GSOA: Growing Self-Organizing Array – Unsupervised Learning for the Close-Enough Traveling Salesman Problem and Other Routing Problems

Jan Faigl

Department of Computer Science, Faculty of Electrical Engineering, Czech Technical University in Prague, Technická 2, 166 27, Prague 6, Czech Republic

Abstract

This paper presents a novel unsupervised learning procedure called the *Growing Self-Organizing Array* (GSOA) that is inspired by principles of the self-organizing maps for the Traveling Salesman Problem (TSP). The proposed procedure is a consolidation of principles deployed in solution of several variants of the generalized TSP with Neighborhoods (TSPN) for which the main ideas of the proposed unsupervised learning already demonstrates a wide range of applicability. The herein presented learning procedure is a conceptually simple algorithm which outperforms previous self-organizing map based approaches for the TSP in terms of the solution quality and required computational time. The main benefit of the proposed learning procedure is in solving routing problems that combine a combinatorial solution of some variant of the TSP with the continuous optimization, i.e., problems where it is needed to determine a sequence of visits to the given sets with determination of the particular waypoint location from each (possibly infinite) set. Low computational requirements of the proposed method are demonstrated in a solution of the Close-Enough Traveling Salesman Problem (CETSP), which is a special case of the TSPN with the disk-shaped neighborhoods. The results indicate the proposed procedure provides competitive solutions to the existing heuristics while it is about one order of magnitude faster and at least about two orders of magnitude faster than a heuristic solution of the discretized variant of the CETSP considered as the Generalized TSP.

Keywords: Unsupervised learning, Self-organizing Map, TSP, GSOA, Data collection planning

1. Introduction

Routing problems are traditionally studied in operational research, and probably the most studied combinatorial routing problem is the Traveling Salesman Problem (TSP) which stands as follows. Given a set of locations, the problem is to determine the shortest tour visiting all the locations exactly once and returning to the starting location. The TSP is known to be NP-hard unless (P=NP) and several approximation algorithms and heuristics have been proposed [1]. In addition to combinatorial heuristics such as the efficient implementation of the Lin-Kernighan heuristic [2], genetic algorithms [3], ant colony optimization [4], and simulated annealing [5], also neural networks have been applied to combinatorial routing problems [6, 7, 8, 9, 10].

Neural networks approaches are of the particular interest in this paper; however, Hopfield-based approaches have been criticized as a suitable technique for combinatorial optimizations [11]. The other type of the neural networks deployed in combinatorial optimization problems is the Self-Organizing Map (SOM) [12]. SOM-based approaches exhibit promising results [13] and several improved modifications of the original ideas have been proposed during the last decades, e.g., see overviews of existing work in [14, 15, 16]. It is reported in the literature that SOM-based approaches have low-computational requirements and can solve large-scale problems, e.g., exploiting massively parallel computational resources [17]. However, the solution quality is usually worse than for more computationally demanding evolutionary methods.

In this paper, a novel unsupervised learning procedure for routing problems is presented. The proposed method is called *Growing Self-Organizing Array* (GSOA) and its main principles follow the existing work on SOM for the TSP, but it is mostly motivated by data collection planning where a robotic vehicle is requested to collect data from a given set of sensing sites [18]. In this type of problems, it may not be necessary to visit the sites precisely, and the robotic vehicle may use remote sensing or wireless communication to collect the required data [19]. Therefore, rather than the ordinary TSP, the problem can be formulated as the TSP with Neighborhoods (TSPN), and more specify with the disk-shaped neighborhood as the Close-Enough TSP (CETSP) [20].

The CETSP stands to determine the closed shortest path that passes each sensing site within a specified sensing range δ. Since the disk-shaped neighborhood is a continuous set, the main computational challenges of the CETSP are related to the infinite possibilities how to visit the δ-neighborhood of each sensing site together with the combinatorial optimization of the sequencing part of the underlying TSP.

In addition to heuristic approaches, e.g., [20, 21], the CETSP can be addressed by a sampling of each particular neighborhood into a finite set of possible locations. Then, a solution of the CETSP is found as a solution of the related Generalized TSP (GTSP) [22], e.g., by heuristics [23, 24]. However, such an explicit sampling of the possible locations increases the size of the problem which becomes quickly computationally demanding despite possibly more sophisticated sampling schema [25].

The proposed GSOA algorithm originates from the ideas developed for the TSPN [26, 27] and further improved in other deployments to solve data collection planning formulated as the Prize-Collecting TSP (PC-TSP) [28] generalized to the PC-TSP with Neighborhoods (PC-TSPN) [29] and also the Orienteering Problem (OP) [30, 31] and its generalization to the Orienteering Problem with Neighborhoods (OPN) [32, 33]. All this effort results in a conceptually simple algorithm that does not require specific parameters tuning as the previous SOM-based approaches. Moreover, it follows one of the early ideas of dynamic SOM structure [9, 34], and the number of neurons is adaptively adjusted during the learning, which is a desirable feature for solving PC-TSP(N) and OP(N) where the tour does not necessarily visit all the sensing sites.

The recent advancements on applying the unsupervised learning principles to the aforementioned routing problems and the used self-adjustment of the number of neurons are the main motivation to call the developed learning procedure the *Growing Self-Organizing Array* (GSOA). Although the GSOA is inspired by the SOM for the TSP, and it shares the usage of neighboring function in the adaptation phase, the structure of the network is specifically designed for routing problems as an array of nodes with the output space identical to the input space, contrary to the usually 2D grid cells mapping of a high-dimensional input space into 2D space for the ordinary Kohonen's networks. Therefore the proposed learning procedure is called the GSOA to emphasize its focus on routing problems.

Even though the herein in presented unsupervised learning principles of the GSOA already demonstrates a wide range of applicability in several routing problems motivated by surveillance and data collection planning [35, 33, 36], they are not described elsewhere in a comprehensive way. Moreover, the performance of the GSOA has not been compared with SOM-based solutions to the TSP, and therefore, in this paper, the GSOA is demonstrated in a solution of the selected instances from the TSPLIB [37] (with up to 2392 locations) and compared with one of the recent SOM-based algorithms for the TSP [16] (further called ORC-SOM) and one of the most sophisticated SOM-based algorithms called Co-adaptive Net [14]; both reported to be the best performing SOM-based algorithms in the literature. However, the main advancements of the proposed GSOA are in a solution of routing problems that combines combinatorial optimization of the underlying TSP with continuous optimization, e.g., in determining the most suitable waypoint locations from an infinite set of possibilities within the disk-shaped neighborhood of each sensing location in the solution of the CETSP. Therefore, the performance of the GSOA in the CETSP is compared with the best performing heuristic from 15 evaluated methods in [21] (further denoted as SZ2 from the Steiner Zones [21]). Besides, it is compared with the sampling-based approach and the GTSP formulation solved by [24]. The presented evaluation is based on several representative instances with up to 1001 sensor locations with various radii of the disk-shaped neighborhood. Based on the performed evaluation and comparison with the existing solutions, the proposed GSOA provides solutions of the competitive quality while the computational requirements are significantly lower.

The paper is structured as follows. An overview of the related approaches to the CETSP together with the brief overview of the most relevant SOM-based solutions to the TSP and other routing problems are presented in the next section to clarify the main contributions of this paper. In Section 3, the TSP and CETSP are formally defined. The proposed GSOA is presented in Section 4. Results on the evaluation of the GSOA in the selected instances of the TSPLIB and CETSP are reported in Section 5. Conclusion and final remarks are in Section 6.

2. Related Work

The herein studied Close-Enough Traveling Salesman Problem (CETSP) is motivated by data collection and environment monitoring missions. The CETSP has been firstly described by Gulczynski et al. in [20] as a suitable problem formulation to plan data collection paths for retrieving data about electricity, water or gas consumption using wireless communication. The authors of [38] consider the CETSP in monitoring scenarios for forest fire detection using aerial vehicles. In [35], a variant of the CETSP is used to plan surveillance missions and to exploit the field of view of the utilized visual sensor to shorten the required trajectory by avoiding a precise visitation of the expected locations of the objects of interest by the aerial vehicles. Another example of data collection planning for underwater vehicles to remotely retrieve data from the sampling stations located on the ocean floor is reported in [29].

The CETSP is a special case of the Traveling Salesman Problem with Neighborhoods (TSPN) in which the requested path has to visit each of the n sensor sites that form the respective neighborhoods. The TSPN is NP-hard as the probem becomes the ordinary TSP for the zero neighborhoods. In the CETSP, the regions are restricted to disks; however, in the TSPN, a particular neighborhood can also be a straight line, convex or even non-convex regions. Several approximate solutions have been proposed for the TSPN with restricted variants of the neighborhood such as disjoint unit disks that are arbitrarily connected [39], disjoint convex fat neighborhoods [40]. Nevertheless, the TSPN is shown to be APX-hard [41] and cannot be approximated to within a factor $2 - \epsilon$, where $\epsilon > 0$. Notice the Mixed Integer Non-linear Programming formulation of the TSPN is too computationally demanding [42], and thus heuristic approaches are studied.

Six heuristics to solve the CETSP with disks of the same radius δ are proposed in [20]. All the heuristics share the following three steps. First, a feasible supernode set F is determined such that each sensing site is within the δ distance from at least one point of F. Then, a solution of the TSP on F is found as a feasible solution of the original CETSP. Finally, the feasible path found in the previous step is shortened. The heuristics differ how the set F is determined, and three of them are based on hexagonal tiling. The other heuristic uses a binary matrix to represent that two nodes are closer than δ, and the additional construction of F is called sweeping circle heuristic where circles are utilized instead of hexagons. The last heuristic determines supernodes using intersections of the neighborhoods that are called Steiner zones.

Two heuristics based on the idea of supernodes are proposed in [43] where the authors used a convex hull and merged tiling (clustering) for the creation of the supernodes. The TSP tour is then improved using simulated annealing algorithm. The reported results are for problem instances with up to 1000 sensing sites with the computational requirements below one second. An evolutionary based approach is proposed in [38], and it is evaluated in five scenarios with up to 300 sensing sites. Although the approach provides best solutions in the evaluated scenarios, it is computationally very demanding.

A systematic evaluation of the existing approaches to the CETSP is presented in [21] where the author also proposes improved heuristics based on Steiner zones. The evaluation is performed in three sets of problem instances. The first set consists of CETSP instances created from seven benchmarks selected from the TSPLIB [37] considered with three values of the overlap ratio R. The value of R is the ratio of the length of the smallest square containing all n disks and the radius of the disk [21]. Three radius patterns are considered to create 21 CETSP instances for *very low* overlap ratio ($R = 0.02$), *moderate* ($R = 0.10$), and *very high* ($R = 0.30$) ratios. Notice that with increasing value of R the neighborhood disks are more and more overlapping, and thus a solution of the related TSP can be very far from the optimal solution of the CETSP.

The second set consists of 27 instances of the CETSP where the particular radius of the disk is constant for all the sensing sites, and thus the set represents problems with various overlap ratios. The third set consists of 14 instances where each neighborhood (disk) has a different radius, and therefore, this set represents problems with arbitrary radii. The total number of problem instances used in the evaluation [21] is 62 with the size of the problems from 17 to 1001 sensing sites. These instances form a set of representative benchmarks that are also used in the evaluation of the herein proposed GSOA in solving the CETSP, see Section 5.

15 heuristics are considered in the evaluation [21] that includes the heuristics mentioned above and also sampling-based approach where the CETSP is transformed to the Generalized TSP (GTSP). A particular instance of the GTSP is created by sampling points at the perimeter of each disk, and the author reports the number of samples should be at least 24 per each disk; otherwise, a poor solution would be found. It is also reported that the best-found solutions are provided by the GTSP-based approach; however, it is computationally very demanding, and the author mentions that some of the problems have been found in hours, weeks, or even months. The author of [21] reports that the best performing approach regarding the trade-off between the computational requirements and quality of found solutions is his improved heuristic based on Steiner zones denoted SZ2.

The computational requirements of the GTSP-based solution of the CETSP are addressed in [25] where authors propose a new discretization schema to improve the solution quality by more samples while keeping the computational requirements low. Although the reported computational times are in units or tens of seconds using iCore5 running at 2.9 GHz, the presented results are only for relatively small problems with up to 30 sensing sites.

2.1. Self-Organizing Maps for the TSP

The proposed unsupervised learning procedure of the GSOA is inspired by the principles of self-organizing maps (SOMs) for the TSP, and therefore, a brief overview of the existing approaches is presented here. The overview is not intended to be exhaustive because detailed description of the previous SOM based solutions to the TSP is presented in [14, 44, 16]. Therefore the main intention of this section is to provide an origin of the ideas used in the proposed GSOA and to clarify the contribution with respect to the previous work.

