Decentralized Multi-Policy Decision Making for Communication Constrained Multi-Robot Coordination

Maximilian Krogius Acshi Haggenmiller Edwin Olson

***Abstract*— Multi-Policy Decision Making (MPDM) has been shown to be an effective method for single-agent navigation tasks. In this paper, we extend MDPM to long-horizon multi-robot planning tasks with uncertain communication. We constrain each team member to choose the best of several simple policies through forward simulation in a decentralized fashion. We demonstrate this algorithm on both a coverage task as well as a challenging adversarial target search scenario, with uncertain communication for both. We also show that our algorithm can generalize to scenarios it was not tuned for.**

I. Introduction

This paper presents a planning algorithm for multi-robot teams with unreliable communications. When planning over long time horizons is necessary, the computational complexity of many methods (e.g. POMDP solvers) becomes intractable. On the other hand, the simplifying assumption of constant reliable communication is often violated by real-world scenarios outside laboratory settings. We extend Multi-Policy Decision Making (MPDM) [1] to the decentralized case in order to solve this problem. This allows us to select each robot's policy based on deep forward simulations of a set of simple policies.

Solving multi-robot planning problems under communications uncertainty is important for applications such as search and rescue. A team of robots can search an area faster than a single robot and they can perform search in areas which are too dangerous for humans to explore. Since search and rescue is often necessary in areas where there is limited time to set up or repair communications infrastructure, it is important to solve the problem without the assumption of communications being available at all times.

Our method, Decentralized Multi-Policy Decision Making (D-MPDM), is an extension to the single-robot planning of MPDM in which the robot's policy is selected from a set of simple policies through a short-term forward simulation. In D-MPDM each robot separately performs selection of a simple policy through a deep forward simulation which includes the policy choices of the other robots and their communications. No communication or synchronization is involved in the planning process so that the system will continue operating, albeit with reduced coordination, when communication is unavailable. When communication is available, the robots broadcast their current state, implicitly coordinating their policy choices.

The authors are with the Computer Science and Engineering Department, University of Michigan, Ann Arbor, MI 48104, USA. {mkrogius,acshikh,ebolson}@umich.edu

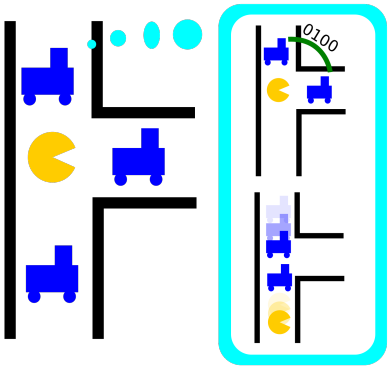

Fig. 1. The robots plan in a decentralized fashion using forward simulations that include future communication between the robots. This is possible because we restrict the robots to choose from a set of simple policies that control their behavior and communications.

We demonstrate the performance of D-MPDM across two different tasks: coverage and adversarial search. The coverage task is a proxy for search and rescue while the adversarial search poses an interesting planning challenge since it requires tighter coordination between the agents. We test D-MPDM at different levels of communications availability to ensure that we have achieved robustness.

The use of handcrafted policies in D-MPDM contrasts with the usual approach of handcrafted value functions in many other optimization algorithms. This allows D-MPDM to search very deep at the cost of not being able to optimize precisely over a short term sequence of actions. In our evaluation we compare against a baseline using a handcrafted heuristic and we show that D-MPDM is able to outperform it in most scenarios.

The contributions of this work include:

- The D-MPDM algorithm, which coordinates the actions of the team of robots in a decentralized manner, with graceful performance degradation when communications drop out.
- Evaluation of D-MPDM on two different tasks and four environments.
- In order to avoid bias while evaluating the performance of our algorithm, we split the environments into training and testing sets, which to the best of our knowledge is novel in this domain.

II. Related Work

The first task we will evaluate our method with is adversarial search. For a survey of adversarial search algorithms

see the work of Chung [2], which overviews both the approaches as well as their field testing. The authors found that robot reliability and communication failures were obstacles to larger scale testing. Kolling [3] presents a method for arranging teams of robots in line formations in order to search an area represented as an occupancy grid. This method provides guaranteed capture on occupancy grids, but does not address the problem of communication. Hollinger [4] presents an online algorithm for generating search plans which guarantee capture of the target, assuming sufficient communication between agents. Our method does not offer the same guarantee, but also does not require a minimum level of communication or a minimum number of searchers. To overcome these limitations and permit larger-scale field testing, we have designed D-MPDM to be decentralized and robust to communications dropout.

