Attrition-Aware Adaptation for Multi-Agent Patrolling

Anthony Goeckner, Xinliang Li, Ermin Wei, and Qi Zhu, *Member, IEEE*

***Abstract*—Multi-agent patrolling is a key problem in a variety of domains such as intrusion detection, area surveillance, and policing, which involves repeated visits by a group of agents to specified points in an environment. While the problem is well-studied, most works do not provide performance guarantees and either do not consider agent attrition or impose significant communication requirements to enable adaptation. In this work, we present the Adaptive Heuristic-based Patrolling Algorithm, which is capable of adaptation to agent loss using minimal communication by taking advantage of Voronoi partitioning, and which meets guaranteed performance bounds. Additionally, we provide new centralized and distributed mathematical programming formulations of the patrolling problem, analyze the properties of Voronoi partitioning, and finally, show the value of our adaptive heuristic algorithm by comparison with various benchmark algorithms using physical robots and simulation based on the Robot Operating System (ROS) 2.**

***Index Terms*—Multi-robot systems, robotics in hazardous fields, path planning for multiple mobile robots or agents.**

I. Introduction

THE multi-agent patrolling or multi-robot patrolling problem is well studied, and for good reason. The problem appears frequently in cases such as area surveillance and monitoring, police beats, intrusion detection, and even has similarities to problems such as assignment of janitorial rounds. However, agent attrition is commonly seen in practice for many of these scenarios [1] and threatens to prevent completion of the patrolling task. This attrition can take many forms, such as vehicle breakdown, robot destruction, or even a human agent calling out on sick leave.

It is important that such an attrition event can be handled in an intuitive and efficient way. For example, consider the case of human-operated vehicle patrols with the potential for vehicle breakdown. When a vehicle breakdown occurs, the other human agents must divide that agent's tasks amongst themselves and continue patrolling. Rather than reallocating all tasks to all remaining agents (the mathematically optimal solution), one natural solution is to take human behavior and limitations into account by changing the allocations of only those agents that directly neighbor the disabled agent. This serves to reduce possible human confusion and misinterpretations by limiting the amount of change required.

Such patrol algorithms should be subject to theoretical bounds, especially a bound on the loss of performance after agent attrition. Further, while many works in the patrolling literature do address agent attrition, they often do so at high communication cost, requiring a large amount of coordination among agents [2]. In this work, we devise a method for agent patrolling that is capable of operating and adapting with almost no communication, and which provides performance guarantees. More specifically, we consider a multi-agent patrolling problem in which a team of agents must continually visit a set of observation points with a goal of minimizing the average time span that any observation point remains unvisited. To address this problem, agents must efficiently allocate observation points amongst themselves and determine an appropriate visitation order.

We first formulate this problem as an integer program and solve it repeatedly over time. The event of agent attrition may occur between problem solves. We then dynamically adjust our solution for the next solve. We note that the agent attrition event may be either stochastic, modeling unpredicted failures, or deterministic, reflecting scheduled breaks or down time with advance notice. This problem is NP-hard, and therefore we propose a Voronoi partition-based heuristic using two-stage decomposition, where the second stage can be solved in a distributed fashion. Based around this heuristic, we devise a distributed adaptive algorithm with theoretical performance bounds. This is the first work that we know of in the multi-agent patrolling literature to dynamically respond to agent attrition disturbances using minimal communication while limiting the amount of change required for the remaining agents and guaranteeing performance bounds. Specifically, this paper makes the following contributions:

- A centralized mathematical programming formulation (Section IV) and a distributed formulation (Section IV-A) for the multi-agent patrolling problem.
- A distributed heuristic algorithm for the multi-agent patrolling problem that can adapt to attrition of patrol agents using minimal communication while only requiring changes to a limited number of agents. (Section V)
- A theoretical analysis of difficulties encountered when using multiplicatively-weighted Voronoi partitioning based on heterogeneous vehicle speeds. (Section V-A2)
- Theoretical performance bounds for our heuristic algorithm. (Section V-B and V-C)
- An experimental comparison of our heuristic algorithm with existing approaches. (Section VI)

Manuscript received 11 January 2024; accepted 29 May 2024. Date of publication 2 July 2024; date of current version 8 July 2024. This letter was recommended for publication by Associate Editor D. Park and Editor H. Moon upon evaluation of the reviewers' comments. This work was supported by National Science Foundation under Grant 1834701, Grant 1724341, Grant 2038853, Grant 2324936, Grant 2030251, and Grant 2024774. *(Corresponding author: Anthony Goeckner.)*

The authors are with the Department of Electrical and Computer Engineering, Northwestern University, Evanston, IL 60208 USA (e-mail: anthony.goeckner@northwestern.edu; xinliangli2023@u.northwestern.edu; ermin.wei@northwestern.edu; qzhu@northwestern.edu).

Digital Object Identifier 10.1109/LRA.2024.3421793

II. Related Work

Multi-agent or multi-robot patrolling has a long history of study in the literature. Curious readers may find the latest general surveys on multi-robot patrolling [3], [4] to be of interest. However, we focus this section on papers especially related to the present work.

A lengthy field report by Taranta et al. from the DARPA OFFSET program highlights the importance of creating a multi-agent patrolling method that minimizes communication in tactical environments and is robust to agent attrition [1].

In [5], a minimum idleness connectivity-constrained multi-robot patrolling problem is introduced. Scherer et al. partition the graph into regions but do not consider the problem of agent attrition. The primary focus of the algorithm is to maintain communication amongst all agents. A similar-looking integer program formulation of the patrolling problem is presented, though it has key differences from ours which we will describe in Section IV.

Recent work by Bapat et al. [6] proposes two algorithms for multi-agent task allocation with highly restricted communications, one using a travelling salesman problem (TSP) heuristic. However, the algorithms are limited to one-time vehicle routing and not well-suited to the patrolling problem.

