Unsupervised Learning for Surveillance Planning with Team of Aerial Vehicles

Jan Faigl and Petr Váňa
Czech Technical University, Faculty of Electrical Engineering
Technická 2, 166 27, Prague, Czech Republic
Email: faiglj@fel.cvut.cz

***Abstract*—In this paper, we extent an existing self-organizing map (SOM)-based approach for the Dubins traveling salesman problem (DTSP) to solve its multi-vehicle variant generalized for visiting target regions called k-DTSP with Neighborhoods (k-DTSPN). The Dubins TSP is a variant of the combinatorial TSP for curvature-constrained vehicles. The problem is to determine a cost efficient path to visit a given set of continuous regions while the path allows to satisfy kinematic constraints of non-holonomic vehicles. The k-DTSPN is a generalization to determine k such paths, one for each vehicle. Although the k-DTSPN has been addressed by evolutionary methods, the proposed approach is able to provide solutions very quickly in units of seconds on conventional computationally resources which makes the proposed SOM-based approach suitable for on-line planning. The studied problem is motivated by surveillance task in which it is required to quickly provide information about the given set of target locations. Therefore, real computational requirements are crucial properties of the desired k-DTSPN solver. The proposed method meets this requirement and feasibility of the found solutions are demonstrated not only in computer simulations but also with a practical deployment on real aerial vehicles.**

I. Introduction

The herein studied problem is motivated by surveillance planning to quickly validate that objects of interest are at the particular locations. Based on the prior information about the locations of the objects, the problem is to verify if the expected objects are really the objects of our interest and what a particular type the object is. The problem is motivated by practical needs of multi-robot team deployment in the robotic competition MBZIRC [1], [2]. In this competition, a fleet of Unmanned Aerial Vehicles (UAVs) has to quickly identify where objects of interest are located. Then, based on the score associated with the object, the system has to decide which objects should be collected and delivered to the dedicated area. In our approach, an initial scan of the area is provided from the overfly at very high altitude which provides a rough estimate about the location of the objects of interest. Then, the UAVs are requested to quickly visit the expected object locations to confirm the locations and reject false positive estimates. The UAVs fly relatively fast (approximately 5 m.s^{-1}) and the most time-consuming operation is the pickup and delivery of the object of interest. However, the time to collect as most valuable objects as possible is limited, and therefore, it is beneficial to identify locations of the most valuable objects and do not spend too much time in planning the trajectories to verify the object locations because any additional second spent in planning is a significant distance the vehicle can travel. Hence, we quickly need a feasible solution of the multi-robot multi-goal path planning problem to allocate the robots in the verification and identification of the most valuable objects of interest. A fast solver for this problem is the main topic addressed in this paper.

The presented work was supported by the Czech Science Foundation (GAČR) under research project No. 16-24206S.

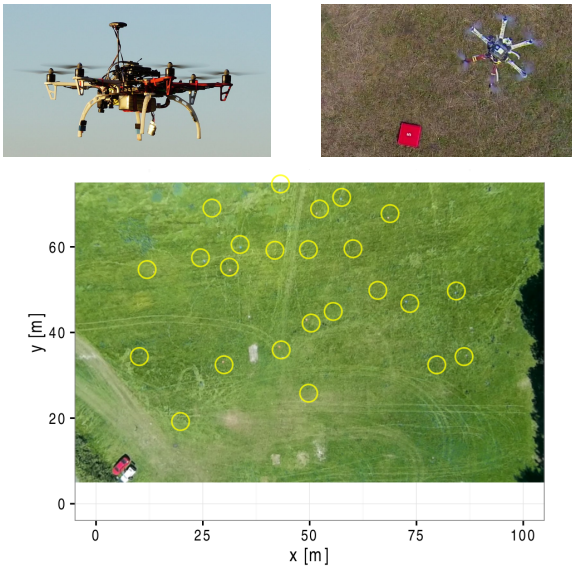


Fig. 1. The used hexa-rotor UAV, the vehicle in the vicinity of the object of interest, and demonstration field site with the objects of interest. The field site is about 80 m $\times$ 60 m large.

A problem of finding a cost efficient path to visit a set of locations is a variant of the combinatorial Traveling Salesman Problem (TSP) for which many approaches have been proposed [3]. However, when planning a path for a non-holonomic vehicle with the limited minimal turning radius, e.g., such as fixed-wing aircraft or car-like vehicle, it is necessary to consider the kinematic model of the vehicle to ensure the found path is feasible for the robot and a real robot will visit the required locations with a sufficient precision. Moreover, the curvature-constrained path is also suitable for micro aerial vehicles such as hexa-rotors used in the motivational practical deployment, see Fig. 1. Curvature-

constrained trajectory is desirable for these vehicles because it allows a fast motion (i.e., the maximal forward velocity) and more precise trajectory following than trajectories with sharp turns with respect to the particular controller employed for the trajectory following [4].

An existing and widely used model of the curvature-constrained vehicles is called the Dubins vehicle and the variant of the curvature-constrained TSP is called the Dubins TSP (DTSP) [5]. Further, considering the surveillance missions with a camera sensor with a particular field of view, it is not necessary to visit the location exactly as the object of interest can be captured from its vicinity respecting the sensing range δ to reliably capture the object. Therefore, the travel cost can be saved by determining not only the order to inspect the objects of interest, but also by determining the most suitable waypoints at which the objects can be captured by a downward looking camera. Thus, a suitable problem formulation of such a planning problem for a single vehicle is called the DTSP with Neighborhoods (DTSPN) [6], [7], [8].

Similarly to the generalization of the TSP to multi-vehicle problem formulation, the DTSPN with a team of Dubins vehicles is called the k-Dubins TSPN (k-DTSPN) and it has been introduced in [9]. Even though the k-DTSPN shares many similarities with the k-TSP and the Vehicle Routing Problem [10], [7], there are only few approaches that directly address the k-DTSPN. The authors of [9] proposed an evolutionary algorithm that provides solutions for a team of Dubins vehicles starting at a common location, but do not report on real required computational time. In [11], a memetic algorithm for the k-DTSPN has been proposed, but again authors did not report on the real required computational times.

