IEEE Robotics and Automation Letters (RAL) paper presented at the
2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
Madrid, Spain, October 1-5, 2018

DCOP: Dubins correlated orienteering problem
Optimising sensing missions of a non-holonomic vehicle under budget constraints

Nikolaos Tsiogkas[1] and David M. Lane[1]

Abstract— **Advances in robotics have allowed mobile robotic platforms to be deployed in an increased number of field operations. The dynamic nature of the operating environment can constrain the robotic mission execution in terms of required time or energy. Current solutions for providing optimal and optimised plans for the resource-constrained sensing mission planning only consider holonomic vehicles. This work proposes the *Dubins Correlated Orienteering Problem* (DCOP) as a solution for resource constrained sensing missions undertaken by a non-holonomic Dubins vehicle. It provides a *Genetic Algorithm* (GA) heuristic for finding optimised solutions in a timely manner. The proposed heuristic is compared against the GA heuristic solution for the *Correlated Orienteering Problem* (COP) and the heuristic for the *Dubins Orienteering Problem* (DOP). It performs as good as the COP heuristic in terms of utility with increased execution time. When tested against the DOP problem it performs comparably good. In the worst case, it produces solutions with 13% less utility but requiring orders of magnitude less time than the state of the art. Finally, the proposed method is tested in a scenario where the budget consumption changes during mission execution. A new plan can be devised online, increasing mission performance. Given the results, it sets the baseline for comparing any future approaches that will address the DCOP problem.**

INTRODUCTION

Advances in the field of robotics have enabled the production of reliable and affordable systems, that are becoming priceless assistants in other fields of science and the industry. Examples can be found in many fields, including underwater archaeology [1] and search and rescue [2].

The nature of the problem solved by the robot can have an effect on the robot's performance. Operating in a dynamic environment can affect the energy usage of the robot, thus leaving it with less energy to complete its mission. This is usually observed in highly tidal environments where the presence of currents affect the mission execution energy and time [3]. In these cases, the mission must be completed in the best possible way despite the limited remaining energy. This can be achieved by replanning online a new trajectory that maximises the mission outcome. In other cases the duration of a mission is essential. For example, in emergency response missions, it is important to maximise the amount of information that can be gathered in a limited amount of time. This way, an effective response plan can be devised and executed in a timely manner.

The work presented in this paper focuses on sensing missions performed by a Dubins vehicle that has a limited operational budget. Dubins vehicles are a common class of non-holonomic vehicles (i.e. not being able to travel in any direction in space). Dubins vehicles have a constraint on the curvature of the path they can follow and it is assumed that they can only travel forward. An example of a Dubins vehicle is fixed wing *Unmanned Aircraft Vehicles* (UAVs). Another class of vehicles that are not classified as Dubins vehicles, but benefit by following Dubins paths are several *Autonomous Underwater Vehicles* (AUVs). For example, vehicles on that class can be even holonomic, such as the Nessie vehicle [4]. However, moving in directions other than forward can lead to high energy and time consumption due to drag caused by the vehicle's body.

The sensing mission studied by this work requires samples to be collected from a user-defined area for estimating a scalar field. Scalar fields are used to represent physical quantities in space. Phenomena like algal blooms, oil spills or chemical pollution can be studied with the use of scalar fields [5]. Such phenomena are dynamic in nature and can potentially cause damage to the environment and infrastructure. Fast and precise monitoring of such phenomena is vital for providing effective solutions. One characteristic example involves the usage of AUVs for monitoring the Deepwater Horizon oil spill.

A solution to sensing missions under budget constraints is presented in [6]. There a variant of the *Orienteering Problem* (OP) is used to estimate a scalar field. It is applied to persistent monitoring tasks with UAVs. Their approach works under the hypothesis that of information among the various sampling points is correlated. The presented variant is called the *Correlated Orienteering Problem* (COP) and *Mixed Integer Quadratic Programming* (MIQP) formulations are presented. Using the MIQP formulations optimal paths for a persistent monitoring task are generated for single and multiple vehicles.

The computational complexity of solving the MIQP problem does not allow optimal solutions to be generated online. To overcome that issue, the work presented in [7] proposes the use of a *Genetic Algorithm* (GA) based heuristic. The heuristic can find close to optimal paths using much less computational resources, enabling online path generation.

This paper focusses in finding heuristic solutions for the COP for a Dubins vehicle. The formed problem is named the *Dubins Correlated Orienteering Problem* (DCOP) and a GA based heuristic is provided. This heuristic tries to generate close to optimal paths for the vehicle while having a low computational complexity so that it can be applied online.

The rest of the paper is organised as follows. In section II relevant work on the topic is presented. Section III focuses

[1] Edinburgh Centre for Robotics, Heriot Watt University, Edinburgh UK
{nt95, d.m.lane}@hw.ac.uk

on the proposed genetic algorithm approach. In section IV the experimental procedure is detailed and relevant results are presented. Finally, section V concludes the paper and proposes potential future exploration paths on the topic.

Relevant work

In the given context, when it comes to utility maximisation under some resource constraint, various solutions have been proposed in the literature. The problem was first presented in [8] with the name of selective travelling salesman problem. The name orienteering problem was first presented in [9]. In general, the OP can be seen as a combination of the knapsack problem and the travelling salesman problem. The vertex selection that maximises the utility can be represented as an instance of the knapsack problem. On the other hand, the search for a minimal path passing from these vertices is an instance of the travelling salesman problem. This problem has found multiple applications in logistics and transportation systems, as presented in [10].

