsymbol{x} - \mathbf{b}_{i,j} \leq 0, \quad i \in \{1,\dots,n\},\, j \in \{1,\dots,m_i\}, \tag{5}$$

where $\mathbf{A}_{i,j}$ is a matrix from the space $\mathbb{R}^{l\times d}$, and $\mathbf{b}_{i,j} \in \mathbb{R}^l$ is the corresponding vector defining the particular half-spaces.

A region $Q_{i,j}$ can be represented as an ellipsoid defined by a symmetric positive definite matrix $\mathbf{P}_{i,j} \in \mathbb{R}^{d\times d}$, region's center $\boldsymbol{c}_{i,j}$, and all points $\boldsymbol{x} \in Q_{i,j}$ for $Q_{i,j} \subset \mathbb{R}^d$ satisfy

$$(\boldsymbol{x} - \boldsymbol{c}_{i,j})^T\,\mathbf{P}_{i,j}\,(\boldsymbol{x} - \boldsymbol{c}_{i,j}) \leq 1, \quad i \in \{1,\dots,n\},\, j \in \{1,\dots,m_i\}. \tag{6}$$

A hybrid region $Q_{i,j}$ is defined as an ellipsoid with the center $\boldsymbol{c}_{i,j}$ clipped by $l = 6$ half-spaces defined by the matrix $\mathbf{A}_{i,j} \in \mathbb{R}^{l\times d}$, and the corresponding vector $\mathbf{b}_{i,j} \in \mathbb{R}^l$, thus both (5) and (6) hold for all points $\boldsymbol{x} \in Q_{i,j}$ for $Q_{i,j} \subset \mathbb{R}^d$.

We assume that the center $\boldsymbol{c}_{i,j}$ of each region is inside the particular region, which is also assured for all instances of the GTSPN benchmark (Vicencio et al., 2014).

3. Proposed Lower Bound Determination based on Branch-and-Bound Method

We propose to determine the lower bound values on the optimal solutions of GTSPN instances by the Branch-and-Bound (BB) method. BB is a combinatorial approach used to address several NP-hard problems (Land and Doig, 1960) that efficiently and systematically searches a solution space of the problem. The approach relies on the decomposition of the problem into smaller subproblems, corresponding to a solution space subset, and computation of the subproblems' solutions to estimate lower and upper bounds on the problem solution.

The proposed approach consists of two parts: the *modeling of subproblems* and *searching the solution space*. In general, the BB approach defines the subproblems in the modeling part (Clausen, 1999), and in the search part, the subproblems are organized in a tree structure (each node representing subproblem) according to a *branching rule*. The tree is searched according to an *ordering function* and pruned according to a *bounding rule*. The proposed method follows the idea of Coutinho

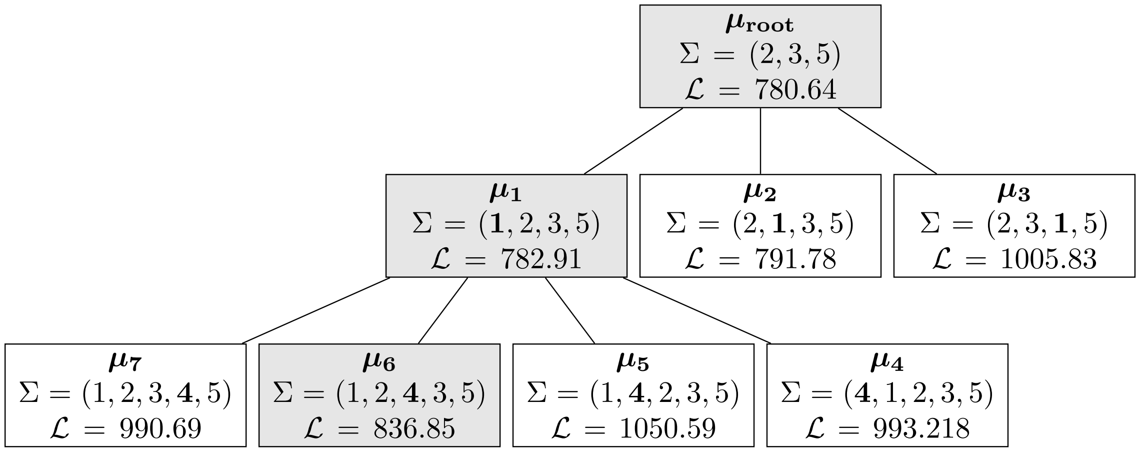


Figure 3: A solution space of the proposed BB-based lower bound determination visualized as a tree of nodes. Each node represents a sequence Σ denoting an ordered set of labels of region sets included in the partial solution encoded in the particular node. In the shown example, the root node μ_{root} is branched by inserting the set S_1 into Σ. Hence three new nodes μ_1, μ_2, and μ_3 (label of the inserted S_1 is in bold) are created. Then, the node μ_1 is branched since its solution value $\mathcal{L}$ is currently the lowest, and four new nodes μ_4, μ_5, μ_6, and μ_7 are created by inserting S_4 at four possible positions in the sequence. The solutions of the highlighted nodes in gray are visualized in Figure 4.

et al. (2016) for the BB addressing the CETSP, where the SOCP is utilized to formulate the CETSP subproblems as optimization models. In the proposed method, we utilize the Mixed-Integer Second-Order Cone Programming (MISOCP) to solve subproblems. Furthermore, we utilize an estimation technique to quickly establish a solution and decrease the computational demands of the method. The proposed method is overviewed in Algorithm 1, and an example of the solution space is depicted in Figure 3. The modeling and search parts with branching and bounding rules are described in the following paragraphs. The MISOCP and the estimation are described in Section 3.1 and Section 3.2, respectively.

