Multi-Agent Multi-Objective Ergodic Search using Branch and Bound

Akshaya Kesarimangalam Srinivasan*, Geordan Gutow*, Zhongqiang Ren*, Ian Abraham†,
Bhaskar Vundurthy* and Howie Choset*

***Abstract*— Search and rescue applications often need multiple agents to complete a set of conflicting tasks. This paper studies a Multi-Agent Multi-Objective Ergodic Search (MA-MO-ES) approach to this problem where each objective or task is to cover a domain subject to an information map. The goal is to allocate coverage tasks to agents so that all maps are explored ergodically. The combinatorial nature of task allocation makes it computationally expensive to solve for optimal allocation using brute force. Apart from a large number of possible allocations, computing the cost of a task allocation is itself an expensive planning problem. To mitigate the computational challenge, we present a branch and bound-based algorithm with pruning techniques that reduce the number of allocations to be searched to find optimal coverage task allocation. We also present an approach to leverage the similarity between information maps to further reduce computation. Extensive testing on 147 randomly generated test cases shows an order of magnitude improvement in runtime compared to an exhaustive brute force approach.**

I. Introduction

Applications such as search and rescue [1], surveillance [2], and planetary exploration [3] all require planning for multiple robots to collectively search a domain to gain information about it. In complex scenarios, there may be multiple competing forms of information to collect, which have different scales across the domain; which we refer to as information maps. In such a scenario, it is natural to cast the problem as a multi-objective optimization where each objective is to cover the domain subject to a single map. In a multi-agent setting, the problem further demands the appropriate allocation of agents to information maps to ensure effective coverage of all maps. We approach this problem as a Multi-Agent Multi-Objective Ergodic Search (MA-MO-ES) problem as shown in Fig. 1.

Prior work [4] found a set of Pareto-optimal solutions for a single agent with multiple objectives in a shared workspace, that leverages the ergodic metric. The ergodic metric measures the quality of coverage and is minimized when the time spent observing a region is proportional to the information in that region. In this paper, we are interested in reconciling the multiple objectives by minimizing the maximum of the objectives, i.e., the worst ergodicity on an information map. Thus, the task allocation aims to allocate each agent to one or more information maps such that the maximum or worst ergodicity on any map is minimized, i.e., minmax optimization.

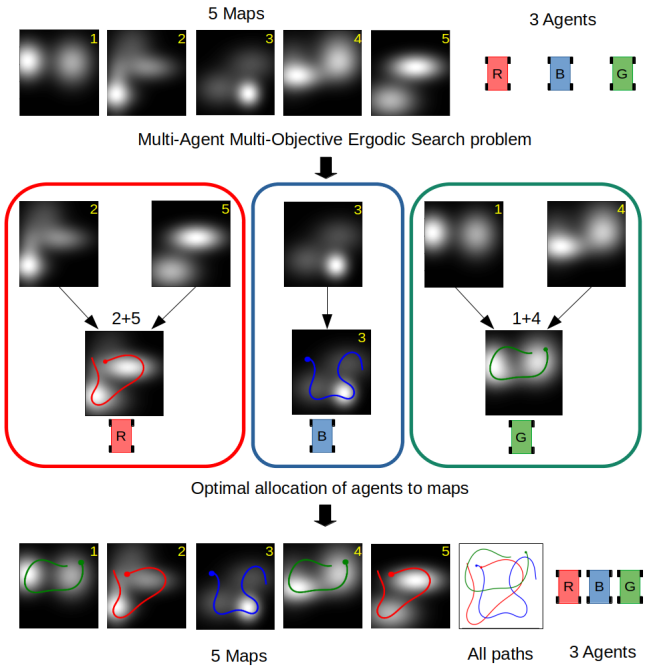


Fig. 1: This figure illustrates the Multi-agent Multi-objective Ergodic Search (MA-MO-ES) problem using five information maps ($1-5$) that span the same physical region, and three agents (Red, Blue, and Green). The optimal allocation is shown with each agent in a different box. An agent's trajectory is optimized on a weighted average of its maps and evaluated by reconstructing the trajectory on each map.

