Traveling Salesman Problem with Neighborhoods on a Sphere in Reflectance Transformation Imaging Scenarios

Jindřiška Deckerová[a], Jan Faigl[a,*], Vít Krátký[a]

[a] *Czech Technical University in Prague, Faculty of Electrical Engineering Technická 2, 166 27, Prague, Czech Republic*

Keywords: Unsupervised Learning, Traveling Salesman Problem, Spherical Geometry

1. Introduction

The Traveling Salesman Problem (TSP), a well-known NP-hard combinatorial optimization problem, stands to determine the shortest tour visiting a given set of target locations (Applegate et al., 2007). In many practical scenarios, the tour cost can be significantly reduced for cases where it is sufficient to visit a close neighborhood of the target locations (Faigl, 2018). In such a case, the TSP becomes the TSP with Neighborhoods (TSPN) (Arkin and Hassin, 1994) that combines the combinatorial optimization of the TSP with continuous optimization of finding the optimal locations within the neighborhoods. The TSPN is a challenging problem (Gudmundsson et al., 2000) and specific cases for particular types of neighborhoods are studied (Elbassioni et al., 2009; Gulczynski et al., 2006). Besides, the diversity of the real-world problems leads to variants of the TSP where the input space is not Euclidean (as in the regular formulation of the TSP) but a geometrical

*Corresponding author

Email addresses: deckejin@fel.cvut.cz (Jindřiška Deckerová), faiglj@fel.cvut.cz (Jan Faigl), kratkvit@fel.cvut.cz (Vít Krátký)

object such as cuboid (Aybars, 2008) or sphere (Uğur et al., 2009).

Figure 1: The motivational scenario of the studied TSPNS is arising from the practical deployment of the autonomous RTI by a team of UAVs (Krátký et al., 2020).

In this paper, we address a variant of the non-Euclidean TSP with Neighborhoods on a Sphere, further denoted as the TSPNS. The problem is motivated by practical applications of the Reflectance Transformation Imaging (RTI) using a team of multi-rotor Unmanned Aerial Vehicles (UAVs) (Krátký et al., 2020) as depicted in Fig. 1. The RTI is a computational photography method to capture a surface shape and color of the object of interest and create a model for the interactive re-lighting of the object from any direction using a software view, thus revealing details not visible with the naked eye. The required input of the RTI is a set of images captured by a static camera, where each image needs to be captured under the illumination of the constant intensity from different directions. In the UAV-based RTI, the light can be carried by a UAV that enlightens the object from a set of directions while another UAV holds its position and captures images. Thus, collecting the required images for the RTI by the UAVs can be formulated as a problem of determining a cost-efficient path for the moving vehicle to

enlighten the object from the necessary directions quickly. Since multiple lighting directions are required even for a single object of interest, the problem of finding the best sequence of directions that yields the shortest path possible becomes a variant of the TSP. However, we need to account for the practical requirements of the lighting.

The UAV carrying the light has to maintain a constant distance from the object to satisfy the requirements of the constant intensity of the illumination. Therefore, we formulate the problem as finding the shortest path on the surface of the sphere to visit the locations from which the enlightening of the object forms the desired shadows to be captured by the static UAV. Particular lighting directions for the RTI can be chosen arbitrarily within the defined tolerance interval; hence, we can exploit the tolerance on the lighting directions by introducing target sites as the neighborhood regions for all the former lighting locations. Thus, the travel cost to visit the sites can be saved by finding suitable locations of the regions with respect to (w.r.t.) the sequence of visits to all the required lighting locations. Hence, the problem is formulated as the TSPN on a Sphere (TSPNS), where the neighborhoods are regions on the sphere's surface, and connections between these regions are also on the surface of the sphere.

Since a direct solver for the introduced TSPNS is not available, two ways to address the TSPNS can be applied. First, we can discretize the continuous neighborhoods and transform the problem to the Generalized TSP (GTSP), where regions are substituted with sets of discrete locations. The GTSP is then to determine a cost-efficient path visiting one location per set. Such discretized instances of the TSPNS can be solved by the available GTSP solvers such as (Helsgaun, 2015) or (Smith and Imeson, 2017). However, the transformation method can quickly be too demanding because high-quality

solutions require dense sampling. Therefore, the second approach to solve the TSPNS is to adapt existing direct heuristic approaches to the TSPN. Besides, a heuristic solution found by both approaches can be further improved by a local post-processing optimization similarly as reported for the GTSP with Neighborhoods in (Faigl et al., 2019).

We investigated both mentioned approaches, and we propose to employ unsupervised learning of the Growing Self-Organizing Array (GSOA) to solve the TSPNS directly. The reported results indicate that the proposed adaptation of the GSOA to the spherical space provides solutions that are about units of percentage points worse than about two orders of magnitude more demanding GTSP-based transformation approach with dense sampling. On the other hand, using a single sample per region, the corresponding discretized instance of the TSPNS can be solved using the efficient TSP heuristic (Helsgaun, 2000) in a competitive time to the GSOA at the cost of significantly worse solutions. However, the initial solutions provided by the methods can be improved by local post-processing optimization. Based on the presented results, the proposed methods represent a suitable approach for solving instances of the TSPNS motivated by the RTI scenarios with aerial vehicles.

The rest of the text is organized as follows. A brief overview of existing approaches to the introduced TSPN is summarized in Section 2. The addressed TSPNS is formally introduced in Section 3. The transformation approach is presented in Section 4, where the baseline method using a solution of the TSP is described together with the proposed GTSP-based method. The employed GSOA-based approach and its adaptation to the addressed TSPNS are described in Section 5. The local post-processing optimization is presented in Section 6. Results on the empirical evaluation are reported

in Section 7, and concluding remarks are outlined in Section 8.

2. Related Work

The proposed Traveling Salesman Problem with Neighborhoods on a Sphere (TSPNS) is a variant of the TSP with Neighborhoods (TSPN) extending the non-Euclidean TSP on a Sphere (TSPS). Therefore, a brief overview of existing approaches to the Euclidean TSPN and the non-Euclidean TSPS is provided in this section.

The TSPN is a generalization of the TSP known to be APX-hard unless P=NP (Gudmundsson et al., 2000). It includes the sequencing part of the TSP to determine the optimal sequence of visits to the given targets. Besides, the TSPN also includes determining the most suitable visiting location of each target neighborhood such that the cost of the tour visiting the neighborhoods is minimal. Approximation algorithms and heuristic approaches have been proposed for variants of the TSPN with restricted types of the neighborhoods, such as neighborhoods shaped to parallel unit segments, translations of polygonal regions, and circles (Arkin and Hassin, 1994). Even though the TSPN with complex-shaped neighborhoods have been studied (Dumitrescu and Mitchell, 2003; De Berg et al., 2005; Elbassioni et al., 2009), there are still many neighborhood types not addressed by the existing approximation methods. For example, non-convex neighborhoods are addressed in (Gentilini et al., 2013), where the authors propose a non-convex mixed-integer non-linear program to find a solution using optimization solvers.

A widely studied variant of the TSPN is with the disk-shaped neighborhoods referred to as the Close Enough TSP (CETSP) (Gulczynski et al.,

2006) with several heuristics and benchmarks reported in (Mennell, 2009). Therein reported results indicate that fast heuristic solvers for instances with overlapping neighborhoods can be based on the sampling of Steiner zones; however, transformation approaches are reported to provide better solution quality. The transformation methods sample the continuous disk-shaped neighborhoods into sets of discrete locations; hence, the problem becomes an instance of the Generalized TSP (GTSP) that is to find the cost-efficient tour visiting just a single location of each set. Although the GTSP-based approaches can find high-quality solutions for dense sampling, they quickly become computationally too demanding with the increasing number of samples.

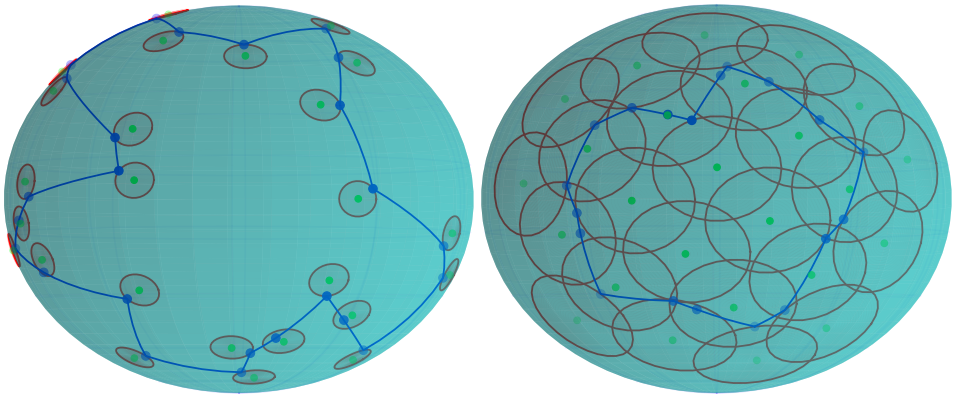
A direct unsupervised learning-based heuristic called the Growing Self-Organizing Array (GSOA) to address routing problems with neighborhoods is proposed in (Faigl, 2018). The GSOA is reported to provide better results than the heuristics of (Mennell, 2009) including the transformation GTSP-based method. At the same time, the computational requirements of the GSOA are only a fraction of those needed by the existing GTSP solvers.

Regarding non-Euclidean instances of the TSP, they can be found in multi-goal path planning in finding paths among obstacles (Faigl et al., 2011), in cuboid domains (Aybars, 2008), and spherical domains (Uğur et al., 2009). The TSP on a Sphere (TSPS) is of particular interest because it shares the spherical domain with the TSPNS. The discrete cuckoo search (Ouyang et al., 2013) has improved probably the first genetic-based solution of the TSPS proposed in (Uğur et al., 2009). The ant colony optimization has been used for solving the non-Euclidean TSPS in (Eldem and Ülker, 2017) that outperforms both the former genetic-based approach and the discrete cuckoo search. However, its computational requirements are relatively high

compared to the results reported for unsupervised learning in (Faigl, 2018; Faigl et al., 2011). Therefore, we consider the unsupervised learning of the GSOA (Faigl, 2018) as a suitable technique to address the TSPNS directly.