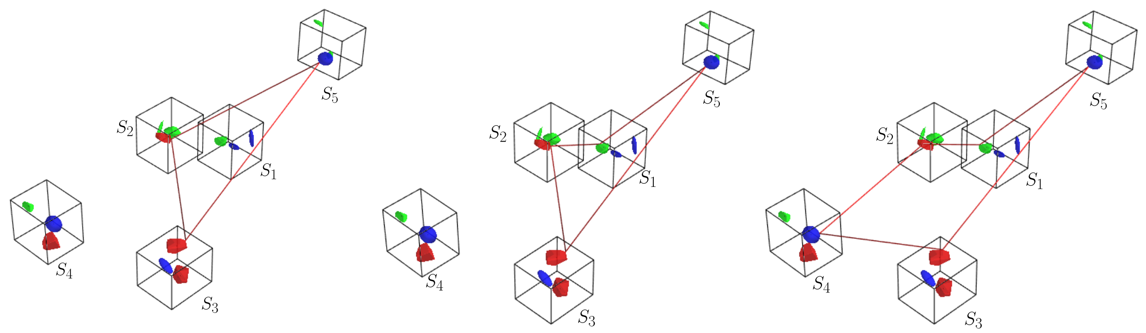
Algorithm 1: The BnB-based solver for the GTSPN

```
Input: $\mathcal{S} = \{S_1, \dots, S_n\}$ – A set of region sets.
Parameters : $t_{\max}$ – Maximal dedicated computational time.
Output: $\mathcal{LB}$ – Lower bound value on the optimal solution. $\mathcal{UB}$ – Upper bound value on the
        optimal solution.

► Initialization
1  $\mu_{\text{root}}.\Sigma \leftarrow$ select_root($\mathcal{S}$)   // Select three sets forming the longest path.
2  $(\mu_{\text{root}}.\Sigma, \mu_{\text{root}}.P) \leftarrow$ compute_partial_solution($\mu_{\text{root}}.\Sigma$)   // Using MISOCP.
3  $(\mu_{\text{root}}.\Sigma_{\text{f}}, \mu_{\text{root}}.P_{\text{f}}) \leftarrow$ compute_feasible_solution($\mathcal{S}, \mu_{\text{root}}.\Sigma, \mu_{\text{root}}.P$)   // Using Algorithm 2.
4  $\mathcal{LB} \leftarrow \mathcal{L}(\mu_{\text{root}}.\Sigma, \mu_{\text{root}}.P)$   // Set the partial solution value as the lower bound.
5  $\mathcal{UB} \leftarrow \mathcal{L}(\mu_{\text{root}}.\Sigma_{\text{f}}, \mu_{\text{root}}.P_{\text{f}})$   // Set the feasible solution value as the upper bound.
6  $\mu_{\text{root}}$.exact $\leftarrow$ true
7  $\mathcal{O} \leftarrow \{\mu_{\text{root}}\}$   // Insert the root node to the open list.

► Solution Loop
8  while $t_{\max}$ is not reached do
9      $\mu \leftarrow$ dequeue($\mathcal{O}$)
10     if not $\mu$.exact then
           // $\mu$ is dequeued for the second time
11         $(\mu.\Sigma, \mu.P) \leftarrow$ compute_partial_solution($\mu.\Sigma$)   // Using the MISOCP.
12         $(\mu.\Sigma_{\text{f}}, \mu.P_{\text{f}}) \leftarrow$ compute_feasible_solution($\mathcal{S}, \mu.\Sigma, \mu.P$)
13         $\mathcal{UB} \leftarrow \min(\mathcal{UB}, \mathcal{L}(\mu.\Sigma_{\text{f}}, \mu.P_{\text{f}}))$   // Update the upper bound of the original problem.
14         $\mu$.exact $\leftarrow$ true
15         $\mathcal{O} \leftarrow$ push($\mathcal{O}, \mu$)
16         Continue
17     $\mathcal{LB} \leftarrow \max(\mathcal{LB}, \mathcal{L}(\mu.\Sigma, \mu.P))$   // Update the lower bound of the original problem.
18     $\beta \leftarrow$ compute_not_covered($\mu$)   // Determine sets not covered by the partial solution.
19     if $\beta \neq \emptyset$ then
20         $S^* \leftarrow \arg\max_{S_j \in \beta} \|S_j - \mu.P\|$   ▷Branching rule
21         for $i$ in $\{1, \dots, k\}$ do
22             $\mu_{\text{child}}.\Sigma \leftarrow (\sigma_1, \dots, \sigma_i, S^*_{\text{idx}}, \sigma_{i+1}, \dots, \sigma_k)$   ▷Branching
23             $(\mu_{\text{child}}.\Sigma, \mu_{\text{child}}.P') \leftarrow$ estimate_partial_solution($\mu_{\text{child}}.\Sigma$)   // Using SOCP-based estimation.
24             $\mu_{\text{child}}$.exact $\leftarrow$ false
25             $\mathcal{O} \leftarrow$ push($\mathcal{O}, \mu_{\text{child}}$)
26     $\mathcal{O} \leftarrow$ filter($\mathcal{O}, \forall \mu : \mathcal{L}(\mu.\Sigma, \mu.P) \leq \mathcal{UB}$)   ▷Bounding rule
27 return $\mathcal{LB}$,$\mathcal{UB}$
```

Modeling part: Each subproblem (*partial problem*) μ for the given Problem 2.1 is modeled by a subset of $k \leq n$ ordered region sets $\mathcal{S}' \subset \mathcal{S}$. The set $\mathcal{S}'$ corresponds to region sets that must be visited and are ordered according to the particular sequence $\mu.\Sigma$. The partial solution is then determined as a path $\mu.P$ connecting the locations of visits in the order given by $\mu.\Sigma$. The path $\mu.P$ is obtained as a solution of the MISOCP, which is detailed in Section 3.1. However, solving the MISOCP can be computationally demanding. Moreover, obtained solutions can be quickly discarded by the bounding rule. Therefore, we first estimate the partial solution $\mu.P'$ using the SOCP model; see Section 3.2, and the exact solution $\mu.P$ is determined once the partial problem is proved to be a suitable lower bound candidate. Furthermore, we can compute the correspoding feasible solution $(\mu.\Sigma_{\mathrm{f}}, \mu.P_{\mathrm{f}})$ to the partial solution $(\mu.\Sigma, \mu.P)$ as described in Algorithm 2. Since $k \leq n$, we can consider the partial solution value as a lower bound value on the original problem; so, $\mathcal{L}(\mathcal{S}') \leq \mathcal{L}(\mathcal{S})$. The corresponding feasible solution can then be considered as an upper bound on the original problem. Note that the used notation of particular solution types is

- $(\mu.\Sigma, \mu.P)$ - An exact partial solution obtained using the MISOCP model;
- $(\mu.\Sigma, \mu.P')$ - An estimated partial solution obtained using the SOCP model;
- $(\mu.\Sigma_{\mathrm{f}}, \mu.P_{\mathrm{f}})$ - A feasible solution obtained by Algorithm 2.

Besides, we use the notation of $\mathcal{LB}$ and $\mathcal{UB}$ to denote the lower and upper bounds of the original problem (not the subproblem), respectively.



(a) Solution of μ_{root}. (b) Solution of μ_1. (c) Solution of μ_6.

Figure 4: Solutions obtained by the proposed BB-based solver in particular nodes of the solution tree depicted in Figure 3.

Search part: BB iteratively searches the solution space denoted $\mathcal{M}$ that consists of nodes μ containing partial problems (subproblems) and their solutions. The solution space $\mathcal{M}$ is searched using a priority queue called the *open list* $\mathcal{O}$ sorted by *the ordering rule*. In the proposed method, the ordering rule is to sort the partial solutions values in the ascending order, since the idea is to process the nodes with the lowest partial solution values first. The root node μ_{root} of $\mathcal{M}$ is determined with the partial problem containing three region sets $\mathcal{S}' \subset \mathcal{S}$ such that $\mathcal{L}(\mu_{\mathrm{root}}.\Sigma, \mu_{\mathrm{root}}.P)$ is maximal over all possible combinations to maximize the lower bound value. The list $\mathcal{O}$ is initialized with μ_{root}, and the method iteratively dequeues $\mathcal{O}$ and processes the dequeued nodes as follows.

For each dequeued node μ, at first, the lower bound on the original problem might be updated according to the values associated with μ (Line 17 of Algorithm 1). Then, the set β is determined such that β consists of all region sets S_i that are not covered by the partial solution $(\mu.\Sigma, \mu.P)$ associated with the node μ (Line 18 of Algorithm 1). Then, we select a region set S^* from β to be used for branching of μ according to the *branching rule*. Since the objective is to find a solution covering all sets, S^* is determined to maximize the prolongation of the current path $\mu.P$, and thus likely cover more sets that are in $\mu.\Sigma$. Therefore, S^* is determined as the farthest region set from the path $\mu.P$ satisfying

$$S^* = \underset{S_j \in \beta}{\arg\max} \left\| S_j - \mu.P \right\|, \tag{7}$$

where $\|S_j - \mu.P\|$ denotes the distance of S_j to the partial solution $(\mu.\Sigma, \mu.P)$, where the path $\mu.P$ is a sequence of straight line segments. The distance is the length of the shortest straight line segment from a point $\boldsymbol{p}_j \in S_j$ to the point on the path $\mu.P$ determined iteratively for each segment of $\mu.P$ and choosing the shortest one.

The region set S^*, selected by the branching rule (7), is used to create possible child nodes of the node μ such that the index S^*_{idx} of the set S^* is inserted into the sequence at every possible position in $\mu.\Sigma$. Thus, for the k indexes long sequence $\mu.\Sigma$, k new child nodes μ_{child} of μ are constructed (Line 22 of Algorithm 1).[1]

Algorithm 2: The feasible solution of the GTSPN for given Σ

```
Input: $\mathcal{S} = \{S1, \dots, S_n\}$ – A set of region sets.
Input: $\mu.\Sigma = (\sigma_1, \dots, \sigma_{\overline{n}})$ – A given sequence.
Input: $\mu.P = \{\boldsymbol{p}_1, \dots, \boldsymbol{p}_{\overline{n}}\}$ – A path covering $\overline{n}$ sets.
Output: $(\Sigma_f, P_f)$ – A feasible solution.

1 $\Sigma \leftarrow \mu.\Sigma$ and $P \leftarrow \mu.P$          // Initialize by the given $\mu.\Sigma$ and $\mu.P$.
2 while solution is updated do
3     for $S \in \mathcal{S}[\mu.\Sigma]$ do
4         $\beta \leftarrow$ get_not_covered_sets$(\mathcal{S}, \Sigma, P)$
5         $\boldsymbol{p}, i \leftarrow$ get_closest_point_and_segment$(S, \Sigma, P)$
6         $\Sigma, P \leftarrow$ insert$(\boldsymbol{p}, i, \Sigma, P)$
7 $(\Sigma_f, P_f) \leftarrow (\Sigma, P)$          // Denote the solution found as the feasible solution.
8 return $(\Sigma_f, P_f)$
```