Voronoi partitioning has long been used in multi-agent task allocation, as seen in [7], [8], [9], [10]. These papers use Voronoi partitioning to divide tasks and are robust to agent failure. We expand on these with further analysis of the Voronoi partitioning with heterogeneous agent speeds and by application of Voronoi partitioning to the patrolling problem.

In a paper on the multi-depot vehicle routing problem (MDVRP), Bompadre et al. present an offline heuristic algorithm for task allocation which is the same as the initial steps of our algorithm: partition the space using the Voronoi method, and then determine the best tour through each partition using a traveling salesman problem heuristic [11]. Alongside [12], they also provide certain performance bounds of the algorithm which we expand on in this paper.

In work by Kim et al. [13], a task allocation system based on a Voronoi diagram for a multi-robot spray system in an orchard is proposed. This is expanded on in [14] by use of a multiplicatively weighted (MW) Voronoi-based task allocation scheme for agricultural robots with heterogeneous speed. We expand on this paper with a discussion of some of the issues of using MW Voronoi partitioning based on vehicle speed, which were not addressed by Kim et al.

Much of the current state-of-the-art patrolling research is based on algorithms, simulators, and environments originally created by [15], [16]. Recently, this simulator is extended by [17] to benchmark existing state-of-the-art (SoTA) algorithms against adversarial "attackers", though attrition of patrol agents or other disturbances are not considered. Further experiments at the University of Bristol [18] were performed using the same simulator to assess the SoTA algorithms' robustness to noisy anomaly detection at observation points.

Multi-agent reinforcement learning (MARL)-based approaches such as [19] are gaining popularity but to our knowledge do not as yet provide the same theoretical guarantees for performance or agent attrition as our method.

Recent work includes a method [20] for balancing agents' priorities amongst important and unimportant observation points which generates routes for agents while taking resource constraints into account. However, this method is not adaptive to disturbances such as agent attrition and unreliable communication. Another recent article by Kobayashi et al. presents a distributed patrol algorithm that ensures situation awareness of operators at a base station throughout the patrol [21], though it has the same drawbacks as [20].

Indeed, many recent publications either do not address disturbances such as agent attrition or do not provide performance guarantees [22], [23]. In contrast, our method enables adaptation to agent attrition, has minimal communication requirements, and provides theoretical performance bounds.

III. Problem Formulation

The goal of the multi-agent patrolling problem is to repeatedly visit a set of observation points such that some metric is minimized, often based on *idleness*, the amount of time each node spends between visits. Many existing works attempt to minimize the worst node idleness, the average idleness, or the standard deviation of the idleness. In our case, we attempt to minimize the *average idleness* of the nodes. We now introduce the problem setting and assumptions, and then provide a brief problem statement.

A. Problem Setting & Description of Symbols

The scalar m denotes the number of agents and n denotes the number of nodes. A patrol graph $G = (V, E)$ is a connected graph composed of a set of nodes $V = \{0, 1, \dots n\}$ and a set of edges $E = \{(i, j),\ i \in V\ j \in V\}$. From now on, we refer to "observation points" as "nodes". Agents are members of the set $A = \{0, 1, \dots m\}$. The binary matrix $y \in \mathbb{Z}_{m\times n}$ describes visits of agents to nodes: $y_i^a = 1$ if and only if agent a visits node i. The binary matrix $x \in \mathbb{Z}_{m\times n^2}$ describes whether an agent travels along an edge: $x_{ij}^a = 1$ if and only if agent a traverses the edge (i, j). The matrix $u \in \mathbb{R}_{m\times n}$ describes the time at which an agent visits a node. The matrix $c \in \mathbb{R}_{m\times n^2}$ describes travel time between all pairs of nodes for each agent.

The vector $o \in \mathbb{Z}_n$ describes the origin node for each agent: $o^a \in \{0, 1, \dots n\}$ is the node at which agent a resides at the allocation time.

We create a m-dimensional vector d with elements $d^a \in D$. These elements are artificial *destination* nodes. The vector $d \in D_k$ denotes the destination node for each agent. One destination node d^a is co-located with each origin node o^a such that the travel time between them is zero: $c^a_{o^a d^a} = c^a_{d^a o^a} = 0, \forall a \in A$.

For convenience, we define the augmented set of vertices (original and artificial) as $\mathring{V} = V \cup D$.

B. Assumptions

Our assumptions are consistent with realistic patrolling scenarios. We itemize them below.

- We assume that every agent a begins at some node $o_a \in V$. This is a realistic assumption, especially for large patrol areas. While we do not cover selection of origin nodes in this paper, the selection does have a significant impact on algorithm performance and may be addressed by future work.
- Agents may become disabled at any time and cease to move. We refer to this as "attrition". Agents may communicate that their vehicle has been disabled. Therefore, agent attrition is immediately known to all remaining agents. We

only consider attrition in terms of mobility, not communication ability.
- All origins are distinct. Given some origin $o^a \in o$,

$$o^a \neq o^{a'}, \quad \forall a' \neq a \in A.$$

- An agent may only travel along the edges E of the graph. If $\langle i, j\rangle \notin E \ \exists i, j \in V$, then an agent may not directly travel from node i to node j. An edge may be traversed by multiple agents simultaneously.
- Agents may be heterogeneous, each with a different speed. Agent speeds are known in advance to all agents.
- The cost c_{ij}^a is defined as travel time of agent a between nodes i and j:

$$c_{ij}^a = \frac{(i \divideontimes j)}{\text{speed}(a)}, \quad \forall a \in A, \quad \forall i, j \in V,$$

where $(\cdot \divideontimes \cdot)$ represents the length of the pairwise shortest path between the two nodes. Since the graph G and agent speed are known to all agents a priori, the cost function is also known a priori. The matrix c describes the all-pairs shortest path.

