Large-scale Heterogeneous Multi-Robot Coverage via Domain Decomposition and Generative Allocation

Jiaheng Hu, Howard Coffin, Julian Whitman,
Matthew Travers, and Howie Choset

Carnegie Mellon University, Pittsburgh PA 15289, USA
{jiahengh, hcoffin, jwhitman, mtravers, choset}@andrew.cmu.edu

Abstract. This paper develops a new approach to direct a set of heterogeneous agents, varying in mobility and sensing capabilities, to quickly cover a large region, say for example in the search for victims after a large-scale disaster. Given that time is of the essence, we seek to mitigate computational complexity, which normally grows exponentially as the number of agents increases. We create a new framework which reduces the planning complexity through simultaneously decomposing a target domain into sub-regions, and assigning a team of agents to each sub-region in the target domain, as a way to decompose a large-scale problem into a set of smaller problems. The teams are formed to optimize the coverage of each sub-regions. Doing so requires both the utilization of individual agents' strengths as well as their collaborative capabilities. We determine the ideal team by introducing a novel evolution-guided generative model based on generative adversarial networks (GANs) that creates allocation plans from the sub-region features in a computationally efficient manner. We validate our framework on a real-world satellite images dataset, and demonstrate that through decomposition and generative allocation, our method has significantly better efficiency and efficacy compared to current centralized multi-robot coverage methods, and is therefore better suited for large-scale time-critical deployment.

Keywords: Mutli-robot Coverage, Task Allocation, Generative Adversarial Networks

1 Introduction

This paper considers large-scale deployment of heterogeneous robots in time-critical scenarios, for applications such as search and rescue or disaster response. In such applications, coverage of a region can be optimized by combining the different motion and sensing capabilities of multiple agents. Coordinating and effectively using these heterogeneous capabilities requires a significant computational effort, which may not be available due to hardware and time constraints. Specifically, centralized multi-robot coverage algorithms typically suffer from exponential growth in planning time as the number of agents grow, making them unsuitable for large-scale deployment [6, 27, 29].

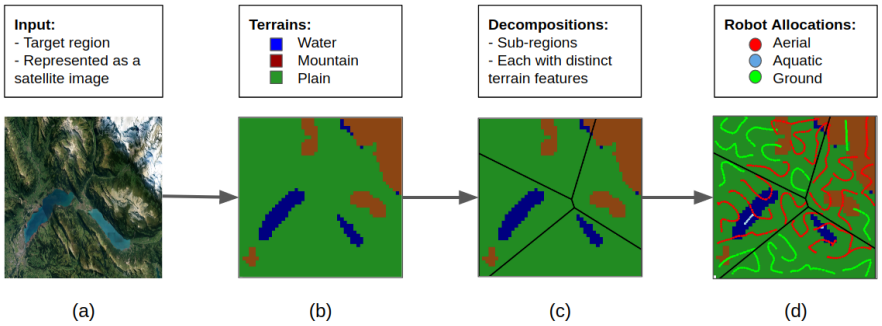

Fig. 1: We consider large-scale deployment of heterogeneous robots in the context of multi-robot coverage. For a given target region (a) to cover, our framework takes as input its corresponding labelled terrain map (b), then decomposes it into sub-regions (c) based on terrain features. Robot teams (d) are subsequently allocated to each of the sub-regions based on the terrain features. Each robot team plans its trajectories in a decentralized fashion, where the colored lines in (d) correspond to the trajectories of each agent type as they search the region.

We alleviate the complexity of coordinating large-scale heterogeneous teams by breaking down the coverage planning problem into a centralized “decomposition and allocation" step, and a decentralized “trajectory planning" step. Our framework decomposes a target region (e.g. a map of an environment) into sub-regions based on terrain features, and then assigns agents to teams that are deployed to each sub-region. By simultaneously partitioning the search domain into sub-regions and the agents into teams, we naturally decompose a large coverage problem into a set of smaller problems. This allows us to plan the trajectory of each team of robots distributively while still considering the global distribution of terrain and information.

Our framework addresses the challenge of determining the robot teams for each sub-region, while accounting for both a sub-region’s features and each robot’s capabilities. This problem, often referred to as centralized multi-robot task allocation (MRTA) [18], is NP-hard [20]. A wide range of approaches have been proposed to obtain approximate solutions for centralized MRTA [31, 26, 23, 36, 35, 40]. However, these approaches typically require repeated evaluation of performance for different team plans on different tasks, and can become computationally expensive when scaling to large teams of robots. Besides computational constraints, an additional concern in multi-robot deployment is deciding how many robots to deploy. Deploying more robots will naturally result in better coverage, but will simultaneously induce higher deployment cost. As a result, a single solution is often not sufficient: a user supervising the robot teams may desire multiple solutions that exemplify the trade-offs between the two conflicting objectives of coverage performance and deployment cost. Adding this property to the team formation problem makes it a multi-objective combinatorial opti-

mization problem where our goal is to find team allocation plans with optimal trade-offs across the objectives; i.e. the Pareto set [38] of allocation plans.

