WiSM: Windowing Surrogate Model for Evaluation of Curvature-Constrained Tours with Dubins vehicle

Jan Drchal, Jan Faigl, and Petr Váňa

***Abstract*—Dubins tours represent a solution of the Dubins Traveling Salesman Problem (DTSP) that is a variant of the optimization routing problem to determine a curvature-constrained shortest path to visit a set of locations such that the path is feasible for Dubins vehicle, which moves only forward and has a limited turning radius. The DTSP combines the NP-hard combinatorial optimization to determine the optimal sequence of visits to the locations, as in the regular TSP, with the continuous optimization of the heading angles at the locations, where the optimal heading values depend on the sequence of visits and vice versa. We address the computationally challenging DTSP by fast evaluation of the sequence of visits by the proposed Windowing Surrogate Model (WiSM) which estimates the length of the optimal Dubins path connecting a sequence of locations in a Dubins tour. The estimation is sped up by a regression model trained using close to optimum solutions of small Dubins tours that are generalized for large-scale instances of the addressed DTSP utilizing the sliding window technique and a cache for already computed results. The reported results support that the proposed WiSM enables a fast convergence of a relatively simple evolutionary algorithm to high-quality solutions of the DTSP. We show that with an increasing number of locations, our algorithm scales significantly better than other state-of-the-art DTSP solvers.**

***Index Terms*—Dubins Vehicle; Dubins Traveling Salesman Problem; Dubins Touring Problem; Surrogate Model**

I. Introduction

In this paper, we address the *Dubins Traveling Salesman Problem* (DTSP) by the proposed regression surrogate model for fast evaluation of possible solution candidates given by a sequence of visits to the locations of interest by speeding up the convergence of an evolutionary solver that supports finding high-quality solutions. The addressed DTSP is motivated by data collection and surveillance missions where a vehicle is requested to visit a set of locations as quickly as possible [1], [2]. The problem is a variant of the routing problem [3], [4] to determine the optimal sequence of visits to the given set of locations such that the length of the curvature-constrained path connecting the locations in the sequence is minimal [5]. The curvature-constrained path is requested because of motion constraints of the vehicle that are modeled by Dubins vehicle [6] which moves only forward with the limited turning radius ρ. A solution of the DTSP consists of sequencing part to determine the optimal sequence of visits to the locations and continuous optimization to determine the optimal heading values. The DTSP is at least NP-hard [7] because for $\rho = 0$, it becomes the *Euclidean TSP* (ETSP) [8].

In 1957, Dubins showed that for two given locations with the prescribed leaving and arrival heading angles of the vehicle, the optimal curvature-constrained path connecting the locations is one of six possible maneuvers that can be found by a closed-form solution [6]. The generalization to an arbitrary number of locations is known as the *Dubins Touring Problem* (DTP). The state-of-the-art DTP solvers are based on a sampling of the possible heading angles [9]. In [10], an informed sampling method, which we further denote the *Iteratively-Refined Informed Sampling* (IRIS), enables finding a solution of the DTP close to optimum, i.e., with the relative optimality gap to the lower bound less than 1%, in tens of seconds for problems with up to 100 locations, which outperforms the uniform sampling [9].

The DTSP solvers reported in the literature can be roughly classified into three groups according to [11], [12]. The first group represents *decoupled* approaches, in which the sequence of visits is determined independently from the headings, e.g., using a solution of the related ETSP. Then, headings are determined in the second step using a heuristic Alternating Algorithm (AA) [5], the Local Iterative Optimization (LIO) [12], or by the IRIS proposed in [10]. The methods of the second group involve a *joint* optimization of both the sequence and the headings using Evolutionary Algorithm (EA) as it is reported in [13], [14]. The last group includes approaches based on *transformation* of the original problem into a purely combinatorial routing problem. In [1] and [15], the heading angles are sampled, and the discretized instance is transformed to the Generalized TSP that is further transformed into the Asymmetric TSP that can be solved by the heuristic LKH solver [16], or Concorde solver [17] for the optimal solution with respect to the selected sampling, or other approaches like [18]. A similar sampling-based strategy is utilized to find a tight lower bound of the DTSP in [19] with tens of locations.

The existing *decoupled* approaches such as [9], [10] use only a single sequence found as a solution of the Euclidean TSP without considering the minimum turning radius ρ of the Dubins vehicle which limits the quality of the DTSP solutions for cases where ρ cannot be ignored (i.e., dense location distribution w.r.t. ρ). The *joint optimization* and *transformation* methods may provide better solutions at the cost of low scalability (dense sampling leads to large problem instances of the transformed problems) and very high computational

Manuscript received July 30, 2019; accepted May 21, 2020. The presented work has been supported by the Czech Science Foundation (GA ČR) under research project No. GA19-20238S. The authors acknowledge access to computing infrastructure under the OP VVV funded project CZ.02.1.01/0.0/0.0/16_019/0000765 "Research Center for Informatics" and National Grid Infrastructure MetaCentrum, provided under the program CESNET LM2015042.

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

requirements because a large search-space needs to be covered by the global optimizer such as the evolutionary algorithm.

The herein presented approach is a novel DTSP solver that combines features of both the *decoupled* approaches and *joint optimization*. The main idea of the proposed method is that a relatively simple EA optimizes the sequence of visits separately from the continuous optimization part, and the sequence quality (i.e., fitness) of each individual is assessed by means of length of the corresponding DTP solution (which also determines the heading angles). The information about headings is not stored in the genome itself, but rather computed during every evaluation of its fitness. It reduces the complexity of the search-space and makes it strictly combinatorial as the sequence of visits is a permutation; however, it also makes the evaluation of each individual more expensive. Thus, the critical part of our method is the need to evaluate possibly many candidate sequences. Even though the solution of the DTP by the IRIS [10] is significantly faster than the uniform sampling [9]; it is still too computationally demanding.

Similarly to [20], where the authors speed up the expensive evaluation of the objective function using a surrogate model, we propose the *Windowing Surrogate Model* (WiSM) leveraging on the high-quality solutions of the IRIS-based DTP solver. The main contribution of the proposed WiSM is considered in the windowing decomposition developed from a lower bound which allows training of the model on small fixed-size DTP instances, while the evaluation of the solution length can be performed on DTP instances of arbitrary size. It makes WiSM easily implementable by most types of regression models.

Based on the presented results, WiSM provides solutions competitive to a very dense sampling of the heading values in the DTP, but with significantly lower computational requirements. The paper is mainly focused on the surrogate model based on the windowing decomposition and not on details and tuning of the sequence optimization; however, we show that the combination of WiSM with a simple EA (further denoted as WiSM-EA) outperforms other DTSP solvers for a wide range of instance sizes.

The paper is organized as follows. The addressed DTSP is defined in the next section. A brief overview of the IRIS, the high-quality DTP solver used in learning of the proposed WiSM, is presented in Section III. The model is employed in the Evolutionary Algorithm summarized in Section IV. The surrogate model itself with the windowing decomposition is proposed in Section V. Empirical evaluation of the WiSM-EA and its comparison with existing approaches for the DTSP are in Section VI. Concluding remarks are given in Section VII.