In addition to the aforementioned evolutionary methods, a novel unsupervised learning technique of the self-organizing map (SOM) for the TSP has been proposed to address the specific properties of the Dubins TSP in [12]. The authors compare the performance of the SOM-based approach with the existing heuristics for the DTSP [5], [13], and the Memetic algorithm [11]. The reported results indicate the SOM-based approach provides better results than the Memetic algorithm [11] in some problem instances while it is computationally less demanding than the Memetic algorithm which runtime has been restricted to 10 seconds. The results [12] together with existing generalization of the SOM for the TSP to address the TSP with Neighborhoods (TSPN) [14], [15] and k-TSP variants [16], [17] motivated us to further consider the ideas of unsupervised learning employed in the SOM for the TSP to develop SOM-based solution of the k-DTSPN that will quickly provide a feasible solution for the real aerial vehicles.

Due to the limited mission time in the motivational deployment of the robotic competition, the expected time needed to find a feasible solution of the k-DTSPN instances arising from scenarios of the competition was specified to do not exceed one second. Regarding the velocity of the UAVs, one second means about five meters and ten seconds is about a quarter of the arena size, and thus the required time to find a solution should be significantly smaller than the time to travel over the arena. Therefore, finding an optimal solution at the cost of high computational requirements is not desirable and it is rather preferred to find a feasible (good enough) solution quickly.

The proposed solution is built on early results on the SOM for the DTSP reported in [12] which has been further developed to meet the desired properties of the surveillance planning with a team of aerial vehicles. In particular, the main contributions of the paper are as follows:

- Improved adaptation procedure to quickly find a feasible solution of the DTSP.
- Generalization of the method to save the travel cost by considering sensing range δ, i.e., a solution of the DTSP with Neighborhoods (DTSPN).
- Generalization of the method to solve the problem with a team of vehicles, i.e., a solution of the k-DTSPN.
- Comparison of the proposed method with the existing solver based on the Memetic algorithm [11].
- Verification of the feasibility of the found solutions using real aerial vehicles.

The paper is organized as follows. The problem statement together with a brief overview of Dubins vehicle model and Dubins multi-goal planning are introduced in Section II. The proposed algorithm is proposed in Section III. Empirical results and reports on experimental deployment are presented in Section IV. Conclusion and final remarks are in Section V.

II. Problem Statement

The motivation of the studied problem is to determine k curvature-constrained paths for the k aerial vehicles such that all n objects of interest can be seen from at least one vehicle traveling along its path. Therefore, it is required that for each object of interest, at least one path is closer than the given sensing range δ to provide a reliable snapshot of the object and identification of the reward value associated with the object. All the vehicles are identical with the same minimal turning radius ρ that allows flying at the constant, maximal, and safe forward velocity while the error of the trajectory following (based on the controller [4]) is at the acceptable level. Finally, each vehicle starts at its individual location $p_d^r \in \mathbb{R}^2$, where the superscript r denotes the r-th vehicle, and the path has to ends at the same location p_d^r. The formal definition of the problem with the necessary background arising from the non-holonomic constraints of the used Dubins vehicle model are presented in the rest of this section.

The used vehicle model is the Dubins vehicle [18] with the constant forward velocity v and the minimal turning radius ρ. The state of each vehicle $q = (x, y, \theta)$ consists of its position $p = (x, y)$ in the plane $p \in \mathbb{R}^2$ and its heading θ, $\theta \in \mathbb{S}^1$, i.e., $q \in SE(2)$. The mathematical model of the vehicle motion can be described by (1), where u is the control input.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = v \begin{bmatrix} \cos\theta \\ \sin\theta \\ u \cdot \rho^{-1} \end{bmatrix}, \quad |u| \leq 1. \tag{1}$$

Having two configurations $q_1, q_2 \in SE(2)$, the optimal path connecting q_1 and q_2 respecting the kinematic constraints (1)

consists of straight line segment (S) and arcs (of two types L – left, R – right) with the curvature ρ. Such a path can be called the Dubins maneuver and only six possible combinations exist: LSL, LSR, RSL, RSR, LRL, and RLR [18].

The optimal Dubins maneuver can be computed analytically [18]; however, to find an optimal path connecting two points $p_1, p_2 \in \mathbb{R}^2$ we need the orientation (heading) of the vehicle at these points. Therefore, a solution of the DTSP consists not only of determining the sequence to visit the locations (as in the ordinary TSP) but also the optimal heading at each location has to be determined. Moreover, if the object of interests at the location $o_i \in \mathbb{R}^2$ can be covered from anywhere at the δ distance to o_i, the problem becomes the DTSPN where in addition to the headings and sequence of visits, we also need to determine a particular waypoint p_i at which each object o_i is covered, i.e., $|(p_i, o_i)| < \delta$, such that the total length of the Dubins path is minimized.

The motivational problem is formulated as the k-DTSPN which stands to find k Dubins paths $\{\mathcal{L}^1, \dots \mathcal{L}^k\}$ for k Dubins vehicles such that locations of the all n objects of interests $\mathcal{O} = \{o_1, \dots, o_n\}$ are in at most the δ-distance from one of the found paths and the length of the longest path is minimal, i.e., the MinMax variant of the multi-vehicle TSP [17]. Each individual path $\mathcal{L}^r$ for the vehicle r can be described as a sequence of waypoints $Q^r = (q^r_d, q^r_1, \dots, q^r_{n^r}, q^r_d)$, where the projection of the waypoint q^r_d to $\mathbb{R}^2$ corresponds to the vehicle starting location p^r_d, i.e., $\mathcal{P}(q^r_d) = \mathcal{P}(q^r_{n^r}) = p^r_d$, which is also supposed to be the vehicle final location. The number of waypoints n^r of the path r corresponds to the number of objects of interests that are covered by the path $\mathcal{L}^r$, i.e.,

$$\mathcal{O}^r = \{o_i \in \mathcal{O} | \text{ for which } |(\mathcal{P}(q^r_i), o_i)| < \delta, q^r_i \in Q^r\}. \quad (2)$$

The length of the path $\mathcal{L}^r$ defined by the waypoints Q^r and the covered objects $\mathcal{O}^r$ can be expressed as

$$L^r(Q^r, \mathcal{O}^r) = \mathcal{L}(q^r_d, q^r_1) + \sum_{i=1}^{n^r} \mathcal{L}(q^r_i, q^r_{i+1}) + \mathcal{L}(q^r_{n^r}, q^r_d), \quad (3)$$

where $\mathcal{L}(q_i, q_j)$ denotes the length of the Dubins maneuver connecting the configurations q_i and q_j. Notice, each configuration q_i consists of the point location $p_i \in \mathbb{R}^2$ and the vehicle heading $\theta_i \in \mathbb{S}^1$, i.e., $q_i = (p_i, \theta_i)$.