The OP is used in various cases to model tasks of sampling or information gathering. A lake and river monitoring application is presented in [11]. Autonomous surface vehicles are used and a submodular OP solved approximately maximises the information gathering. The work presented in [12] uses a recursive greedy algorithm to maximise the additional information gathered in a sensor network. The usage of the submodular OP is found to be inapplicable for real-time applications in the work of [13]. A linear approximation is proposed for solving the problem of area coverage using a micro-UAV. Scalar field estimation using Gaussian processes is modelled as an instance of the OP in [14]. It is solved using a branch-and-bound method. The problem of marine sampling is studied in [15] using the OP and a GA based heuristic. The work of [16] uses an evolutionary strategy to solve the problem of online terrain classification using UAVs. The work of [17] addresses the informative path planning problem using a dynamic programming solution. While, in [18] it is solved using a randomised algorithm. The approach has a low computational complexity allows the problem to be solved online.

Regarding path optimisation of Dubins vehicles, approaches have been presented for solving the *Dubins Travelling Salesman Problem* (DTSP) [19]. Sampling-based solutions to this problem are presented in [20], [21], [22]. In these approaches, a uniform sample of the vehicle's heading is performed at each vertex and combinatorial optimisation is used. Solutions can be obtained by solving the problem as an instance of the *Asymmetric Travelling Salesman Problem* (ATSP). The resulting ATSP can be solved using the Lin-Kernighan algorithm [23]. A heuristic approach for the DTSP is shown in [24] where a GA approach is used. A solution for the resource-constrained version of the DTSP is presented in [25] where maximising TSP subroutes were found. The *Dubins Orienteering Problem* (DOP) is first presented in [26]. This problem is the extension of the standard OP for Dubins vehicles. In that work a *Variable Neighbourhood Search* (VNS) approach is used to find maximising paths, but its computational complexity is prohibitive for online usage. To the best of the author's knowledge, there is no prior work addressing the correlated informative path planning under budget constraints for a Dubins vehicle.

Methods

This section presents the proposed genetic algorithm heuristic by describing in detail the methods that compose it. Subsection III-A presents in brief a Gaussian process method for scalar field estimation. In III-B the kinematic model of a Dubins vehicle is presented. Subsection III-C presents the Correlated Orienteering Problem. In subsection III-D, finally, the proposed genetic algorithm heuristic is presented in detail.

Scalar field estimation

As mentioned in the previous section, the mission that the vehicle is trying to accomplish is the one of sample collection for scalar field estimation. In [14] a way to model such a field is presented. The proposed method is based on *Gaussian processes* to model the field. This is achieved by using a grid of sampling points for approximating the field. Its values are then modelled as a multivariate Gaussian distribution, where each sampling point represents one dimension. The covariance of the distribution is calculated with the help of a kernel function. A popular choice for a kernel function is the square exponential kernel shown in (1).

$$k(x_i, x_j) = exp(-\frac{\|x_i - x_j\|}{2l^2}) \tag{1}$$

The kernel models the correlation of the value of two points in space. The correlation is based on the distance of the two points and is dictated by parameter l. The value of this parameter can be estimated from past data. In general, as l grows larger, the distance where the value of two points is correlated is also larger. This correlation is used in the COP formulation described in section I.

Dubins vehicle kinematics

The state q of a non-holonomic vehicle on the plane is defined by its position on the plane $(x, y) \in \mathbb{R}^2$ and its orientation $\theta \in \mathbb{S}^1$. One of the main characteristics of a non-holonomic vehicle is its turning radius ρ. The value of ρ has an effect on the minimal path length between two states q_1 and q_2. The kinematic model of a non-holonomic vehicle moving with a constant forward velocity v and having a control input u is described by (2).

$$\dot{q} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = v \begin{bmatrix} \cos\theta \\ \sin\theta \\ \frac{u}{\rho} \end{bmatrix}, u \in [-1, 1] \tag{2}$$

According to [27], the shortest path between two states q_1 and q_2 can be described by using a straight segment and two circular segments or three circular segments. Therefore, the shortest path connecting two states is described by one of the six Dubins manoeuvres $\{LSL, LSR, RSL, RSR, LRL, RLR\}$. There, S describes the straight segment, while L and R are the left and right circular segments of the vehicle turning on a circle with maximum radius ρ. In the scope of this work, the Dubins manoeuvres along with the length of the path $\mathcal{L}$ are calculated using [28].

Dubins Correlated Orienteering Problem

The correlated orienteering problem, as presented in [6], aims at optimising the mission execution by maximising the following objective function:

$$\sum_{i \in V} (r_i x_i + \sum_{v_j \in N_i} r_j w_{ij} x_i (x_i - x_j)) \tag{3}$$

The objective function is represented as the sum of the reward of the visited vertices in a given solution. Set V includes all the available vertices. The reward for visiting vertex i in a solution is given by variable r_i. The binary variable x_i is used to denote if vertex i is visited in a given solution. As described in the COP definition, by visiting a vertex i extra reward can be collected from its neighbourhood, represented by N_i. The extra reward is the sum of each unvisited neighbour j multiplied by a weight w_{ij}. This weight can be represented by the kernel function shown in III-A. The quadratic term $x_i(x_i - x_j)$ enforces that only reward from unvisited vertices is added.

In addition to the objective function, the COP is respecting the available budget of the vehicle. This is enforced by constraint (4).

$$\sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij} \leq C_{max} \tag{4}$$

The maximum allowed budget is denoted by C_{max}. The cost of a path is then defined as the sum of the travelling cost from vertex i to the next vertex j as denoted by $c_{ij} x_{ij}$. In the case presented in [6] the cost of travelling between two vertices is set as their Euclidean distance. In that case, only a holonomic vehicle can follow the produced path without violating the budget constraint.

