Abstract: Many multi-robot planning problems are burdened by the curse of dimensionality, which compounds the difficulty of applying solutions to large-scale problem instances. The use of learning-based methods in multi-robot planning holds great promise as it enables us to offload the online computational burden of expensive, yet optimal solvers, to an offline learning procedure. Simply put, the idea is to train a policy to copy an optimal pattern generated by a small-scale system, and then transfer that policy to much larger systems, in the hope that the learned strategy scales, while maintaining near-optimal performance. Yet, a number of issues impede us from leveraging this idea to its full potential. This blue-sky paper elaborates some of the key challenges that remain.

Keywords: Multi-Robot Planning, Imitation Learning

1 Introduction

Learning-based methods have proven effective at designing robot control policies for an increasing number of tasks [1, 2]. The application of learning-based methods to multi-robot planning has attracted particular attention due to their capability of handling high-dimensional joint state-space representations, by offloading the online computational burden to an offline learning procedure [3, 4]. We argue that these developments point to a fundamental approach that combines ideas around the application of learning to optimization and produce a flexible framework that could tackle many hard but important problems in robotics, including multi-agent path planning [5], area coverage [6, 7], task allocation [8, 9, 10], formation control [11], and target-tracking [12]. In this paper, we motivate this approach and discuss the crucial challenges and research questions.

Figure 1: Applications of learning to optimization problems. (A) embodies techniques for learning optimization heuristics; (B) embodies techniques for learning to solve POMDPs; (C) is the emerging topic discussed here, embodying techniques for learning to coordinate large systems in real-world applications.

The ideas here sit within a larger landscape of the application of learning to the solution of optimization problems. Consider Figure 1, where we consider how learning is applied to either increase the scale of solvable problems or to increase the ability to deal with practical, partial-information problems. Along the problem scale axis, for example, the operations research community has made
use of learned heuristics to solve TSPs [13], VRPs [14], and general MILPs [15]. Along the information axis, which includes dealing with POMDPs, techniques such as RL play a major role, as well as ideas such as tuning Monte-Carlo Tree Search [16], embedding learned components into optimal control frameworks [17], and learning how to bias sampling planners [18].

Practical multi-robot planning and control builds on the progress along both of these axes: the degrees of freedom and environment complexity increase, while the ability to communicate and coordinate at scale decreases. Traditional centralized approaches would use a planning unit to produces coordinated plans that agents use for real-time on-board control; these have the advantage of producing optimal and complete plans in the joint configuration space but true optimality is NP-hard in many cases [5] and they will struggle when communications are degraded and frequent replanning is required. By contrast, Decentralized approaches reduce the computational overhead [19] and relax the dependence on centralized units [20, 21] to deal with challenged communications, but deal in purely local objectives and cannot explicitly optimize global objectives (e.g., path efficiency).

What the directions of Figure 1 teach us is that success follows from starting with simple problems and using their examples to approach complex ones. This progression from example to application is reminiscent of Imitation Learning, and we use this crucial observation to understand how learning can play a role in mitigating the shortcomings of decentralized approaches in solving challenging multi-robot problems.

Bridging the gap between the qualities of centralized and decentralized approaches, learning-based methods promise to find solutions that balance optimality and real-world efficiency. The process of generating data-driven solutions for multi-robot systems, however, cannot directly borrow from single-robot learning methods because (a) hidden (unobservable) information about other robots must be incorporated through learned communication strategies, and (b), although policies are executed locally, the ensuing actions should lead to plans with a performance near to that of coupled systems. This agenda means that we need to address (i) how to generate multi-robot training data, (ii) how to generate decentralizable policies, and (iii) how to transfer these policies to real-world systems.

The following section elaborates these three key challenges and indicates promising directions.

## 2 Learning Decentralized Policies by Copying Centralized Experts

Though planning complexity is reduced with a decentralized approach, use of a learning-based approach requires consideration of state-action space coverage, especially since introducing multiple agents means the size of the joint state-action again grows exponentially. This core challenge is the reason why the development of learning-based multi-robot controllers is a nascent field. While a number of learning paradigms have been applied to this topic (e.g., RL [22, 23]), this position paper focuses on imitation learning strategies. The following paragraphs discuss three key topics that are central to the learning process: data generation, communication strategies, and sim-to-real transfer.

