Approximating the Value of Collaborative Team Actions for Efficient Multiagent Navigation in Uncertain Graphs

Martina Stadler, Jacopo Banfi, Nicholas Roy
MIT CSAIL
32 Vassar Street
Cambridge, MA 02139-4309 USA
{mstadler, jbanfi, nickroy}@mit.edu

Abstract

For a team of collaborative agents navigating through an unknown environment, collaborative actions such as sensing the traversability of a route can have a large impact on aggregate team performance. However, planning over the full space of joint team actions is generally computationally intractable. Furthermore, typically only a small number of collaborative actions is useful for a given team task, but it is not obvious how to assess the usefulness of a given action. In this work, we model collaborative team policies on stochastic graphs using macro-actions, where each macro-action for a given agent can consist of a sequence of movements, sensing actions, and actions of waiting to receive information from other agents. To reduce the number of macro-actions considered during planning, we generate optimistic approximations of candidate future team states, then restrict the planning domain to a small policy class which consists of only macro-actions which are likely to lead to high-reward future team states. We optimize team plans over the small policy class, and demonstrate that the approach enables a team to find policies which actively balance between reducing task-relevant environmental uncertainty and efficiently navigating to goals in toy graph and island road network domains, finding better plans than policies that do not act to reduce environmental uncertainty.

Introduction

We would like to enable a multiagent team to navigate through an unknown environment, represented as a roadmap or motion graph, where some edges may be untraversable. For a single agent, this problem is known as the Canadian Traveller's Problem (CTP), and can be formulated as a Partially Observable Markov Decision Process (POMDP), where the optimal policy trades off between exploiting paths that are traversable with high probability, but with potentially higher cost, against exploration actions that determine the traversability of potentially lower-cost paths. A collaborative multiagent team considers the same problem, but has additional flexibility to explore the environment with different team members. This flexibility is especially interesting for teams of agents with heterogeneous capabilities (e.g., an air vehicle/ground vehicle), or for tasks where agents are not co-located in the environment (and can provide non-local information to their teammates). For example, consider a team of air and ground vehicles entering an urban environment after a climate disaster, as in Fig. 1. The weather may have rendered some of the routes through the city untraversable for some of the agents, and some of the agents may be able to sense the true state of some paths more quickly than others. A collaborative plan which tasks some of the agents with determining which roadways are impassable due to weather will lead to better overall team performance.

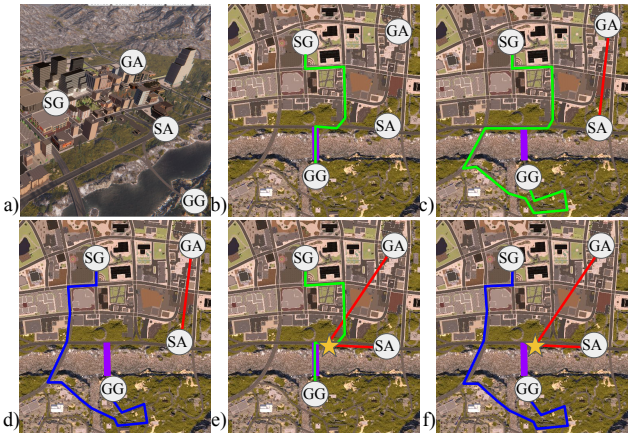

Figure 1: Motivating Example. Consider an air vehicle/-ground vehicle team navigating in a city environment (a) subject to a climate disaster which has impacted the traversability of city infrastructure, like pedestrian bridges (purple). A ground vehicle starting at SG and navigating to GG with no additional information has to select between an unknown trajectory which could lead to either a short path (b, green) or a long detour (c, green) and a medium-length, known trajectory (d, blue), either of which can be suboptimal given the unknown state of the pedestrian bridge. By taking a small diversion to the gold star, the air vehicle (with start SA and goal GA) can sense and share the state of the bridge with the ground vehicle, allowing it to select the best available path to GG for every environmental state (e, green and f, blue), reducing expected total team navigation cost.