To extend the COP for a Dubins vehicle one must calculate the set $V' \subseteq V$ of the visited vertices in a solution, as well as, a set of angles Θ representing the angle of the vehicle at each vertex. Accordingly, the cost c_{ij} of (4) is replaced by the Dubins cost (5).

$$c_{ij} = \mathcal{L}((x_i, y_i, \theta_i), (x_j, y_j, \theta_j)) \tag{5}$$

Genetic Algorithm heuristic

Many of the optimisation problems are NP-Hard [6], including the COP and therefore the DCOP. For that, heuristics have appeared in the literature for allowing faster computation times in exchange of optimality. One of the main heuristics used in the literature are genetic algorithms. The optimisation procedure of a genetic algorithm imitates the evolutionary process taking place in nature by having a population of chromosomes that evolve using natural selection, reproduction and mutation [29].

The encoding of each chromosome in this work resembles the one used for the solution of the COP presented in [7]. For the COP the chromosome is encoded by a vector representing the vertices in the order that must be visited. For the DCOP case, an additional vector is used for storing the angles Θ that the vehicle has at each vertex.

Algorithm 1 Genetic Algorithm for the Dubins Correlated Orienteering Problem

```
Input: $TravelCosts, Rewards, MaxCost, Start, Finish$
Output: $FittestChromosome$
 1: GenerAtePopulation
 2: while $gen \leq MaxGen$ or $FittestChromosome$ stable for 10 generations do
 3:     SelectNewPopulation
 4:     Crossover
 5:     Mutate
 6: select $FittestChromosome$
 7: return $FittestChromosome$
```

Algorithm 1 presents an outline of the optimisation procedure followed by the GA. The algorithm can be divided into two distinct parts. In the first part the population is generated, while in the second part selection, crossover and mutation operations are applied for optimising the population.

As already mentioned, the first part of the GA is responsible for generating the population. This work uses a modified version of the *Nearest Neighbour Randomised Adaptive Search Procedure* (NN-RASP) generation method, presented in [30]. This method considers the particular nature of the DCOP problem regarding the information correlation among sampling points. An algorithmic representation of it can be seen in 2.

Algorithm 2 Dubins NN-RASP

```
Input: $TravelCosts, Rewards, MaxCost, Start, Finish$
Output: $Chromosome$
 1: Insert($V, Start$)
 2: $v \leftarrow$ GetLast($V$)
 3: $a \leftarrow$ GetLast($A$)
 4: $N \leftarrow$ GetNeighbours($v$)
 5: while GetSize($N$) $> 0$ do
 6:     for each $n \in N$ do
 7:         GetWeight($n$)
 8:     $nv \leftarrow$ GetWeightedRandom($N$)
 9:     $nva \leftarrow$ GetMinCostAngle($v, a, nv$)
10:     if CostInsert($V, nv, nva$) $\leq MaxCost$ then
11:         Insert($V, nv$)
12:         Insert($A, nva$)
13:         $v \leftarrow$ GetLast($V$)
14:         $N \leftarrow$ GetNeighbours($v$)
15:     else
16:         $N \leftarrow \{\varnothing\}$
17: $fa \leftarrow$ GetMinCostAngle($Finish$)
18: Insert($V, Finish$)
19: Insert($A, fa$)
20: return $Chromosome$
```

This method constructs a path by iteratively choosing one of the neighbouring vertices of the last inserted vertex. Each neighbour is assigned a probability and the choice is done by sampling from a categorical distribution. The categorical distribution is a discrete distribution describing the possible results that a random variable can take. In this case, the possible results are the neighbouring vertices. The assigned probability of each vertex is calculated based on two metrics. The first metric is its distance from all the other visited vertices in the solution so far. The second metric is the number of free neighbours it has. The distance-based metric is used as it can lead the constructed path in new unexplored areas. The amount of free neighbours ensures that increased reward will be gained from correlation of information, while additionally providing the process with more options to continue the path construction.

After the next vertex is chosen, the angle that minimises the travel cost from the previous vertex is calculated based on the previous vertex's angle. To reduce the computational effort not all angles in the interval $\theta \in [0, 2\pi)$ are considered. Instead, it is searched in a set of m uniformly sampled angles from $[0, 2\pi)$ in a similar manner as presented in [26].

If the cost to insert the vertex to the solution does not violate the maximum cost constraint, the vertex is inserted and the search continues. If the vertex cannot be inserted, the search stops. Finally, the best angle to reach the finish vertex is calculated. The angle and the finishing vertex are inserted to the chromosome and the chromosome is returned. This process is repeated until all the population is generated.

Following their generation, the chromosomes are evaluated. The chromosome's fitness is calculated as the reward collected from the path to the third power divided by the path's cost. This is suggested by a GA heuristic for the OP presented in [31]. This choice emphasises the reward a path collects, while when paths have similar rewards the shortest path is fitter.

After the population is generated a repetitive search process starts. This search process is repeated until the maximum amount of generations is reached or until the best chromosome is stable for ten consecutive generations.

The first phase of the search process is selecting a new chromosome generation based on the previous one. This process is performed in two steps. In the first step, a percentage of the best chromosomes of the old generation is selected and passed to the next generation. This step is known as *Elitism*. It is used to speed up the search performance and prevent good solutions from being lost during the search process [32]. The second step fills up the rest of the new generation using *Tournament selection*. For this method, n random individuals are chosen from the old population and the fittest is added to the new one. This is repeated until the new generation reaches its maximum size.

The second phase of the search requires a crossover operation to be performed to a percentage of the population. The operation is presented in algorithm 3. This operation is aiming at exploring more of the search space.