Having discussed the setting and assumptions above, we now present a problem statement.

Problem 1: Given a patrolling graph $G = (V, E)$, set of agents A, and related setting and assumptions as described above, determine assignment of agents to nodes using matrix y and visitation order of assigned nodes using matrix x in order to minimize the average time between visits (idleness time) of nodes in the patrol graph.

IV. Mathematical Programming Approach

We initially based our formulation on the deterministic single-agent Profitable Tour Problem (PTP) as described in [24], [25], which has an objective of minimizing the travel cost minus profits collected at each node. However, we realized that the problem could be simplified by assuming that each agent repeatedly visits the same nodes in a closed cycle and by attempting to minimize the average length of these cycles. Our formulation is most accurately categorized as a multi-depot vehicle routing problem (MDVRP), since we find a cycle beginning at a pre-selected origin point for each agent. This makes our formulation more similar to that discussed in [5], which seeks to minimize the maximum amount of time that any node is left idle (between visits). However, our objective is to minimize the *average* time that all nodes are left idle. We formulate this as a mixed-integer nonlinear program for readability, although there may be methods available to linearize the problem.

Based on the problem formulation above, we develop our multi-agent patrolling approach as follows. Note that we use a "big-M" formulation for constraint linearization, with some sufficiently large constant M that is significantly greater than any other variables.

$$\min_{x,y,u} z = \frac{1}{n} \sum_{a=0}^{m} \left(u_{d_a}^a \sum_{i=0}^{n} y_i^a \right) \tag{1}$$

$$\text{s.t.} \sum_{\substack{i=0 \\ i \neq j}}^{n} x_{ij}^a = y_j^a \qquad \forall a \in A, \forall j \in \mathring{V} \tag{1a}$$

$$\sum_{\substack{j=0 \\ j \neq i}}^{n} x_{ij}^a = y_i^a \qquad \forall a \in A, \forall i \in \mathring{V} \tag{1b}$$

$$u_i^a + c_{ij}^a - M(1 - x_{ij}^a) \leqslant u_j^a \qquad \forall a \in A, \forall i \in V, \ \forall j \in \mathring{V} \setminus \{i\} \tag{1c}$$

$$u_{o_a}^a = 0 \qquad \forall a \in A \tag{1d}$$

$$x_{d^a o^a}^a = 1 \qquad \forall a \in A \tag{1e}$$

$$\sum_{i \in D \setminus \{d^a\}} y_i^a = 0 \qquad \forall a \in A \tag{1f}$$

$$\sum_{a=0}^{m} y_i^a \geqslant 1 \qquad \forall i \in V$$

$$x_{ij}^a \in \{0, 1\} \qquad \forall a \in A, \forall i, j \in \mathring{V}$$

$$y_i^a \in \{0, 1\} \qquad \forall a \in A, \forall i \in \mathring{V}$$

$$u_i^a \geqslant 0 \qquad \forall a \in A, \forall i \in \mathring{V} \tag{1g}$$

This formulation results in each agent being assigned an ordered series of nodes to visit, beginning and ending at the same location (since o^a and d^a are co-located $\forall a$). This creates a closed patrol cycle for each agent.

a) Constraints: Constraint (1a) enforces that if agent a visits node j, a must have traversed some edge connected to j (a "come-from" constraint). Constraint (1b) enforces that agent a may only traverse a single edge immediately succeeding node i (a "go-to" constraint). Constraint (1c) enforces that there are no subtours and records the visit time of agent a to node j as the visit time at node i plus the travel cost (time) between nodes i and j. This is similar to the "subtour elimination constraint" from the Miller-Tucker-Zemlin TSP [26]. Constraint (1d) enforces that the origin node is visited at time 0. Constraint (1e) enforces that an agent a must travel from its artificial destination node d^a to its co-located origin node o^a, creating a cycle. Constraint (1f) enforces that an agent not visit the artificial destination nodes of other agents. Constraint (1g) enforces that all nodes must be visited.

b) Objective Function: The objective function (1) describes the *average idleness time* of all nodes. For each agent $a \in A$, we multiply the total time $u_{d^a}^a$ taken to complete a cycle by the number of nodes $\sum_{i \in V} y_i^a$ visited in that cycle. We define *idleness* as the time span between agent visits. Each of these nodes will then have the same idleness, since only one agent is assigned to each cycle. Summing the results for each agent a provides the total idleness time of all nodes, which we divide by n to find the average idleness time. Node idleness time is a commonly used metric in the literature for comparison of patrolling algorithms, as seen in e.g. [15], [2], [25]. By using this objective function, we can more easily compare our solution with existing approaches.

Note an important difference here between our formulation and that of [5]'s min-max vertex cycle cover problem. While Scherer et al. attempt to minimize the worst idleness time of any node, we attempt to minimize the average idleness time. Further, by use of the assignment decision matrix y, our problem may be more easily decomposed to a distributed problem (see Section IV-A).

A. Distributed Approach

In multi-agent systems where attrition is expected to occur or in systems that must be robust to disturbances, such as those studied in [1], a single point of failure is often unacceptable. Therefore, we also devise a distributed approach to the problem which we will later use as the basis for our adaptive heuristic algorithm in Section V.

To create a two-stage distributed version of our problem, we break it into master and sub-problems. The master problem should allocate nodes to agents, and the sub-problem should determine the visitation order of the allocated nodes. Once the master problem generates an assignment, the sub-problems can be solved using local information in a distributed and parallel fashion. We observe that y (node allocation) is a confounding variable, which when fixed the problem naturally decomposes into m sub-problems, one for each agent. As described by [27], each of these sub-problems is a traveling salesman problem (TSP).