II. Problem Statement

The proposed solution of the DTSP is based on an evaluation of the candidate sequences to visit the given set of n target locations $\mathcal{T} = \{t_1, \dots, t_n\}$, $t_i \in \mathbb{R}^2$ using a solution of the DTP where the path connecting the locations $\mathcal{T}$ has to respect the motion constraints of Dubins vehicle [6] that is moving only forward with a constant forward velocity v and a minimum turning radius ρ. The state of the vehicle can be expressed as $q = (x, y, \theta)$, where $(x, y) \in \mathbb{R}^2$ is the position of the vehicle and $\theta \in \mathbb{S}^1$ is its heading angle. Thus the state q is from the Special Euclidean group, $q \in SE(2)$. The motion of the vehicle can be described as

$$\dot{q} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = v \begin{bmatrix} \cos\theta \\ \sin\theta \\ \frac{u}{\rho} \end{bmatrix}, \quad |u| \leq 1, \tag{1}$$

where u is the bounded control input. For simplicity and w.l.o.g., we further consider $v = 1$ and $\rho = 1$.

A solution of the DTSP can be expressed as a permutation $\Sigma = (\sigma_1, \dots, \sigma_n)$ for $1 \leq \sigma_i \leq n$ that defines a sequence of visits to the locations $\mathcal{T}$ and particular heading values $\Theta = \{\theta_1, \dots, \theta_n\}$ corresponding to each particular location.

The DTSP stands to find a sequence of visits Σ to $\mathcal{T}$ with the respective heading values Θ such that the length $C(\Sigma, \Theta)$ of the Dubins path connecting the locations $\mathcal{T}$ is minimal. The problem can be considered as combinatorial optimization over all possible sequences Σ and n-variable continuous optimization of the heading values Θ.

Problem 1 (Dubins Traveling Salesman Problem - DTSP):

$$\begin{aligned} \text{minimize}_{\,\Sigma,\Theta} \quad & C(\Sigma, \Theta) = \sum_{i=1}^{n} \mathcal{L}(q_{\sigma_i}, q_{\sigma_{i+1}}) \\ \text{s. t.} \quad & \\ & q_i = (t_i, \theta_i),\ q_i \in SE(2),\ t_i \in \mathcal{T}, \\ & \Sigma = (\sigma_1, \dots, \sigma_n),\ \sigma_i \in \{1, \dots, n\}, \\ & \sigma_i \neq \sigma_j \text{ for } i \neq j, \\ & \Theta = \{\theta_1, \dots, \theta_n\}, \end{aligned} \tag{2}$$

where $\mathcal{L}(q_i, q_j)$ denotes the analytically computed length of the optimal Dubins maneuver [6] between q_i and q_j, and $\sigma_i \triangleq \sigma_{i-n}$ for $i > n$ is defined to simplify the notation in (2).

Finding a solution of the DTSP is an optimization problem with a combinatorial part over Σ and continuous part over Θ. If Σ is given, the problem becomes strictly continuous optimization with n variables in Θ. The problem is then called the *Dubins Touring Problem* (DTP), and the solution cost for a particular sequence Σ is denoted $C(\Sigma)$.

Problem 2 (Dubins Touring Problem - DTP):

$$C(\Sigma) = \min_{\Theta} C(\Sigma, \Theta). \tag{3}$$

Due to the combinatorial nature of the TSP, many candidate sequences need to be evaluated. Thus, the evaluation of (3) should be fast enough to provide competitive results to existing DTSP approaches [11]. In Section V, we propose a surrogate approximator of (3) trained using high-quality solutions of the DTP found by the IRIS method [10] briefly described in the following section.

III. Background – Dubins Touring Problem (DTP)

A solution of the DTP is needed to evaluate the cost (3), and thus the final solution of the DTSP, but the solution is also needed to train the proposed surrogate model for fast evaluation of possible candidate sequences. Finding optimal heading angles for the DTP with dense locations is a challenging problem because the length of the Dubins tour connecting a sequence of locations is a piecewise continuous

function [21]. Sampling-based approaches are thus utilized to sample the domains of the continuous variables into finite sets of discrete values of possible heading angles. A solution can be then found as the shortest path in a graph representing the discretized instance of the DTP as follows.

Let have k samples of the heading angle per each location $t_i \in \mathcal{T}$ which form k possible states per each t_i. Then each pair of the states corresponding to two consecutive locations in the sequence can be connected by the optimal curvature-constrained path determined as the optimal Dubins maneuver [6]. Having the states and their connections, a search graph can be created, and the shortest path for the given sequence of n locations and defined sampling k can be found in $O(nk^3)$ by a forward search based on dynamic programming [10]. Such a path is a feasible solution to the DTP.

High-quality solutions, however, require dense sampling [9], which can be computationally demanding. The IRIS method [10] decreases the computational requirements by an iterative refinement of the heading intervals based on the tight lower bound solution from [9].

The DTSP solver presented in this paper requires a high number of sequence evaluations, which makes even the IRIS approach [10] too computationally demanding. Therefore, we propose to employ a surrogate model for fast evaluation of Dubins tour lengths to speed up the convergence towards high-quality DTSP solutions using generated sequences by the Evolutionary Algorithm described in the following section.

IV. Evolutionary Algorithm for the DTSP

In this section, we present the Evolutionary Algorithm (EA) for solving the DTSP. Contrary to existing evolutionary approaches to the DTSP, such as [14] where sequences and headings are encoded together, the proposed evolutionary approach is utilized only for generating proper sequences Σ, and the heading values are determined as a solution of the corresponding DTP. The particular cost function $C(\Sigma)$ can be determined by IRIS for the DTP [10], but also by its surrogate model estimation proposed in Section V. A generational EA with a simple elitism, where an individual of the population is represented by a permutation of n target locations (path representation [22]) is used, and it is summarized in Algorithm 1.

The initial population is filled by random permutations in the `initializePopulation` function representing sequences of visits to the given locations. The fitness of each individual in the population is evaluated using the solution (or its estimation) of the corresponding DTP by the `evaluatePopulation` function. The main loop (Line 3, Algorithm 1) iterates over the generations until the overall running time T_{CPU} reaches the termination time T_{S}.

The evolution is performed as follows. A new population of N individuals is generated either by `mutation` with the probability of p_m, or `crossover` otherwise (Lines 4–12, Algorithm 1). The `mutation` operator is the *Simple Inversion Mutation* (SIM) [23], [22] which reverses a random sub-sequence. The `crossover` method implements the well-known Order crossover (OX) [24] which copies a random sub-sequence of the first parent and then adds locations from the other parent preserving their order but leaving out those already used. We specifically use the OX1 variety as described in [22], although we generate only a single offspring. Both `mutation` and `crossover` employ tournament selection with the tournament size t (`tournamentSelection`) to select the parent(s).

Algorithm 1: Evolutionary Algorithm for the DTSP

Input : $\mathcal{T}$ – a set of the given n target locations; N – population size; t – tournament size; p_m – mutation probability; e – elite size; T_{S} – termination time;
Output: DTSP solution – (Σ, Θ, $C(\Sigma)$)

```
1  P ← initializePopulation(N, n);
2  evaluatePopulation(P, T);
3  while T_CPU < T_S do
4      P_new ← {};
5      for i ← 1 to N do
6          a_1 ← tournamentSelection(P, t);
7          if random() < p_m then
8              c ← mutation(a_1);
9          else
10             a_2 ← tournamentSelection(P, t);
11             c ← crossover(a_1, a_2);
12         P_new ← P_new ∪ {c};
13     evaluatePopulation(P_new, T);
14     b ← best(P ∪ P_new);
15     P ← {};
16     for i ← 1 to e do P ← P ∪ {b} ;
17     B ← sort(P_new);
18     for i ← 1 to N − e do P ← P ∪ {B[i]} ;
19 Σ ← b;
20 (Θ, C(Σ)) ← DTP(Σ);
21 return (Σ, Θ, C(Σ));
```

The new population is then evaluated (Line 13, Algorithm 1) and the best solution of the both current P and new P_{new} populations is selected as b. Finally, e least viable individuals of P_{new} are replaced by the copies of the best so far solution b introducing the elitism (Lines 16–18, Algorithm 1).