While the flexibility of multiagent teams can improve navigation outcomes in unknown environments, generating team plans in this domain requires answering three ques-

tions: *what* is useful to explore given the team objective(s), by *when* does the team need the explored information to improve planning performance, and *who* should explore. Answering these questions can require a multiagent team to reason over a POMDP with large action and observation spaces, long time horizons, and delayed rewards, each of which are individually studied as difficult problems in the POMDP literature (e.g., Silver and Veness 2010; Sunberg and Kochenderfer 2018; Somani et al. 2013; Theocharous and Kaelbling 2003; He, Brunskill, and Roy 2011; Lim, Sun, and Hsu 2011; Lee, Cai, and Hsu 2021; Arjona-Medina et al. 2019; see Kurniawati (2022) for a comprehensive review). We choose to simplify our multiagent planning model by relying on improved terrain scoring algorithms, such as cost-to-go prediction from satellite imagery (Everett, Miller, and How 2019) and traversability prediction from onboard sensors (Guan et al. 2022), to compactly represent perceptual uncertainty using a sparse graph, enabling us to focus on the challenge of finding good collaborative team plans at long time horizons with delayed rewards.

In this work, we model collaborative team policies on stochastic graphs using macro-actions, where each macro-action can consist of multi-step movement actions, sensing actions, and actions to wait for sensor data from other teammates. To reduce the number of macro-actions considered during planning, we generate optimistic approximations of candidate future team states, then generate a small policy class for planning which only includes macro-actions which are likely to lead to high-reward future team states. We optimize team plans over the small policy class. Our approach enables a team to find policies which balance between reducing task-relevant environmental uncertainty and efficiently navigating to goals in toy graph and island road network domains. We compare our approach to two planners which do not use collaborative macro-actions for planning, and demonstrate improved plan quality. We also compare our approach to a planner which does not use approximations to generate a small policy class for collaborative team planning, and demonstrate improved planning efficiency.

Related Work

Our problem and approach are based on the Canadian Traveller's Problem (CTP) and Deterministic POMDP (Det-POMDP) formulations. We model multiagent team navigation as a Canadian Traveller's Problem (CTP), or navigation on graphs with stochastic edges, which is commonly used as the model for planning in environments with averse weathers or stale map information (Papadimitriou and Yannakakis 1991; Eyerich, Keller, and Helmert 2010). While determining exact solutions to the CTP for non-trivial graphs is PSPACE-Complete (Papadimitriou and Yannakakis 1991; Fried et al. 2013), a number of high-quality approximate policies have been developed to solve the single-agent navigation problem on CTP graphs (Eyerich, Keller, and Helmert 2010). A number of extensions have also been considered for the CTP, including CTP graphs with uncertain edge traversal costs instead of stochastic edges (Nikolova and Karger 2008) and disjoint-path CTPs (Bnaya, Felner, and Shimony 2009; Bnaya et al. 2011). Most similar to our approach are Bnaya et al. (2011) and Bnaya, Felner, and Shimony (2009), which consider multiagent planning and remote sensing on CTPs, respectively, yet both approaches have limited scope; Bnaya et al. (2011) considered the special multiagent planning case when there is no cost for an agent to remain at its current vertex, reducing the approach to a series of sequential CTP problems, and Bnaya, Felner, and Shimony (2009) developed solutions for a sensing agent with limited dynamics and no independent planning task. In our approach, we explicitly consider the cost of waiting to receive environmental information from a teammate, and consider sensing agents which are required to execute individual goal-directed navigation tasks.