### 2.1 Experts and Data Generation

**How to generate expert data?** The work by Li et al. [4] shows that it is possible to train decentralized controllers to learn communication and action policies that optimize a global objective by imitating a centralized optimal expert. The former work considered the specific case-study of multi-agent path planning, and used Conflict-Based Search (CBS) [24] to find optimal solutions (i.e., sets of optimal, collision-free paths). Although their results demonstrated unprecedented performance in decentralized systems (i.e., achieving higher than 96% success rates with lower than 7% flowtime increase compared to the expert solution), but observed poor generalization. Simply training the models through behavior cloning leads to bias and over-fitting, since the performance of the network is intrinsically constrained by the dataset. Alternative approaches include learning curricula [25] to optimize the usage of the existing training set, or the introduction of data augmentation mechanisms (such as DAgger [26]), which allow experts to teach the learner how to recover from past mistakes.

**How to augment existing datasets?** One of the major limitations of behavior cloning is that it does not learn to recover from failures, and is unable to handle unseen situations [27]. For example, if the model has deviated from the optimal trajectory at one-time step, the model fails in getting back to states seen by the expert and results in a cascade of errors. One solution (i.e., DAgger [26]) is to
introduce the expert during training to teach the learner how to recover from past mistakes. In [4],
the authors demonstrate the utility of this approach by making use of a novel dataset aggregation
method that leverages an online expert to resolve hard cases during training. Other approaches are
to directly extract a policy from training data, such as GAIL [28]. More broadly speaking, with data
augmentation, one can produce arbitrary amounts of training data from arbitrary probability distri-
butions of interesting parameters, such as roadmap structure, local environment structure, obstacle
density and movement characteristics, and local robots’ configurations. The carefully controlled
parameter distribution enables us to introduce different levels of local coordination difficulties and
generate the most challenging instances in each training stage, to achieve curriculum learning. In
addition, data augmentation allows us to understand the ability boundary of the trained model, to an-
alyze the correlation between different factors, and to find identify parameters that have the strongest
effect on the system performance.

2.2 Communication Strategies for Decentralized Control

What, how and when to send information? While effective communication is key to decentralized
control, it is far from obvious what information is crucial to the task, and what must be shared
among agents. This question differs from problem to problem and the optimal strategy is often
unknown. Hand-engineered coordination strategies often fail to deliver the desired performance,
and despite ongoing progress in this domain, they still require substantial design effort. Recent
work has shown the promise of Graph Neural Networks (GNNs) to learn explicit communication
strategies that enable complex multi-agent coordination [29, 30, 3, 4]. In the context of multi-robot
systems, individual robots are modeled as nodes, the communication links between them as edges,
and the internal state of each robot as graph signals. By sending messages over the communication
links, each robot in the graph indirectly receives access to the global state. The key attribute of GNNs
is that they compress data as it flows through the communication graph. In effect, this compresses
the global state, affording each agent access to global data without inundating them with the entire
raw global state. Since compression is performed on local networks with parameters that are shared
across the entire graph, GNNs are able to compress previously unseen global states. In the process of
learning how to compress the global state, GNNs also learn which elements of the signal are the most
important, and discard the irrelevant information [29]. This produces a non-injective mapping from
global states to latent states, where similar global states “overlap”, further improving generalization.

Are all messages equally important? Unfortunately, if communication happens concurrently and
equivalently among many neighboring robots, it is likely to cause redundant information, burden
the computational capacity and adversely affect overall team performance. Hence, new approaches
towards communication-aware planning are required. A potential approach is to introduce attention
mechanisms to actively measure the relative importance of messages (and their senders). Attention
mechanisms have been actively studied and widely adopted in various learning-based models [31],
which can be viewed as dynamically amplifying or reducing the weights of features based on their
relative importance computed by a given mechanism. Hence, the network can be trained to fo-
cus on task-relevant parts of the graph [32]. Learning attention over static graphs has shown to be
efficient. Liu et al. [33] developed a learning-based communication model that constructs the com-
munication group on a static graph to address what to transmit and which agent to communicate for
collaborative perception. However, its permutation equivariance, time invariance and its practical
effectiveness in dynamic multi-agent communication graphs have not yet been verified. Recently, Li
et al. [34] integrate an attention mechanism with a GNN-based communication strategy to allow for
message-dependent attention in a multi-agent path planning problem. A key-query-like mechanism
is developed to determine the relative importance of features in the messages received from various
neighboring robots. Their results show that it is possible to achieve a performance close to that of a
coupled centralized expert algorithm, while scaling to problem instances that are ×100 larger than
the training instances.