Our second task is the coverage problem. A recent survey from Amigoni [5] categorizes algorithms by the degree of communications needed. Our algorithm falls into the least-restrictive category, since it makes no assumption that any communication at all will necessarily be available. Zlot [6] used a market economy to control the coverage task, with tasks being auctioned to the agent most willing to perform them. This approach only indirectly optimizes the objective function, whereas D-MPDM directly optimizes it. Baxter [7] uses a potential field based method to perform the task in a purely reactive manner. Similarly in D-MPDM, our agents switch between a set of reactive policies. Brass [8] has the robots explore the graph in a depth-first fashion, communicating which nodes have been explored by marking the environment. This appears to still constitute a minimum level of required communication that may not always be available depending on the environment. Matignon [9] formalizes the problem as a Dec-POMDP and solves it by splitting the problem into a solution for when the agents are close enough to interact and a solution for when the robots are far enough apart that they do not need to coordinate. Compared to D-MPDM this approach is small scale and focused mostly on low level motion planning. Salman [10] uses the principle of ergodicity to derive control laws for a multi-agent team for the related problem of ongoing coverage where the agents must return to already visited states periodically. Their method contrasts with D-MPDM in that the simple policies of D-MPDM handle the low-level control laws so that the planning effort is focused on the higher-level objective.

Other researchers have also demonstrated methods that can handle a variety of tasks. An early approach [11] formalized the problem as a Dec-POMDP. This allows the algorithm to optimize the semantics of the communication channel, however the computational complexity of this approach limits it to small environments. Best [12] uses a probabilistic form of monte-carlo tree search, which they demonstrate on the problems of team orienteering and active object recognition with some limits on communication. This approach is also computationally heavy, although it does not have the same problems with scaling to larger environments as the DEC-POMDP approach. In contrast, our method is both scalable and has the additional advantage of being able to reason about the effects of future communication.

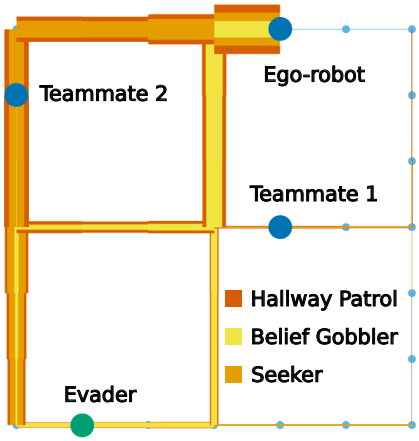

Fig. 2. Visualization of forward monte-carlo rollouts of one robot. The trajectories of each policy are shown in different colors while the width corresponds to the number of times that edge was traversed by that policy in the simulation. Each robot performs these rollouts as the ego-robot to make its own policy decision. The depth of the rollouts here is limited to 10 in order to produce a simpler visualization.

Our approach is based on MPDM, which was introduced by Cunningham [1] for single-agent autonomous driving. Mehta [13] showed that MPDM could also be applied to a single-agent social navigation task. We build on these works to extend MPDM to the multi-agent domain.

III. Methods

A. Decentralized MPDM

We consider the problem of planning for a team of robots R in an environment represented by a graph where nodes correspond to physical locations and edges to the paths between them. Each robot is located at some node and can then move to an adjacent node each timestep. The robots communicate over unreliable links such that each communication link has some probability p_{comms}, known to the robots ahead of time, of functioning at any timestep. The team of robots must act together in order to minimize the time to achieve some task-specific goal, T_{goal}.

In D-MPDM we add the restriction that each agent must follow one of a set of handcrafted simple policies, $P \equiv \{p_1, p_2, ..., p_n\}$, as in MPDM [1]. This reduces the planning problem from finding the optimal of all possible policies, to picking the best policy from a fixed set, an approach that sacrifices theoretical optimality for tractability. We want each robot r to choose the best policy p^r_{best}, such that:

$$p^r_{\text{best}} = \arg\min_{p^r \in P} \mathop{\mathbb{E}}_{p^r}[T_{\text{goal}} | \vec{b^r}, \{\hat{\text{state}}(p^s) | s \in R, s \neq r\}] \quad (1)$$

where the expectation is over trajectories resulting from this robot's choice of the policy p^r given this robot's belief about the world, $\vec{b^r}$ as well as its belief about the states

of its teammates, $\{\hat{\text{state}}(p^s)|s \in R, s \neq r\}$. In practice we approximate this expectation and evaluate it through Monte Carlo sampling as described below.

