Team Orienteering Problem with Communication Constraints using Surrogate Model

Marco Túlio P. T. Tristão Douglas G. Macharet

***Abstract*— Multi-Robot Systems (MRS) are increasingly utilized in applications such as surveillance, environmental monitoring, and search and rescue, where maximizing mission rewards under budget constraints is critical. The Team Orienteering Problem (TOP) provides a framework for optimizing task coverage and resource allocation in such scenarios. However, traditional TOP formulations often overlook real-world constraints, such as limited communication ranges and the necessity of persistent connectivity among robots. These constraints are particularly relevant in environments like disaster zones and remote areas, where communication infrastructure is unreliable or absent. To address this gap, we propose a multi-objective formulation that balances task coverage, communication quality and energy expenditure under a fixed budget. Our approach accommodates teams of any size and heterogeneous vehicles with varying velocities and constant thrust. We validate our approach through extensive experiments across diverse scenarios and team configurations.**

I. Introduction

Multi-Robot Systems (MRS) play a vital role in various applications, such as search and rescue, environmental monitoring and autonomous surveillance. These tasks often require robots to manage resource constraints, such as operational costs and energy consumption, while navigating through the environment. Beyond these restrictions, in cooperative missions, inter-robot communication is essential. For example, in disaster zones where delivering critical resources is essential, robots must maintain communication to detect failures in real time and reallocate tasks to ensure mission success. Thus, it is very important to plan agents' routes while ensuring continuous inter-agent connectivity.

In this context, determining the most efficient task schedule for a robot is crucial. This problem can be formulated as the Orienteering Problem (OP) [1], where the objective is to find an optimal sequence of tasks while considering constraints such as the robot's limited travel budget and the reward associated with each task. Since the budget constraint may prevent the robot from visiting all available tasks (*e.g.*, locations), the problem also involves selecting an optimal subset of nodes to visit. A well-known special case of the OP is the Traveling Salesperson Problem, which is NP-hard. Consequently, solving the OP is also NP-hard.

When multiple agents are involved, the problem extends to the Team Orienteering Problem (TOP) [2], where each team member must efficiently select and schedule a subset of tasks while coordinating with others.

This work was supported by CAPES/Brazil - Finance Code 001, CNPq/Brazil, and FAPEMIG/Brazil.

The authors are with the Computer Vision and Robotics Laboratory (VeRLab), Department of Computer Science, Universidade Federal de Minas Gerais, Brazil. E-mails: marcotuliopin@ufmg.br, doug@dcc.ufmg.br.

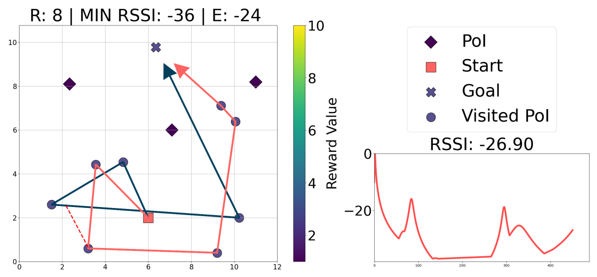

Fig. 1: Example of a two-vehicle team navigating multiple task locations, maintaining a worst-case RSSI value (related to the maximum inter-agent distance – red dashed line) of -36.91 dBm. Video: https://youtu.be/wqPwqnb0mzY.

In this paper, we explore the Team Orienteering Problem with Communication Constraints (TOP-CC), where a team of autonomous robots must maximize the collected reward while operating within limited travel budgets, optimizing energy consumption and inter-agent communication (Fig. 1). We follow a multi-objective optimization framework that balances three key objectives: (i) task coverage, (ii) energy efficiency, and (iii) connectivity preservation. This formulation captures the trade-offs between reward collection, network reliability and path efficiency. To tackle this challenge, we develop a hybrid metaheuristic task allocation and path-planning algorithm that optimizes all three objectives simultaneously. Our approach provides a scalable and practical solution for real-world multi-robot applications where both mobility and communication are critical to mission success.

Previous research has extensively explored the TOP and its variants, incorporating factors such as motion constraints [3], fault tolerance [4], and uncertain rewards [5]. However, existing studies mostly overlook the necessity of preserving the communication between agents throughout the mission. Alternatively, those that do consider connectivity typically rely on a leader-follower approach, where a single leader dictates the routes of the other agents [6]. In contrast, our work incorporates communication constraints directly into the TOP framework, ensuring that agents not only optimize their routes for efficiency and reward collection, but also actively optimize decentralized communication.