SOM-based solution to the TSP is very similar to the Elastic net [8] proposed in 1987. However, Smith noted [45] that before this work has been published, Fort had been working on using Kohonen's SOM to solve the TSP [10]. Moreover, at the same time, Angéniol et al. [9] propose the pioneering SOM-based solution to the TSP in 1988 which uses a dynamic structure of the network. Later on, a growing structure has been further used in FLEXMAP proposed by Fritzke and Wilke [34]; however, the approach does not utilize a deletion mechanism. On the other hand, most of the SOM-based approaches to the TSP use a fixed structure with the number of neurons equal, but mostly greater than the number of the sensing sites n. A recommended number of neurons M is between $2n \leq M \leq 3n$, e.g., $M = 2.5n$ [46], which is also the observed maximal number of the required neurons in FLEXMAP [34].

The SOM for the TSP follows the standard Kohonen's SOM [12] with few modifications due to the solution of the routing problems. It can be considered as a two-layered neural network, where the input layer serves for presenting the sensing sites (input signals) towards which the network is adapted using the unsupervised learning. However, the output layer is an array of neurons, and the neurons' weights directly represent point locations in the input space. The neurons can be connected into a ring because of the array structure of the output layer, and thus the ring represents a path in the input space.

The learning is usually performed in a sequence of learning epochs, where for each learning epoch, all input signals are presented to the network, and the best matching neuron is selected as the winner neuron. Then, the winner neuron is adapted towards the particular input signal together with its neighboring neurons that are adapted to the input signal with decreasing power according to the neighboring function. Thus, the learning epoch consists of the winner selection and adaptation phases that are repeatably performed for every sensing site.

The complexity of a single learning step is polynomial in n, and it can usually be bounded by $O(n^2)$ [16]. Once the network is stabilized, the winner neurons match the sensing sites, and a solution of the TSP can be retrieved by traversing the ring, i.e., following the array structure, and using the sensing site associated to the winner neurons. The above-presented description represents a basic SOM schema for solving the TSP utilized by most of the existing SOM-based approaches, and the most related and remarkable ones are as follows.

In 1997, Somhom et al. propose their variant of the SOM-based solution to the TSP [46] which significantly outperforms the previous approaches [9, 47] including the Elastic net [8] regarding the quality of found solutions and also the required computational time. The important feature of the Somhom's approach is an explicit inhibition of the neurons, which avoids to repeatably select a single neuron as a winner neuron multiple times during a single learning epoch. Although this can be considered as a tiny detail, it provides the anytime property to the SOM algorithm, because it is no longer necessary to wait for the convergence of the network to a stable state to obtain a solution. A feasible solution of the TSP can be retrieved from the network after each learning epoch using the associated sensing site to their respective winner neurons because each sensing site has a unique winner neuron. Besides, the approach [46] was the best performing SOM for the TSP at that time.

Probably the most sophisticated SOM-based solution to the TSP is the Co-adaptive Net proposed by Cochrane and Beasley in 2003 [14]. In addition to several improvements, the computational requirements are decreased by a restricted search for the best matching neuron and the winner neurons are searched from the whole network only every β learning epoch [14]. The authors report improved results over the previous approaches using TSPLIB benchmarks with instances up to 1400 sensing sites.

One of the relatively recent SOM-based approaches for the TSP using conventional CPU (e.g., in addition to parallel implementations [17]) is the ORC-SOM [16]. It follows the previous SOM schemata, but the authors introduce a displacement function for a smooth transition from the tour outlining (by overall competitive rule) to the tour refining (by the regional competitive rule). The reported results are for selected benchmarks from the TSPLIB with up to 2392 sensing sites. In comparison to the previous approaches (including the Co-adaptive Net [14]), the ORC-SOM is reported [16] to be the best performing pure SOM-based procedure regarding the quality of solutions at the expense of higher computational requirements than the Co-adaptive Net.

Regarding the existing SOM-based solvers for the TSP, to the best of our knowledge, the most representative are the Co-adaptive Net [14] and ORC-SOM [16] because they are reported in the literature as the best performing purely SOM-based algorithms in comparison to other SOM approaches. Although there can be found more recent papers concerning SOM for the TSP, they are mostly based on the previous work applied in some extended problems such as [48], or they are combined with other techniques, e.g., [44, 49], or do not provide significantly better solutions [50] than the results presented elsewhere and also here. Therefore, Co-adaptive Net and ORC-SOM are the selected competitors to the proposed GSOA in the evaluation of the unsupervised learning based approaches in TSP instances.

The selected TSP benchmarks are the TSPLIB problems considered in [16] as they represent various types of scenarios including instances with "clusters" of closely located sensing sites in which SOM approaches usually do not provide good performance. Note that none of the SOM-based approaches outperforms the tuned implementation of the most power-full combinatorial Lin-Kernighan heuristics [51] developed by Helsgaun in [2] and denoted LKH. Thus, the herein presented comparison for the TSPLIB instances is mostly to show a small step forward in the solution quality provided by unsupervised learning based approaches. On the other hand, the main advantage of the unsupervised learning, and especially of the proposed GSOA, is in the solution of combinatorial problems that also include a continuous optimization part, e.g., a solution of the CETSP.

2.2. GSOA Motivation and Background

The herein proposed GSOA can be considered as a result of the evolution of the author work on the SOM-based solution of the multi-goal planning introduced in [52], where Somhom's approach [46] has been deployed in solving routing problems in the polygonal domain. Later on, the idea of determining an alternate location (to be visited) during the winner selection has been introduced in [53] to solve a variant of the coverage planning problem. Then, the principles have been combined in a solution of the TSPN [26] including problems with non-convex regions [27]. Based on the review of the existing SOM schemata and parameters, a novel schema has been proposed in [54] which outperforms the previous works regarding the solution quality and computational requirements.

The developed schema [54] has been then deployed in a solution of data collection planning missions formulated as the PC-TSPN in [29] and further extended for the problems where the penalties for not visited sensing sites depend on the determined subset of the sensing sites to be visited [55]. Besides, the schema as been employed in a solution of the so-called Dubins TSP (DTSP), where the path connecting two sensing sites has to respect kinematic constraints of the utilized aerial vehicle [56]. Then, it has been employed in the solution of the DTSP with Neighborhoods for which the unsupervised learning based approach provides the best trade-off between the solution quality and the required computational time [35]. In addition, the developed unsupervised learning procedure has been considered in the Orienteering Problem (OP) [57, 32] and the Dubins OP [58] and both problems also with disk-shaped neighborhoods in [33] and [59], respectively. Finally, a solution of the OP with Neighborhoods has been utilized for online planning in [60].

Although almost all the aforementioned deployments build on the ideas firstly introduced in [26, 53], the learning procedure itself slightly evolves in each particular deployment. It has also been observed that the approaches for the TSP(N), PC-TSP(N) and OP(N) share common properties that enable to use the principles of unsupervised learning in various types of routing problems, especially where combinatorial optimization of determining a sequence of visits to the sensing locations is combined with the selection of the subset of the visited sensors together with determining particular locations to visit the sensors. Therefore, these properties have been synthesized, and the *Growing Self-Organizing Array* (GSOA) is proposed with the perspective to simplify the description of the further algorithms following these ideas. Besides, the unsupervised learning of the GSOA has not been deployed in the CETSP nor evaluated in the standard TSPLIB benchmarks. Therefore, the main contribution of this paper is in the introduction of the GSOA as a general unsupervised learning technique for routing problems that are variants of the TSP. The performance and benefits of the GSOA are supported in comparison with representative SOM-based approaches using TSPLIB benchmarks and comparison to existing heuristics to the CETSP in a huge set of problems proposed in [21].

3. Problem Formulations

Even though the TSP can be considered as a well-know problem, a formal definition of the problem is presented here to introduce the notation and clarify the optimization criterion. Besides, the main motivation of the proposed unsupervised learning for the TSP-like routing problems, which is specifically deployed in the Close-Enough TSP (CETSP), are problems arising in data collection planning where a vehicle is requested to visit a set of sensor sites to retrieve data from them.

The set of n sensor locations to be visited is denoted $S = \{\boldsymbol{s}_1, \ldots, \boldsymbol{s}_n\}$, where the notation is slightly overloaded and the sensor s_i is denoted as the sensor location $\boldsymbol{s}_i$ for simplicity. Following the Euclidean TSP, each sensor is located in a plane $\boldsymbol{s}_i \in \mathbb{R}^2$ and the cost of the travel between any two points $\boldsymbol{p}_1, \boldsymbol{p}_2 \in \mathbb{R}^2$ is the Euclidean distance $\|(\boldsymbol{p}_1, \boldsymbol{p}_2)\|$.

In the TSP, it is requested to determine a shortest tour visiting all the sensors S starting from some initial location and returning to the same location. The sequence of visits to the sensors S can be described as a permutation $\Sigma = (\sigma_1, \ldots, \sigma_n)$ of the sensor labels such that $1 \le \sigma_i \le n$ and $\sigma_i \neq \sigma_j$ for any two $\sigma_i, \sigma_j \in \Sigma$. It is assumed w.l.o.g. the initial (and terminal) location of the vehicle is $\boldsymbol{s}_1$. Then, the TSP can be formally defined as the combinatorial optimization problem to determine the sequence Σ minimizing the total tour length $L(\Sigma, S)$.

Problem 3.1 (TSP).

$$\begin{aligned}
&\textit{minimize}_{\ \Sigma}\ L(\Sigma, S)\\
&L(\Sigma, S) = \sum_{i=1}^{n-1} \left\|(\boldsymbol{s}_{\sigma_i}, \boldsymbol{s}_{\sigma_{i+1}})\right\| + \|(\boldsymbol{s}_{\sigma_n}, \boldsymbol{s}_{\sigma_1})\|\\
&\textit{subject to}\\
&\qquad 1 \le \sigma_i \le n;\ \sigma_i \in \Sigma;\ \boldsymbol{s}_{\sigma_i} \in S\\
&\qquad\qquad \boldsymbol{s}_{\sigma_1} = \boldsymbol{s}_1
\end{aligned}$$

In the CETSP, motivated by remote data reading from the sensors, the sensor $\boldsymbol{s}_i \in S$ is considered to be visited if the data collection path is in the distance less or equal to the communication (sensing) range δ. Thus, the range forms a disk-shaped neighborhood with the radius δ around each sensor. Moreover, each sensor can have its individual range, and therefore, we can distinguish individual δ_i corresponding to the particular sensor $\boldsymbol{s}_i$. The problem can be formulated as a combination of the combinatorial optimization to determine the sequence Σ of visits to the sensors S together with the determination of the most suitable locations $\boldsymbol{p}_i \in \mathbb{R}^2$ such that each $\boldsymbol{s}_i$ has its location $\boldsymbol{p}_i$ within the distance δ_i, i.e., $\|(\boldsymbol{s}_i, \boldsymbol{p}_i)\| \le \delta_i$.

Problem 3.2 (CETSP).

$$\begin{aligned}
&\textit{minimize}_{\ \Sigma,P}\ L(\Sigma, P, S)\\
&L(\Sigma, P, S) = \sum_{i=1}^{n-1} \left\|(\boldsymbol{p}_{\sigma_i}, \boldsymbol{p}_{\sigma_{i+1}})\right\| + \left\|(\boldsymbol{p}_{\sigma_n}, \boldsymbol{p}_{\sigma_1})\right\|\\
&\textit{subject to}\\
&\qquad \Sigma = (\sigma_1, \ldots, \sigma_n);\ 1 \le \sigma_i \le n\\
&\qquad S = \{\boldsymbol{s}_1, \ldots, \boldsymbol{s}_n\};\ \boldsymbol{s}_i \in \mathbb{R}^2\\
&\qquad P = \{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n\};\ \boldsymbol{p}_i \in \mathbb{R}^2\\
&\qquad \boldsymbol{s}_{\sigma_1} = \boldsymbol{s}_1 \text{ and } \|(\boldsymbol{s}_i, \boldsymbol{p}_i)\| \le \delta_i,\ \text{for}\ \ 0 \le i \le n
\end{aligned}$$

The determined locations P are points of the data collection path and they are further referred as waypoints.

Following [21], we can assume $\delta_1 = 0$ because the vehicle starts at the specific location and it is also requested the vehicle returns to that prescribed location.

4. Growing Self-Organizing Array (GSOA)

The proposed *Growing Self-Organizing Array* (GSOA) for routing problems consists of the route representation as a sequence of nodes accompanied by an unsupervised learning procedure that is inspired by the Self-Organizing Map (SOM). A slightly different notation from SOM-based approaches is introduced in the following section for clarity and simplification of learning procedure. A detail description of the individual parts of the GSOA for solving instances of the TSP and CETSP is presented in Section 4.2, the convergence properties are discussed in Section 4.3, and computational requirements in Section 4.4. Beside the GSOA for the TSP and CETSP, the main idea of the straightforward extension of the GSOA for routing problems with the selection of the target locations, such as the PC-TSP or OP, is presented in Section 4.5 to motivate for a further application of the GSOA in additional routing problems.