To optimize for multiple objectives while maintaining computational efficiency, we introduce a novel generative approach that learns correlations between tasks and desirable team allocations. We train an *allocation generator*, implemented as a neural network, to learn a one-to-many mapping from tasks, represented as sub-regions, to a set of Pareto-optimal allocations reflecting the trade-off between conflicting objectives. Unlike past works using GANs, where a dataset is present prior to training [12, 30], we have no data *a priori* from which a mapping from tasks to allocation plans can be distilled. Worse yet, collecting such a dataset from scratch is enormously computationally burdensome, as obtaining each tasks-allocation pair requires solving a combinatorial optimization problem. Instead, our approach actively collects data on-line during training through a novel evolution-guided data creation process inspired by multi-objective evolutionary algorithms. This data creation process looks for new promising allocation plans around the current generated allocations, promoting the generator to improve the quality of its output. Once the generator is trained, we can query for allocation plans through an inexpensive neural network forward pass, making task allocation computationally efficient at run-time.

The decomposition and allocation steps give us a set of sub-problems for each team-region combination. We apply a local trajectory planner [28] to plan trajectories maximizing coverage of each sub-region. We tested our algorithm on the DroneDeploy dataset of labelled satellite images [32], and demonstrated significantly improved coverage performance compared to fully centralized coverage planning. Importantly, after the off-line training of the allocation generator is completed, the entire decomposition and allocation pipeline can be completed for a team of 60 robots within 30 seconds on a laptop computer during testing, and is therefore suitable for time-critical deployment.

2 Domain Decomposition

The first step in our framework is domain decomposition, where the search domain is partitioned into smaller regions, to which agents will be allocated. The domain decomposition step takes in a labelled terrain map M_{terr} where each pixel value belongs to a finite list of terrain types, and partitions it into k non-overlapping sub-regions $[M_1, ..., M_k]$, with k determined by the user. The terrain feature distribution f_i of each sub-region M_i can be represented as a categorical distribution over terrain types, with each entry corresponding to the proportion of pixels of a specific terrain type in the given sub-region. In addition, we keep track of the size of each sub-region as $S = [\text{area}(M_1), ..., \text{area}(M_k)]$.

The approach we use for decomposition is to place and optimize the location of generator points of a Voronoi diagram [15, 25] with respect to a cost function $\mathcal{C}$, i.e., we seek:

$$P^\star = \arg\min_P \mathcal{C}(P) \tag{1}$$

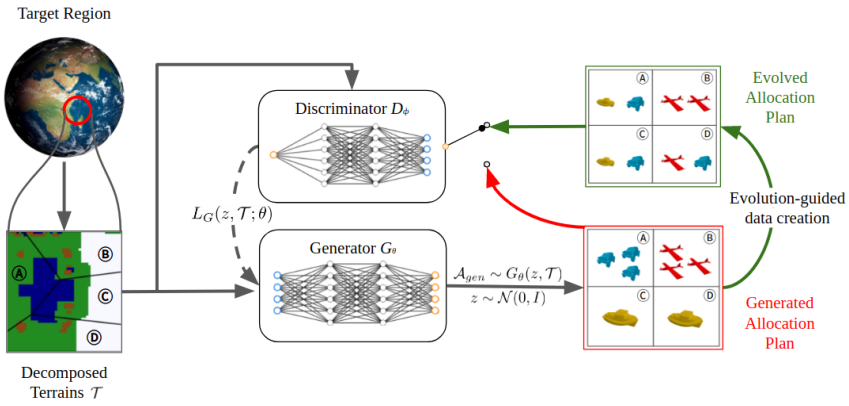

Fig. 2: Illustration of our generative allocation training algorithm. At each training iteration, a partitioned terrain is sampled from the training dataset and passed into the generator. The generator (bottom center) maps this partitioned terrain into a population of allocation plans (bottom right, showing one example allocation plan). The evolution-guided data creation step explores around the generated allocation plans by evolving them using a procedure inspired by Multi-objective Evolutionary Algorithms, and creates a population of evolved allocation plans (top right, showing one example allocation plan). The discriminator (top center) takes as input the terrain and an allocation plan that is either from the generated allocations or the evolved allocations, and tries to distinguish which population the allocation comes from. The output of the discriminator feeds into the loss function L_G, which encourages the generator to learn to "trick" discriminator by generating high-quality allocation plans.

where $P = (p_1, ..., p_k)$ is the 2D positions of the generator of each of the k sub-regions.

An ideal cost function should utilize terrain information to guide the decomposition process. In this work, we proposed a hand-crafted cost function that encourages heterogeneity in the terrain distribution of each sub-regions, thereby encouraging the formation of multi-ability teams. Formally,

$$\begin{aligned} \mathcal{C}_f(P) &= -\frac{1}{k}\sum_{i=1}^{k} H(f_i) \\ \mathcal{C}_b(P) &= \text{var}(S) \\ \mathcal{C}(P) &= \mathcal{C}_f(P) + \lambda_{terr} * \mathcal{C}_b(P) \end{aligned} \tag{2}$$

with $H(f_i)$ the entropy of the terrain features distribution for sub-region M_i associated with the generator $p_i \in P$, $\text{var}(S)$ the variance of the sub-region areas, and λ_{terr} the weight. In other words, we seek to maximize the entropy

of each sub-region's terrain distribution ($\mathcal{C}_f$) while minimizing the variance of each sub-region's size ($\mathcal{C}_b$). Notice that alternative objective functions may also be considered, and it is unclear which objective function is best suited for the decomposition task. Theoretical analysis of how different decomposition strategy will affect the eventual coverage results is an open question that may be addressed by future works.

Due to the time-critical nature of our applications, we do not look for an exact solution to (1). Instead, we approximately solve for the decomposition through differential evolution [33], a gradient-free black box optimization method. An example of result of domain decomposition is shown in Fig. 1.