3. Problem Statement

The herein addressed TSP with Neighborhood on a Sphere (TSPNS) is a variant of the TSP with neighborhoods that stands to determine the closed shortest path visiting each of the given set of possibly overlapping neighborhoods. Since the neighborhoods are regions on the sphere's surface in the TSPNS, the problem is to find the closed shortest path connecting all of the given regions on the sphere's surface as depicted in Fig. 2. The TSPNS can be formally defined as follows.



(a) Non-overlapping instance (b) Overlapping instance

Figure 2: Instance of the TSPNS, each with a solution depicted in blue denoting a path connecting all regions on the sphere's surface.

Let $\mathcal{Q}$ be a sphere given by its center $Q \in \mathbb{R}^3$ and radius $\delta > 0$, and let $S = \{S_1, \ldots, S_n\}$ be a set of n regions on the surface of $\mathcal{Q}$. For the motivational RTI scenario, each spherical cap region S_i is defined based on

the required lightning direction $\boldsymbol{c}_i$ and the solid angle Ω_i representing the allowed tolerance to the direction $\boldsymbol{c}_i$. Thus, for the normalized vector $\boldsymbol{x}$ with the origin at Q, the region S_i is a set of all points P at a distance δ from Q that are within the angle Ω_i:

$$S_i = \{P \,|\, P = Q + \delta\boldsymbol{x} \text{ and } \|\boldsymbol{x}\| = 1 \text{ and } \angle(\boldsymbol{x}, \boldsymbol{c}_i) \leq \Omega_i\}. \tag{1}$$

The solution of the TSPNS can be described by the sequence of visits to the regions $\Sigma = (\sigma_1, \ldots, \sigma_n)$, $1 \leq \sigma_i \leq n$, and $\sigma_i \neq \sigma_j$ for $i \neq j$, together with the set of corresponding waypoints $P = \{P_1, \ldots, P_n\}$, $P_i \in \mathbb{R}^3$, each with the distance $\|P_i - Q\| = \delta$ from the center Q; thus, $P_i \in S_i$. The final path visiting the regions S can be constructed by connecting each pair of consecutive waypoints of the sequence in the shortest possible way on the surface of the sphere $\mathcal{Q}$[1]. Since the TSPNS stands to determine the sequence Σ and the waypoints P, the length of the multi-goal path is denoted $\mathcal{L}(\Sigma, P)$.

The waypoints P can be represented by a normalized direction vectors $\boldsymbol{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ from the center Q, because it significantly simplifies the notation. The waypoints are therefore denoted as the waypoint vectors

$$P_i = Q + \delta\,\boldsymbol{x}_i, \quad \|\boldsymbol{x}_i\| = 1. \tag{2}$$

Having the introduced notation, the TSPNS can be formulated as the optimization Problem 3.1.

[1] We restrict the connections of the regions on the sphere's surface even for an aerial vehicle in the RTI scenario because of the required lightening from the defined distance δ for the whole flight of the vehicle.

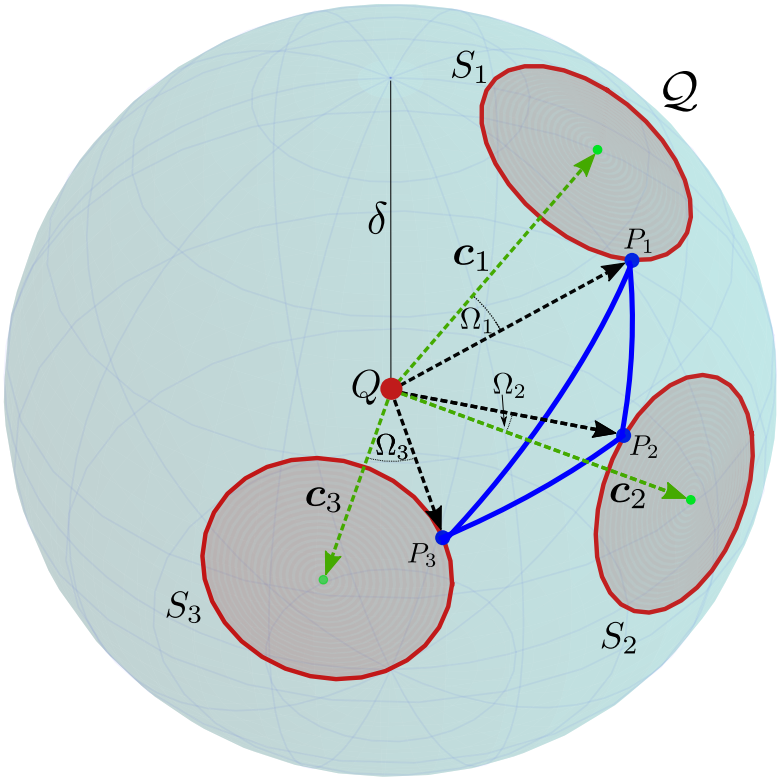


Figure 3: An instance of the TSPNS with the locations $S = \{S_1, S_2, S_3\}$ on the sphere $\mathcal{Q}$ with the center Q and radius δ. The regions are illustrated with the red boundary of the spherical cap. The associated direction $\boldsymbol{c}_i$ for S_i is in green. The solution is in blue, where P_i represents the waypoint for the region S_i.

Problem 3.1 (TSP with Neighborhoods on a Sphere - TSPNS).

$$\text{minimize}_{\Sigma,P}\, \mathcal{L}(\Sigma, P) \;=\; \sum_{i=1}^{n-1} \mathcal{L}(P_i, P_{i+1}) + \mathcal{L}(P_n, P_1)$$

s.t.

$$\Sigma = (\sigma_1, \ldots, \sigma_n),\ 1 \le \sigma_i \le n;\ \sigma_i \neq \sigma_j \text{ for } i \neq j,$$

$$P = \{P_1, \ldots, P_n\},\ \boldsymbol{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\},$$

$$P_i = Q + \delta\, \boldsymbol{x}_i,\ \|\boldsymbol{x}_i\| = 1,\ \angle(\boldsymbol{x}_i, \boldsymbol{c}_{\sigma_i}) \le \Omega_{\sigma_i},\ i \in \{1, \ldots, n\},$$

where $\mathcal{L}(P_i, P_j)$ is the length of the shortest path on the surface of the sphere $\mathcal{Q}$ connecting the waypoint P_i with P_j. The computation of the shortest path is detailed in the rest of this section. An example of the TSPNS instance with the defined notation is depicted in Fig. 3.

3.1. Shortest Distance on a Sphere

The shortest path between two waypoints P_i and P_j on the sphere's surface is often referred to as the *great-circle distance* or *orthodromic distance* in the literature. It can be computed based on the angle $\angle P_iQP_j$ and the radius δ as

$$\mathcal{L}(P_i, P_j) = \delta \angle P_iQP_j. \tag{3}$$

If the solution is represented by the waypoint vectors $\boldsymbol{X}$, (3) can be expressed as

$$\mathcal{L}(P_i, P_j) = \delta \angle(\boldsymbol{x}_i, \boldsymbol{x}_j) = \delta \arccos\left(\boldsymbol{x}_i \cdot \boldsymbol{x}_j\right), \tag{4}$$

where $\boldsymbol{x}_i \cdot \boldsymbol{x}_j$ is the scalar product of two vectors.

4. Discretized TSPNS – Transformation-based Solution of the TSPNS

The studied TSPNS with continuous neighborhoods can be addressed by the transformation approach and creating a discrete instance of the GTSP using sampling of the regions. Since the regions are spherical caps defined by $\boldsymbol{c}_i$, we can sample S_i and transform the problem into the non-Euclidean TSP with the distance matrix D,

$$D = \begin{bmatrix} 0 & d_{12} & \cdots & d_{1n} \\ d_{21} & 0 & \cdots & d_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ d_{n1} & d_{n2} & \cdots & 0 \end{bmatrix}, \tag{5}$$

where d_{ij} is the shortest distance between the regions' points P_i and P_j determined by (4) using $\boldsymbol{c}_i$ for the regions $i \in \{1, \ldots, n\}$. Since the dot product in (4) is commutative, $d_{ij} = d_{ji}$ applies, and the matrix D is symmetric.

Baseline Solution. For a single sample per each region S_i, e.g., using the center of the region computed from $\boldsymbol{c}_i$, the discretized TSPNS becomes an instance of the TSP that can be solved by the LKH solver (Helsgaun, 2018). A single sample per region does not allow exploiting the tolerance of the lighting direction defined by the solid angle Ω_i in the motivational RTI scenario. However, it represents a baseline solution (of the TSPS) to show the benefits of the introduced TSPNS in the reduction of the solution cost provided by the proposed GTSP-based and GSOA-based approaches to the TSPNS.

4.1. Proposed GTSP-based Solution to the TSPNS

The identical procedure of determining the distance matrix (5) can be utilized for the transformation of the TSPNS to an instance of the GTSP. The only difference is that each spherical cap region S_i is sampled into m discrete locations expressed as the set $B_i = \{B_i^1, \ldots, B_i^m\}$. The problem is then to determine the shortest path visiting each set B_i exactly once, i.e., visiting a single sample from each set B_i. Since the solution of the TSPNS is limited to the sphere's surface, every path visiting a certain region has to pass its circular border. Hence, we can sample each region S_i by even distribution of m discrete points along its circular border. The distance matrix corresponding to an instance of the GTSP can be computed similarly to the former case, but n^2m^2 distances have to be determined in total. Then, the GTSP instance with the pre-computed distance matrix can be solved by an available purely combinatorial GTSP solver. In this work, we utilize the GLKH solver (Helsgaun, 2013).

5. Proposed GSOA-based Solver to the TSPNS

The proposed direct solution to the TSPNS is based on the Growing Self-Organizing Array (GSOA) (Faigl, 2018), which is a general heuristic approach to solve routing problems with neighborhoods. Since the addressed instances of the TSPNS are non-Euclidean, it is necessary to modify the learning procedure of the GSOA appropriately. The proposed modifications can be considered relatively straightforward within the context of the GSOA. Therefore, an overview of the GSOA is presented in Section 5.1 to make the paper self-contained. The main proposed modifications are detailed in Section 5.2. The presented overview of the GSOA follows (Faigl, 2018); however, we use the notation of the TSPNS for consistency.

