Multi-robot Scheduling for Environmental Monitoring as a Team Orienteering Problem

Ariella Mansfield[1], Sandeep Manjanna[1], Douglas G. Macharet[2], and M. Ani Hsieh[1]

***Abstract*— In this paper, we propose an evolutionary algorithm for solving the multi-robot orienteering problem where a team of cooperative robots aims to maximize the total information collected by visiting a subset of given nodes within a fixed budget on travel costs. Multi-robot orienteering problems are relevant to applications such as logistic delivery services, precision agriculture, and environmental sampling and monitoring. We consider the case where the information gain at each node is related to the service time each robot spends at the node. As such, we address a variant of the Orienteering Problem where the collected rewards are a function of the time a robot spends at a given location. We present a genetic algorithm solver to this cooperative Team Orienteering Problem with service-time dependent rewards. We evaluate the approach over a diverse set of node configurations and for different team sizes. Lastly, we evaluate the effects of team heterogeneity on overall task performance through numerical simulations.**

I. Introduction

This paper considers the cooperative team orienteering problem with service-time dependent rewards. The Orienteering Problem (OP) is a routing and scheduling problem where the goal is to plan a fixed-budget route through a set of predetermined locations such that the total rewards collected is maximized. We address the multi-robot variant of this problem where the rewards at each location are service time dependent, *e.g.*, the collected rewards depend on the amount of time a robot spends at a location. Cooperative team orienteering problems are relevant to many real-world applications such as logistic delivery services, precision agriculture, and environmental sampling and monitoring.

In this work, we are particularly interested in environmental monitoring tasks for coordinating search and rescue plans, anomaly detection, and performing geological and biological surveys. Obtaining good estimates of these complex spatial processes is critical in these applications and requires persistent information gathering throughout the region of interest. In the context of environmental monitoring, persistent sensing has been tackled by deploying a uniform network of static sensor nodes across the region of interest [1]. However, sensor networks are difficult to scale to larger environments and most processes of interest are usually non-uniform. In general, mapping and tracking these large-scale non-uniform fields requires sampling at various distinct locations in the space. As such, robots must target locations in the workspace that yield maximum information gain while minimizing costs associated with energy expenditure, distance traveled, inter-robot communication, and time spent at a given location. Heterogeneous multi-robot teams have been shown to achieve better performance in many environmental monitoring tasks [2], [3]. We evaluate our algorithm on a team of robots operating at different travel speeds and with different budgets.

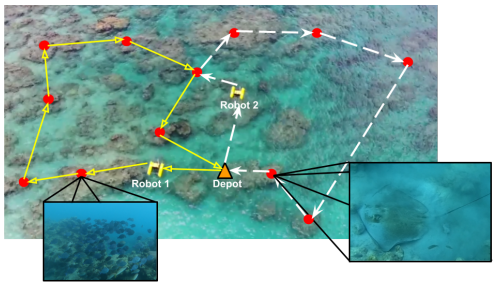


Fig. 1. An example real world application: Here, scientists are interested in monitoring marine life by collecting images at fixed locations over coral heads (red circles). Embedded images show aquatic animals like stingray and a school of blue tang fish visiting the reef at Barbados marine reserve.

Consider the example application shown in Fig. 1, where the objective is to monitor marine life by collecting visual data using a robotic boat [4]. Marine biologists need persistent data from pre-selected sighting locations with high aquatic life activity to understand and study the population variations and migration patterns of different species in these locations. Predicting when a particular animal would visit one of these locations is challenging, but one can increase the probability of sighting by spending more time at one location. However, collecting data at all the locations is equally important. Hence, a data logging robot can increase the chance of sighting by carefully dividing its time between observing at one location and exploring other locations. As such, the information or rewards garnered at each location increases with time.

We propose a planning and scheduling algorithm to design multi-robot paths over a fixed set of nodes that maximizes the total information gathered within a fixed robot budget.

The primary contribution in this paper is the formulation of a cooperative team orienteering problem with service-time dependent rewards and an empirical analysis of the overall performance of heterogeneous robotic teams in such scenarios. Since this is the first study of this problem formulation, our goal is to provide insights into this category of problems rather than providing the fastest algorithm. Therefore, we show our results using a genetic algorithm, which is a well known method for solving routing and scheduling problems.

[1]The authors are affiliated with the GRASP Laboratory at the University of Pennsylvania, Philadelphia, USA {ariellam,msandeep,m.hsieh}@seas.upenn.edu

[2]Douglas G. Macharet is affiliated with the Computer Vision and Robotics Laboratory (VeRLab), Computer Science Department, Universidade Federal de Minas Gerais, MG, Brazil doug@dcc.ufmg.br

II. Related Work

The Orienteering Problem, introduced as the Selective Traveling Salesman Problem [5], is usually described as the combination of the Knapsack Problem (KP) and the Traveling Salesman Problem (TSP). This is due to the fact that, given the limited travel budget, we might need to select a subset of the points to be visited, as well as the sequence of the visit. The classical OP formulation has been generalized to different variants [6], such as robots with motion constraints [7], correlated profits [8], time-varying profits [9], and multi-agent ensembles [10].