The algorithm terminates when the stop condition is met. The best sequence Σ is extracted from the population (Line 19, Algorithm 1) and the corresponding headings Θ are determined (Line 20, Algorithm 1) calling the DTP solver [10] to provide a feasible solution. On the other hand, sampling-based approaches would be too computationally demanding for evaluation of $C(\Sigma)$ of all population individuals in `evaluatePopulation`, and therefore, we propose a surrogate model to get a considerable speedup.

V. Proposed Windowing Surrogate Model (WiSM) for a Fast Estimation of the DTP Solution Cost

The proposed *Windowing Surrogate Model* (WiSM) is designed to approximate $C(\Sigma)$ with a surrogate function to significantly speed up the evaluation of the DTP instance costs. The WiSM is evaluated based on overlapping windows in a

convolutional manner, and the partial results are averaged to get the final cost estimation.

In the studied DTSP, the number of locations n can vary depending on the particular problem instance. Thus, we need to address the limitation of the regression models (e.g., neural networks, random forests, etc.) that are mostly limited to fixed sized inputs. While this issue can be directly approached by recurrent neural networks that allow processing inputs of variable-length as sequences (e.g., LSTM [25] or GRU [26]), a collection of training data would be non-trivial as we would need to collect examples of DTP instances having a varying number of locations with a little guarantee on how the network would generalize on sequences of the unseen lengths. To overcome the problem of the variable-size input, we propose a decomposition of the cost function $C(\Sigma)$ to fixed-size subproblems based on the sliding window technique.

A. Sliding Windowing based Cost Estimation of the DTP

The proposed idea is to evaluate the cost of Dubins tour for small sub-sequences of the target locations limited by the window size w. The total tour cost can be then estimated as an aggregate of the particular costs.

For a specific sequence Σ we can define the particular cost $C^*_{w,i}(\Sigma)$ of the subtour of Dubins optimal tour for the fixed w-size window as

$$C^*_{w,i}(\Sigma) = \sum_{j=i}^{i+w-1} \mathcal{L}(q^*_{\sigma_j}, q^*_{\sigma_{j+1}}), \quad i \in \{1, \dots, n\}. \tag{4}$$

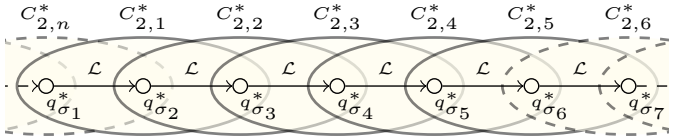


Fig. 1. Sliding window principle with overlapping windows of the size $w = 2$. Each window covers 2 trips, i.e., 3 locations.

We propose to utilize a sliding window that starts at each target location i as it enables to address arbitrarily-length instances for which the number of locations n is not necessarily divisible by the utilized window size w. The whole Dubins tour is divided into n sub-sequences of the windows $(\sigma_i, \dots, \sigma_{i+w})$ for $i \in \{1, \dots, n\}$ each connecting $w+1$ successive locations as it is depicted in Fig. 1. Then the optimal cost $C^*(\Sigma)$ for the specific Σ and known optimal heading angles Θ^* can be computed from the window costs exactly as

$$C^*(\Sigma) = \frac{1}{w} \sum_{i=1}^{n} C^*_{w,i}(\Sigma). \tag{5}$$

The optimal heading angles Θ^* are apparently not available, and thus (5) provides only an intuition behind the proposed approach for fast cost estimation of the Dubins tour.

The cost of the specific window is estimated independently on the final solution and finding the cost of the window subsequence is considered as the *Open DTP* [10], which can be defined as the continuous optimization problem

$$\overline{C}(\Sigma) = \min_{\Theta} \overline{C}(\Sigma, \Theta). \tag{6}$$

The Open DTP is further denoted as $\overline{\text{DTP}}$ to emphasize it does not involve a closed tour and the heading angles at both end locations are unconstrained. Note that the unlike in (2), the length of the closing Dubins maneuver $\mathcal{L}(q_{\sigma_n}, q_{\sigma_1})$ is omitted from the $\overline{C}(\Sigma, \Theta)$ definition:

$$\overline{C}(\Sigma, \Theta) = \sum_{i=1}^{n-1} \mathcal{L}(q_{\sigma_i}, q_{\sigma_{i+1}}). \tag{7}$$

Comparing the particular DTP cost $C^*_{w,i}(\Sigma)$ of the i-th window defined by a sub-sequence $\Sigma_{w,i} \triangleq (\sigma_i, \dots, \sigma_{i+w})$ and the corresponding $\overline{\text{DTP}}$ cost $\overline{C}(\Sigma_{w,i})$, it can be shown that

$$\overline{C}(\Sigma_{w,i}) \leq C^*_{w,i}(\Sigma). \tag{8}$$

The proof by contradiction is based on the fact that $\overline{C}(\Sigma_{w,i})$ is a relaxed version of $C^*_{w,i}(\Sigma)$. More specifically, unlike for $C^*_{w,i}(\Sigma)$, the boundary angles θ^*_i and θ^*_{i+w} of $\overline{C}(\Sigma_{w,i})$ are not constrained. Thus $(q_{\sigma_1}, q_{\sigma_2})$ and $(q_{\sigma_{n-1}}, q_{\sigma_n})$ can take advantage of arbitrarily chosen heading angles at the endpoints.

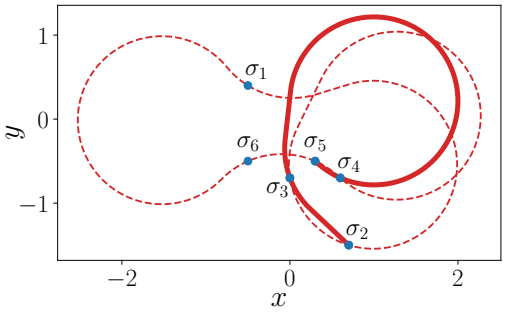


Fig. 2. A solution of the closed DTP (dashed line) with one of n open DTP subtours $\Sigma_{3,2}$ (thick solid line).

An example of the Closed DTP with $n = 6$ locations with a corresponding instance of the Open DTP over four locations $\sigma_2, \sigma_3, \sigma_4$, and σ_5 is depicted in Fig. 2. Note that the optimal tours between σ_2 and σ_5 differ for the open and closed cases. The open version is shorter which corresponds with (8).

The cost of the whole sequence Σ can be estimated as $\tilde{C}_w(\Sigma)$ using n windows with the size w as

$$\tilde{C}_w(\Sigma) = \frac{1}{w} \sum_{i=1}^{n} \overline{C}(\Sigma_{w,i}). \tag{9}$$

Based on (8), the estimation $\tilde{C}_w(\Sigma)$ is a lower bound of the optimal cost $C^*(\Sigma)$ and we get

$$\tilde{C}_w(\Sigma) \leq C^*(\Sigma). \tag{10}$$

In the following, we expect the lower bound (9) is tight enough even for relatively small size w and the minimization w.r.t. $\tilde{C}_w(\Sigma)$ can be employed as a proxy to the direct minimization of $C(\Sigma)$. Thus, we can evaluate DTP costs for instances of a fixed size $w + 1$ and use (9) to get the cost estimate for an instance of arbitrary size n.