This operation requires two chromosomes to be randomly chosen as the parents, which will be replaced by two produced offsprings. Initially, a set of common vertices of the parents is computed. If the set is empty the crossover operation cannot be completed and the parents are returned as offsprings. If there are common vertices, a random vertex is selected. The two parents are then sliced at that vertex and two offsprings are created by swapping the tail parts of the parents. Then, any duplicate vertices are removed from each offspring. Following, the respective angle for each vertex is computed in a way that minimises the whole path's cost. After the offsprings have been fully defined, their feasibility is checked by comparing their cost against the maximum cost. If an offspring is infeasible, it is replaced by the respective parent. Finally, the two offsprings are evaluated and returned.

Algorithm 3 Crossover

```
Input: $Parent^1, Parent^2, MaxCost$
Output: $Child^1, Child^2$
 1: $I \leftarrow$ GetCommonVertices($Parent^1_V, Parent^2_V$)
 2: if $I \neq [\varnothing]$ then
 3:     $RG \leftarrow$ SelectRandom($I, 1$)
 4:     $Child^1_V \leftarrow [Parent^1_{[1,RG]}, Parent^2_{[RG,end]}]$
 5:     $Child^2_V \leftarrow [Parent^2_{[1,RG]}, Parent^1_{[RG,end]}]$
 6:     $Child^1_V \leftarrow$ RemoveDuplicates
 7:     $Child^1_A \leftarrow$ CalculateAngles
 8:     $Child^2_V \leftarrow$ RemoveDuplicates
 9:     $Child^2_A \leftarrow$ CalculateAngles
10:     if GetCost($Child_1$) $> MaxCost$ then
11:         $Child^1 \leftarrow Parent^1$
12:     if GetCost($Child_2$) $> MaxCost$ then
13:         $Child^2 \leftarrow Parent^2$
14:     EvaluateChromosome($Child^1$)
15:     EvaluateChromosome($Child^2$)
16: else
17:     $Child^1 \leftarrow Parent^1$
18:     $Child^2 \leftarrow Parent^2$
19: return $Child^1, Child^2$
```

The final phase of the iterative search process requires a percentage of the population to evolve through mutation. The mutation aims to improve the fitness of the chromosome by local search methods. The mutation method in this work is similar to the one presented in [7] and is described in algorithm 4.

Algorithm 4 Mutate

```
Input: $Chromosome, MaxCost, NumMut, AddProb$
Output: $Chromosome$
 1: $C \leftarrow Chromosome$
 2: for $i \leftarrow 1, NumMut$ do
 3:     $p \leftarrow$ GetRandom($0, 1$)
 4:     if $p \leq AddProb$ then
 5:         if GetCost($g$) $\geq 0.95 * MaxCost$ then
 6:             $v \leftarrow$ SelectRandom($V, 1$)
 7:             $n, a \leftarrow$ GetMaxFreeNeighbour($v$)
 8:             if FitnessIncrease($C$) then
 9:                 Replace($C, v, n, a$)
10:         else
11:             if $FV \neq [\varnothing]$ then
12:                 $v \leftarrow$ SelectRandom($FV, 1$)
13:                 $idx, a \leftarrow$ BestInsertion($C, v$)
14:                 Insert($C, v, a, idx$)
15:     else
16:         $v \leftarrow$ GetMinLossVertex($C$)
17:         $C \leftarrow$ Remove($C, v$)
18: EvaluateChromosome($C$)
19: return $C$
```

Mutation is applied to a single chromosome. Each time the mutation operation is applied to a chromosome, the selected chromosome is mutated $NumMut$ times. This operation tries to improve the chromosome's fitness by either adding or removing a vertex with some probability $AddProb$. The vertex addition has two behaviours based on the cost of the chromosome. The first behaviour is applied if the cost of the existing path is more than 95% of the maximum cost. Then, a vertex of the path is swapped with one of its free neighbours. A vertex from the path is randomly chosen. This vertex is removed from the path. Then n paths are created by inserting each of the free neighbours of the removed vertex in its place. Each time the optimal neighbour's angle is searched so that the swap cost increase is minimised. The path with the maximum fitness is compared to the current path. If the new path has higher fitness and its cost is less than the maximum cost it is kept as a new solution.

The second addition behaviour takes place in cases where the cost is lower than 95% of the maximum cost. In that case, a free vertex is attempted to be added to the solution. A free vertex is randomly chosen and the best insertion position and angle are found. The best insertion position is determined by the highest $fitness\ increase/cost\ increase$ ratio. If the insertion does not violate the cost constraint it is kept as the new solution.

For vertex removal, the minimum loss vertex is chosen. The minimum loss vertex is the one that has the minimal ratio of $fitness\ loss/cost\ decrease$. When all $NumMut$ mutations are performed, the chromosome is evaluated and is returned.

RESULTS

This section presents the results and is divided into four parts. In the first part, a standard literature method to tune the genetic algorithm parameters is detailed and its results are presented.

In the second part, the heuristic is evaluated in a sensing mission scenario as presented in [7]. In the examined scenario a simulated sensing mission is taking place, for which a vehicle must plan a trajectory. The mission consists of a grid of sampling points the vehicle must visit. The starting and ending points of the mission are provided. In addition, the vehicle has a limited amount of budget for completing the mission. The aim of the vehicle is to maximise the gained utility by visiting sampling points, without violating the budget constraint. Evaluation is performed on two different points. The first point refers to the utility gathered by the vehicle. The second point is about the time required to generate a trajectory. Given that the GA is a stochastic optimisation process, the produced solution will be different each time it is run. Therefore, only statistical results can be extracted. For that 1000 trials are run and a statistical analysis of the results is performed.