Having a notion of the individual path $\mathcal{L}^r$, its waypoints, and objects $\mathcal{O}^r$ covered from $\mathcal{L}^r$, the k-DTSPN can be defined as a problem to determine for each vehicle r, the sequence of waypoints Q^r and a subset of n^r locations of the objects of interest $\mathcal{O}^r \subseteq \mathcal{O}$ such that all objects are covered and the length of the longest tour is minimal. The problem can be formally defined as:

$$\textit{minimize}_{(Q^r, \mathcal{O}^r) \text{ for } r \in \{1,\dots,k\}} \quad \max_{r \in \{1,\dots,k\}} L^r(Q^r, \mathcal{O}^r) \quad (4)$$

$$\begin{aligned} \textit{subject to} \quad & \mathcal{O} = \bigcup_{r=1}^{k} \mathcal{O}^r \text{ and for } o \in \mathcal{O} \text{ there is} \\ & q^r \in \textstyle\bigcup_{r=1}^{k} Q^r \text{ such that } |(\mathcal{P}(q^r), o)| < \delta. \end{aligned} \quad (5)$$

Notice, it is sufficient that each object is covered by at least one vehicle but regarding (4) it can be covered by more vehicles if it does not affect the length of the longest path.

III. Self-Organizing Map for the k-DTSPN

The proposed approach to address the k-DTSPN is directly based on the self-organizing map (SOM) for the DTSP introduced in [12]. Here, the unsupervised learning technique is extended to address the neighborhood defined as the disk with the radius δ centered at the location of each object of interest, i.e., a solution of the Dubins TSP with Neighborhoods (DTSPN). This extension is based on the idea to determine the waypoint at the δ-distance from the location during the winner selection. The idea has been proposed in [19] and utilized in the solution of the TSPN in [14] and further developed in [15]. The generalization of the proposed SOM-based solution for the DTSPN to the k-DTSPN follows the idea of the SOM-based solution of the k-TSP [16], [17] in which an individual neural network is created for each vehicle. Each network (called a ring) represents a path for one vehicle and during the winner selection, the length of the current path is taken into account to prefer winner neurons from shorter rings (networks), and thus minimize the longest path [16].

The SOM for the k-DTSPN is a direct extension of the SOM for the DTSPN. Moreover, the solution of the DTSPN follows the general adaptation procedure of the SOM for the TSP. Therefore, a general idea of the employed unsupervised learning is presented prior the proposed modifications to make the reader familiar with the main principles of the employed learning technique. Then, we introduce an extension of the learning to address the curvature-constrained TSP [12] together with the proposed modification of the winner selection to address the DTSPN variant. Finally, the rule to prefer winner neurons from shorter rings, and thus solve the k-DTSPN is presented.

A. Self-Organizing Map for the TSP

The SOM for the TSP is two-layered neural network which maps the input space, the set of target locations (cities) $\mathcal{O}$, into a finite number of the output neurons organized into an array of the output units [20]. Contrary to a conventional SOM for clustering or classification problems [21], the SOM for the TSP does not represent a 2D lattice but the output layer is one dimensional and the neuron weights share the space with the input signals, and thus connected neurons form a ring of nodes that represents the desired path to visit the target locations [22].

For each vehicle r, a separate neural network is created with the neurons $\mathcal{N}^r = (\nu^r_1, \dots, \nu^r_{m^r})$ where ν^r_i represents the vehicle configuration in the input space and m^r is the actual number of neurons in the r-th ring. In the ordinary Euclidean TSP, each neuron ν^r_i represents a position in the input space $\nu^r_i \in \mathbb{R}^2$ and the final path is constructed simply by connecting the neurons in the ring by straight line segments [22]. However, for the DTSPN, we need to consider also the vehicle heading, and therefore, the neuron represents the Dubins vehicle configuration, i.e., $\nu^r_i \in SE(2)$, and the final

curvature-constrained path has to be constructed by connecting the neurons by the Dubins maneuver. Even though the neuron representation differs in a solution of the TSP and the DTSP, a solution of the both problems can utilize the same SOM-based framework for the unsupervised learning. The framework can be summarized as follows.

1) *Initialization:* For n target locations $\mathcal{O}$, create a ring of neurons with randomly initialized weights, e.g., with $2n$ neurons [14]. Initialize the learning parameters as follows: the learning gain $\sigma = 10$, the learning rate $\mu = 0.6$, the gain decreasing rate $\alpha = 0.1$, and set the learning epoch counter $i = 1$.
2) *Randomizing:* Create a random permutation of locations $\Pi(\mathcal{O})$ to avoid local minima [23].
3) *Learning epoch*: For each $o \in \Pi(\mathcal{O})$
   a) *Select winner* neuron ν^* for o as the best matching neuron, i.e., the closest neuron to o.
   b) *Adapt* ν^* and its neighbors to o using the neighbouring function (6), i.e., set the neuron weights to the new locations ν' determined as:
   $$\nu' = \nu + \mu f(\sigma, d)(o - \nu).$$
4) *Ring regeneration* to improve headings associated with the neurons to optimize the length of the Dubins path represented by the ring. (For solving DTSP and DTSPN)
5) *Update learning parameters:* $\sigma = \sigma(1 - i\alpha)$, $i = i + 1$.
6) *Termination condition:* If solution is not improving or $i \geq i_{max}$ `Stop` the adaptation. Otherwise go to Step 2.
7) *Construct the final (Dubins) tour* using the last winners.