For each partial problem of μ_{child}, the partial solution $\mu_{\text{child}}.P'$ is be estimated (Line 23 of Algorithm 1), and the node is inserted into $\mathcal{O}$. The exact partial solution $\mu_{\text{child}}.P$ is determined once the node is dequeued from $\mathcal{O}$ (Line 11 of Algorithm 1) for the second time. Together with the exact solution $\mu_{\text{child}}.P$, the corresponding feasible solution $(\mu_{\text{child}}.\Sigma_f, \mu_{\text{child}}.P_f)$ is determined using Algorithm 2, and the global

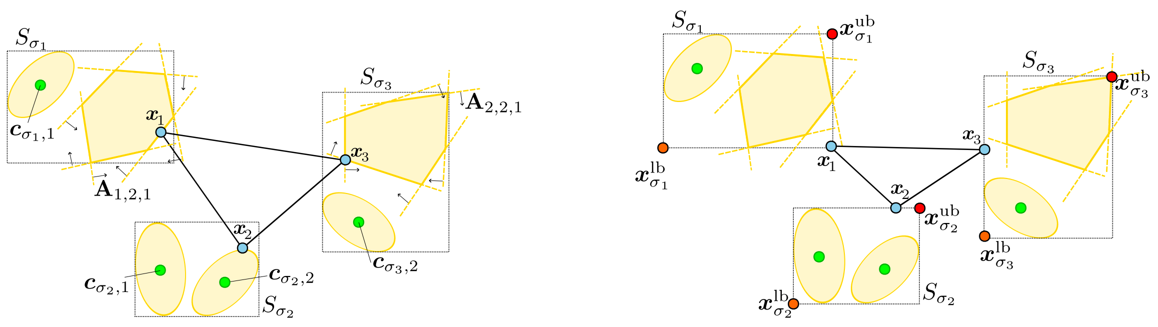
[1]Note that *branching* is based on the sequences, not assigning the values to the variables.

upper bound $\mathcal{UB}$ can be updated (Line 13 of Algorithm 1). The open list $\mathcal{O}$ is filtered using the *bounding rule* that is to remove the nodes with $\mathcal{L}(\mu.\Sigma, \mu.P') > \mathcal{UB}$.

The BB-based search is terminated if the optimal solution is found and $\mathcal{O}$ becomes empty, or predefined maximal computational time $t_{\max}$ is reached.

3.1. Mixed-Integer Second-Order Cone Programming Model

The utilized optimization model consists of a linear objective function, linear constraints, convex quadratic constraints, a second-order cone constraint, and binary variables; hence, we refer to it as the Mixed-Integer Second-Order Cone Program (MISOCP). The MISOCP described in Model 3.1 follows the formulation of Problem 2.1 to determine P for the partial problem given by a fixed sequence Σ. Model 3.1 can be solved optimally using an optimization solver, such as CPLEX 12.9.



S_{σ_1}
$\boldsymbol{c}_{\sigma_1,1}$
$\boldsymbol{x}_1$
$\mathbf{A}_{1,2,1}$
S_{σ_3}
$\mathbf{A}_{2,2,1}$
$\boldsymbol{x}_3$
$\boldsymbol{x}_2$
$\boldsymbol{c}_{\sigma_2,1}$
$\boldsymbol{c}_{\sigma_2,2}$
S_{σ_2}
$\boldsymbol{c}_{\sigma_3,2}$
S_{σ_1}
$\boldsymbol{x}^{\mathrm{ub}}_{\sigma_1}$
$\boldsymbol{x}^{\mathrm{lb}}_{\sigma_1}$
S_{σ_3}
$\boldsymbol{x}^{\mathrm{ub}}_{\sigma_3}$
$\boldsymbol{x}_3$
$\boldsymbol{x}_1$
$\boldsymbol{x}_2$
$\boldsymbol{x}^{\mathrm{ub}}_{\sigma_2}$
$\boldsymbol{x}^{\mathrm{lb}}_{\sigma_3}$
$\boldsymbol{x}^{\mathrm{lb}}_{\sigma_2}$
S_{σ_2}

(a) The MISOCP model.
(b) The SOCP model.

Figure 5: An instance of the GTSPN with $\Sigma = (1, 2, 3)$ and the depiction of the continuous variables $\boldsymbol{x}$ that correspond to the locations of visits P and constants utilized in the determination of the exact solution with Model 3.1 (Figure 5a) and estimated solution with Model 3.2 (Figure 5b).

Model 3.1 (Mixed-Integer Second-Order Cone Program (MISOCP))**.**

$$\underset{\boldsymbol{x}\in\mathbb{R}^{k\times d}}{\text{minimize}}\; f_1 + f_2 + \ldots + f_k \tag{8}$$

$$\text{subject to } 0 \geq (\boldsymbol{x}_i - \boldsymbol{x}_{i+1})^T(\boldsymbol{x}_i - \boldsymbol{x}_{i+1}) - f_i^2 \qquad \forall i \in \{1,\ldots,k\} \tag{9}$$

$$\mathbf{A}_{i,j}\,\boldsymbol{x}_i - b_{i,j} \leq \mathbf{M}_{i,j}\,z_{i,j} \qquad \forall i \in \{1,\ldots,k\},\ \forall j \in \{1,\ldots,m_i\} \tag{10}$$

$$(\boldsymbol{x}_i - \boldsymbol{c}_{i,j})^T\,\mathbf{P}_{i,j}\,(\boldsymbol{x}_i - \boldsymbol{c}_{i,j}) \leq 1 + \mathbf{M}_{i,j}\,z_{i,j} \qquad \forall i \in \{1,\ldots,k\},\ \forall j \in \{1,\ldots,m_i\} \tag{11}$$

$$z_{i,1} + z_{i,2} + \ldots + z_{i,m_i} = m_i - 1 \qquad \forall i \in \{1,\ldots,k\} \tag{12}$$

$$z_{i,j} \in \{0,1\} \qquad \forall i \in \{1,\ldots,k\},\ \forall j \in \{1,\ldots,m_i\} \tag{13}$$

$$f_i \geq 0 \qquad \forall i \in \{1,\ldots,k\} \tag{14}$$

The optimization of Model 3.1 is to determine a set of $k \leq n$ locations of visits $\boldsymbol{p}_i$, such that the path formed by the locations has the minimal length, and each location is within the corresponding region set.

Model 3.1 consists of

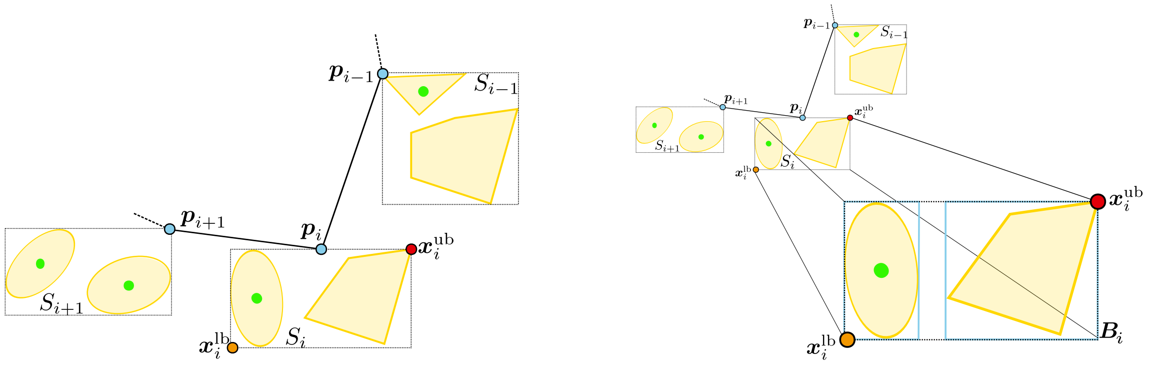
- *Decision variables* $\boldsymbol{x} \in \mathbb{R}^{k \times d}$, where each variable $\boldsymbol{x}_i \in \mathbb{R}^d$ corresponds to locations of visit $\boldsymbol{p}_i \in P$ ($\boldsymbol{x}_i = \boldsymbol{p}_i$);
- *Auxiliary variables* $\boldsymbol{f} \in \mathbb{R}^k$, such as $f_i \geq 0$; and
- *Binary variables* $\boldsymbol{z} \in \mathbb{Z}_2^{k \times m_i}$, where $\mathbb{Z}_2 = \{0, 1\}$, that are used for the disjunctive constraints describing the selection of the particular region based on the *big-M* approach (Griva et al., 2008) with $\mathbf{M}$, which is a matrix $\mathbf{M} \in \mathbb{R}^{k \times m_i}$. If the j-th region $Q_{i,j}$ of the set S_i is to be included in the solution, then $z_{i,j} = 0$ and remaining variables $z_{i,\cdot}$ are set to 1 to disable constraints (10) and (11).