We define a sub-problem for each agent a as follows, where $\hat{y}^a$ is the fixed y^a chosen by the master problem:

$$\min_{x^a,u^a} z^a = \frac{u^a_{d_a}}{n} \sum_{i=0}^{n} \hat{y}^a_i \tag{2}$$

$$\text{s.t.} \sum_{\substack{i=0\\ i\neq j}}^{n} x^a_{ij} = \hat{y}^a_j \qquad \forall j \in \mathring{V} \tag{2a}$$

$$\sum_{\substack{j=0\\ j\neq i}}^{n} x^a_{ij} = \hat{y}^a_i \qquad \forall i \in \mathring{V} \tag{2b}$$

$$u^a_i + c^a_{ij} - M(1 - x^a_{ij}) \leqslant u^a_j \qquad \forall i \in V,\ \forall j \in \mathring{V} \setminus \{i\} \tag{2c}$$

$$u^a_{o_a} = 0 \tag{2d}$$

$$x^a_{d^a o^a} = 1$$

$$x^a_{ij} \in \{0,1\} \qquad \forall i,j \in \mathring{V}$$

$$u^a_i \geqslant 0 \qquad \forall i \in \mathring{V} \tag{2e}$$

Given a predefined set of nodes to visit $\{i|\hat{y}^a_i = 1\}$ by the master problem, this sub-problem finds the optimal tour for agent a to visit all nodes in the set. This is clearly a TSP.

B. Agent Attrition

When an agent a suffers attrition during execution, nodes assigned to that agent must be reassigned to other agents. In terms of the original MIP described in Section IV, this is equivalent to adding constraints $y^a_i = 0 \quad \forall i \in \mathring{V}$ and modifying (1e) to have a right-hand side of 0.

Performing this constraint modification is impractical for real-time adaptation, requiring a re-solve of the MDVRP, which is NP-Hard. Therefore, we introduce a heuristic approach to the problem to allow for real-time adaptation.

Algorithm 1: The Adaptive Heuristic-Based Patrolling Alg.

```
y ← voronoiPartition(G)
x^a, u^a ← TSP(y^a)
Begin patrolling.
while patrolling do
    if another agent ã suffers attrition then
        y' ← voronoiPartition(G)
        if y'^a ≠ y^a then        ▷ Did our alloc. change?
            x^a, u^a ← TSP(y'^a)  ▷ Yes, do TSP again.
        end if
        y ← y'
    end if
end while
```

V. Heuristic Approach

The nonlinear optimization problem described in Section III is NP-Hard and thus is not suitable for computation at runtime or with large numbers of agents or nodes.

As seen in Section IV-A, the problem may be divided into two phases: node assignment phase and visitation ordering phase. While these phases are somewhat interdependent, we attempt to create a heuristic for each of the two phases. Then we describe an adaptive algorithm to solve the problem of patrolling in the face of agent attrition.

a) Node Assignment Heuristic: As a heuristic for the node assignment phase, we use Voronoi partitioning of nodes based on the agent's origin point and using travel time as the distance measure. For each node, we calculate its shortest wait time to each of the origins and assign it to the best one. Formally, for each node i, we first define an optimal agent selection function $a^*(i)$ by solving the problem of

$$a^*(i) \in \arg\min_{a\in A} c^a_{io^a} = (i \circledast o^a)/\text{speed}(a) \tag{3}$$

If there are multiple minimizers for this problem, we randomly pick one to assign to $a^*(i)$. Then we assign the decision variable

$$\hat{y}^a_i = \begin{cases} 1 & \text{if } a = a^*(i), \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

We select this simple geometric heuristic for good reason: it allows for the loss of an agent and subsequent reallocation of that agent's nodes without reallocation of the entire graph. See Section V-A1 for more information and proof.

b) Visitation Order Heuristic: The visitation ordering phase is a TSP [27]. We use a simple nearest-neighbor heuristic to find an approximate solution. This is a greedy algorithm which, starting at the origin, selects the nearest node that is yet to be visited, then performs the same operation at the selected node, and so on until all nodes have been visited.

A. The Adaptive Heuristic Algorithm

We create a runtime heuristic algorithm that enables patrolling while dynamically responding to agent attrition (due to vehicle breakdown, etc.) by reallocating select agents to cover for the lost agent. The attrition and adaptation process is covered in detail below in Section V-A1.

Before patrolling can begin, all nodes in the graph must be allocated to agents using Voronoi partitioning based on the agent's starting position.

We refer to this algorithm as the Adaptive Heuristic-based Patrolling Algorithm (AHPA), and it is labelled as "AHPA" in any figures or graphs.

AHPA runs concurrently on all agents. The only inter-agent communication required by our algorithm is a notification of agent attrition. While we currently model this as an explicitly communicated message, it could just as easily be an observation or other non-networked indication. The potential use of observations is aided by the fact that only neighboring agents must change their allocation, as will be shown in Theorem 1.

1) Agent Attrition & Adaptation: When agent attrition occurs due to vehicle breakdown, the agent's observation points must instead be visited by other agents. As seen in Algorithm 1, we perform Voronoi partitioning of the entire graph G, resulting in a new assignment matrix y'. We feed this new assignment matrix into our visitation order heuristic to calculate routes for the remaining agents.

One of our design goals is to minimize disruption to remaining agents upon agent loss. Therefore, we use Voronoi partitioning for the assignment heuristic and consider the case where all vehicle speeds are the same. In case of agent loss, only the assignments of the agent's immediate Voronoi neighbors (see Definition 1) will change.

Definition 1 (Neighboring Agent): The agent $b \in A$ is considered a neighbor of agent $a \in A$ if for all nodes i with $a^*(i) = a$ and j with $a^*(j) = b$ and for all $t \in \text{path}(i, j)$, $a^*(t) = a$ or $a^*(t) = b$, where $\text{path}(i, j)$ refers to any shortest path between i and j.