5.1. Growing Self-Organizing Array (GSOA)

In the GSOA, the solution of the routing problem is represented as a growing array of M nodes $\mathcal{N} = \{\nu_1, \ldots, \nu_M\}$ that is iteratively adapted to the regions (neighborhoods) S in a finite number of the learning epochs. Each node ν_j corresponds to a point in the solution domain. In our TSPNS case, the domain is the sphere's surface, and ν_j corresponds to vector $\boldsymbol{\nu}_j$ denoting a point on the surface of the sphere $\mathcal{Q}$ based on (2). Furthermore, each node ν_j is associated with a single region S_k and a waypoint vector $\boldsymbol{s}_p$ at which S_k is visited by the current solution encoded in $\mathcal{N}$.

The adaptation is unsupervised learning, where the best matching node (called the *winner node*) is iteratively determined for the randomly selected region S_k. The winner node and its neighboring nodes are then moved towards S_k. During the determination of the *winner node*, the waypoint vector $\boldsymbol{s}_p$ is determined as the closest point of the particular region S_k to the location of the newly determined *winner node* denoted ν^*. The relation

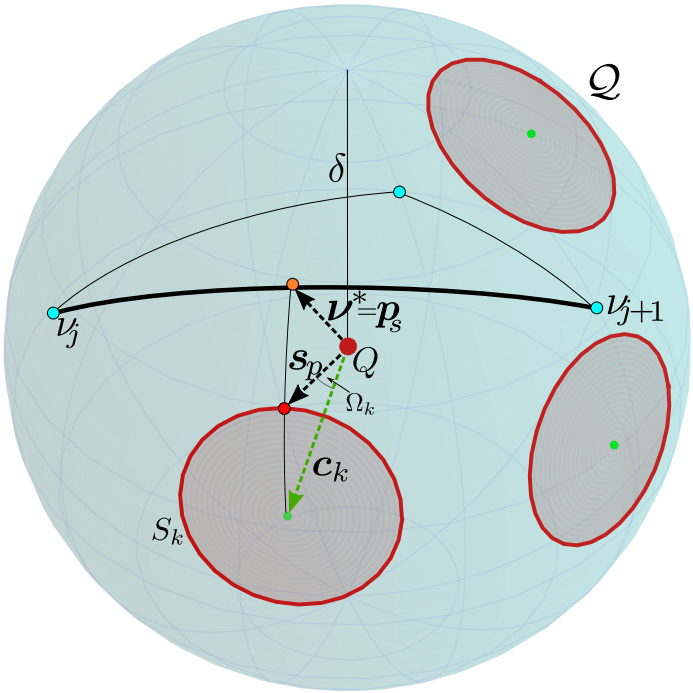


Figure 4: A visualization of the GSOA nodes $\mathcal{N}$ at some learning epoch in which a *winner node* ν^* for the region S_k is determined at the location $\boldsymbol{\nu}^* = \boldsymbol{p}_s$ together with the waypoint $\boldsymbol{s}_p$ using the shortest distance $\mathcal{L}$ from the arc defined by (ν_j, ν_{j+1}) to the region S_k represented by its center $\boldsymbol{c}_k$ and solid angle Ω_k.

of the nodes, *winner node* ν^*, and its waypoint to the region S_k is visualized in Fig. 4. The learning epoch is an adaptation of $\mathcal{N}$ to all the regions S.

The adaptation is repeated until the termination condition is met that is defined by the maximal number of learning epochs or when the array of nodes $\mathcal{N}$ converges to a stable state (Faigl, 2018). Since a new node is determined for each region of S together with the waypoint $\boldsymbol{s}_p$ in every epoch, a feasible solution can be retrieved at the end of the epoch by traversing $\mathcal{N}$ through $\boldsymbol{s}_p$. The GSOA algorithm is summarized in Algorithm 1, and it is detailed in the

rest of this subsection.

The GSOA starts with the initialization of the array $\mathcal{N}$ by randomly selecting three regions from S and for each region creating a node ν_j with the lighting direction $\boldsymbol{c}_k$ as the node position on the sphere $\boldsymbol{\nu}_j = \boldsymbol{c}_k$ that is also used for the associated waypoint. The actual maximal number of iterations i_{max} is set to the smaller value of either the pre-specified i_{max} or $1/\alpha$ to ensure the utilized learning gain G is always above zero, where α is the gain decreasing rate. Besides the pre-specified value of i_{max}, the initial value of G, and the value of the learning rate α, the GSOA can be parameterized by the learning rate μ. The parameters influence the power of adaptation and thus the solution convergence; however, as described in (Faigl, 2018) based on the empirical evaluation, the used values $i_{max} = 150$, $\mu = 0.6$, $G = 10$, and $\alpha = 0.0005$ provide a suitable tradeoff between the computational requirements and solution quality. Hence, the GSOA is considered to be parameterless (Faigl, 2018), and for solving the TSPNS, we follow the recommended values since there was a negligible influence on the performance of the solver.

After the initialization, the adaptation of $\mathcal{N}$ towards S is performed in the finite number of learning epochs. During each epoch, all the regions $S_k \in S$ are processed in a random order to escape local optima. For each S_k, a new *winner node* ν^* is determined (Algorithm 1, Line 6) and inserted into $\mathcal{N}$ as follows. The array $\mathcal{N}$ is examined as a sequence of paths connecting ν_j with ν_{j+1}, where $\nu_j \triangleq \nu_{j-M}$ for $j > M$, to reflect the closed multi-point path connecting all the regions. Therefore, the *winner node* ν^* is determined as the closest point of the path represented by $\mathcal{N}$ to the region S_k on the sphere's surface. The location of ν^* is determined as $\boldsymbol{p}_s$ representing the point of the arc defined by (ν_j, ν_{j+1}) that is an endpoint of the arc with

Algorithm 1: GSOA for the TSPNS

Input: $S = \{S_1, \ldots, S_n\}$ – a set of the target regions.

Parameters : the maximal no. of iterations $i_{max} = 150$; learning rate $\mu = 0.6$; learning gain $G = 10$; gain decreasing rate $\alpha = 0.0005$.

Output: $(\Sigma,\ P)$ – Σ sequence of visits to S with the corresponding waypoints P.

$\mathcal{N} \leftarrow \{\nu_1, \nu_2, \nu_3\}$ `// Random initialization using three regions of` S

$i_{max} \leftarrow \min(i_{max}, 1/\alpha)$ `// Ensure` G `will be always above 0`

$i \leftarrow 1$ `// Init. epoch counter, the current` $\mathcal{N}$ `is considered the previous epoch`

while $i \leq i_{max}$ `and` solution (Σ, P) is evolving **do**

  foreach S_k in a random permutation of S **do**

    $(\boldsymbol{\nu}^*, \nu_j, \nu_{j+1}, \boldsymbol{p}_s, \boldsymbol{s}_p) \leftarrow \text{selectWinner}(\mathcal{N}, S_k)$

    $\mathcal{N} \leftarrow \text{insertNode}(\mathcal{N}, \boldsymbol{\nu}^*, \nu_j, \nu_{j+1})$

    $\mathcal{N} \leftarrow \text{adapt}(\mathcal{N}, \boldsymbol{\nu}^*, \boldsymbol{s}_p)$

  $i \leftarrow i + 1$ `// Increase the learning epoch counter`

  $\mathcal{N} \leftarrow \text{removeNodes}(\mathcal{N}, i-1)$ `// Remove nodes from the previous epoch`

  $G \leftarrow (1 - i\,\alpha)\,G$ `// Update the learning gain`

  $(\Sigma', P') \leftarrow \text{extractSolution}(\mathcal{N})$

  if $\mathcal{L}(\Sigma', P') < \mathcal{L}(\Sigma, P)$ **then**

    $(\Sigma, P) \leftarrow (\Sigma', P')$ `// Update the best solution found so far`

$(\Sigma, P) \leftarrow \text{twoOpt}(\Sigma, P)$ `// Improve the solution by Two-Opt (Croes, 1958)`

$(\Sigma, P) \leftarrow \text{ppOptimization}(\Sigma, P)$ `// Post-processing optimization (Section 6)`

return (Σ, P)

the shortest distance $\mathcal{L}$ to S_k. Hence, $\boldsymbol{p}_s$ is determined together with the waypoint of the visit to S_k expressed as $\boldsymbol{s}_p$ that lies at the opposite endpoint of this arc, see Fig. 4. Both vectors $\boldsymbol{p}_s$ and $\boldsymbol{s}_p$ are determined using the center $\boldsymbol{c}_k$ of the region S_k. If the arc defined by (ν_j, ν_{j+1}) passes the region, both vectors $\boldsymbol{p}_s$ and $\boldsymbol{s}_p$ would be identical.

Once all the arcs defined by the consecutive nodes (ν_j, ν_{j+1}) of the array $\mathcal{N}$ are processed, the *winner node* ν^* corresponds to the point of $\mathcal{N}$ with the shortest distance $\mathcal{L}$ to the currently examined $S_k \in S$. The *winner node* ν^* is then inserted into $\mathcal{N}$ between the corresponding ν_j and ν_{j+1}, and its location $\boldsymbol{\nu}^*$ is set to $\boldsymbol{\nu}^* = \boldsymbol{p}_s$. The region S_k and waypoint $\boldsymbol{s}_p$ are associated with ν^* to enable solution retrieval. After that, ν^* is adapted to $\boldsymbol{s}_p$ together with its neighboring nodes. The adaptation of the node ν (the *winner node* ν^* or one of its neighboring nodes) can be considered as an adjustment of its location $\boldsymbol{\nu}$ as

$$\boldsymbol{\nu} \leftarrow \boldsymbol{\nu} + \mu f(G, d)(\boldsymbol{s}_p - \boldsymbol{\nu}), \tag{6}$$

where μ is the learning rate, G is the learning gain, d is the number of nodes between the neighboring node ν to the current winner node ν^*, and $f(G, d)$ is the neighboring function that adjust the power of adaptation according to

$$f(G, d) = \begin{cases} e^{-\frac{d^2}{G^2}} & \text{if } d < 0.2M \\ 0 & \text{otherwise} \end{cases}. \tag{7}$$

For the solution of the TSPNS, the vector $\boldsymbol{\nu}$ is normalized so that $\|\boldsymbol{\nu}\| = 1$ holds after the adaptation.

Because n new nodes are added to $\mathcal{N}$ in each learning epoch, the nodes from the previous epoch are removed from $\mathcal{N}$. Thus, after each learning epoch, the number of nodes M equals to n, and $\mathcal{N}$ encodes a feasible solution

by the array of nodes defining the sequence Σ and waypoints $\boldsymbol{s}_p$ associated to the nodes that represent the requested waypoints P (2).