The model further contains the linear objective function (8) that is minimized. The objective expresses minimization of the Euclidean distances between each two consecutive variables $\boldsymbol{x}_i$ and $\boldsymbol{x}_{i+1}$, i.e., $\|\boldsymbol{x}_i - \boldsymbol{x}_{i+1}\|$. The Euclidean norms are defined by quadratic cones (Ben-Tal and Nemirovski, 2001) and can be cast as second-order cones as $f_i \geq \|\boldsymbol{x}_i - \boldsymbol{x}_{i+1}\|$. Furthermore, it can be alternatively expressed as $0 \geq \|\boldsymbol{x}_i - \boldsymbol{x}_{i+1}\|^2 - f_i^2$, with $f_i \geq 0$ (14), and the second order cone can be thus be deduced to $0 \geq (\boldsymbol{x}_i - \boldsymbol{x}_{i+1})^T(\boldsymbol{x}_i - \boldsymbol{x}_{i+1}) - f_i^2$ (9). Indeed, we need to determine at least one region $Q_{i,j} \in S_i$ of m_i regions of the set S_i to be visited. We employ the mixed-integer optimization with the *big-M* technique to model the disjunctive constraints representing the particular regions, where the polyhedron is defined by $\mathbf{A}_{i,j}$ and $b_{i,j}$ and the ellipsoid is defined by $\mathbf{P}_{i,j}$ and $\boldsymbol{c}_{i,j}$ The binary variables $\boldsymbol{z}$ together with the large enough variables $\mathbf{M} \in \mathbb{R}^{k \times m_i}$ ensure only one of the constraints (10) or (11) corresponding to the particular region is in effect. Note that a constraint is in effect if corresponding z_i is set to 0 and others to 1; hence, the constraint (12) is employed.

The variable $\mathbf{M}$ is a penalty constant and needs to be large enough so that all infeasible solutions of Model 3.1 are discarded. Therefore, $\mathbf{M}_{i,j}$ is determined for each region $Q_{i,j}$ of each set $S_i \in \mathcal{S}$ as the maximal distance between the region and its bounding box $B_{i,j}$. A quadratic model with (5) and (6) (as the particular constraints) is used to determine the bounding box $B_{i,j}$ of the region $Q_{i,j}$, see Figure 6b. Depending on the coordinate, minimization (or maximization) of the bounding box coordinate x_l, $l \in \{1, \ldots, d\}$ is performed. In particular, the bounding box $B_{i,j}$ is determined from the point $\boldsymbol{x}^{\mathrm{lb}} \in \mathbb{R}^d$ with the minimum coordinate values and $\boldsymbol{x}^{\mathrm{ub}} \in \mathbb{R}^d$ with the maximal coordinate values as the lower left and upper right point of the bounding box, respectively.

3.2. *Estimation of the Partial Solution*

The estimation of the partial solutions is utilized to decrease the computational burden of the BB-based search by avoiding the costly determination of the exact solution of the partial problem using the MISOCP. Having a partial problem consisting of a subset of k region sets $\mathcal{S}' \subset \mathcal{S}$, given by a sequence $\Sigma = (\sigma_1, \ldots, \sigma_k)$, the cost of the partial solution defined as the partial path $P' = (\boldsymbol{p}_1, \ldots, \boldsymbol{p}_k)$ is estimated using a computationally efficient approach based on the bounding boxes of the region sets, where each region set S_i is replaced by a tight bounding box B_i around the set, see Figure 6.



(a) Estimation of the partial solution.

(b) Detail of bounding box B_i.

Figure 6: Principle of estimating the partial solution of Problem 2.1 using Model 3.2 with bounding boxes. The bounding box B_i of the region set S_i is determined as the point $\boldsymbol{x}_i^{\mathrm{lb}}$ with the minimal values of the coordinates (the lower left point) from all the bounding boxes of the regions $Q_{i,j}$, $j \in \{1, \ldots, m_i\}$, and $\boldsymbol{x}_i^{\mathrm{ub}}$ with the maximal values of the coordinates (the upper right point) from all region bounding boxes (depicted as blue boxes), i.e., $\boldsymbol{x}_i^{\mathrm{lb}} = \min_{j=1,\ldots,m_i} \boldsymbol{x}_{i,j}^{\mathrm{lb}}$ and $\boldsymbol{x}_i^{\mathrm{ub}} a = \max_{j=1,\ldots,m_i} \boldsymbol{x}_{i,j}^{\mathrm{ub}}$.

Model 3.2 (Second-Order Cone Programming (SOCP) for Solution Estimation)**.**

$$\underset{\boldsymbol{x} \in \mathbb{R}^{k \times d}}{\text{minimize}} \; f_1 + f_2 + \ldots + f_k \tag{15}$$

$$\text{subject to } f_i^2 \geq (\boldsymbol{x}_{i+1} - \boldsymbol{x}_i)^T (\boldsymbol{x}_{i+1} - \boldsymbol{x}_i) \qquad \forall i \in \{1, \ldots, k\} \tag{16}$$

$$\boldsymbol{x}_i^{\mathrm{lb}} \leq \boldsymbol{x}_i \leq \boldsymbol{x}_i^{\mathrm{ub}} \qquad \forall i \in \{1, \ldots, k\} \tag{17}$$

$$f_i \geq 0 \qquad \forall i \in \{1, \ldots, k\} \tag{18}$$

For the estimation of the partial solution, the partial problem is formulated as a problem where we have a given sequence of visits together with a set of bounding boxes corresponding to $\mathcal{S}'$. The task is to find the shortest path visiting each bounding box. The utilized optimization model uses a linear objective function and linear and second-order cone constraints. Therefore, the model is referred to as the Second-Order Cone Program (SOCP) and is described in Model 3.2. The mathematical solver CPLEX 12.9 is employed to solve the model. Note that the found path formed by P' of the partial solution (Σ, P') is not a feasible solution to the partial problem, but its estimation because of relaxing the constraints (10) and (11), which allows the variable $\boldsymbol{x}_i$ be within the bounding box and not necessarily in the regions.

Model 3.2, similarly to Model 3.1, is to determine a set of $k \leq n$ points $\boldsymbol{x}_i$ such that the path formed by $\boldsymbol{x}_i$ is of the minimal length and each point is within the corresponding bounding box. Thus, the objective is to minimize the Euclidean distance between two consecutive points. The objective function (15) can be represented in the same manner as in Model 3.1, as a summation over the variables $\boldsymbol{f}$ subject to $f_i \geq (\boldsymbol{x}_{i+1} - \boldsymbol{x}_i)^T(\boldsymbol{x}_{i+1} - \boldsymbol{x}_i)$ (16). Besides, we express that the point $\boldsymbol{x}_i$ is within the bounding box B_i using the linear constraint (17), where the points $\boldsymbol{x}_i^{\mathrm{lb}}$ and $\boldsymbol{x}_i^{\mathrm{ub}}$ are the minimal and maximal coordinate values, respectively, as depicted in Figure 6.