4.1. Data Representation and Notation

The GSOA is a set of nodes $\{\nu_1, \ldots, \nu_M\}$ organized in a one-dimensional structure that is called a ring of nodes and an example of the ring in a solution of the CETSP is visualized in Fig. 1a. During the learning, new winner nodes are created according to the shortest distance of the ring to the particular sensing locations $\boldsymbol{s}_i \in S$, and therefore, each node ν_i represents a particular location $\boldsymbol{\nu}_i$ in the same space as the sensing locations. Hence, during the learning, the nodes form a path that evolves in the input space to visit the sensors S. The evolution is performed by the unsupervised learning procedure in which new winner nodes are repeatably determined for each sensor of S. A single adaptation of the ring to all sensor locations S is called a learning epoch.

Each node ν_i is associated with its locations $\boldsymbol{\nu}_i$ and the particular sensor s for which the node has been determined as the winner node. Moreover, each newly created winner node ν^* is also associated with a particular waypoint location $\boldsymbol{s}_p$ at which the corresponding sensor s can be covered, i.e., $\boldsymbol{s}_p$ is within a distance to $\boldsymbol{s}$ shorter than the range δ. The waypoint $\boldsymbol{s}_p$ is determined during the winner selection as the point $\boldsymbol{s}_p$ on the segment $(\boldsymbol{p}_s, \boldsymbol{s})$, where $\boldsymbol{p}_s$ is the closest point of the ring to $\boldsymbol{s}$, such that $\boldsymbol{s}_p$ is in the corresponding neighborhood of $\boldsymbol{s}$ and $\boldsymbol{s}$ can be covered from $\boldsymbol{s}_p$, see Fig. 1b. Overall, each node $\nu_i \in \mathcal{N}$ consists of the location $\boldsymbol{\nu}_i$ that is initialized to $\boldsymbol{p}_s$ when the node is created, the waypoint $\boldsymbol{s}_p$, and it is further associated with the respective sensor s. In the case $\delta = 0$, the waypoint is directly the sensor location $\boldsymbol{s}$.

Since nodes are organized in the one-dimensional structure $\mathcal{N} = (\nu_1, \ldots, \nu_M)$, a feasible route connecting the waypoints that are in the respective δ-neighborhood of the corresponding sensor locations (or directly the sensor locations themselves) can be created by traversing the ring and using the waypoints associated with the nodes. Hence, a ring of nodes $\mathcal{N}$ represents the requested route as visualized in Fig. 1c. Notice, the ring can be considered as an array of nodes; however, it can be efficiently implemented as a double linked list because the nodes are accessed only by the sequential traversing of the ring $\mathcal{N}$. Besides, new winner nodes are added to the ring during the learning, and the previous nodes are removed from the ring at the end of each learning epoch.

The following notation is used to distinguish between the nodes, nodes locations, and the waypoints associated with the nodes. The ring of nodes $\mathcal{N}$ consists of nodes $\nu_i \in \mathcal{N}$ and each ν_i is further associated with two points that are in the same space as the sensor locations. Each node ν_i has its location $\boldsymbol{\nu}_i$, and therefore, in the case of the Euclidean TSP in a plane, $\boldsymbol{\nu}_i$ is a two dimensional vector $\boldsymbol{\nu}_i \in \mathbb{R}^2$, $\boldsymbol{\nu}_i = (x_{\nu_i}, y_{\nu_i})$. The node ν_i is further associated with the waypoint $\boldsymbol{s}_p$, which is also two dimensional vector $\boldsymbol{s}_p \in \mathbb{R}^2$, and it is denoted as $\nu_i.\boldsymbol{s}_p$ whenever it is needed to explicitly mention with which node $\boldsymbol{s}_p$ is associated to.

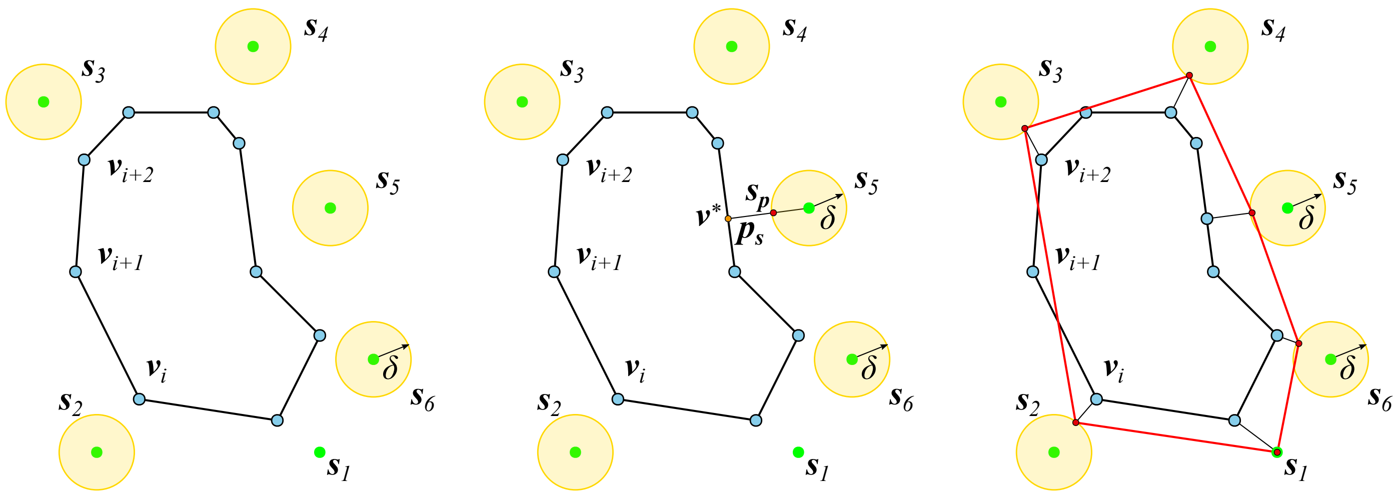
Regarding Problem 3.2, the associated waypoints to the nodes are the requested waypoints $\boldsymbol{p}_i \in P$; however, the sequence of visits to the sensors is prescribed by the sequence of nodes in the ring $\mathcal{N}$. Hence, after traversing the ring, the associated $\boldsymbol{s}_p$ to the winner nodes become the waypoints P corresponding to the respective sensing locations.

4.2. Unsupervised Learning of the GSOA

The GSOA is a growing array structure $\mathcal{N}$ which is iteratively adapted towards the sensor locations S. The learning procedure starts with a single node ν_1 which location $\boldsymbol{\nu}_1$ can be initialized as the starting location $\boldsymbol{s}_1$ or it can be set to the centroids of the sensor locations S. The learning procedure is performed in a finite number of learning epochs, where each learning epoch is an adaptation of the ring of nodes $\mathcal{N}$ to all the sensor locations S. The sensor locations are selected in a random order to avoid local optima similarly as in [46]. For each $s \in S$ a new winner node is determined that is then adapted towards s. After the adaptation of the ring to all n sensors of S (i.e., after the end of a single learning epoch), the ring has n new nodes, and all the previous nodes in the ring are removed to balance the number of nodes with the number of sensors. The learning procedure is then repeated for a fixed number of learning epochs i_{max}.

An overview of the GSOA unsupervised learning procedure is depicted in Algorithm 1[1]. The procedure consists of three parts: (i) winner node determination; (ii); adaptation of the winner; and (iii) extraction of the solution after

[1]An implementation of the algorithm is available at `https://purl.org/faigl/sw`.



(a) Ring of nodes (b) Determination of the winner nodes (c) Route represented by the ring of nodes

Figure 1: (**a**) The sensor locations $\boldsymbol{s} \in S$ are visualized as small green disks with its neighborhood as a surrounding yellow disk of the radius δ. The locations $\boldsymbol{\nu}_i$ of the nodes $\nu \in \mathcal{N}$ are shown as small disks connected by black straight line segments into a ring. (**b**) An example of the determination of the new winner node ν^* to the sensor location $\boldsymbol{s}_5$ as the closest point $\boldsymbol{p}_s$ of the ring to $\boldsymbol{s}_5$. The corresponding waypoint $\boldsymbol{s}_p$ to cover $\boldsymbol{s}_5$ is a point on the segment $(\boldsymbol{p}_s, \boldsymbol{s}_5)$ that is inside the δ-neighborhood of $\boldsymbol{s}_5$. (**c**) A feasible solution of the CETSP (and TSP) is represented as a route (shown in red) connecting the associated waypoints (shown as small red disks) of the winner nodes of the ring, and it can be constructed by traversing the ring $\mathcal{N}$.

```
Algorithm 1: GSOA – Unsupervised Learning
   Input: $S = \{\boldsymbol{s}_1, \ldots, \boldsymbol{s}_n\}$ – a set of sensor locations to be visited, each with particular disk-shaped $\delta$-neighborhood
   Input: $\boldsymbol{s}_1$ – the requested initial and terminal location with $\delta = 0$
   Input: $(\sigma, \mu, \alpha)$ – the learning parameters: the initial value of the learning gain $\sigma = 10$, the gain decreasing rate
          $\alpha = 0.0005$, and learning rate $\mu = 0.6$
   Input: $i_{max}$ – the maximal number of learning epochs, i.e., $i_{max} = 150$
   Output: $(\Sigma, P)$ – $\Sigma$ defines the order of visits to the sensors $S$ and $P$ are the waypoints to visits the sensors
 1 $\mathcal{N} \leftarrow \{\nu_1\}$ such that $\boldsymbol{\nu}_1 = \boldsymbol{s}_1$                      // init the ring
 2 $i_{max} \leftarrow \min(i_{max}, 1/\alpha)$                  // ensure $\sigma$ will always be above 0
 3 $i \leftarrow 0$                                             // set the learning epoch counter
 4 while $i \le i_{max}$ and termination condition is not satisfied do
     ▷ Learning epoch
 5   foreach $\boldsymbol{s}$ in a random permutation of $S$ do
 6     $\nu^*(\boldsymbol{\nu}^*, \nu^*.\boldsymbol{s}_p) \leftarrow$ determine_winner$(\mathcal{N}, s, \delta_s)$
 7     $\mathcal{N} \leftarrow$ insert_winner$(\mathcal{N}, \nu^*)$
 8     foreach node $\nu \in \mathcal{N}$ in the d-neighborhood of the node $\nu^*$ such that $0 \le d \le 0.2M$ do
 9       Adapt $\nu$ towards $\nu^*.\boldsymbol{s}_p$ using (4) with the neighbouring function (3)
     ▷ Update the best solution found so far
10   Remove all not winning nodes from $\mathcal{N}$
11   $i \leftarrow i + 1$                                         // update the epoch counter
12   $\sigma \leftarrow (1 - i\alpha)\sigma$                             // crease the learning gain
13   Traverse the ring and construct the sequence of visits $\Sigma'$ to $S$ with the corresponding waypoints $P'$ as the $\boldsymbol{s}_p$ points
     associated to the nodes of the ring
14   if $(\Sigma', P')$ represents a shorter route than $(\Sigma, P)$ then
15     $(\Sigma, P) \leftarrow (\Sigma', P')$                                   // update the solution
16 $(\Sigma, P) \leftarrow$ two_opt$(\Sigma, P)$          // improve the found solution using the Two-opt heuristic [61]
17 return $(\Sigma, P)$
```

each learning epoch. The individual parts are described in detail in the following sections.