The learning is repeated up to the selected i_{max} epochs or terminated whenever the solution does not change, such as the node locations $\boldsymbol{\nu}$ fit their waypoints $\boldsymbol{s}_p$. Since learning is randomized, the best solution among the epochs is maintained. In addition, the Two-Opt heuristic (Croes, 1958) is used to improve the final sequence Σ and the post-processing optimization procedure can be further applied to adjust the waypoints P. The proposed procedure for the TSPNS is detailed in Section 6.

The deployment of the original GSOA (Faigl, 2018) to the addressed TSPNS is relatively straightforward and the only part that needs to be adjusted is the *winner node* selection by `selectWinner()` (Algorithm 1, Line 6) that has to reflect paths on the surface of the sphere. Besides, the paths on the sphere's surface are also utilized during the node adaptation in `adapt()` (Algorithm 1, Line 8). The proposed winner node determination is detailed in the remaining part of this section.

5.2. Winner Node Determination in the GSOA-based Solution of the TSPNS

The GSOA has been introduced in (Faigl, 2018) for 2D instances and Euclidean distance for the *winner node* determination. However, poor solutions would be found if the problem domain is not Euclidean as for the regions on the sphere's surface. Therefore, we propose to determine the *winner node* ν^* and its corresponding waypoint on the sphere as the closest point of the region S_k to the array $\mathcal{N}$ using the following procedure.

Assuming the array $\mathcal{N}$ represents a sequence of paths connecting two consecutive nodes (ν_j, ν_{j+1}), $1 \leq j \leq M$ and $\nu_{M+1} = \nu_1$, the waypoint vector $\boldsymbol{p}_s$ for the region S_k is determined as follows. First, the vector $\boldsymbol{u}_k$ perpen-

dicular to the plane formed by the path segment $(\boldsymbol{\nu}_j, \boldsymbol{\nu}_{j+1})$ is determined as the cross product of two vectors

$$\boldsymbol{u}_k = \boldsymbol{\nu}_j \times \boldsymbol{\nu}_{j+1}, \tag{8}$$

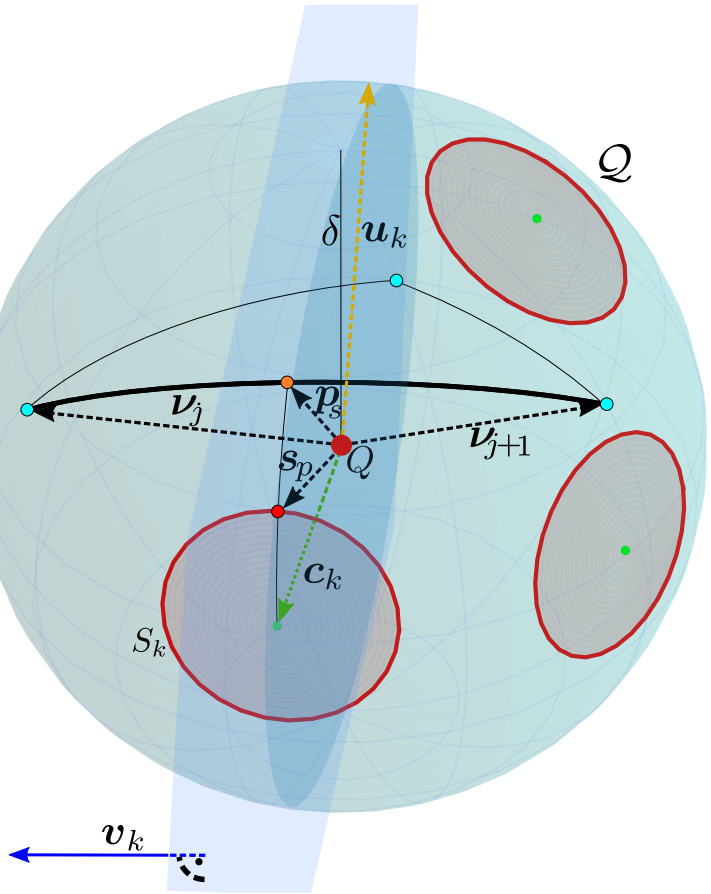


Figure 5: Visualization of the proposed *winner node* selection for solving the TSPNS using the GSOA. The location of the *winner node* for the arc (ν_j, ν_{j+1}) is defined by $\boldsymbol{p}_s$ that corresponds to the closest point of the arc to S_k that is determined as the cross product of the vectors $\boldsymbol{v}_k$ and $\boldsymbol{u}_k$. The waypoint associated with the *winner node* denoting the point of the visit to the region S_k is defined by the vector $\boldsymbol{s}_p$ derived from the rotation of $\boldsymbol{p}_s$ to the close neighborhood of $\boldsymbol{c}_k$.

where $\boldsymbol{\nu}_j$ and $\boldsymbol{\nu}_{j+1}$ are the corresponding vectors defining current locations of the nodes ν_j and ν_{j+1}, respectively. Then, a perpendicular vector $\boldsymbol{v}_k$ to the plane formed by the region's center $\boldsymbol{c}_k$ and $\boldsymbol{u}_k$ is determined as the cross product

$$\boldsymbol{v}_k = \boldsymbol{c}_k \times \boldsymbol{u}_k. \tag{9}$$

The vector $\boldsymbol{p}_s$ defining the point of the path segment is determined by the rotation of $\boldsymbol{v}_k$ to the plane formed by the path segment

$$\boldsymbol{p}_s = \boldsymbol{v}_k \times \boldsymbol{u}_k. \tag{10}$$

Having $\boldsymbol{p}_s$ of the arc (ν_j, ν_{j+1}) on the sphere, $\boldsymbol{s}_p$ defining the waypoint of visit to S_k is determined by rotating $\boldsymbol{p}_s$ so that $\angle(\boldsymbol{c}_k, \boldsymbol{s}_p) \leq \Omega_k$ holds. Hence, $\boldsymbol{s}_p$ is determined for S_k as

$$\boldsymbol{s}_p \quad = \quad \frac{\boldsymbol{s}'_p}{\left\|\boldsymbol{s}'_p\right\|} \tag{11}$$

where

$$\boldsymbol{s}'_p \quad = \quad \begin{cases} \boldsymbol{c}_k + \tan\Omega_k \left(\boldsymbol{p}_s \times \boldsymbol{c}_k\right) \times \boldsymbol{c}_k, & \text{if } \angle(\boldsymbol{c}_k, \boldsymbol{p}_s) > \Omega_k, \\ \boldsymbol{p}_s & \text{otherwise.} \end{cases} \tag{12}$$

A visualization of the proposed determination of the *winner node* is depicted in Fig. 5.

The adaptation is a movement of the node's corresponding location towards the waypoint locations using the shortest path on a surface.

6. Post-processing Optimization for Waypoints Adjustment in a Solution of the TSPNS

A solution of the TSPNS consists of the sequencing part of the underlying TSP and the determination of the optimal waypoints to visit the regions.

Having a sequence of visits to the regions, e.g., found by the herein presented approaches based on the TSP using regions' centers (Section 4), the GTSP (Section 4.1), and the GSOA (Section 5), the waypoints' locations can be adjusted by a local optimization such as a hill-climbing technique. Since the optimization is called after a solution of the TSPNS is determined, we call it the post-processing optimization denoted `ppOptimization()` in Algorithm 1. A solution of the TSPNS is represented by the sequence Σ and waypoint locations P, and the goal of the optimization is adjusting P to shorten the final path length. The process works as follows.

The idea is to optimize all the waypoint locations P represented by the vectors $\boldsymbol{X}$. However, only a single vector $\boldsymbol{x}_{\sigma_i} \in \boldsymbol{X}$ is optimized at a single moment to simplify the computation. The individual vectors $\boldsymbol{X}$ are iteratively optimized locally one-by-one to converge towards local optima for each adjusted vector. For a sequence of visits Σ, each vector $\boldsymbol{x}_{\sigma_i}$ is optimized taking into account the previous $\boldsymbol{x}_{\sigma_{i-1}}$ and the next $\boldsymbol{x}_{\sigma_{i+1}}$ vectors. Since $\boldsymbol{x}_{\sigma_i}$ corresponds to the visit of S_{σ_i}, the perpendicular vector $\boldsymbol{u}_{\sigma_i}$ to the plane containing an arc connecting $\boldsymbol{x}_{\sigma_{i-1}}$ and $\boldsymbol{x}_{\sigma_{i+1}}$ on the surface of $\mathcal{Q}$ is determined as

$$\boldsymbol{u}_{\sigma_i} = \boldsymbol{x}_{\sigma_{i-1}} \times \boldsymbol{x}_{\sigma_{i+1}}. \tag{13}$$

Then, the vector $\boldsymbol{v}_{\sigma_i}$ perpendicular to $\boldsymbol{u}_{\sigma_i}$ and $\boldsymbol{c}_{\sigma_i}$ is determined as

$$\boldsymbol{v}_{\sigma_i} = \boldsymbol{c}_{\sigma_i} \times \boldsymbol{u}_{\sigma_i}. \tag{14}$$

Using $\boldsymbol{v}_{\sigma_i}$, a vector $\boldsymbol{x}_{\sigma_i}$ is determined within the region S_{σ_i} using vectors $\boldsymbol{d}_{\sigma_i,1}$ and $\boldsymbol{d}_{\sigma_i,2}$ defined as

$$\begin{aligned} \boldsymbol{d}_{\sigma_i,1} &= \tan \Omega_{\sigma_i} \left((\boldsymbol{v}_{\sigma_i} \times \boldsymbol{u}_{\sigma_i}) \times \boldsymbol{c}_{\sigma_i} \right) \times \boldsymbol{c}_{\sigma_i} \\ \boldsymbol{d}_{\sigma_i,2} &= \tan \Omega_{\sigma_i} \left(\boldsymbol{d}_{\sigma_i,1} \times \boldsymbol{c}_{\sigma_i} \right) \end{aligned}. \tag{15}$$