According to the taxonomy defined in [5], our problem is a multi-task single-robot time-extended variation of the Multi-Robot Task Allocation (MRTA) problem that is NP-Hard [6]. Prior works have addressed multi-agent multi-objective task allocation problems using auctioning systems [7], greedy allocation [8], evolutionary methods [9] etc. While the approaches in [8],[7] are fast, they do not guarantee finding the optimal allocation scheme in the minmax sense.

In this work, we leverage the minmax formulation to eliminate non-optimal solutions without evaluating the performance on all the maps for each possible allocation. If the worst ergodicity on a map for a partial allocation, i.e., assignment of maps to a subset of agents exceeds that of the current best allocation, then the worst ergodicity for the

*Akshaya Kesarimangalam Srinivasan, Geordan Gutow, Zhongqiang Ren, Bhaskar Vundurthy, and Howie Choset are with Carnegie Mellon University, 5000 Forbes Ave., Pittsburgh, PA 15213, USA. (email: {akesarim, ggutow, zhongqir, pvundurt, choset}@andrew.cmu.edu).

†Ian Abraham is with Yale University, 17 Hillhouse Avenue, New Haven, CT 06511, USA. (email: ian.abraham@yale.edu)

complete allocation would also exceed the current best. Using this idea, we present a branch and bound formulation for allocating maps to agents. Compared to exhaustive search, the branch and bound approach reduces runtime by an order of magnitude while guaranteeing minmax optimal allocation.

We also propose an approach to leverage similarity between information maps to reduce further the number of possible allocations to be considered. The runtime and minmax metric of these approaches were compared against baseline approaches on 147 randomly generated test cases.

The remainder of the paper is organized as follows: Section III-A introduces basic concepts. The problem formulation is presented in Section III-B. Section IV-A explains a branch and bound approach to the problem. Section IV-B proposes a similarity-based clustering algorithm with branch and bound. Section V provides numerical results demonstrating improvement over the baseline approach, and the task allocation approach in [8]. Finally, Section VI presents some conclusions and possible directions for future work.

II. Related Works

A. Ergodic Coverage

Given an information map, a trajectory is ergodic when the Fourier basis functions along the trajectory converge to the spatial averages of the basis functions [10], i.e., when the time spent in an area is proportional to the information present in that area. This trajectory can be achieved by minimizing the ergodic metric [10]. Ergodic trajectory planning has been studied in various contexts such as receding horizon control [11], stochastic optimization [12], active learning and search [13][14], decentralized exploration [15], and real-time area coverage and target localization [16]. Our prior work [4] addresses the single-agent multi-objective ergodic search problem. It illustrates a local sequential optimization approach to compute a Pareto optimal front of solutions quickly and accurately [17]. However, we are unaware of any ergodic search method that considers covering a domain subject to multiple information maps with multiple agents, which is the focus of this work.

B. Multi-agent Multi-objective Task Allocation and Path Planning

The core contribution of this work is task allocation for multiple agents. Multi-agent task allocation is widely studied in literature [18]. The authors of [19] extend the frontier-based algorithm to multi-objective problems, but with a greedy task allocation approach. Work in [20] addresses tasks with unknown allocation costs like in ours but with a single objective. While [1] optimizes multi-agent path planning to detect multiple targets modeled using prior cell occupancy probability distribution using a MILP approach, the task allocation is not explicit, making it difficult to cover multiple conflicting distributions.

Two prior works related to the approach of the current work are [7] and [8]. The authors of [8] propose two frameworks for task allocation, the Compromise View model and the Nearest Neighbour Search model. The former uses the agent's distance from the target as the heuristic for greedy task assignments whereas the latter clusters target locations using k-means clustering for faster but higher path-length solutions. The work in [7] presents a framework for multiple agents to perform exploration, rendezvous, and tasks in a cell-decomposed environment. The cells to be visited are clustered, one for each agent, and assigned via centralized auction based on agents' distance to the cluster centroids and a multi-objective optimization method based on weighted prioritization of exploration and task completion. Both works demonstrate the effectiveness of clustering for task allocation in multi-agent scenarios. Our work differs from [7] and [8] in two ways. Computing the cost of allocation is itself a trajectory optimization problem and hence expensive to obtain accurately. The tasks are not waypoints that need to be visited but are coverage tasks subject to information maps. Clustering has also been used for task assignment of parallel programs to processors [21]. It aims to allocate tasks to processors for load balancing while reducing the cost of communication overhead and processor turn-around time. First, clusters are identified in a task graph, with tasks as nodes and communication between tasks as edge weights, such that intra-cluster communication is high and inter-cluster communications are minimum. This clustering with an estimate of the cost of a complete allocation is used as pruning rules in a branch and bound formulation to compute the task assignment. Our problem differs from [21] in that the cost of an allocation cannot be estimated beforehand, the tasks and hence the clustering approach are different.