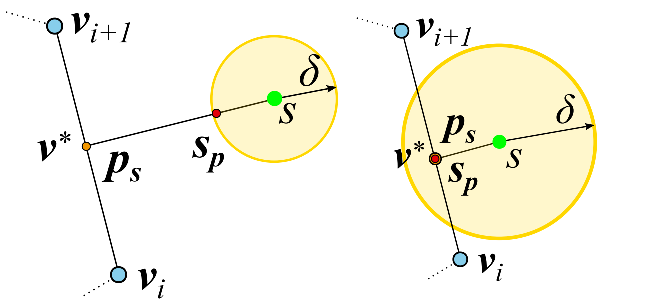
4.2.1. Winner Selection

The winner selection (Lines 5–7, Algorithm 1) determines the closest point $\boldsymbol{p}_s$ of the ring to the location $\boldsymbol{s}$

of the current sensor s together with the expected waypoint location $\boldsymbol{s}_p$ at which s can be covered. The ring of nodes $\mathcal{N}$ can be considered as a sequence of straight line segments defined by two consecutive node locations $\boldsymbol{\nu}_i$ and $\boldsymbol{\nu}_{i+1}$, and therefore, the closest point $\boldsymbol{p}_s$ of the ring to $\boldsymbol{s}$ is determined as the closest point of the segments (ν_i, ν_{i+1}), where $1 \leq i \leq M$ and ν_{M+1} is ν_1, i.e., the array subscripts are closed to modulo M because the ring represents a closed path. Thus, we can iterate over all segments (ν_i, ν_{i+1}) of the ring $\mathcal{N}$, determine the closest point $\boldsymbol{p}'_i$ of each i-th segment to $\boldsymbol{s}$, and select the closest point $\boldsymbol{p}_s$ of the ring to $\boldsymbol{s}$ as the point $\boldsymbol{p}'_i$ with the shortest distance to $\boldsymbol{s}$, i.e.,

$$\begin{aligned} &\boldsymbol{p}_s = \operatorname{argmin}_{1\leq i\leq M} \|(\boldsymbol{p}'_i, \boldsymbol{s})\| \\ &\textit{subject to} \qquad\qquad\qquad\qquad (1) \\ &\boldsymbol{p}'_i \in (\boldsymbol{\nu}_i, \boldsymbol{\nu}_{i+1}) \text{ and } \boldsymbol{\nu}_{M+1} = \boldsymbol{\nu}_1 \ . \end{aligned}$$

Having the point $\boldsymbol{p}_s$, a new winner node ν^* is created and inserted between the respective ν_i and ν_{i+1} nodes. The position of the winner node $\boldsymbol{\nu}^*$ is set to $\boldsymbol{p}_s$, i.e., $\boldsymbol{\nu}^* = \boldsymbol{p}_s$. Besides, a waypoint $\boldsymbol{s}_p$ is determined to cover the sensor s using the sensing range δ, and thus save the travel cost by avoiding precise visitation of the location $\boldsymbol{s}$. A relation of the winner node, waypoint, and sensor are



(a) Winner node outside the δ-neighborhood of s (b) Winner node already inside δ-neighborhood of s

Figure 2: Relation of the closest point $\boldsymbol{p}_s$ of the ring to the sensor s, waypoint $\boldsymbol{s}_p$, and sensor location $\boldsymbol{s}$ in the winner node determination. If $\boldsymbol{s}_p$ is already inside the δ neighborhood of the sensor s, the waypoint $\boldsymbol{s}_p$ is identical to $\boldsymbol{p}_s$, and thus the winner node is set to be already at the desired waypoint and the network is not adapted towards s.

visualized in Fig. 2a. If the winner node is outside the δ-neighborhood of s, its waypoint $\boldsymbol{s}_p$ is determined as the point on the segment $(\boldsymbol{p}_s, \boldsymbol{s})$ that is in $\delta - \epsilon$ distance from $\boldsymbol{s}$

$$\boldsymbol{s}_p = \boldsymbol{s} + (\boldsymbol{p}_s - \boldsymbol{s})\frac{\delta - \epsilon}{\|(\boldsymbol{p}_s, \boldsymbol{s})\|}, \qquad (2)$$

where ϵ is a small constant, e.g., $\epsilon = 0.001$, to ensure the waypoint is inside the δ-neighborhood of s because of eventual numerical imprecisions.[2] On the other hand, if the new winner node is already inside the δ-neighborhood (see Fig. 2b), its waypoint location is set to be identical to the location of the winner node, i.e., $\boldsymbol{s}_p = \boldsymbol{\nu}^*$, which in turn causes the winner node is not effectively adapted towards s, and it stays at its location.

4.2.2. Adaptation

The adaptation (Lines 8–9, Algorithm 1) is a movement of the winner node ν^* (together with its neighbouring nodes) towards its waypoint $\boldsymbol{s}_p$. The adaptation of the winner should be stronger than the adaptation of its neighbors to distribute the ring across the input space and support stabilization of the ring. Therefore, the neighboring function used in SOM adaptation [12] is also employed in the GSOA. The power of adaptation is decreasing with the increase distance of the node from the winner node in the ring and only nodes in the $0.2M$ neighborhood are adapted. The neighbouring function has the form

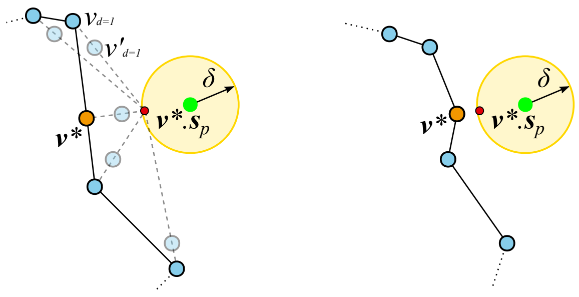
$$f(\sigma, d) = \begin{cases} e^{-\frac{d^2}{\sigma^2}} & \text{for } d < 0.2M \\ 0 & \text{otherwise} \end{cases} , \qquad (3)$$

where M is the current number of nodes in the ring $\mathcal{N}$, σ is the learning gain, and d is the distance of the adapted node ν to the new winner node ν^* in the number of nodes in the ring.

Each node in the $0.2M$ neighborhood of the winner node ν^* is adapted towards $\nu^*.\boldsymbol{s}_p$ which efficiently means a movement of the node ν to a new location $\boldsymbol{\nu}'$ according to

$$\boldsymbol{\nu}' = \boldsymbol{\nu} + \mu f(\sigma, d)(\nu^*.\boldsymbol{s}_p - \boldsymbol{\nu}), \qquad (4)$$

where μ is the learning rate, $f(\sigma, d)$ is the neighbouring function (3), d is the distance of ν from ν^* in the number of nodes, and $\nu^*.\boldsymbol{s}_p$ is the waypoint location associated with the newly added winner node ν^*. The principle of a



$\boldsymbol{\nu}_{d=1}$
$\boldsymbol{\nu}'_{d=1}$
δ
$\boldsymbol{\nu}^*$
$\boldsymbol{\nu}^*.\boldsymbol{s}_p$
δ
$\boldsymbol{\nu}^*$
$\boldsymbol{\nu}^*.\boldsymbol{s}_p$

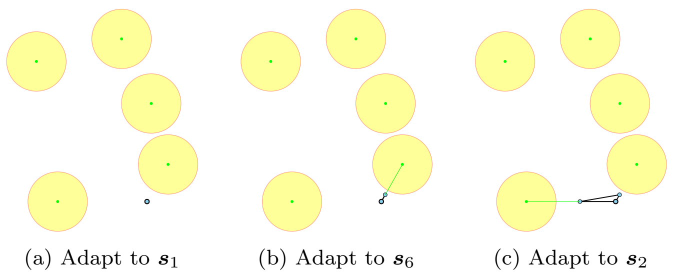
(a) before adaptation (b) after adaptation

Figure 3: Example of adaptation of the winner node ν^* and its neighbouring nodes towards $\boldsymbol{\nu}.\boldsymbol{s}_p$.

single adaptation with the movement of the node locations is visualized in Fig. 3.

The GSOA starts with a single node, and therefore, in the first learning epoch, the first winner node determination together with the adaptation creates from the initial

[2]In the case $\delta \approx \epsilon$, the waypoint location $\boldsymbol{s}_p$ can be set to the sensor location $\boldsymbol{s}$, i.e., $\boldsymbol{s}_p = \boldsymbol{s}$.



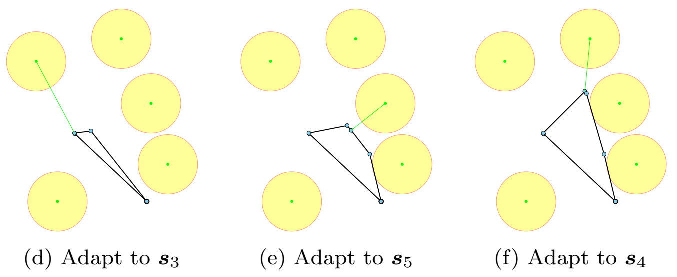


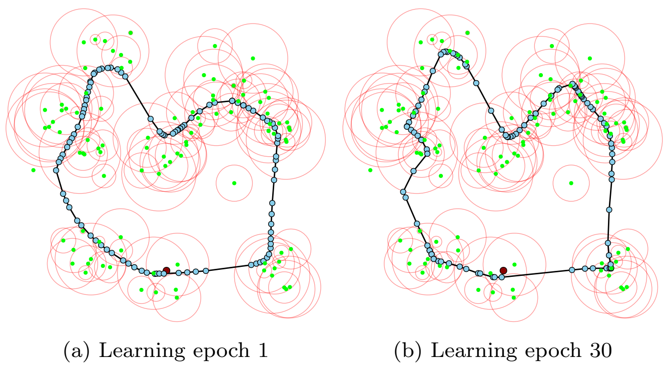
Figure 4: A ring of nodes after the adaptation to the particular sensor location during the first learning epoch of the GSOA. Small blue disks are nodes of the ring that are connected by black line segments. The green straight line segment connects the winner node with its particular sensor location. Notice, some of the nodes may overlap in the visualization because of very similar locations.

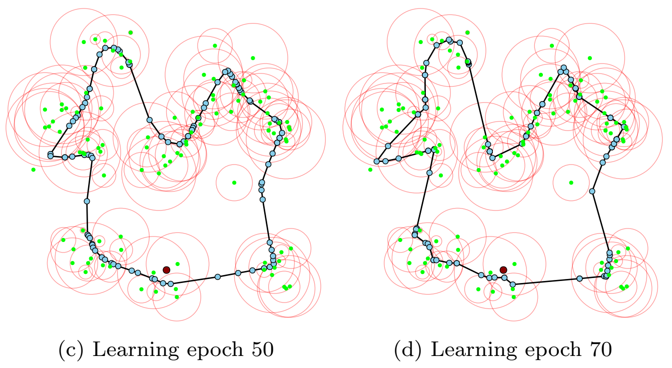
degenerated path a closed ring with two nodes representing the path visiting the first sensor towards which the ring is adapted. After that, the second winner node determination and adaptation add one more node, and so on, see Fig. 4. Thus, the number of nodes is $n + 1$ after the adaptation to all sensors S. Then, all nodes that are not winner nodes in the current epoch are removed, and a solution is extracted from the ring. For the next learning epoch(s), the ring already has n nodes from the winner nodes of the previous epoch, and therefore, the number of nodes will be $2n$ after the next adaptation to all n sensors of S.

4.2.3. Solution Extraction

After the adaptation of the ring towards all sensor locations S, learning parameters are adjusted, and a solution from the ring is extracted (Lines 10–15, Algorithm 1). In each learning epoch, n new winner nodes are added to the ring together with the particularly associated waypoints as a result of the winner node determination for each sensor $s \in S$; thus, we can remove all other nodes from the ring to balance the number of nodes with the number of sensors and keep only the necessary nodes for constructing a feasible solution of the TSP or CETSP. The learning gain is then decreased using the gain decreasing rate α to support the convergence of the GSOA to a stable state.

Since all winner nodes in the ring have associated waypoints, a feasible solution can be constructed by traversing the ring and connecting the waypoints, e.g., see Fig. 1c. Thus, after a single learning epoch, a feasible solution can be determined, and the best solution found so far can be maintained during the learning epochs (similarly to [14]). The learning can be terminated after a finite number of





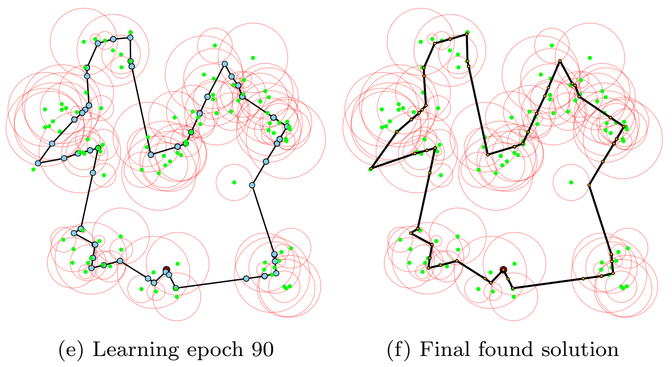


Figure 5: An example of the GSOA evolution in a solution of the CETSP instance team1_100*rdmRad* (from [21]) with individual δ radius at each sensor location visualized as red circles around the sensor locations that are visualized as small green disks. The final solution is found in 102 learning epochs. Notice in the first epochs, the ring itselfs is not a feasible solution, but the associated waypoints to the ring nodes represent a feasible solution.

learning epochs, or whenever the ring is stabilized, and it is not further modified by the adaptation. Also similarly to the Co-adaptive Net [14], the final solution is improved by the Two-opt heuristic [61] to quickly avoid unnecessary crossings of the final route (Line 16, Algorithm 1).

An example of the ring evolution in the solution of the CETSP is depicted in Fig. 5. Notice, the connected ring of nodes do not represent a feasible solution, the connected waypoints form the solution; however, in the final learning epochs, the locations of the nodes fit the waypoints, and the ring converges to the stable state that is a feasible solution.

