Fast Heuristics for the 3D Multi-Goal Path Planning based on the Generalized Traveling Salesman Problem with Neighborhoods

Jan Faigl, Petr Váňa, and Jindřiška Deckerová

***Abstract*—In this paper, we address the multi-goal path planning problem to determine a cost-efficient path to visit a set of 3D regions. The problem is a variant of the Traveling Salesman Problem with Neighborhoods (TSPN) where an individual neighborhood consists of multiple regions, and the problem is to determine a shortest multi-goal path to visit at least one region of each neighborhood. Because each neighborhood may consist of several regions, it forms a neighborhood set, and the problem is called the Generalized TSPN (GTSPN) in the literature. We propose two heuristic algorithms to provide a feasible solution of the GTSPN quickly. The first algorithm is based on a decoupled approach using a solution of the Generalized TSP that is further improved by a quick post-processing procedure. Besides, we propose to employ the existing unsupervised learning technique called the Growing Self-Organizing Array (GSOA) to quickly find a feasible solution of the GTSPN that can be further improved by more demanding optimization. The reported results on existing benchmarks for the GTSPN indicate that both proposed heuristics provide better or competitive solutions than the state-of-the-art reference algorithm, but they are up to two orders of magnitude faster.**

***Index Terms*—Motion and Path Planning; Planning, Scheduling and Coordination; Aerial Systems: Applications**

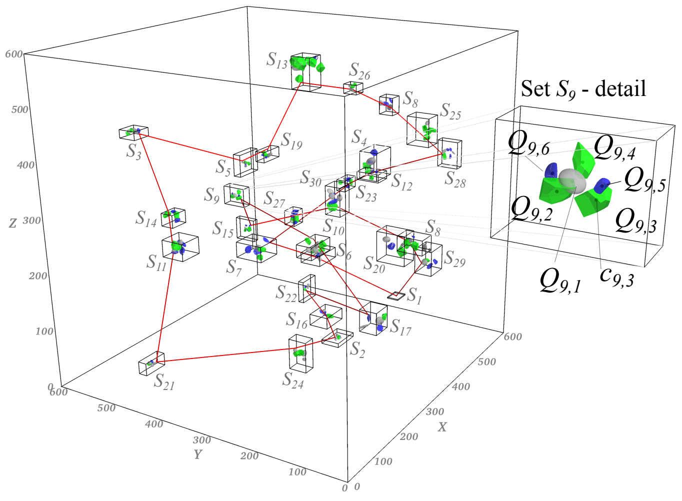


Fig. 1. An instance of the GTSPN with 3D regions and its solution found by the proposed method. An individual neighborhood set $S_i = \{Q_{i,1}, \ldots, Q_{i,m_i}\}$ is shown in a rectangular bounding box and consists of particular regions $Q_{i,k} \subset \mathbb{R}^3$. Each region is specified by its centroid $\boldsymbol{c}_{i,k} \in \mathbb{R}^3$. Regions of the set S_9 are further visualized in Fig. 2.

I. Introduction

MULTI-GOAL path planning (MTP) [1] can be considered as a robotic variant of the Traveling Salesman Problem (TSP). It as a well-studied combinatorial optimization problem of the operational research with many existing approaches [2], [3]. Having a set of locations, the problem is to determine an optimal sequence to visit all the locations such that the length of the path to the locations is minimal. Since we are looking for the optimal permutation of the visits to the locations, the problem is also studied for the industrial robotic applications as the robotic task sequencing problem [1], [4] in addition to, e.g., routing problems with aerial vehicles [5].

On the other hand, it is not always desirable to visit the target location precisely as a single point, and it may be more suitable to describe the targets as regions that allow exploiting a certain degree of freedom [6]. Then, the problem becomes a variant of the TSP with Neighborhoods (TSPN), which is reported to be a suitable problem formulation for robotic data collection and monitoring missions [7], [8], [9], but also for complex manipulator tasks [10], [11], [12]. Continuous neighborhoods in the TSPN can be sampled into discrete sets, and the problem becomes a variant of the Generalized TSP (GTSP) [13] which can be addressed by heuristic approaches [14], [15] or further transformed to the TSP [16] at the cost of significantly increased problem size. Although the GTSP-based approach provides a feasible solution of the TSPN, even the Euclidean TSPN is known to be APX-hard [17], and therefore, specific variants of the TSPN with restricted types of the neighborhoods are studied [18], [19]. However, the TSPN is still challenging problem for most of the neighborhood types [20].

Regarding robotic applications, the TSPN with disk-shaped neighborhoods attracts the attention of the community as the Close Enough TSP (CETSP) formulation that has been introduced for data collection planning using wireless communication in [21], where the authors propose a concept of supernodes. The supernodes are explicit samples of the neighborhood regions, and several heuristic methods have been proposed in the literature [21], [22], [9]. An extensive com-

Manuscript received: September, 10, 2018; Revised December, 11, 2018; Accepted January, 29, 2019.

This paper was recommended for publication by Editor Nancy Amato upon evaluation of the Associate Editor and Reviewers' comments.

Authors are with the Czech Technical University, Faculty of Electrical Engineering, Technicka 2, 166 27, Prague, Czech Republic `{faiglj|vanapet1|deckejin}@fel.cvut.cz`

The presented work has been supported by the Czech Science Foundation (GAČR) under research project No. 16-24206S and No. 19-20238S. The authors acknowledge the support of the OP VVV funded project CZ.02.1.01/0.0/0.0/16_019/0000765 "Research Center for Informatics".

Digital Object Identifier (DOI): see top of this page.

putational survey of existing approaches is presented in [23], where the author reports that the best trade-off between the solution quality and computational requirements is provided by his sampling of the supernodes in the Steiner zones, which are the overlapping parts of the neighborhood regions. Besides, he reports that the best results are provided by the GTSP-based solution, but with very high computational requirements (in hours). In [24], the CETSP is addressed by an unsupervised learning technique called the Growing Self-Organizing Array (GSOA) in which the neighborhood regions are sampled during the learning and solution of the sequencing part of the problem. Therein reported results for the benchmarks [23] show competitive solutions to the Steiner zones while the GSOA-based approach is about one order of magnitude faster. Moreover, in several cases, new best solutions are found in a fraction of second, which significantly outperforms GTSP-based approaches with an explicit sampling of the neighborhood before the solution of the sequencing part.

The promising results of the GSOA reported in [24] motivates us to employ the technique in a more challenging variant of the multi-goal path planning with 3D regions. In particular, we consider an extended variant of the TSPN defined for sets of neighborhoods introduced in [25] to overcome limitations of the continuous and discrete formulation of the TSPN. The problem is called the Generalized TSPN (GTSPN) in [11], where the authors propose to use their Hybrid Random-Key Genetic Algorithm (HRKGA) [26] to find high-quality solutions of the introduced GTSPN benchmark instances. The GTSPN instances provided in [26] are motivated by tasks for redundant robotic manipulators, computer vision-based inspection with multiple possible views [10], but also multi-goal aircraft missions [27]. The authors proposed to model each neighborhood as a set of regions to provide enough flexibility for a definition of the target regions and solution of practical problems. In particular, the regions are defined as ellipsoids, polyhedra, and a combination of both; and the benchmark instances include 30 problems with up to 50 neighborhood sets, where each set consists of six 3D regions. An example of the GTSPN is visualized in Fig. 1 and region types are in Fig. 2.