While employing multiple robots can yield significant advantages, it raises many challenges in regards to planning and coordination. The Team Orienteering Problem (TOP) is the multi-robot variant of the classic OP formulation [10], where robots must uniquely visit a disjoint set of locations and maximize the total amount of rewards collected. Additional variations include situations where multiple samples are needed at the same location or requiring multiple agents to collectively obtain that information. The cooperative version of the TOP [11], [12] deals with the case that requires multiple robots to simultaneously visit a location to collect the reward. However, this assumption can be too restricted if we have, for example, a scenario with time-varying rewards [9], [13].

Most existing literature related to orienteering problems consider a fixed value for the reward associated to a location. Erdogan *et al.* introduced a variant where the reward collected depends on the time the robot spends at a location [14]. In this variant of the problem, determination of an appropriate service time at each location is another variable that must be optimized. Traditional optimization approaches might be computationally demanding [15], and heuristics have proven to be a suitable alternative [16]. An efficient matheuristic algorithm is proposed by Yu *et al.* combines a metaheuristic approach with mathematical programming [17]. A TOP variant that considers heterogeneous robots and precedence constraints was considered by Hanafi *et al.* [18]. In their approach, each location contains a set of tasks, and the reward can only be acquired when all its tasks are accomplished following a predefined order. More recently, the Generalized TOP (GTOP) was introduced by Sakamoto *et al.* in [19]. They formalize the concept of heterogeneity in both tasks and teams, TOP being a particular case where they are homogeneous.

In this paper, we extend the ideas of [14], [18] and consider a heterogeneous robot team variant of the TOP problem. We also consider the case where the reward at a given location is given by a nonlinear function over the service time [14]. Different from the traditional time-windows-based approach, we consider that the visits can occur in an asynchronous fashion, similar to [18].

III. Problem Formulation

Let $\boldsymbol{\mathcal{N}} = \{\mathbf{n}_1, \ldots, \mathbf{n}_N\}$ be the set of spatially distributed target locations, with $\mathbf{n}_i \in \mathbb{R}^n$, and let $\boldsymbol{\mathcal{A}} = \{\mathbf{a}_1, \ldots, \mathbf{a}_M\}$ be a team of cooperative robots. For the remainder of this section, we use the indices $i, j, k \in \{1, \ldots, N\}$ to denote location indices and the index $m \in \{1, \ldots, M\}$ to denote the robot, unless indicated otherwise.

Each location has a known positive maximum reward r_i and an associated function $f_i \in [0, 1]$. Function $f(\cdot)$ is strictly increasing, differentiable and concave, and determines the ratio of the maximum reward obtained by the robots $\boldsymbol{\mathcal{A}}$ given their collective service time $\sum_{m=1}^{M} s_{im}$, *i.e.*, $r_i f_i(\sum_{m=1}^{M} s_{im})$. In this work, we consider cost to be the time spent doing a certain task, however our approach could be applied to other definitions of cost such as energy consumption, and sensor depletion. Thus, service cost is the time spent while the robot performs a particular service at a location. Travel cost is the time spent travelling between locations. The initial location $\mathbf{n}_1$ and the final location $\mathbf{n}_N$ are fixed and their maximum rewards are assumed to be $r_1 = r_N = 0$.

The cost to travel between $\mathbf{n}_i$ and $\mathbf{n}_j$ for robot $\mathbf{a}_m$ is given by t_{ijm}. We assume the costs are non-negative and both travel cost and service cost are in same scale and units. Heterogeneity in robots can be incorporated in t_{ijm}, for example, if robot $\mathbf{a}_1$ is twice as fast as $\mathbf{a}_2$, then $t_{ij1} = \frac{1}{2} t_{ij2}$.

Our objective is to determine a collection of paths connecting $\mathbf{n}_1$ and $\mathbf{n}_N$ that maximizes the total collected reward by the team $\boldsymbol{\mathcal{A}}$. The budget for each robot $\mathbf{a}_m$ is predefined and fixed at $T_{\max,m}$. Note that the total budget includes the budget for both travel and service costs. To be consistent with existing literature, we use similar notation to that presented by Vantsteenwegen *et al.* [6] and Yu *et al.* [17]. A formal mathematical formulation of this problem is as follows:

Problem 1 (Cooperative TOP with Service-Based Rewards)**.** *Given a set $\boldsymbol{\mathcal{N}}$ of spatially distributed target locations, and a team of cooperative robots $\boldsymbol{\mathcal{A}}$. The objective is to maximize the total rewards collected by the team, i.e.:*

$$\max \sum_{i=2}^{N-1} r_i f_i\Big(\sum_{m=1}^{M} s_{im}\Big) \ , \tag{1}$$

where r_i is the maximum reward that can be collected at node i. The function $f_i \in [0, 1]$ returns the ratio between the maximum reward that can be collected at node i and the reward collected by all the robots. The service cost, s_{im}, is the time that robot $\mathbf{a}_m$ spends servicing at location $\mathbf{n}_i$.