POMDP planning with deterministic actions and observations has also been studied as the Deterministic POMDP (Det-POMDP, Bonet 2009); we combine the Det-POMDP formulation with the stochastic graph structure given by the Canadian Traveller's Problem (CTP) to generate a model of the planning environment in which it is feasible to consider long-horizon, multiagent collaborative actions with delayed rewards. The multiagent planning problem in large POMDPs has also been studied in the decentralized setting (Bernstein et al. 2002; Amato, Konidaris, and Kaelbling 2014; Amato et al. 2016). In our approach, we assume perfect, centralized communication between agents to enable online policy adaptation based on team sensing actions in the environment, rather than requiring multiagent collaboration to occur during offline planning. Despite abstracting away the complexities of perception and communication in unknown environments, the computation required to identify high-quality collaborative sensing actions in large, outdoor environments can result in problem intractability.

Approach

Multiagent Planning on Unknown Graphs

We would like to model a team Λ of l (possibly heterogeneous) robots λ, $\Lambda = \{\lambda_1, ..., \lambda_l\}$, tasked with goal-directed navigation in a static environment using an imperfect roadmap. For example, consider a team navigating in an outdoor environment guided by a graph derived from a stale or incomplete overhead image (Eyerich, Keller, and Helmert 2010), or a team navigating on a graph of a previously known environment subject to a significant event, like a climate disaster. This scenario is well captured by the probabilistic graph formulation proposed in the Canadian Traveller's Problem (CTP). We formalize the navigation environment as follows: let G be an undirected graph with vertices V and m edges E, $G = \langle V, E \rangle$, where the cost of traversing edge e by agent λ is given by a known scalar cost function[1], $c_\lambda : E \rightarrow \mathbb{R}^+$; we assume c_λ is agent finishing time. Additionally, we assume that the traversability (weather) of each edge w_e is initially unknown but takes a binary value, $w_e \in \{0, 1\}$, and an environment weather $w = \{w_1, w_2, ..., w_m\}$ is defined as a joint assignment of graph edges to weathers w_e. At the beginning of each planning instance, w is drawn from m independent Bernoulli distribu-

[1]For notational convenience, we overload the cost function to take the endpoints of an edge as input: $c_\lambda : V \times V \rightarrow \mathbb{R}^+$.

tions with parameters $1 - \rho_e$, where $\rho_e \in [0, 1)$ is the probability that an edge is blocked, or untraversable, such that $p(w) = \prod_{e \in E} \text{Bern}(1 - \rho_e)$. We assume that the weather changes slowly relative to the timescale of team navigation (e.g., in the aftermath of a single event which changed the environment, like a climate disaster, or due to long timescale infrastructure wear or environmental change), so we assume w is constant during planning and policy execution. We also assume that each agent has a local, noiseless sensor which can acquire ground truth observations $z_e = w_e$ of the traversability of edges incident to the agent's current vertex. These two assumptions mean that the only nondeterminism in the problem is caused by the initially unknown weather, and once the traversability of an edge is (noiselessly) observed, it does not change, allowing agents to act deterministically on the observed subgraph. We also assume that team communication is perfect and centralized, and that each agent is equipped with a low-level controller capable of avoiding collisions, enabling two or more agents to occupy a graph vertex or travel along the same edge simultaneously.

The goal of the multiagent planning problem on the CTP is to find a valid team policy π which takes each robot from its initial vertex v_i^s to its goal vertex v_i^g, where $v_i^s, v_i^g \in V \, \forall i \in \{1, ..., l\}$, and which maximizes a scalar team reward function R (e.g., makespan) in expectation over weathers. The problem can be formulated as a Multi-Agent Deterministic Mixed Observability Markov Decision Process (MADMOMDP):