In this paper, we consider the 3D benchmark instances of the GTSPN provided in [11], and we propose two new heuristics for solving the GTSPN with neighborhoods defined as polyhedra and ellipsoids to quickly find a feasible solution. In particular, we propose to utilize centroids of the regions and a solution of the GTSP followed by local iterative post-processing optimization. Besides, we propose a simple heuristic to quickly determine shortest distances from the array of the GSOA to the neighborhood regions that enables to use the existing GSOA [24] to solve 3D instances of the GTSPN. Based on the herein reported results, the proposed approaches provide competitive or better solutions than the HRKGA [11]. Moreover, they are up to two orders of magnitude faster than the reported computational times for the HRKGA. Hence, the proposed fast heuristics are suitable alternatives for solving the GTSPN. In addition, new best-found solutions of the evaluated instances have been established using multiple trials of the proposed approaches.

The rest of the paper is organized as follows. The GTSPN is formally defined in the next section together with the description of the neighborhood sets. The decoupled approach to the GTSPN based on the GTSP and post-processing local optimization is proposed in Section III. The GSOA-based solution to the GTSPN is presented in Section IV and results from the empirical evaluation are reported in Section V. Concluding remarks together with our future work are summarized in Section VI.

II. Problem Statement

In the addressed Generalized Traveling Salesman Problem with Neighborhoods (GTSPN), the neighborhoods are represented as neighborhood sets and the problem is to determine a shortest path such that, each set is visited by the path. We follow the notation used in [11] and since we consider 3D instances of the GTSPN proposed there, the problem is formulated for 3D regions and a cost function computed as the Euclidean distance between two points $\boldsymbol{p}_i, \boldsymbol{p}_j \in \mathbb{R}^3$ denoted $\left\|(\boldsymbol{p}_i, \boldsymbol{p}_j)\right\|$.

Let $\mathcal{S}$ be the given set of n neighborhood sets, $\mathcal{S} = \{S_1, \ldots, S_n\}$, where each set $S_i \in \mathcal{S}$ consists of m_i regions $S_i = \{Q_{i,1}, \ldots, Q_{i,m_i}\}$, $Q_{i,k} \subset \mathbb{R}^3$ for $1 \le i \le n$ and $1 \le k \le m_i$, see an example of the GTSPN instance in Fig. 1. Then, the GTSPN stands to determine a sequence of visits $\Sigma = (\sigma_1, \ldots, \sigma_n)$ to the sets $S_{\sigma_i} \in \mathcal{S}$ together with the corresponding waypoint locations $P = \{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n\}$ such that $\boldsymbol{p}_i \in S_i$, i.e., $\boldsymbol{p}_i$ is inside of at least one region $Q_{i,k} \in S_i$, which can be expressed as $\bigcup_{k=1}^{m_i}(\{\boldsymbol{p}_i\} \cap Q_{i,k}) \neq \emptyset$, and the path connecting the locations P has the minimal length.

Problem 2.1 (GTSPN):

$$\begin{array}{l} \text{minimize}_{\Sigma,P} \\ \quad L(\Sigma, P) = \sum_{i=1}^{n-1} \left\|(\boldsymbol{p}_{\sigma_i}, \boldsymbol{p}_{\sigma_{i+1}})\right\| + \left\|(\boldsymbol{p}_{\sigma_n}, \boldsymbol{p}_{\sigma_1})\right\| \\ \text{s.t.} \\ \quad \Sigma = (\sigma_1, \ldots, \sigma_n),\ 1 \le \sigma_i \le n,\ \sigma_i \neq \sigma_j \text{ for } i \neq j \\ \quad P = \{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n\},\ \bigcup_{k=1}^{m_i}(\{\boldsymbol{p}_i\} \cap Q_{i,k}) \neq \emptyset \end{array} .$$

Notice, there is no explicit constraint that each neighborhood set is visited exactly once, e.g., as in the formulation [11]; however, each set is considered visited only at a single waypoint location $\boldsymbol{p} \in P$ from eventually infinite possibilities. On the other hand, a single waypoint can be inside multiple neighborhood sets, e.g., as in the overlapping regions in the CETSP [24]. In such a case, we consider multiple (consecutive) identical waypoint locations, which does not affect the length of the path $L(\Sigma, P)$.

The GTSPN combines the combinatorial optimization in determining the sequence Σ with the continuous optimization to determine the waypoint locations P that are restricted to be inside the respective regions of the neighborhood sets.

A. Representation of 3D Regions of the Neighborhood Sets

3D instances of the GTSPN are specified as neighborhood sets that consist of individual 3D regions. The authors of [11]

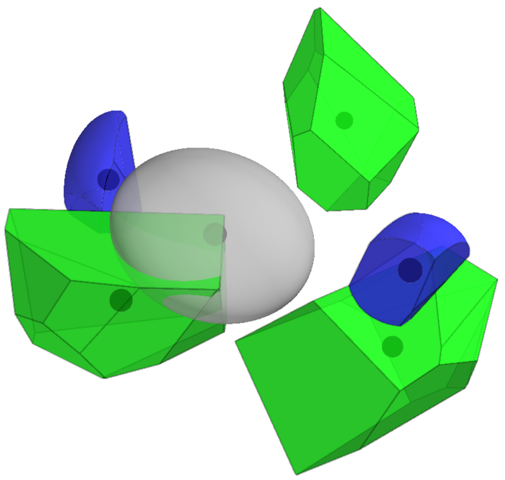


Fig. 2. Example of 3D regions: the ellipsoid is shown in the gray, the polyhedra are in the green, and the hybrid regions are in the blue.

propose three types of the regions: ellipsoid, polyhedron, and their combination, see Fig. 2. Even though a detail description of the regions and also the process of their creation can be found in [11], the utilized representation is presented in this section to make the paper self-contained.

A region $Q_{i,k}$ of the ellipsoid type is defined by its center $\boldsymbol{c}_{i,k}$ and symmetric positive definite matrix $\mathbf{P}_{i,k}$ with the dimension (3×3) as a set of all points $\boldsymbol{x} \in \mathbb{R}^3$ satisfying

$$(\boldsymbol{x} - \boldsymbol{c}_{i,k})^T \mathbf{P}_{i,k}^{-1} (\boldsymbol{x} - \boldsymbol{c}_{i,k}) \leq 1. \tag{1}$$

The polyhedron is defined by 12 half-spaces, and a point $\boldsymbol{x}$ is inside the polyhedron region $Q_{i,k}$ if the corresponding set of equations holds[1]

$$\mathbf{A}_{i,k}\boldsymbol{x} - \mathbf{b}_{i,k} \leq 0, \tag{2}$$

where $\mathbf{A}_{i,k}$ is a matrix of the dimensions (12×3) and $\mathbf{b}_{i,k}$ is a matrix (vector) with the dimensions (12×1). We assume the center of the polyhedron $\boldsymbol{c}_{i,k}$ is inside the polyhedron, which holds for the instances [11] because the polyhedra are convex. Since the neighborhood set may consist of multiple (overlapping) regions, the assumption on the convexity still provides sufficient flexibility, e.g., using a decomposition of a complex polyhedron to convex polyhedra.