The used neighbouring function follows the existing SOM for the TSP [23], [22] and it has the form

$$f(\sigma, d) = \begin{cases} e^{\frac{-d^2}{\sigma^2}} & \text{for } d < 0.2m^r \\ 0 & \text{otherwise [23]} \end{cases}, \qquad (6)$$

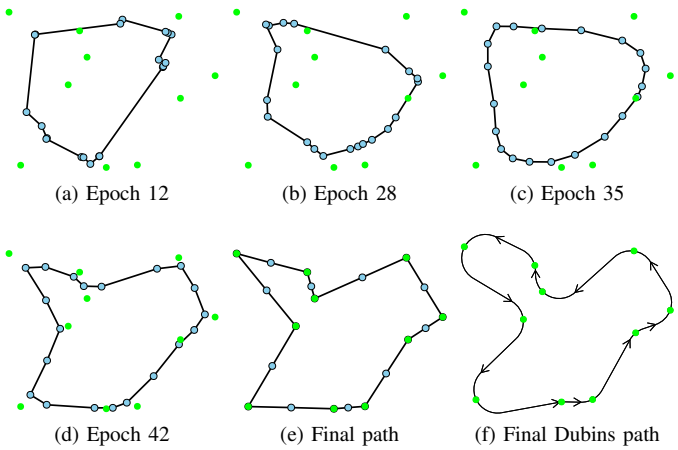
which decreases the power of adaptation of the neighbouring nodes to the winner neuron ν^* with increasing distance d of the neuron to ν^* counted in the number of neurons of the ring.

The adaptation can be viewed as a movement of the neurons to new position ν' which replaces the neuron weights ν. An evolution of the ring of neurons is shown in Fig. 2.

B. SOM for the DTSP and Winner Selection for the DTSPN

The neurons represent the particular waypoints of the path and to directly solve the DTSP by SOM, the neurons also contain information about the expected vehicle headings [12]. Then, the Dubins path can be directly computed as a sequence of the Dubins maneuvers connecting the neurons in the ring. However, the heading value at the particular waypoint can significantly affect the total path length, therefore each neuron $\nu_i \in \mathcal{N}$ has associated a set of heading values $\Theta_i = \{\theta_i^{-k}, \theta_i^{-k+1} \ldots, \theta_i, \theta_i^k, \ldots, \theta_i^k\}$ which are used to find Dubins path represented by the current ring.

Having a sequence of neurons in the ring and each neuron ν_i has a set of possible heading values Θ_i, the best heading values to represent the shortest possible Dubins path connecting the neurons in the ring can be found as follows. We can consider



(a) Epoch 12 (b) Epoch 28 (c) Epoch 35

(d) Epoch 42 (e) Final path (f) Final Dubins path

Fig. 2. An example of the ring evolution of SOM for the TSP with 10 target locations. The target locations are represented as the green disks while the current neurons are in blue. After 53 epochs a solution of the Euclidean TSP is determined which provides a sequence of the target locations which can be used to determine headings at the waypoints and a solution of the DTSP.

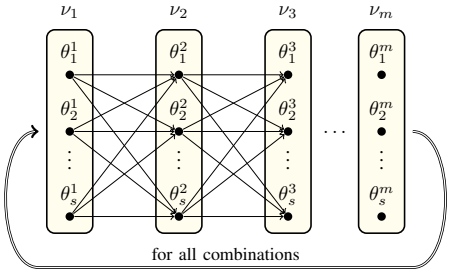


Fig. 3. A search graph where each layer corresponds to one neuron with particular heading values Θ_i. Two neighboring layers are fully connected by the oriented edges representing the direction of the vehicle.

the ring of connect neurons as a search graph, where each neuron represents a layer with possible heading values Θ_i and two neighbouring layers are fully connected, see Fig. 3. Then, the best heading values are selected by a feed-forward search with the time complexity bounded by $O(ms^3)$, where m is the current number of neurons in the ring and s is the number of heading values associated with each neuron.

The above described procedure allows to determine Dubins path that is represented by the current ring of neurons. This path is further utilized in the novel procedure to select the winner neuron. Contrary to the approach [12], the winner neuron is determined as the closest point p_o of the Dubins path connecting the current neurons in the ring. Therefore, prior the actual selection of the winner neuron to the currently presented location o to the network, the point p_o as the closest point of the Dubins path to the location o is determined, see Fig. 4. Then, the winner neuron ν^* is the neuron with the weights identical to p_o. If such a neuron does not exist in the ring, a new neuron is created and its weights are set to be identical to the point p_o. Notice, the vehicle heading θ_p at the point p_o is utilized as the main heading of the winner neuron and the other heading values of Θ_{ν^*} are set around θ_p as $\Theta_{\nu^*} = \{\theta_p, \theta_p^1, \ldots, \theta_p^i, \theta_p^{-1}, \ldots, \theta_p^{-i}\}$, where $\theta_p^i = \theta_p + i\pi/l$, $1 \leq i \leq l$ and l is set to $l = 12$.

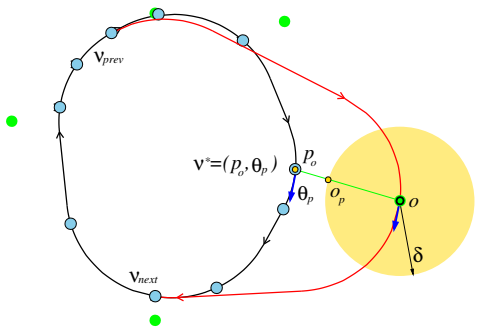


Fig. 4. Proposed winner selection procedure: the location o is presented to the network where the current ring of neurons represents the Dubins path showed as the black curve connecting the blue neurons. The closet point p_o of the Dubins path to o is used as the neuron weights for the winner neuron. The point o_p corresponds to the alternate target location towards which the network is adapted to save the travel cost by covering o within δ distance from o. The shortest possible path connecting ν_{prev} and ν_{next} through the point o using the vehicle heading θ_p is in red.

In addition to the determination of the point p_o, the point o_p as the intersection of the straight line segment (p_o, o) (defined by the points p_o and o) with the δ-radius circle centered at o is determined if $|(p_o, o)| > \delta$. The point o_p is than considered as the target location at which the object o can be covered with the sensing range δ, and thus the network is adapted towards o_p and not to o to save the travel cost. However, if p_o is already within the δ-distance from o, the particular winner neuron is determined, but the network is not adapted as o can be already covered from p_o.