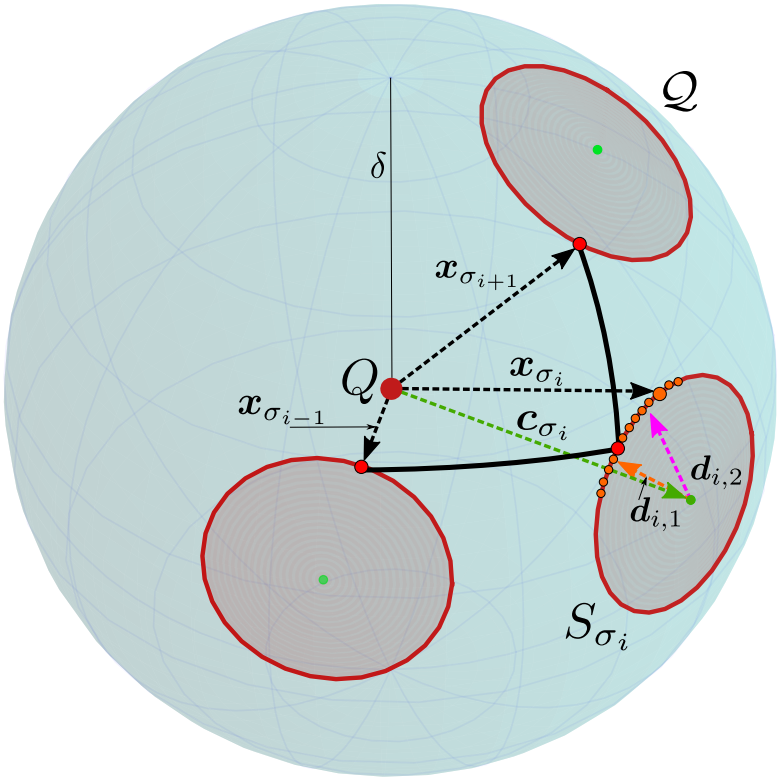


Figure 6: Visualization of the local post-processing optimization of the vector $\boldsymbol{x}_{\sigma_i}$ that represents the waypoint location $P_{\sigma_i} \in P$ of visit to the region S_{σ_i} denoted as a large disk with a red outline. During the optimization, the previous waypoint $\boldsymbol{x}_{\sigma_{i-1}}$ and the next waypoint $\boldsymbol{x}_{\sigma_{i+1}}$ of $\boldsymbol{x}_{\sigma_i}$ in the sequence Σ are taken into account. New locations where S_{σ_i} is visited are sampled at the boundary of the region (small orange disks) around the actual waypoint using a decreasing sampling step until the path length is improving.

Then, the initial value of $\boldsymbol{x}_{\sigma_i}$ is determined as

$$\boldsymbol{x}_{\sigma_i} = \frac{\boldsymbol{x}'_{\sigma_i}}{\left\|\boldsymbol{x}'_{\sigma_i}\right\|}, \quad \boldsymbol{x}'_{\sigma_i} = \boldsymbol{c}_{\sigma_i} + \boldsymbol{d}_{\sigma_i,1}. \tag{16}$$

If $\boldsymbol{x}_{\sigma_i}$ can be further improved, it is iteratively rotated on the spherical

surface of the region S_{σ_i}

$$\begin{aligned} \boldsymbol{x}_{\sigma_i} &= \frac{\boldsymbol{x}'_{\sigma_i}}{\left\|\boldsymbol{x}'_{\sigma_i}\right\|} \\ \boldsymbol{x}'_{\sigma_i} &= \begin{cases} \boldsymbol{c}_{\sigma_i} + \boldsymbol{d}_{\sigma_i,1}\cos(\gamma) + \boldsymbol{d}_{\sigma_i,2}\sin(\gamma), & \text{if } \angle(\boldsymbol{c}_k, \boldsymbol{v}_{\sigma_i}) > \Omega_k \\ \boldsymbol{v}_{\sigma_i} & \text{otherwise} \end{cases}, \end{aligned} \tag{17}$$

where γ is updated by the sampling step after each iteration of the local optimization of $\boldsymbol{x}_{\sigma_i}$. Thus, each waypoint is iteratively optimized to shorten the path until the sampling step is less than a predefined threshold $\beta = 1 \times 10^{-10}$. The whole sequence is optimized k times. The post-processing optimization is summarized in Algorithm 2, and a visualization of the optimization and used vectors are depicted in Fig. 6.

Algorithm 2: Post-processing Optimization for the TSPNS

Input: S – a set of the target regions

Input: (Σ, P) – a solution of the TSPNS as a sequence $\Sigma = (\sigma_1, \ldots, \sigma_n)$ of visits to the regions S and waypoints P represented by the normalized vectors $\boldsymbol{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$.

Parameters : the sampling step threshold $\beta = 1 \times 10^{-10}$; the no. of optimization loops $k = 5$.

Output: (Σ, P) – the solution with the adjusted waypoints P.

```
repeat k times
    foreach x_{σ_i} ∈ X do
        x_{σ_i} ← optimizeWaypoint(x_{σ_i}, β, S_{σ_i}, x_{σ_{i-1}}, x_{σ_{i+1}})    // Use (17)
return Q, Σ
```

In the presented description, we assume the waypoint vectors correspond to locations on the boundary of the regions (a cap of the sphere's surface), which holds in most cases. If the waypoint vector corresponds to a location

inside the region, the local optimization is skipped for such a waypoint because it would lie on the arc connecting the neighboring waypoints, and local improvement would not be effective.

7. Empirical Evaluation

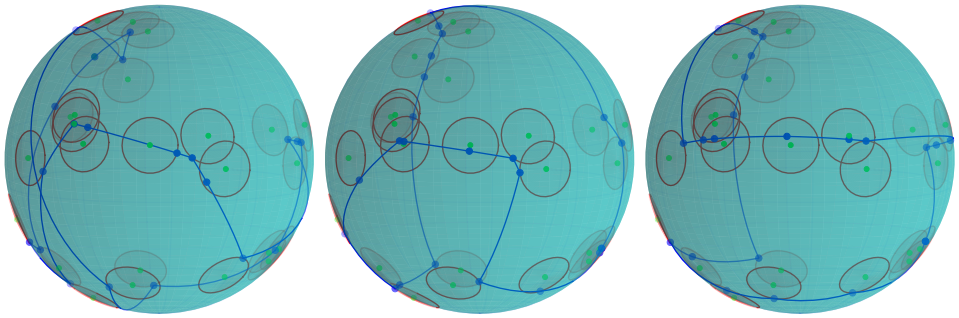
The proposed solvers to the TSPNS (both the transformation and GSOA based) have been empirically evaluated using 28 test instances and compared with the baseline solutions using the TSPS. Furthermore, the efficiency of the post-processing optimization procedure on the solution improvement has been evaluated for all three examined solvers. The structure of the reported results is following. The solvers have been evaluated on test instances described in Section 7.1 using statistical performance indicators. The evaluation setup, together with the description of the used performance indicators, is presented in Section 7.2. Results on finding a suitable sampling needed for the transformation-based method are reported in Section 7.3. Finally, the results on the empirical evaluation are presented in Section 7.4.

7.1. Test Instances of the TSPNS

The instances[2] are motivated by the practical RTI scenarios to provide desired lighting conditions by visiting defined locations on a sphere around an object of interest. The instances differ in the number of regions n, the volume of the sphere surface covered by the regions, and the distribution of regions over the sphere surface. The instance name encodes the instance character to support the readability of the evaluation results. The size of the

[2]All the benchmark instances are available at https://github.com/comrob/tspns.

instances is selected from the range $n \in \{10, 25, 50, 100, 500\}$ and n is added as a suffix to the instance name.



(a) TSPS-LKH: $\mathcal{L} = 13.1$, T = 0.03 s

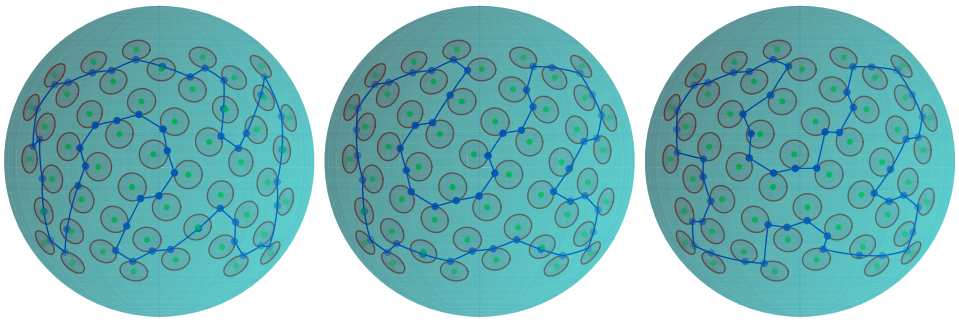
(b) GTSP-GLKH: $\mathcal{L} = 9.9$, T = 3.97 s

(c) GSOA: $\mathcal{L} = 9.9$, T = 0.02 s

Figure 7: The `sphere_rand_ol_25` instance with the selected solutions found by the TSPS-LKH, GTSP-GLKH, and GSOA solvers.

The distribution of the regions over the sphere's surface is random or tends to the even distribution. The even distribution of the locations corresponds to a typical RTI scanning scenario, and a Fibonacci lattice is employed to achieve an approximately even distribution of the regions (their centers) over the sphere. The distribution is encoded in the name of the instance as `rand` for random regions, and `reg` for even distribution.

The test instances are also created to distinguish regions located in the middle part of the sphere, encoded in the instance name as `band`. The instances with regions on top of the sphere either in the rectangular or spherical pattern are encoded as `rect_cap` and `cap`, respectively. Finally, the instances are divided into two main categories with fully non-overlapping regions encoded as `nol` in the instance name, and instances with at least two overlapping regions encoded as `ol`. Examples of the selected instances with their solutions are depicted in Fig. 7 and Fig. 8.



(a) TSPS-LKH: $\mathcal{L} = 8.7$, T = 0.15 s

(b) GTSP-GLKH: $\mathcal{L} = 8.0$, T = 22.27 s

(c) GSOA: $\mathcal{L} = 8.2$, T = 0.09 s

Figure 8: The `sphere_rect_cap_reg_nol_50` instance with the selected solution found by the TSPS-LKH, GTSP-GLKH, and GSOA solvers.

7.2. Evaluation Setup

Three solvers to the TSPNS are examined in the reported empirical evaluation. The baseline approach is the transformation method denoted TSPS-LKH because the heuristic solver LKH (Helsgaun, 2018) is used for the solution of the TSP instances created from the transformed TSPNS using the centers of the regions. Similarly, the transformation method based on more samples is denoted GTSP-GLKH because the transformed instances of the GTSP are solved by the heuristic GLKH (Helsgaun, 2013). The proposed GSOA-based method is denoted simply GSOA. Furthermore, the results without the procedure are denoted by TSPS-LKH', GTSP-GLKH', and GSOA', because the post-processing optimization procedure noticeably improves the solutions with a negligible increase of the computational requirements.