II. Related Work

The Orienteering Problem (OP) [1], is a widely studied problem in logistics and robotics [7]. Over the years, different variants of this problem have been proposed to better fit real-world scenarios. Angelelli *et al.*. [8] address the Clustered Orienteering Problem (COP), where all tasks within a cluster must be visited to collect the reward. In contrast, the Set Orienteering Problem (SOP) [9] allows rewards to be collected by visiting at least one node in a group.

Additional variants of the OP include the Probabilistic OP [10], that incorporates uncertainty about the task availability, and the Orienteering Problem with Time Windows (OPTW) [11].

In our application, the most relevant variant of the OP is the Team Orienteering Problem (TOP) [2]. This problem integrates multi-robot task allocation and path planning while offering a framework to account for communication constraints. To solve the TOP, El-Hajj *et al.* [12] proposes a cutting plane method, which solves smaller sub-problems to extract useful information before gradually addressing the original problem. The algorithm proposed by Ke *et al.* [13] maintains a population of current solutions, which are iteratively refined using Pareto dominance principles. Their method also employs a mimic operator, which generates new solutions by emulating existing ones.

Another approach, presented by Xu *et al.* [14], focuses on an approximation algorithm for the TOP. Additionally, Mansfield *et al.* [15] demonstrated the effectiveness of genetic algorithms in solving the TOP. More recently, Li *et al.* [16] developed an approach for the TOP with interval-varying profits using a Branch-Price-and-Cut algorithm.

Several variants of the TOP have been proposed to address different real-word challenges. The Time-Dependent TOP accounts for dynamic travel times [17], where the timing of each visit is a critical factor in the route optimization. Mansfield *et al.* [18] consider the influence of flow currents on the vehicle's movement. Yahiaoui *et al.* [19] extend the Set OP to propose the Clustered TOP, where a group vehicles collaborates to visit customers within the clusters efficiently.

In optimization problems with expensive objective functions, surrogate models have emerged as an effective strategy to accelerate evaluation. Surrogate models approximate the true objective function using a learned representation, enabling faster convergence in metaheuristic algorithms.

In this work, we employ an adapted multi-objective metaheuristic to iteratively refine the non-dominated set, following the methodologies of [13], [15]. Unlike [6], which utilizes a leader-follower strategy, our approach simultaneously plans all agents routes, ensuring a fully coordinated and globally optimized solution.

III. Problem formulation

A. Team and Environment Models

Given a set of N task locations, denoted as $\mathcal{V} = \{v_1, v_2, \dots, v_N\}$, where each location v_i is associated with a non-negative reward r_i. These task locations are spatially distributed within a 2D environment $\mathbb{W} \subset \mathbb{R}^2$ and may have different reward values. A team of M autonomous vehicles, denoted as $\mathcal{A} = \{a_1, a_2, \dots, a_M\}$, operates within this environment. Each vehicle a_i route starts from a common initial location v_1 and finish at an ending location v_N. Additionally, each vehicle has a maximum allowable travel budget b_i, forming the set $\mathcal{B} = \{b_1, b_2, \dots, b_M\}$, and a constant speed s_i, forming the set $\mathcal{S} = \{s_1, s_2, \dots, s_M\}$.

B. Optimization Objectives and Constraints

The general objective is threefold: (i) to maximize the total accumulated reward, (ii) to enhance inter-agent communication efficiency, and (iii) to minimize overall energy consumption. We formally define these objectives as follows.

The objective related to the *total reward* is given by:

$$\max \sum_{i \in \mathcal{A}} \sum_{j \in \pi_i} r_j y_{ij} \ , \tag{1}$$

where π_i represents the ordered sequence of locations visited by vehicle a_i, r_j is the reward associated with task location v_j, and y_{ij} is a binary variable indicating whether vehicle a_i visits location v_j.

Since energy consumption is directly related to the distance traveled, to minimize energy consumption, we consider the *average path length* of all vehicles, *i.e.*:

$$\min \frac{1}{M} \sum_{i \in \mathcal{A}} \sum_{j=1}^{|\pi_i|-1} d(v_j, v_{j+1}) \ , \tag{2}$$

where $d(v_j, v_{j+1})$ represents the Euclidean distance between consecutive locations in the route.

Additionally, we introduce *communication efficiency* as an optimization objective. We model this constraint by maximizing the minimum Received Signal Strength Indicator (RSSI), which occurs when the two most distant vehicles reach their farthest separation point, determined as:

$$\max \min_{i,j \in \mathcal{A}, i \neq j} \text{RSSI}(a_i, a_j) \ , \tag{3}$$

with

$$\text{RSSI}(a_i, a_j) = P_t - 10 \cdot \gamma \cdot \log_{10}(d(a_i, a_j)) \ , \tag{4}$$

where P_t is the transmission power and γ is the path loss exponent.