3 Generative Task Allocation

The task allocation step in our pipeline takes in the sub-regions (tasks) generated by the decomposition phase and determines a set of heterogeneous robots $\mathcal{R} = \{r_1, ..., r_N\}$ to allocate to each sub-region. Each robot r_i belongs to a species [34], $o_i \in \mathcal{O}$ which determines its motion and sensing capability. A robot team a_i is defined as a subset of available robots $\mathcal{R}$ which act as a team to complete a task. An allocation plan $\mathcal{A} = \{a_1, \dots a_K\}$ consists of K teams, one for each task, where $\bigcup_{i=1}^{K} a_i \subseteq \mathcal{R}$, and $\forall i \neq j : a_i \cap a_j = \varnothing$. We numerically represent an allocation plan A as a $K \times \mathcal{O}$ integer matrix in this work, where $A(i, j)$ represents the the number of robot of species j to allocate to task i. We denote the *coverage cost* of a robot team a on task t as $f(a, t) \in \mathbb{R}$, and use the sum of the coverage cost on all tasks as the coverage cost for the team. The coverage cost measure is calculated based on the local trajectory planner, explained in detail in Sec. 4. In addition, each robot r_i deployed incurs a deployment cost c_i, where the total deployment cost is the sum of the cost of all deployed robots. Both the deployment cost and the coverage cost are objectives to be minimized. We frame task allocation as an multi-objective optimization problem, where the goal is to find a set of allocations with optimal trade-offs between the two conflicting objectives, i.e. the Pareto-optimal allocations.

3.1 Preliminaries

Due to the time-critical nature of multi-robot search and coverage, we seek team allocation methods that can operate in near real-time, and propose a novel generative task allocation approach combining ideas from evolutionary algorithms and generative adversarial networks. We briefly review evolutionary algorithms and generative models in the following section.

Evolutionary Algorithms Evolutionary algorithms (EAs) have been widely adopted for solving multi-robot task allocation problems [41, 24, 39]. EAs represent a class of population-based metaheuristic optimization algorithms that are inspired by biological evolution. This class includes genetic algorithms [16],

differential evolution [33], ant colony optimization [11], and particle swarm optimization [17]. EAs maintain and update a population of candidate solutions through a set of operations, and are capable of finding optimal or near-optimal solutions to NP-hard problems within tractable time [9]. They are also able to find multiple solutions due to their population-based nature, and are thus suitable for multi-objective optimizations [8, 37]. A multi-objective genetic algorithm was used in [3] to obtain a Pareto-optimal set of solutions for MRTA. A multi-objective particle swarm optimization (MOPSO) was used in [31] to handle task allocation with multiple objectives. However, since EAs require repeated fitness function evaluations, each of which may involve planning and simulation, EAs can quickly become computationally expensive when dealing with relatively costly fitness functions [7] as are often found in real-world multi-robot applications. As a result, these algorithms are not suitable for time-critical situations.

Generative Models To generate allocation plans in a time-efficient manner, we propose to learn a mapping from a list of tasks to a *distribution* of allocation plans representing a set of Pareto-optimal solutions, through training a generative model. Generative Adversarial Networks [12] (GANs) are implicit generative models that learn patterns within a dataset, such that the model can be used to generate new samples as if they were drawn from the same underlying distribution as the data. A GAN consists of two components: a generator that learns a mapping from a noise vector to generated data sample, and a discriminator that learns to distinguish generated samples from real samples. These two components are typically implemented as deep neural networks and are optimized simultaneously through gradient descent. Since the two networks have competing objectives, training a GAN can be viewed as two players playing a minimax game [12]. The conditional GAN [30] was later introduced as a variant that output labeled samples by conditioning both the generator and the discriminator on a given label. Another variant of the GAN known as the Wasserstein GAN (WGAN) [4] minimizes an approximation of the Earth Mover distance [21] between the target distribution and the generated distribution. Improved WGAN [13] was later introduced to stabilize GAN optimization by using a gradient penalty term instead of the original gradient clipping scheme in WGAN. In our work, we adopt a conditional variant of the improved WGAN to learn a mapping from tasks to allocations, and train it with a novel evolution-guided data creation step.

3.2 Evolution-Guided GAN

Our generative task allocation system is composed of three key components: a generator, a discriminator, and a evolution-guided data creator. An illustration of these components is shown in Fig. 2. The generator maps tasks to allocation plans. The discriminator tries to distinguish between allocation plans generated by the generator, and allocation plans produced by the evolution-guided data creator. Both the generator and the discriminator are implemented as neural