- Team: The team Λ of l agents, $\{\lambda_1, ..., \lambda_l\}$.
- Fully observable state S: The set of graph vertices for each agent V^l.
- Partially observable state: The set of weathers W.
- Actions: The set of incident, traversable graph edges for each agent from each vertex, including self-loops.
- Observations: The traversability w_e of an edge, $z_e = w_e \in \{0, 1\}$.
- Observable state transition function: Deterministic transitions along observed, incident, traversable graph edges to adjacent vertices.
- Partially observable state transition function: No transition, as weather w is assumed constant during a trial.
- Observation function: Deterministic observations of the traversability w_e of sensed, incident graph edges.
- Reward R: The negative team makespan, or the negative maximum completion time of any agent in the team, $R = -\max_{\lambda \in \Lambda} \sum_{t=0}^{\infty} c_\lambda(v_{\lambda,t}, v_{\lambda,t+1})$, where, by assuming the reward of an agent remaining in its goal state is zero, we ensure the stationary policy has zero reward and the infinite sum is well defined.
- Belief b: The probability distribution over the partially observed state, the weather, $b = p(w)$, where $b \in B$.
- Initial belief b_0: The prior belief over the weather, $b_0 = \{\text{Bern}(1 - \rho_1), ..., \text{Bern}(1 - \rho_m)\}$.

We can write the multiagent planning objective as the maximization of the MADMOMDP value function $V^\pi(v, b)$ over multiagent policies $\pi \in \Pi$,

$$\pi^*(v, b) = \arg\max_{\pi \in \Pi} V^\pi(v, b), \tag{1}$$

where Π is the space of multiagent policies, $\pi : S \times B \to A$,

$$V^\pi(v_a, b_a) = \mathop{\mathbb{E}}_{w \sim b_a} \left[-\max_{\lambda \in \Lambda} \sum_{t=a}^{\infty} c_\lambda(v_{\lambda,t}^\pi, v_{\lambda,t+1}^\pi) \right], \tag{2}$$

and $v_{\lambda,t}^\pi$ is the vertex reached by agent λ at time t when executing policy π from time a until time t.

Macro-actions for Information-gathering Teams

In practice, optimizing Eq. 1 over the space of all collaborative team policies Π is intractable due to the size of the team action space and the length of team plans (Bnaya et al. 2011). One approach to increasing planning tractability is using macro-actions $a \in A$, or multi-step actions, which are often selected using domain knowledge. For the single-agent CTP, optimal plans can be generated by optimizing over single-agent macro-action policies $\pi \in \Pi_{sa}$ which consist of only two different types of macro-actions: 1) optimal navigation in the observed subgraph to vertices adjacent to an unknown edge and 2) navigation to the goal (Eyerich, Keller, and Helmert 2010), where Π_{sa} is the set of policies which can be represented using the two single agent macro-action types. Intuitively, the single-agent macro-action based polices are optimal because single-agent MADMOMDP policies depend only on the agent's belief b about the unknown weather w. By construction, an agent's belief is constant during the execution of a single-agent macro-action a, so the highest reward agent policy π does not change. Planning using macro-actions reduces policy search depth and breadth, as they reduce the number of timesteps to the goal, and they eliminate policies with suboptimal known-graph traversals from the policy space.

We would like to apply macro-actions to multiagent CTPs to increase the tractability of solving Eq. 1. The collaborative nature of multiagent systems means we need to consider additional macro-actions, specifically 1) sensing an edge and 2) waiting for the edge to be sensed and the result transmitted. However, remote team sensing invalidates the optimality of the single-agent macro-action set, as teammates can update the team belief during one agent's macro-action execution. Additionally, waiting actions can increase the branching factor of macro-action based search, as optimal wait durations are not known before planning, so wait actions with multiple durations must be considered. For example, consider the GV in Fig. 1; the GV should combine waiting and moving so it is positioned at the fork in the road where the green and blue paths diverge when the AV senses the bridge.