All the examined solvers TSPS-LKH, GTSP-GLKH, and the GSOA are randomized, and their performance is therefore measured among solving 50

trials for each instance. All the solvers are implemented in C++[3] and run within the same computational environment using a single core of the Intel Core i7-9700 with the base frequency 3.0 GHz. The computational requirements are thus reported as the real average computational time T in seconds that can be directly compared. The statistical comparison is based on the statistical test using a null hypothesis H_0 similarly as in (Faigl et al., 2013).

Although the learning parameters of the GSOA can be eventually tuned, for all herein reported results, we utilize the parameter values suggested in (Faigl, 2018), where the GSOA is called parameterless. Based on evaluating the influence of increasing learning gain up to $G = 50$ and various adjustments of the gain decreasing rate α (both affect the maximal number of learning epochs), we can confirm their negligible influence on the solution of the TSPNS. Hence, the values reported in Algorithm 1 are utilized.

Only the transformation-based solver GTSP-GLKH requires selection of the number of samples per region to determine a suitable tradeoff between the solution quality and computational requirements. The effect of the number of samples on the solution is reported in Section 7.3, from which we chose $m = 10$.

Since all the examined solvers are randomized, the performance indicators are computed among the performed trials. The solution quality obtained by the particular methods is measured as %PDB that is the percentage deviation of the best solution value $\mathcal{L}_{\text{best}}$ found among all performed trials of the particular solver to the reference solution value $\mathcal{L}_{\text{ref}}$. It is determined as $\%\text{PDB} = (\mathcal{L}_{\text{best}} - \mathcal{L}_{\text{ref}})/\mathcal{L}_{\text{ref}} \cdot 100\%$, where $\mathcal{L}_{\text{ref}}$ is the best solu-

[3]The proposed transformation-based methods TSPS-LKH and GTSP-GLKH, and the direct GSOA-based method are available at https://github.com/comrob/tspns.

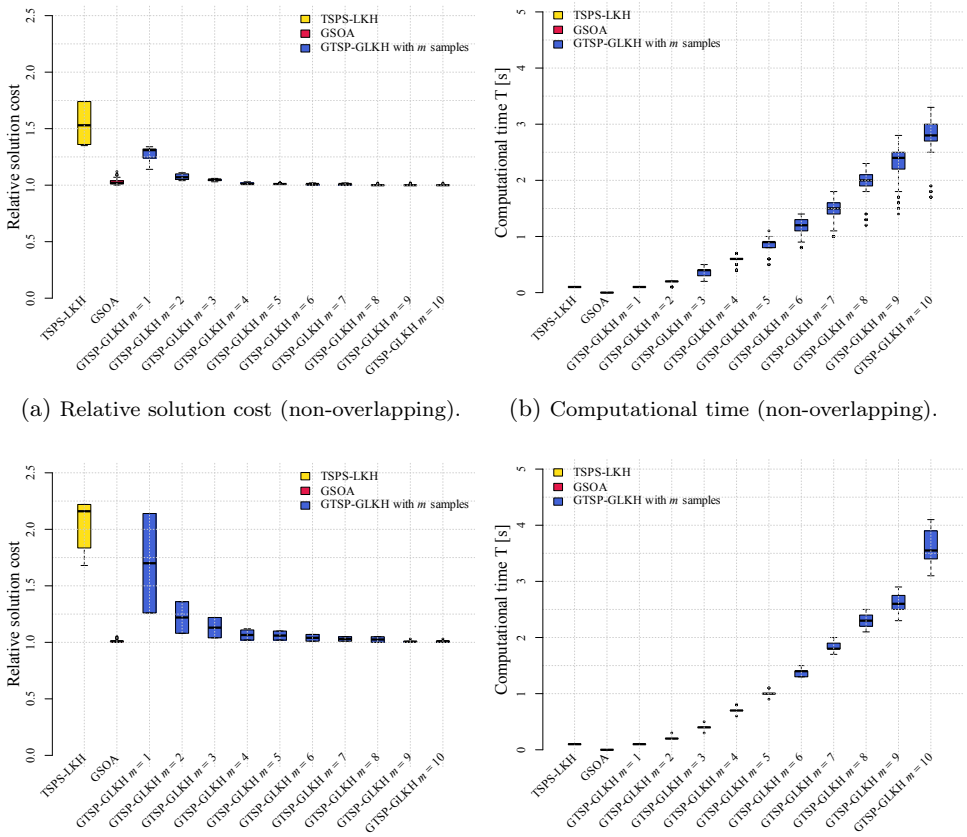
tion value of the particular instance determined among all solvers examined. The robustness of the solution is measured as the percentage deviation of the mean solution value $\mathcal{L}_{\text{avg}}$ among the trials to $\mathcal{L}_{\text{ref}}$ computed as $\%\text{PDM} = (\mathcal{L}_{\text{avg}} - \mathcal{L}_{\text{ref}})/\mathcal{L}_{\text{ref}} \cdot 100\%$.

The statistical significance of the performance indicators is evaluated using the *Wilcoxon test* (Arcuri and Briand, 2011) to report that one algorithm provides statistically better results than the other. The null hypothesis H_0 states that the methods provide statistically similar results in the quality of the found solutions and required computational time. Based on the p-value obtained by the test, we can accept or reject the null hypothesis. For almost all the results, the p-values obtained by the *Wilcoxon test* are less than the selected significance level 0.001; therefore, the characters '+', '−', and '=' are used to report that a particular algorithm a_1 performs better, worse, or equally as well as algorithm a_2.

7.3. Effect of Sampling Density on the Solution Quality of the GTSP-GLKH

For the transformation method GTSP-GLKH, it is desirable to select a suitable number of samples per each region m, where better solutions can be expected when increasing m at the cost of increased computational requirements. Therefore, the performance of the GTSP-GLKH solver has been studied up to $m \leq 10$ because more samples do not provide significantly better solutions, but the computational requirements are prohibitively high.

The effect of increasing the number of samples m per each target region to the solution quality and computational requirements has been studied, and results for the test instances of the TSPNS with $n = 25$ and $n = 100$ regions are depicted in Fig. 9 and Fig. 10, respectively. In this case, the solution quality is reported as the relative solution cost to $\mathcal{L}_{\text{ref}}$ visualized as



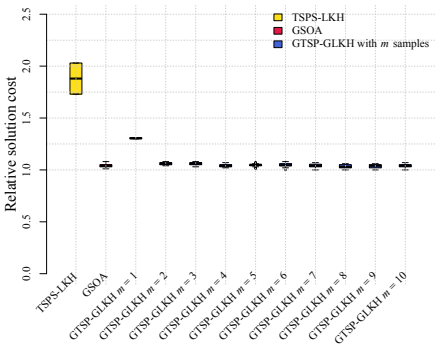
(a) Relative solution cost (non-overlapping). (b) Computational time (non-overlapping).

(c) Relative solution cost (overlapping). (d) Computational time (overlapping).

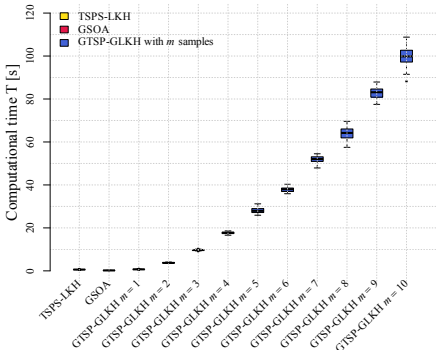
Figure 9: Effect of increasing number of samples m to the relative solution cost and computational requirements of the GTSP-GLKH for the instances with $n = 25$ regions, depicted as the five-number summary.

the five-number summary.

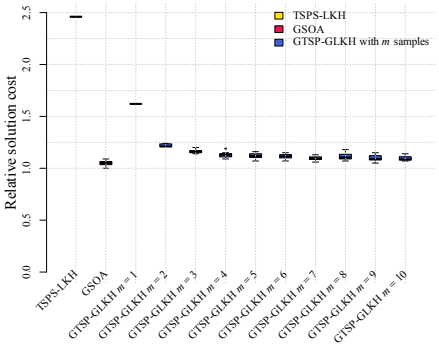
For the relatively small TSPNS instances with 25 regions, finding the best solutions requires a high number of samples that are more visible for instances with overlapping regions. On the other hand, solution improvement with increasing m is not directly visible for instances with 100 regions. However, at least two samples per region ($m = 2$) are needed to find com-



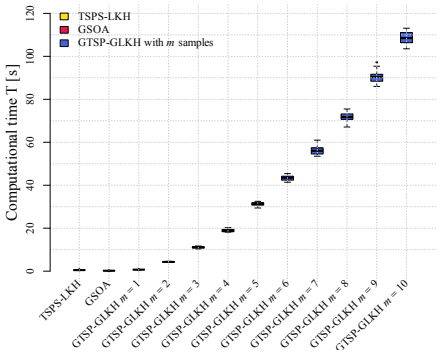
(a) Relative solution cost (non-overlapping).



(b) Computational time (non-overlapping).



(c) Relative solution cost (overlapping).



(d) Computational time (overlapping).

Figure 10: Effect of increasing number of samples m to the relative solution cost and computational requirements of the GTSP-GLKH for instances with $n = 100$ regions, depicted as the five-number summary.

petitive solutions to solutions found by the GSOA. It is also not surprising that instances with overlapping regions need more samples than for non-overlapping regions, and even for large instances, $m = 10$ does not yield competitive solutions; however the computational requirements increase significantly. The visible trend is an exponential growth of T with m; therefore, $m = 10$ is selected as a suitable tradeoff between the solution quality and

required computational time of the GTSP-GLKH for the examined test instances with an only limited number of large instances.

7.4. Results

Detailed evaluation results of the examined TSPNS solvers are presented in Table 1 and Table 2. The results support the motivation to formulate the UAV-based RTI tasks as the TSPNS because it allows exploiting the tolerance in lightning directions as both the GTSP-GLKH and GSOA solvers provide significantly better solutions than the TSPS-LKH solving the TSPS.