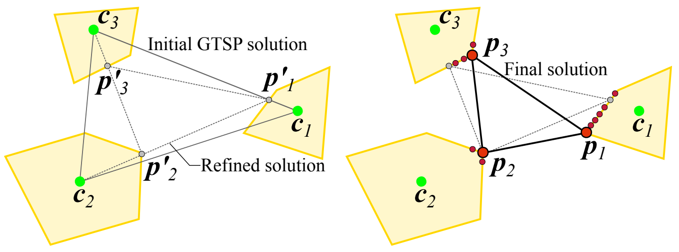
Finally, the hybrid region $Q_{i,k}$ is represented as the ellipsoid defined by (1) with its center $\boldsymbol{c}_{i,k}$ and six half-spaces defined similarly as (2), but with dimensions of the matrix $\mathbf{A}_{i,k}$ (6×3) and correspondingly $\mathbf{b}_{i,k}$ with the dimensions (6×1). For the hybrid region, we consider $\boldsymbol{c}_{i,k}$ is inside the region and satisfies both equations for the ellipsoid and also for six half-spaces, which is assured for all instances of the GTSPN benchmarks introduced in [11].

The centroid $\boldsymbol{c}_{i,k}$ of each region, which is inside the region itself, is assumed in the proposed heuristics for determination of suitable waypoint locations inside $Q_{i,k} \in S_i$ in the proposed solutions to the GTSPN. Besides, it is assumed the regions are convex, but they can mutually overlap. The assumptions on the centroid and convexity are satisfied for all the addressed instances reported in [11]. Besides, the proposed methods would also work for any point inside the region instead of its centroid $\boldsymbol{c}_{i,k}$, but we can expect a decrease in the solution quality. However, it is not considered an issue, as the proposed solutions are intended to be a fast construction heuristic.

[1]There is a mistake in (7) presented in [11] and it should be as the herein presented (2) also regarding the provided GTSPN instances [28].

III. Decoupled Approach to the GTSPN

The main idea of the decoupled approach is to determine the sequence of visits to the regions prior determination of the waypoint locations for the found sequence. Since the neighborhood sets consist of a set of convex regions, the centroids of the regions can be used as possible locations of visits and the sequence can be determined by solving the Euclidean GTSP [14]. Such a solution of the GTSP for the centroids of the regions is a rough approximation of the original GTSPN because it unnecessarily enters the regions. Therefore, we can improve the solution by post-processing procedure as follows.



(a) Initial GTSP and refined solutions (b) Iterative solution adjustment

Fig. 3. Principle of the decoupled approach with post-processing optimization. For simplicity, we consider centroids of the GTSP solution be $\boldsymbol{c}_1$, $\boldsymbol{c}_2$, and $\boldsymbol{c}_3$. The initial solution is a tour connecting the centroids from which the point $\boldsymbol{p}'_1$ can be determined at the boundary of the first region and the segment $(\boldsymbol{c}_3, \boldsymbol{c}_1)$. Similarly, $\boldsymbol{p}'_2$ at the boundary of the second region can be determined using $(\boldsymbol{p}'_1, \boldsymbol{c}_2)$, etc. The sequence $(\boldsymbol{p}'_1, \boldsymbol{p}'_2, \boldsymbol{p}'_3)$ forms a refined solution which can be further iteratively adjusted by a local optimization.

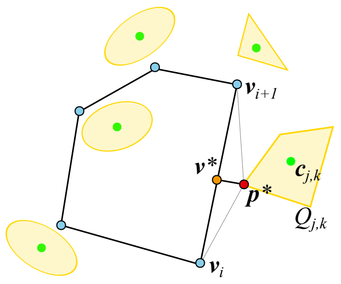
First, we can determine a refined solution using intersections of the regions with the straight line segments connecting the centroids, see Fig. 3a. Then, we can locally adjust the waypoint locations at the boundary of the regions using the hill-climbing technique, which is schematically visualized in Fig. 3b. Hence, each waypoint location in the tour is iteratively optimized to shorten the tour until the sampling step of the hill climbing optimization is negligibly small, e.g., less 10^{-10}. The locations are optimized locally, and thus all the regions in the sequence are consecutively examined, and the whole tour is processed several times, e.g., 10 000 times.

The GLKH [29] solver for the GTSP can be used to find a feasible solution of the GTSPN as a tour connecting centroids of the selected regions. We refer such a straightforward approach as Centroids-GTSP. Then, the solution can be further improved by the described post-processing procedure denoted as `pp_optimization` and the whole decoupled approach is denoted Centroids-GTSP+. Due to the convexity of the regions, the local optimization quickly converges and based on the empirical evaluation; the complete post-processing procedure is computationally efficient.

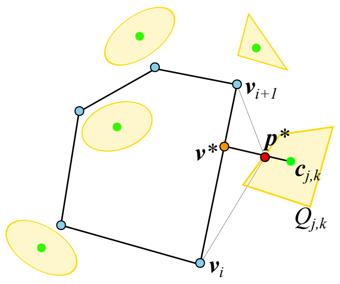
IV. Proposed GSOA-based Solution to the GTSPN

The proposed heuristic solution to the GTSPN is based on the Growing Self-Organizing Array (GSOA) for routing problems [24]. It is a growing array of nodes that is iteratively adapted towards the neighborhood sets $\mathcal{S}$ in a finite number of learning epochs. The number of nodes M in the array $\mathcal{N} = \{\nu_1, \ldots, \nu_M\}$ is varying during the adaptation as new nodes are added to the array, and nodes from the previous epoch are removed. Each node $\nu_i \in \mathcal{N}$ is associated to the node location $\boldsymbol{\nu}_i$ in the input space $\boldsymbol{\nu}_i \in \mathbb{R}^3$. The following notation is used to distinguish regions, sets, and locations that are associated with the particular node ν_i. The covered region $Q_{j,k}$ of the neighborhood set S visited at the waypoint location of ν_i is $\nu_i.Q_{j,k}$ and $\nu_i.Q_{j,k} \in \nu_i.S$. The waypoint location at which $\nu_i.S$ is visited is denoted $\nu_i.\boldsymbol{p}$. The array defines a sequence of visits Σ to $\mathcal{S}$, and the associated waypoint locations to the nodes $\mathcal{N}$ form P. The connected node locations $\boldsymbol{\nu} \in \mathcal{N}$ form a closed path that evolves in $\mathbb{R}^3$ to visit all the neighborhood sets $\mathcal{S}$, and the array converges to a stable state where nodes fit their waypoint locations [24].

During the adaptation of $\mathcal{N}$ to each $S_j \in \mathcal{S}$, a new node ν^* is determined together with the corresponding waypoint location to visit S_j. The new node ν^* is determined from the shortest distance of the path defined by the connected node locations of the array $\mathcal{N}$ to S_j. The two points defining the shortest distance are a new node location $\boldsymbol{\nu}^*$ on the path and the corresponding waypoint location $\boldsymbol{p}^*$, which is at the boundary of S_j or it is identical to $\boldsymbol{\nu}^*$ if the path intersect S_j. Since the path created from the node locations $\mathcal{N}$ consists of the sequence of straight line segments defined by the consecutively connected nodes $(\boldsymbol{\nu}_i, \boldsymbol{\nu}_{i+1})$, the new node ν^* is inserted to the array between the nodes ν_i and ν_{i+1} with the position set to $\boldsymbol{\nu}^*$, see Fig. 4.