Theorem 1: The loss of a single agent $\tilde{a}$ will only change the allocations of its neighboring agents for the heuristics described in (4) when all agents move at the same speed.

Proof: For any nodes i with $a^*(i) \neq \tilde{a}$, the original optimal agent assignment remains optimal and therefore we only change the allocation for the nodes with $a^*(i) = \tilde{a}$. We proceed to prove by contradiction. Suppose for such node i,

$$a^*(i) \in \underset{a \in A - \{\tilde{a}\}}{\arg\min}\, c^a_{io^a} = c, \tag{5}$$

where c is not a neighboring agent of $\tilde{a}$. This implies that the path between i and c goes through a node j, with $a^*(j) = b$ and b is a neighboring agent of $\tilde{a}$. For this node j, we also have that

$$(j \circledast b) < (j \circledast c), \tag{6}$$

and therefore it is in the interior of the partition associated with agent b. If such j does not exist, then agents a and c would be neighbors by Definition 1.

By (5), we have

$$c^c_{ic} \leq c^b_{ib}.$$

Furthermore, since all agents' speeds are the same, we have

$$(i \circledast j) + (j \circledast c) = (i \circledast c) \leq (i \circledast b).$$

Since the definition of $(i \circledast b)$ calculates the shortest path distance between i and b, we have

$$(i \circledast b) \leq (i \circledast j) + (j \circledast b).$$

The previous two relations imply that

$$(j \circledast c) \leq (j \circledast b),$$

which contradicts (6) and completes the proof. □

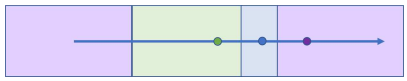

Fig. 1. Discontinuous weighted Voronoi partition based on heterogeneous agent speed, illustrating the dilemma described in Section V-A2. Dots represent agents and colored boxes represent partitions assigned to the agent of the same color. The right-most agent (purple) has significantly higher speed than the other two, resulting in unexpected partitioning results.

2) The Case of Heterogeneous Speed: We note that Theorem 1 does not hold when the agents have different speeds. In fact, multiplicatively-weighted Voronoi partitioning based on different agent speeds results in unexpected nonconvex/non-continuous allocation results. Consider the following simple case, where many nodes are allocated to agents on the real line with three agents placed at locations 0,1,2, respectively, from left to right. We refer to these agents by their locations. The agent at 2 has a speed of 2 units/s and agents at $0, 1$ both move at a speed of 1 unit/s. Therefore, any node i which lies in $(-\infty, -2]$ has $a^*(i) = 2$, any node i in $(-2, 1/2]$ has $a^*(i) = 0$, any node i in $(1/2, 4/3]$ has $a^*(i) = 1$, and any node i in $(4/3, \infty)$ has $a^*(i) = 2$. Note that due to the different speeds, even before any agent attrition, the partitions are not contiguous, as shown in Fig. 1. Namely, agent 2 covers the area nearby and also area far enough to the left of agent 0, since its speed is the highest. If agent 0 breaks down, then the point -1 would take time 2 s to reach for the agent at 1 and $3/2$ s for the agent at 2, and therefore will be assigned to the agent at location 2, who is not a neighbor of agent 0, thus invalidating Theorem 1. This aspect of multiplicatively-weighted Voronoi partitioning is not addressed by earlier works such as [14]. While our Theorem 1 is unable to handle this case, we wish to present it for further discussion.

B. AHPA Performance Bound

At its core, AHPA uses Voronoi partitioning to divide the environment and then determines the best tour of each partition using a TSP heuristic. Bompadre et al. derive an $\Omega(m)$ lower bound for the Voronoi partitioning with TSP method in the context of the multi-depot vehicle routing problem, which describes it as an m-approximation process, where m is the number of depots [11]. In our case, we view depots as agent origins (depots serving a single agent), and AHPA can be reduced to the same algorithm as [11] by considering only the initial operations at lines 1-3 of Algorithm 1. Therefore, AHPA is an m-approximation of the optimization problem presented in Section IV.

C. Bound on Performance Loss After Attrition

One of AHPA's main goals is robustness to agent attrition with little communication overhead. In this section, we provide a bound on the algorithm's maximum performance loss after agent attrition occurs.

Thanks to prior works, we know that the approximation ratio of AHPA to the optimal solution is $\Omega(m)$ [11] and $O(m)$ [12]. Therefore, the tightest approximation ratio of AHPA is $\Theta(m)$. The bound on performance loss after agent attrition is the ratio of the original approximation ratio for m agents and the new approximation ratio for $m - 1$ agents after attrition:

$$\frac{\Theta(m)}{\Theta(m-1)}$$

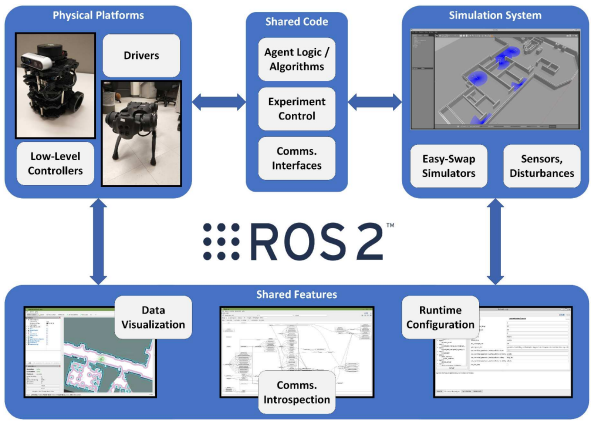

Fig. 2. We use the MAS framework, Grex, for straightforward experimentation in both simulated environments and with physical robots. At top-left are the TurtleBot3 robots, which we use in physical experiments. At top-right, simulated robots patrol the "Cumberland" environment from [15].