The third part compares the heuristic's performance to results from the literature regarding the DOP. The DOP is a specialised case of the DCOP where correlation of information does not provide any extra reward. Finally, results of an online replanning scenario are presented in the fourth part. In that case, a new plan is generated online to cope with changes in the mission budget happening during mission execution.

TABLE I. Parameter ranges and tuned values for the two different methods

Parameter	Value Range	Tuned value
Population Size	$[25, 50, \ldots, 500]$	450
Generations Number	$[5, 10, \ldots, 50]$	35
Tournament Size	$[3, 4, \ldots, 10]$	3
CX Probability	$[0.0, 0.1, \ldots, 0.9]$	0.6
Mutation Probability	$[0.0, 0.1, \ldots, 0.9]$	0.8
Elitist Percentage	$[0.01, 0.02, \ldots, 0.20]$	0.07
Mutation NumMut		10
Mutation AddProb		0.9

The parameter tuning and the experiments were run on a system having an Intel i5-7600 CPU, 16GB of RAM and running Ubuntu 16.04. The heuristic is coded in C++11 and compiled using g++ version 5.4.0. Source code for the heuristic can be found online[1].

Parameter tuning

In section III a GA heuristic was described. The heuristic's behaviour is governed by a set of parameters. Choosing the appropriate values for these parameters is essential for the optimal performance of the heuristic. Parameter tuning approaches for GA heuristics have been introduced in the literature. For example, the work of [33] presents a general framework for parameter tuning and discusses various tuning methods. The procedure used in this work is based on the work first presented in [34].

The method that is presented [34] is also based on an evolutionary approach, using the *Glicko2* chess rating system presented in [35]. In this method, each chromosome is represented using a vector of integer and real values. These values represent the set of the parameters to be tuned. To avoid over-fitting the parameters to a specific problem instance, each parameter set is evaluated using a diverse set of problems.

The proposed GA is tuned using thirty different problem instances. These instances are created by changing the problem size, the available budget and the turning radius of the vehicle. Specifically, instances with grid sizes of $5x5$, $7x7$ and $9x9$ are used. The maximum available budget is set to be 75% and 50% of the maximum required to visit all the vertices. Finally, the turning radius ρ has values from 0.0 to 1.0 in 0.25 increments. Each parameter combination is run 25 times against each instance. They are then ranked using the *Glicko2* rating system and the best are selected to form the new generation. The parameter search space is explored by mutating part of the population. In total, the parameters are tuned by running 5 generations of optimisation. The tuned parameter values, along with the parameter ranges can be seen in table I.

Sensing mission results

To evaluate the performance of the proposed heuristic method two different sets of tests are used. The first set studies the performance of the heuristic for different values of the vehicle's turning radius. In this case, the maximum allowed budget is allocated to the vehicle. The maximum

[1]`https://github.com/lounick/lwga`

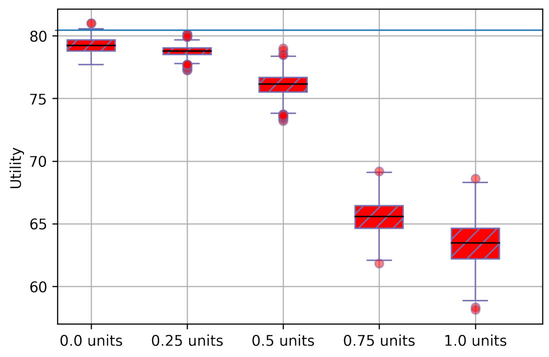

Fig. 1. Utility box plot for different turning radii. The horizontal line represents the average utility obtained by the GA for a holonomic vehicle presented in [7].

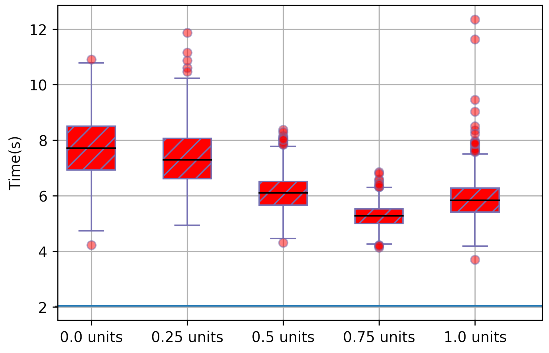

Fig. 2. Time box plots for different turning radii on a $9x9$ grid. The horizontal line represents the average time required to obtain a path from the GA heuristic presented in [7].

allowed budget is given by the minimum length euclidean path that traverses all the vertices in the grid. It can be calculated by solving an *Open Vehicle Routing Problem* (OVRP) instance [36]. The results can be seen in fig. 1 and 2.

Statistical results regarding the gathered utility for various turning radii can be seen in fig. 1. For a turning radius of 0.0, the problem is transformed to the one of the COP for a holonomic vehicle as presented in [7]. The horizontal line in fig. 1 shows the average utility gathered by the GA heuristic for the holonomic vehicle of [7]. It can be seen that the heuristic presented in this work is performing in a comparable manner, achieving the same level of performance. As it is expected, increasing the turning radius of the vehicle reduces the obtained utility. This happens because the vehicle is forced to spend more budget travelling due to the turning radius constraints.

In fig. 2 one can see statistical results regarding the computation time of a path for different turning radii on a $9x9$ grid. The time to compute a path is comparable for all different radii. The horizontal line represents the average time required to get a path from the GA heuristic of [7]. It should be noted that these times are obtained by running the experiments of [7] in the same hardware used for the experiments of this work. The comparison shows that the newly proposed heuristic is at least three times slower. This is attributed to the extra computational effort required to find angles that minimise the path for the Dubins vehicle.