4.3. Convergence Analysis and Termination Condition

During each learning epoch, the ring of nodes $\mathcal{N}$ is adapted towards all the sensor locations S and a feasible solution is available at the end of each learning epoch (Line 13, Algorithm 1). For simplicity, the number of learning epochs is limited by i_{max}. However, the locations of the nodes in the ring are stabilized in a finite number of learning epochs, which can be shown similarly as in [62] because of $\mu < 1$. Besides, the stability can be intuitively shown as follows.

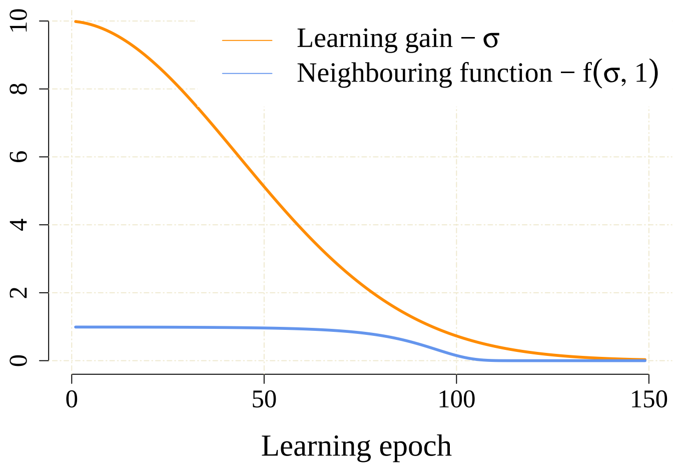


Figure 6: Evolution of the value of the learning gain σ and the neighbouring function (3) for the first neighbor of the winner node ($d = 1$) in particular learning epochs. The initial value of the learning gain σ is set to $\sigma = 10$ and the gain decreasing rate α is $\alpha = 0.0005$.

The learning rate σ is decreased after each learning epoch (Line 12, Algorithm 1), and thus for a small value of σ, the value of the neighbouring function (3) is pragmatically zero for $d > 0$. For example in the case of the recommended initial value of $\sigma = 10$ and $\alpha = 0.005$, the value of the neighbouring function (3) is about 1.41e-304 at the learning epoch 146 and it is zero in the next epoch (using IEEE 754 data representation). The values of σ and $f(\sigma, d)$ at learning epochs are shown in Fig. 6. Besides, it is assured the learning gain σ is never negative, which can be made by i_{max} (see Line 2, Algorithm 1) or by explicit setting $\sigma = 0$ when it would be negative, e.g., at the Line 12, Algorithm 1. Hence, only the winner nodes are adapted towards the waypoints since that.

A winner node ν^* is always moved towards the corresponding $\boldsymbol{s}_p$ about $\mu \|(\boldsymbol{\nu}^*, \boldsymbol{s}_p)\|$ distance.[3] Since such a node is then never moved towards a different $\boldsymbol{s}_p$ because (3) is zero for $d > 0$, it will be the closest point to the corresponding sensor location. In a case the ring is eventually moved closer to that location as a result of the adaptation of another winner node, a new node is always created in the winner selection of the GSOA, and such a node is moved towards the particular sensor location. Therefore, the ring is always moved towards the waypoints, and from a certain learning epoch, the distance of the winner nodes to the waypoints is only decreasing because the respective winner nodes are not moved in the adaptation of other nodes.

Based on the convergence analysis of the GSOA, the adaptation can also be terminated whenever the winner nodes are negligibly close to the corresponding waypoints (or the sensor locations themselves in the case of the TSP), which can be an additional termination condition (Line 4, Algorithm 1). Moreover, for a solution of the CETSP, the adaptation of the nodes that are inside the corresponding δ-neighborhood can be suppressed, and the adaptation can be terminated once all the winner nodes are inside the δ neighborhood of the sensor locations, which can further decrease the required number of learning epochs.

Regarding the results on the empirical evaluation reported in Section 5, the GSOA converges in about one hundred learning epochs and the recommended value of i_{max} is $i_{max} = 150$, which also corresponds to the number of learning epochs from which the neighbouring function (3) is zero for the neighbouring nodes of the winner node (for $\alpha = 0.0005$ and the initial value $\sigma = 10$).

Finally, notice that a solution is not determined using the node locations but using the associated waypoints. Therefore the solution constructed at the end of each learning epoch (Line 13, Algorithm 1) is always feasible. Hence, there is not an issue with the convergence of the GSOA, and the proposed learning procedure has the anytime property.

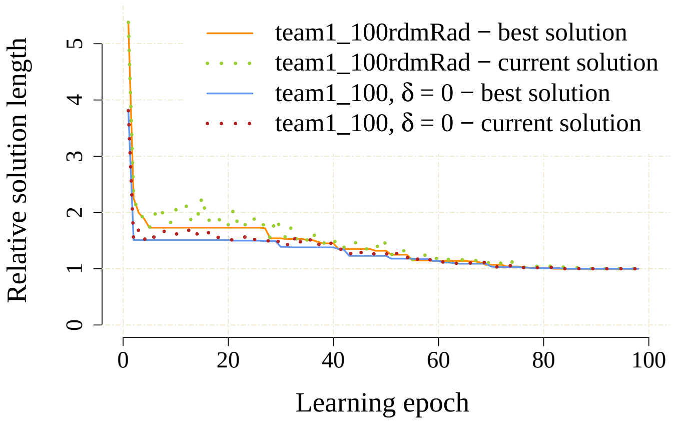


Figure 7: Evolution of the solution length in the GSOA solution of the CETSP (*team1_100rdmRad*) and TSP (*team1_100*, $\delta = 0$). The solution length is shown as the length of the best solution found so far and the current solution represented by the ring at the particular learning epoch, both lengths are shown as the relative ratio to the length of the final found solution.

An example of the evolution of the solution length and its convergence during the GSOA unsupervised learning is depicted in Fig. 7 for the CETSP instance *team1_100rdmRad* and its variant with the zero neighborhood, i.e., an instance of the TSP with the same sensor locations. The corresponding final solutions are visualized in Fig. 8. It can be observed that in both cases, the GSOA converges in less than one hundred learning epochs. The quality

[3]For the winner node adaptation, the neighbouring function (3) is always greater than zero even for $\sigma = 0$ and $d = 0$, i.e., for both possible values of the exponent 0 and -1.

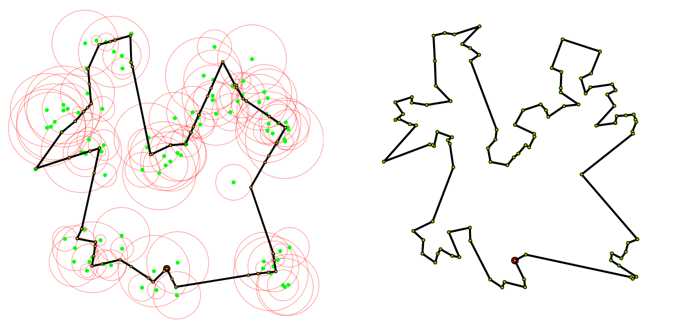

(a) CETSP – *team1_100rdmRad*, $L = 402.6$, GSOA converges in 99 epochs (b) TSP – *team1_100* with $\delta = 0$, $L = 641.7$, GSOA converges in 97 epochs

Figure 8: An example of the found solutions of the CETSP instance *team1_100rdmRad* [21] and TSP instance created from the same problem but with the zero neighborhood ($\delta = 0$).

of the solution represented by the current ring may oscillate because of the randomized nature of the unsupervised learning; however, maintaining the best solution found so far provides anytime property with improving solution until the GSOA is in the stable state.

4.4. Computational Complexity of the GSOA

The computational complexity of a single learning epoch of the GSOA can be bounded according to the performed number of winner selections and adaptations. For each sensor location, a new winner node is searched from the ring with no more than $2n$ nodes. Each winner node is then adapted together with its neighboring nodes that are within the distance of $0.2M$ nodes from the winner node that can be bounded by n nodes. Hence the complexity of the two nested *foreach* loops of Algorithm 1 can be bounded by $O(n^2)$.

At the end of each learning epoch, not winning nodes are removed which requires traversing the ring in $2n$ steps and the solution is constructed by traversing the ring with n nodes. Therefore, the complexity of the learning epoch can be bounded by $O(n^2)$.

The complexity of the whole GSOA learning depends on the number of learning epochs that is a constant and complexity of the Two-opt heuristic [61] that can be bounded by $O(n^2)$. Therefore, the computational complexity of the proposed GSOA can be bounded by $O(n^2)$. The real required computational times are reported in Section 5.

4.5. GSOA in Problems with Sensors Selection

The herein proposed GSOA is based on the previous successful deployments of the unsupervised learning to other routing problems such as the Prize-Collecting TSP (PC-TSP) [29] or Orienteering problem (OP) [33], where the solution of the problems includes a selection of the most suitable (i.e., most rewarding) sensors. Although a detail description of the GSOA deployment in these problems is out of the scope of this paper, the main idea is presented here to emphasize the most important parts of the GSOA that allow solving these problems.

The selection of the sensors during the unsupervised learning needs to trade-off between the adding rewards in the OP (or reducing the penalty in the PC-TSPN) and the solution cost regarding the length of the final path. Therefore, it is not desirable to adapt the ring to all sensors, but only to the most suitable sensors. This can be realized by the conditional adaptation proposed in [29] where the winner node is firstly selected, but it is inserted into the ring only if certain criteria are met.

For example in the case of the OP, it makes sense to insert a new winner node to the ring only if the route represented by the ring would satisfy the limited travel budget after the adaptation. Hence, it is also required to evaluate the final tour length during the learning which is one of the benefits of the GSOA because the solution can be directly determined using the sequence of nodes in the ring and the associated waypoints that are determined during the winner node selection.

Besides, the principles of the GSOA immediately allows solving the OP with the disk-shaped neighborhoods because the determination of the waypoint locations is made during the winner node selection. Examples of solutions to these problems by the unsupervised learning procedures that inspire and motivate the herein proposed GSOA can be found in the previous work, e.g., [29, 32, 33].

5. Results

The performance of the proposed GSOA has been empirically evaluated in a series of TSP and CETSP instances. Regarding the TSP, the same Euclidean TSP instances from the TSPLIB [37] with up to 2392 locations as in [16] are used to compare the performance of the GSOA with the selected best performing SOM-based TSP solvers reported in the literature. Therefore the Co-adaptive Net [14] and ORC-SOM [16] have been implemented within the same C++ framework as the proposed GSOA to make a fair comparison. In addition, probably the best performing combinatorial heuristic LKH [2] is employed to show a gap between the combinatorial heuristics of the operational research with unsupervised learning based approaches.

The performance of the GSOA in the CETSP is compared using instances proposed in [21] and the results and heuristic SZ2 proposed therein. Besides, the CETSP instances are solved as a variant of the GTSP where each disk-shaped neighborhood is sampled into 24 locations at the perimeter of the particular δ-neighborhood as recommended in [21]. These GTSP instances are solved using the GLNS [24] heuristic. The GLNS implementation in the Julia programming language is made available by the authors of [24] and it has been used within the same computational environment as the implemented unsupervised learning based solvers.

The used performance indicators are the quality of the found solutions and the required computational time.