The computational complexity of $\tilde{C}_w(\Sigma)$ approximation employing the sampling methods such as IRIS [10] can be bounded by $O(nwk^2)$, where k is the number of heading values per each location. In practice, the window size w is

expected to be constant and significantly lower than both k and n ($w \ll k$ and $w \ll n$). Thus, the computational complexity can be bounded by $O(nk^2)$, which is a noticeable improvement in comparison to $O(nk^3)$ for the evaluation of $C(\Sigma)$. While the performance improvement is appealing, it turns out, that in practice, the sampling methods are still too slow to allow for efficient evolutionary search. Nevertheless, the decomposition, based on a repeated evaluation of fixed-sized sub-tours, enables approximation of the relatively slow sampling-based methods by a wide selection of regression models.

The utilized sliding window approach is a commonly used general technique to process variable length data; however, to the best of the authors' knowledge, it has not been used for tour length estimates yet. In the particular case of the proposed WiSM, the specific combination of the sliding window with the selected aggregation scheme gave a clear interpretation (10), which is only possible by incorporating domain-specific knowledge of the addressed DTSP using results of the DTP.

B. Windowing Surrogate Model (WiSM) Approximation of the Cost Estimation $\tilde{C}_w(\Sigma)$

The key idea of the proposed WiSM-based approximation of the DTP solution cost is based on a fast evaluation of the partial costs of $\overline{C}(\Sigma_{w,i})$ by a surrogate regression model. In existing deployments of the surrogate models in evolutionary algorithms, the model is a proxy to the *real fitness* that is trained online while EA runs. The switching between the real and surrogate fitness can be realized by more or less complex *model management* [27], [28]. On the other hand, the real-time response of the proposed DTSP solver can benefit from pre-trained surrogate model [29], and therefore, the model is learned offline using a large training dataset based on high-quality solutions of the open-loop $\overline{\text{DTP}}$.

The WiSM can be implemented using virtually any type of regression model, but we consider the Multi-Layer Perceptron (MLP) [30] in this paper. It is because the feed-forward neural network supports fast evaluation of all partial costs $\overline{C}(\Sigma_{w,i})$ in a single batch. Thus, it can take advantage of highly optimized matrix multiplication routines[1].

Each input vector fed to the surrogate model is constructed as $(x_{\sigma_i}, y_{\sigma_i}, \ldots, x_{\sigma_{i+w}}, y_{\sigma_{i+w}}) \in \mathbb{R}^{2(w+1)}$ while the output constitutes single real value corresponding to $\overline{C}(\Sigma_{w,i})$. Notice, the number of locations is $w+1$ because the cost of Dubins tour is defined for two locations and more, see (4). Hence, the MLP architecture has $2(w+1)$ inputs, several hidden layers of non-linear units such as ReLU [31], and a single linear output neuron which is common for regression tasks.

VI. Results

The proposed Windowing Surrogate Model (WiSM) has been evaluated in combination with the evolutionary algorithm presented in Section IV and the combined method is further denoted WiSM-EA. The solved DTSP instances are of various sizes and different densities of the locations.

[1]We can evaluate the whole population of the evolutionary algorithm in a single batch, which has been utilized for the herein reported empirical results.

The performance of the WiSM-EA is compared with the existing DTSP approaches. Namely, the Alternating Algorithm (AA) [5], the Local Iterative Optimization (LIO) [12], and the Sampling-based Algorithm (SA) [32]. The SA transforms the problem to the Generalized TSP that is further transformed into Asymmetric TSP solved by the LKH solver [16].

As a *baseline*, we take the decoupled approach from [10] where the DTSP solution takes the location sequence Σ from the solution of the corresponding Euclidean TSP (obtained by the Concorde solver [17]). The headings Θ are consequently determined by the IRIS method. Besides, we also initially considered the Memetic algorithm [14]; however, the achieved results are not competitive with the selected approaches regarding the solution quality and computational requirements. Therefore it is not included in the reported results.

In addition to the baseline, we also computed a lower bound [19] that provides an estimate of the gap to the optimal solution. Such a tight enough lower bound is computationally very demanding and intractable for instances with hundreds of locations. Therefore, the prohibitively demanding lower bound cannot be used as a full reference, and it is thus reported only for small instances with $n = 25$. The value of the determined lower bound is shown as LB in Fig. 5a for the heading resolution within $\{4, 8, 16, 32, 64\}$, where it can be observed, the LB convergences to the solution provided by the proposed method.

The DTSP methods were compared using instances, randomly generated similarly to [10]. The benchmark set consists of 10 instances per each possible pair (n, d), where $n \in \{25, 100, 500\}$ denotes the number of target locations and $d \in \{0.1, 1, 10\}$ is the density of the locations. The targets $\mathcal{T}$ of each instance were uniformly sampled from a square region with a side $b = \sqrt{n/d}$, i.e., $t_i \in \left[-\frac{1}{2}b, \frac{1}{2}b\right] \times \left[-\frac{1}{2}b, \frac{1}{2}b\right]$.

Due to unknown optimal solutions of the DTSP instances, the results are evaluated using the *normalized cost* defined as

$$C_r = \frac{C(\Sigma, \Theta)}{C(\Sigma_{\text{base}}, \Theta_{\text{base}})}, \tag{11}$$

where $C(\Sigma_{\text{base}}, \Theta_{\text{base}})$ is the cost provided by the baseline.

The headings Θ for the baseline and also for the WiSM-EA (Line 20, Algorithm 1) were determined using IRIS [10] with the maximum number of samples set to $k_{\max} = 1024$.

The parameters used by the evolutionary algorithm were: the population size $N = 100$, tournament size $t = 3$, mutation probability $p_m = 0.8$, and elitist size $e = 20$.

The WiSM-EA was implemented in Julia language [33], run with `-O3` and `--check-bounds=no` options and the computational requirements were further significantly decreased by caching model approximations $\overline{C}(\Sigma_{w,i})$ using $\Sigma_{w,i}$ as a key. The IRIS method (called from Line 20, Algorithm 1) and also all the other algorithms (AA, LIO, SA) were implemented in C++ and compiled by the gcc compiler with `-O3` and `-march=native` flags, which also holds for the LKH. In all the cases, the algorithms were run on a single core of the Intel Xeon Gold 6130 @ 2.10 GHz processor.

A. Learning Setup of the Proposed WiSM

The choice of an appropriate window size w of the WiSM deals is a trade-off between the model complexity, accuracy, and computational time needed to evaluate the model. Increased window size leads to the increased number of the model inputs and parameters that also increases demands on the size of the training dataset. Besides, a prediction for a complex model is computationally demanding, which might be critical for the evolutionary search because it can limit the total number of evaluations achievable under the particular computational time limit T_{S} (Line 20, Algorithm 3).

Empirical evaluation has been performed for $2 \leq w \leq 10$ in the same way as described in this section, but data are not shown for brevity. Based on that, we found out that $w = 3$ provides overall best results, and therefore, this window size is considered for the rest of the presented results. The achieved average computational time per a single window for $w = 3$ is $0.2\,\mu$s and $3.3\,\mu$s with and without the caching, respectively.

The empirically selected neural network architecture was the MLP as described in Section V-B having three hidden layers of 256 ReLU units each. The training dataset consisted of 64×10^6 samples of $2(w + 1) = 8$ coordinates and the corresponding target cost value was computed using a solution of the open DTP found by the IRIS method with $k_{\max} = 1024$. The individual coordinates of the learning datasets were independently drawn from the normal distribution $\mathcal{N}(0, 1)$ which introduced a small fraction of longer distances between the target locations improving generalization overall considered densities d of the benchmark instances[2]. The distribution was also selected empirically, where the process was bootstrapped by observing the distributions of locations in sub tours of the random DTSP instances. We found WiSM-EA to be reasonably robust w.r.t. to the training distributions; nevertheless, future research should focus on training data generation methods.