Hence, attrition of one agent roughly translates to $\Theta(\frac{m}{m-1})$ increase in average node idleness time.

VI. Experimentation & Results

We experimentally compare our approach to several existing state-of-the-art dynamic patrolling methods using both physical robots and a realistic simulation environment. Both physical and simulated experiments use the *Grex* framework, which we developed for general MAS research.

A. Experimental Environment

Much prior work on patrolling algorithms, including that by [15], [16], [2], [28], [29], uses a patrolling simulator originally developed by David Portugal et al. in [15] and further papers. Many of the patrolling algorithms developed using this simulator are still considered to be the state-of-the-art.

Separately, we have developed a multi-agent simulation and experimentation framework based on ROS 2, the Robot Operating System, which may be seen in Fig. 2. Our framework, Grex, is capable of execution on real robots or may be used with a variety of compatible simulators. Using this multi-agent framework, we are able to write agent and experiment code which is common both to physical platforms and to a variety of simulators. ROS 2 provides us with features that are shared across these physical and simulated platforms including data visualization, communications introspection, runtime configuration, and a wide-ranging library of third-party modules for easy integration of new capabilities.

For the purposes of this paper, we leverage that extensibility to integrate the environments and patrolling algorithms created by Portugal et al. [15] and others with our framework[1]. This greatly simplifies comparison with existing algorithms.

We simulate sensor noise based on Gaussian distributions; this is the primary source of uncertainty in the simulation. Each agent performs its own localization and navigation. Agents may collide or interference with each other, resulting in delay

[1]The integrated experiment code is publicly available at https://github.com/NU-IDEAS-Lab/patrolling_sim

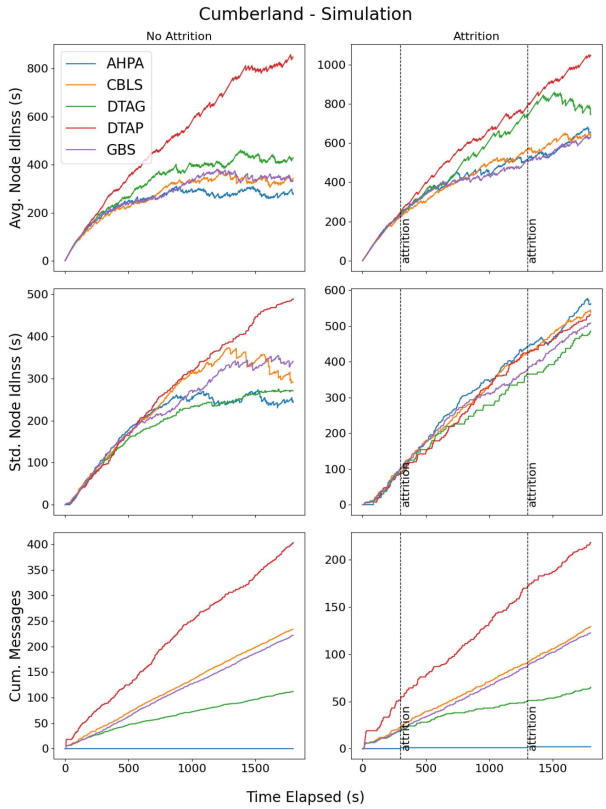

Fig. 3. Performance of the algorithms over time in the simulated Cumberland environment originally used in [15]. At left, results with no agent attrition. Note the excellent comparative performance of AHPA (in blue). At right, with two instances of agent attrition. Note the cumulative message difference between AHPA and the other algorithms.

reaching goals. For physical robots, real disturbances are at play, including sensor noise, localization uncertainty, communication losses, and robot collisions.

B. Benchmark Algorithms

We select four benchmark algorithms to compare against our own solution. Two are greedy algorithms that serve as a baseline: Greedy Bayesian Strategy (GBS) [15] and DTA-Greedy [2]. We also compare against the more sophisticated Concurrent Bayesian Learning Strategy (CBLS) [16] and DTA-Partitioning (DTAP) [2].

C. Experimental Procedure

For each of the algorithms described in Section VI-B plus our own Algorithm 1 (AHPA), we perform tests using six agents in both the 40-node "Cumberland" environment from [15] and our own 18-node "L440" environment, an empty conference room. A "Cumberland" test lasts thirty minutes (1800 seconds). Each algorithm is tested twice: once with no agent attrition and once with attrition of single randomly-selected agents at the 300-second and 1300-second marks. A test in the "L440" environment lasts for five minutes.

For each algorithm and each test, we execute three runs over which results are averaged to account for possible discrepancies. The test system is automated, and experimental monitoring

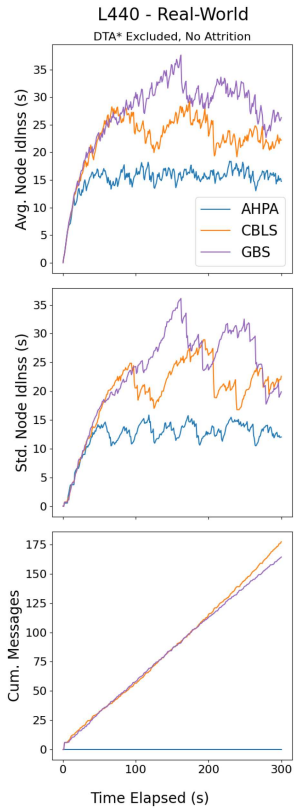

Fig. 4. Performance of AHPA and selected benchmark algorithms in real experiments. Note the stability of AHPA throughout. The DTAP and DTAG algorithms are not shown, as their performance was so poor that the graph became difficult to read.