III. Problem Description

A. Mathematical Preliminaries

Let $\mathscr{W} = [0, L_1] \times [0, L_2] \times \cdots \times [0, L_\nu] \subset \mathbb{R}^\nu$ denote a ν-dimensional workspace that is to be explored by the robots. Each robot has an (identical) n-dimensional state space $Q = \mathscr{W} \times V$ ($n \geq \nu$). V is comprised of the robot state components, such as velocities or orientations, that does not affect what the sensor sees. Let $q_i : [0, T] \to Q$ denote a trajectory of the i^{th} robot in its state space with $T \in \mathbb{R}^+$ representing the time horizon. Let the set of all state space trajectories be H. Let $P : Q \to \mathscr{W}$ project the state space into the workspace. The robots have deterministic dynamics given by $\dot{q}_i(t) = f(q_i(t), u_i(t))$, where $u_i(t) \in \mathbb{R}^m$ is the control input of the i^{th} robot.

Let $c(x, q_i) : \mathscr{W} \times H \to [0, 1]$ denote the time-averaged statistics of a trajectory q_i, which is defined as:

$$c(x, q_i) = \frac{1}{T} \int_0^T \delta(x - P(q_i(\tau)))d\tau, \tag{1}$$

where δ is a Dirac function. Let $\phi : \mathscr{W} \to \mathbb{R}$ denote a static information map that describes the amount of information at each location in the workspace. In this work, each information map is a probability distribution with $\int_{\mathscr{W}} d\phi = 1$ and $\phi(x) \geq 0, \forall x \in \mathscr{W}$. An ergodic metric [10] for a trajectory q_i

and an information map ϕ is defined as:

$$\mathscr{E}(\phi,q_i)=\sum_{k=0}^{K}\lambda_k(c_k-FC_k)^2 \qquad (2)$$
$$=\sum_{k=0}^{K}\lambda_k\left(\frac{1}{T}\int_0^T F_k(q(\tau))d\tau-FC_k\right)^2$$

where (i) $FC_k=\int_{\mathscr{W}}\phi(x)F_k(x)dx$ represents the k^{th} Fourier coefficient of the information map, $F_k(x)=\frac{1}{h_k}\Pi_{j=1}^{v}\cos(\frac{k_j\pi x_j}{L_j})$ is the cosine basis function for index $k\in\mathbb{N}^v$ and K is the number of Fourier bases considered, (ii) c_k denotes the k^{th} Fourier coefficient of $c(x,q_i)$, (iii) h_k denotes the normalization factor as defined in [10], and (iv) $\lambda_k=(1+||k||^2)^{-\frac{v+1}{2}}$ denotes the weight for each corresponding Fourier coefficient.

Consider a set Φ of M information maps and a set R of N agents with $M\geq N$. Let $\mathscr{A}$ denote the set of all possible allocations of agents to maps. Each allocation is a *surjective* function $A:\Phi\rightarrow R$. Let $h(r,A)$ be the set of maps assigned to robot r by allocation $A\in\mathscr{A}$.

A scalarized information map combines a set S of s information maps, using weights w_i for each map, into a single map on which ergodicity can be optimized:

$$\phi_S=\frac{\sum_{i=1}^{s}w_i\phi_i}{\sum_{i=1}^{s}w_i} \qquad (3)$$

B. Problem Formulation

Each problem has N agents tasked to cover a domain subject to M information maps that span the same physical region, $M\geq N$. We present results for identical forward-moving-only differential-drive robots, varying only in initial position, with $\mathscr{W}=[0,100]\times[0,100]$ and $Q=\mathscr{W}\times SO\{2\}$. The dynamics for the i^{th} robot are shown in (4), in which $v=2,n=3,m=2$.

$$\dot{q_i}=f(q_i(t),u_i(t))=\begin{bmatrix}v_i cos(\theta_i(t))\\ v_i sin(\theta_i(t))\\ \alpha*\omega_i\end{bmatrix} \qquad (4)$$

where,$[v_i,\omega_i]$ is the control input for robot i, α is a constant and $q_i(t)=[x_i(t),y_i(t),\theta_i(t)]$ is the state of robot i at time t.