Table 1: GSOA and SOM-based Solvers in the TSPLIB instances of the TSP

Set	L_{opt}	**LKH** [2]			**Co-adaptive Net** [14]			**ORC-SOM** [16]			**GSOA**		
		%PDB	%PDM	T_{CPU}	%PDB	%PDM	T_{CPU}	%PDB	%PDM	T_{CPU}	%PDB	%PDM	T_{CPU}
eil51	426	0.00	0.00	0.012	**2.63**	**3.60**	0.007	3.88	4.32	0.072	2.94	4.11	0.003
st70	675	0.00	0.00	0.016	1.94	2.88	0.009	1.18	2.34	0.129	**1.13**	**2.20**	0.005
eil76	538	0.00	0.00	0.014	**3.72**	5.80	0.009	4.47	5.09	0.148	3.86	**4.80**	0.005
rd100	7 910	0.00	0.00	0.021	**0.65**	3.41	0.016	2.33	**2.85**	0.249	0.76	3.53	0.009
lin105	14 379	0.00	0.00	0.016	7.22	15.71	0.019	0.25	0.39	0.280	**0.03**	**0.32**	0.009
pr107	44 303	0.00	0.00	0.021	0.48	1.13	0.017	0.17	**0.30**	0.314	**0.05**	0.72	0.010
pr136	96 772	0.00	0.00	0.035	**1.71**	**2.93**	0.027	3.54	4.18	0.479	3.36	3.91	0.015
pr152	73 682	0.00	0.10	0.027	1.36	4.28	0.042	1.23	**1.71**	0.587	**0.82**	**1.71**	0.017
rat195	2 323	0.00	0.07	0.039	27.25	30.72	0.056	8.42	10.26	0.960	**6.36**	**7.72**	0.029
kroA200	28 568	2.80	2.81	0.058	12.23	14.43	0.068	6.44	6.85	1.010	**3.63**	**6.02**	0.032
pcb442	50 778	0.01	0.12	0.131	5.64	8.44	0.240	7.03	8.78	4.785	**4.38**	**7.04**	0.152
pr1002	259 045	0.00	0.17	0.897	7.95	9.81	0.757	4.91	5.81	24.655	**4.48**	**5.15**	0.800
pcb1173	56 892	0.04	0.30	0.904	7.53	9.50	0.935	9.33	10.28	34.811	**6.84**	**8.04**	1.151
d1655	62 128	0.01	0.25	1.854	10.15	11.72	1.492	10.21	10.82	72.445	**7.88**	**9.34**	2.660
vm1748	336 556	0.08	0.18	2.148	9.34	11.02	1.662	7.25	8.18	78.568	**5.59**	**7.01**	3.041
pr2392	378 032	0.05	0.21	3.826	7.80	8.92	2.594	6.97	7.54	148.056	**5.75**	**6.82**	6.956

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

Since the evaluated algorithms are randomized, the quality is measured as the percentage deviation from the optimum of the best solution value over the performed trials [17] further denoted %PDB.

$$\%\text{PDB} = \frac{L - L_{ref}}{L_{ref}} \cdot 100\%, \tag{5}$$

where L is the length of the best found solution among the trials and L_{ref} is the optimal or the reference solution length of the particular problem instance.

The robustness of the algorithm over the trials is measured as the percentage deviation from the optimum of the mean solution value over the performed trials [17] denoted %PDM.

$$\%\text{PDM} = \frac{L_{avg} - L_{ref}}{L_{ref}} \cdot 100\%, \tag{6}$$

where L_{avg} is the average solution length among the trials.

Optimal solutions of the evaluated TSP instances are known or can be found by the Concorde solver [63] or LKH [64]. On the other hand, an optimal solver for the CETSP is not available, and therefore, the best-found solutions of the evaluated problems reported in [21] are used for L_{ref} in (5) and (6) in cases where the evaluated algorithms do not provide better solution.

The computational requirements are reported as the real required computational times using a single core of the Intel Core i7-6700K processor running at 4 GHz. The proposed GSOA, Co-adaptive Net, and ORC-SOM are implemented in C++ and compiled with the clang compiler ver. 5.0, which has been also used to compile the LKH [64]. Therefore, the computational requirements are presented as average values of the required computational time T_{CPU} per trial (reported in seconds) and all the reported times can be thus directly compared. The only exception are the computational times of the CETSP heuristic SZ2 that are adapted from [21], where therein reported computational requirements are for a Pentium processor running at 2.4 GHz, without further information about the type. It is expected it is the Intel Pentium E2220 processor, which has been introduced at the beginning of 2008. Considering single thread performance [65], the processor is about 2.63× slower than the utilized Intel Core i7-67000K CPU. Hence, the times reported in [21] are divided by the factor of 2.63 to make the required computational times of the SZ2 heuristic reported in [21] comparable to the used computational environment. Such computational times are denoted as ${T_{CPU}}^*$.

The individual results in the TSP and CETSP instances are reported in the following sections.

5.1. Evaluation of the GSOA in the TSP Benchmarks

The performance indicators of the Co-adaptive Net [14], ORC-SOM [16] (both with the settings recommended by the authors), the proposed GSOA (with the settings as in Section 4), and the LKH (with the fastest possible settings) for the selected TSP instances are depicted in Table 1. The solved TSPLIB instances cover different types of the TSP instances where unsupervised learning based solvers perform well but also poorly, i.e., instances with clusters of locations such as *pr152*, *pcb442*, *d1655*, *vm1748*, *pr2396*, etc. All the problems can be solved optimally, and the length of the optimal solution is in the second column denoted L_{opt}. The best solutions found by the evaluated unsupervised learning based solvers are highlighted in bold.

The results of the LKH are included for completeness, as it is not surprising it outperforms all the unsupervised learning based approaches, which is known and discusesed in almost any comparision of the unsupervised learning approaches with combinatorial heuristics of the operational research, e.g., mentioned and advocated in [14, 54]. Notice, the LKH provides optimal solutions in less than one second for most of the problems with less than thousand of nodes. The most demanding is the optimal solution of *pr2392* which requires about 25 seconds using the evaluation computational environment. However, for the presented results in Table 1, the LKH has been run with the computational time limited to 0.01 seconds, but the solution is terminated after the initialization, and therefore, the reported times in the $\mathrm{T_{CPU}}$ column are higher than this limit. Thus, in few cases, the LKH is more demanding than unsupervised learning approaches, but it always provides significantly better solution. Therefore, in the rest of this section, only the unsupervised learning based approaches are discusesed to emphasize improvements provided by the GSOA for this type of solvers.

The parameters of the GSOA that can be tuned are the initial value of the learning gain σ, the learning rate μ, the gain decreasing rate α, and the maximal number of learning epochs i_{max}. Based on the experimental evaluation, the changes in the learning parameters σ, μ, and α influence the solution only slightly. Therefore, the GSOA can be considered as parameterless. The computational requirements can be simply decreased by reducing the number of learning epochs i_{max} as a solution is provided after each learning epoch; however, worse solutions can be expected, as it is indicated in Fig. 7.

The proposed GSOA seems to provide a suitable tradeoff between the quality of the found solutions, computational requirements, and also the complexity of the implementation. The GSOA benefits from a lower number of nodes than the ORC-SOM and also from the growing structure. The reported computational complexity of the ORC-SOM is $O(n^3)$ while the GSOA is $O(n^2)$. On the other hand, the Co-adaptive Net is faster than the GSOA in solving large instances with thousands of locations. It is because the Co-adaptive Net uses the restricted search for winner selection and the winner is selected from the whole ring only every $\beta = 10$ learning epochs [14]. A similar searching strategy can also be employed in the GSOA at the cost of a worse solution, which can be observed for the Co-adaptive Net.

In comparision of implementations, the GSOA is much simpler than the rather complex Co-adaptive Net with many parameters that can be tuned or the ORC-SOM with specific more complex adaptation formulas. However, there is not a significant reason to solve instances of the regular TSP by the unsupervised learning approaches as the solution quality is significantly worse than a solution provided by the available LKH solver [64] with competitive computational requirements. Therefore the main benefit of the proposed GSOA is in solving the combinatorial optimization problems that also includes continuous optimization part, where the unsupervised learning method can take advantage of the online sampling of the most suitable waypoints, which is reported for solving the CETSP in the next section.

5.2. Evaluation of the GSOA in the CETSP Benchmarks

Based on the literature review, the evaluation of the proposed GSOA is performed using benchmarks defined in [21]. The performance of the proposed unsupervised learning is compared with the SZ2 heuristic [21] and GTSP-based approach using 24 samples per each disk that is solved by the GLNS heuristic [24]. A solution of the GTSP can be demanding, and therefore, the GLNS is used in the fast mode and the computational time is limited to 300 seconds, which has been reached in few cases that are indicated by the emphasized values in the reported tables. The solutions found by the GTSP-based approach are denoted GTSP-GLNS.

The found solutions are compared with the reference values provided in [21] (as the best-found solutions therein); however, new best-known solutions are found for several CETSP benchmarks by the GTSP-GLNS and in several cases also by the proposed GSOA. Therefore, the overall best-known solutions among the reported values in [21] (further denoted as L'_{ref}) and the found solutions by the GSOA and GTSP-GLNS approaches are considered as the reference solutions for the computation of the %PDB and %PDM. Thus, the zero value of the %PDB indicates the best solution is found by the particular approach. The cost of the best-known solutions is reported in the column denoted L_{ref} and better solutions than L'_{ref} are highlighted in bold.

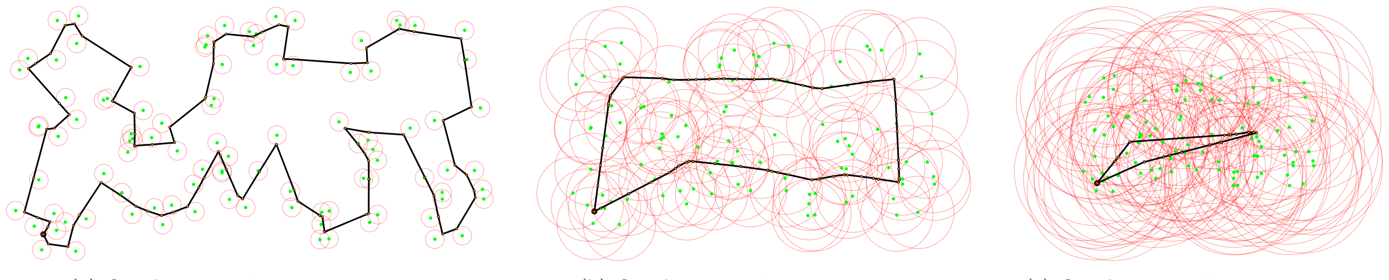
The performance comparison of the proposed GSOA in CETSP instances is organized in three sets of benchmarks according to [21]. The first set of benchmarks is focused on the influence of the δ radius of the disk-shaped neighborhood, and the results are reported in Table 2, where seven instances are considered with three different overlap ratios R. For a particular instance of the CETSP, all the locations have the same δ (except the initial/terminal location with $\delta = 0$) defined by R [21]. The second set of CETSP benchmarks is reported in Table 3 and also in these instances, the radii of all locations to be visited are the same. Finally, individual values of the δ radius per each location are considered in the third set for which the reported results are in Table 4.

The results indicate that the proposed GSOA founds competitive solutions to the SZ2 heuristic [21] and GTSP-GLNS [24] in a fraction of second and for each solved

Table 2: Results of the CETSP solvers in instances with different overlap ratio R

Set	# of nodes	L_{ref}	L'_{ref} [21]	**SZ2** [21]		**GTSP-GLNS** [24]			**GSOA**		
				%PDB	T$\mathrm{T_{CPU}}$*	%PDB	%PDM	$\mathrm{T_{CPU}}$	%PDB	%PDM	$\mathrm{T_{CPU}}$
						Very low **overlap ratio** $R = 0.02$					
kroD100	100	159.0	159.0	2.96	0.113	2.91	**3.01**	11.4	**2.34**	3.68	0.010
rat195	195	158.8	158.8	7.87	0.205	**3.53**	**4.43**	49.7	5.24	7.19	0.033
lin318	318	2 863.4	2 863.4	1.78	0.562	**0.23**	**0.94**	162.4	1.65	3.27	0.082
rd400	400	1 033.4	1 033.4	3.17	0.692	**0.59**	**2.46**	214.2	2.46	4.31	0.135
pcb442	442	323.0	323.0	**3.16**	0.549	4.68	**5.98**	282.1	5.77	8.01	0.154
d493	493	**202.5**	202.8	5.29	0.959	**0.00**	**1.08**	*300.0*	2.37	5.31	0.170
dsj1000	1 000	**879.1**	935.7	8.03	3.496	5.34	8.17	*300.0*	**0.00**	**1.49**	0.624
						Moderate **overlap ratio** $R = 0.10$					
kroD100	100	89.7	89.7	7.82	0.010	10.50	10.58	13.1	**0.25**	**0.52**	0.005
rat195	195	68.1	68.1	8.93	0.172	16.58	16.79	50.0	**0.31**	**0.58**	0.011
lin318	318	1 408.5	1 408.5	10.53	0.327	24.79	27.23	146.8	**0.62**	**1.78**	0.053
rd400	400	**461.4**	466.1	11.97	0.303	19.98	22.40	243.8	**0.00**	**2.78**	0.064
pcb442	442	147.2	147.2	10.20	0.606	22.37	23.59	285.0	**2.18**	**4.10**	0.061
d493	493	101.7	101.7	9.01	0.678	40.66	41.90	*300.0*	**2.03**	**4.49**	0.065
dsj1000	1 000	376.1	376.1	7.58	1.771	94.38	100.85	*300.0*	**0.82**	**4.80**	0.263
						Very high **overlap ratio** $R = 0.30$					
kroD100	100	58.5	58.5	**0.00**	0.051	60.18	62.06	15.7	0.54	**1.29**	0.003
rat195	195	45.7	45.7	**0.20**	0.128	95.22	100.67	62.2	0.21	**0.62**	0.010
lin318	318	766.0	766.0	2.15	0.226	125.69	129.08	159.6	**0.46**	**2.21**	0.036
rd400	400	224.8	224.8	3.76	0.333	141.93	145.26	257.7	**1.16**	**6.17**	0.040
pcb442	442	83.5	83.5	**1.89**	0.273	115.19	125.77	296.6	2.03	**4.47**	0.013
d493	493	69.8	69.8	1.42	0.259	112.15	125.49	*300.0*	**0.72**	**2.20**	0.012
dsj1000	1 000	**194.4**	200.0	0.99	0.989	320.12	331.65	*300.0*	**0.00**	**4.71**	0.245

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



(a) Overlap ratio $R = 0.02$, L=162.8 (b) Overlap ratio $R = 0.10$, L=89.9 (c) Overlap ratio $R = 0.30$, L=58.9

Figure 9: Selected best solutions found by the proposed GSOA. CETSP instances of the *kroD100* problem with a different overlap ratio R [21].