Given x_{ijm} a binary variable indicating whether agent a_m travels along the path from v_i to v_j, and y_{im} a binary variable indicating whether the agent a_m visits the task v_i, this problem is subject to the following constraints:

$$\sum_{j \in \mathcal{V}, j \neq v_1} x_{1jm} = \sum_{j \in \mathcal{V}, j \neq v_1} x_{1Nm} = 1, \quad \forall m \in \mathcal{A} \tag{5a}$$

$$\sum_{j \in \mathcal{V}, j \neq i} x_{ijm} = y_{im}, \quad \forall i \in \mathcal{V}, \forall m \in \mathcal{A} \tag{5b}$$

$$\sum_{i \in \mathcal{V}, i \neq j} x_{ijm} = y_{jm}, \quad \forall j \in \mathcal{V}, \forall m \in \mathcal{A} \tag{5c}$$

Constraint (5a) ensures that every route starts at v_1 and ends at v_N, while constraints (5b)-(5c) enforce all routes to be fully connected.

Furthermore, each individual route must respect the agent's maximum travel budget (*e.g.*, distance traveled):

$$\sum_{i \in \mathcal{V}} \sum_{j \in \mathcal{V}, j \neq i} d(v_i, v_j) x_{ijm} \leq b_m, \quad \forall m \in \mathcal{A} \ , \tag{6}$$

and each node must be visited at most once:

$$\sum_{m \in \mathcal{A}} y_{im} \leq 1, \quad \forall i \in \mathcal{V} \ . \tag{7}$$

Finally, the solutions follow the Miller-Tucker-Zemlin subtour elimination formulation [20]:

$$\begin{aligned} u_i - u_j + 1 \leq (|\mathcal{V}| - 1)(1 - x_{ijm}), \\ \forall i, j \in \mathcal{V}, i \neq j, \forall m \in \mathcal{A} \end{aligned} \tag{8a}$$

$$1 \leq u_i \leq |\mathcal{V}| - 1, \quad \forall i \in \mathcal{V}, i \neq v_1, v_N \ . \tag{8b}$$

where u_i represents the relative position of node v_i in a vehicle's path.

C. Combined Task Visit

We also propose an extension that, unlike other TOP formulations, permits multiple robots to visit the same node, while ensuring that its reward is considered only once.

Since we account for team proximity, an agent may choose to visit a task already being attended by another agent, prioritizing team cohesion and thereby improving communication. This variant is defined by the following objective equation:

$$\max \sum_{i \in \mathcal{V}} r_i \left(\max_{m \in \mathcal{A}} y_{im} \right) \ . \tag{9}$$

IV. Methodology

The Variable Neighborhood Search (VNS) [21] is a metaheuristic optimization technique that explores multiple neighborhood structures through local search. Despite its effectiveness, the original VNS lacked mechanisms for handling multiple objectives. To overcome this limitation, a multi-objective extension was introduced in [22]. In this approach, a solution is randomly selected at each iteration, and neighborhood structures are applied sequentially to explore the search space. The resulting solutions are then evaluated and used to update the non-dominated set, ensuring a diverse and optimal trade-off among objectives. We adapted the Multi-Objective VNS (MOVNS) [22] to our problem. A high-level overview of the adapted algorithm is presented in Alg. 1.

Algorithm 1: Multi-Objective Optimization VNS

```
   Input: Problem instance, stopping condition
   Output: Solution archive
 1 Generate initial archive
 2 while stopping condition is not satisfied do
 3     $S$ ← Select a solution
 4     foreach neighborhood do
 5         $S'$ ← Perturb solution $S$
 6         Neighbors ← Local search on $S'$
 7         Update archive with Neighbors
 8     end
 9 end
10 return archive
```

A. Solution Encoding

Each solution is encoded as a matrix of dimensions $M \times (N-2)$, where each *row* corresponds to an agent's path, and each *column* represents a task, excluding the initial and final depots. The matrix values range from -1 to 1, where *negative values* indicate that the corresponding task is not visited, while *positive values* denote the visit order of the task within the agent's path.

After generating a new solution, it undergoes a validation process to ensure that each path π_i adheres to the respective budget b_i. If an agent's path exceeds its budget, nodes are iteratively removed until the constraint is satisfied.