To avoid introducing a global consensus problem where the robots all choose and agree on their policies in lock-step, we make several simplifications. First, the currently planning robot, or *ego-agent*, assumes that all robots which have recently communicated with it will choose the same policy as itself. The other robots should hold a similar belief about the world state so this should result in implicit coordination when there is sufficient communication between the robots. Second, the ego-agent assumes that robots which have not recently communicated with it will choose a policy uniformly at random. With these relaxations, we obtain a system that continues operating even if communication fails entirely.

Given the above assumptions, each robot evaluates the expectation of a policy p_i by assigning that policy to itself as well as robots that have recently communicated with it and by sampling a policy choice for the non-communicating agents. The ego-agent then runs a forward simulation and calculates the number of timesteps until completion of the goal. Since the simulation involves many random elements we repeat this process of sampling policy choices and running forward simulations multiple times for each potential ego-robot policy, p_i. D-MPDM then chooses and executes the policy with minimal T_{goal} averaged over all samples. Figure 2 illustrates the trajectories produced by each policy in this forward simulation.

Since the robots have limited sensing and communication, each robot maintains a belief about the state of the world, which we sample from in order to run the forward simulations described above. We represent the robot's knowledge as a belief over the possible world states. Since this domain is discrete, it is possible to compute the belief at the next timestep from the belief at the current timestep, given a model of the environment. We run the environment model each timestep to know how the belief should propagate. This belief distribution represents some task specific quantity. For the coverage task, the belief distribution represents the probability that each node has not been visited, which is one until it has been visited by an agent, and zero afterwards. For the adversarial search task, the belief distribution represents the probability that the adversary is currently at each node, so for this task the belief distribution has dynamics defined by the model of the adversary.

We do not use consensus or synchronization in the communication of these task beliefs so that D-MPDM will be tolerant to poor and intermittent communications. Every timestep each robot will attempt to broadcast their current belief. When the ego-robot receives a communication, it will fuse the other robot's task belief with its own belief. For application to the coverage and adversarial search tasks, we follow Hollinger [14] in fusing the belief of the agents about the environment using the minimum rule, i.e. the belief value b_j associated with each node j is set to the minimum b_j value of the ego-robot and each other communicating robot. Some simple policies also have additional state, such as using the robot's prior position to prevent backtracking. The robots will also communicate these policy specific states so that the robots can more accurately simulate each other.

Algorithm 1: D-MPDM Main loop.

```
Function EstimateValue (policy)
    avg_time = 0
    for i = 1..width do
        /* ForwardSimulate returns the amount
           of time it took to complete the task
           given the sampled initial
           conditions.                          */
        avg_time = avg_time +
            ForwardSimulate(policy,
            SampleInitialConditions())
    end
    avg_time = avg_time / width
    return -avg_time

while task is incomplete do
    best_policy = max(policies,
        key(policy)=EstimateValue(policy))
    execute policy for one timestep
    broadcast current state
    receive and fuse states from teammates
end
```

When teammates fail to communicate, we must also maintain some distribution over their possible locations, policy-specific states, and task-specific belief about the world. We model these non-communicating teammates with a particle filter. This filter is initialized at the last known location of the teammate, with a uniform distribution over simple policies. This consists of 10 particles for the Hallway Patrol policy, and one particle for each of the other policies. At each timestep, each of the particles in the filter is updated according to its associated simple policy, assuming the robot has not communicated with any teammates. When a robot teammate does manage to communicate, the particle filter used by the ego-robot to represent that teammate is no longer needed, but may be reinitialized later. This allows us to consider a limited set of realistic positions and beliefs for each non-communicating agent.

In each forward simulation we simulate the ego-agent, teammates (with sampled positions/policies if they are not communicating), evader (for the adversarial search task), as well as communications availability. We run 50 simulations for each choice of policy, sampling over teammate state and our belief about the world. The forward simulation is run until either the goal is achieved or until the depth exceeds the maximum depth. For any simulation that exceeds this maximum depth we assume that the goal is accomplished on the next timestep.