Fortunately, while good team policies can be spatially and temporally complex, in many problems of interest, only a small number of collaborative macro-action policies are useful for a specific team planning task. For example, in Fig. 1, the only useful collaborative macro-action for the AV is to sense the traversability of the purple bridge on the potentially low cost (green) GV path to the goal; all other sensing actions will not improve team planning performance. Our key realization is that we can use the structure of the Det-POMDP belief space to approximate the value of a hypothetical team belief without explicitly planning the collaborative actions that lead to the belief. This approximation allows us

to focus computation on the (often small) set of collaborative team policies $\hat{\Pi}$ that are likely to lead to high-reward team beliefs, while pruning collaborative team policies that are likely to lead to low-reward team beliefs and are therefore unlikely to improve team planning performance.

Policy Class Augmentation for Planning

In this work, we assume we are given a small base class of single-agent macro-action policies Π_{sa} that involve moving agents to graph vertices and a much larger class of complex multiagent macro-action policies Π_{ma} that include sensing and waiting actions. We would like to generate a small, high quality policy class $\hat{\Pi}$ for planning by augmenting the base policy class Π_{sa} with a small number of candidate policy classes $\Pi_c \subset \Pi_{ma}$ which are likely to improve team planning performance. While Π_c can in principle be any subset of Π_{ma}, it can be useful when each candidate policy class has a meaningful interpretation in policy space (e.g., Π_c is the set of team policies where an AV senses edge e). In this work, we define candidate policy classes as the set of macro-action policies which include a specific macro-action a. The value of a candidate policy class Π_c for planning is the expected increase in planning performance when planning using $\hat{\Pi}$ as compared to planning using the single-agent macro-action policy class Π_{sa}:

$$\Delta(\Pi_{sa}, \Pi_c, v, b) = \max_{\pi \in \Pi_{sa} \cup \Pi_c} V^{\pi}(v, b) - \max_{\pi \in \Pi_{sa}} V^{\pi}(v, b). \quad (3)$$

We include a candidate policy class Π_c in $\hat{\Pi}$ if planning using the augmented class results in an improvement in planning performance $\Delta(\Pi_{sa}, \Pi_c, v, b)$:

$$\hat{\Pi}(v, b) = \Pi_{sa} \cup \{\Pi_c | \Pi_c \in \Pi_{ma}; \Delta(\Pi_{sa}, \Pi_c, v, b) > \gamma\}, \quad (4)$$

where $\gamma \geq 0$ is a tunable parameter. When γ is low, collaborative actions are added to the policy class if they can lead to any improvement in total team cost; when γ is high, collaborative actions are only added to the policy class if they can lead to a significant improvement in total team cost. In our experiments, we use a low γ which reduces the impact of floating point errors on policy class selection, but does not otherwise impact action selection.

While the defined policy class $\hat{\Pi}$ is likely to contain a small number of high-reward collaborative team policies and result in efficient, high-reward planning, unfortunately, computing $\Delta(\Pi_{sa}, \Pi_c, v, b)$ for all $\Pi_c \in \Pi_{ma}$ can be expensive, as evaluating each $\Delta(\Pi_{sa}, \Pi_c, v, b)$ may require calculating value functions based on complex multiagent macro-action policies. Next, we 1) discuss the structure of the Det-POMDP belief space, which results from deterministic actions and observations, and 2) discuss team compositions for which we can efficiently generate good approximations of $\Delta(\Pi_{sa}, \Pi_c, v, b)$, and therefore efficiently generate good $\hat{\Pi}$.

Approximations for Policy Selection with One Stochastic Agent

Preliminaries: Belief-based Value Functions First, we discuss the structure of the Det-POMDP belief space. Consider the beliefs b and b^z, where b^z is the updated team belief after receiving observation z, i.e., $b^z = h(b, z)$ given belief update function h. Note that the observation z is independent of the sensing agent. Because the team operates in a Det-POMDP where the hidden environment weather is constant during planning, the (deterministic) agent observations never increase team environmental uncertainty, i.e., the size of the support of the team belief decreases monotonically over the planning trial (Bonet 2009). Under the assumption of *rational* planning (Russell and Norvig 2021), or that each agent maximizes its reward given its knowledge of the environment, additional environmental information does not decrease planning performance, and the value of a policy generated using b^z will be at least as good as the value of a policy generated using b:

$$V^{\pi}(v, b^z) \geq V^{\pi}(v, b). \quad (5)$$

We use this insight to compare the relative values of belief states for different team compositions to generate a good policy class for planning. Specifically, we generate plans for teams with one agent which acts on the CTP graph with unknown edge traversabilities, like a ground vehicle (GV) navigating on a damaged road network, and one or more agents which act on a fully traversable graph of the environment[2], like an air vehicle (AV) flying over damaged terrain. In the remainder of the exposition, we will use the terms stochastic agent/GV and deterministic agent/AV interchangeably.

We focus on the team composition of one GV and multiple AVs because, for a given team plan, the time at which the team belief will change is known – it is the earliest time at which any AV reaches an edge to sense, or when the GV navigates to a vertex with an unknown incident edge. In future work, we are interested in developing approximations for team compositions with multiple GVs.

Value Function Approximations for a 1-GV/n-AV Team Consider a 1-GV/n-AV team navigating on a CTP graph. We assume that AVs do not observe the traversability of edges while navigating unless specifically tasked to sense, and that the GV immediately observes the traversability of all incident edges of its current vertex.

We would like to generate an augmented policy class $\hat{\Pi}$ which is likely to contain high-quality solutions to the optimization in Eq. 1. We begin by identifying the types of macro-action classes $\Pi_c \in \Pi_{ma}$ to consider for planning, assuming a base class Π_{sa} of single-agent, non-collaborative macro-action policies. The first is collaborative sensing macro-action policies Π_{c_s}, in which one or more AV(s) travel to a vertex and sense an incident edge's traversability and transmit the information to the GV. The second is collaborative waiting macro-actions Π_{c_w}, in which one or more AV(s) travel to a vertex to sense the environment, and the GV combines movement and waiting macro-actions to optimally take advantage of the sensed information. For example, the GV may select movement and waiting actions to arrive at a fork between two paths just as additional information about the paths is sensed and communicated.

In the following sections, we develop approximations for $\Delta(\Pi_{sa}, \Pi_c, v, b)$ for sensing and waiting macro-actions.

[2]The fully traversable graph may include vertices and edges which are never reachable or traversable for the stochastic agent, respectively, and are therefore not included in the CTP graph.

Approximating $\Delta(\Pi_{sa}, \Pi_{c_s}, v, b)$ for Sensing Actions First, we discuss approximations for sensing policies Π_{c_s}. Each sensing action can be thought of as two separate single-agent actions: an AV sensing the desired edge e with a corresponding cost of traveling to sense the edge, and the GV planning using the sensed information with a potential reduction in overall plan cost resulting from knowing edge traversabilities, both of which can impact team makespan. Because all graph edges are traversable for the AV, the cost c_s of detouring from the optimal AV path (with cost $c(v, G)$) to sense edge e before continuing to the goal can be computed exactly using a shortest paths algorithm,

$$c_s = \underset{v' \in \{v_{e_0}, v_{e_1}\}}{\arg\min} \; c(v, v') + c(v', G), \tag{6}$$

where we assume an edge can be sensed at either of its endpoints v_{e_0} and v_{e_1}, and that if either v_{e_0} or v_{e_1} is on the AV's shortest path to the goal, the expected loss due to sensing is zero. We also recover the deterministic edge sensing time,

$$t_s = c(v, v'), \tag{7}$$

where v' is the minimizing vertex in Equation 6.

We can now consider how the sensing action reduces the GV cost. To directly calculate the benefit of the sensing action to the GV, we would need to find an optimal joint GV/AV policy, which may contain a combination of movement, sensing, and waiting actions, and may be computationally intractable to generate. However, we can use Eq. 5 to generate bounds for the joint policy that are inexpensive to compute, based on the value functions of an agent which does not receive sensed information and an agent which receives sensed information. First, we consider an agent at vertex v with traversability belief b which does not receive additional sensing information and acts using a single-agent policy π_{sa}, with value function $V^{\pi_{sa}}(v, b)$. This value function can be well-approximated using a high-quality single-agent CTP policy.