(a) Shortest distance between the path defined by $\mathcal{N}$ and $Q_{j,k}$



(b) Proposed heuristic based on the shortest distance to the centroid $\boldsymbol{c}_{j,k}$

Fig. 4. Principle of the new node determination using shortest distance of the path (formed from the array $\mathcal{N}$) to the region $Q_{j,k}$. For better readability, a 2D case is shown as a projection of the 3D regions. In the GSOA, the new node ν^* is determined together with the waypoint location $\boldsymbol{p}^*$ using the shortest distance between $\mathcal{N}$, which forms a sequence of straight line segments $(\boldsymbol{\nu}_i, \boldsymbol{\nu}_{i+1})$, and a particular region. The shortest distance can be quickly determined in 2D, but it is not trivial in 3D cases. Therefore, a new node location $\boldsymbol{\nu}^*$ is found as the closest point of segments $(\boldsymbol{\nu}_i, \boldsymbol{\nu}_{i+1})$ to $\boldsymbol{c}_{j,k}$, which is always inside $Q_{j,k}$. Then, the segment defined by $(\boldsymbol{\nu}^*, \boldsymbol{c}_{j,k})$ is used to determine $\boldsymbol{p}^*$, see Fig. 5.

The deployment of the GSOA [24] to the solution of the GTSPN is relatively straightforward and the main challenge is in computationally efficient determination of the shortest distances between the path defined by the array of nodes $\mathcal{N}$ and the neighborhood sets $\mathcal{S}$. In the solution of the CETSP [24], the neighborhood S_j is a disk with the radius δ_i, and therefore, the location of the new node $\boldsymbol{\nu}^*$ and its corresponding $\boldsymbol{p}^*$ can be found analytically using equations for lines and circle, which is fast and straightforward to compute. However, it is not the case of 3D regions defined as ellipsoids and polyhedra. Therefore, we propose simple heuristic overviewed in Fig. 5 that is based on the centroids of the individual regions. Besides, each neighborhood set $S_j \in \mathcal{S}$ consists of multiple regions $Q_{j,k} \in S_j$, and thus the regions are iteratively examined and the region with the shortest distance $\|(\boldsymbol{\nu}^*, \boldsymbol{p}^*)\|$ is considered to be visited using the corresponding new node ν^* and its waypoint location $\nu^*.\boldsymbol{p}$ becomes the determined $\boldsymbol{p}^*$. Based on the performed empirical evaluation, the heuristic show to be computationally efficient while yielding competitive solutions to the approach [11], which is reported in Section V.

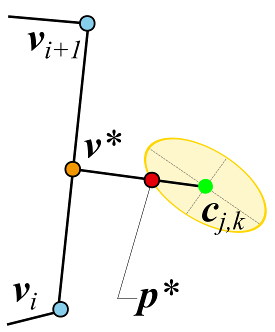
Algorithm 1: GSOA for solving GTSPN with 3D regions

Input: $\mathcal{S} = \{S_1, \ldots, S_n\}$ – a set of 3D regions
Input: c_{max} – the number of learning epochs
Output: (Σ, P) – Σ sequence of visits to $S_i \in \mathcal{S}$ and corresponding waypoint locations P

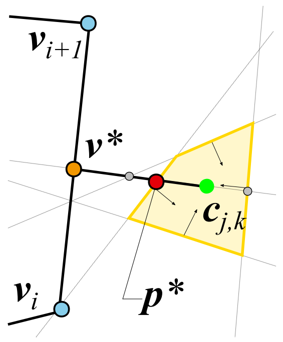
```
1  $c \leftarrow 0$    // set the learning epoch counter
2  $\mathcal{N} \leftarrow \{\nu_1\}$, $\boldsymbol{\nu}_1$ is a geometric center of all regions in $\mathcal{S}$
3  $\sigma \leftarrow 10$; $\alpha \leftarrow 0.0005$; $\mu \leftarrow 0.6$   // init. learning params
4  $c_{max} \leftarrow \min(c_{max}, 1/\alpha)$   // ensure $\sigma$ will be above 0
5  while $c \le c_{max}$ and termination cond. is not satisfied do
6      foreach $S_j$ in a random permutation of $\mathcal{S}$ do
7          $(\boldsymbol{\nu}^*, \boldsymbol{p}^*, \nu_i, \nu_{i+1}) \leftarrow$ new_node$(\mathcal{N}, S_j)$
8          $\mathcal{N} \leftarrow$ insert_node$(\boldsymbol{\nu}^*, \nu_i, \nu_{i+1}, \mathcal{N})$
9          $\mathcal{N} \leftarrow$ adapt$(\mu, \sigma, \nu^*, \boldsymbol{p}^*, \mathcal{N})$
10     $c \leftarrow c + 1$   // update the epoch counter
11     $\mathcal{N} \leftarrow$ Remove all nodes from the epoch $c - 1$
12     $\sigma \leftarrow (1 - c \cdot \alpha)\sigma$   // decrease the learning gain
13     $(\Sigma', \mathcal{Q}', P') \leftarrow$ Traverse $\mathcal{N}$, determine $\Sigma'$, a sequence of
       the visited regions $\mathcal{Q}'$, and $P'$ using the node waypoints
14     if $c = 1$ or $L(\Sigma', P') \le L(\Sigma, P)$ then
15         $(\Sigma, \mathcal{Q}, P) \leftarrow (\Sigma', \mathcal{Q}', P')$   // update the solution
16 $(\Sigma, P) \leftarrow$ two_opt$(\Sigma, P)$   // call Two-opt heuristic [30]
17 $P \leftarrow$ pp_optimization$(\Sigma, \mathcal{Q}, P)$   // improve the solution
18 return $(\Sigma, P)$
```

A summary of the GSOA learning algorithm for the GTSPN is depicted in Algorithm 1. The learning starts with the initialization of the array $\mathcal{N}$ using a node ν_1 placed at the geometric center of the regions, but it can also be randomly set to any point within the bounding box of $\mathcal{S}$. The learning parameters are the gain decreasing rate α, the learning rate μ, and the initial value of the learning gain σ that are set to the values recommended in [24] that originate from the evaluation study [31]. The maximal number of learning epoch is adjusted according to α to avoid negative value of the learning gain σ because it is decreased after each learning epoch (see Line 12, Algorithm 1).

The algorithm continues with the iterative adaptation of $\mathcal{N}$ to $\mathcal{S}$ for up to c_{max} learning epochs. In each learning epoch, a new node ν^* is determined (Line 7, Algorithm 1) together with the respective waypoint location $\boldsymbol{p}^*$ for each $S_j \in \mathcal{S}$, which



(a) Ellipsoid



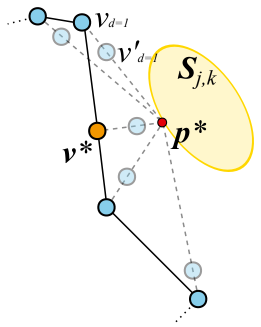
(b) Polyhedron

Fig. 5. Proposed determination of the waypoint location for ellipsoid and polyhedron (defined by a set of half-spaces) regions, which are visualized in 2D for better readability. The centroid $\boldsymbol{c}_{j,k}$ is always inside the region, and therefore, it is used to determine the new node location $\boldsymbol{\nu}^*$ at the shortest distance from $\mathcal{N}$ to $\boldsymbol{c}_{j,k}$, i.e., using an iterative examination of all segments defined by two nodes $(\boldsymbol{\nu}_i, \boldsymbol{\nu}_{i+1})$ from $\mathcal{N}$. For the ellipsoid, (1) is used to examine if $\boldsymbol{\nu}^*$ is already inside the ellipsoid; otherwise, the waypoint location $\boldsymbol{p}^*$ is determined as a new point on $(\boldsymbol{\nu}^*, \boldsymbol{c}_{j,k})$ for which the left side of (1) is equal to one. For the polyhedron, intersection points of $(\boldsymbol{\nu}^*, \boldsymbol{c}_{j,k})$ with all half-spaces are found and $\boldsymbol{p}^*$ is the intersection point that is closest to $\boldsymbol{\nu}^*$ while still inside the polyhedron $Q_{j,k}$, i.e., it satisfies (2). The point $\boldsymbol{p}^*$ becomes the waypoint location $\nu^*.\boldsymbol{p}$ associated with the new node ν^* inserted into $\mathcal{N}$ between ν_i and ν_{i+1} with the location set to $\boldsymbol{\nu}^*$. The unused intersection points are shown as small gray disks.