The second set of experiments studies how the proposed heuristic performs in cases of a limited budget. For this, the turning radius of the vehicle ρ is set to be equal to 0.5 and experiments are run for budgets of 100% down to 25% of the minimum required budget to visit all the vertices. Results regarding the collected utility and computation time can be seen in fig. 3 and 4 respectively.

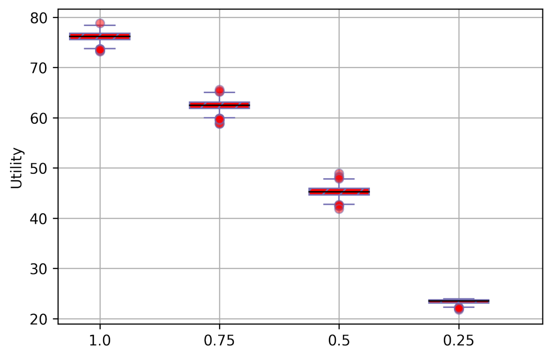

Fig. 3. Utility box plots for different budgets. The heuristic behaviour is rather consistent statistically. The fall in utility follows a linear rate to the reduction of the budget.

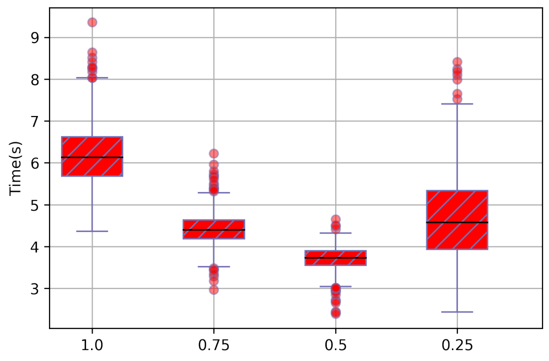

Fig. 4. Time box plots for different budgets. For all the budgets the time to plan a path is statistically similar.

In fig. 3 statistical results regarding the utility gathered for different budget values are presented. It can be seen that for each available budget level the performance of the heuristic is rather consistent with only a few outliers. The fact that some outliers have higher utility suggests that the heuristic performance is not usually finding the best solution. In addition, the reduction of the utility appears to be linear to the reduction of the budget.

Regarding the time required for planning a trajectory for different budgets, fig. 4 presents statistical results. The time required is similar for all budgets. It can also be seen that the full budget case tends to require more computational time.

DOP comparison results

If the reward received from correlation of information is set to zero the problem to be solved becomes an instance of the orienteering problem. A solution method of the OP for a Dubins vehicle is proposed in [26]. It is tested using benchmark instances for the OP that can be found in the literature [9], [37]. The work of [26] uses a VNS heuristic to find optimised solutions. In this section, the GA proposed method is compared against the VNS for the DOP using the same instances. Results are presented in tables II and III. The comparison is performed in the same way as in [26] where the best solution out of 10 runs is chosen. Due to limitation

TABLE II. Tsiligirides OP Set 3

T_{max}	Alg.	Turning raduis (ρ)						
		0.0	0.3	0.5	0.7	0.9	1.1	1.3
15	GA	170	170	160	160	160	150	130
	VNS	170	170	160	160	160	150	**140**
35	GA	390	380	380	**380**	380	380	360
	VNS	390	380	380	360	380	380	360
60	GA	580	580	570	560	560	560	530
	VNS	580	580	570	560	560	560	**550**
85	GA	740	730	720	**720**	**710**	710	690
	VNS	740	730	**730**	700	700	710	**710**
110	GA	800	800	800	800	800	800	790
	VNS	800	800	800	800	800	800	**800**

TABLE III. Chao OP Set 66

T_{max}	Alg.	Turning raduis (ρ)						
		0.0	0.3	0.5	0.7	0.9	1.1	1.3
5	GA	10	10	10	10	0	0	0
	VNS	10	10	10	10	0	0	0
30	GA	380	370	370	370	370	370	365
	VNS	**400**	**400**	**380**	370	370	370	**370**
55	GA	780	820	820	**820**	820	790	755
	VNS	**825**	**825**	**825**	800	820	**795**	**790**
80	GA	1150	**1200**	1180	**1190**	1165	1145	1095
	VNS	**1215**	1195	**1190**	1170	**1175**	**1165**	**1155**
105	GA	1445	1495	1480	1440	1435	1425	1340
	VNS	**1520**	1495	**1505**	**1495**	**1485**	**1470**	**1445**
130	GA	1655	1670	1670	1670	1655	1635	1585
	VNS	**1680**	**1680**	**1675**	**1675**	**1670**	**1670**	**1655**

in space partial results are presented, while the whole set of results can be found online[2].

As it can be seen the proposed GA performs comparably with the VNS for Set 3. On the other hand in Set 66 the VNS performs better in most cases. Nevertheless, the maximum difference in utility is in a single worst case 13%. Usually, it is less than 5% or even better than the VNS solution. The main benefit of the GA is the execution time to find a solution. For example, in the worst case of Set 66 it requires on average 4 seconds of time to produce a path, while in the case of [26] 800 seconds are required.

A possible cause of this is the way the GA operations are designed. For example, during chromosome generation, new vertices are evaluated based on their distance from already visited vertices and the number of free neighbours they have. This choice is natural based on the nature of the sampling mission. On the one hand, in a sampling mission, each vertex is considered to be of equal importance. On the other hand, in cases where the budget is not enough to visit all the vertices, it is desirable to have the maximum exposure to free neighbours where information can be gathered by correlation.

In the same manner, the mutation operation tries to randomly insert or exchange vertices ignoring the actual reward of each vertex. Instead, it tries to maximise the vertices that are directly or indirectly visited. Despite the specialised design for a specific problem, the proposed solution performs really close or even better than the one found in the literature for the DOP. Additionally, it has much fewer requirements in computation time making it applicable online.