Another important modification of the SOM for the Dubins TSP is the adaptation of the neurons. Instead of adaptation of the neighbouring neurons as in the ordinary SOM for the TSP using the fixed neighborhood, e.g., $d < 0.2m$, the neighborhood is defined by the neurons ν_{prev} and ν_{next} that are determined such that the length of the expected Dubins path to visit the location o is minimized:

$$L_g = \mathcal{L}(\nu_{prev}, (o, \theta)) + \mathcal{L}((o, \theta), \nu_{next}), \tag{7}$$

where ν_{prev}, ν_{next} are neurons from the activation bubble $\mathcal{A}(\nu^*)$ around ν^* and θ is one of the heading values $\theta \in \Theta_{\nu^*}$ of the winner neuron ν^*. The activation bubble $\mathcal{A}(\nu^*)$ consists of all neurons around the winner ν^* for which the current value of the neighbouring function (6) is above the activation threshold, which is set to 10^{-5}. The neighbouring function $f(\sigma, d)$ depends on the learning gain σ which is decreasing after each learning epoch, and therefore, neurons ν_{prev} and ν_{next} may not be necessarily found. In such a case, only the particular winner neuron is adapted towards the point o_p.

Although neurons ν_{prev} and ν_{next} are determined according to the Dubins path connecting o, the neurons are adapted towards the alternate location o_p to save the total tour length as in [15]. Since, the shape of the optimal Dubins maneuver depends on the departure and arrival angles at ν_{prev} and ν_{next}, we further modify the original adaptation step of SOM, and only neurons between ν_{prev} and ν_{next} (including the winner ν^*) are adapted towards o_p. This is performed by moving ν^* to o_p and determination of the optimal Dubins maneuvers from ν_{prev} to ν^* and from ν^* to ν_{next}. Then, the weights of each neuron between ν_{prev} and ν_{next} are set to the corresponding closest points on these maneuvers. The neurons ν_{prev}, ν_{next}, and ν^* are marked as active neurons for the current epoch.

Because the winner ν^* may be added to the network in the winner selection step, one another neuron between ν_{prev} and ν_{next} is removed (if such exists) from the network during the adaptation to decrease the computational burden and keep the number of neurons below $2n$. Moreover, after each learning epoch, a ring regeneration is performed to improve a solution of the DTSP and remove unused neurons. Thus, only the active neurons are preserved and their main headings ν_θ are set according to the determined the shortest Dubins path connecting neurons by the forward search procedure using the graph representation visualized in Fig. 3.

C. SOM for the k-DTSPN

The generalization of the proposed SOM for the DTSPN to k-DTSPN is straightforward and it is based on the creation of the individual ring of neurons $\mathcal{N}^r$ for each vehicle $r \in \{1, \ldots, k\}$. Then, the main objective of the k-DTSPN (4) is addressed by preferring selection of the winner neuron from shorter rings, and thus minimize the length of the longest path. Therefore, an average length of the Dubins paths represented by each ring of neurons $\mathcal{N}^r$ denoted $L(\mathcal{N}^r)$ is determined during the SOM learning

$$L_{avg} = \frac{1}{k} \sum_{r=1}^{k} L(\mathcal{N}^r). \tag{8}$$

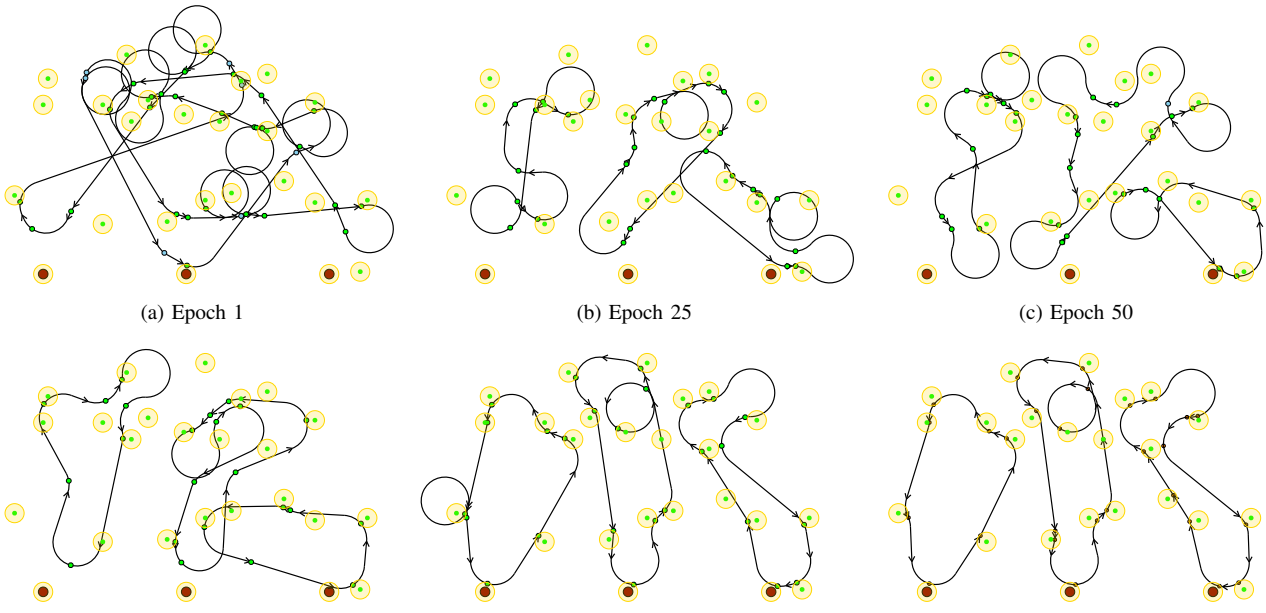
Then, the weights of the winner neuron are still selected as the closest point p_o^r of the individual ring r to the location o. However, instead selecting the p_o^r from the ring r for which $|(p_o^r, o)|$ is minimal, the distance between p_o^r and o is weighted by the multiplication factor proportional to the difference of the Dubins path length represented by the particular ring $\mathcal{N}^r$ to the average length of the Dubins paths L_{avg}. Thus, the ring r from which the particular point p_o^r is used for the determination of the winner neuron ν^* is determined as

$$r = \underset{r \in \{1,\ldots,k\}}{\operatorname{argmin}} \left(1 + \frac{L(\mathcal{N}^r) - L_{avg}}{L_{avg}}\right) |(p_o^r, o)|. \tag{9}$$

Then, the particular p_o^r of the selected ring r for the adaptation is used as in the solution of the single vehicle DTSPN described in Section III-B.