4. Computational Results

The proposed BB-based solver provides the lower and upper bounds on the optimal solution values of the GTSPN instances. Hence, the existing approaches to the GTSPN can be computationally evaluated using the obtained bounds. The presented evaluation results are obtained for 35 benchmark instances proposed by Vicencio et al. (2014) and 410 additional instances[2] derived from (Vicencio, 2014) to show the scalability of the proposed method. The evaluated instances includes 420 random instances in the 3D and 25 randomly generated instances in the 7D. Polyhedra regions are defined by $l = 12$ half-spaces for the 3D instances and $l = 20$ half-spaces for polyhedra regions in the 7D. The number of half-spaces for the hybrid regions is $l = 6$ for both 3D and 7D instances. The benchmark instances vary in the number of region sets $n \in \{30, 35, 40, 45, 50\}$, where each region set consists of $m_i = 6$ regions of the shape types of the polyhedron, ellipsoid, and hybrid. The number of region sets n and the number of the regions m_i per each set, together with the dimension d, are encoded in the instance name as $d\mathtt{D}\_n\_m_i\_v$, where $v \in \{a, b, c, d, e, f\}$ denotes a particular instance variant. The derived instances are created from the benchmark instances by randomly selecting k region sets from the instances $d\mathtt{D}\_n\_6\_v$ for $k \in \{5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 20, 25\}$ for the 3D instances, and $k \in \{5, 10, 15, 20\}$ for the 7D instances. The name of the derived

[2]Derived instances area available at `https://github.com/comrob/gtspn-bb`.

instances encodes the parameters of the benchmark instance as $d\mathtt{D}\_n\_k\_m_i\_v$.

The performance of the proposed solver is compared with the existing hybrid method denoted as the HRKGA (Vicencio et al., 2014), the unsupervised learning-based method denoted as the GSOA and the decoupled approach denoted as the Centroid-GTSP$^+$ (Faigl et al., 2019). The HRKGA results (Vicencio et al., 2014) are reported to be obtained using the Uniform Crossover Operator and 30 trials executed on the `Intel Xeon CPU @3.20 GHz`. The results of the GSOA and the decoupled Centroid-GTSP$^+$ approaches are reported to be obtained using a single core of the `Intel i7-6700K` and both executed for 50 trials (Faigl et al., 2019). The real computational requirements are reported as the mean computational times denoted $\overline{\mathrm{T}_{\mathrm{CPU}}}$.[3]

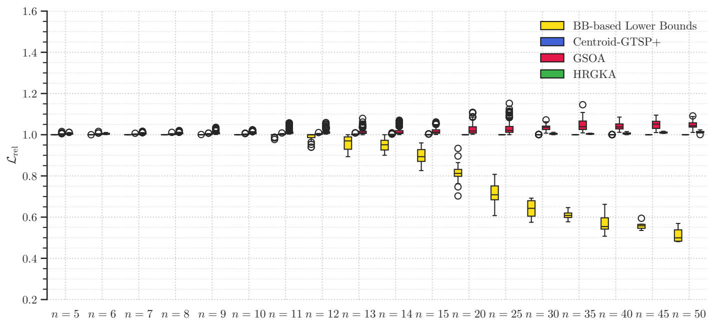
The proposed BB-based solver is implemented in the `Julia` programming language utilizing the `JuMP` modeling language (Dunning et al., 2017) with the CPLEX 12.9 optimization solver to solve the SOCP and MISOCP optimizations models. Because the BB-based solver is deterministic, it is executed for a single trial using a computational environment consisting of 24 cores of two `Intel Xeon Scalable Gold 6146` processors. The initial upper bound value $\mathcal{UB}$ set as the best existing solution selected from the results reported in (Vicencio et al., 2014; Faigl et al., 2019). The maximal running time of the BB-based solver (as of Algorithm 1) $t_{\max}$ is set to 10 h, which is considered computationally tractable. However, a single call of the `CPLEX` is allowed to exceed the limit because an optimal solution of the particular problem is requested to determine the lower bound. Therefore, the real required computational time of the reported found solutions is denoted $\mathrm{T}^{\mathrm{BB}}_{\mathrm{CPU}}$.

The numerical evaluation is based on measuring two performance indicators: the optimality of solutions using the proposed lower bounds method and the relative length. The solution optimality %GAP is measured as the relative gap between the length of the best-obtained solution $\mathcal{L}$ among the performed trials for the particular solver and the lower bound value of the problem instance $\mathcal{LB}$ determined by the proposed BB solver.

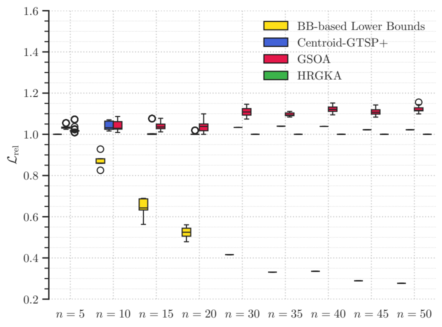
$$\%\mathrm{GAP} = \frac{\mathcal{L} - \mathcal{LB}}{\mathcal{L}}\, 100\,\% \tag{19}$$

The second indicator is the five-number summary of the relative solution length $\mathcal{L}_{\mathrm{rel}} = \mathcal{L}/\mathcal{L}_{\mathrm{best}}$, where $\mathcal{L}$ is the best-found solution among trials performed by the particular solver, and $\mathcal{L}_{\mathrm{best}}$ is the best-known solution of the instance.

[3]$\overline{\mathrm{T}_{\mathrm{CPU}}}$ is not of primary interest for showing results on the computed lower bound values. Besides, times significantly vary for each solver depending on the parallel computation capability. Therefore, the reported times are not normalized and are listed as reported and measured.



(a) Relative length $\mathcal{L}_{\text{rel}}$ for 3D



(b) Relative length $\mathcal{L}_{\text{rel}}$ for 7D

Figure 7: The five-number summary of the relative lengths $\mathcal{L}_{\text{rel}}$ for the 3D and 7D instances. For the results denoted BB-based Lower Bounds, $\mathcal{L}_{\text{rel}}$ values are computed using the found lower bound values $\mathcal{LB}$ as $\mathcal{L}_{\text{rel}} = \mathcal{LB}/\mathcal{L}_{\text{best}}$.

Besides, we present statistics indicators as a percentage ratio of the number of OPENED nodes, PRUNED nodes, and UNPROCESSED nodes to the number of CREATED nodes to provide insight to the studied BB-based method. The percentage of nodes for which the EXACT solutions have been computed and have been reinserted into the open list is also reported. Note that the number of UNPROCESSED nodes is computed as CREATED − OPENED − PRUNED + EXACT.

Due to the relatively high number of evaluated instances, we aggregated results among particular variants v for the 3D and 7D instances so that the instances are grouped according to the number of sets for $m_i = 6$. Hence, the performance indicators are reported as the mean values $\overline{\mathcal{L}_{\text{best}}}$, $\overline{\mathcal{LB}_{\text{BB}}}$, and $\overline{\%\text{GAP}_{\text{BB}}}$ for the instances with the name in the form of $d\texttt{D}\_n\_m_i$. The aggregated evaluation results are reported in Tables 1 and 2, and the five-number summary indicators are plotted in Figure 7. Besides, in Figure 8, we report on the comparison of the proposed BB-based method with and without the employment of the estimation of the partial solutions as described in Section 3.2 to show the benefits of the SOCP estimation. The aggregated statistic is reported in Tables 3 and 4. Detail results with the determined lower bound values for all the instances can be found in Tables A.1 to A.8 in Supplementary materials. Note that the best-found solutions are highlighted in bold in the tables. In addition, detailed statistics can be further found in Tables B.9 to B.15 in Supplementary materials.

The results for the 3D instances in Table 1 indicate that the proposed BB-based GTSPN solver finds tight lower bounds for instances with a small number of sets. The optimal solutions are found for all instances with n up to 10, for most of the instances

Table 1: Aggregated results of the 3D GTSPN instances.