B. Simple Policies

An MPDM system relies on a good selection of simple policies. Here we present the three simple policies used in

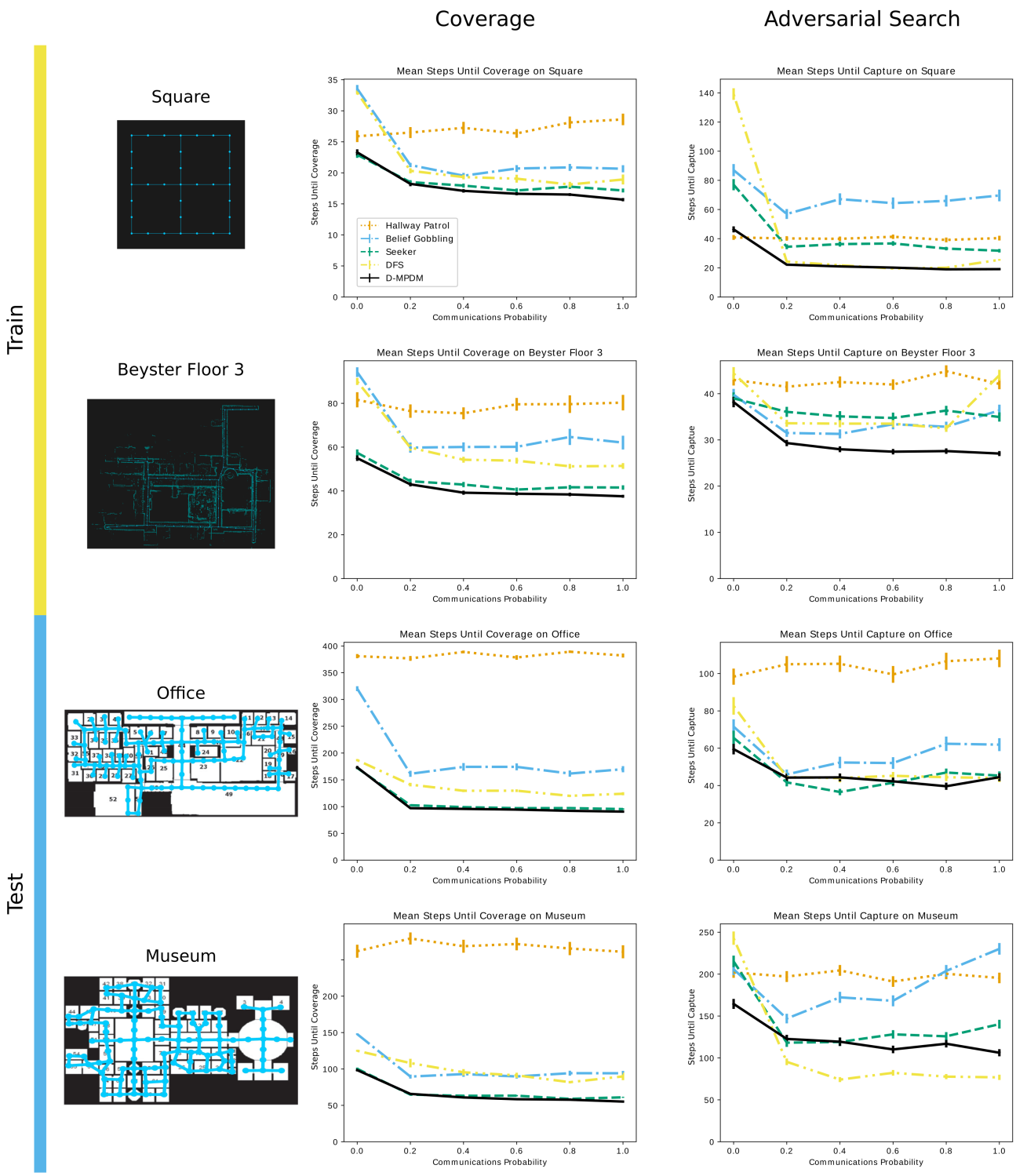

Fig. 3. Column 1: Visualizations of the different maps used for evaluation. The first two maps, Square and Beyster, were used for development and testing of the algorithm and are thus designated "Train". We used the performance on these environments as feedback while we designed the policies and choose parameters. The next two maps, Office and Museum, are designated "Test" since no changes were made to the algorithm, policies, or parameters based on the results on these maps, with the exception of reducing the depth of the baseline's search from 50 to 20 in order for it to finish in a feasible time frame. The maximum depth parameter for D-MPDM is set to 150 and the number of simulations per policy (width) is set to 50. Column 2: The performance of the Hallway Patrol, Belief Gobbling, Seeker, DFS, and D-MPDM (Ours) algorithms on the four maps for the task of coverage. Column 3: The performance of the Hallway Patrol, Belief Gobbling, Seeker, DFS, and D-MPDM (Ours) algorithms on the four maps for the task of adversarial search. D-MPDM outperforms the baseline on all maps except for the Museum map adversarial search.

this system for both tasks.

1) Hallway Patrol: This policy proceeds down corridors and makes random choices at intersections without backtracking (unless it reaches a dead end). This policy should result in reasonably efficient exploration even in poor communication conditions.

2) Belief Gobbling: This policy chooses the action which takes it to the neighboring node with the maximum amount of belief. This can be thought of as essentially a greedy policy. This policy will communicate and fuse its belief state with its teammates, when communications are available. This allows for some degree of coordination since if one robot proceeds down a dead end, a nearby agent using belief gobbling will not follow it down that corridor since the belief behind the first agent will be small.

3) Seeker: This policy computes a target node and then plans the shortest path to that node. The target is selected by the equation:

$$\text{target} = \arg\max_j \frac{b_j}{d(i,j)} \tag{2}$$

where i is the current location/graph node, j ranges over all nodes, b_j is the ego-robot's task belief either that node j is not yet covered or that it contains the evader, and $d(i,j)$ is the path length through the graph between i and j. This rule for target selection is similar to the rule for optimal search from Matula [15], although the proofs pertaining to optimality do not extend from that work to this more complex case. Because it plans the shortest path to the target, this policy can still efficiently explore when nearby nodes have insignificant belief compared to those farther away. This policy also attempts to reduce unproductive synchronization by choosing a random action when it knows it is at the same location as another agent.

C. Task Types