The model training was done as follows. We split the dataset to training part (80%) and the validation part (20%). The loss function to train the surrogate model was the Mean Squared Error (MSE), which is a common choice for the regression. No regularization method such as L1, L2, or drop out [34] was used as overfitting was not a problem due to the size of the training data. We employed Adam [35] using recommended parameter values (the learning rate 0.001, $\beta_1 = 0.9$ and $\beta_2 = 0.999$). The training was terminated using *early stopping* when there was no improvement in the validation loss for ten successive epochs. The single WiSM employed for all (n, d) pairs in the following experiments needed 134 epochs to converge, achieving MSE of 1.94×10^{-2} on the validation set[3].

B. WiSM-EA Performance Evaluation

The evaluation results of the proposed WiSM-EA and its comparison with the other DTSP solvers is depicted in Figures 3, 4 and 5. Each data point represents the mean value of the real required computational time and the mean value of the corresponding *normalized cost* C_r computed over 100 runs of the particular algorithm, i.e., ten runs per each of the ten random instances for each problem setup (n, d). The boxes in the presented plots delimit lower and upper quartiles. Note that, in general, C_r increases with the decreasing density of d as the problem becomes more similar to its underlying Euclidean TSP.

The WiSM and SA methods were run using multiple settings in order to study the tradeoff between the solution quality and computational requirements. In particular, for the WiSM-EA, we used eight values of the time limit T_{S} (evaluated by the `stopCondition(`T_{S}`)` method) ranging from 1 second to 600 seconds, denoting the algorithm configuration as WiSM-EA$_{T_{\mathrm{S}}}$. The total required computational time $T_{\mathtt{CPU}} > T_{\mathrm{S}}$, as in its final stage WiSM-EA calls IRIS to compute the headings Θ for the sequence Σ found by the evolutionary search (Line 20, Algorithm 1). The SA was run for four sample sizes $k \in \{4, 8, 16, 32\}$ which are referred to as SA$_k$.

The benefit of the surrogate model is supported by a comparison with two more methods denoted the WIRIS-EA and IRIS-EA. The WIRIS-EA is the WISM-EA (including the caching) with a sole exception of using IRIS instead of its neural network surrogate model to compute the partial costs $\overline{C}(\Sigma_{w,i})$. The IRIS-EA simply combines the evolutionary algorithm presented in Section IV with IRIS approximation of the closed DTP of the complete sequence Σ, i.e., $C^*(\Sigma)$.

Due to computational requirements of IRIS, its sampling precision was reduced to $k_{\max} = 16$ for both WIRIS-EA and IRIS-EA. However, in many cases, we were unable to get results competitive to other methods for lower values of the T_{S}. Nevertheless, the value of $k_{\max}$ was selected empirically based on the overall best tradeoff between the DTP length approximation precision and the number of evaluations achievable in T_{S}. Notice that the final IRIS call (Line 20, Algorithm 1) was still executed using $k_{\max} = 1024$ to show impact of the found sequences by the EA.

Both the methods significantly underperform the WiSM-EA with the windowing WIRIS-EA being a better alternative. The reason is evident from aggregated results over all densities for $n = 100$, where the WiSM-EA achieves a significantly higher rate of 75.4×10^3 evaluations of the DTP solution cost per second in comparison to much slower WIRIS-EA and IRIS-EA with 221 and 7.84 evaluations per second, respectively. Examples of the found solutions are shown in Fig. 6 and a detailed report of the evaluation follows.

The achieved results for the DTSP instances with the density $d = 1$ and $n = 100$ locations are shown in Fig. 3, where the Pareto front is formed by three methods only: the AA, baseline, and the proposed WiSM-EA. Taking only the shortest $T_{\mathrm{S}} = 1\,\mathrm{s}$ limit into consideration, all three methods give results in units of seconds. In particular, the AA in $T_{\mathtt{CPU}} = 1.2\,\mathrm{s}$, the baseline in $T_{\mathtt{CPU}} = 1.3\,\mathrm{s}$, and the WiSM-EA in $T_{\mathtt{CPU}} = 2.6\,\mathrm{s}$. The WiSM-EA achieves a significantly better $C_r = 0.61$ than the AA with $C_r = 1.3$ and the baseline with $C_r = 1.0$. Besides, the normalized cost further improves to $C_r = 0.51$ in approximately ten minutes (WiSM-EA$_{600}$). The only rival

[2]It would take roughly 21 days to generate the full dataset for $w = 3$ on a single core, and thus 64 cores were used to speed up the process.

[3]The target values were standardized as usual when training neural networks.

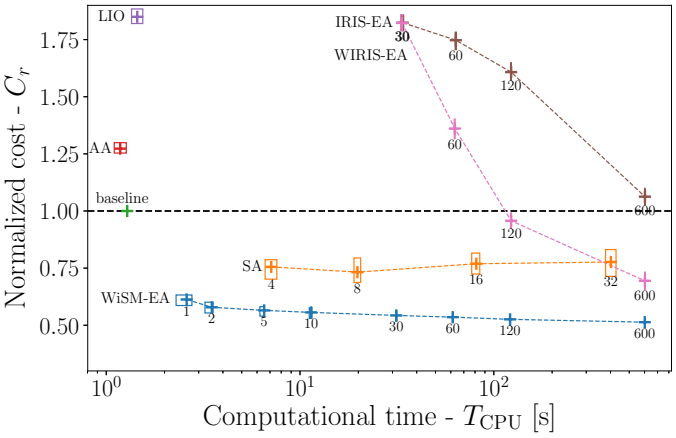


Fig. 3. Normalized cost C_r and real computational requirements for DTSP instances with the density $d = 1.0$ and $n = 100$ target location. Several values for the WiSM-EA indicate different time limit and for SA the particular number of the samples. Note the logarithmic scale of the time axis.

in the solution cost is the SA method with the best value $C_r = 0.73$ achieved by the SA_8 in $T_{\mathrm{CPU}} = 20\,\mathrm{s}$. The solution cost provided by the SA should theoretically decrease with the increasing number of samples; however, the observed behavior is different, and the opposite trend can be noticed for the SA_{16} and SA_{32}. This behavior can be explained by too large instances of the underlying Generalized TSP and heuristic solution of the transformed TSP instance. Similar behavior was observed for all tested instances having $n \geq 100$.

Evaluation results for the DTSP instances with $n = 100$ locations and high and low densities are depicted in Fig. 4. For the high density instances with $d = 10$, the results are similar to $d = 1$ (compare Fig. 3 and Fig. 4a). The AA reaches the mean $C_r = 1.08$, while the proposed WiSM-EA achieves significantly better results ranging from $C_r = 0.5$ ($\mathrm{WiSM\text{-}EA}_1$) to $C_r = 0.4$ ($\mathrm{WiSM\text{-}EA}_{600}$). LIO does not perform well, while the SA provides competitive results for the time limit around 10 s.