CETSP instance, a solution is found in less than one second. The GSOA provides the best performance in instances with the moderate and very high overlap ratio ($R = 0.3$), see Table 2, and in instances with various radii per each location, see Table 4. In these instances, the Steiner zones are becoming large for the increasing overlap ratio (see Fig. 9) and the finite sampling does not provide enough options to find short tours. Thus, the employed

Table 3: Results of the CETSP solvers in intances with constant δ radii

Set	# of nodes	L_{ref}	L'_{ref} [21]	**SZ2** [21]		**GTSP-GLNS** [24]			**GSOA**		
				%PDB	T_{CPU}*	%PDB	%PDM	T_{CPU}	%PDB	%PDM	T_{CPU}
team1_100	101	307.3	307.3	6.31	0.119	9.64	10.07	2.1	**0.83**	**1.47**	0.012
team2_200	201	246.7	246.7	12.54	0.241	56.02	56.66	14.5	**0.58**	**1.02**	0.013
team3_300	301	466.2	466.2	8.45	0.422	13.96	15.37	32.3	**5.41**	**9.91**	0.054
team4_400	401	**678.6**	680.2	9.21	0.544	6.18	7.40	59.3	**0.00**	**2.54**	0.105
team5_499	500	702.8	702.8	3.52	0.995	4.48	**5.72**	72.3	**2.63**	6.61	0.170
team6_500	501	225.2	225.2	3.43	0.413	140.95	147.15	136.5	**1.20**	**7.60**	0.032
bonus1000	1 001	**400.9**	402.5	9.22	1.845	81.49	86.38	*300.0*	**0.00**	**4.27**	0.457
rotatingDiamonds1	21	32.4	32.4	2.35	0.018	15.84	15.84	0.5	**1.35**	**2.22**	0.001
rotatingDiamonds2	61	140.5	140.5	1.26	0.054	1.78	2.63	0.7	**0.62**	**1.71**	0.004
rotatingDiamonds3	181	380.9	380.9	**0.65**	0.190	2.37	**2.39**	6.4	0.99	3.63	0.023
rotatingDiamonds4	321	770.7	770.7	**0.54**	0.383	0.74	**1.74**	21.9	2.95	8.97	0.073
rotatingDiamonds5	681	1 510.8	1 510.8	**0.31**	1.022	2.24	**3.71**	151.4	2.13	11.19	0.265
bubbles1	37	349.1	349.1	0.70	0.033	7.48	7.48	0.6	**0.26**	**0.69**	0.002
bubbles2	77	428.3	428.3	0.24	1.010	6.73	6.75	1.1	**0.18**	**0.69**	0.005
bubbles3	127	530.7	530.7	12.21	0.407	5.08	5.08	2.6	**0.17**	**0.38**	0.010
bubbles4	185	**812.1**	829.9	4.05	0.294	12.41	14.23	7.1	**0.00**	**0.62**	0.022
bubbles5	251	1 062.3	1 062.3	7.49	1.568	**1.34**	**1.37**	17.4	2.07	2.84	0.040
bubbles6	325	**1 264.9**	1 383.1	5.87	0.517	**0.00**	**0.06**	36.6	6.81	8.11	0.069
bubbles7	407	**1 545.3**	1 720.2	6.96	0.850	**0.00**	**1.87**	49.4	1.19	2.69	0.108
bubbles8	497	**1 670.9**	2 101.4	7.10	1.117	**0.00**	**1.64**	85.0	3.95	5.08	0.153
bubbles9	595	**1 788.6**	2 426.3	8.24	2.178	**0.00**	**2.27**	158.0	5.33	7.60	0.215
concentricCircles1	17	53.2	53.2	3.30	0.021	4.00	4.00	0.5	**1.68**	**2.54**	0.001
concentricCircles2	37	153.1	153.1	5.73	0.044	**0.02**	**0.05**	0.6	2.37	4.04	0.002
concentricCircles3	61	**268.5**	271.1	1.40	0.048	**0.00**	**0.27**	0.8	2.82	3.63	0.005
concentricCircles4	105	**449.1**	454.5	5.61	0.104	**0.00**	**0.05**	2.0	2.77	3.40	0.012
concentricCircles5	149	**631.3**	645.4	3.31	0.273	**0.00**	**0.67**	4.2	5.46	6.41	0.022
chaoSingleDep	201	1 022.9	1 022.9	**0.01**	0.158	4.79	**4.97**	9.5	3.80	7.61	0.028

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

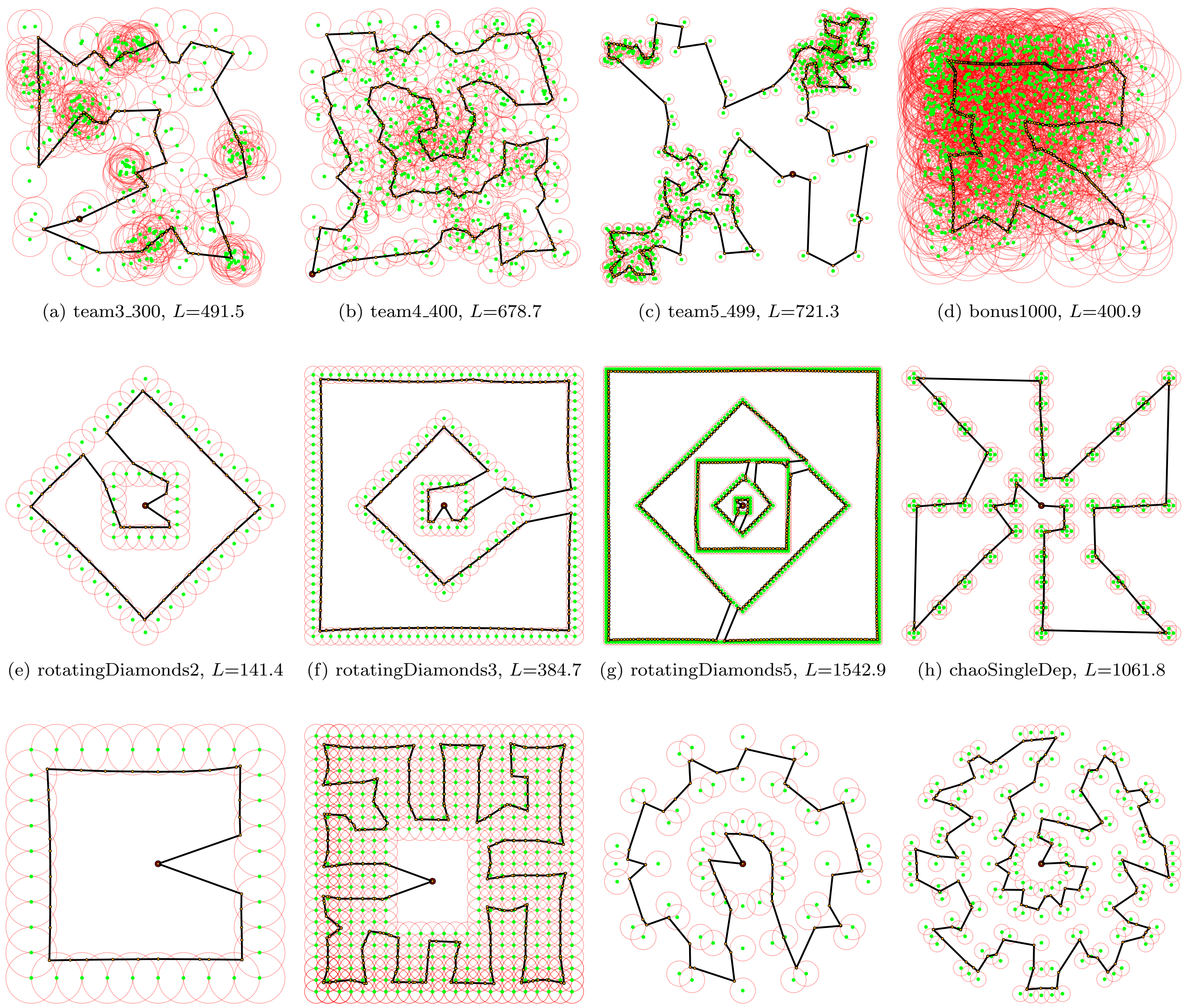
GLNS heuristic provides worse results in these problems, which is caused by the poor explicit sampling of the neighborhood into 24 possible waypoints. Moreover, the solution of the related GTSP instances is demanding, and the GLNS heuristic does not provide solution of suficient quality in the fast mode as the computational limit is reached in few cases. The author of [21] reported that the best solution can be found by the GTSP-based approach, which is mostly the approach by which reference solutions with L'_{ref} are found, at the cost of very high computational times.

Selected best solutions found by the proposed GSOA are visualized in Fig. 9 and in Fig. 10.

5.3. Discussion

It is known that SOM-based approaches usually do not compete well with the best performing combinatorial heuristics for the TSP, which is reported here in Table 1 for the results of the LKH [2]. On the other hand, the main benefit of the GSOA in CETSP instances is in the online-sampling of the neighborhood during the unsupervised learning. The GSOA seems to be a suitable solver for the CETSP in cases where the computational time is limited; however, the solution can be probably improved by a combination with memetic techniques as in [15], at the cost of the increased computational burden.

Since the computational requirements of the GSOA are very low, it can be employed as a construction heuristic providing an initial solution that can be further improved


(a) team3_300, L=491.5 (b) team4_400, L=678.7 (c) team5_499, L=721.3 (d) bonus1000, L=400.9

(e) rotatingDiamonds2, L=141.4 (f) rotatingDiamonds3, L=384.7 (g) rotatingDiamonds5, L=1542.9 (h) chaoSingleDep, L=1061.8

(i) bubbles1, L=350.0 (j) bubbles9, L=1883.8 (k) concentricCircles3, L=276.1 (l) concentricCircles5, L=665.7

Figure 10: Selected best solutions of the CETSP instances with constant radius [21] found by the proposed GSOA.

Table 4: Results of the CETSP solvers in intances with arbitrary δ radii

Set	# of nodes	L_{ref}	L'_{ref} [21]	**GTSP-GLNS** [24] %PDB	%PDM	T_{CPU}	**GSOA** %PDB	%PDM	T_{CPU}
team1_100*rdmRad*	101	388.54	388.54	6.25	6.25	2.3	**3.29**	**3.52**	0.009
team2_200*rdmRad*	201	**622.11**	622.74	7.13	7.49	10.1	**0.00**	**1.98**	0.030
team3_300*rdmRad*	301	381.83	381.83	39.31	40.88	28.9	**1.16**	**6.84**	0.048
team4_400*rdmRad*	401	1 011.77	1 011.77	**1.10**	**2.81**	56.3	1.33	3.24	0.165
team5_499*rdmRad*	500	454.33	454.33	38.63	40.79	99.8	**0.75**	**4.87**	0.127
team6_500*rdmRad*	501	**640.09**	666.15	14.84	16.04	99.0	**0.00**	**2.49**	0.147
bonus1000*rdmRad*	1 001	**965.38**	987.11	11.97	13.27	*300.0*	**0.00**	**2.04**	0.610
kroD100*rdmRad*	100	141.83	141.83	5.94	6.09	2.1	**3.37**	**4.12**	0.009
rat195*rdmRad*	195	68.22	68.22	85.97	88.87	11.5	**0.50**	**2.24**	0.013
lin318*rdmRad*	318	2 080.57	2 080.57	16.75	19.37	33.3	**0.01**	**4.09**	0.062
rd400*rdmRad*	400	1 252.38	1 252.38	**1.76**	**2.66**	52.2	3.33	4.80	0.195
pcb442*rdmRad*	442	**224.36**	235.19	14.15	14.99	69.1	**0.00**	**3.17**	0.120
d493*rdmRad*	493	**138.97**	140.12	33.69	34.65	92.6	**0.00**	**1.78**	0.129
dsj1000*rdmRad*	1 000	**627.80**	653.13	43.83	45.95	*300.0*	**0.00**	**3.75**	0.569

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

by another optimization metaheuristics such as the Variable Neighborhood Search (VNS) [66], which is one of the subjects for future work. Besides, the great flexibility of the GSOA arising from its simplicity allows to employ it in another routing problems such as the orienteering problems or even problems where the length of the arc connecting two waypoints is not the Euclidean distance [36].