[2]`https://goo.gl/ysNdQn`

Online replanning results

The final set of results focuses on the solution quality improvement in the presence of budget changes. These budget changes can be caused by changes in the environment. For example, the presence of underwater currents may increase the energy consumption to the point that a mission cannot be completed within a specified budget.

The results presented here are synthetically generated by altering the budget consumption and causing a replanning at runtime. Specifically, after completing 10% of the mission the agent senses that 20% of the budget was consumed. This causes a replanning procedure, leading to an optimised trajectory for the new budget. During replanning, the GA is run three times and the best result is kept.

The replanning benefit is calculated as the difference between the utility of the new trajectory and the utility gathered by the original trajectory until the budget is consumed. Averaging over 1000 experiments the replanning improves the utility by 9.978% and it requires 7.126 seconds. Example trajectories can be seen in fig. 5. The quality of the replanned solution is noteworthy as it maximises the area coverage, thus giving information to model the whole area more accurately.

Conclusion

The work presented by this paper studies the problem of sensing missions performed by a single non-holonomic vehicle operating under resource constraints. It proposes the Dubins correlated orienteering problem. The DCOP maximises the outcome of the mission by taking into account the correlation in the sensed information among the sensing points. It ensures that resource usage is constrained and that the vehicle kinematic constraints are respected. A genetic algorithm heuristic is presented which is able to find optimised solutions in a timely manner. The heuristic is compared against existing heuristic solutions for the Euclidean COP and is shown to perform in a comparable manner, validating the correctness of the approach. Additionally, it presents results for Dubins vehicles with various turning radii, setting the baseline for comparison of future approaches solving the DCOP. The proposed GA is also tested on DOP instances from the literature, producing comparable results with the state of the art, while requiring a fraction of time to compute. Finally, a simulated scenario demonstrates the application of the heuristic by replanning maximising trajectories online after changes in the vehicle's budget consumption. A natural extension to the current work is the solution of the problem for multiple vehicles, potentially having heterogeneous characteristics both in the resource or the kinematic constraints.

References

[1] B. Allotta, *et al.*, "The arrows project: adapting and developing robotics technologies for underwater archaeology," *IFAC-PapersOnLine*, vol. 48, no. 2, pp. 194–199, 2015.

[2] T. Tomic, *et al.*, "Toward a fully autonomous uav: Research platform for indoor and outdoor urban search and rescue," *IEEE robotics & automation magazine*, vol. 19, no. 3, pp. 46–56, 2012.

[3] J. Bellingham *et al.*, "Optimizing auv oceanographic surveys," in *Autonomous Underwater Vehicle Technology, 1996. AUV '96., Proceedings of the 1996 Symposium on*, Jun 1996, pp. 391–398.

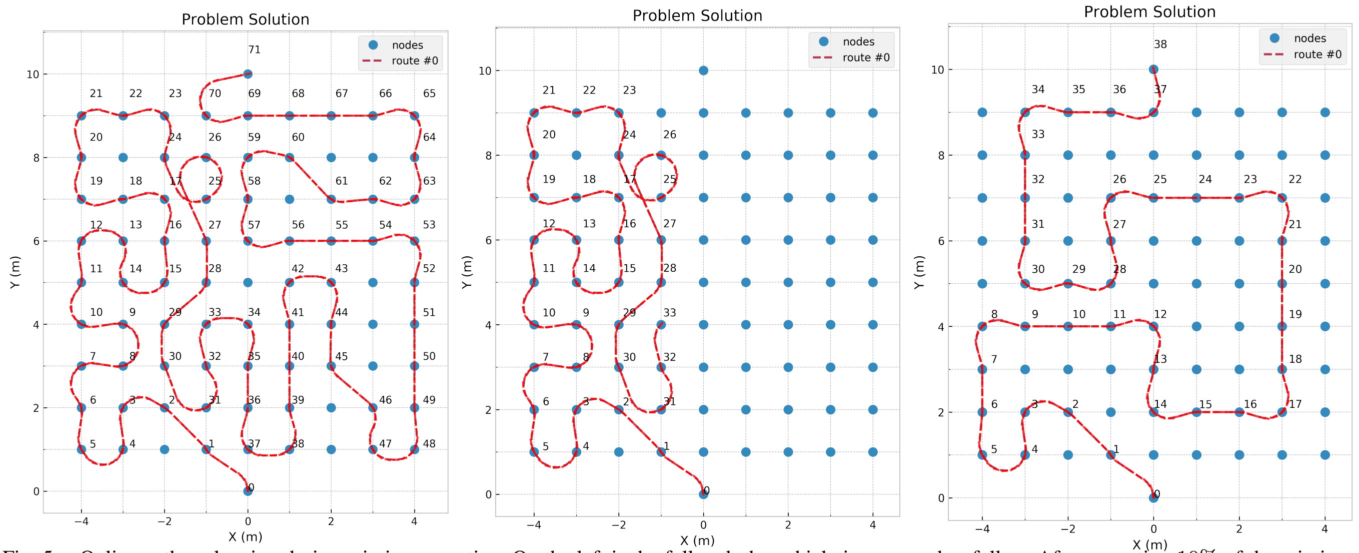

Fig. 5. Online path replanning during mission execution. On the left is the full path the vehicle is supposed to follow. After executing 10% of the mission the agent senses that it consumed 20% of the budget. The middle figure shows where the mission would have ended if no replan was performed. On the right the optimised replanned path is shown.