2.3 Sim-to-Real Transfer

Expert data is typically generated in a simulation, yet policies trained in simulation often do not
generalize to the real world. This is referred to as the reality gap [35].

Why is sim-to-real transfer difficult? Even though simulations have become more realistic and
easily accessible over recent years [36, 37], it is computationally infeasible to replicate all aspects
of real-world physics in a simulation since the uncertainty and randomness of complex robot-world interactions are difficult to model. Domain randomization is an intuitive solution to this problem, but also makes the task to learn harder than necessary and therefore results in sub-optimal policies. While the reality gap is a major challenge in computer vision, robotics also deals with the physical interaction with the real world and physical constraints such as inertia, for example in robotic grasping [38, 39], drone flight [40, 41] or robotic locomotion [42, 43].

**Why is sim-to-real transfer even more difficult for multi-robot systems?** While sim-to-real in the single-robot domain typically deals with robot-world interaction, the multi-robot domain is also concerned about robot-robot interactions. An example of this is a swarm of drones flying closely to each other and turbulence affecting the motions of other drones in the vicinity. We already have established that communication is key to efficient multi-robot interaction, but it is not obvious how such communications are affected by the reality gap. Multi-robot coordination is typically trained in a synchronous manner, but when deploying these policies to the real-world, decentralized communication is asynchronous. Furthermore, randomness such as message dropouts and delays are typically not considered during synchronous training. To the best of our knowledge, no research has been conducted that evaluates those factors and the impact they have on the performance of policies. Decentralization is key to successful multi-agent systems, therefore decentralized mesh communication networks are required to operate multi-robot systems in the real world, which may pose additional challenges to the sim-to-real transfer. Lastly, during cooperative training it is typically assumed that all agents are being truthful about their communications, but faulty and malicious agents can be part of the real world and cause additional problems [23, 44].

**How can we close the reality gap?** We see a few possible avenues to tackle the sim-to-real transfer for multi-robot communication. Domain randomization is an intuitive method for making the real-world a permutation of the training and likely improves performance, potentially even against faulty agents and adversarial attacks [23], yet leading to sub-optimal policies. More realistic (network) simulations [45] are always helpful, but also costly alternatives. Methods such as sim-to-real via real-to-sim [46] or training agents in the real-world in a mixed reality setting [47] and federated, decentralized learning where individual robots collect data and use it to update a local model that is then aggregated into a global model can benefit the sim-to-real transfer [48, 49].

### 3 Future Avenues

The sections above lay out the challenges entailed by the described approach. Yet, this begs the following two questions:

**Is imitation learning the right paradigm?** There are two main approaches to training a controller for a multi-robot system: imitation learning (e.g., [50]) and reinforcement learning (e.g., [51]). The most obvious benefit to RL is that it does not require an expert algorithm, as it simply optimizes a reward. However, the reward function requires careful consideration to guarantee that the learned controller does not exploit it by using unsafe or inappropriate actions. Conversely, IL is often biased around regions which can be reached by the expert and, consequently, if the controller ever finds itself in a previously unseen situation, it might exhibit unpredictable behavior. Finally, IL is inherently limited by the expert algorithm. As such, possible future directions should explore the combination of both IL and RL (e.g., [22]) in the context of decentralized multi-robot systems.

**Is it possible to learn small-scale coordination patterns for large-scale systems?** Ideally, we hope that controllers trained on only a few robots (which not only facilitates data generation, but also accelerates the training process), can then be deployed on large-scale systems with hundreds and even thousands of robots. Achieving the above expectation may not be far away from us. A recent example can be found in [52], where the local coordination behaviors and conventions learned in a partially observable world successfully scales up to 2048 mobile robots in crowded and highly-structured environments. In [53], a promising demonstration shows that the policy trained in $20 \times 20$ maps with only 10 robots obtains a success rate above 80% in $200 \times 200$ maps with 1000 robots, and more impressively, the learned policy only spends $\frac{1}{39}$ computation time, compared to the centralized expert. Overall, these preliminary results give us confidence that we should continue leveraging methods, such as IL, to distill offline-optimal algorithms to online-scalable controllers.
References