Instance	n	$\overline{\mathcal{L}_{\text{best}}}$	$\overline{\mathcal{LB}_{\text{BB}}}$	$\overline{\%\text{GAP}_{\text{BB}}}$	$\overline{\text{T}^{\text{BB}}_{\text{CPU}}}$	HRGKA		GSOA		Centroid-GTSP$^+$	
						%GAP	$\overline{\text{T}_{\text{CPU}}}$	%GAP	$\overline{\text{T}_{\text{CPU}}}$	%GAP	$\overline{\text{T}_{\text{CPU}}}$
3D_05_6	05	**1 309.28**	1 674.82	0.00	≈ 1 min	-*	-*	0.43	2.66 s	0.26	0.17 s
3D_06_6	06	1 458.04	2 021.16	0.00	≈ 1 min	-*	-*	0.47	3.91 s	0.26	0.23 s
3D_07_6	07	**1 640.90**	2 047.28	0.00	≈ 3 min	-*	-*	0.50	5.39 s	0.17	0.20 s
3D_08_6	08	**1 745.46**	2 277.66	0.00	≈ 10 min	-*	-*	0.68	7.43 s	0.21	0.33 s
3D_09_6	09	1 813.43	2 387.71	0.00	≈ 26 min	-*	-*	0.79	9.26 s	0.21	0.30 s
3D_10_6	10	**1 997.32**	2 425.86	0.00	≈ 47 min	-*	-*	0.76	11.65 s	0.17	0.23 s
3D_11_6	11	2 080.06	2 441.85	0.17	≈ 3 h	-*	-*	1.31	14.20 s	0.39	0.20 s
3D_12_6	12	2 129.42	2 608.91	1.08	≈ 5 h	-*	-*	2.22	17.24 s	1.34	0.57 s
3D_13_6	13	2 306.85	2 765.65	4.11	≈ 9 h	-*	-*	5.16	20.02 s	4.18	0.47 s
3D_14_6	14	2 385.79	2 676.82	4.90	≈ 9 h	-*	-*	6.10	23.58 s	4.79	0.37 s
3D_15_6	15	2 543.80	2 650.26	10.72	≈ 10 h	-*	-*	14.71	26.44 s	13.38	0.37 s
3D_20_6	20	2 950.02	2 787.98	19.72	≈ 10 h	-*	-*	20.55	47.04 s	18.56	0.50 s
3D_25_6	25	3 416.19	2 836.54	31.21	≈ 10 h	-*	-*	30.41	≈ 1 min	28.50	0.67 s
3D_30_6	30	3 645.52	2 551.23	39.63	≈ 10 h	36.37	50.98 s	38.24	0.13 s	**36.08**	< 10 ms
3D_35_6	35	4 064.69	2 609.81	43.46	≈ 10 h	39.28	≈ 1 min	41.71	0.17 s	**39.02**	< 10 ms
3D_40_6	40	4 264.24	2 739.98	46.27	≈ 10 h	43.28	≈ 1 min	45.16	0.23 s	**42.90**	< 10 ms
3D_45_6	45	4 750.11	2 900.72	48.50	≈ 10 h	44.68	≈ 1 min	46.74	0.28 s	**44.09**	< 10 ms
3D_50_6	50	4 924.53	2 693.10	53.09	≈ 10 h	49.33	≈ 1 min	50.94	0.34 s	**48.70**	< 10 ms

*Values are not reported in Vicencio et al. (2014)

with n up to 13, and for some of the instances with up to 30 region sets.[4] The BB-based solver outperforms all the other solvers (HRKGA, GSOA, and Centroid-GTSP$^+$) for instances with n to 12; see Figure 7a. However, for more than 14 region sets, the BB-based solver (in the given $t_{\max}$) provides relatively poor solutions compared to the examined heuristics.

Regarding the results for the 7D instances reported in Table 2, the GSOA and Centroid-GTSP$^+$ solvers are outperformed by the HRKGA for the available results. The HRKGA provides the lowest %GAP, albeit significantly higher than for the 3D instances. However, for some of the instances with the number of region sets $n \in \{5, 10, 20\}$, the proposed BB-based solver provides solutions with the tighter %GAP than the GSOA and Centroid-GTSP$^+$.

Regarding the performance of the BB-based solver summarized in Tables 3 and 4, it processed all created tree nodes of instances with n up to 10. For instances with $n = 15$ and above, the solver prunes a low number of nodes and more than 50 % of nodes remains unprocessed.[5]

[4]See detailed results in Tables A.1 to A.6 in Supplementary materials.

[5]See the detailed results in Tables B.9 to B.15 in Supplementary materials.

Table 2: Aggregated results of the 7D GTSPN instances.

Instance	n	$\overline{\mathcal{L}_{\text{best}}}$	$\overline{\mathcal{LB}_{\text{BB}}}$	$\overline{\%\text{GAP}_{\text{BB}}}$	$\overline{\text{T}^{\text{BB}}_{\text{CPU}}}$	HRGKA %GAP	HRGKA $\overline{\text{T}_{\text{CPU}}}$	GSOA %GAP	GSOA $\overline{\text{T}_{\text{CPU}}}$	Centroid-GTSP$^+$ %GAP	Centroid-GTSP$^+$ $\overline{\text{T}_{\text{CPU}}}$
7D_05_6	05	**2 158.05**	2 431.84	0.00	≈ 2 min	- *	- *	2.20	51.70 s	3.38	0.47 s
7D_10_6	10	3 874.16	3 518.50	12.90	≈ 10 h	- *	- *	16.36	≈ 2 min	16.38	0.78 s
7D_15_6	15	5 541.13	3 918.00	36.17	≈ 10 h	- *	- *	38.11	≈ 4 min	36.66	1.18 s
7D_20_6	20	6 718.83	3 689.91	48.22	≈ 10 h	- *	- *	49.59	≈ 8 min	47.90	1.60 s
7D_30_6	30	9 263.10	3 852.95	61.02	≈ 10 h	**58.41**	≈ 1 min	62.44	1.45 s	59.74	< 10 ms
7D_35_6	35	10 704.00	3 545.05	68.57	≈ 10 h	**66.88**	≈ 1 min	69.78	2.08 s	68.14	< 10 ms
7D_40_6	40	10 871.40	3 642.63	68.35	≈ 10 h	**66.49**	≈ 1 min	70.13	2.62 s	67.72	< 10 ms
7D_45_6	45	12 142.20	3 507.62	73.00	≈ 10 h	**71.11**	≈ 2 min	73.97	3.21 s	71.74	< 10 ms
7D_50_6	50	13 868.40	3 838.44	73.63	≈ 10 h	**72.32**	≈ 2 min	75.31	4.09 s	72.92	< 10 ms

*Values are not reported in Vicencio et al. (2014)

Table 3: Aggregated statistic evaluation of the 3D GTSPN instances.

Instance	n	$\overline{\mathcal{LB}_{\text{BB}}}$	$\overline{\%\text{GAP}_{\text{BB}}}$	$\overline{\text{T}^{\text{BB}}_{\text{CPU}}}$	$\overline{\text{CREATED}}$ nodes	OPENED nodes	PRUNED nodes	EXACT nodes	UNPROCESSED nodes
3D_05_6	05	**1 309.28**	**0.00**	≈ 1 min	10.5	77.85%	65.51%	43.35%	0.00%
3D_06_6	06	**1 458.04**	**0.00**	≈ 1 min	21.7	66.77%	72.77%	39.54%	0.00%
3D_07_6	07	**1 640.90**	**0.00**	≈ 3 min	46.6	57.05%	77.67%	34.72%	0.00%
3D_08_6	08	**1 745.46**	**0.00**	≈ 10 min	128.4	51.77%	81.57%	33.33%	0.00%
3D_09_6	09	**1 813.50**	**0.00**	≈ 26 min	268.0	47.93%	83.93%	31.86%	0.00%
3D_10_6	10	**1 997.32**	**0.00**	≈ 47 min	475.1	45.76%	85.17%	30.93%	0.00%
3D_11_6	11	2 076.16	0.17	≈ 3 h	1 035.1	44.22%	83.82%	30.69%	2.65%
3D_12_6	12	2 104.73	1.08	≈ 5 h	1 921.6	40.90%	78.42%	28.29%	8.97%
3D_13_6	13	2 211.29	4.11	≈ 9 h	2 264.1	42.25%	58.26%	29.87%	29.35%
3D_14_6	14	2 272.57	4.90	≈ 9 h	2 807.9	39.22%	55.15%	27.18%	32.81%
3D_15_6	15	2 281.18	10.72	≈ 10 h	2 558.8	40.62%	28.50%	28.33%	59.21%
3D_20_6	20	2 399.77	19.72	≈ 10 h	2 340.9	40.94%	6.32%	28.49%	81.23%
3D_25_6	25	2 440.15	31.21	≈ 10 h	2 399.9	40.81%	0.08%	28.38%	87.48%
3D_30_6	30	2 330.62	39.63	≈ 10 h	2 090.0	40.81%	0.00%	28.11%	87.30%
3D_35_6	35	2 476.95	43.46	≈ 10 h	2 241.8	40.87%	0.00%	28.33%	87.45%
3D_40_6	40	2 430.57	46.27	≈ 10 h	2 194.5	42.61%	0.00%	29.95%	87.34%
3D_45_6	45	2 657.69	48.50	≈ 10 h	2 668.5	37.29%	0.00%	24.90%	87.61%
3D_50_6	50	2 521.85	53.09	≈ 10 h	2 144.0	39.13%	0.00%	26.40%	87.27%

As for the comparison of the proposed BB-based solver with and without SOCP-based estimation of the partial solution. The results depicted in Figure 8 supports the employment of the estimation since the solutions of the problems are obtained in shorter computational times.