Besides, the adaptation is further modify to respect individual starting locations of each vehicle as follows. At the beginning of each learning epoch, each ring $\mathcal{N}^r$ is adapted towards the particular starting locations p_d^r using the first neuron of the ring without competition between neurons. This ensures that each Dubins path represented by the ring starts and ends at the particular p_d^r.

Here, we would like to comment that all the results presented in this paper consider the p_d^r also with the δ distance. Even though the SOM adaptation procedure allows to use an individual δ per each object of interest, for simplicity



(a) Epoch 1 (b) Epoch 25 (c) Epoch 50

(d) Epoch 75 (e) Epoch 104 – SOM solution (f) Epoch 103 - Final tours

Fig. 5. Evolution of the proposed SOM for the k-DTSPN with $n = 22$ target locations represented by small green disks. The yellow disks represent $\delta = 2$ (in meters) neighborhood to cover the target locations. The minimal turning radius of the Dubins vehicle is $\rho = 5$ meters. The initial positions of the vehicles are the red disks. The final found solution has the maximal path 146.2 meters long.

of the algorithm and comparison with existing methods, we allowed to do not exactly start at p_d^r. Regarding the practical deployment, this is not an issue as the vehicles have to be firstly positioned at the particular locations before they are asked to follow the planned trajectories. Additional travels about δ distance do not add a significant delay because of moving speed of the real vehicles.

The evolution of SOM is similar to the solution of the TSP; however, the ring of neurons directly represents Dubins path and the network not only provides estimation of the headings at the waypoints, but also the particular locations to cover the objects of interests at the δ-distance to them, and thus the rings represent a solution of the k-DTSPN. An example of the network evolution and the final found solution is shown in Fig. 5. Empirical evaluation of the proposed algorithm and comparison with existing approach are presented in the next section together with the reported results on real deployment of the method in the field.

IV. Results

The proposed SOM-based solver for the k-DTSPN has been evaluated in testing scenarios arising from the motivational deployment of the solver in the robotic competition MBZIRC [2], [1]. All the test instances contains 22 objects of interests[1] and the goal is to take a detail photo of each object for which the vehicle needs to be closer than $\delta = 2$ meters. An example of the object with a particular label denoting the object reward is shown in Fig. 1 together with the field site and positions of the objects. We consider three scenarios depending on the number of vehicles used in the mission. The starting locations of the vehicles are prescribed for each particular setup for $k \in \{1, 2, 3\}$. [2] The used UAVs are model as Dubins vehicles (1) with the forward velocity u limited to 5 m.s^{-1} and the physical constraints limit the vehicle lateral acceleration, and thus the minimal turning radius is $\rho = 5$ m.

A. Comparison of the Solution Quality

The evaluation of the proposed SOM-based approach for the k-DTSPN has been performed to validate if the approach provides solutions of sufficient quality with low computational requirements. Therefore, the performance of the proposed solver has been compared with the existing Memetic algorithm [11] that has been used as follows. The same mutation and crossover operators together with the local optimization of the visiting configuration at the δ-distance from the location of the object of interests as in [11] have been utilized. The size of the population has been set to 1000 individuals which is close to the value suggested by the authors of [11]. The mutation probability has been set to 0.1. On the other hand, the only parameter of the proposed SOM-based approach presented in Section III is the maximal number of the learning epochs i_{max} that has been set to $i_{max} = 500$.

Both algorithms are stochastic procedures, therefore each problem instance for $k \in \{1, 2, 3\}$ has been solved 10 times and the solution quality indicators is an average length of the

[1]Locations of the objects of interests (in meters): {(27.5, 46), (10, 35.5), (51.5, 40.5), (32, 36.5), (67, 15), (44, 48), (44, 15.5), (49.5, 17), (60.5, 19.5), (39.5, 33.5), (78, 15.5), (67, 36), (76.5, 0.5), (28.5, 32), (22.5, 10.5), (57, 30), (47, 32), (4, 16.5), (36, 11), (57, 42), (22.5, 35.5), (11, 41)}

[2]The starting locations are (in meters): {(40, 0)} for one vehicle, {(10, 1), (70, 1)} for two vehicles, and {(10, 1), (40, 1), (70, 1)} for three vehicles.

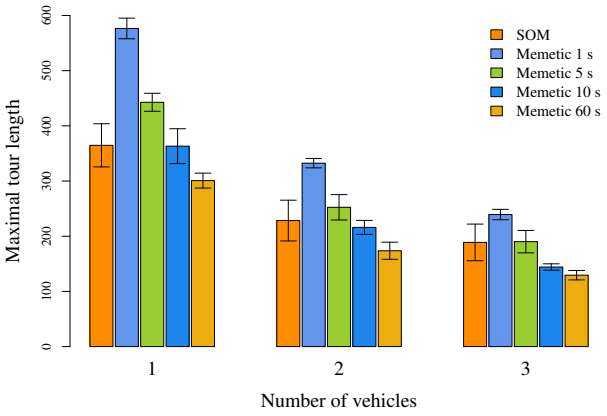


Fig. 6. Average values of the length of the longest tour from 10 trials found by the proposed SOM-based algorithm and the Memetic algorithm [11] with the computational time restricted to 1, 5, 10, and 60 seconds. The error bars denote standard deviations.

longest tour L_{max}. The Memetic algorithm [11] improves the solution according to the available computational time, and therefore, we consider four variants of the algorithm for the computational time limited to 1, 5, 10, and 60 seconds. Both algorithms have been implemented in C++, compiled by the same compiler, and executed within the same computational environment using a single core of the iCore7 processor running at 4 GHz. The average lengths of the longest tours are depicted in Fig. 6.