Let $q_r^*(A)$ be the trajectory (satisfying the dynamics above) that minimizes ergodicity on the scalarization of the maps assigned to robot r by allocation A:

$$q_r^*(A)=\underset{q\in H}{\operatorname{argmin}}\,\mathscr{E}(\phi_{h(r,A)},q) \qquad (5)$$
$$s.t.\quad q(0)=q_r(0)$$

$q_r^*(A)$ is obtained in practice via trajectory optimization on the scalarized information map as in [4] with all maps weighted equally. Then the allocation optimization problem can be stated as follows:

$$\underset{A\in\mathscr{A}}{\operatorname{argmin}}\max_{\phi\in\Phi}\mathscr{E}(\phi,q^*_{A(\phi)}(A)) \qquad (6)$$

which finds the allocation for which the map with the largest ergodicity has the smallest achievable ergodicity.

IV. Method

We consider three baseline approaches explained further in Section V-A. One approach involves optimizing a single joint trajectory for all agents on the average of all the maps, which may not be optimal for individual maps. Another approach is a suboptimal greedy allocation method that assigns each agent to a map based on the information around the agent. The optimal solution can be found by exhaustive search, but this is not practical for larger numbers of agents and maps. To tackle this we present a branch and bound formulation.

A. Branch and bound approach

The branch and bound algorithm is commonly used in literature to speed up the exhaustive search in combinatorial problems such as equation (6). It constructs a tree structure and uses bounds to eliminate sub-problems that cannot contain the optimal solution. In a minimization problem, if the cost of a sub-problem is greater than the current best cost (upper bound), this sub-problem can be eliminated. In this approach, we leverage the minmax metric to construct a branch and bound algorithm that reduces the number of possible allocations to be checked to find the optimal allocation. The formulation is described in Algorithm 1.

The algorithm begins by computing an incumbent solution using the greedy allocation described in Section V-A.2 (Step 1). For this incumbent allocation, the ergodic trajectory for each agent is computed using (5), and the resulting individual ergodicities (using **getIndvErg()**) on the information maps are computed using (2) (Steps 2,3). The maximum of the individual ergodicities is set as the initial upper bound for the branch and bound algorithm (Step 4).

The algorithm works on a tree structure where each node corresponds to an allocation of maps to one agent. It first creates a root node with a null allocation (Step 5). Each subsequent level of the tree contains nodes corresponding to possible allocations of maps to one agent. Hence, the depth of the tree generated is equal to the number of agents in the problem, and a path from the root node to a leaf node corresponds to one complete allocation.

For every node in the tree, the individual ergodicities on the maps are computed using equations (5) and (2) (Step 12). If the maximum of the individual ergodicities is greater than the current upper bound, any solution containing this partial allocation will have a minmax metric greater than the current upper bound, and hence the node is pruned (Steps $13-15$). Whenever a complete solution is obtained, the upper bound is updated to facilitate more node pruning (Steps $17-24$).

On the tested problems this branch and bound algorithm achieves an average speedup of 16 times compared to the exhaustive search algorithm while maintaining optimality. However, the branching factor is equal to the size of the power set of the number of unassigned information maps and it overlooks the similarity between information maps while allocating agents. To this end, we present an approach based on clustering to reduce the size of the allocation search space and thereby significantly reduce the runtime of the algorithm.