B. Neighborhood Operators

In our approach, neighborhoods are explored sequentially, with the total number determined by the product of local search and perturbation operators. Each perturbation operator is applied once per path to generate a new solution, while local search operations create neighboring solutions.

C. Computational Bottleneck

The primary computational cost in our baseline MOVNS stems from evaluating inter-agent communication quality. Computing the exact minimum RSSI between agents requires:

1) Dense temporal interpolation of all agent trajectories at discrete time steps
2) Pairwise distance calculations $d(a_i(t), a_j(t))$ between all agent pairs at each time step t
3) RSSI computation using the propagation model
4) Identification of the minimum RSSI across all time steps and agent pairs

For the multi-vehicle visit case, this results in a time complexity per solution evaluation of:

$$O(M^4 N^3), \tag{10}$$

where M is the number of agents and N is the number of task locations. For the single-vehicle visit case, path lengths are bounded by N/M, reducing complexity to $O(M^2 N^3)$.

Since the MOVNS algorithm evaluates thousands of candidate solutions during optimization, this exact RSSI computation becomes the dominant computational bottleneck, particularly for large teams and task sets.

D. Surrogate Model for Accelerated Evaluation

To address this bottleneck, we develop a neural network surrogate that efficiently ranks solutions by their communication quality without computing exact RSSI values. The surrogate model is trained using the RankNet [23] approach, a pairwise ranking method that learns to predict relative solution quality rather than absolute RSSI values.

1) Data Collection and Ground Truth: We generated a dataset by running multiple instances of the MOVNS algorithm across various problem configurations. For each solution in the archive, we computed the ground truth minimum RSSI.

This process yielded a dataset of solution instances paired with their true minimum RSSI values, which we used to train our ranking model.

2) RankNet Training Strategy: Rather than training the model to predict absolute RSSI values through standard regression, we employ the RankNet pairwise ranking approach [23].

The training procedure operates as follows:

1) Sample two instances A and B from the training dataset
2) Perform forward passes to obtain scores $s_A = g_\theta(A)$ and $s_B = g_\theta(B)$, where g_θ is our neural network with parameters θ
3) Define the ground truth label:

$$\bar{P}_{AB} = \begin{cases} 1 & \text{if RSSI}(A) > \text{RSSI}(B) \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

4) Compute the predicted probability using sigmoid:

$$P_{AB} = \sigma(s_A - s_B) \tag{12}$$

5) Calculate the binary cross-entropy loss:

$$\mathcal{L} = -\bar{P}_{AB}\log(P_{AB}) - (1-\bar{P}_{AB})\log(1-P_{AB}) \tag{13}$$

6) Update parameters θ via backpropagation

This training regime encourages the model to learn relative orderings rather than absolute values, resulting in a more robust ranking function that generalizes better across different problem scales and configurations.

a) Input Representation: Each problem instance is represented as a set of agent paths $\{P_1, P_2, \ldots, P_M\}$, where path $P_m = \{(x_1^m, y_1^m), (x_2^m, y_2^m), \ldots, (x_{T_m}^m, y_{T_m}^m)\}$ contains the coordinates of task locations visited by agent m. Each agent also has an associated speed s_m. The input thus consists of:

- A variable number of agents M
- Variable-length paths T_m for each agent
- Coordinate pairs $(x, y) \in \mathbb{R}^2$ representing visited task locations
- Scalar speed values $s_m \in \mathbb{R}^+$ for each agent

b) Path Encoding via LSTM: Since paths have variable lengths, we process them through a recurrent architecture. Each path is padded to a maximum length $T_{\max}$ and encoded using a Long Short-Term Memory (LSTM) network [24].

The LSTM processes the sequence of coordinate pairs $\mathbf{x}_t = (x_t, y_t)$ for $t = 1, \ldots, T_m$, maintaining internal memory states that capture sequential dependencies along the trajectory. After processing all waypoints, we extract the final hidden state as a fixed-dimensional embedding $\mathbf{h}_m = \mathbf{h}_{T_m} \in \mathbb{R}^{d_h}$, where d_h is the LSTM hidden dimension.

c) Speed Integration: Inter-agent distances at any time t depend not only on the paths themselves but also on how quickly agents traverse them.

To incorporate this temporal information, each path embedding is concatenated with the agent's speed and processed through a multi-layer perceptron (MLP):

$$\mathbf{e}_m = \text{MLP}_1([\mathbf{h}_m; s_m]) \in \mathbb{R}^{d_e}, \tag{14}$$

where d_e is the embedding dimension.

d) Inter-Agent Interaction Encoding: To capture the pairwise spatial relationships, we apply a Transformer layer [25] with multi-head self-attention to the set of agent embeddings.