We define the binary decision variables x_{ijm} that indicate whether the route for robot $\mathbf{a}_m$ includes the arc $(\mathbf{n}_i, \mathbf{n}_j)$, such that

$$x_{ijm} = \begin{cases} 1, & \textit{route } m \textit{ contains the arc } (\mathbf{n}_i, \mathbf{n}_j) \\ 0, & \textit{otherwise.} \end{cases}$$

Optimization of the objective given in (1) *is subject to the following constraints:*

$$\sum_{i=2}^{N-1} s_{im} + \sum_{i=1}^{N-1} \sum_{j=2}^{N} t_{ijm} x_{ijm} \leq T_{max,m} \qquad \forall m = 1, \ldots, M \tag{2a}$$

$$\sum_{m=1}^{M} \sum_{j=2}^{N} x_{1jm} = \sum_{m=1}^{M} \sum_{i=1}^{N-1} x_{iNm} = M \tag{2b}$$

$$\sum_{i=1}^{N-1} x_{ikm} = \sum_{j=2}^{N} x_{kjm} \leq 1, \quad \forall m = 1, \ldots, M, \quad \forall k = 2, \ldots, (N-1) \tag{2c}$$

$$2 \leq u_{im} \leq N \quad \forall i = 2, \ldots, N, \quad \forall m = 1, \ldots, M \tag{2d}$$

$$u_{im} - u_{jm} + 1 \leq (N-1)(1 - x_{ijm}) \quad \forall i, j = 2, \ldots, N, \quad \forall m = 1, \ldots, M \tag{2e}$$

Constraint (2a) ensures each robot adheres to the budget constraint. Constraint (2b) ensures that each robot starts and ends at the designated depots. Constraint (2c) ensures the connectivity of all routes and that no robot will visit the same node twice. The Miller-Tucker-Zemlin subtour constraints [20] are enforced by (2d)-(2e) for all robot routes.

IV. Methodology

The cooperative multi-robot routing problem presented here is NP-hard, since its structure is based upon a Multiple TSP instance and the KP. Several efficient heuristics have been proposed to tackle such class of problems, and here we consider the use of evolutionary algorithms.

A. Evolutionary algorithm

Genetic Algorithms (GAs) are population-based meta-heuristics that mimic the evolution process found in nature [21]. GAs have shown to be a competitive approach for similar optimization problems considering the high-dimensional search spaces. Another advantage is that this technique presents an *anytime* behavior, *i.e.*, can return a valid solution before the execution ends, which might classify it also proper for on-line tasks. However, the solution tends to be improved along with more evolution time (generations).

A GA starts by initializing a population of individuals, where each individual represents a possible solution to the problem. Individuals are then evaluated to compute their fitness function, which is the value that the algorithm is trying to optimize. Next, a part of the existing population is used to generate an offspring generation through crossover (re-combining different individuals to create new individuals) and mutation (adding random changes into an individual to create a modified individual). This cycle continues until a stopping criteria is reached, as shown in Fig. 2.

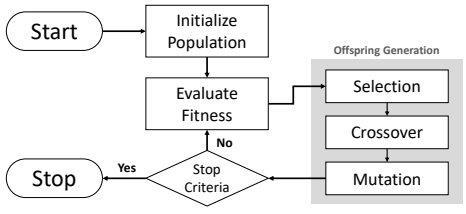


Fig. 2. Flow chart of a Genetic Algorithm.

B. Encoding, initialization, and fitness evaluation

A fundamental step in GAs is the selection of a good representation for the individuals that compose the population. An individual is described by a *chromosome*, which represents a candidate solution to the problem, in our case, for each robot a sequence of visits and the time it will spend servicing each location. This information is encoded in matrix form as shown in Fig. 3, where each column gives the route for a single robot. The row indices represent the locations in $\mathcal{N}$. An individual element of this matrix is called a *gene*.

	$\mathbf{a}_1$	$\mathbf{a}_2$		$\mathbf{a}_M$
$\mathbf{n}_1$	$\{\pi_{11}, s_{11}\}$	$\{\pi_{12}, s_{12}\}$	$\ldots$	$\{\pi_{1M}, s_{1M}\}$
$\mathbf{n}_2$	$\{\pi_{21}, s_{21}\}$	$\{\pi_{22}, s_{22}\}$	$\ldots$	$\{\pi_{2M}, s_{2M}\}$
		$\vdots$		
$\mathbf{n}_N$	$\{\pi_{N1}, s_{N1}\}$	$\{\pi_{N2}, s_{N2}\}$	$\ldots$	$\{\pi_{NM}, s_{NM}\}$

Fig. 3. Example of an individual's chromosome representing a solution, consisting of $N \times M$ genes, each defined by $\{\pi_{im}, s_{im}\}$.

The sequence in which the locations will be visited is determined by a random-key scheme as proposed by [22]. Each non-depot position has an associated parameter in the open unit interval $\pi_{im} \in (0,1) \forall i \in \{2, \ldots N-1\} \forall m \in \{1, \ldots M\}$, that defines the relative order in which it may be visited by robot m. For example, if a chromosome includes the values $\pi_{42} = 0.81$ and $\pi_{72} = 0.23$, that indicates that in the tour of robot $\mathbf{a}_2$, a visit to node $\mathbf{n}_4$ would be preceded by a visit to node $\mathbf{n}_7$. We initialize these values using a uniform random distribution. The start and goal depots have fixed values $\pi_{1m} = 0$ and $\pi_{Nm} = \infty$, respectively, ensuring their relative positions in all tours.