Based on the tight lower bounds found for small instances and assuming heuristic

Table 4: Aggregated statistic evaluation of the 7D GTSPN instances.

Instance	n	$\overline{\mathcal{LB}_{\mathrm{BB}}}$	$\overline{\%\mathrm{GAP}_{\mathrm{BB}}}$	$\overline{\mathrm{T}^{\mathrm{BB}}_{\mathrm{CPU}}}$	$\overline{\text{CREATED}}$ nodes	OPENED nodes	PRUNED nodes	EXACT nodes	UNPROCESSED nodes
7D_05_6	05	**2 158.05**	**0.00**	≈ 2 min	14.4	76.39%	68.06%	44.44%	0.00%
7D_10_6	10	3 372.03	12.90	≈ 10 h	1 049.8	58.54%	7.49%	43.32%	77.29%
7D_15_6	15	3 559.28	36.17	≈ 10 h	782.6	55.87%	0.00%	40.12%	84.26%
7D_20_6	20	3 505.66	48.22	≈ 10 h	752.6	53.55%	0.00%	38.03%	84.48%
7D_30_6	30	3 852.95	61.02	≈ 10 h	710.0	47.46%	0.00%	31.55%	84.08%
7D_35_6	35	3 545.05	68.57	≈ 10 h	1 120.0	55.36%	0.00%	40.45%	85.09%
7D_40_6	40	3 642.63	68.35	≈ 10 h	947.0	51.53%	0.00%	36.11%	84.58%
7D_45_6	45	3 507.62	73.00	≈ 10 h	989.0	54.70%	0.00%	39.23%	84.53%
7D_50_6	50	3 838.44	73.63	≈ 10 h	314.0	72.93%	0.00%	55.73%	82.80%

solvers provide relatively satisfactory solutions, the performance of the proposed BB-based solver decreases with the increasing number of region sets for the given $t_{\max}$. Nevertheless, the proposed solver is (to the authors' best knowledge) the only solver that provides the lower bounds to the GTSPN. Thus, the promising results for small instances motivate further research on improvements to address larger instances of the GTSPN.

5. Conclusion

We propose the first lower bounds on the optimal solution values of the GT-SPN instances with supportive results for the 3D and 7D problem instances. The bounds are obtained by the BB-based solver utilizing the MISOCP to obtain optimal solutions to partial problems and utilizing the SOCP to estimate the solution costs. Regarding the reported evaluation results, the proposed BB-solver provides the lower bound values on the solutions of the GTSPN instances that allow benchmarking existing heuristic solvers using the absolute quality measure to the found solutions. Due to optimal solutions to the partial problems, the proposed solver is too demanding to be competitive with the existing heuristics, specifically for large instances. However, it is the only solver providing the lower bound values on the GT-SPN solutions. Thus, the proposed method allows for assessing the existing heuristic solvers. The reported results support that the provided lower bounds are tight for the instances with a few region sets. Besides, the first optimal solutions are reported for several evaluated benchmark instances. The results motivate further research on computational improvements for solving larger instances.

Moreover, we are prompted to address robotic applications motivating the studied

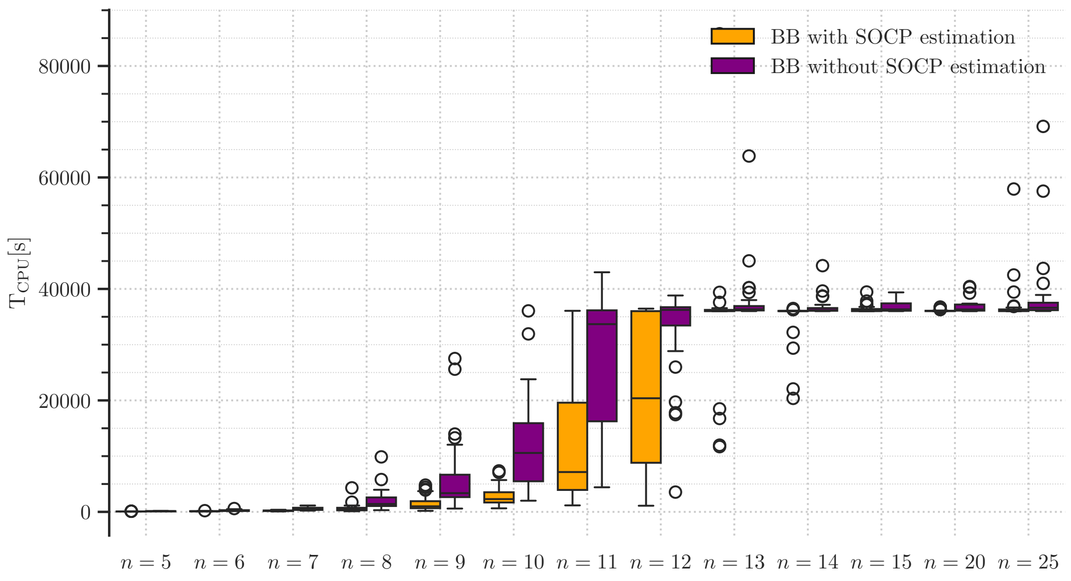


Figure 8: Aggregated computational requirements of the proposed method. Computational requirements of the proposed method, including the estimation using the bounding boxes with SOCP, are depicted in orange, and without the SOCP estimation are depicted in purple.

GTSPN (since the GTSPN formulation allows us to define complex shapes of neighborhoods), where further computational challenges arise from the robotic systems' motion constraints, such as the viewpoint planning problems with mobile robots.

Acknowledgements

The presented work has been initiated under research project No. 19-20238S and finished under project No. 22-05762S, supported by the Czech Science Foundation (GAČR). The access to the computational infrastructure of the OP VVV funded project CZ.02.1.01/0.0/0.0/16_019/0000765 "Research Center for Informatics" is also gratefully acknowledged.

References

Alatartsev, S., Stellmacher, S., Ortmeier, F., 2015. Robotic Task Sequencing Problem: A Survey. Journal of Intelligent & Robotic Systems 80, 279–298. doi:10.1007/s10846-015-0190-6.

Arkin, E.M., Hassin, R., 1994. Approximation algorithms for the geometric covering salesman problem. Discrete Applied Mathematics 55, 197–218. doi:10.1016/0166-218X(94)90008-6.

Behdani, B., Smith, J.C., 2014. An integer-programming-based approach to the close-enough traveling salesman problem. INFORMS: Journal on Computing 26, 415–432. doi:10.1287/ijoc.2013.0574.

Ben-Tal, A., Nemirovski, A., 2001. Lectures on Modern Convex Optimization. Society for Industrial and Applied Mathematics. doi:10.1137/1.9780898718829.

de Berg, M., Gudmundsson, J., Katz, M.J., Levcopoulos, C., Overmars, M.H., van der Stappen, A.F., 2005. TSP with neighborhoods of varying size. Journal of Algorithms 57, 22–36. doi:10.1016/j.jalgor.2005.01.010.

Carrabs, F., Cerrone, C., Cerulli, R., D'Ambrosio, C., 2017a. Improved upper and lower bounds for the close enough traveling salesman problem, in: Green, Pervasive, and Cloud Computing, pp. 165–177. doi:10.1007/978-3-319-57186-7_14.

Carrabs, F., Cerrone, C., Cerulli, R., Gaudioso, M., 2017b. A novel discretization scheme for the close enough traveling salesman problem. Computers & Operations Research 78, 163–171. doi:10.1016/j.cor.2016.09.003.