Let $\mathbf{E} = [\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_M]^\top \in \mathbb{R}^{M \times d_e}$ be the matrix of agent embeddings. For each attention head, we compute:

$$\mathbf{Q} = \mathbf{E}\mathbf{W}^Q, \quad \mathbf{K} = \mathbf{E}\mathbf{W}^K, \quad \mathbf{V} = \mathbf{E}\mathbf{W}^V \tag{15}$$

$$\text{head}_h = \text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V} \tag{16}$$

where $\mathbf{W}^Q, \mathbf{W}^K, \mathbf{W}^V$ are learned projection matrices and d_k is the key dimension. The multi-head attention combines H parallel attention mechanisms:

$$\mathbf{E}' = \text{MultiHead}(\mathbf{E}) = \text{Concat}(\text{head}_1, \ldots, \text{head}_H)\mathbf{W}^O, \tag{17}$$

where $\mathbf{W}^O$ is an output projection matrix. The attention mechanism allows each agent embedding to attend to all other agents, learning representations that encode relative positions and potential communication bottlenecks.

Following the DeepSets framework [26], we achieve permutation invariance through a symmetric aggregation operation:

$$\mathbf{z} = \max_{m=1}^{M} \mathbf{e}'_m \in \mathbb{R}^{d_e}, \tag{18}$$

where the max operation is applied independently to each dimension. Since $\max$ is a symmetric function, this ensures that $\mathbf{z}$ is invariant to agent ordering.

e) Output Layer: The aggregated embedding is passed through a final MLP to produce the scalar score:

$$s = \text{MLP}_2(\mathbf{z}) \in \mathbb{R}. \tag{19}$$

This score has no direct physical interpretation, but is trained such that higher scores correspond to better communication quality. During optimization, solutions are ranked by their scores, and only relative ordering matters for archive management and solution selection.

3) Architecture Instantiation: In our implementation, the LSTM encoder uses a hidden dimension of $d_h = 64$ with 2 layers to process agent path sequences. The Transformer layer employs $H = 2$ attention heads to capture inter-agent relationships. We train the model using the Adam optimizer [27] with a learning rate of $1e-3$, processing data in batches of 32 examples over 100 epochs. Our dataset comprises 100,000 solution instances split into 60% for training, 20% for validation, and 20% for testing.

E. Complexity Analysis

The surrogate model replaces the $O(M^4N^3)$ exact RSSI computation with a forward pass through the neural network. The complexity of this forward pass is:

$$\text{LSTM encoding:} \quad O(M \cdot T \cdot d_h^2) \tag{20}$$

$$\text{Speed integration:} \quad O(M \cdot d_e \cdot d_{mlp}) \tag{21}$$

$$\text{Transformer layer:} \quad O(M^2 \cdot d_e) \tag{22}$$

$$\text{Aggregation + Output:} \quad O(M \cdot d_e + d_{mlp}) \tag{23}$$

where T is the average path length, d_h is the LSTM hidden dimension ($d_h = 64$), d_e is the embedding dimension, and d_{mlp} represents MLP hidden dimensions. Since these dimensions are small constants ($d_h, d_e, d_{mlp} \ll N$), and $T \approx N/M$ in practice, the surrogate evaluation complexity simplifies to:

$$O(M \cdot N \cdot d_h^2 + M^2 \cdot d_e) \approx O(MN). \tag{24}$$

Table I summarizes the computational complexity comparison:

TABLE I: Computational complexity comparison between exact RSSI evaluation and surrogate model.

Method	Multi-vehicle	Single-vehicle
Exact RSSI	$O(M^4N^3)$	$O(M^2N^3)$
Surrogate Model	$O(MN)$	$O(MN)$

For large teams ($M \geq 5$) with many tasks ($N \geq 50$), the surrogate model provides substantial speedup. This enables the algorithm to evaluate significantly more candidate solutions within the same computational budget, potentially leading to improved solution quality.

V. Experiments

We evaluated our method on both benchmark datasets and custom-designed problem instances. Table II presents the MOVNS parameters used in our experiments. The algorithm was implemented in Python 3.12.

TABLE II: Algorithm parameters.

Parameter	Value	Parameter	Value
Transmission Power (P_t)	-30	Path Loss Exp. (γ)	2
Maximum Iterations	300	Archive Size (A)	40
β	0.9	α	0.75

Experiments ran on a 13th Gen Intel i7-13620H (2.40 GHz) with 24 GB RAM, using WSL2 Ubuntu 22.04.