Every robot has a service cost $s_{im} \in [0, T_{\max,m})$ assigned for each location, which is used to determine the collected reward. We initialize service costs using the beta distribution: $s_{im}^{\text{init}} \sim \text{Beta}(\alpha, \beta) \times T_{\max,m}$. This distribution was chosen due to the ability to tune its' shape parameters. These parameters were chosen so that initial service costs would be in the interval $[0, T_{\max,m})$ but would be skewed away from $T_{\max,m}$.

The fitness of an individual is defined as the sum of the collected rewards by all agents. This value represents the argument of (1) which we are trying to maximize. The objective function is dependent on f_i, which determines the ratio of the maximum reward obtained. For the purposes of environmental monitoring, we are interested in maximizing the probability that our robots will capture data related to an event (*e.g.* marine life activity). Such a scenario can be modelled by the cumulative distribution function of an exponential distribution, which is typically used to model the time until an event occurs in memoryless processes [23]:

$$f_i(\sum_{m=1}^{M} s_{im}) = 1 - e^{-\beta_i \sum_{m=1}^{M} s_{im}},$$

where β_i represents the rate of decay of profit collection for the node, or the expected number of events per unit of time.

Since genetic algorithms improve based on the fitness function, our algorithm will work for other definitions of f_i as well. Furthermore, GAs do not require derivatives, therefore the same procedure would even apply for a noisy fitness function. Unlike traditional mixed-integer programs, GAs can find solutions for non-convex optimization problems, making them flexible enough to solve a wide array of scenarios.

C. Tour validation

The tours extracted from individuals encoded by the scheme above do not necessarily adhere to the budget constraint given in (2a). We introduce the following procedure, inspired by [24], that enforces the budget constraint.

In order to extract a solution from an individual's chromosome, first, for each robot $\mathbf{a}_m$, we order both the nodes and associated service costs based on the sorted order of π_{im}. This is equivalent to sorting each of the columns of the chromosome given in Fig. 3 independently.

Next, we calculate the cost of travelling and servicing all the nodes in the ordered tour. Finally, starting from the end of the tour, we flag nodes as invalid until the tour is within the budget, *i.e.*, $\sum_{i=2}^{N-1} s_{im} + \sum_{i=1}^{N-1}\sum_{j=2}^{N} t_{ijm}x_{ijm} \leq T_{\max,m} \forall m \in \{1,\dots,M\}$. Note that we maintain a flag for invalid nodes so that we can use the original service costs and a full set of N random key values during the offspring generation. The service cost for the last node in the tour is chosen such that the total travel plus service cost of the tour is exactly equal to $T_{\max,m}$.

D. Offspring generation

The evolution process is simulated by the successive improvement of the population, such that later generations of the population produce better solutions. Repeatedly, *genetic operators* are applied to a portion of the existing population to generate an offspring. There are three main types of operators: selection, crossover, and mutation. These operators stochastically select the more fit individuals and modify the chromosome by the recombination of genes from different individuals or by random changes intrinsic to the individual.

The offspring generation comprises the following steps:

1) Selection: The first step is the selection of some individuals in the population, which will be called *parents* and used later for breeding. Different approaches can be used, such as fitness-related (*e.g.* roulette wheel, where parents are selected with a probability proportional to their relative fitness in the population) or ordinal-based ones (*e.g.* tournament, where k_{tour} randomly selected individuals are placed in tournaments against each other and the individuals with the best fitness are selected from each tournament). We use a tournament selection method in our work.

2) Crossover: Next, the previously selected individuals are used as *parents* in the crossover step, which exploits the search space by creating new individuals based on the already known solutions (parents). We use a two point crossover approach. Two indices in $\{1,\dots,N\}$ are chosen randomly, these are designated as the crossover points $cp_1, cp_2 | cp_1 < cp_2$. Given two parent individuals, we combine their genes to produce two children. The first child is generated by taking the first cp_1 genes (across all robots) from the first parent, the next $cp_2 - cp_1$ genes are taken from the second parent, and the remaining genes are once again taken from the first parent. The second child is the mirror image of the first child.

3) Mutation: When generating the offspring population, an individual may be mutated with a probability of $p^{\text{i}}_{\text{mut}}$. For a mutated chromosome, genes will be mutated with a probability of $p^{\text{g}}_{\text{mut}}$. A mutated gene may change the tour ordering and the service time relative to the associated node. Tour ordering is mutated by adding an amount chosen from the normal distribution, *i.e.*:

$$\pi_{im} = \pi_{im} + n_\pi \mid n_\pi \sim \mathcal{N}(0, \sigma^2_{\text{mut}}) \ .$$

We do not allow $\pi_{im} \leq 0$ for $m \neq 1$, therefore we set any negative ordering values to $\epsilon \ll 1$.