Carrabs, F., Cerrone, C., Cerulli, R., Golden, B., 2020. An adaptive heuristic approach to compute upper and lower bounds for the close-enough traveling salesman problem. INFORMS: Journal on Computing doi:10.1287/ijoc.2020.0962.

Clausen, J., 1999. Branch and bound algorithms-principles and examples. Department of Computer Science, University of Copenhagen , 1–30.

Coutinho, W.P., Nascimento, R.Q.d., Pessoa, A.A., Subramanian, A., 2016. A branch-and-bound algorithm for the close-enough traveling salesman problem. INFORMS: Journal on Computing 28, 752–765. doi:10.1287/ijoc.2016.0711.

CPLEX 12.9, . IBM ILOG CPLEX Optimization Studio. URL: https://www.ibm.com/products/ilog-cplex-optimization-studio. accessed June 29, 2022.

Dumitrescu, A., Mitchell, J.S.B., 2003. Approximation algorithms for TSP with neighborhoods in the plane. Journal of Algorithms 48, 135–159. doi:10.1016/S0196-6774(03)00047-6.

Dunbabin, M., Marques, L., 2012. Robots for Environmental Monitoring: Significant Advancements and Applications. IEEE Robotics & Automation Magazine 19, 24–39. doi:10.1109/MRA.2011.2181683.

Dunning, I., Huchette, J., Lubin, M., 2017. Jump: A modeling language for mathematical optimization. SIAM Review 59, 295–320. doi:10.1137/15M1020575.

Elbassioni, K., Fishkin, A.V., Sitters, R., 2009. Approximation algorithms for the Euclidean traveling salesman problem with discrete and continuous neighborhoods. International Journal of Computational Geometry & Applications 19, 173–193. doi:10.1142/S0218195909002897.

Faigl, J., 2018. GSOA: growing self-organizing array - unsupervised learning for the close-enough traveling salesman problem and other routing problems. Neurocomputing 312, 120–134. doi:10.1016/j.neucom.2018.05.079.

Faigl, J., 2019. Data collection path planning with spatially correlated measurements using growing self-organizing array. Applied Soft Computing 75, 130–147. doi:10.1016/j.asoc.2018.11.005.

Faigl, J., Váňa, P., Deckerová, J., 2019. Fast heuristics for the 3-d multi-goal path planning based on the generalized traveling salesman problem with neighborhoods. IEEE Robotics and Automation Letters 4, 2439–2446. doi:10.1109/LRA.2019.2900507.

Faigl, J., Vonásek, V., Přeučil, L., 2013. Visiting convex regions in a polygonal map. Robotics and Autonomous Systems 61, 1070–1083. doi:10.1016/j.robot.2012.08.013.

Gentilini, I., 2012. Multi-goal path optimization for robotic systems with redundancy based on the traveling salesman problem with neighborhoods. Ph.D. thesis. Carnegie Mellon University. doi:10.1184/R1/6720746.v1.

Gentilini, I., Margot, F., Shimada, K., 2013. The travelling salesman problem with neighbourhoods: MINLP solution. Optimization Methods and Software 28, 364–378. doi:10.1080/10556788.2011.648932.

Griva, I., Nash, S.G., Sofer, A., 2008. Linear and Nonlinear Optimization 2nd Edition. SIAM.

Gulczynski, D.J., Heath, J.W., Price, C.C., 2006. The Close Enough Traveling Salesman Problem: A Discussion of Several Heuristics. pp. 271–283. doi:10.1007/978-0-387-39934-8_16.

Gutin, G., Punnen, A.P. (Eds.), 2007. The Traveling Salesman Problem and Its Variations. Springer US. doi:10.1007/b101971.

Helsgaun, K., 2015. Solving the equality generalized traveling salesman problem using the lin-kernighan-helsgaun algorithm. Mathematical Programming Computation 7, 269–287. doi:10.1007/s12532-015-0080-8.

Karapetyan, D., Gutin, G., 2012. Efficient local search algorithms for known and new neighborhoods for the generalized traveling salesman problem. European Journal of Operational Research 219, 234–251. doi:10.1016/j.ejor.2012.01.011.

Land, A.H., Doig, A.G., 1960. An automatic method of solving discrete programming problems. Econometrica 28, 497–520.

Mennell, W.K., 2009. Heuristics for Solving Three Routing Problems: Close-Enough Traveling Salesman Problem, Close-Enough Vehicle Routing Problem, Sequence-Dependent Team Orienteering Problem. Ph.D. thesis. University of Maryland.

Noon, C.E., Bean, J.C., 1993. An efficient transformation of the generalized traveling salesman problem. INFORMS: Information Systems and Operational Research 31, 39–44. doi:10.1080/03155986.1993.11732212.

Oberlin, P., Rathinam, S., Darbha, S., 2010. Today's traveling salesman problem. IEEE Robotics & Automation Magazine 17, 70–77. doi:10.1109/MRA.2010.938844.

Pintea, C.M., Pop, P.C., Chira, C., 2007. The generalized traveling salesman problem solved with ant algorithms. Journal of Universal Computer Science 13, 1065–1075. doi:10.1186/s40294-017-0048-9.

Silberholz, J., Golden, B., 2007. The generalized traveling salesman problem: A new genetic algorithm approach, in: Extending the horizons: advances in computing, optimization, and decision technologies. Springer, pp. 165–181. doi:10.1007/978-0-387-48793-9_11.

Smith, S.L., Imeson, F., 2017. GLNS: An effective large neighborhood search heuristic for the generalized traveling salesman problem. Computers & Operations Research 87, 1–19. doi:10.1016/j.cor.2017.05.010.

Suárez-Ruiz, F., Lembono, T.S., Pham, Q.C., 2018. Robotsp – a fast solution to the robotic task sequencing problem, in: IEEE International Conference on Robotics and Automation (ICRA), pp. 1611–1616. doi:10.1109/ICRA.2018.8460581.

Vicencio, K., 2014. Randomly generated GTSPN instances. URL: http://robotics.pr.erau.edu/data/gtspn.zip.

Vicencio, K., Davis, B., Gentilini, I., 2014. Multi-goal path planning based on the generalized traveling salesman problem with neighborhoods, in: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 2985–2990. doi:10.1109/IROS.2014.6942974.

Yang, Z., Xiao, M.Q., Ge, Y.W., Feng, D.L., Zhang, L., Song, H.F., Tang, X.L., 2018. A double-loop hybrid algorithm for the traveling salesman problem with arbitrary neighbourhoods. European Journal of Operational Research 265, 65–80. doi:10.1016/j.ejor.2017.07.024.

Yuan, B., Orlowska, M., Sadiq, S., 2007. On the Optimal Robot Routing Problem in Wireless Sensor Networks. IEEE Transactions on Knowledge and Data Engineering 19, 1252–1261. doi:10.1109/TKDE.2007.1062.

Appendix A. Used Notation

Table A.5: Used symbols

S_i	The i-th region set.	$\mathcal{S}$	Set of n region sets S_i.
d	Dimension of the input space.	$\boldsymbol{c}_i$	Center of region S_i.
$Q_{i,k}$	The k-th region of region set S_i	$\mathbf{A}, \boldsymbol{b}$.	Matrix and vector defining polyhedron.
$\mathbf{P}$	Symmetric positive definite matrix defining ellipsoid.	$\boldsymbol{p}_i$	Location of visit to the region set S_i.
P	Set of points of visits to $\mathcal{S}$.	Σ	Sequence of visits to target regions.
σ_i	The i-th visit to region set S_i.		
$\mathcal{M}$	Solution space of the BB.	$t_{\max}$	Maximal computational time of BB.
μ	Node of solution space $\mathcal{M}$.	$\mathcal{O}$	Open list.
β	Set of not covered regions.	$\boldsymbol{f}$	Objective variables.
$\boldsymbol{x}$	Continuous variables representing P.	$\boldsymbol{z}$	Binary variables for *big M*.
$\mathbf{M}$	*Big M* variables.	B_i	Bounding box of the region set S_i.
$\boldsymbol{x}_i^{\mathrm{lb}}$	Coordinate values of the lower left point of B_i.	$\boldsymbol{x}_i^{\mathrm{ub}}$	Coordinate values of the upper right point of B_i.