In this work we would like to demonstrate that our multi-agent planning algorithm is capable of generalizing across tasks, environments, and levels of communication availability. To this end we have two tasks to test in the simulation environment.

The first is a coverage task. The goal of this task is to visit every state in as few timesteps as possible. In order to be efficient at this task, the agents must spread out as well as avoid leaving behind isolated unvisited nodes. The robots are unable to sense directly whether a state has previously been visited by a teammate and so must rely purely on communication of the belief in order to compute this information.

The second task is an adversarial search. The goal is to capture an adversary who is mildly adversarial, moving away from the robots when they get within two steps of the adversary. The adversary has the advantage that it knows where all the robots are. The robots have the disadvantage of not being able to sense the adversary at a distance. The only feedback they receive is whether or not they have captured the adversary yet. To succeed at this task, the robots must execute pincer movements where the adversary ends up trapped between multiple advancing robots. The robots should also try to make their belief about the location of the adversary more informative, by planning actions to group up large amounts of belief.

D. Baseline

Since we could not find another work proposing a general-purpose multi robot algorithm which allows for imperfect communication, we define our own baseline. In our baseline, each agent performs a limited-depth tree search over the game tree to find the best plan. Agents share plans, and other agents' plans are assumed fixed when the ego-agent is performing its tree search to generate its own plan.

This form of best-response planning can lead to oscillation, because when all the robots are synchronously computing their new plans, they can all change their plans every turn between two sets of plans. This does not happen in the real world since the process of planning and communicating is asynchronous. To mitigate this failure mode we have implemented cycle detection. If an agent has received enough communications to know that it has returned to the location it was at two timesteps ago and that all other robots have returned to their locations from two timesteps ago, then it will choose a random action. In addition, if two robots are at the same location then they will both choose random actions. In this way we can mitigate the issue of synchronization, but we suspect that this does not completely remove all negative effects. Planning in the joint action space may be necessary in order to achieve truly optimal plans.

The depth of the tree search is selected such that the algorithm runs at a reasonable ($<$ 1s per timestep) speed for both graphs and task types. In practice this means a maximum depth of 50 timesteps for the training graphs, and a much lower 20 timesteps for the test graphs due to their higher average node degree. This demonstrates that an approach like D-MPDM is necessary to compute longer-horizon plans.