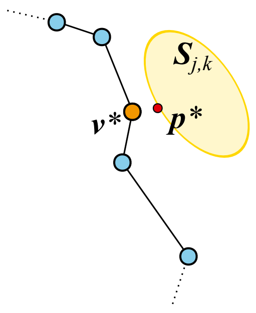
is selected from the neighborhood sets in a random order to avoid possible local extremes. The new node ν^* is inserted to $\mathcal{N}$ between the corresponding nodes ν_i and ν_{i+1} (Line 8, Algorithm 1), see Fig. 5 for the relation of the nodes. The array $\mathcal{N}$ becomes M nodes long and ν^* is adapted towards its $\nu^*.\boldsymbol{p}$ (Line 9, Algorithm 1). The adaptation is not performed only for the winner node ν^* but also for its neighboring nodes that are in the distance $d < 0.2M$ from ν^* in the array (i.e., d is counted in the number of nodes in $\mathcal{N}$); and the node locations are updated according to

$$\boldsymbol{\nu} \leftarrow \boldsymbol{\nu} + \mu f(\sigma, d)(\nu^*.\boldsymbol{p} - \boldsymbol{\nu}) \qquad (3)$$

with decreasing power for the farther nodes from ν^* using the neighboring function $f(\sigma, d) = \exp(-d^2/\sigma^2)$. The adaptation is schematically visualized in Fig. 6.



(a) Winner node and its neighboring nodes before adaptation



(b) Adapted nodes towards the waypoint location $\boldsymbol{p}^*$

Fig. 6. Visualization of the adaptation of the winner node ν^* and its neighboring nodes towards the determined waypoint location $\nu^*.\boldsymbol{p} = \boldsymbol{p}^*$.

After adaptation to all neighborhood sets, only the newly added nodes to $\mathcal{N}$ are preserved (Line 11, Algorithm 1) and the array has exactly n nodes, each with the associated waypoint location. Therefore, a feasible solution can be extracted by traversing $\mathcal{N}$. The adaptation is repeated for the next learning epoch until a stable solution is found or c_{max} is reached. A solution is considered stable if nodes are negligibly close to their waypoint locations, e.g., $\|(\boldsymbol{\nu}, \nu.\boldsymbol{p})\| \leq 10^{-4}$ for all $\nu \in \mathcal{N}$. The final solution found as the best solution from the performed epochs is optimized by the Two-opt heuristic [30] and further improved by the post-processing procedure (Line 17, Algorithm 1) described in Section III.

A. Approximate Shortest Distance for the GSOA Learning

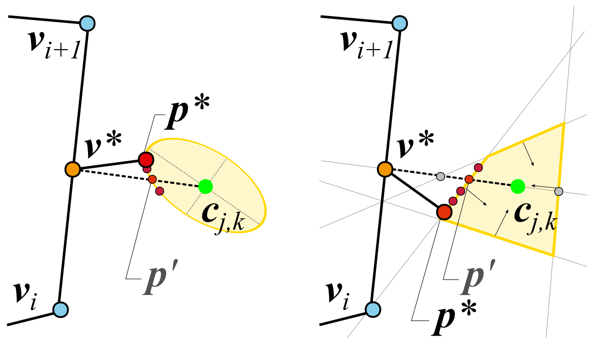
The proposed approach is intended to be a fast constructive heuristic to find a feasible solution quickly. The computational times reported in [24] for the CETSP are in milliseconds, and we have been motivated to achieve similar times also for the herein addressed 3D instances of the GTSPN. The learning procedure is identical, and the only change is in the increased dimension of the node and waypoint locations. In addition to the number of regions because of the neighboring sets, the most important part is the determination of the new node and its waypoint which depends on the shape of the regions. Here, we can exploit the convexity of the region $Q_{j,k}$ and use the centroid $\boldsymbol{c}_{j,k}$ to determine the closest point of $\mathcal{N}$ to $Q_{j,k}$ as the location $\boldsymbol{\nu}^*$. Notice, if $\boldsymbol{\nu}^*$ is already inside $Q_{j,k}$, its waypoint is set to $\nu^*.\boldsymbol{p} = \boldsymbol{\nu}^*$. Otherwise, we distinguish the cases according to the type of the region as follows.

For the ellipsoid, the segment $(\boldsymbol{\nu}^*, \boldsymbol{c}_{j,k})$ is used to determine $\boldsymbol{p}^*$ on the segment and also be inside the ellipsoid. Let λ be the left side of (1); then for $\lambda \leq 1$, $\boldsymbol{\nu}^*$ is already inside the ellipsoid. Otherwise, $\boldsymbol{p}^*$ is determined as a point on $(\boldsymbol{\nu}^*, \boldsymbol{c}_{j,k})$ that is at the distance $1/\lambda$ using (1) from $\boldsymbol{c}_{j,k}$, which is certainly inside the ellipsoid because $\lambda > 1$, and thus $1/\lambda \leq 1$. For the polyhedron, intersection points of $(\boldsymbol{\nu}^*, \boldsymbol{c}_{j,k})$ with the polyhedron's half-spaces are computed. The waypoint is chosen as the closest intersection point to $\boldsymbol{\nu}^*$ that is inside the polyhedron according to (2), see Fig. 5b. Finally, both methods are combined for the hybrid regions, and $\boldsymbol{p}^*$ is the closest point to $\boldsymbol{\nu}^*$ that satisfies (1) and (2).

B. Improving Shortest Distance Approximation

The shortest distance between a straight line segment and a 3D region may not necessarily be part of the connection of the segment and centroid of the region. However, the initial rough approximation of the shortest distance (visualized in Fig. 5) can be further improved by a local optimization to find a more suitable location on the boundary of the region. The approach is similar to the post-processing procedure (Section III), and it is schematically visualized in Fig. 7.

The initial step size is 0.1 and the waypoint location is iteratively updated to shorten the distance between $\boldsymbol{\nu}^*$ and $\boldsymbol{p}'$. The optimization is terminated for $\boldsymbol{p}'$ with the shortest distance, and the step size is lower than 10^{-10}. The shape of the regions is convex, and thus the algorithm quickly converges to a local extreme. A real impact of the algorithm performance is reported in the next section.