A. Experimental Design

We compare two variants of the MOVNS algorithm:

- **Baseline MOVNS:** Uses exact RSSI computation for solution evaluation
- **Surrogate MOVNS:** Uses the trained neural network surrogate for ranking solutions during optimization

Both variants use identical neighborhood operators, selection strategies, and stopping criteria to ensure a fair comparison. The surrogate model is trained once on data collected from baseline MOVNS runs and then deployed without further updates during optimization.

B. Model Training Losses

Figure 2 shows the training and validation loss curves over 12 epochs. The model demonstrates excellent convergence, with both training and validation losses decreasing consistently. Notably, the final training loss (0.0284) and validation loss (0.0272) are nearly identical, with a train-to-validation ratio of 1.044. This indicates that the model generalizes well to unseen data without overfitting.

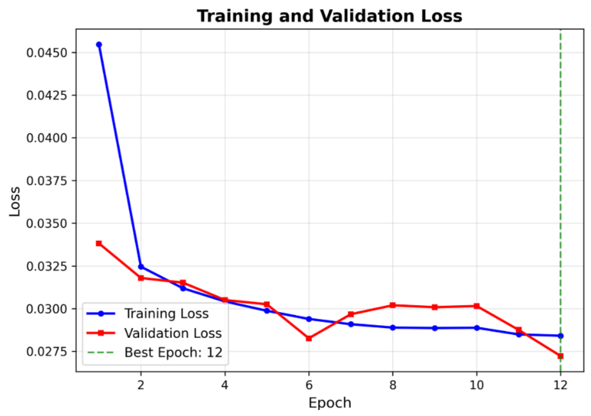

Fig. 2: Train and validation loss for the surrogate model use for calculation of the minimum RSSI in an instance.

C. Computational Efficiency Analysis

We analyze the runtime characteristics of both methods across a range of team sizes M while keeping the number of environment samples fixed at $N = 100$.

Table III reports the empirical runtime per evaluation for both the exact model and the surrogate across increasing team sizes. As expected, the cost of the exact computation grows rapidly with M because it must evaluate all NM pairwise robot–environment interactions. In contrast, the surrogate maintains a significantly flatter growth profile, reflecting its reduced computational complexity. The resulting empirical speedups increase monotonically with M and reach up to $4.58\times$ for $M = 50$, confirming the scalability advantage of the surrogate approach.

TABLE III: Computational time comparison (seconds per evaluation) Results for $N = 100$, $M \in 2, 3, 5, 10, 20, 30, 50$

M	Exact (s)	Surrogate (s)	Speedup
2	387.16	318.74	$1.21\times$
3	526.30	383.54	$1.37\times$
5	646.48	452.15	$1.43\times$
10	1028.00	579.84	$1.77\times$
20	1513.78	619.74	$2.44\times$
30	2311.92	681.10	$3.39\times$
50	4112.59	898.52	$4.58\times$

Figure 3 complements these empirical findings. While the absolute speedup values are lower than the theoretical speedup model due to GPU scheduling overhead and constant-time effects not reflected in the theoretical model, the observed curve is qualitatively consistent. In particular, the surrogate becomes strictly faster than the exact method already at $M \geq 2$ and the speedup increases steadily thereafter, matching the predicted monotonic growth.

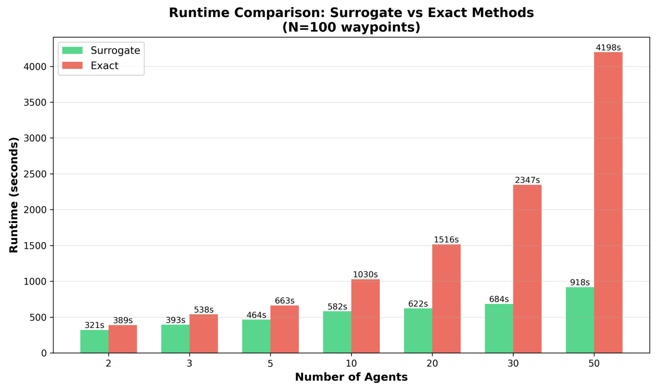

Fig. 3: Instance runtime for the surrogate model and the exact models as a function of team size M, for $N = 100$.

D. Communication Quality Tradeoff

We analyze the degradation of communication quality when using the surrogate model instead of an exact calculation method. Figure 4 shows a scatterplot where each grey dot is a problem solution. We used benchmark problems from Chao problem set [2], aswell as problems used in the timing benchmark to test the RSSI quality. We analyze that as problem size increases, instances have a minimal RSSI degradation combined with substantial time savings.