Algorithm 1: Branch and Bound Algorithm

Data: Information maps: $\Phi = \{\phi_1, \cdots, \phi_M\}$, Agent start positions: $S_0 = \{s_1, \cdots, s_N\}$
Result: Optimal allocation scheme

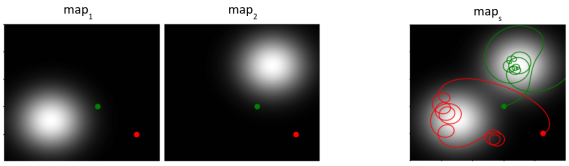
```
incumbent ← GreedyAllocation(Φ, S_0);
best_alloc ← incumbent;
indv_erg ← getIndvErg(incumbent, S_0);
UB ← max(indv_erg);
root_node ← ∅;
candidate_nodes ← [root];
for i ∈ [1, N] do
    new_nodes ← ∅
    for n ∈ candidate_nodes do
        allocs ← Possible assignments of maps left
          for agent i;
        for alloc ∈ allocs do
            indv_erg ← getIndvErg(alloc, s_i);
            if max(indv_erg) > UB then
                continue; // Prune
            end
            new_nodes.append(alloc);
            if i == N then
                new_alloc ← path from root to node;
                new_UB ←
                  max(getIndvErg(new_alloc, S_0));
                if new_UB < UB then
                    UB ← new_UB;
                    best_alloc ← new_alloc;
                end
            end
        end
    end
    candidate_nodes ← new_nodes;
end
return best_alloc
```

B. Branch and bound with similarity clustering

This approach is motivated by the fact that if two information maps are similar, then a single agent can be assigned both information maps. We define the similarity (sim_{kl}) between two information maps ϕ_k and ϕ_l to be the norm of the difference in the Fourier coefficients of the information maps as shown in (7) [10][4].

$$sim_{kl} = \|FC_k - FC_l\| \tag{7}$$

where, FC_i and FC_j are the Fourier coefficients of ϕ_k and ϕ_l respectively. The information maps are then clustered based on this similarity metric using K-means clustering. The number of clusters is picked using the kneedle algorithm [22], which identifies the points of maximum curvature on a dataset while ensuring it is greater than or equal to N. We then follow the branch and bound algorithm as described in Algorithm 1, except now, each level corresponds to the possible allocations of clusters to one agent. The tree thus formed has the same depth but has a lower branching factor than the branch and bound approach.



(a) Information maps (b) Scalarized information map

Fig. 2: (a) Example problem with two maps and two agents R (red) and G (green) with specified start position and zero orientation, (b) Trajectories of agents on the scalarized information map (map_s)

Method	Ergodicity evaluated on		
	Scalarized Map	Map 1	Map 2
JTO	0.0079101	0.2536889	0.1626450
Greedy Allocation	-	0.0009996	0.0009993

TABLE I: Individual ergodicities on maps in Fig 2a using joint trajectory optimization (JTO) and greedy allocation

V. Key Results and Analysis

A. Baseline Methods and Implementation

1) Joint Trajectory Optimization (JTO): We adapt our prior work in single-agent multi-objective ergodic search [4] to multiple agents, in a similar way as in [10], by optimizing a concatenated trajectory of all agents for the ergodic metric on a scalarization of all maps with $w_i = 1$ in equation (3).

For each agent, i, consider a trajectory $q_i(t)$ of length T. The concatenated trajectory of length NT can be represented as $q(t) = [q_1, q_2, \cdots, q_N]'$. The concatenated trajectory is then optimized to minimize the ergodic metric ($\mathscr{E}(\phi_\Phi, q)$), calculated using equation (2). This strategy is implemented on an example problem with two information maps and two agents with random start positions, as shown in Fig. 2a. The resulting trajectories of the two agents and the individual ergodicities are shown in Fig. 2b and Table I, respectively.

Since it is a joint trajectory optimization, the agents naturally divide the high information regions among them. This seems favorable, however, since the agents are trying to consider all the M maps simultaneously, the best ergodicity it can achieve on each map without compromising others lies on the Pareto optimal front by definition. A visual representation of this limitation can be seen in Fig. 3 where agents spend a lot of time in low-information regions of one map, indicated by the portion of the trajectory in the white box, as that region spatially corresponds to a high information region on another map. This problem can be overcome by allocating these conflicting maps instead of having each agent consider all the maps.