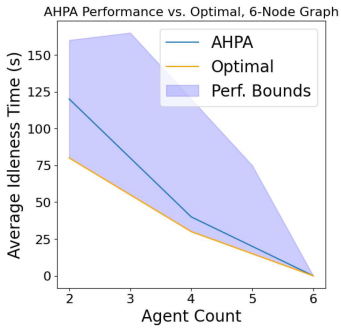

Fig. 5. Performance of AHPA compared to the optimal performance. Note that AHPA remains within the performance bound described in Section V-B.

and control is performed identically for each algorithm. All simulation tests were performed on the same machine.

D. Results

In analysis of the results, our primary focus is on the "average idleness" metric which we identify as our objective function in (1). We look at this value over time to evaluate the performance of our algorithm.

Overall, our AHPA algorithm performs well in comparison with existing approaches and effectively addresses the problem statement in Section III. As seen in the left-hand column of Fig. 3, the AHPA algorithm outperforms all others in non-attrition trials. We attribute this to AHPA's partitioning of the environment and surmise via observation of experiments that many of the other algorithms suffer from either poor/overlapping partitioning (DTAP) or from physical collisions amongst the agents (DTAG, CBLS), which significantly impacts performance. AHPA's solution results in non-overlapping partitions which improves allocations while also decreasing physical interference and collisions amongst the agents. As expected, use of AHPA results in a lower standard deviation of node idleness than other approaches, since each agent is assigned a tour which they then patrol continually. This may be seen in the left-hand column of Fig. 3.

In the attrition test, we remove a randomly-selected agent from the simulation at 300 seconds and another agent at 1300 seconds. The results of this are visible in the right-hand column of Fig. 3. While AHPA does not recover faster than other algorithms, it requires far fewer inter-agent messages than the other algorithms to do so. As seen in Fig. 3, it only requires a single message in case of agent attrition. Contrast this with the many messages used by other algorithms, which can be problematic in communication-constrained environments.

Physical experiments demonstrate results consistent with those from simulation. AHPA exhibits stable operation throughout the test and outperforms the benchmark algorithms. However, we are surprised by the poor performance of the DTAP and DTAG algorithms. We cannot reproduce the high performance of those algorithms seen in [2]. In fact, DTAP and DTAG perform so poorly in our physical tests, with average idleness times nearing 252 and 216 seconds, respectively, that we exclude them from Fig. 4 for readability. We observe that their poor performance is due to physical collisions and interference amongst agents during mission execution. However, other algorithms do not suffer from this problem to the same extent.

Performance Bounds: To demonstrate the AHPA performance bound described in Section V-B, we also compare AHPA with the optimal solution (computed using a commercial solver) for a small problem with $n = 6$ nodes and $m = 2 \ldots 6$ agents. The results may be seen in Fig. 5, with useful performance bounds provided.

VII. Conclusion and Future Work

In this work, we analyze the multi-agent patrolling problem and present an adaptive heuristic algorithm (AHPA), using Voronoi partitioning to allocate nodes to agents and a TSP heuristic to determine visitation order for each agent. We provide performance guarantees for this algorithm, including in the case of agent attrition. This is the first patrolling algorithm that we know of to provide such guarantees. Further, we devise new centralized and decentralized integer programming formulations that are better-suited than existing MDVRP formulations to many patrolling scenarios. We show that in the case of heterogeneous agent speeds, the behavior of Voronoi partitioning based on speed yields unexpected results. This has not been reported before to our knowledge, even though existing works use speed-weighted Voronoi partitioning for allocation.

We also show that in the case of homogeneous agent speeds, the attrition of a single agent will only affect its neighboring agents' allocations, which is beneficial in certain patrolling scenarios, especially when humans are involved.

By only changing the allocations of neighboring agents after attrition, we hope that future research will be able to devise a communication-free patrolling methodology relying only on local agent observations of neighbors to perform adaptation in the face of attrition. Future work may investigate the use of algorithm- and communication-layer co-design for cyber-physical systems in practical environments [30], [31] to meet this need.

Together, our contributions form a highly effective multi-agent patrolling algorithm, AHPA, which is capable of adapting to agent attrition with minimal communication requirements and is subject to guaranteed performance bounds. As seen in [1], this work is highly valuable in practice and we hope that it will lead to additional research focus on these practical considerations of attrition and disturbances in MAS.

ACKNOWLEDGMENT

The authors would like to thank Dr. Yixuan Wang, Mr. Henry Abrahamson, Dr. Xiangguo Liu, and Dr. Karen Smilowitz for their feedback and advice during the development of this paper.

REFERENCES

[1] E. Taranta, A. Seiwert, A. Goeckner, K. Nguyen, and E. Cherry, "From warfighting needs to robot actuation: A complete rapid integration swarming solution," *Field Robot.*, vol. 3, no. 1, pp. 460–515, Jan. 2023.

[2] A. Farinelli, L. Iocchi, and D. Nardi, "Distributed on-line dynamic task assignment for multi-robot patrolling," *Auton. Robots*, vol. 41, no. 6, pp. 1321–1345, Aug. 2017.

[3] N. Basilico, "Recent trends in robotic patrolling," *Curr. Robot. Rep.*, vol. 3, no. 2, pp. 65–76, Jun. 2022.

[4] L. Huang, M. Zhou, K. Hao, and E. Hou, "A survey of multi-robot regular and adversarial patrolling," *IEEE/CAA J. Automatica Sinica*, vol. 6, no. 4, pp. 894–903, Jul. 2019.

[5] J. Scherer, A. P. Schoellig, and B. Rinner, "Min-max vertex cycle covers with connectivity constraints for multi-robot patrolling," *IEEE Robot. Automat. Lett.*, vol. 7, no. 4, pp. 10152–10159, Oct. 2022.