Service times are mutated relative to the mean value of the beta distribution used to initialize them. The mean value of the initial service time values is given by:

$$\bar{s}_{\text{mut}} \equiv \mathbb{E}[\text{Beta}(\alpha,\beta) \times T_{\max,m}] = \frac{\alpha T_{\max,m}}{\alpha+\beta} \ ,$$

and the corresponding mutation is determined as:

$$s_{im} = s_{im} + n_s | n_s \sim \mathcal{N}(\bar{s}_{\text{mut}}, \sigma^2_{\text{mut}}) \ .$$

Since we do not allow $s_{im} < 0$, we set any negative service times to zero and mark their associated nodes as invalid in the tour. Note that we do not mutate the depot values.

4) Elitism: In order to preserve the best individuals and maintain the solution quality over generations, we make use of elitism. We select the k_{elite} individuals with the highest fitness from both current population and offspring, and enforce they are carried over into the next generation without modification.

5) Stopping Criteria: The previous steps repeat until a stopping criteria is reached. In this work we chose a constant number of generations for all experiments. The number of generations was chosen empirically so that the improvement in fitness function had sufficiently plateaued.

V. Experiments

The simulation framework was implemented with Python 3.7 and uses the DEAP [25] library. Table I summarizes the experimental parameters used for our algorithm. The parameters were initially selected based on the values used in others works tackling similar problems and they were fine-tuned empirically through a brief sensitivity analysis. We have also considered simple well-known guidelines in order to, for example, avoid not getting a good coverage of the search space or premature convergence.

Test instances were obtained from the authors of [17] so that they could be compared to the single robot case. These test cases have target locations $\mathbf{n}_i \in \boldsymbol{\mathcal{N}}$ randomly located in $\mathbb{R}^2$ and associated reward caps r_i as well as decay parameters β_i. Tests are derived from three sets of data: kroA, kroB, and kroC. For each test case, the number of nodes available is appended to the test name (*e.g.* kroA25 uses 25 nodes from the kroA test set). See [14], [17] for additional

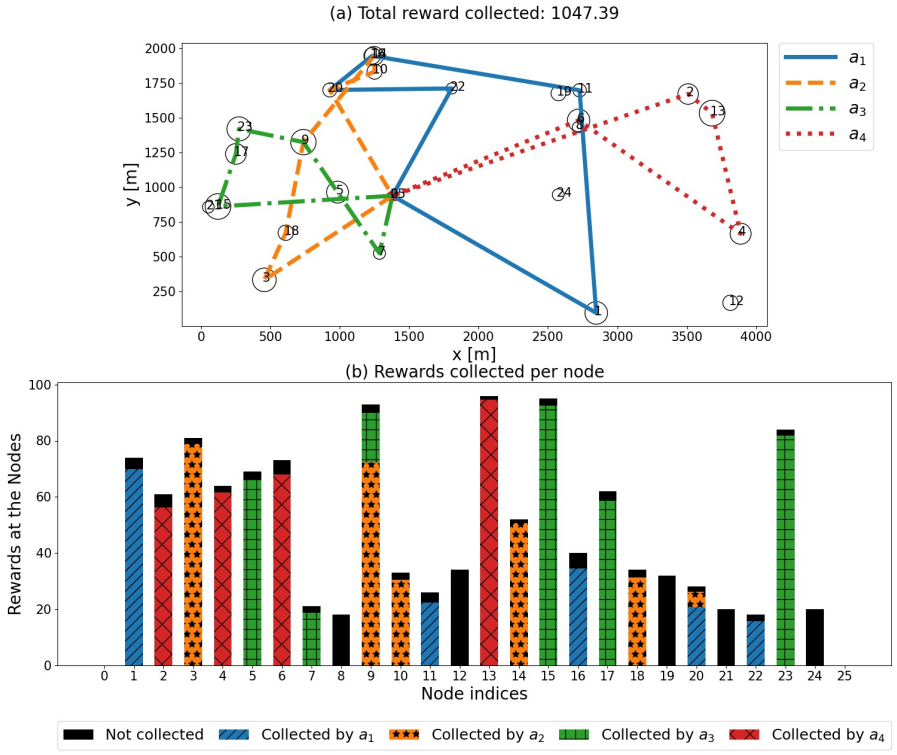


Fig. 4. (a) Illustration of the planned paths for 4 robots ($\mathbf{a}_1$, $\mathbf{a}_2$, $\mathbf{a}_3$, and $\mathbf{a}_4$) through 25 nodes from simulations on kroA25 dataset. Size of the disks at the nodes are proportional to the rewards associated with the node. (b) Bar chart showing the rewards collected by different robots at each of the nodes. Overlapping bars (for example on node 9) illustrate that two robots shared the rewards at that particular node.

TABLE I

PARAMETER VALUES USED IN EXPERIMENTS.

Parameter	**Symbol**	**Value**
Population size		400
Number of generations		200
Tournament size	k_{tour}	4
Elitism selection size	k_{elite}	40
Crossover probability	p_{cross}	0.8
Mutation probability (individual)	$p^{\text{i}}_{\text{mut}}$	0.5
Mutation probability (gene)	$p^{\text{g}}_{\text{mut}}$	0.2
Mutation variance	σ^2_{mut}	0.05
Beta distribution shape parameters	(α, β)	(0.5,2.5)

details on how these test cases were created. Additionally, we generated custom test cases to validate that the algorithm provides reasonable solutions to cases where there is an obvious optimal distribution of routes between robots.