IV. Results

A. Simulation Results

In order to achieve statistical significance across a wide range of scenarios, we have tested the performance of D-MPDM and our baseline in simulation. We have simulated results for two task types, four graphs, and five communication levels in Figure 3.

We have labeled the Square and Beyster maps as "Train" since these maps were used for development and tuning of our algorithm through the policies and parameters we chose. We only ran D-MPDM on the "Test" maps once we had finished making changes to the algorithm. We did this in order to test the generalization performance of our system to new maps. The "Test" maps here are also qualitatively different than the "Train" maps, as the "Test" maps are both substantially larger.

In Figure 3 we can see the performance of the algorithms for the "Train" maps. Our method (D-MPDM) is the best

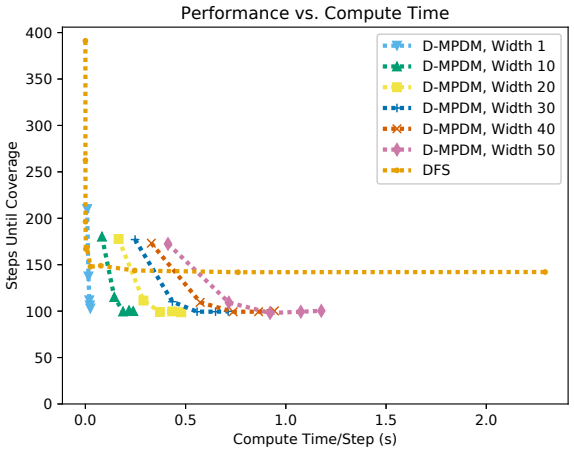

Fig. 4. Performance vs compute tradeoffs for the DFS baseline and D-MPDM. Each datapoint in a series is at a different depth. For DFS, the depth varies from 2 to 20 in steps of 2. For D-MPDM the depth varies from 30 to 150 in steps of 30. The width of D-MPDM refers to the number of forward simulations per simple policy, which should capture the variance of D-MPDM's estimates of each policy's value. This data was collected for the coverage task on the Office map at a communication level of 0.2.

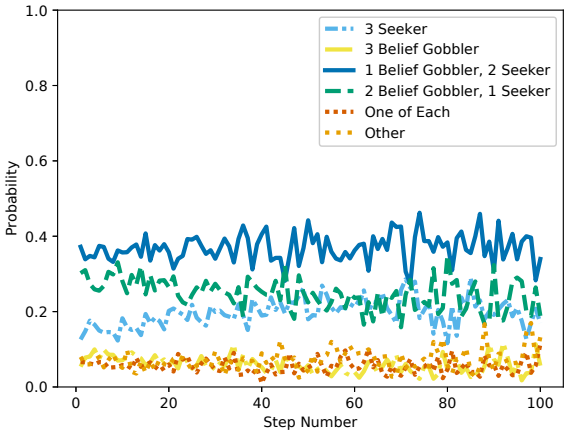

Fig. 5. Relative frequencies of different policy combinations by the three robots vs step number for 500 runs of the adversarial search task on the Office map at a communication level of 0.2. We have truncated the plot after 100 timesteps because there are few trajectories that go beyond 100 timesteps so the data becomes noisy.

performing algorithm for both maps across all communication levels and both tasks. The performance of D-MPDM matches and sometimes exceeds the performance of the simple policies that it chooses between. This is due to the replanning which allows D-MPDM to switch policies as the task proceeds. D-MPDM also exceeds the performance of the more computationally-expensive DFS baseline by a significant amount.

On the test maps, D-MPDM has the best performance on the coverage tasks. On the adversarial tasks, the performance is typically either the best or the second best method over the operating spectrum. Only the DFS-based search outperforms it on one map, and only then when communications are reasonably reliable.

In Figure 4 we sweep hyperparameters for D-MPDM and DFS. We can see that for both algorithms there is a tradeoff between compute time and performance. There is a much steeper increase in compute time for DFS vs D-MPDM as the search depth increases. This is as expected since DFS is exploring a tree whereas D-MPDM is running a fixed number of simulation runs.

In Figure 5 we can see which combinations of policies D-MPDM chooses as a function of step number. Even though the communication level here is 0.2, we can see that the three agents are all choosing the same policy (all 3 choose Seeker) a substantial amount of the time.