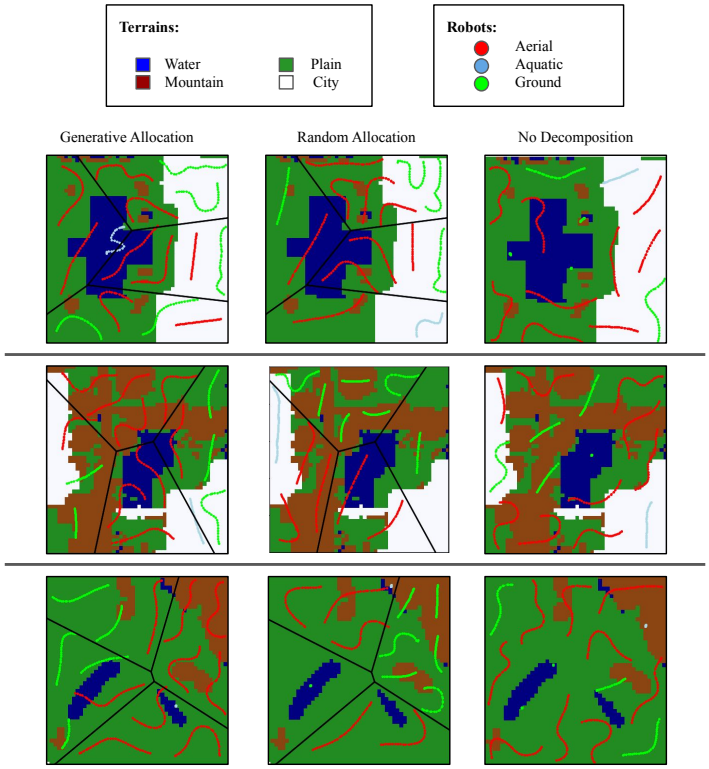

Fig. 3: A visual comparison of coverage performance between decomposition with generative allocation (left column, ours), random allocation (middle column), or no decomposition (right column), on three different test terrains (top, middle, and bottom rows). On all three test terrains, our method achieves higher coverage over the target region than the other two methods.

networks and are trained using a conditional variant of improved WGAN [30, 13]. The evolution-guided data creator is introduced to obtain training data for the discriminator, by performing n steps of multi-objective evolution on the generated allocation plans at each iteration. These evolution steps iteratively improve the output of the generator, guiding the generator to approximate a distribution of increasingly higher-quality candidate solutions.

Generator The generator $\mathcal{G}_\theta(z, \mathcal{T})$ is implemented as a multi-layer perceptron (MLP) network with 3 hidden layers of size 64, with batch normalization [14] and ReLU activation [2] at each hidden layer. $\mathcal{G}_\theta(z, \mathcal{T})$ takes in a P-dimensional vector $z \in \mathbb{R}^P$ sampled from a standard multivariate Gaussian distribution, $z \sim \mathcal{N}(0, I)$, and a $K \times M$ terrain features matrix representing the queried tasks $\mathcal{T}$, where K is the number of sub-regions, and M is the number of distinct terrain

types. Each row of the terrain features matrix correspond to the terrain features distribution of a specific sub-region, and sum to 1. The output of the generator is an allocation plan, represented as a dense non-negative $K \times \mathcal{O}$ matrix $\mathcal{A}_{gen}$, where O is the total number of robot species. Each entry of the output $\mathcal{A}_{gen}(i, j)$ specifies the fraction of robots of species j to deploy to task i. The integer number of robots of species j to deploy to sub-region i can then be calculated through a continuous relaxation $N_j^{(i)} = \text{round}(N_j \times \mathcal{A}_{gen}(i, j))$, where N_j is the total available number of robots of species j.

The generator loss to be minimized is:

$$\mathcal{L}_{\mathcal{G}}(z, \mathcal{T}; \theta) = -\mathcal{D}_{\phi}(\mathcal{G}_{\theta}(z, \mathcal{T}), \mathcal{T}) \tag{3}$$

where $\mathcal{D}_{\phi}$ is the discriminator parameterized by ϕ, explained in detail in the next paragraph.

Discriminator The discriminator $\mathcal{D}_{\phi}(\mathcal{A}, \mathcal{T})$ is also implemented as a multi-layer perceptron (MLP) network with 3 hidden layers of size 64, with batch normalization [14] and ReLU activation [2] at each hidden layer. $\mathcal{D}_{\phi}(\mathcal{A}, \mathcal{T})$ takes in an allocation plan $\mathcal{A}$ and a queried set of tasks $\mathcal{T}$ and outputs a scalar. Similar to the canonical GAN, this scalar can be viewed as a prediction of whether the design is generated by the generator. The training data for the discriminator comes from two sources. Half of the allocation plans are synthesised by the generator, and are labeled 0 ("fake"). The other half are obtained from the evolution-guided data creation step (described in the next sub-section) and are labeled 1 ("real").

The discriminator is then optimized to maximize the difference between its output for real and fake data, with real data scoring higher, to provide a gradient signal to the generator. Formally, the discriminator loss to be minimized is:

$$\begin{aligned} \mathcal{L}_{\mathcal{D}}(z, \mathcal{A}_{evo}, \mathcal{T}; \phi) = -\mathcal{D}_{\phi}(\mathcal{A}_{evo}, \mathcal{T}) + \mathcal{D}_{\phi}(\mathcal{G}_{\theta}(z, \mathcal{T}), \mathcal{T}) \\ +\alpha(\|\nabla_{\tilde{A}} D_{\phi}(x, \mathcal{T})\| - 1)^2 \end{aligned} \tag{4}$$

where $\mathcal{A}_{evo}$ is the training data produced by the evolutionary algorithm. The additional gradient penalty term is introduced in the improved WGAN [13] to stabilize training. We use $\alpha = 10$ and $\tilde{A} = \epsilon \mathcal{A}_{evo} + (1-\epsilon)\mathcal{G}_{\theta}(z, \mathcal{T})$ with $\epsilon \sim \mathcal{U}(0, 1)$ as done in [13].

Evolution-guided Data Creation Our method is fundamentally different from a canonical GAN since we do not have a dataset collected *a priori*. Instead, we generate training data online through a novel evolution-guided process, inspired by evolutionary algorithms, which iteratively pushes the generator towards generating designs with higher quality.