E. TOP benchmark instances

We also evaluated our method using the TOP benchmark problem set introduced by Chao *et al.*. [2], executing instances from sets 1–5, totaling 278 problems. The number of agents varies between 2 and 4, while the number of reward locations N ranges from 21 to 100.

Table IV shows a comparison of our methods in terms of reward collected, execution time and agent connectivity in benchmark instances.

The results demonstrate that the surrogate distance model successfully preserves solution quality while providing meaningful computational benefits across all problem instances. Both exact and surrogate methods achieve identical average rewards, confirming that the learned model maintains the primary optimization objective without compromising waypoint coverage performance. The surrogate model delivers consistent computational speedups across all problem sizes, with particularly impressive gains of 8-20% on smaller instances and sustained 4-5% improvements even on the largest problems. The trade-off for these computational gains is a modest and predictable reduction in communication quality (RSSI), which decreases from 9-12% on smaller

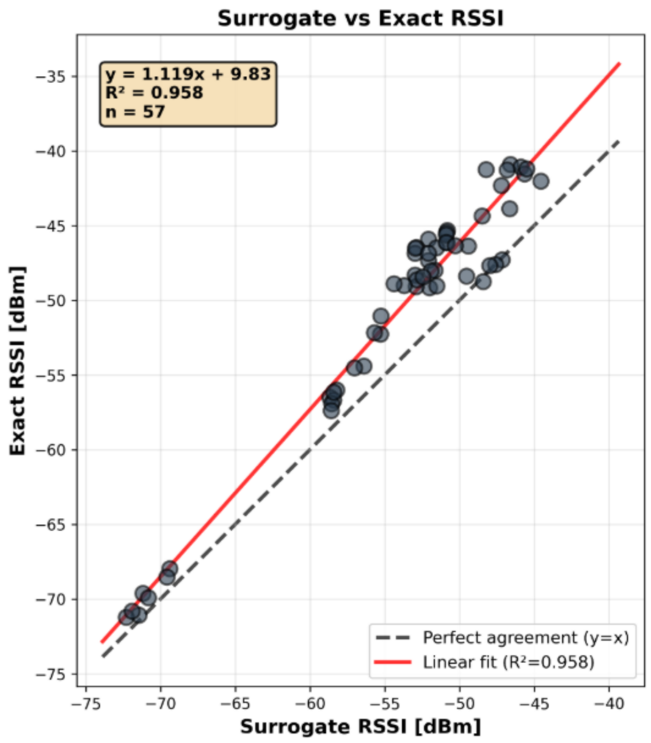

Fig. 4: Speedup factor of the surrogate model as a function of team size M, for $N = 100$.

problems to just 4% on larger instances, suggesting that the learned model's approximations have less relative impact on complex scenarios.

VI. Conclusion and Future Work

When working with multiple autonomous agents, it is crucial to optimize both routing and task allocation, as addressed in the Team Orienteering Problem (TOP), while simultaneously considering constraints such as energy consumption and communication range. This type of problem is particularly critical in real-world scenarios, including environmental monitoring and search-and-rescue operations.

We introduced the TOP with Communication Constraints, where a team of autonomous robots must optimize reward collection while maintaining continuous inter-agent connectivity. Unlike other approaches, our method explicitly incorporates communication constraints into the path-planning process, ensuring that agents remain connected throughout the mission. To solve this multi-objective problem for heterogeneous teams, we develop a hybrid VNS-based metaheuristic that integrates both task allocation and trajectory optimization, using a surrogate model to approximate the communication quality calculation.

Experimental results on benchmark instances show that our approach achieves significant reward collection while effectively adapting to varying communication quality constraints. Additionally, the approximate communication model speeds up the algorithm without degrading reward collection, and with little tradeoff in RSSI score.

TABLE IV: Performance comparison between exact and surrogate distance models on Chao problem sets (p1–p5) [2].

Set	N	Exact			Surrogate		
		avg. reward	avg. time (s)	avg. rssi	avg. reward	avg. time (s)	avg. rssi
p1	32	285.0	63.2	-46.4	285.0	52.9	-51.8
p2	21	450.0	17.7	-42.0	450.0	17.2	-46.6
p3	33	800.0	68.1	-48.3	800.0	54.6	-52.9
p4	100	1306.0	305.0	-55.3	1306.0	291.7	-57.5
p5	66	1680.0	302.2	-47.8	1680.0	287.1	-49.6

In future work, we plan to explore vehicles with variable speeds to better model realistic teams and develop new strategies for maintaining inter-agent connectivity. Additionally, we aim to investigate the integration of curvilinear trajectories, allowing agents to adapt their paths for improved proximity maintenance and more efficient coordination.