A. Illustrative examples

Let us first consider an intuitive test case, depicted in Fig. 5. Nodes are evenly spaced around two circles equidistant from the depot and rewards are equally distributed. The optimal solution is for each robot to cover a different circle, spending equal amounts of time at the visited nodes. Over the course of 10 trials, our method achieves an average of 82% of the optimal solution, with a standard deviation of 4%. Figs. 5(b), (c) illustrate how the solution to this test case changes if one robot travels more quickly or if another robot is introduced.

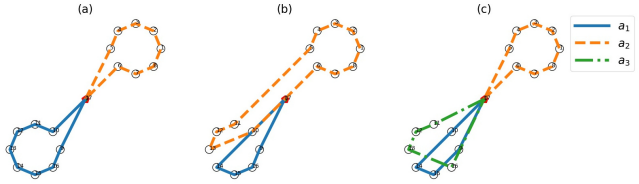


Fig. 5. (a) A test case where the optimal solution is known: each robot visits the nodes around a different circle, spending equal amounts of time at every node. (b) The second robot is able to travel at 3 times the speed as the first robot. (c) A third homogeneous robot is introduced.

Fig. 4 presents the output from one of the simulation runs on the test dataset kroA25. Fig. 4(a) illustrates the paths planned for four robots. The nodes are numbered from 0 to 25 and the size of the disks at the nodes indicate the rewards associated with the node. The rewards distributed among the nodes is collected by different robots and this reward sharing between the robots is demonstrated in the bar chart presented in Fig. 4(b). Fig. 4(b) also shows the least rewarding nodes (*e.g.*, nodes 8, 21 and 24) are not visited by any robot while others are visited by multiple robots (*e.g.*, nodes 9 and 20).

B. Numerical analysis

Initially, we show the evolution of the fitness function across different trials in Fig. 6, considering the test case shown in Fig. 4. As can be seen, the average result of our approach

is consistent, and the number of generations was chosen so that we expect to converge before the final generation.

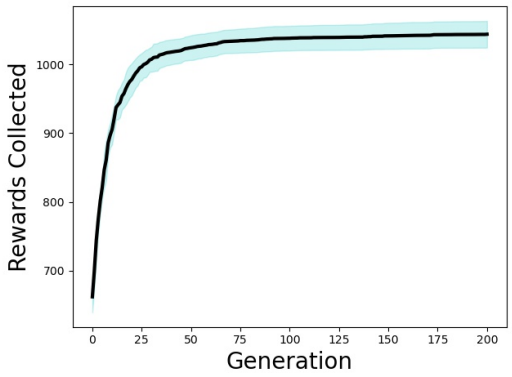


Fig. 6. Progression of the GA's fitness function (rewards collected) for the kroA25 dataset with 4 robots. The black line shows the average best fitness over 10 trials. The shaded region presents the standard deviation.

In the case of a single robot, Problem 1 is equivalent to that solved most recently by [17]. We evaluated our algorithm on all test cases kroA, kroB, and kroC with 25-100 nodes over the course of 10 trials for $M = 1$. For the single robot case, we find that the best results achieved by our method reaches $81 - 99\%$ of that presented in [17]. In Table II, we compare the rewards collected in 3 test cases by [17] to that achieved by our method. The table compares the best result reported in [17] to the best result from our method over 10 trials. Additionally we present the rewards accumulated when the increasing number of robots. Each additional robot adds budget capacity to the system. However, there are diminishing returns on adding more robots indefinitely since the most lucrative rewards are already collected in the single robot case. This trend is further illustrated by the exclusive budget scenario presented in Fig. 7.

TABLE II
REWARDS COLLECTED FROM OUR METHOD VS. MATHEURISTIC [17].

	# of Robots	kroA25	kroB25	kroC25
Yu et al. [17]	1	509.18	529.97	512.25
Our Method	1	491.6	495.1	511.41
	2	807.28	813.88	761.96
	3	948.25	1000.09	965.34
	4	1064.65	1104.85	1075.57
	5	1140.93	1170.54	1167.12
	6	1176.76	1227.6	1242.89
	7	1197.71	1250.78	1261.45
	8	1213.03	1264.51	1281.04
	9	1220.54	1272.52	1289.72
	10	1224.59	1274.88	1294.55

Next, we evaluate the performance of our algorithm by varying the number of robots in our planning approach. Fig. 7 shows the impact of increasing the number of robots on the team's overall performance. We consider two scenarios:

- **Shared Budget**: The total budget for the team ($T_{team} = T$) is fixed irrespective of the size of the team. This fixed budget is divided between all the robots in the team and each robot gets a budget of $T_{\max} = T/M$. If more robots are added to the team, then each robot ends up getting a smaller budget. For example, when a team has a fixed amount of fuel that must be divided among its members at deployment.
- **Exclusive Budget**: Each robot has its own budget ($T_{\max} = T$) which is not shared with other members in the team. If more robots are added to the team, the budget for individual robots remains the same, but the total team budget increases ($T_{team} = T \times M$). For example, when every robot has its own built-in battery, adding robots simply increases the total team budget.