During each training iteration, we pass a list of tasks $\mathcal{T}$ and a batch of randomly sampled latent vectors z through the generator to obtain a batch of allocation plans. These allocation plans are then treated as if they are the population

of an multi-objective EA, and iterate through n evolution steps (e.g. mutations, cross-over, evaluation, and elite selection) to create an evolved population. The evolution steps improve the Pareto optimality of the allocation plans, which are passed into the discriminator in place of what would be considered the "real" data in a conventional GAN. By training the GAN with the evolved samples, we effectively guide the generator to model a task-conditioned distribution that is iteratively shifted towards higher-quality regions in the solution space.

The generator and discriminator are both conditioned on the task description. Without this task-conditioning, the training procedure would be similar to a standard EA, wherein a population of allocation plans are evolved for a single target region. By conditioning the generator and the discriminator on the tasks, and randomly sampling tasks at each iteration, the GAN learns to interpolate between different tasks, which is the key to how our approach is able to generalize to unseen tasks during deployment.

Any variety of multi-objective population-based optimization algorithm could be used inside the data creation process. However, the specific algorithm chosen will affect the convergence behavior of the generator. In this work, we used NSGA-ii [10], a well-known fast sorting and elite multi-objective evolutionary algorithm [42].

4 Local Trajectory Planning and Coverage Calculation

Given the decomposed sub-regions and the allocated teams, each robot team is deployed to cover the corresponding sub-region distributively. Specifically, for each given robot team, a local trajectory planner is called to optimize the joint trajectory of all robots within it. The same planner is called both during the training of the generator for calculation of coverage cost, and during execution for deriving the robots' trajectories.

The coverage performance for each robot team a_i is measured via the path ergodicity [27]. The ergodic metric has been used previously as the objective function for many information-gathering and search tasks [29, 28, 5, 1], and its non-linearity with respect to the number of and type of robots covering an area poses an interesting challenge for the task allocation problem.

The ergodic metric $\mathcal{E}(\mathbf{x})$ is a function of the time averaged spatial statistics C of the robots' trajectories $\mathbf{x}$:

$$C(x) = \frac{1}{\sum_{j=1}^{|a|} \sum_t w_j} \sum_{j=1}^{|a|} \sum_t w_j \delta(\mathbf{x}_{j,t} - x), \tag{5}$$

where $x \in \mathbb{R}^D$ is a point in the search space and $\mathbf{x}_{j,t} \in \mathbb{R}^D$ is the location of robot j at time t. w_j is a weight assigned to each species of robot which specifies how useful its measurements will be. We note that this definition is slightly different from [27], where all the weights are assumed to be 1 (i.e., that all robots take

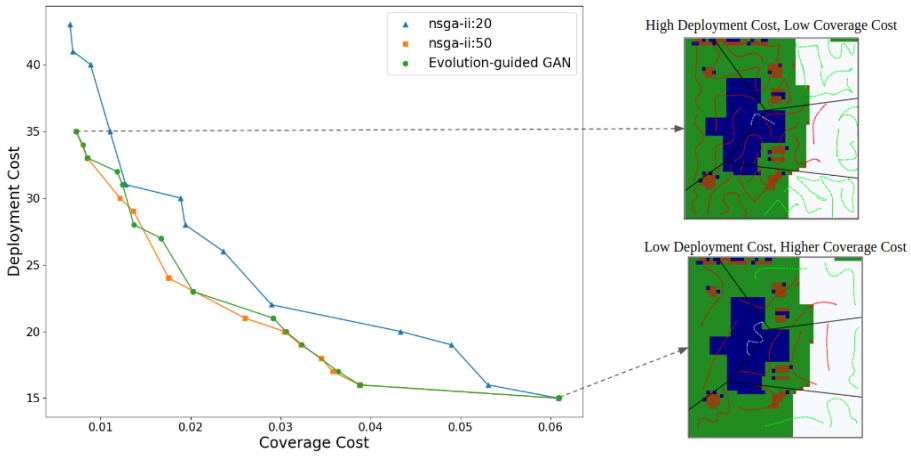

Fig. 4: Our generator outputs a set of allocation plans that show the trade-off between coverage cost and deployment cost, i.e., a Pareto front of allocations. This plot shows the solution sets obtained by NSGA-ii [10] with population size 20, NSGA-ii with population size 50, and our generator, all evaluated on the same test terrain. Here, our generator performs similarly to NSGA-ii of population size 50 and outperforms NSGA-ii of population size 20, with an average runtime that is significantly shorter. On the right, we also include two different allocation plans from the generated solution set, which illustrate how the allocation plan varies with the deployment cost.

equally useful observations). The ergodic metric is then defined as

$$\mathcal{E}(\mathbf{x}) = \sum_k \lambda_k |\mathcal{F}(C) - \mathcal{F}(f_i)|^2, \tag{6}$$

where $\mathcal{F}$ is the Fourier transform and λ_k are weights for each of the spectral components (see [27] for details).

The goal of the local trajectory planner is to minimize the ergodic metric. Intuitively, and disregarding the effect of the species weights w_j, the ergodic metric is minimized when the number of observations taken in any area of the full region is proportional to the amount of information in that area. In this work, we assume that the information is uniformly distributed over the entire region, so, ideally, observations should be equally spaced. We used a model predictive control policy similar to [28] to optimize the trajectories of the robots relative to the ergodic metric.