(a) Ellipsoid (b) Polyhedron

Fig. 7. Improved shortest distance approximation by a local search. The improvement is demonstrated for the cases visualized in Fig. 5. New locations (small red disks) are iteratively sampled around the candidate waypoint location using sampling step with decreasing size.

Notice, the proposed improvement adjusts only $\boldsymbol{p}^*$ and the winner node location $\boldsymbol{\nu}^*$ is fixed. A further improvement of the shortest distance approximation can also adjust the winner node location at the cost of more demanding algorithm. However, we consider such optimization out of the scope of this paper, and we dedicate it for future work.

C. Computational Complexity

The computational complexity of the proposed GSOA-based algorithm depends on the number of nodes and number of regions as each region has to be considered individually. Let n be the number of neighborhood sets and the maximal number of the regions in any set be n_r. A region can consist of polyhedra, each defined by up to n_h half-spaces. The number of nodes can be bounded by $O(2n)$ because one node is created for each set in every learning epoch, and only new nodes are preserved for the next epoch. Therefore, a new node is determined in less than $2n$ examination of the segments of $\mathcal{N}$, each examined for n_r regions with possible n_h half-spaces, which gives $O(2nn_rn_h)$ and a single learning epoch can be bounded by $O(n^2n_rn_h)$. The number of learning epochs is fixed, and the post-processing procedure depends on n, and Two-opt heuristic can be bounded by $O(n^2)$. Furthermore, we can consider that each set consists of a fixed number of regions, and also the regions may consist of a fixed maximal number of half-spaces, and thus we can consider $n \gg n_r$ and $n \gg n_h$, and the complexity of the proposed algorithm can be bounded by $O(n^2)$.

The local optimization described in Section IV-B is performed in a fixed number of steps because of the convex shape of the regions, but it is expected to be more demanding than the simple heuristic from Section IV-A. The real computational requirements are reported in Section V.

V. Results

The proposed decoupled and GSOA-based heuristics have been empirically evaluated in 3D instances of the GTSPN proposed in [11]. We evaluate 30 instances with the number of neighborhood sets $n \in \{30, 35, 40, 45, 50\}$, where each set consists of six regions of variate type (ellipsoid, polyhedra, and hybrid) [28]. To the best of the authors' knowledge, the only existing GTSPN solver capable of solving the addressed instances is the HRKGA proposed in [11], and therefore, we consider the results reported therein for a comparison.

The GLKH [29] used for the solution of the GTSP in the decoupled heuristic and also the GSOA are randomized algorithms, and thus we solve every problem instance 50 times by each method. The GLKH is used with the default options, but with a random seed and only with a single run per individual solution. We distinguish Centroids-GTSP and Centroids-GTSP$^+$ to report on the performance of improving post-processing procedure and its computational requirements. The recommended values of the GSOA learning parameters are used, but we further employ the local optimization of the waypoint location from Section IV-B which is denoted GSOA-OPT. Thus, four proposed methods are evaluated.

The quality of the found solutions is measured as the percentage deviation from the reference solution of the best solution value among the performed trials as PDB $= (L - L_{ref})/L_{ref} \cdot 100\%$, where L_{ref} is the best-known solution of the particular problem instance, and L is the length of the best solution found among the performed trials. Besides, the robustness of the algorithm is measured as the percentage deviation from the reference of the mean solution value over the performed trials that is computed as PDM $= (L_{avg} - L_{ref})/L_{ref} \cdot 100\%$, where L_{avg} is the average solution length among the performed trials. Notice that zero value of the PDB indicates the particular method provides the best-known solution.

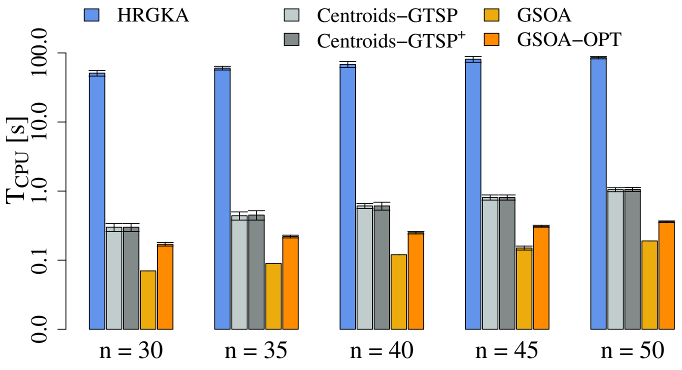


Fig. 8. Average computational requirements of the evaluated methods.

The GLKH is implemented in C [29], and the proposed post-processing procedure and GSOA have been implemented in C++ and they are made available [32]. All the algorithms have been compiled by the Clang version 6.0.1 with the -O3 and -march=native flags, and all the reported results have been obtained using a single core of Intel i7-6700K CPU running at 4.0 GHz. In [11], the authors note the reported results for the HRKGA have been computed using Intel Xeon CPU running at 3.2 GHz, which is supposed to be Intel Xeon E3-1225 v3. Therefore we normalized the computational times regarding [33] that indicates the proposed methods have been run on about 1.17 faster computer for a single thread rating. Hence, the computational times reported in [11] are divided by 1.17. The achieved results for the 3D instances of the