REFERENCES

[1] B. L. Golden, L. Levy, and R. Vohra, "The orienteering problem," *Naval Research Logistics (NRL)*, vol. 34, no. 3, pp. 307–318, 1987.

[2] 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.

[3] C. Bayliss, A. A. Juan, C. S. Currie, and J. Panadero, "A learnheuristic approach for the team orienteering problem with aerial drone motion constraints," *Applied Soft Computing*, vol. 92, p. 106280, 2020.

[4] R. F. Santos, E. R. Nascimento, and D. G. Macharet, "Anytime Fault-tolerant Adaptive Routing for Multi-Robot Teams," in *Proc. of IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 7936–7942.

[5] B. Liu, X. Xiao, and P. Stone, "Team orienteering coverage planning with uncertain reward," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 9728–9733.

[6] A. Shetty, T. Hussain, and G. Gao, "Decentralized connectivity maintenance for multi-robot systems under motion and sensing uncertainties," *NAVIGATION: Journal of the Institute of Navigation*, vol. 70, no. 1, 2023. [Online]. Available: https://navi.ion.org/content/70/1/navi.552

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

[8] E. Angelelli, C. Archetti, and M. Vindigni, "The Clustered Orienteering Problem," *European Journal of Operational Research*, vol. 238, no. 2, pp. 404–414, 2014.

[9] "The Set Orienteering Problem," *European Journal of Operational Research*, vol. 267, no. 1, pp. 264–272, 2018.

[10] "The probabilistic orienteering problem," *Computers & Operations Research*, vol. 81, pp. 269–281, 2017.

[11] M. G. Kantor and M. B. Rosenwein, "The Orienteering Problem with Time Windows," *The Journal of the Operational Research Society*, vol. 43, no. 6, pp. 629–635, 2025/02/20/ 1992.

[12] R. El-Hajj, D.-C. Dang, and A. Moukrim, "Solving the team orienteering problem with cutting planes," *Computers & Operations Research*, vol. 74, pp. 21–30, 2016.

[13] L. Ke, L. Zhai, J. Li, and F. T. Chan, "Pareto mimic algorithm: An approach to the team orienteering problem," *Omega*, vol. 61, pp. 155–166, 2016.

[14] W. Xu, Z. Xu, J. Peng, W. Liang, T. Liu, X. Jia, and S. K. Das, "Approximation algorithms for the team orienteering problem," in *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, 2020, pp. 1389–1398.

[15] A. Mansfield, S. Manjanna, D. G. Macharet, and M. Ani Hsieh, "Multi-robot scheduling for environmental monitoring as a team orienteering problem," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 6398–6404.

[16] J. Li, J. Zhu, G. Peng, J. Wang, L. Zhen, and E. Demeulemeester, "Branch-Price-and-Cut algorithms for the team orienteering problem with interval-varying profits," *European Journal of Operational Research*, vol. 319, no. 3, pp. 793–807, 2024.

[17] 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.

[18] A. Mansfield, D. G. Macharet, and M. A. Hsieh, "Energy-efficient team orienteering problem in the presence of time-varying ocean currents," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023, pp. 838–843.

[19] A.-E. Yahiaoui, A. Moukrim, and M. Serairi, "The clustered team orienteering problem," *Computers & Operations Research*, vol. 111, pp. 386–399, 2019.

[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] "Variable neighborhood search," *Computers & Operations Research*, vol. 24, no. 11, pp. 1097–1100, 1997.

[22] M. J. Geiger, "Randomised variable neighbourhood search for multi objective optimisation," *arXiv preprint arXiv:0809.0271*, 2008.

[23] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender, "Learning to rank using gradient descent," in *Proceedings of the 22nd International Conference on Machine Learning*, ser. ICML '05. New York, NY, USA: Association for Computing Machinery, 2005, p. 89–96. [Online]. Available: https://doi.org/10.1145/1102351.1102363

[24] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, p. 1735–1780, Nov. 1997. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735

[25] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2023. [Online]. Available: https://arxiv.org/abs/1706.03762

[26] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. Salakhutdinov, and A. Smola, "Deep sets," 2018. [Online]. Available: https://arxiv.org/abs/1703.06114

[27] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017. [Online]. Available: https://arxiv.org/abs/1412.6980