5 Experiments

We test our algorithm by simulating multi-robot coverage on the DroneDeploy real-world satellite image dataset [32]. The dataset contains 6888 satelite image

chips of 300x300 pixels, where each pixel has a corresponding terrain label. In our experiment, each terrain label belongs to one of the following four categories: [Water, Mountain, Plain, City].

The motion capability of a robot is quantified by its maximum velocity in each terrain. Each robot takes 20 observations during an episode. These observations are given a "coverage weight" based on the species, with a higher coverage weight indicating better sensing capabilities. We set our robot bank to contain three different species of robots: aerial robots, ground robots and aquatic robots. We hand-selected the max velocities and coverage weights of each species, as summarized in Table 1. We conducted all training and testing on a desktop computer with Ubuntu 18.04, Intel i7 eight-core processor at 1.9 GHz, and an NVIDIA GTX 1070 graphics card. Our generative model for task allocation is trained for 20 hours on i.i.d. satellite images sampled from the training dataset before testing. Both the generator and the discriminator are optimized using Adam optimizer [19] with a learning rate of $1e^{-5}$.

Table 1: In our experiment, each species of robot has different motion and sensing capabilities. The maximum velocity of a robot is terrain-dependent while the coverage weight is constant. This table shows the capabilities of each species of robot.

	Max velocity for different terrains (m/s)				
Species	Water	Mountain	Plain	City	Coverage Weight
Ground robot	0	20	40	50	20
Aerial robot	50	50	50	30	10
Aquatic robot	50	0	0	30	20

5.1 Coverage Comparison

We first focus on examining how decomposition and allocation methods affect coverage performance. Specifically, we examine the performance of three variants:

- *No Decomposition.* Robots are randomly distributed across the entire search and directly planned their trajectories as described in Sec. 4.
- *Decomposition + Random Allocation.* A decomposition step is first carried out and then the robots are randomly grouped into teams and allocated to each sub-region, as a baseline.
- *Decomposition + Generative Allocation (ours).* The robots are grouped into teams based on the output of our trained generative model.

We present coverage performance under different deployment cost limits. In this work, we assume for simplicity that each robot incurs a deployment cost of 1. Therefore, the total deployment cost corresponds to the number of robots deployed. A higher deployment cost corresponds to more deployable robots, which

naturally results in higher coverage performance. Notice that with generative allocation, the allocation plans under different deployment cost limits are generated simultaneously as solutions on the same Pareto front, i.e., different parts of the latent space input to the generator map to different Pareto-optimal solutions for the same task. Visualizations of selected results can be seen in Fig. 3. Quantitative results of the experiments are collected through evaluation on 100 test terrains, and are presented in Table 2.

We observe from the visualization that with *No Decomposition*, some of the robots would get trapped in a undesirable local optima due to unfavorable initialization; while with *Decomposition + Random Allocation*, there tends to be an uneven distribution of robots that leaves certain sub-regions relatively unexplored. As a result, both of them leave a considerable amount of area uncovered. Our framework outperforms these other approaches in this setting, demonstrating the effectiveness of both the decomposition and the generative allocation.

Table 2: We studied the effect of decomposition and allocation on the coverage performance under different deployment cost budgets. ↓ indicates that a lower value is better, and ↑ that higher is better. RanA stands for Random Allocation, GenA stands for Generative Allocation. Each value is averaged over 100 test terrains. Our framework outperforms the other tested methods under all three deployment cost limits, demonstrating the effectiveness of both the decomposition and the generative allocation.

	Coverage Cost ($\times 10^{-2}$) ↓		
Methods	Deploy Cost ≤ 15	Deploy Cost ≤ 30	Deploy Cost ≤ 45
No Decomposition	0.365 ± 0.076	0.153 ± 0.035	0.056 ± 0.037
Dec + RanA	0.732 ± 0.231	0.401 ± 0.144	0.062 ± 0.015
Dec + GenA (ours)	$\mathbf{0.049} \pm 0.006$	$\mathbf{0.018} \pm 0.004$	$\mathbf{0.007} \pm 0.001$

5.2 Pareto front Comparison

Traditional multi-objective optimization approaches such as multi-objective genetic algorithms do not fit our problem statement needs due to their high computational cost. Nevertheless, they can be used as a benchmark for evaluating the quality of the Pareto front solutions generated by our generative model. The goal of this experiment is to determine whether generative allocation can produce allocations that match or surpass traditional centralized task allocation approaches, while reducing computational expense at run-time. We therefore provide additional comparison between the generative allocation and NSGA-ii [10], a well-known fast sorting and elite multi-objective evolutionary algorithm [42]. Our implementation of NSGA-ii used single-point arithmetic crossover with a crossover probability of 0.5 and uniform mutation with a mutation rate of 0.1.

The maximum number of iterations is set to be 50, where NSGA-ii:20 has a population size of 20, and NSGA-ii:50 has a population size of 50.

We numerically evaluate the Pareto fronts discovered by the tested algorithms on *hypervolume* [43], a common set-quality indicators for stochastic multi-objective optimizers [22]. The hypervolume set-quality indicator maps a point set in $\mathbb{R}^d$ to the measure of the region dominated by that set and bounded above by a given reference point, also in $\mathbb{R}^d$, where d is the number of objectives. As is visually evident, a larger hypervolume is better. In our experiment, we used $(0.25, 60)$ as the reference point. we additionally report *average runtime*, which measures the wall time of each algorithm during execution in minutes, to demonstrate the computational efficiency of our algorithm.