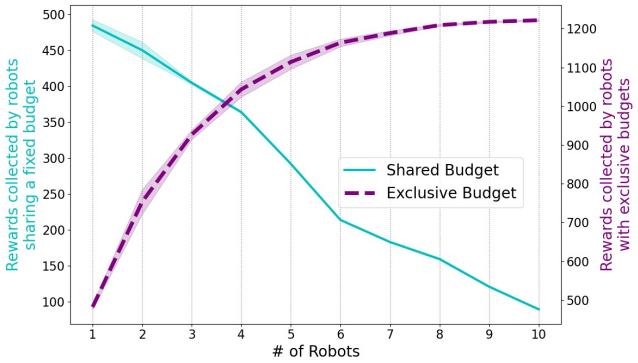
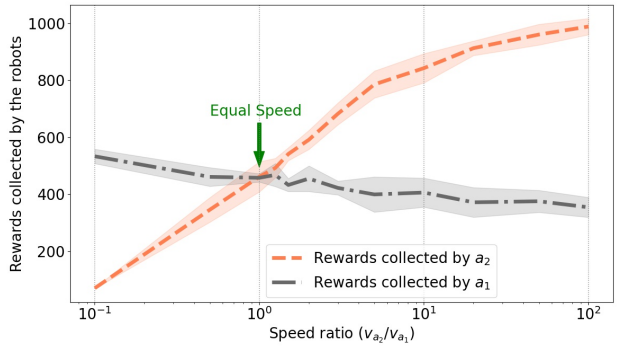


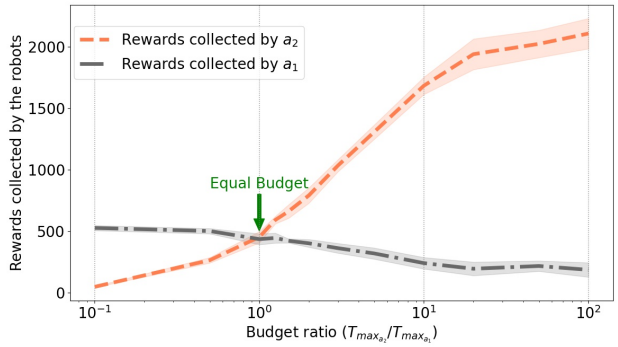
Fig. 7. Plots showing the changes in total rewards accumulated as the number of robots is increased (simulations on kroA25 testset). The shaded region presents the standard deviation over 10 trials.

Fig. 7 illustrates that increasing the team size is beneficial when every robot has its own budget. If the budget is shared among team members, then adding more robots to the team reduces the overall performance as it reduces each robot's budget. Shared budget forces all the robots to visit only a few nearby nodes. In the exclusive budget scenario, the purple dotted curve saturates, indicating that a team of size ≥ 9 collects most of the available rewards (1228.0). Similar trends were seen using other datasets (kroB25, kroC25, and kroA75).

We further investigate the effect of having a team of heterogeneous robots on the performance of our planning algorithm. Fig. 8 presents the comparison between the rewards shared by two robots operating at different travel speeds and with different budget allocation. We analyse the effect of varying the speed of one robot ($\mathbf{a}_2$) on the performance of the individual robots. As expected, Fig. 8(a) shows that increasing the speed of robot $\mathbf{a}_2$ allows it to collect more rewards. The performance of the constant speed robot, $\mathbf{a}_1$, decreases slightly due to the increased sharing of rewards with the faster robot, $\mathbf{a}_2$. These effects level out when the service time dominates the travel time. However, this levelling out is only observed when the speed ratio is above 10, which is beyond most practical applications. We see a similar trend when the robots have different budgets in Fig. 8(b). The robot with higher budget is able to collect more rewards. However, we see the saturation of the reward curve for robot $\mathbf{a}_2$ only after collecting higher rewards as the increase in budget gives both additional service time and travel time allowing the robot to collect more rewards. This curve is expected to completely saturate once all the rewards in the world are collected.



(a) Heterogeneity in travel speed



(b) Heterogeneity in budget allocation

Fig. 8. Reward sharing between two heterogeneous robots as the travel speed (a) or the allocated budget (b) of one of the robots ($\mathbf{a}_2$) is increased (simulations on kroA50 testset). Shaded region presents the standard deviation over 10 trials. Green arrows indicate where the robots are homogeneous.

VI. Conclusion and Future Work

We proposed a planning algorithm to design multi-robot paths over a fixed set of nodes such that the total information gathered is maximized within a fixed robot budget. We formulated the planning problem as a cooperative team orienteering problem with service-time dependent rewards and proposed the use of a genetic algorithm to solve it. We validated our algorithm on multiple test scenarios with varying size of the world, number of robots in the team, and heterogeneous robots with different speeds and budgets. Our findings suggest that adding more robots to the team improves the overall performance of the team if every robot is given its own exclusive budget. Our simulations show that having some robots with faster speeds can improve a team's performance until the service time dominates the travel time.