2) Greedy Allocation: The greedy baseline is presented in algorithm 2. For each agent and information map combination, the amount of information inside an arbitrary-sized window centered on the agent in that information map is computed as the score of that agent-map combination (Steps $3-7$). The agent with the maximum score on a map is assigned to that map unless this would leave more unassigned

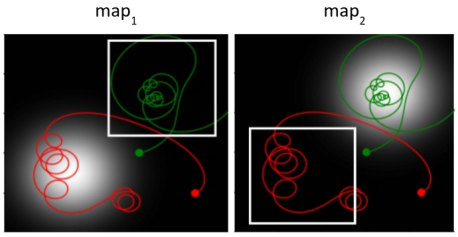


Fig. 3: Agent trajectories on the individual maps in Fig. 2a

Algorithm 2: GreedyAllocation

Data: Information maps: $\Phi = \{\phi_1, \cdots, \phi_M\}$, Agent start positions: $S_0 = \{s_1, \cdots, s_N\}$
Result: Allocation Scheme

1 $scores \leftarrow zero\_matrix(M,N)$;
2 $W \leftarrow$ window centered on each start position;
3 **for** $\phi_i \in \Phi$ **do**
4 **for** $s_j \in S_0$ **do**
5 $scores[i][j] \leftarrow \sum_{x,y \in W_j} \phi_i(x,y)$;
6 **end**
7 **end**
8 $allocation \leftarrow \{\}$;
9 **for** $i \in [1,M]$ **do**
10 $allocation[i] \leftarrow$ **getBestAgent**$(scores[i])$;
11 **end**

agents than maps, in which case the next highest score is assigned. This ensures all agents are assigned at least one map (implemented as **getBestAgent**(scores) in Step 10). This is tested on the example in Fig. 2a, and the trajectories obtained from that allocation are plotted in Fig. 4. The agents now spend less time in areas of low information. It can be seen from Table I that the individual maps are better covered when the agents are explicitly allocated. Though the greedy allocation performs better than JTO, it does not guarantee an optimal allocation as defined in Section III-A.

3) Exhaustive Search: An optimal baseline is an exhaustive search over $\mathscr{A}$. The algorithm iterates through all possible allocations for the given set of agents and information maps. For each allocation, the individual ergodicities on the information maps are computed. It is worth noting that when an agent is assigned more than one information map, we scalarize the maps using equal weights, but this can be adapted according to user/application requirements. Finally, the optimal allocation is chosen as described in equation (6).

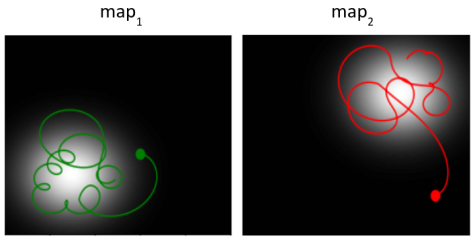


Fig. 4: Trajectories obtained using a greedy allocation of agents green and red to map_1 and map_2, respectively

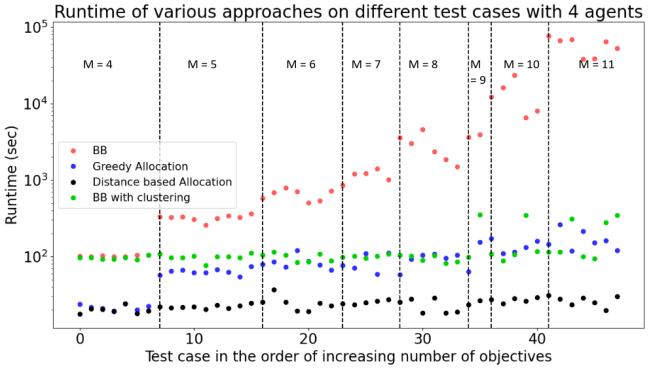


Fig. 5: Runtime comparison of all approaches on test cases with 4 agents and 4 to 11 maps as indicated by M

B. Numerical Results

The effectiveness of the proposed branch and bound (BB) algorithm was tested on 100 and 47 randomly generated test cases with 3 and 4 agents, respectively. Each test case contained 3 to 11 information maps (objectives) and a random start pose for each of the agents. Each information map was modeled as a mixture of Gaussian distributions with the number of Gaussian distributions being uniformly distributed in the range 1 to 6. Each Gaussian peak had a mean and deviation randomly sampled between 0.05 to 0.8 and 0.01 and 0.05 respectively along both the x and y axis. The branch and bound algorithm was compared against (a) Exhaustive Search, (b) Greedy Allocation, and (c) Multi-agent Multi-target task allocation [8] and (d) Branch and bound with clustering. The exhaustive search was run with a net time limit of 100 hr. The approaches are evaluated against the branch and bound algorithm in terms of runtime and the difference in the minmax metric. The results for test cases with 4 agents are shown in Table II.