[4] N. Valeyrie, *et al.*, "Nessie v turbo: a new hover and power slide capable torpedo shaped auv for survey, inspection and intervention," in *AUVSI North America 2010 Conference*, 2010.

[5] H. M. La, *et al.*, "Cooperative and active sensing in mobile sensor networks for scalar field mapping," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 45, no. 1, 2015.

[6] J. Yu, *et al.*, "Correlated orienteering problem and its application to persistent monitoring tasks," *IEEE Transactions on Robotics*, vol. 32, no. 5, pp. 1106–1118, 2016.

[7] N. Tsiogkas, *et al.*, "Towards an online heuristic method for energy-constrained underwater sensing mission planning," in *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*. IEEE, 2017.

[8] G. Laporte *et al.*, "The selective travelling salesman problem," *Discrete applied mathematics*, vol. 26, no. 2-3, 1990.

[9] I.-M. Chao, *et al.*, "A fast and effective heuristic for the orienteering problem," *European Journal of Operational Research*, vol. 88, no. 3, pp. 475–489, 1996.

[10] A. Gunawan, *et al.*, "Orienteering problem: A survey of recent variants, solution approaches and applications," *European Journal of Operational Research*, vol. 255, no. 2, pp. 315–332, 2016.

[11] A. Singh, *et al.*, "Efficient informative sensing using multiple robots," *Journal of Artificial Intelligence Research*, pp. 707–755, 2009.

[12] J. Binney, *et al.*, "Informative path planning for an autonomous underwater vehicle," in *Robotics and automation (icra), 2010 IEEE international conference on*. IEEE, 2010.

[13] L. Heng, *et al.*, "Efficient visual exploration and coverage with a micro aerial vehicle in unknown environments," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, 2015.

[14] J. Binney *et al.*, "Branch and bound for informative path planning," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 2147–2154.

[15] S. Frolov, *et al.*, "Can we do better than the grid survey: Optimal synoptic surveys in presence of variable uncertainty and decorrelation scales," *Journal of Geophysical Research: Oceans*, vol. 119, no. 8, pp. 5071–5090, 2014.

[16] M. Popović, *et al.*, "Online informative path planning for active classification using uavs," in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE, 2017, pp. 5753–5758.

[17] K.-C. Ma, *et al.*, "An information-driven and disturbance-aware planning method for long-term ocean monitoring," in *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. IEEE, 2016, pp. 2102–2108.

[18] S. Arora *et al.*, "Randomized algorithm for informative path planning with budget constraints," in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE, 2017, pp. 4997–5004.

[19] K. Savla, *et al.*, "Traveling salesperson problems for the dubins vehicle," *IEEE Transactions on Automatic Control*, vol. 53, no. 6, pp. 1378–1391, 2008.

[20] J. Le Ny, *et al.*, "On the dubins traveling salesman problem," *IEEE Transactions on Automatic Control*, vol. 57, no. 1, pp. 265–270, 2012.

[21] P. Oberlin, *et al.*, "A transformation for a heterogeneous, multiple depot, multiple traveling salesman problem," in *American Control Conference, 2009. ACC'09.* IEEE, 2009.

[22] J. T. Isaacs, *et al.*, "Algorithms for the traveling salesman problem with neighborhoods involving a dubins vehicle," in *American Control Conference (ACC), 2011.* IEEE, 2011.

[23] K. Helsgaun, "An effective implementation of the lin–kernighan traveling salesman heuristic," *European Journal of Operational Research*, vol. 126, no. 1, pp. 106–130, 2000.

[24] X. Yu *et al.*, "A genetic algorithm for the dubins traveling salesman problem," in *Industrial Electronics (ISIE), 2012 IEEE International Symposium on*. IEEE, 2012, pp. 1256–1261.

[25] A. Adler *et al.*, "The stochastic traveling salesman problem and orienteering for kinodynamic vehicles," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE, 2016.

[26] R. Pěnička, *et al.*, "Dubins orienteering problem," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 1210–1217, 2017.

[27] 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*, vol. 79, no. 3, pp. 497–516, 1957.

[28] A. Walker, "Dubins-curves: an open implementation of shortest paths for the forward only car," 2008–. [Online]. Available: https://github.com/AndrewWalker/Dubins-Curves

[29] M. Mitchell, *An introduction to genetic algorithms*. MIT press, 1998.

[30] N. Tsiogkas *et al.*, "An evolutionary algorithm for online, resource constrained, multi-vehicle sensing mission planning," *IEEE Robotics and Automation Letters*, vol. PP, no. 99, pp. 1–1, 2018.

[31] J. Karbowska-Chilinska, *et al.*, "Genetic algorithm solving orienteering problem in large networks." in *KES*, 2012, pp. 28–38.

[32] H. Li *et al.*, "An elitist grasp metaheuristic for the multi-objective quadratic assignment problem," in *Evolutionary multi-criterion optimization*. Springer, 2009, pp. 481–494.

[33] A. E. Eiben *et al.*, "Parameter tuning for configuring and analyzing evolutionary algorithms," *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 19–31, 2011.

[34] N. Veček, *et al.*, "Parameter tuning with chess rating system (crs-tuning) for meta-heuristic algorithms," *Information Sciences*, vol. 372, pp. 446–469, 2016.

[35] M. E. Glickman, "Example of the glicko-2 system," *Boston University*, 2012.

[36] F. Li, *et al.*, "The open vehicle routing problem: Algorithms, large-scale test problems, and computational results," *Computers & Operations Research*, vol. 34, no. 10, pp. 2918–2930, 2007.

[37] T. Tsiligirides, "Heuristic methods applied to orienteering," *Journal of the Operational Research Society*, vol. 35, no. 9, pp. 797–809, 1984.