[6] A. Bapat, B. R. Bora, J. W. Herrmann, S. Azarm, H. Xu, and M. W. Otte, "Distributed task allocation algorithms for multi-agent systems with very low communication," *IEEE Access*, vol. 10, pp. 124083–124102, 2022.

[7] J. Cortes, S. Martinez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," *IEEE Trans. Robot. Automat.*, vol. 20, no. 2, pp. 243–255, Apr. 2004.

[8] J. G. M. Fu, T. Bandyopadhyay, and M. H. Ang, "Local voronoi decomposition for multi-agent task allocation," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2009, pp. 1935–1940.

[9] P. Dasgupta, A. Muñoz-Meléndez, and K. R. Guruprasad, "Multi-robot terrain coverage and task allocation for autonomous detection of landmines," *Proc. SPIE*, vol. 8359, pp. 98–111, 2012.

[10] C. Nowzari and J. Cortés, "Self-triggered coordination of robotic networks for optimal deployment," *Automatica*, vol. 48, no. 6, pp. 1077–1087, Jun. 2012.

[11] A. Bompadre, M. Dror, and J. B. Orlin, "Probabilistic analysis of unit-demand vehicle routeing problems," *J. Appl. Probability*, vol. 44, no. 1, pp. 259–278, 2007.

[12] D. Hwang, P. Jaillet, and Z. Zhou, "Distributed multi-depot routing without communications," 2014, *arXiv:1412.2123*.

[13] J. Kim and H. I. Son, "A voronoi diagram-based workspace partition for weak cooperation of multi-robot system in orchard," *IEEE Access*, vol. 8, pp. 20676–20686, 2020.

[14] J. Kim, C. Ju, and H. I. Son, "A multiplicatively weighted voronoi-based workspace partition for heterogeneous seeding robots," *Electronics*, vol. 9, no. 11, Nov. 2020, Art. no. 1813.

[15] D. Portugal and R. P. Rocha, "Distributed multi-robot patrol: A scalable and fault-tolerant framework," *Robot. Auton. Syst.*, vol. 61, no. 12, pp. 1572–1587, Dec. 2013.

[16] D. Portugal and R. P. Rocha, "Cooperative multi-robot patrol with Bayesian learning," *Auton Robot*, vol. 40, no. 5, pp. 929–953, Jun. 2016.

[17] J. C. Ward and E. R. Hunt, "An empirical method for benchmarking multi-robot patrol strategies in adversarial environments," in *Proc. 38th ACM/SIGAPP Symp. Appl. Comput.*, 2023, pp. 787–790.

[18] Z. R. Madin, J. Lawry, and E. R. Hunt, "Collective anomaly perception during multi-robot patrol: Constrained interactions can promote accurate consensus," in *Proc. 39th ACM/SIGAPP Symp. Appl. Comput.*, Dec. 2024, pp. 630–637.

[19] L. Guo, H. Pan, X. Duan, and J. He, "Balancing efficiency and unpredictability in multi-robot patrolling: A MARL-based approach," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2023, pp. 3504–3509.

[20] R. Katole, D. Mallya, L. Vachhani, and A. Sinha, "Balancing priorities in patrolling with Rabbit walks," Dec. 2023, *arXiv:2312.16564*.

[21] K. Kobayashi, S. Ueno, and T. Higuchi, "Multi-robot patrol algorithm with distributed coordination and consciousness of the base station's situation awareness," Oct. 2023, *arXiv:2307.08966*.

[22] N. De Bona, L. Santoro, D. Brunelli, and D. Fontanelli, "Adaptive expected reactive algorithm for heterogeneous patrolling systems based on target uncertainty," in *Proc. IEEE 47th Annu. Comput., Softw., Appl. Conf.*, 2023, pp. 51–56.

[23] L. Huang, M. Zhou, K. Hao, and H. Han, "Multirobot cooperative patrolling strategy for moving objects," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 53, no. 5, pp. 2995–3007, May 2023.

[24] C. Archetti, M. G. Speranza, and D. Vigo, "Chapter 10: Vehicle routing problems with profits," in *Vehicle Routing: Problems, Methods, and Applications* (MOS-SIAM Series on Optimization), 2nd ed. Philadelphia, PA, USA: SIAM, 2014, pp. 273–297.

[25] M. Bruni, L. Brusco, G. Ielpa, and P. Beraldi, "The risk-averse profitable tour problem:," in *Proc. 8th Int. Conf. Operations Res. Enterprise Syst.* 2019, pp. 459–466.

[26] C. E. Miller, A. W. Tucker, and R. A. Zemlin, "Integer programming formulation of traveling salesman problems," *J. ACM*, vol. 7, no. 4, pp. 326–329, Oct. 1960.

[27] A. Mor and M. G. Speranza, "Vehicle routing problems over time: A survey," *Ann Oper Res*, vol. 314, no. 1, pp. 255–275, Jul. 2022.

[28] B. Wiandt, V. Simon, and A. Kokuti, "Self-organized graph partitioning approach for multi-agent patrolling in generic graphs," in *Proc. IEEE 17th Int. Conf. Smart Technol.*, 2017, pp. 605–610.

[29] H. ElGibreen and K. Youcef-Toumi, "Dynamic task allocation in an uncertain environment with heterogeneous multi-agents," *Auton Robot*, vol. 43, no. 7, pp. 1639–1664, Oct. 2019.

[30] Q. Zhu et al., "Know the unknowns: Addressing disturbances and uncertainties in autonomous systems," in *Proc. 39th Int. Conf. Comput.-Aided Des.*, 2020, pp. 1–9. [Online]. Available: https://doi.org/10.1145/3400302.3415768

[31] Q. Zhu and A. Sangiovanni-Vincentelli, "Codesign methodologies and tools for cyber–physical systems," *Proc. IEEE*, vol. 106, no. 9, pp. 1484–1500, Sep. 2018.