For the low-density instances with $d = 0.1$, the DTSP becomes closer to the corresponding Euclidean TSP because locations are relatively far from each other and the minimum turning radius constraint does not significantly affect the length of the final Dubins tour. It is also indicated by the results for the baseline with the sequence determined by the ETSP solver. The proposed WiSM-EA outperforms all other solvers, but the baseline is outperformed only for the time limits $T_{\mathrm{S}} \geq 30\,\mathrm{s}$. Based on that, we employed a solution of the corresponding Euclidean TSP provided by the Concorde solver [17] in the initialization of the whole initial generation in the `initializePopulation` method (Line 1, Algorithm 1) of the proposed EA instead of a random permutation to improve the performance. The modified algorithm is denoted iWiSM-EA in Fig. 4b. Although the iWiSM-EA provides outstanding performance in low-density instances, it performs similarly to the WiSM-EA for $d = 1$ and $d = 10$. For large instances, we observed slightly higher final costs, which can be explained by the zero diversity of the initial population making the optimization prone to local minima (data not shown for readability). We conclude that the initialization of the iWiSM-EA approach is beneficial for low-density instances only.

Computational performance for instances with small ($n = 25$) and high ($n = 500$) number of locations is shown in Fig. 5. For small instances, the Pareto front is formed by the AA, SA, and WiSM-EA solvers. Note that WIRIS-EA achieves the best result for $T_{\mathrm{S}} = 30\,\mathrm{s}$ while its performance deteriorates for longer time limits which can be explained by the reduced precision using $k_{\max} = 16$. Solutions found by the $\mathrm{WiSM\text{-}EA}_1$ have the mean cost $C_r = 0.59$ that is significantly lower than for the AA with $C_r = 1.32$, the SA_4 with $C_r = 0.8$, but also the SA_8 with $C_r = 0.7$ with the mean computational time $T_{\mathrm{CPU}} = 1.3\,\mathrm{s}$. The AA is the fastest approach with the mean $T_{\mathrm{CPU}} = 0.06\,\mathrm{s}$ and the SA_4 needs $T_{\mathrm{CPU}} = 0.4\,\mathrm{s}$. For large instances with $n = 500$, the proposed WiSM-EA dominates the other methods. The $\mathrm{WiSM\text{-}EA}_{10}$ provides the mean $C_r = 0.89$ in $T_{\mathrm{CPU}} = 21\,\mathrm{s}$ and reaches $C_r = 0.64$ in approximately ten minutes.

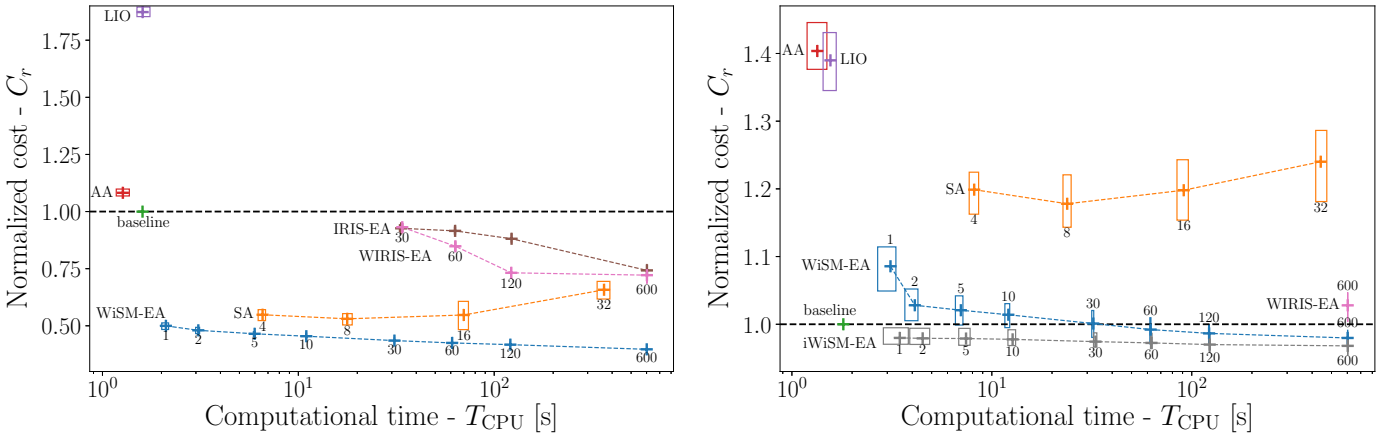
C. Discussion and Possible Future Work

Based on the reported results, the proposed approximator WiSM is a vital approach that enables the solution of the DTSP instances by a relatively simple evolutionary algorithm. The developed WiSM-EA scales significantly better than the other evaluated methods in both the problem size n and density d. For large instances, it dominates the other methods in the computational requirements and the quality of the found solutions. From a practical point of view, the WiSM-EA gives the best results in units of seconds for small and medium-sized instances, while for the large instances with hundreds of locations, it needs only low tens of seconds. To improve the rate of convergence for low-density instances of the DTSP, we suggest using the ETSP initialization of the WiSM-EA.

Regarding the other methods, the AA is the fastest algorithm, but it always provides $C_r > 1$ and thus worse solutions than the baseline. The SA provides competitive results to the proposed WiSM-EA in medium and high-density instances, but it does not scale with the problem size and the number of the utilized samples, mostly because of the limitations of the underlying solver to the transformed Generalized TSP.

Although it is not the aim of this paper, our initial experiments indicated (data not shown), that WiSM is robust w.r.t. different neural network architectures. Nevertheless, future research should focus on finding possibly more effective regressors. The similar applies to approaches generating representative training data for WiSMs learning. We expect that the convergence might be further improved with techniques such as trainable genetic operators in [36].

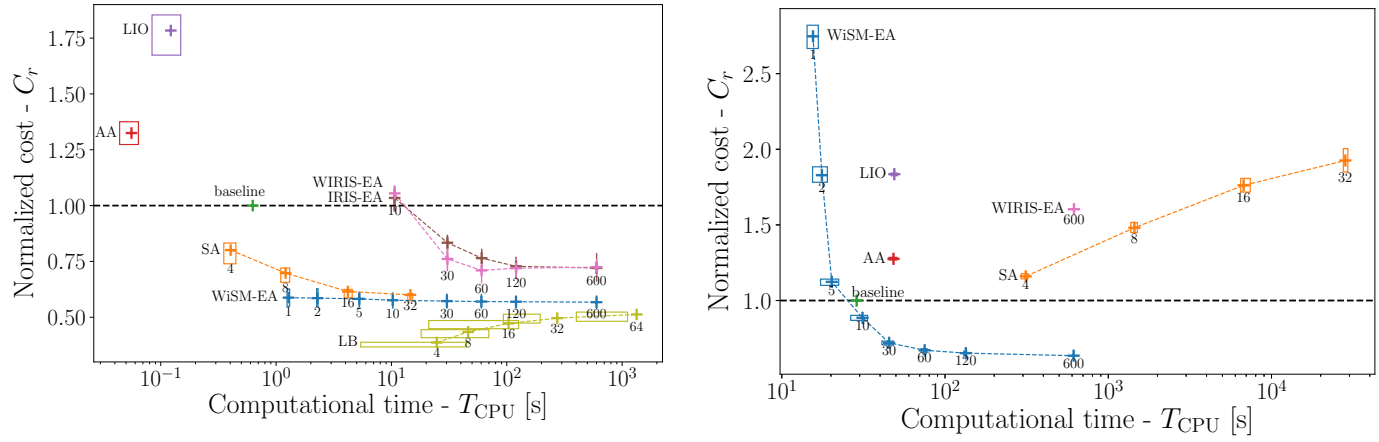
Regarding the future deployments of the proposed surrogate approximator of the Dubins tour costs, we believe it can also be utilized in other Dubins routing problems such as the Dubins Orienteering Problem [37] where many sequences have to be evaluated. Moreover, the model can be generalized for touring problems with neighborhood areas instead of single locations, where the recently introduced Generalized Dubins Interval Problem [38] can be utilized for the model learning using high-quality solutions, and then for solving the Dubins TSP with Neighborhoods [2]. Finally, the herein presented



(a) high density instances, $d = 10.0$, $n = 100$

(b) low density instances, $d = 0.1$, $n = 100$

Fig. 4. Performance of the evaluated DTSP solvers in high ($d = 10$) and low ($d = 0.1$) density instances with $n = 100$ target locations. The achieved normalized cost C_r is shown according to the required computational time or a given time limit (for the WiSM-EA). For $d = 0.1$, the iWiSM-EA refers to the proposed WiSM-EA with the initialization of the population by the solution of the Euclidean TSP provided by the Concorde solver.