TABLE I
SOLUTIONS QUALITY AND REQUIRED COMPUTATIONAL TIME

k	**Memetic 1s** L_{max} [m]	**Memetic 10s** L_{max} [m]	**Proposed SOM** L_{max} [m]	**Proposed SOM** T [s]
1	586.01 (24.22)	376.52 (27.17)	363.38 (36.56)	0.55 (0.07)
2	335.83 (10.67)	212.18 (18.73)	223.76 (40.76)	0.53 (0.01)
3	240.67 (6.63)	153.37 (12.79)	180.12 (29.49)	0.53 (0.03)

The results indicate that the proposed SOM-based approach provides better results than the Memetic algorithm [11] with the computational time limited to 1 and 5 seconds. In the case of 10 seconds and $k = 3$, it seems the Memetic algorithm starts providing better results. The computational time of the SOM approach is detailed in Table I which is always less than 0.6 seconds, and thus it is below the specified requirement. The SOM-base solver is able to provide solution with the similar quality to the Memetic algorithm running for 10 seconds. Therefore, the speed up is more than about one order of magnitude. Moreover, the vehicle is capable of traveling 45 meters in 9 seconds, therefore we consider the solutions provided by the SOM-based solver of the sufficient quality for the real deployment on the real aerial vehicles. Selected best-found solutions by each method are depicted in Fig. 8.

B. Real deployment

The real practical experiment with three aerial vehicles has been conducted to demonstrate the found solution is feasible for the real UAVs. The setup of the problem corresponds to the test instance with $k = 3$ evaluated in the simulation. The altitude of the UAVs has been set to 7 meters and the size of the field is approximately 60 m $\times$ 80 m. The GPS has been used for the localization and model predictive controller [4] has been utilized for the trajectory following. The planned and real trajectories are depicted in Fig. 7. Notice, due to noise and trajectory following imperfections, the value of δ for the planning is set to a shorter distance than the actual sensing range to ensure a reliable identification of the objects.

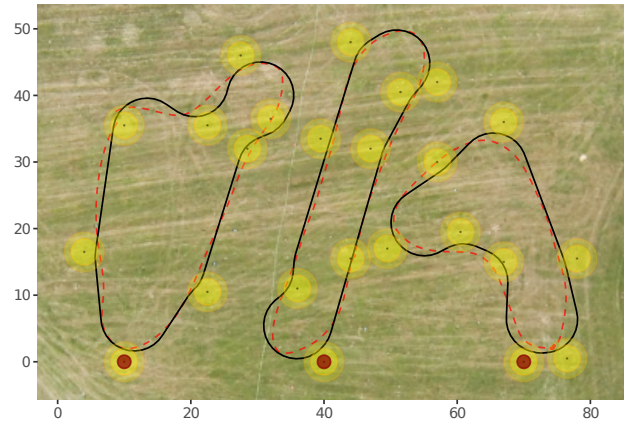


Fig. 7. Planned and real executed trajectories by UAVs

V. CONCLUSION

In this paper, we propose a novel unsupervised learning of SOM-based algorithm to solve the k-DTSPN. The proposed algorithm is based on the practical needs to find a feasible solution under time critical constraints. The SOM-based algorithm satisfies the specified requirements and demonstrated practical usability in field experiments. The solutions of sufficient quality are found in less than 0.6 second and are better than the solution provided by the Memetic algorithm. Thus, the proposed SOM-based algorithm seems to be more suitable for the motivational deployment.

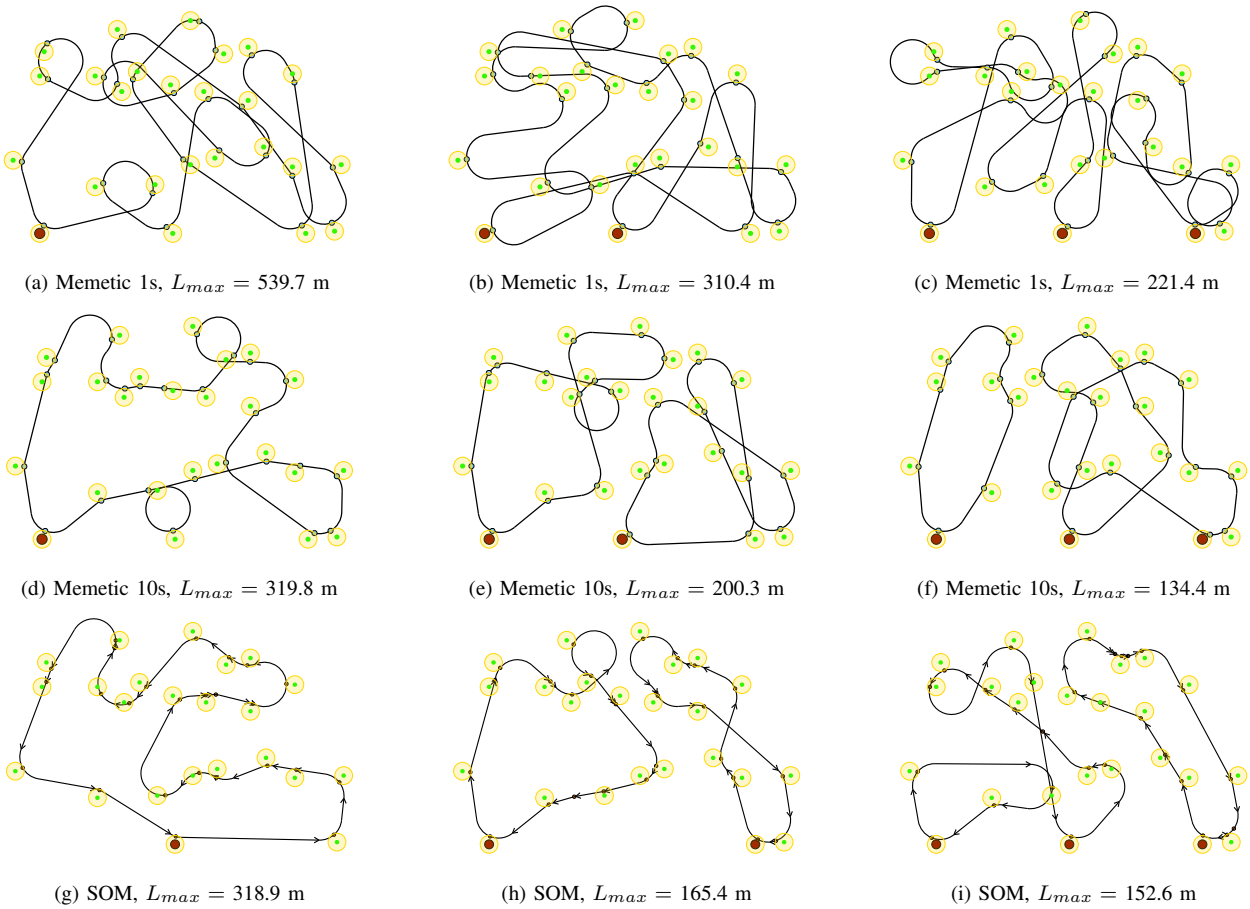
On the other hand, the SOM-based solution does not improve if more computational time is available, which is not the case of the Memetic algorithm. Moreover, the results indicate the solution quality of SOM-based approach does not scale with the increasing number of vehicles as the Memetic algorithm. Therefore, a combination of the proposed SOM as a construction heuristic in the Memetic algorithm may combine advantages of both approaches. Such a combination is a subject of our future work.