C. Comparison of BB and Exhaustive Search

The branch and bound algorithm finished solving all the test cases within 100 hr for both 3 and 4 agents while the exhaustive search algorithm solved only 4 test cases with 3 agents within 100 hr. Thus, the branch and bound algorithm described in Section IV-A achieves an average speedup of 16 times while maintaining optimal allocation.

D. Comparison BB and Greedy Allocation

The greedy allocation, on average, showed 89.62% reduction in runtime compared to the branch and bound algorithm. However, the minmax metric is higher compared to the branch and bound approach, as shown in Table II, as the approach only takes information in the local neighborhood of agents for task assignment. The comparison of runtime for the greedy algorithm and the branch and bound algorithm for test cases with 4 agents is shown in Fig. 5.

E. Comparison BB and Distance-based assignment

For this comparison, the information maps in a test case are clustered as described in Section IV-B but assigned based on distance as in [8]. The peaks on the average of the maps

Method	Average Percentage		Average		
	Increase in minmax metric	Improvement in Runtime	Runtime (s)	Increase in minmax metric	Maximum difference in minmax metric
Greedy Allocation	**330.01%**	**89.62%**	**91.4**	**0.1029**	**1.3778**
Distance-based Allocation	**239.72%**	**94.85%**	**23.9**	**0.0722**	**1.3778**
Branch and bound with similarity clustering	**53.05%**	**77.97%**	**123.4**	**0.0148**	**0.1138**

TABLE II: Comparing runtime and minmax metric against the branch and bound algorithm (optimal)

in a cluster are identified. The agents are assigned to the clusters based on the distance of the agent to the centroid of these peaks. The algorithms are compared on runtime as shown in Fig. 5. Assigning based on distance shows a 94.85% improvement in runtime compared to the branch and bound algorithm. However, the minmax metric is higher for the former as the approach considers visiting peaks in the information map rather than ergodic coverage.

F. Comparison BB and BB with clustering

As indicated in Table II, the branch and bound approach with clustering shows 77.97% lesser runtime compared to one without clustering (32 times faster than exhaustive search). Even though branch and bound with clustering is not exhaustive the difference in the minmax metric of its solution had an average difference and standard deviation of only 0.0148 and 0.1138 compared to the optimal solution. The runtime for the algorithms on test cases with 4 agents is shown in Fig. 5.

VI. Conclusion and Future Work

The multi-agent multi-objective ergodic search problem is an allocation problem that is NP-hard to solve optimally. An exhaustive search algorithm, while optimal in allocation, quickly becomes intractable in terms of runtime as the number of agents and maps increases. We thus present a branch and bound algorithm that helps reduce the average runtime by a factor of 16 while still providing the optimal allocation scheme. Further, we present an approach to cluster similar information maps and assign agents to these clusters. This method is 32 times faster than the exhaustive search, and it achieves a better trade-off between solution quality and runtime. Future directions include adapting the task allocation scheme to heterogeneous agents. Additional metrics such as workload division and agent-specific task allocation can also be investigated.

References

[1] J. Berger, N. Lo, and M. Noel, "A new multi-target, multi-agent search-and-rescue path planning approach," *International Journal of Computer and Information Engineering*, vol. 8, no. 6, pp. 978 – 987, 2014.

[2] F. M. D. Fave, S. Canu, L. Iocchi, D. Nardi, and V. A. Ziparo, "Multi-objective multi-robot surveillance," *2009 4th International Conference on Autonomous Robots and Agents*, pp. 68–73, 2000.

[3] S. G. Satpute, P. Bodin, and G. Nikolakopoulos, "Cooperative planning for multi-site asteroid visual coverage," *Advanced Robotics*, vol. 35, no. 21-22, pp. 1332–1346, 2021.