(a) small instances $n = 25$, $d = 1.0$

(b) large instances $n = 500$, $d = 1.0$

Fig. 5. Performance of the evaluated DTSP solvers in small ($n = 25$) and large ($n = 500$) instances with the density $d = 1.0$. For the small instances with $n = 25$, LB is the lower bound solution [19] computed with resolutions of 4, 8, 16, 32, and 64 heading samples.

results motivate for future work on the approximation of costs of curvature-constrained tours for more complex vehicle models than Dubins vehicle, e.g., Bézier curves [39].

VII. Conclusion

We present a novel approach to solve the Dubins Traveling Salesman Problem (DTSP) by a relatively simple Evolutionary Algorithm that is based on the proposed surrogate approximator of the Dubins tour cost called WiSM. Even though collecting enough training data might take considerable time, once the dataset is built and the WiSM is trained, it can provide Dubins tour cost estimates in a very fast rate, which can be exploited by robust global search methods such as Evolutionary Algorithms. The developed WiSM-EA has been evaluated on DTSP instances of varying size and also the density of the locations to be visited. Based on the reported results, the WiSM-EA outperforms the existing state-of-the-art approaches in the quality of the found solutions and also computational requirements. The results demonstrate the proposed method scales with the problem size and density, and thus, it is a suitable heuristic for finding high-quality solutions of curvature-constrained routing problems with Dubins vehicle.

References

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

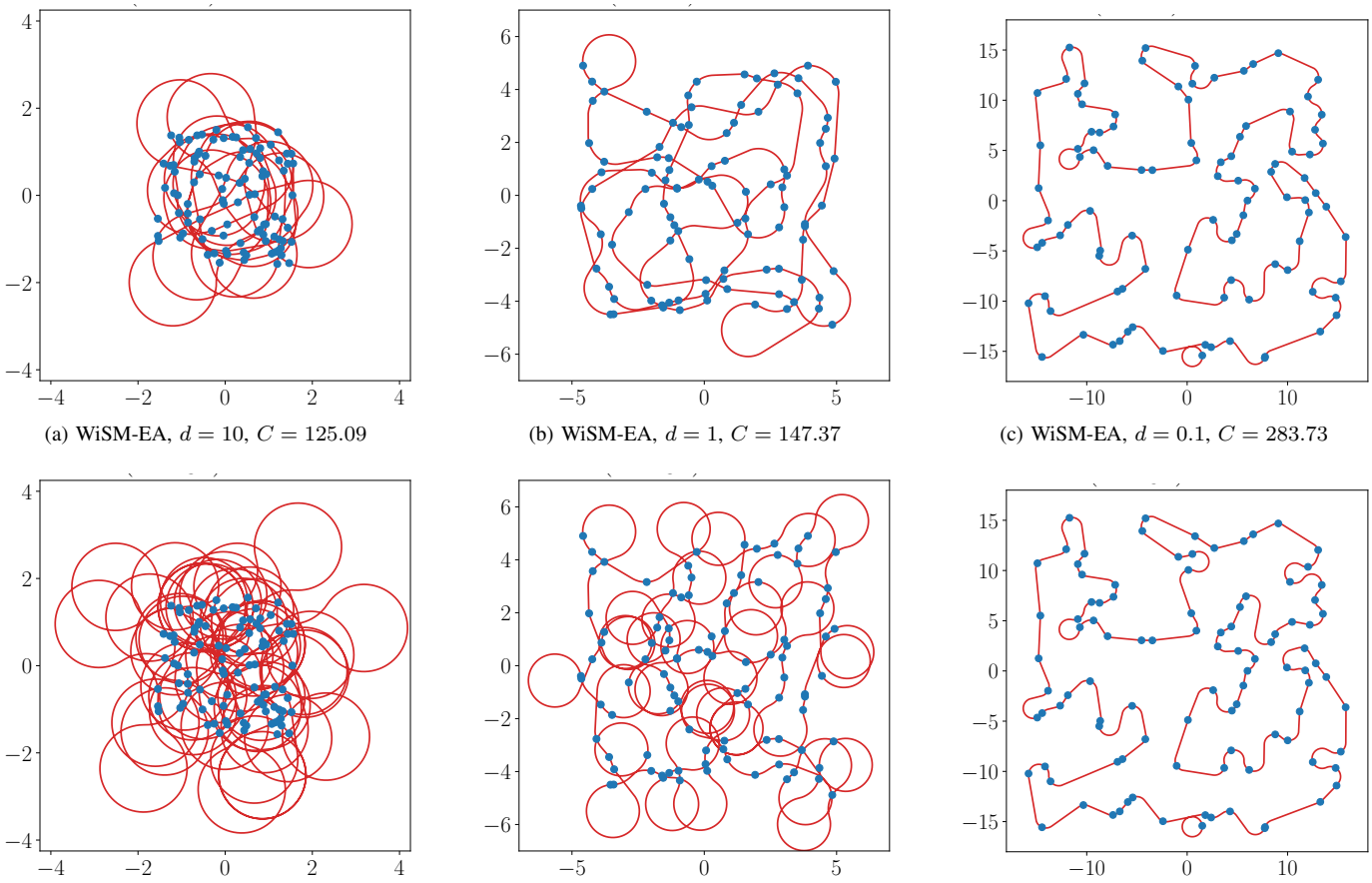
[2] J. Faigl, P. Váňa, R. Pěnička, and M. Saska, "Unsupervised learning-based flexible framework for surveillance planning with aerial vehicles," *Journal of Field Robotics*, vol. 36, no. 1, pp. 270–301, 2019.

[3] M. Niendorf, P. T. Kabamba, and A. R. Girard, "Stability of Solutions to Classes of Traveling Salesman Problems," *IEEE Transactions on Cybernetics*, vol. 46, no. 4, pp. 973–985, Apr. 2016.

[4] M. Niendorf and A. R. Girard, "Exact and Approximate Stability of Solutions to Traveling Salesman Problems," *IEEE Transactions on Cybernetics*, vol. 48, no. 2, pp. 583–595, Feb. 2018.

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

[6] 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.



(a) WiSM-EA, $d = 10$, $C = 125.09$ (b) WiSM-EA, $d = 1$, $C = 147.37$ (c) WiSM-EA, $d = 0.1$, $C = 283.73$

(d) Baseline, $d = 10$, $C = 321.32$ (e) Baseline, $d = 1$, $C = 280.28$ (f) Baseline, $d = 0.1$, $C = 289.35$

Fig. 6. Solutions of the DTSP with $n = 100$ target locations and density d found by the proposed WiSM-EA (upper) and baseline (bottom). The target locations are depicted as small blue disks, and the found Dubins tour is in the red. Each column represents the same single instance, and thus the solutions can be directly compared. The final cost C is computed using the IRIS with the maximum resolution $k_{\max} = 1024$.