V. Discussion

The goal of this paper was to develop a method that generalizes well to new environments. At the same time, any method being developed needs to be debugged and have parameters tuned on a reasonable amount of data or sample environments. In most machine learning work, this is achieved through segregating the training and test datasets. To test how well our algorithm generalizes, we have followed a similar idea and used separate "Test" and "Train" maps. To the best of our knowledge this is novel for this problem space. We made no changes to our algorithm to perform well on "Test" maps, however we did reduce the search depth for the DFS baseline on the "Test" maps from 50 to 20 in order to make its compute requirements feasible. The performance of D-MPDM on the "Test" maps supports our claim that D-MPDM generalizes to new environments.

VI. Conclusion

In this paper we presented D-MPDM, a multi-agent planning method that uses forward simulation to choose a policy for each agent out of a set of simple policies. We have shown that D-MPDM outperforms the baseline across all communication levels (including zero communications), environments (except for the Museum map), and task types. We have used a set of "Train" and "Test" maps to show that D-MPDM generalizes to never-before-seen environments without needing additional policy or parameter changes.

We will make our source code available upon publication of this paper.

References

[1] A. G. Cunningham, E. Galceran, R. M. Eustice, and E. Olson, "MPDM: Multipolicy decision-making in dynamic, uncertain environments for autonomous driving," in *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2015-June, no. June. IEEE, 2015, pp. 1670–1677.

[2] T. H. Chung, G. A. Hollinger, and V. Isler, "Search and pursuit-evasion in mobile robotics A survey," pp. 299–316, 2011.

[3] A. Kolling, A. Kleiner, and S. Carpin, "Coordinated search with multiple robots arranged in line formations," *IEEE Transactions on Robotics*, vol. 34, no. 2, pp. 459–473, April 2018.

[4] G. Hollinger, A. Kehagias, and S. Singh, "GSST: Anytime guaranteed search," *Autonomous Robots*, vol. 29, no. 1, pp. 99–118, jul 2010.

[5] F. Amigoni, J. Banfi, and N. Basilico, "Multirobot exploration of communication-restricted environments: A survey," *IEEE Intelligent Systems*, vol. 32, no. 6, pp. 48–57, November 2017.

[6] R. Zlot, A. T. Stentz, M. B. Dias, and S. Thayer, "Multi-robot exploration controlled by a market economy," in *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 3, 2002, pp. 3016–3023.

[7] J. L. Baxter, E. K. Burke, J. M. Garibaldi, and M. Norman, *Multi-Robot Search and Rescue: A Potential Field Based Approach*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 9–16.

[8] P. Brass, F. Cabrera-Mora, A. Gasparri, and J. Xiao, "Multirobot tree and graph exploration," *IEEE Transactions on Robotics*, vol. 27, no. 4, pp. 707–717, aug 2011.

[9] L. Matignon, L. Jeanpierre, and A.-I. Mouaddib, "Coordinated multi-robot exploration under communication constraints using decentralized markov decision processes," in *Twenty-sixth AAAI conference on artificial intelligence*, 2012.

[10] H. Salman, E. Ayvali, and H. Choset, "Multi-agent ergodic coverage with obstacle avoidance," in *Proceedings International Conference on Automated Planning and Scheduling, ICAPS*, 2017, pp. 242–249.

[11] M. T. Spaan, G. J. Gordon, and N. Vlassis, "Decentralized planning under uncertainty for teams of communicating agents," in *Proceedings of the International Conference on Autonomous Agents*, vol. 2006, 2006, pp. 249–256.

[12] G. Best, O. M. Cliff, T. Patten, R. R. Mettu, and R. Fitch, "Dec-MCTS: Decentralized planning for multi-robot active perception," *International Journal of Robotics Research*, vol. 38, no. 2-3, pp. 316–337, 2019.

[13] D. Mehta, G. Ferrer, and E. Olson, "Autonomous navigation in dynamic social environments using multi-policy decision making," in *IEEE International Conference on Intelligent Robots and Systems*, vol. 2016-November, oct 2016, pp. 1190–1197.

[14] G. A. Hollinger, S. Yerramalli, S. Singh, U. Mitra, and G. S. Sukhatme, "Distributed data fusion for multirobot search," *IEEE Transactions on Robotics*, vol. 31, no. 1, pp. 55–66, 2015.

[15] D. Matula, "A periodic optimal search," *The American Mathematical Monthly*, vol. 71, no. 1, pp. 15–21, 1964.