Second, we consider an agent at vertex v with traversability belief b which receives an observation of edge e at time t_s and acts using policy π_{ma}, a multiagent sensing policy, with value function $V^{\pi_{ma}}(v, b, e, t_s)$. Calculating this value function exactly requires reasoning over multiagent sensing policies. To avoid solving this complex planning problem, we instead find an upper bound for the expression by observing that Eq. 5 implies that

$$V^{\pi_{ma}}(v, b, e, t_s) \leq V^{\pi_{ma}}(v, b, e, 0), \tag{8}$$

or that the value of plan generated by immediately sensing the unknown edge will be at least as good as the value of a plan where the edge is sensed later during planning. Next, we realize that the value of a multiagent policy with immediate sensing is equivalent to the value of a single-agent policy calculated using the updated belief after sensing, b^z,

$$V^{\pi_{ma}}(v, b, e, 0) = \sum\nolimits_{z \in Z} p(z) V^{\pi_{sa}}(v, b^z | z = z), \tag{9}$$

where b^z is the agent's traversability belief after receiving an observation of edge e. Combining Eqs. 8 and 9, we recover the following upper bound,

$$V^{\pi_{ma}}(v, b, e, t_s) \leq \sum\nolimits_{z \in Z} p(z) V^{\pi_{sa}}(v, b^z | z = z), \tag{10}$$

which can be well-approximated using a rollout-based single-agent CTP policy. Using Eqs. 6 and 10, we recover an upper bound for Δ, the expected change in team makespan caused by using sensing actions, that is efficient to compute,

$$\Delta(\Pi_{sa}, \Pi_{c_s}, v, b) \leq \max\big\{\sum\nolimits_{z \in Z} p(z) V^{\pi_{sa}}(v, b^z | z = z), c_s\big\} - \max\big\{V^{\pi_{sa}}(v, b), c(v, G)\big\}. \tag{11}$$

Approximating $\Delta(\Pi_{sa}, \Pi_{c_w}, v, b)$ for Waiting Actions Next, we consider waiting policies Π_{c_w}. Each waiting action can be thought of as two separate single-agent actions taken by the GV: 1) waiting to begin moving, and 2) following a policy to the goal. First, we consider the wait-to-move action. Because determining the optimal waiting time requires solving the multiagent problem, we can compute a lower bound of the cost of the wait-to-move action by assuming that the agent does not wait to move, and the wait cost is zero. Second, we approximate the value of planning to the goal after waiting. We can again use Eq. 5 to develop efficiently computable bounds for the joint policy. To generate the bound, we consider the value functions of two agents: one which waits, and one which does not wait. First, we consider the value function of an agent which receives a new edge observation at t_s and which cannot select wait-to-move actions at any time during planning, $V^{\pi_{ma}}(v, b, e, t_s)$. The value of this policy is equivalent to the value of a two-step single-agent policy which first navigates until the sensing horizon with no information about e or the AV's sensing action, then replans using the updated belief and navigates from the sensing horizon to the goal:

$$V^{\pi_{ma}}(v, b, e, t_s) = V^{\pi_{sa}}_{t_s}(v, b) + V^{\pi_{sa}}(v_{t_s}, b^z_{t_s}), \tag{12}$$

where V_{t_s} indicates the value function for navigating until time t_s with belief b, and v_{t_s} and $b^z_{t_s}$ are the vertex and belief of the agent at t_s, assuming π_{sa} was followed until t_s and observation z was received. Second, we consider the value function of an agent which receives a new edge observation from the AV executing a_s at macro-action sensing time t_s and which can select wait-to-move actions until some waiting time t_w before or during sensing, $V^{\pi_{ma}}(v, b, e, t_s, t_w)$, where $t_w \leq t_s$. Determining this value function can require considering waiting team actions; we again avoid this computation by generating an upper bound. First, we note that Eq. 5 implies that waiting for additional sensing information can only improve the value of future actions in the waiting policy (assuming no wait cost):

$$V^{\pi_{ma}}(v, b, e, t_s, t_w) \leq V^{\pi_{ma}}(v, b, e, t_s, t_s). \tag{13}$$