[7] J. Le Ny, E. Feron, and E. Frazzoli, "On the Dubins Traveling Salesman Problem," *IEEE Transactions on Automatic Control*, vol. 57, no. 1, pp. 265–270, 2012.

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

[9] S. G. Manyam, S. Rathinam, D. Casbeer, and E. Garcia, "Tightly Bounding the Shortest Dubins Paths Through a Sequence of Points," *Journal of Intelligent & Robotic Systems*, pp. 1–17, 2017.

[10] J. Faigl, P. Váňa, M. Saska, T. Báča, and V. Spurný, "On solution of the dubins touring problem," in *European Conference on Mobile Robots (ECMR)*, 2017, pp. 1–6.

[11] D. G. Macharet and M. F. M. Campos, "A survey on routing problems and robotic systems," *Robotica*, vol. 36, no. 12, pp. 1781–1803, 2018.

[12] P. Váňa and J. Faigl, "On the dubins traveling salesman problem with neighborhoods," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 4029–4034.

[13] D. Guimaraes Macharet, A. Alves Neto, V. Fiuza da Camara Neto, and M. Montenegro Campos, "An evolutionary approach for the Dubins' traveling salesman problem with neighborhoods," in *Annual Conference on Genetic and Evolutionary Computation*. ACM, 2012, pp. 377–384.

[14] 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.

[15] K. J. Obermeyer, P. Oberlin, and S. Darbha, "Sampling-based path planning for a visual reconnaissance unmanned air vehicle," *Journal of Guidance, Control, and Dynamics*, vol. 35, no. 2, pp. 619–631, 2012.

[16] K. Helsgaun, "LKH solver 2.0.9," 2018, [cited 23 Dec 2019]. [Online]. Available: http://www.akira.ruc.dk/~keld/research/LKH/

[17] D. Applegate, R. Bixby, V. Chvátal, and W. Cook, "Concorde TSP Solver," 2003, [cited 22 Aug 2019]. [Online]. Available: http://www.tsp.gatech.edu/concorde.html

[18] J. Shi, J. Sun, Q. Zhang, and K. Ye, "Homotopic convex transformation: A new landscape smoothing method for the traveling salesman problem," *IEEE Transactions on Cybernetics*, 2020.

[19] S. G. Manyam and S. Rathinam, "On tightly bounding the dubins traveling salesman's optimum," *Journal of Dynamic Systems, Measurement, and Control*, vol. 140, no. 7, p. 071013, 2018.

[20] Y. Wang, D. Yin, S. Yang, and G. Sun, "Global and Local Surrogate-Assisted Differential Evolution for Expensive Constrained Optimization Problems With Inequality Constraints," *IEEE Transactions on Cybernetics*, pp. 1–15, 2018.

[21] X. Goaoc, H.-S. Kim, and S. Lazard, "Bounded-curvature shortest paths through a sequence of points using convex optimization," *SIAM Journal on Computing*, vol. 42, no. 2, pp. 662–684, 2013.

[22] P. Larranaga, C. M. H. Kuijpers, R. H. Murga, I. Inza, and S. Dizdarevic, "Genetic algorithms for the travelling salesman problem: A review of representations and operators," *Artificial Intelligence Review*, vol. 13, no. 2, pp. 129–170, 1999.

[23] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. MIT Press, 1992.

[24] L. Davis, "Applying adaptive algorithms to epistatic domains," in *International Joint Conference on Artificial Intelligence (IJCAI)*, vol. 85, 1985, pp. 162–164.

[25] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[26] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *Conference*

on Empirical Methods in Natural Language Processing (EMNLP), 2014, pp. 1724–1734.
[27] Y. Jin, "A comprehensive survey of fitness approximation in evolutionary computation," *Soft Computing*, vol. 9, no. 1, pp. 3–12, Jan 2005.
[28] ——, "Surrogate-assisted evolutionary computation: Recent advances and future challenges," *Swarm and Evolutionary Computation*, vol. 1, no. 2, pp. 61–70, 2011.
[29] S. Horng, S. Lin, L. H. Lee, and C. Chen, "Memetic algorithm for real-time combinatorial stochastic simulation optimization problems with performance analysis," *IEEE Transactions on Cybernetics*, vol. 43, no. 5, pp. 1495–1509, 2013.
[30] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, 2006.
[31] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *International Conference on Artificial Intelligence and Statistics*, 2011, pp. 315–323.
[32] 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.
[33] J. Bezanzon, S. Karpinski, V. Shah, and A. Edelman, "Julia: A fast dynamic language for technical computing," *CoRR*, vol. abs/1209.5145, 2012.
[34] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.
[35] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations (ICLR)*, 2015.
[36] C. He, S. Huang, R. Cheng, K. C. Tan, and Y. Jin, "Evolutionary multiobjective optimization driven by generative adversarial networks (gans)," *IEEE Transactions on Cybernetics*, pp. 1–14, 2020.
[37] R. Pěnička, J. Faigl, P. Váňa, and M. Saska, "Dubins orienteering problem," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 1210–1217, 2017.
[38] P. Váňa and J. Faigl, "Optimal Solution of the Generalized Dubins Interval Problem," in *Robotics: Science and Systems (RSS)*, 2018.
[39] J. Faigl and P. Váňa, "Surveillance planning with Bézier curves," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 750–757, 2018.

Jan Drchal Jan Drchal is an assistant professor at the Faculty of Electrical Engineering (FEE), Czech Technical University in Prague (CTU). He received his Ph.D. (2013) and Ing. (2006) in Electronics, Computer Science and Engineering from CTU. He is head of machine learning applications at the Artificial Intelligence Center (http://aic.fel.cvut.cz).

His research interests include machine learning with a focus on artificial neural networks and neuro-evolutionary approaches. Currently, he mostly deals with applications in robotics, NLP, and transportation, focusing on tasks which have variable-sized input/output representation.

Jan Faigl Jan Faigl is a professor of computer science at the Faculty of Electrical Engineering (FEE), Czech Technical University in Prague (CTU). He received his Ph.D. (2010) in Artificial Intelligence and Biocybernetics and Ing. (2003) in Cybernetics from CTU. In 2013/2014, he was Fulbright visit scholar at the University of Southern California. He has been awarded the Antonin Svoboda Award from the Czech Society for Cybernetics and Informatics in 2011. He served as the guest editor of the special issue on "Online decision making in Multi-Robot Coordination" in Autonomous Robots journal.

He currently serves as the associate editor of the IEEE Transactions on Automation Science and Engineering (T-ASE). Since 2013, he is leading the Computational Robotics Laboratory (http://comrob.fel.cvut.cz) within the Artificial Intelligence Center (http://aic.fel.cvut.cz). He is also co-founder of the Center for Robotics and Autonomous Systems (http://robotics.fel.cvut.cz). His current research interests include combinatorial motion planning, robotic information gathering, long-term autonomous missions with online incremental learning, autonomous navigation, aerial systems, and multi-legged locomotion control.

Petr Váňa Petr Váňa is a Ph.D. candidate at Czech Technical University in Prague (CTU), Czech Republic. In 2015, he received his master (Ing.) degree in robotics and computer vision at the Faculty of Electrical Engineering (FEE), CTU.

Since that, he cooperated with his supervisor Jan Faigl and continued in his research focused mainly on multi-goal path/trajectory planning for non-holonomic vehicles, especially the Dubins vehicle. Besides, he has already contributed to unsupervised learning for trajectory planning, and other related topics. In 2020, he aims to finish his Ph.D. in Computational Robotics Laboratory with the Artificial Intelligence Center, FEE, CTU (http://aic.fel.cvut.cz).