TABLE I
COMPUTATIONAL RESULTS FOR 3D INSTANCES OF THE GTSPN

Set	L_{ref}	HRGKA [11]			**Centroids-GTSP**			**Centroids-GTSP⁺**			**GSOA**			**GSOA-OPT**		
		PDB	PDM	T_{CPU}	PDB	PDM	T_{CPU}	PDB	PDM	T_{CPU}	PDB	PDM	T_{CPU}	PDB	PDM	T_{CPU}
3D_30_6_a	**3 540.55**	0.57	1.06	45.1	3.52	3.67	0.357	**0.00**	0.27	0.358	0.03	4.08	0.081	0.57	3.87	0.196
3D_30_6_b	**3 856.31**	0.60	1.14	43.9	3.97	4.37	0.329	0.06	0.51	0.330	0.08	3.06	0.085	**0.00**	3.05	0.211
3D_30_6_c	**3 687.14**	0.74	1.46	47.6	3.68	3.83	0.332	**0.00**	0.14	0.333	0.35	1.34	0.086	0.54	1.55	0.200
3D_30_6_d	**3 598.01**	0.27	0.77	34.7	5.79	5.96	0.357	0.37	0.53	0.357	0.39	3.64	0.082	**0.00**	3.89	0.199
3D_30_6_e	**3 543.01**	1.42	1.97	44.6	4.21	4.67	0.317	0.24	0.54	0.318	0.01	1.94	0.083	**0.00**	2.47	0.217
3D_30_6_f	**3 611.44**	0.38	1.09	45.5	5.08	5.47	0.416	0.20	0.65	0.417	0.19	2.12	0.082	**0.00**	1.97	0.196
3D_35_6_a	**4 301.46**	0.40	0.82	56.0	4.35	4.53	0.630	**0.00**	0.28	0.631	0.99	3.39	0.113	0.76	3.14	0.252
3D_35_6_b	**4 174.98**	0.95	1.99	45.3	5.21	5.21	0.422	0.20	0.24	0.423	0.09	3.72	0.105	**0.00**	4.41	0.267
3D_35_6_c	**3 683.72**	0.45	1.20	52.2	4.31	4.38	0.551	**0.00**	0.11	0.554	2.34	6.41	0.113	2.45	6.80	0.261
3D_35_6_d	**4 274.05**	0.68	1.63	52.8	4.22	4.41	0.494	0.03	0.31	0.495	**0.00**	1.42	0.110	0.14	1.18	0.254
3D_35_6_e	**3 881.87**	0.44	1.51	51.8	4.31	4.40	0.494	0.27	0.41	0.516	0.58	6.19	0.113	**0.00**	6.48	0.260
3D_35_6_f	**4 039.00**	0.46	1.93	51.1	5.26	5.56	0.508	**0.00**	0.50	0.509	1.24	2.63	0.110	1.14	2.40	0.280
3D_40_6_a	**4 465.79**	0.53	1.46	48.6	4.68	4.84	0.683	**0.00**	0.32	0.684	0.39	2.08	0.139	0.38	2.06	0.298
3D_40_6_b	**4 124.42**	1.26	2.01	68.5	5.11	5.50	0.669	0.09	0.53	0.671	**0.00**	1.73	0.140	**0.00**	1.54	0.298
3D_40_6_c	**4 354.23**	0.59	1.36	56.2	4.32	4.89	0.768	**0.00**	0.57	0.769	0.51	3.53	0.149	1.23	4.12	0.310
3D_40_6_d	**4 530.89**	0.43	1.01	61.0	4.66	5.06	0.773	**0.00**	0.60	0.814	1.55	3.86	0.145	1.64	4.00	0.306
3D_40_6_e	**4 085.19**	0.81	1.37	58.7	4.38	4.51	0.677	**0.00**	0.08	0.678	0.53	5.57	0.140	2.35	6.27	0.294
3D_40_6_f	**3 976.35**	1.52	2.21	57.4	6.25	6.49	0.688	**0.00**	0.42	0.690	1.66	2.73	0.142	1.64	2.58	0.283
3D_45_6_a	**4 450.18**	1.15	2.23	59.1	5.05	5.87	0.960	**0.00**	0.89	0.963	1.72	4.02	0.175	0.88	3.76	0.348
3D_45_6_b	**4 873.62**	0.95	1.50	77.4	3.90	4.05	0.874	0.10	0.20	0.875	0.06	4.24	0.184	**0.00**	4.47	0.373
3D_45_6_c	**4 784.13**	0.63	1.36	69.2	4.71	5.22	1.048	**0.00**	0.51	1.049	0.66	2.39	0.173	0.23	2.90	0.352
3D_45_6_d	**4 810.52**	1.37	2.07	63.0	4.25	4.37	0.947	**0.00**	0.20	0.948	1.59	4.18	0.180	0.91	3.92	0.375
3D_45_6_e	**4 867.92**	1.46	2.19	69.7	4.10	4.54	0.908	0.04	0.55	0.909	**0.00**	3.88	0.181	**0.00**	4.32	0.359
3D_45_6_f	**4 674.98**	1.71	2.43	77.1	4.77	5.56	0.935	**0.00**	1.03	0.937	0.85	4.03	0.180	0.67	4.09	0.362
3D_50_6_a	**4 904.12**	1.40	2.71	72.3	5.18	5.47	1.250	**0.00**	0.32	1.251	0.48	3.71	0.216	0.33	3.56	0.419
3D_50_6_b	**4 957.25**	2.47	4.06	72.0	5.57	6.02	1.245	0.11	0.97	1.248	**0.00**	0.91	0.223	0.03	1.00	0.417
3D_50_6_c	**4 870.90**	1.41	2.33	79.7	4.68	5.18	1.238	**0.00**	0.37	1.239	0.19	4.40	0.227	1.45	4.12	0.419
3D_50_6_d	**4 542.25**	0.59	2.00	71.7	4.78	5.14	1.219	**0.00**	0.68	1.220	2.00	3.46	0.228	2.12	3.37	0.433
3D_50_6_e	**5 323.22**	0.84	1.58	71.8	4.59	5.32	1.260	**0.00**	0.88	1.262	2.09	4.99	0.218	2.01	4.84	0.416
3D_50_6_f	**4 917.35**	1.67	2.35	71.5	5.28	5.76	1.187	**0.00**	0.53	1.190	1.49	3.24	0.218	0.91	3.17	0.432
Average values		*0.94*	*1.76*	*59.2*	*4.67*	*5.01*	***0.751***	***0.06***	***0.47***	***0.755***	***0.74***	*3.43*	***0.147***	***0.75***	*3.51*	***0.310***

All the required computational times T_{CPU} are in seconds.

GTSPN are reported in Table I, where the new best results are highlighted in bold in the column L_{ref}, and average performance indicators better than the HRKGA are highlighted in the last row. An overview of the computational requirements is depicted in Fig. 8.

The results indicate that both proposed algorithms are vital heuristics and provide better or competitive solutions to the HRKGA but with significantly lower computational requirements. In all cases, new best solutions are found mostly by the Centroids-GTSP⁺, but in few cases also by the GSOA-based approaches. Regarding the average solution quality, the Centroids-GTSP⁺ provides the most stable solutions with the path length less than 1.0% longer than L_{ref}.

Solutions of the GTSP are weak, which is not surprising as the path connecting the centroids is unnecessarily long. However, the proposed post-processing procedure improves the solution significantly while it costs no more than three milliseconds. Thus it seems to be efficient and effective.

Although implementation details about the HRKGA are not described in [11], both the decoupled and GSOA-based approaches are up to two orders of magnitude faster. Considering the reported times in [11], a solution of the 50 trials by the GSOA is less demanding, and thus a solution of the similar or better quality than the HRKGA can be found by the proposed GSOA with competitive computational time.

Regarding the solution quality of the GSOA, it is known that unsupervised learning approaches to the TSP provide about 3-5% worse solution than the optimum [31]. Thus, we can assume that instances, where the GSOA and GSOA-OPT provide the best results, need to be solved directly as the GTSPN because the decoupled approach is stuck at a local extreme. It is mainly because of close neighborhood sets, and it would be more evident for instances with overlapping neighborhoods. The improvement of the shortest distance approximation based on the local optimization of the waypoint location noticeably improves the solution quality while it is only about two times more demanding. Hence, the proposed method provides a groundwork for further optimization of the shortest distances and using the GSOA-based approach as a quick constructive heuristic for solving the GTSPN, especially for dense instances with overlapping neighborhoods.

VI. CONCLUSION

In this paper, we propose two heuristics for the 3D multi-goal path planning formulated as the GTSPN where the particular neighborhoods are sets of 3D regions defined as a combination of ellipsoids and polyhedra. First, we propose a decoupled approach based on the solution of the GTSP that is followed by simple, yet effective, post-processing procedure that significantly improves the solution quality. Besides, we propose to employ the GSOA (an unsupervised learning algorithm for routing problems) to the solution of the 3D instances of the GTSPN. We propose a simple heuristic to determine suitable waypoint locations to visit the regions that is utilized in the online sampling of the regions during the unsupervised learning of the GSOA. Regarding the reported empirical evaluation, the both approaches seem to be suitable heuristics for the addressed variant of the GTSPN, and they provide competitive or better solutions than the HRKGA, but they are up two orders of magnitude faster. The low computational requirements make the proposed approaches suitable choice for fast construction heuristic to quickly find a feasible (and relatively high-quality) solutions of the GTSPN, which can be further improved by additional optimizations. On the other hand, the HRKGA has also been employed in a solution of the 7D instances of the GTSPN motivated by applications of robotic manipulators, while the proposed heuristics exploit properties of the 3D instances with convex regions. Although non-convex regions can be eventually split into a set of convex regions, an extension of the proposed approach for high-dimensional problems is a subject of our future work. Moreover, the proposed GSOA to the GTSPN has been motivated to develop a fast construction heuristic for the GTSPN, and therefore, we aim to further improve the solution quality by more optimizations employed directly into GSOA learning.