For future work, we plan to explore the impact of a more diverse set of heterogeneous robot capabilities on the team's performance. Additionally, we are interested in comparing the results of our method with those from deterministic approaches. Experimental validation using multiple autonomous surface vehicles for data collection and monitoring of marine life is another immediate direction for future work.

References

[1] A. Howard, M. J. Matarić, and G. S. Sukhatme, "Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem," in *Distributed Autonomous Robotic Systems 5*. Springer, 2002, pp. 299–308.

[2] A. Prorok, M. A. Hsieh, and V. Kumar, "Fast redistribution of a swarm of heterogeneous robots," in *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS)*, 2016, pp. 249–255.

[3] M. Santos, Y. Diaz-Mercado, and M. Egerstedt, "Coverage control for multirobot teams with heterogeneous sensing capabilities," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 919–925, 2018.

[4] S. Manjanna, H. Van Hoof, and G. Dudek, "Policy search on aggregated state space for active sampling," in *International Symposium on Experimental Robotics*. Springer, 2018, pp. 211–221.

[5] G. Laporte and S. Martello, "The selective travelling salesman problem," *Discrete Applied Mathematics*, vol. 26, no. 2, pp. 193 – 207, 1990.

[6] P. Vansteenwegen and A. Gunawan, *Orienteering problems: Models and algorithms for vehicle routing problems with profits*. Springer Nature, 2019.

[7] R. Pěnička, J. Faigl, P. Váňa, and M. Saska, "Dubins Orienteering Problem," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 1210–1217, April 2017.

[8] J. Yu, M. Schwager, and D. Rus, "Correlated Orienteering Problem and its Application to Persistent Monitoring Tasks," *IEEE Transactions on Robotics*, vol. 32, no. 5, pp. 1106–1118, 10 2016.

[9] Z. Ma, K. Yin, L. Liu, and G. S. Sukhatme, "A spatio-temporal representation for the orienteering problem with time-varying profits," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 6785–6792.

[10] I.-M. Chao, B. L. Golden, and E. A. Wasil, "The team orienteering problem," *European Journal of Operational Research*, vol. 88, no. 3, pp. 464–474, 2 1996.

[11] M. van der Merwe, J. P. Minas, M. Ozlen, and J. W. Hearne, "The cooperative orienteering problem with time windows," in *Optimization Online*, 2014.

[12] I. Roozbeh, M. Ozlen, and J. W. Hearne, "A heuristic scheme for the Cooperative Team Orienteering Problem with Time Windows," *CoRR*, 2016.

[13] L. Reyes-Rubiano, A. A. Juan, C. Bayliss, J. Panadero, J. Faulin, and P. Copado, "A Biased-Randomized Learnheuristic for Solving the Team Orienteering Problem with Dynamic Rewards," *Transportation Research Procedia*, vol. 47, pp. 680–687, 2020.

[14] G. Erdoğan and G. Laporte, "The orienteering problem with variable profits," *Networks*, vol. 61, no. 2, pp. 104–116, 2013.

[15] J. Yu, J. Aslam, S. Karaman, and D. Rus, "Anytime planning of optimal schedules for a mobile sensing robot," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 5279–5286.

[16] D. Gavalas, C. Konstantopoulos, K. Mastakas, G. Pantziou, and N. Vathis, "Heuristics for the time dependent team orienteering problem: Application to tourist route planning," *Computers & Operations Research*, vol. 62, pp. 36 – 50, 2015.

[17] Q. Yu, K. Fang, N. Zhu, and S. Ma, "A matheuristic approach to the orienteering problem with service time dependent profits," *European Journal of Operational Research*, vol. 273, no. 2, pp. 488–503, 2019.

[18] S. Hanafi, R. Mansini, and R. Zanotti, "The multi-visit team orienteering problem with precedence constraints," *European Journal of Operational Research*, vol. 282, no. 2, pp. 515 – 529, 2020.

[19] T. Sakamoto, S. Bonardi, and T. Kubota, "A Routing Framework for Heterogeneous Multi-Robot Teams in Exploration Tasks," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6662–6669, 2020.

[20] C. E. Miller, A. W. Tucker, and R. A. Zemlin, "Integer programming formulation of traveling salesman problems," *Journal of the ACM (JACM)*, vol. 7, no. 4, pp. 326–329, 1960.

[21] M. Mitchell, *An Introduction to Genetic Algorithms*. Cambridge, MA, USA: MIT Press, 1998.

[22] J. C. Bean, "Genetic Algorithms and Random Keys for Sequencing and Optimization," *ORSA Journal on Computing*, vol. 6, no. 2, pp. 154–160, 1994.

[23] J. K. Blitzstein and J. Hwang, *Introduction to probability*. Chapman and Hall/CRC, 2019.

[24] T. Ilhan, S. M. Iravani, and M. S. Daskin, "The orienteering problem with stochastic profits," *Iie Transactions*, vol. 40, no. 4, pp. 406–421, 2008.

[25] F.-A. Fortin, F.-M. De Rainville, M.-A. Gardner, M. Parizeau, and C. Gagné, "DEAP: Evolutionary algorithms made easy," *Journal of Machine Learning Research*, vol. 13, pp. 2171–2175, jul 2012.