ACKNOWLEDGMENTS

The authors would like to thank to Vojtěch Spurný, Tomáš Báča, Robert Pěnička, and Martin Saska for their assistance with the experimental deployment of the presented method with the real UAVs in the field.

REFERENCES

[1] "MBZIRC team of the Czech Technical University," [cited 28 Nov 2016]. [Online]. Available: http://mrs.felk.cvut.cz/projects/mbzirc
[2] "Mohamed Bin Zayed International Robotics Challenge (MBZIRC)," [cited 28 Nov 2016]. [Online]. Available: http://www.mbzirc.com



(a) Memetic 1s, $L_{max} = 539.7$ m (b) Memetic 1s, $L_{max} = 310.4$ m (c) Memetic 1s, $L_{max} = 221.4$ m

(d) Memetic 10s, $L_{max} = 319.8$ m (e) Memetic 10s, $L_{max} = 200.3$ m (f) Memetic 10s, $L_{max} = 134.4$ m

(g) SOM, $L_{max} = 318.9$ m (h) SOM, $L_{max} = 165.4$ m (i) SOM, $L_{max} = 152.6$ m

Fig. 8. Selected best found solutions provided by the Memetic algorithm [11] with the computational time limited to 1 and 10 seconds and the best found solutions provided by the proposed SOM-based solver that are found in less than 0.6 seconds

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

[4] T. Baca, G. Loianno, and M. Saska, "Embedded model predictive control of unmanned micro aerial vehicles," in *21st International Conference on Methods and Models in Automation and Robotics (MMAR)*, 2016, pp. 992–997.

[5] K. Savla, E. Frazzoli, and F. Bullo, "On the point-to-point and traveling salesperson problems for Dubins' vehicle," in *Proceedings of the American Control Conference*. IEEE, 2005, pp. 786–791.

[6] K. J. Obermeyer, "Path planning for a uav performing reconnaissance of static ground targets in terrain," in *AIAA Guidance, Navigation, and Control Conference*, 2009, pp. 10–13.

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

[8] J. T. Isaacs, D. J. Klein, and J. P. Hespanha, "Algorithms for the Traveling Salesman Problem with Neighborhoods Involving a Dubins Vehicle," in *American Control Conference*, 2011, pp. 1704–1709.

[9] D. G. Macharet, A. Alves Neto, V. F. da Camara Neto, and M. F. Campos, "Efficient target visiting path planning for multiple vehicles with bounded curvature," in *IROS*, 2013, pp. 3830–3836.

[10] T. Bektas, "The multiple traveling salesman problem: an overview of formulations and solution procedures," *Omega*, vol. 34, no. 3, pp. 209–219, 2006.

[11] X. Zhang, J. Chen, B. Xin, and Z. Peng, "A memetic algorithm for path planning of curvature-constrained uavs performing surveillance of multiple ground targets," *Chinese Journal of Aeronautics*, vol. 27, no. 3, pp. 622–633, 2014.

[12] J. Faigl and P. Váňa, "Self-organizing map for the curvature-constrained traveling salesman problem," in *International Conference on Artificial Neural Networks*. Springer International Publishing, 2016, pp. 497–505.

[13] P. Váňa and J. Faigl, "On the dubins traveling salesman problem with neighborhoods," in *IROS*, 2015, pp. 4029–4034.

[14] J. Faigl and L. Přeučil, "Self-Organizing Map for the Multi-Goal Path Planning with Polygonal Goals," in *ICANN*, 2011, pp. 85–92.

[15] J. Faigl and G. Hollinger, "Unifying multi-goal path planning for autonomous data collection," in *IROS*, 2014, pp. 2937–2942.

[16] S. Somhom, A. Modares, and T. Enkawa, "Competition-based neural network for the multiple travelling salesmen problem with minmax objective," *Computers & Operations Research*, vol. 26, no. 4, pp. 395–407, 1999.

[17] J. Faigl, "An application of self-organizing map for multirobot multigoal path planning with minmax objective," *Computational Intelligence and Neuroscience*, p. 15, 2016.

[18] L. E. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents," *American Journal of Mathematics*, pp. 497–516, 1957.

[19] J. Faigl, "Approximate Solution of the Multiple Watchman Routes Problem with Restricted Visibility Range," *IEEE Transactions on Neural Networks*, vol. 21, no. 10, pp. 1668–1679, 2010.

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

[21] T. Kohonen, M. R. Schroeder, and T. S. Huang, Eds., *Self-Organizing Maps*, 3rd ed. Springer-Verlag New York, Inc., 2001.

[22] E. M. Cochrane and J. E. Beasley, "The co-adaptive neural network approach to the Euclidean travelling salesman problem," *Neural Networks*, vol. 16, no. 10, pp. 1499–1525, 2003.

[23] S. Somhom, A. Modares, and T. Enkawa, "A self-organising model for the travelling salesman problem," *Journal of the Operational Research Society*, pp. 919–928, 1997.