The post-processing optimization (described in Section 6) improves the solutions, especially for the transformation-based method TSPS-LKH and GTSP-GLKH, without a significant increase of the computational time. Although the computational requirements of the TSPS-LKH are very small, the solutions found are significantly more costly than solutions found by the GTSP-GLKH and GSOA solvers. Regarding the solution quality, most of the best-found solutions are provided by the GTSP-GLKH with the post-processing optimization. In a few cases, it stacks in a local optimum that is significantly worse than the solution of the GSOA. However, smaller values of the %PDM for the GTSP-GLKH than for the GSOA indicates that the GTSP-GLKH is more robust hence it provides the best solutons. Therefore, the statistical *Wilcoxon test* has been performed for the individual instances to identify which solution method is better in a statistically significant way.

The statistical comparison of the methods with and without the post-processing optimization is depicted in Table 3. The results further support the benefits of the post-processing optimization that helps both the GTSP-GLKH and GSOA methods. The GSOA provides better results for instances

Table 1: Performance indicators of the TSPNS solvers without post-processing optimization.

Instance	$\mathcal{L}_{\text{ref}}$	TSPS-LKH'			GTSP-GLKH'			GSOA'		
		%PDB	%PDM	T [s]	%PDB	%PDM	T [s]	%PDB	%PDM	T [s]
sphere_band_rand_nol_25	7.41	217.56	240.54	0.023	0.38	**0.38**	1.479	**0.25**	0.80	**0.012**
sphere_band_rand_nol_50	10.12	324.33	324.33	0.094	**0.78**	**0.86**	14.090	1.29	2.85	**0.046**
sphere_band_rand_ol_50	9.55	165.97	165.97	0.080	**0.74**	**0.75**	15.153	0.94	2.38	**0.046**
sphere_cap_rand_nol_10	6.12	18.89	40.54	0.005	**0.11**	**0.11**	0.212	0.15	0.34	**0.002**
sphere_cap_rand_nol_25	6.18	36.58	36.58	0.021	0.57	**0.57**	2.512	**0.53**	1.78	**0.011**
sphere_cap_rand_nol_50	10.74	75.16	75.16	0.092	**0.48**	**0.84**	15.443	1.82	6.11	**0.044**
sphere_cap_reg_nol_10	5.01	34.90	34.90	0.006	0.38	**0.38**	0.178	**0.21**	0.43	**0.002**
sphere_cap_reg_nol_25	6.74	35.80	35.80	0.018	**0.58**	**0.69**	2.474	0.73	3.42	**0.011**
sphere_cap_reg_nol_50	8.35	58.97	58.97	0.131	**1.07**	**2.64**	15.968	2.37	5.78	**0.044**
sphere_cap_reg_ol_25	4.09	124.08	124.08	0.017	1.54	1.58	3.506	**0.52**	**1.46**	**0.010**
sphere_rand_nol_100	22.79	73.88	73.88	0.495	**0.50**	**3.92**	88.627	1.86	5.14	**0.177**
sphere_rand_nol_10	8.50	19.64	43.43	0.007	**0.26**	**0.34**	0.231	0.32	1.73	**0.002**
sphere_rand_nol_25	10.62	74.68	74.68	0.015	0.70	**0.70**	2.784	**0.31**	4.62	**0.011**
sphere_rand_nol_500	50.26	101.98	109.78	20.712	2.79	4.11	1 277.769	**2.17**	**4.05**	**4.140**
sphere_rand_nol_50	15.21	122.90	122.90	0.180	**0.76**	**2.27**	20.422	1.20	5.67	**0.044**
sphere_rand_ol_100	15.59	149.05	149.05	0.414	7.76	11.20	112.117	**1.02**	**6.15**	**0.170**
sphere_rand_ol_10	7.56	38.98	67.48	0.005	0.27	**0.31**	0.262	**0.10**	0.51	**0.002**
sphere_rand_ol_25	9.86	67.45	86.88	0.019	0.63	**0.96**	3.295	**0.36**	3.12	**0.011**
sphere_rand_ol_50	12.89	126.09	126.09	0.075	**0.95**	**1.63**	19.842	1.33	6.28	**0.044**
sphere_rect_cap_rand_nol_25	8.56	53.24	53.24	0.021	**0.31**	**0.31**	2.603	0.71	2.62	**0.011**
sphere_rect_cap_rand_nol_50	7.60	54.51	54.51	0.107	**0.64**	**1.07**	18.279	1.71	3.48	**0.044**
sphere_rect_cap_reg_nol_10	4.91	32.59	32.59	0.006	0.24	**0.24**	0.232	**0.17**	0.76	**0.002**
sphere_rect_cap_reg_nol_25	6.46	40.25	40.25	0.027	**0.67**	**0.71**	2.639	0.79	2.23	**0.011**
sphere_rect_cap_reg_nol_50	8.00	60.46	60.46	0.095	**0.92**	**2.50**	17.978	1.64	4.95	**0.043**
sphere_reg_nol_100	25.88	105.18	105.18	0.540	2.56	4.91	107.016	**1.29**	**3.72**	**0.173**
sphere_reg_nol_10	10.11	33.11	57.32	0.005	**0.11**	**0.11**	0.250	0.20	2.07	**0.002**
sphere_reg_nol_25	14.92	62.31	62.31	0.019	**0.12**	**0.92**	2.915	1.50	4.86	**0.011**
sphere_reg_nol_50	20.23	84.27	84.27	0.093	**0.28**	**1.33**	17.088	1.26	3.58	**0.044**
Mean values	-	*85.46*	*90.76*	*0.833*	*0.97*	*1.66*	*63.049*	*0.96*	*3.25*	*0.185*

The smallest performance indicators per each instance are highlighted in bold.

with 500 regions, which indicates a better scalability of the GSOA than GTSP-GLKH. It can also be seen that the computational requirements of the GSOA are statistically increased by the post-processing optimization;

Table 2: Performance indicators of the TSPNS solvers with the post-processing optimization.

Instance	$\mathcal{L}_{\text{ref}}$	TSPS-LKH			GTSP-GLKH			GSOA		
		%PDB	%PDM	T [s]	%PDB	%PDM	T [s]	%PDB	%PDM	T [s]
sphere_band_rand_nol_25	7.41	193.52	215.53	0.023	**0.00**	**0.00**	1.479	**0.00**	0.24	**0.012**
sphere_band_rand_nol_50	10.12	291.03	291.03	0.095	**0.00**	**0.11**	14.091	**0.00**	1.44	**0.047**
sphere_band_rand_ol_50	9.55	129.35	129.35	0.080	**0.00**	**0.06**	15.153	0.17	1.27	**0.047**
sphere_cap_rand_nol_10	6.12	0.00	18.93	0.005	**0.00**	**0.00**	0.212	**0.00**	0.00	**0.002**
sphere_cap_rand_nol_25	6.18	9.14	9.14	0.022	**0.00**	**0.00**	2.512	**0.00**	0.79	**0.011**
sphere_cap_rand_nol_50	10.74	44.94	44.94	0.093	**0.00**	**0.34**	15.444	1.08	4.97	**0.045**
sphere_cap_reg_nol_10	5.01	13.14	13.14	0.007	**0.00**	**0.00**	0.179	**0.00**	**0.00**	**0.002**
sphere_cap_reg_nol_25	6.74	5.98	5.98	0.018	**0.00**	**0.15**	2.474	**0.00**	2.10	**0.011**
sphere_cap_reg_nol_50	8.35	12.26	12.26	0.132	**0.00**	**1.78**	15.968	1.24	4.17	**0.045**
sphere_cap_reg_ol_25	4.09	30.79	31.29	0.017	0.11	**0.19**	3.506	**0.00**	0.39	**0.010**
sphere_rand_nol_100	22.79	42.81	42.81	0.496	**0.00**	**3.38**	88.629	0.52	3.82	**0.179**
sphere_rand_nol_10	8.50	4.46	22.74	0.007	**0.00**	**0.02**	0.231	**0.00**	0.96	**0.002**
sphere_rand_nol_25	10.62	32.51	32.51	0.015	**0.00**	**0.00**	2.784	**0.00**	3.66	**0.011**
sphere_rand_nol_500	50.26	69.99	78.30	20.719	2.15	3.47	1 277.778	**0.00**	**2.07**	**4.148**
sphere_rand_nol_50	15.21	68.13	68.13	0.181	**0.00**	**1.50**	20.423	0.02	4.24	**0.045**
sphere_rand_ol_100	15.59	94.84	98.98	0.415	6.45	9.72	112.118	**0.00**	**5.36**	**0.171**
sphere_rand_ol_10	7.56	17.09	40.49	0.005	**0.00**	**0.05**	0.263	**0.00**	0.09	**0.002**
sphere_rand_ol_25	9.86	29.59	53.94	0.019	0.00	**0.37**	3.295	**0.00**	2.52	**0.011**
sphere_rand_ol_50	12.89	74.05	74.49	0.075	**0.00**	**0.62**	19.843	0.71	5.10	**0.044**
sphere_rect_cap_rand_nol_25	8.56	34.77	34.77	0.022	**0.00**	**0.00**	2.604	**0.00**	1.63	**0.011**
sphere_rect_cap_rand_nol_50	7.60	8.74	8.75	0.107	**0.00**	**0.44**	18.280	0.53	1.82	**0.045**
sphere_rect_cap_reg_nol_10	4.91	13.44	13.44	0.006	**0.00**	**0.00**	0.232	**0.00**	0.04	**0.002**
sphere_rect_cap_reg_nol_25	6.46	12.73	12.73	0.027	**0.00**	**0.00**	2.640	0.00	0.94	**0.011**
sphere_rect_cap_reg_nol_50	8.00	8.32	8.32	0.096	**0.00**	**1.73**	17.979	0.35	3.33	**0.044**
sphere_reg_nol_100	25.88	77.46	77.46	0.541	1.90	4.38	107.018	**0.00**	**2.29**	**0.175**
sphere_reg_nol_10	10.11	21.14	45.62	0.006	**0.00**	**0.00**	0.250	**0.00**	0.82	**0.002**
sphere_reg_nol_25	14.92	46.34	46.34	0.019	**0.00**	**0.78**	2.915	0.95	4.04	**0.011**
sphere_reg_nol_50	20.23	61.51	61.51	0.094	**0.00**	**0.98**	17.089	0.07	2.44	**0.045**
Mean values	-	*51.72*	*56.89*	*0.834*	*0.38*	*1.07*	*63.050*	*0.20*	*2.16*	*0.185*

Zero value of %PDB indicates the solver provides the best found solution among the performed trials, i.e., $\mathcal{L}_{\text{ref}}$.

however, the absolute increase in the computational time is negligible. The overall statistical evaluation using the aggregated data on the solution quality from Table 2 is depicted in Table 4 for non-overlapping and overlapping test

Table 3: Comparison of the TSPNS solvers.