Acknowledgements

We would like to thank the anonymous reviewers and editor for their valuable comments that lead to a significant improvement of the proposed heuristics, and thus the solution of the addressed GTSPN instances.

References

[1] S. Alatartsev, S. Stellmacher, and F. Ortmeier, "Robotic Task Sequencing Problem: A Survey," *Journal of Intelligent & Robotic Systems*, vol. 80, no. 2, pp. 279–298, Nov. 2015.

[2] D. Applegate, R. Bixby, V. Chvátal, and W. Cook, *The Traveling Salesman Problem: A Computational Study*. Princeton, NJ, USA: Princeton University Press, 2007.

[3] G. Gutin and A. P. Punnen, Eds., *The Traveling Salesman Problem and Its Variations*. Springer US, 2007.

[4] F. Suárez-Ruiz, T. S. Lembono, and Q.-C. Pham, "Robotsp – a fast solution to the robotic task sequencing problem," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 1611–1616.

[5] P. Oberlin, S. Rathinam, and S. Darbha, "Today's traveling salesman problem," *IEEE Robotics & Automation Magazine*, vol. 17, no. 4, pp. 70–77, Dec 2010.

[6] S. Alatartsev, M. Augustine, and F. Ortmeier, "Constricting insertion heuristic for traveling salesman problem with neighborhoods," in *International Conference on International Conference on Automated Planning and Scheduling (ICAPS)*, 2013, pp. 2–10.

[7] J. Faigl and G. A. Hollinger, "Autonomous data collection using a self-organizing map," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 5, pp. 1703–1715, May 2018.

[8] M. Dunbabin and L. Marques, "Robots for Environmental Monitoring: Significant Advancements and Applications," *IEEE Robotics & Automation Magazine*, vol. 19, no. 1, pp. 24–39, Mar. 2012.

[9] Bo Yuan, M. Orlowska, and S. Sadiq, "On the Optimal Robot Routing Problem in Wireless Sensor Networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 9, pp. 1252–1261, 2007.

[10] I. Gentilini, F. Margot, and K. Shimada, "The travelling salesman problem with neighbourhoods: MINLP solution," *Optimization Methods and Software*, vol. 28, no. 2, pp. 364–378, 2013.

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

[12] A. Kovács, "Integrated task sequencing and path planning for robotic remote laser welding," *International Journal of Production Research*, vol. 54, no. 4, pp. 1210–1224, Feb. 2016.

[13] G. Laporte, A. Asef-Vaziri, and C. Sriskandarajah, "Some applications of the generalized travelling salesman problem," *Journal of the Operational Research Society*, vol. 47, no. 12, pp. 1461–1467, 1996.

[14] K. Helsgaun, "Solving the equality generalized traveling salesman problem using the lin-kernighan-helsgaun algorithm," *Mathematical Programming Computation*, vol. 7, no. 3, pp. 269–287, Sep 2015.

[15] S. L. Smith and F. Imeson, "GLNS: An effective large neighborhood search heuristic for the generalized traveling salesman problem," *Computers & Operations Research*, vol. 87, no. C, pp. 1–19, Nov. 2017.

[16] C. E. Noon and J. C. Bean, "An efficient transformation of the generalized traveling salesman problem," *INFOR: Information Systems and Operational Research*, vol. 31, no. 1, pp. 39–44, 1993.

[17] M. de Berga, J. Gudmundssonb, M. J. Katzc, C. Levcopoulosd, M. H. Overmarse, and A. F. van der Stappen, "TSP with neighborhoods of varying size," *Journal of Algorithms*, vol. 57, no. 1, pp. 22–36, 2005.

[18] A. Dumitrescu and J. S. B. Mitchell, "Approximation algorithms for TSP with neighborhoods in the plane," *Journal of Algorithms*, vol. 48, no. 1, pp. 135–159, Aug. 2003.

[19] H. Jonsson, "The Traveling Salesman Problem for lines in the plane," *Information Processing Letters*, vol. 82, no. 3, pp. 137–142, May 2002.

[20] A. Antoniadis, K. Fleszar, R. Hoeksma, and K. Schewior, "A PTAS for Euclidean TSP with Hyperplane Neighborhoods," *arXiv:1804.03953 [cs]*, Apr. 2018. [Online]. Available: http://arxiv.org/abs/1804.03953

[21] D. J. Gulczynski, J. W. Heath, and C. C. Price, *The Close Enough Traveling Salesman Problem: A Discussion of Several Heuristics*. Boston, MA: Springer US, 2006, pp. 271–283.

[22] J. Dong, N. Yang, and M. Chen, *Heuristic Approaches for a TSP Variant: The Automatic Meter Reading Shortest Tour Problem*. Boston, MA: Springer US, 2007, pp. 145–163.

[23] W. K. Mennell, "Heuristics for solving three routing problems: Close-enough traveling salesman problem, close-enough vehicle routing problem, sequence-dependent team orienteering problem," Ph.D. dissertation, University of Maryland, 2009.

[24] J. Faigl, "GSOA: growing self-organizing array - unsupervised learning for the close-enough traveling salesman problem and other routing problems," *Neurocomputing*, vol. 312, pp. 120–134, 2018.

[25] K. Elbassioni, A. V. Fishkin, and R. Sitters, "Approximation algorithms for the Euclidean traveling salesman problem with discrete and continuous neighborhoods," *International Journal of Computational Geometry & Applications*, vol. 19, no. 02, pp. 173–193, 2009.

[26] I. Gentilini, "Multi-goal path optimization for robotic systems with redundancy based on the traveling salesman problem with neighborhoods," Ph.D. dissertation, Carnegie Mellon University, 2012.

[27] K. Vicencio, T. Korras, K. A. Bordignon, and I. Gentilini, "Energy-optimal path planning for six-rotors on multi-target missions," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 2481–2487.

[28] K. Vicencio, "Randomly generated GTSPN instances," 2014, [cited on 2018-Sep-9]. [Online]. Available: http://robotics.pr.erau.edu/data/gtspn.zip

[29] K. Helsgaun, "GLKH," 2013, [cited on 2018-Dec-5]. [Online]. Available: http://akira.ruc.dk/~keld/research/GLKH/

[30] G. A. Croes, "A method for solving traveling-salesman problems," *Operations Research*, vol. 6, no. 6, pp. 791–812, 1958.

[31] J. Faigl, "On the performance of self-organizing maps for the non-euclidean traveling salesman problem in the polygonal domain," *Information Sciences*, vol. 181, pp. 4214–4229, October 2011.

[32] J. Faigl, P. Váňa, and J. Deckerová, "Fast heuristics for GTSPN-based 3d multi-goal path planning," [cited on 2018-Dec-9]. [Online]. Available: https://purl.org/comrob/sw

[33] PassMark® Software Pty Ltd, "Cpu performance comparison," 2018, [cited on 2018-Sep-9]. [Online]. Available: https://www.cpubenchmark.net/compare.php?cmp[]=1993&cmp[]=2565