6. Conclusion

In this paper, a new unsupervised learning based approach called the *Growing Self-Organizing Array* (GSOA) is proposed to solve routing problems with underlying TSP-like solution. The main ideas of the unsupervised learning employed in the GSOA originate from several successful deployments of unsupervised learning based solvers in various routing problems. The ideas are consolidated in a relatively simple to implement unsupervised learning procedure of the GSAO. Although SOM-based approaches for the TSP have been studied for almost more than three decades, the application of unsupervised learning in a solution of the TSPN and more specifically in its herein addressed variant CETSP is mostly unattended. The unsupervised learning based solvers are usually not very competitive with the most powerful combinatorial heuristics for routing problems. However, the presented results show that by conceptually simple determination of the waypoints during the learning, the proposed GSOA can provide competitive solutions in the CETSP. Moreover, the computational requirements are significantly lower than the approach based on the computationally demanding solution of the purely combinatorial GTSP.

The proposed GSOA exploits the benefit of the unsupervised learning in the way that the suitable waypoints are sampled during the solution of the sequencing part of the CETSP, and thus probably a poor performance of the unsupervised learning based solution of the combinatorial optimization part of the underlying TSP is overcome by the selection of suitable waypoints. This is the main advantage of the proposed GSOA over the purely combinatorial solution based on the GTSP. Together with the low computational requirements, the presented results motivate to further study a combination of the GSOA with other metaheuristics such as Memetic SOM or VNS. Besides, it also motivates to investigate properties of the GSOA in additional challenging routing problems in 3D space or with a selection of the sensor locations to be visited in orienteering problems.

Acknowledgement

The presented work has been supported by the Czech Science Foundation (GAČR) under research project 15-09600Y.

7. References

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

[2] K. Helsgaun, An Effective Implementation of the Lin-Kernighan Traveling Salesman Heuristic, European Journal of Operational Research 126 (1) (2000) 106–130.

[3] J.-Y. Potvin, Genetic algorithms for the traveling salesman problem, Annals of Operations Research 63 (3) (1996) 337–370.

[4] M. Dorigo, V. Maniezzo, A. Colorni, Ant system: optimization by a colony of cooperating agents, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 26 (1) (1996) 29–41.

[5] S. Kirkpatrick, C. D. Gelatt, M. P. Vecchi, Optimization by simulated annealing, Science 220 (4598) (1983) 671–680. doi:10.1126/science.220.4598.671.

[6] J. J. Hopfield, D. W. Tank, "Neural" computation of decisions in optimization problems, Biological Cybernetics 52 (3) (1985) 141–152.

[7] Q. Wang, X. Sun, B. L. Golden, J. Jia, Using artificial neural networks to solve the orienteering problem, Annals of Operations Research 61 (1) (1995) 111–120.

[8] R. Durbin, D. Willshaw, An analogue approach to the travelling salesman problem using an elastic net method, Nature 326 (16) (1987) 689–691.

[9] B. Angéniol, G. de la C. Vaubois, J.-Y. L. Texier., Self-organizing feature maps and the travelling salesman problem, Neural Networks 1 (1988) 289–293.

[10] J. C. Fort, Solving a combinatorial problem via self-organizing process: An application of the Kohonen algorithm to the traveling salesman problem, Biological Cybernetics 59 (1) (1988) 33–40.

[11] K. Smith, An argument for abandoning the travelling salesman problem as a neural-network benchmark, IEEE Transactions on Neural Networks 7 (6) (1996) 1542–1544.

[12] T. Kohonen, M. R. Schroeder, T. S. Huang (Eds.), Self-Organizing Maps, 3rd Edition, Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2001.

[13] K. Smith, M. Palaniswami, M. Krishnamoorthy, Neural techniques for combinatorial optimization with applications, IEEE Transactions on Neural Networks 9 (6) (1998) 1301–1318.

[14] E. M. Cochrane, J. E. Beasley, The co-adaptive neural network approach to the Euclidean travelling salesman problem, Neural Networks 16 (10) (2003) 1499–1525.

[15] J. Créput, A. Koukam, A memetic neural network for the euclidean traveling salesman problem, Neurocomputing 72 (4-6) (2009) 1250–1264.

[16] J. Zhang, X. Feng, B. Zhou, D. Ren, An overall-regional competitive self-organizing map neural network for the euclidean traveling salesman problem, Neurocomputing 89 (0) (2012) 1–11.

[17] H. Wang, N. Zhang, J.-C. Crput, A massively parallel neural network approach to large-scale euclidean traveling salesman problems, Neurocomputing 240 (Supplement C) (2017) 137 – 151.

[18] M. Dunbabin, L. Marques, Robots for Environmental Monitoring: Significant Advancements and Applications, IEEE Robotics & Automation Magazine 19 (1) (2012) 24–39.

[19] K. Vicencio, B. Davis, 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.

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

[21] 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. thesis, University of Maryland (2009).

[22] G. Laporte, A. Asef-Vaziri, C. Sriskandarajah, Some applications of the generalized travelling salesman problem, The Journal of the Operational Research Society 47 (12) (1996) 1461–1467.

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

[24] S. L. Smith, F. Imeson, GLNS: An effective large neighborhood search heuristic for the generalized traveling salesman problem, Comput. Oper. Res. 87 (C) (2017) 1–19.

[25] F. Carrabs, C. Cerrone, R. Cerulli, M. Gaudioso, A novel discretization scheme for the close enough traveling salesman problem, Computers & Operations Research 78 (2017) 163–171.

[26] J. Faigl, L. Přeučil, Self-Organizing Map for the Multi-Goal Path Planning with Polygonal Goals, in: International Conference on Artificial Neural Networks (ICANN), Springer, Heidelberg, 2011, pp. 85–92.

[27] J. Faigl, V. Vonásek, L. Přeučil, Visiting Convex Regions in a Polygonal Map, Robotics and Autonomous Systems 61 (10) (2013) 1070–1083. doi:10.1016/j.robot.2012.08.013.

[28] E. Balas, The prize collecting traveling salesman problems, Networks 19 (1989) 621–636.

[29] J. Faigl, G. A. Hollinger, Autonomous data collection using a self-organizing map, IEEE Transactions on Neural Networks and Learning Systems 29 (5) (2018) 1703–1715. doi:10.1109/TNNLS.2017.2678482.

[30] B. L. Golden, L. Levy, R. Vohra, The orienteering problem, Naval Research Logistics (NRL) 34 (3) (1987) 307–318.

[31] A. Gunawan, H. C. Lau, P. Vansteenwegen, Orienteering Problem: A survey of recent variants, solution approaches and applications, European Journal of Operational Research 255 (2) (2016) 315–332.

[32] J. Faigl, R. Pěnička, G. Best, Self-organizing map-based solution for the orienteering problem with neighborhoods, in: IEEE International Conference on Systems, Man, and Cybernetics (SMC), 2016, pp. 1315–1321.

[33] J. Faigl, On self-organizing maps for orienteering problems, in: International Joint Conference on Neural Networks (IJCNN), 2017, pp. 2611–2620.

[34] B. Fritzke, P. Wilke, FLEXMAP - A Neural Network For The Traveling Salesman Problem With Linear Time And Space Complexity, in: International Joint Conference on Neural Networks (IJCNN), 1991, pp. 929–934.

[35] J. Faigl, P. Váňa, Unsupervised learning for surveillance planning with team of aerial vehicles, in: International Joint Conference on Neural Networks (IJCNN), 2017, pp. 4340–4347.

[36] J. Faigl, P. Váňa, Surveillance planning with Bézier curves, IEEE Robotics and Automation Letters 3 (2) (2018) 750–757.

[37] G. Reinelt, TSPLIB - A Traveling Salesman Problem Library, Journal on Computing 3 (4) (1991) 376–384.

[38] Bo Yuan, M. Orlowska, S. Sadiq, On the Optimal Robot Routing Problem in Wireless Sensor Networks, IEEE Transactions on Knowledge and Data Engineering 19 (9) (2007) 1252–1261.

[39] A. Dumitrescu, J. Mitchell, Approximation algorithms for TSP with neighborhoods in the plane, J. Algorithms 48 (1) (2003) 135–159.

[40] M. de Berga, J. Gudmundssonb, M. J. Katzc, C. Levcopoulosd, M. H. Overmarse, A. F. van der Stappen, TSP with neighborhoods of varying size, J. Algorithms 57 (1) (2005) 22–36.

[41] S. Safra, O. Schwartz, On the complexity of approximating tsp with neighborhoods and related problems, Computational Complexity 14 (4) (2006) 281–307.

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

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

[44] J.-C. Créput, A. Koukam, A memetic neural network for the Euclidean traveling salesman problem, Neurocomputing 72 (4-6) (2009) 1250–1264.

[45] K. A. Smith, Neural networks for combinatorial optimization: A review of more than a decade of research, INFORMS Journal on Computing 11 (1) (1999) 15–34.

[46] S. Somhom, A. Modares, T. Enkawa, A self-organising model for the travelling salesman problem, Journal of the Operational Research Society 48 (9) (1997) 919–928.

[47] L. I. Burke, P. Damany, The guilty net for the traveling salesman problem, Computers and Operations Research 19 (3-4) (1992) 255–265.

[48] Y. Liu, R. Bucknall, Efficient multi-task allocation and path planning for unmanned surface vehicle in support of ocean op-

erations, Neurocomputing 275 (2018) 1550–1566.

[49] B. Avar, D. E. Aliabadi, Parallelized neural network system for solving Euclidean traveling salesman problem, Applied Soft Computing 34 (2015) 862–873.

[50] R. Ahmad, D. Kim, An Extended Self-Organizing Map based on 2-opt algorithm for solving symmetrical Traveling Salesperson Problem, Neural Computing and Applications 26 (4) (2015) 987–994.

[51] S. Lin, B. W. Kernighan, An Effective Heuristic Algorithm for the Traveling-Salesman Problem, Operations Research 21 (2) (1973) 498–516.

[52] J. Faigl, M. Kulich, V. Vonásek, L. Přeučil, An application of self-organizing map in the non-euclidean traveling salesman problem, Neurocomputing 74 (5) (2011) 671–679.

[53] J. Faigl, Approximate Solution of the Multiple Watchman Routes Problem with Restricted Visibility Range, IEEE Transactions on Neural Networks 21 (10) (2010) 1668–1679.

[54] J. Faigl, On the performance of self-organizing maps for the non-euclidean traveling salesman problem in the polygonal domain, Information Sciences 181 (2011) 4214–4229.

[55] J. Faigl, P. Va, Self-organizing map for data collection planning in persistent monitoring with spatial correlations, in: IEEE International Conference on Systems, Man, and Cybernetics (SMC), 2016, pp. 3264–3269.

[56] J. Faigl, P. Váňa, Self-organizing map for the curvature-constrained traveling salesman problem, in: International Conference on Artificial Neural Networks (ICANN), Springer International Publishing, 2016, pp. 497–505.

[57] G. Best, J. Faigl, R. Fitch, Multi-robot path planning for budgeted active perception with self-organising maps, in: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2016, pp. 3164–3171.

[58] J. Faigl, Self-organizing map for orienteering problem with dubins vehicle, in: 12th International Workshop on Self-Organizing Maps and Learning Vector Quantization, Clustering and Data Visualization (WSOM+), 2017, pp. 125–132.

[59] J. Faigl, R. Pěnička, On close enough orienteering problem with dubins vehicle, in: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2017, pp. 5646–5652.

[60] G. Best, J. Faigl, R. Fitch, Online planning for multi-robot active perception with self-organising maps, Autonomous Robots 42 (4) (2018) 715–736.

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

[62] M. Tucci, M. Raugi, Stability analysis of self-organizing maps and vector quantization algorithms, in: International Joint Conference on Neural Networks (IJCNN), 2010, pp. 1–5.

[63] D. Applegate, R. Bixby, V. Chvátal, W. Cook, CONCORDE TSP Solver, [cited 12 Dec 2017] (2003).
URL `http://www.tsp.gatech.edu/concorde.html`

[64] K. Helsgaun, LKH, version 2.0.7, [cited 14 Apr 2018] (2012).
URL `http://www.akira.ruc.dk/~keld/research/LKH/LKH-2.0.7.tgz`

[65] PassMark Software Pty Ltd, Cpu performance comparison, [cited 12 Dec 2017] (2017).
URL `\url{https://www.cpubenchmark.net/compare.php?cmp[]=1137\&cmp[]=2565}`

[66] P. Hansen, N. Mladenović, Variable neighborhood search: Principles and applications, European Journal of Operational Research 130 (3) (2001) 449–467.