[4] Z. Ren, A. K. Srinivasan, H. Coffin, I. Abraham, and H. Choset, "A local optimization framework for multi-objective ergodic search," in *Robotics: Science and Systems XVIII*. Robotics: Science and Systems Foundation, jun 2022.

[5] B. P. Gerkey and M. J. Matarić, "A formal analysis and taxonomy of task allocation in multi-robot systems," *The International Journal of Robotics Research*, vol. 23, pp. 939 – 954, 2004.

[6] G. A. Korsah, A. Stentz, and M. B. Dias, "A comprehensive taxonomy for multi-robot task allocation," *The International Journal of Robotics Research*, vol. 32, no. 12, pp. 1495–1512, 2013.

[7] L. Bramblett, R. Peddi, and N. Bezzo, "Coordinated multi-agent exploration, rendezvous, and task allocation in unknown environments with limited connectivity," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 12 706–12 712.

[8] S. Biswas, S. G. Anavatti, and M. A. Garratt, "A time-efficient co-operative path planning model combined with task assignment for multi-agent systems," *Robotics*, vol. 8, no. 2, 2019.

[9] C. Ramirez Atencia, J. Del Ser, and D. Camacho, "Weighted strategies to guide a multi-objective evolutionary algorithm for multi-uav mission planning," *Swarm and Evolutionary Computation*, vol. 44, pp. 480–495, 2019.

[10] G. Mathew and I. Mezić, "Metrics for ergodicity and design of ergodic dynamics for multi-agent systems," *Physica D: Nonlinear Phenomena*, vol. 240, no. 4, pp. 432–442, 2011.

[11] L. M. Miller and T. D. Murphey, "Trajectory optimization for continuous ergodic exploration," in *2013 American Control Conference*, 2013, pp. 4196–4201.

[12] E. Ayvali, H. Salman, and H. Choset, "Ergodic coverage in constrained environments using stochastic trajectory optimization," 2017.

[13] I. Abraham, A. Prabhakar, and T. D. Murphey, "An ergodic measure for active learning from equilibrium," *IEEE Transactions on Automation Science and Engineering*, vol. 18, no. 3, pp. 917–931, 2021.

[14] L. M. Miller, Y. Silverman, M. A. MacIver, and T. D. Murphey, "Ergodic exploration of distributed information," *IEEE Transactions on Robotics*, vol. 32, no. 1, pp. 36–52, 2016.

[15] I. Abraham and T. D. Murphey, "Decentralized ergodic control: Distribution-driven sensing and exploration for multiagent systems," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 2987–2994, Oct 2018.

[16] A. Mavrommati, E. Tzorakoleftherakis, I. Abraham, and T. D. Murphey, "Real-time area coverage and target localization using receding-horizon ergodic exploration," 2017.

[17] Z. Ren, A. K. Srinivasan, B. Vundurthy, I. Abraham, and H. Choset, "A pareto-optimal local optimization framework for multiobjective ergodic search," *IEEE Transactions on Robotics*, pp. 1–12, 2023.

[18] B. Gerkey and M. Mataric, "Multi-robot task allocation: analyzing the complexity and optimality of key architectures," in *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, vol. 3, 2003, pp. 3862–3868 vol.3.

[19] J. Butzke and M. Likhachev, "Planning for multi-robot exploration with multiple objective utility functions," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011, pp. 3254–3259.

[20] H. Karami, A. Thomas, and F. Mastrogiovanni, *Task Allocation for Multi-robot Task and Motion Planning: A Case for Object Picking in Cluttered Workspaces*, 01 2022, pp. 3–17.

[21] Y.-C. Ma, T.-F. Chen, and C.-P. Chung, "Branch-and-bound task allocation with task clustering-based pruning," *Journal of Parallel and Distributed Computing*, vol. 64, no. 11, pp. 1223–1240, 2004.

[22] V. Satopaa, J. Albrecht, D. Irwin, and B. Raghavan, "Finding a "kneedle" in a haystack: Detecting knee points in system behavior," in *2011 31st International Conference on Distributed Computing Systems Workshops*, 2011, pp. 166–171.