Instance	a_1:	GSOA'		GSOA		GSOA'	
	a_2:	GTSP-GLKH'		GTSP-GLKH		GSOA	
		Length	*Time*	*Length*	*Time*	*Length*	*Time*
sphere_band_rand_nol_25		−	+	−	+	−	+
sphere_band_rand_nol_50		−	+	−	+	−	+
sphere_band_rand_ol_50		−	+	−	+	−	+
sphere_cap_rand_nol_10		−	+	=	+	−	+
sphere_cap_rand_nol_25		−	+	−	+	−	+
sphere_cap_rand_nol_50		−	+	−	+	−	+
sphere_cap_reg_nol_10		=	+	=	+	−	+
sphere_cap_reg_nol_25		−	+	−	+	−	+
sphere_cap_reg_nol_50		−	+	−	+	−	+
sphere_cap_reg_ol_25		=	+	−	+	−	+
sphere_rand_nol_100		=	+	=	+	−	+
sphere_rand_nol_10		−	+	−	+	−	+
sphere_rand_nol_25		−	+	−	+	−	+
sphere_rand_nol_500		=	+	+	+	−	+
sphere_rand_nol_50		=	=	=	=	−	+
sphere_rand_ol_100		=	=	=	=	−	+
sphere_rand_ol_10		=	=	=	=	−	+
sphere_rand_ol_25		=	=	=	=	−	+
sphere_rand_ol_50		=	=	=	=	−	+
sphere_rect_cap_rand_nol_25		=	=	=	=	−	+
sphere_rect_cap_rand_nol_50		=	=	=	=	−	+
sphere_rect_cap_reg_nol_10		=	=	=	=	−	+
sphere_rect_cap_reg_nol_25		=	=	=	=	−	+
sphere_rect_cap_reg_nol_50		=	=	=	=	−	+
sphere_reg_nol_100		=	=	=	=	−	+
sphere_reg_nol_10		=	=	=	=	−	+
sphere_reg_nol_25		=	=	=	=	−	+
sphere_reg_nol_50		=	=	=	=	−	+

The symbols '+', '−', and '=' denote the method a_1 provides statistically better, worse, and equal results than the method a_2, respectively, by means of the solution length and computational time.

Table 4: Comparison of the TSPNS solvers using aggregated results.

Instances	a_1:	GSOA'		GSOA		GSOA'	
	a_2:	GTSP-GLKH'		GTSP-GLKH		GSOA	
		%PDB	%PDM	%PDB	%PDM	%PDB	%PDM
Non-overlapping		=	−	=	=	−	-
Overlapping†		=	=	=	=	=	=

The symbols '+', '−', and '=' denote the method a_1 provides statistically better, worse, and equal results than the method a_2, respectively.

†Note that the results are computed only from six overlapping instances.

instances, from which we can see that GTSP-GLKH and GSOA methods provide competitive solutions. Note that the test instances include only six overlapping instances; hence, the second row of Table 4 is only indicative, and detailed results, computed from 50 trials, are in Table 3.

7.5. Discussion

The post-processing optimization procedure improves the solutions, but the solution quality of the baseline TSPS-LKH remains far below the solvers of the introduced TSPNS. Even though the post-processing optimization procedure is similar to the determination of the waypoints used in the GSOA solver, the procedure improves the GSOA-based solutions, as depicted in Table 3. The GTSP-GLKH solver provides most of the best solutions among the examined instances. However, for the `sphere_rand_ol_100` instance, the determined sequence yields a significantly worse solution than the sequence obtained by the GSOA solver. Besides, it is also the case of the `sphere_reg_nol_100` instance, albeit the differences are not that significant. Overall, the best solutions are found by the GTSP-GLKH and GSOA solvers; however, the GTSP-GLKH solver demands significantly higher com-

putational times than both TSPS-LKH and GSOA. The used GLKH heuristic (Helsgaun, 2013) in the GTSP-GLKH is computationally expensive for large instances such as the `sphere_rand_nol_500`, which took about twenty minutes to obtain a solution. On the other hand, the GSOA-based solver provides a better solution in less than a few seconds, which further motivates us to employ it as a construction heuristic with the follow-up solution improvement using some evolutionary method, which might be a subject for future work.

8. Conclusion

The TSPNS has been introduced as a suitable problem formulation for tasks motivated by the RTI using a team of multi-rotor unmanned aerial vehicles. The introduced TSPNS has been addressed by two types of approaches, the sampling-based transformation methods and a novel direct solution based on the unsupervised learning of the GSOA. Besides, we propose improving post-processing optimization, which is most effective for the transformation methods that otherwise provide relatively poor solutions unless dense sampling is employed. Based on the evaluation results, the proposed GSOA-based solver performs similarly or provides about units of percentage points worse solutions than the discretized GTSP-based approach with relatively dense sampling of ten samples per target region. However, the computational requirements of the GSOA are about two orders of magnitude lower than the computational requirements of the GTSP-based solution using the heuristic GLKH solver. Therefore, the proposed GSOA-based solver can be preferred in deployments where the available computational time is very limited. On the other hand, the GTSP-based solver fits scenarios where the

computational cost is not an issue or the number of regions is low. However, for the field deployments and onsite setups of the RTI, the proposed GSOA solver is much more suitable as it requires typically less than a few seconds even for the largest solved instance and not tens of seconds or even tens of minutes as the GTSP-GLKH, which would compare to the overall flight time of the vehicles.

Acknowledgments

This work was supported by the Czech Science Foundation under research project No. 19-20238S. The support of project No. DG18P02OVV069 under program NAKI II to the third author is also acknowledged. 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. The authors would also like to acknowledge the support of Petr Váňa for the help with the implementation of the proposed method and its empirical evaluation.

References

Applegate, D. L., Bixby, R. E., Chvatal, V., and Cook, W. J. (2007). *The Traveling Salesman Problem: A Computational Study (Princeton Series in Applied Mathematics)*. Princeton University Press.

Arcuri, A. and Briand, L. (2011). A practical guide for using statistical tests to assess randomized algorithms in software engineering. In *33rd International Conference on Software Engineering (ICSE)*, pages 1—10.

Arkin, E. M. and Hassin, R. (1994). Approximation algorithms for the geometric covering salesman problem. *Discrete Applied Mathematics*, 55(3):197–218.

Aybars, U. (2008). Path planning on a cuboid using genetic algorithms. *Information Sciences*, 178(16):3275–3287.

Croes, G. A. (1958). A method for solving traveling-salesman problems. *Operations Research*, 6(6):791–812.

De Berg, M., Gudmundsson, J., Katz, M. J., Levcopoulos, C., Overmars, M. H., and van der Stappen, A. F. (2005). Tsp with neighborhoods of varying size. *Journal of Algorithms*, 57(1):22 – 36.

Dumitrescu, A. and Mitchell, J. S. (2003). Approximation algorithms for tsp with neighborhoods in the plane. *Journal of Algorithms*, 48(1):135–159.

Elbassioni, K., Fishkin, A. V., and Sitters, R. (2009). Approximation algorithms for the euclidean traveling salesman problem with discrete and continuous neighborhoods. *International Journal of Computational Geometry & Applications*, 19(02):173–193.

Eldem, H. and Ülker, E. (2017). The application of ant colony optimization in the solution of 3d traveling salesman problem on a sphere. *Engineering Science and Technology, an International Journal*, 20(4):1242–1248.

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.

Faigl, J., Kulich, M., Vonásek, V., and Přeučil, L. (2011). An application of

the self-organizing map in the non-euclidean traveling salesman problem. *Neurocomputing*, 74(5):671–679.

Faigl, J., Váňa, P., and 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(3):2439–2446.

Faigl, J., Vonásek, V., and Přeučil, L. (2013). Visiting convex regions in a polygonal map. *Robotics and Autonomous Systems*, 61(10):1070–1083.

Gentilini, I., Margot, F., and Shimada, K. (2013). The travelling salesman problem with neighbourhoods: Minlp solution. *Optimization Methods and Software*, 28(2):364–378.

Gudmundsson, J., , Gudmundsson, J., and Levcopoulos, C. (2000). Hardness result for tsp with neighborhoods. Technical report, Lund University.

Gulczynski, D. J., Heath, J. W., and Price, C. C. (2006). The close enough traveling salesman problem: A discussion of several heuristics. In Alt, F. B., Fu, M. C., and Golden, B. L., editors, *Perspectives in Operations Research: Papers in Honor of Saul Gass' 80th Birthday*, pages 271–283. Springer US, Boston, MA.

Helsgaun, K. (2000). An Effective Implementation of the Lin-Kernighan Traveling Salesman Heuristic. *European Journal of Operational Research*, 126(1).

Helsgaun, K. (2013). GLKH, ver. 1.0. http://akira.ruc.dk/~keld/research/GLKH/. [cited 22 Jan 2021].

Helsgaun, K. (2015). Solving the equality generalized traveling salesman problem using the lin–kernighan–helsgaun algorithm. *Mathematical Programming Computation*, 7(3):269–287.

Helsgaun, K. (2018). LKH, ver. 2.0.9. http://www.akira.ruc.dk/~keld/research/LKH/. [cited 22 Jan 2021].

Krátký, V., Petráček, P., Spurný, V., and Saska, M. (2020). Autonomous reflectance transformation imaging by a team of unmanned aerial vehicles. *IEEE Robotics and Automation Letters*, 5(2):2302–2309.

Mennell, W. (2009). *Heuristics for Solving Three Routing Problems: Close-Enough Traveling Salesman Problem, Close-Enough Vehicle Routing Problem, Sequence-Dependent Team Orienteering Problem*. PhD thesis, University of Maryland.

Ouyang, X., Zhou, Y., Luo, Q., and Chen, H. (2013). A novel discrete cuckoo search algorithm for spherical traveling salesman problem. *Applied Mathematics & Information Sciences*, 7(2):777.

Smith, S. L. and Imeson, F. (2017). GLNS: An effective large neighborhood search heuristic for the generalized traveling salesman problem. *Computers & Operations Research*, 87:1–19.

Uğur, A., Korukoğlu, S., Çalıskan, A., Cinsdikici, M., and Alp, A. (2009). Genetic algorithm based solution for tsp on a sphere. *Mathematical and computational applications*, 14(3):219–228.