Quantitative results of the experiments are presented in Table 3, where each entry is averaged over five independent runs. A visualization of the Pareto fronts discovered by the tested algorithms for a specific terrain and the corresponding solutions can be seen in Fig. 4. Compared to NSGA-ii, our trained generator is able to generate comparably effective allocations more efficiently, since its execution only consist of a neural network forward pass. It is important to note that our execution efficiency comes at the cost of the training time, which NSGA-ii do not require. However, the training takes place before the generator is deployed, and need only be trained once before being used for many tasks. Also note that NSGA-ii solution quality tends to improve solutions given larger population sizes and more iterations, but its time cost scales proportionally.

Table 3: Performance comparison between tested algorithms. ↓ indicates that lower measures are better, ↑ that higher are better. NSGA-ii:20 and NSGA-ii:50 [10] have a population size of 20 and 50 respectively. Compared to evolutionary algorithms, our generative allocation (GenA) method produces allocations of similar quality with significantly faster runtime, making it suitable for time-critical deployment.

Methods	Hypervolume ↑	Avg. runtime (min.) ↓
NSGA-ii:20	9.4 ± 1.0	43.1 ± 0.6
NSGA-ii:50	$\mathbf{11.9} \pm 1.5$	107.6 ± 0.9
GenA (ours)	11.8 ± 0.8	**0.02**

6 Conclusion

In this work, we proposed a framework that uses domain decomposition and generative allocation to efficiently solve large-scale heterogeneous multi-robot coverage problems. Our framework decomposes a large coverage problem into smaller sub-problems, which allows us to coordinate the teams distributively

while still considering the global distribution of terrain and information. To determine the team formation in a computationally efficient manner, we introduce a novel generative allocation algorithm which learns to generate suitable allocation plans for heterogeneous robots by taking advantage of correlations between tasks. One limitation of our generative allocation algorithm, shared by other machine learning methods, is the assumption that new tasks will be from the same distribution as those seen during training, such that the outputs from out-of-distribution tasks may be poor. Another limitation of our framework is the need for a labelled terrain map, which may not always be available in real-world applications.

There are several interesting future directions for this work. Firstly, our work only uses a single decomposition and allocation step. Exploration of the possibility of applying the decomposition and allocation in a recursive manner may be beneficial to extend this framework to larger-scale applications. Secondly, we only experiment with two objectives in our generative allocation. It would be interesting to see if this algorithm will scale to higher dimensional objectives spaces. Lastly, we do not explore how different decomposition objectives would affect the coverage performance. It is possible that other decomposition objectives may improve the full coverage achieved by the coalitions, and therefore further analysis of the choice of decomposition would be valuable.

References

1. Abraham, I., Mavrommati, A., Murphey, T.: Data-driven measurement models for active localization in sparse environments. In: Proceedings of Robotics: Science and Systems. Pittsburgh, Pennsylvania (June 2018). https://doi.org/10.15607/RSS.2018.XIV.045
2. Agarap, A.F.: Deep learning using rectified linear units (relu). arXiv preprint arXiv:1803.08375 (2018)
3. Agarwal, M., Agrawal, N., Sharma, S., Vig, L., Kumar, N.: Parallel multi-objective multi-robot coalition formation. Expert Systems with Applications **42**(21), 7797–7811 (2015)
4. Arjovsky, M., Chintala, S., Bottou, L.: Wasserstein generative adversarial networks. In: Int. conference on machine learning. pp. 214–223. PMLR (2017)
5. Ayvali, E., Ansari, A., Wang, L., Simaan, N., Choset, H.: Utility-guided palpation for locating tissue abnormalities. IEEE Robotics and Automation Letters **2**(2), 864–871 (2017)
6. Ayvali, E., Salman, H., Choset, H.: Ergodic coverage in constrained environments using stochastic trajectory optimization. In: 2017 IEEE/RSJ Int. Conference on Intelligent Robots and Systems (IROS). pp. 5204–5210. IEEE (2017)
7. Badreldin, M., Hussein, A., Khamis, A.: A comparative study between optimization and market-based approaches to multi-robot task allocation. Advances in Artificial Intelligence (16877470) (2013)
8. Casas, N.: Genetic algorithms for multimodal optimization: a review. arXiv preprint arXiv:1508.05342 (2015)
9. Črepinšek, M., Liu, S.H., Mernik, M.: Exploration and exploitation in evolutionary algorithms: A survey. ACM computing surveys (CSUR) **45**(3), 1–33 (2013)

10. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: Nsga-ii. IEEE transactions on evolutionary computation **6**(2), 182–197 (2002)
11. Dorigo, M., Birattari, M., Stutzle, T.: Ant colony optimization. IEEE computational intelligence magazine **1**(4), 28–39 (2006)
12. Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A.C., Bengio, Y.: Generative adversarial nets. In: Neural Information Processing Systems (2014)
13. Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., Courville, A.C.: Improved training of wasserstein gans. Advances in neural information processing systems **30** (2017)
14. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: International conference on machine learning. pp. 448–456. PMLR (2015)
15. Kantaros, Y., Zavlanos, M.M.: Distributed communication-aware coverage control by mobile sensor networks. Automatica **63**, 209–220 (2016)
16. Katoch, S., Chauhan, S.S., Kumar, V.: A review on genetic algorithm: past, present, and future. Multimedia Tools and Applications pp. 1–36 (2020)
17. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of ICNN'95-international conference on neural networks. vol. 4, pp. 1942–1948. IEEE (1995)
18. Khamis, A., Hussein, A., Elmogy, A.: Multi-robot task allocation: A review of the state-of-the-art. Cooperative Robots and Sensor Networks 2015 pp. 31–51 (2015)
19. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
20. Korsah, G.A., Stentz, A., Dias, M.B.: A comprehensive taxonomy for multi-robot task allocation. The Int. Journal of Robotics Research **32**(12), 1495–1512 (2013)
21. Levina, E., Bickel, P.: The earth mover's distance is the mallows distance: Some insights from statistics. In: Proceedings Eighth IEEE Int. Conference on Computer Vision. ICCV 2001. vol. 2, pp. 251–256. IEEE (2001)
22. Li, M., Chen, T., Yao, X.: How to evaluate solutions in pareto-based search-based software engineering? a critical review and methodological guidance. IEEE Transactions on Software Engineering p. 1–1 (2020). https://doi.org/10.1109/tse.2020.3036108, http://dx.doi.org/10.1109/TSE.2020.3036108
23. Liu, C., Kroll, A.: A centralized multi-robot task allocation for industrial plant inspection by using a* and genetic algorithms. In: Int. Conference on Artificial Intelligence and Soft Computing. pp. 466–474. Springer (2012)
24. Liu, H.Y., Chen, J.F.: Multi-robot cooperation coalition formation based on genetic algorithm. In: 2006 Int. conference on machine learning and cybernetics. pp. 85–88. IEEE (2006)
25. Liu, Y., Wang, W., Lévy, B., Sun, F., Yan, D.M., Lu, L., Yang, C.: On centroidal voronoi tessellation—energy smoothness and fast computation. ACM Transactions on Graphics (ToG) **28**(4), 1–17 (2009)
26. López-González, A., Campaña, J.M., Martínez, E.H., Contro, P.P.: Multi robot distance based formation using parallel genetic algorithm. Applied Soft Computing **86**, 105929 (2020)
27. Mathew, G., Mezić, I.: Metrics for ergodicity and design of ergodic dynamics for multi-agent systems. Physica D: Nonlinear Phenomena **240**(4-5), 432–442 (2011)

28. Mavrommati, A., Tzorakoleftherakis, E., Abraham, I., Murphey, T.D.: Real-time area coverage and target localization using receding-horizon ergodic exploration. IEEE Transactions on Robotics **34**(1), 62–80 (2017)
29. Miller, L.M., Silverman, Y., MacIver, M.A., Murphey, T.D.: Ergodic exploration of distributed information. IEEE Transactions on Robotics **32**(1), 36–52 (2015)
30. Mirza, M., Osindero, S.: Conditional generative adversarial nets. arXiv preprint arXiv:1411.1784 (2014)
31. Mouradian, C., Sahoo, J., Glitho, R.H., Morrow, M.J., Polakos, P.A.: A coalition formation algorithm for multi-robot task allocation in large-scale natural disasters. In: 2017 13th Int. Wireless Communications and Mobile Computing Conference (IWCMC). pp. 1909–1914. IEEE (2017)
32. Pilkington, N., Svetlichnaya, S., Holmes, T.: Github - dronedeploy/ddml-segmentation-benchmark: Dronedeploy machine learning segmentation benchmark (2019)
33. Price, K.V.: Differential evolution. In: Handbook of optimization, pp. 187–214. Springer (2013)
34. Prorok, A., Hsieh, M.A., Kumar, V.: Fast redistribution of a swarm of heterogeneous robots. In: Proceedings of the 9th EAI Int. Conference on Bio-Inspired Information and Communications Technologies (Formerly BIONETICS). p. 249–255. BICT'15 (2016)
35. Rauniyar, A., Muhuri, P.K.: Multi-robot coalition formation and task allocation using immigrant based adaptive genetic algorithms. In: Computational Intelligence in Emerging Technologies for Engineering Applications, pp. 205–225. Springer (2020)
36. Shehory, O., Kraus, S.: Methods for task allocation via agent coalition formation. Artificial intelligence **101**(1-2), 165–200 (1998)
37. Van Veldhuizen, D.A., Lamont, G.B.: Multiobjective evolutionary algorithms: Analyzing the state-of-the-art. Evolutionary computation **8**(2), 125–147 (2000)
38. Van Veldhuizen, D.A., Lamont, G.B., et al.: Evolutionary computation and convergence to a pareto front. In: Late breaking papers at the genetic programming 1998 conference. pp. 221–228. Citeseer (1998)
39. Wang, J., Gu, Y., Li, X.: Multi-robot task allocation based on ant colony algorithm. Journal of Computers **7**(9), 2160–2167 (2012)
40. Wei, C., Ji, Z., Cai, B.: Particle swarm optimization for cooperative multi-robot task allocation: a multi-objective approach. IEEE Robotics and Automation Letters **5**(2), 2530–2537 (2020)
41. Xu, B., Yang, Z., Ge, Y., Peng, Z.: Coalition formation in multi-agent systems based on improved particle swarm optimization algorithm. Int. Journal of Hybrid Information Technology **8**(3), 1–8 (2015)
42. Yusoff, Y., Ngadiman, M.S., Zain, A.M.: Overview of nsga-ii for optimizing machining process parameters. Procedia Engineering **15**, 3978–3983 (2011)
43. Zitzler, E., Thiele, L.: Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. IEEE transactions on Evolutionary Computation **3**(4), 257–271 (1999)