Next, we realize that the value of a policy which waits until the sensing time to take an action is equivalent to the value of a policy which senses the edge immediately and does not wait, again assuming no wait cost.

$$V^{\pi_{ma}}(v, b, e, t_s, t_s) = V^{\pi_{ma}}(v, b, e, 0). \tag{14}$$

Using Eqs. 9, 13, and 14, we recover a bound for $V^{\pi_{ma}}(v, b, e, t_s, t_w)$,

$$V^{\pi_{ma}}(v, b, e, t_s, t_w) \leq \sum\nolimits_{z \in Z} p(z) V^{\pi_{sa}}(v, b^z | z = z), \tag{15}$$

which can be well-approximated using a rollout-based single-agent CTP policy. We recover the tractable Δ bound:

$$\Delta(\Pi_{sa}, \Pi_{c_w}, v, b) \leq \max\big\{\sum\nolimits_{z \in Z} p(z) V^{\pi_{sa}}(v, b^z | z = z), c_s\big\} - \max\big\{V^{\pi_{sa}}_{t_s}(v, b) + V^{\pi_{sa}}(v_{t_s}, b^z_{t_s}), c_s\big\}. \tag{16}$$

Using Problem Structure to Reduce the Number of Candidate Policy Classes

Unfortunately, even calculating approximations of policy quality Δ for all candidate policy classes Π_c to generate $\hat{\Pi}$ can be computationally expensive. However, in many structured environments, there exists a macro-action ordering property which can be used to select an order in which to try to add Π_c to $\hat{\Pi}$, and to prune some candidate policies Π_c without approximation. For example, a policy class with a waiting action can only improve planning performance if it also includes a sensing action that was useful for planning; otherwise, the agent will consider waiting for information that will never come or is not useful. For the single AV case, we first calculate $\Delta(\Pi_{sa}, \Pi_{c_s}, v, b)$ for a sensing policy class Π_{c_s}. If the sensing action does not improve performance, the action is not included in $\hat{\Pi}$, and we do not approximate the value of the corresponding waiting candidate policy class Π_{c_w}. Otherwise, we calculate $\Delta(\Pi_{sa}, \Pi_{c_w}, v, b)$ for waiting macro-action policy class Π_{c_w} to determine if it is beneficial to sense and wait. If waiting improves performance, the waiting macro-action is added to $\hat{\Pi}$; otherwise, only the sensing macro-action is added to $\hat{\Pi}$. For the n-AV case, we calculate $\Delta(\Pi_{sa}, \Pi_{c_s}, v, b)$ for every edge, then greedily assign an edge to each agent (note that the improvement due to sensing is sensing-agent independent, while the cost of the sensing action is sensing-agent dependent). Given the assignment, we calculate the value of sensing and the value of waiting for the multi-agent action, as in the single-agent case, and generate $\hat{\Pi}$. While we consider a single greedy edge assignment per timestep, more complex assignment and approximation strategies could be used.

Planning using Δ and Multiagent Macro-Actions on CTPs

Finally, we discuss a planner which uses a macro-action policy class $\hat{\Pi}$ to optimize Eq. 1 in an online, centralized multiagent CTP solver with perfect communication. As input, we take a MADMOMDP, a baseline policy class to use in all planning instances, like single-agent macro-action policies Π_{sa}, and a set of multiagent macro-action policies to consider, Π_{ma}. In some cases, we are also given an ordering property. We assume that edges incident to GVs at the start of an instance are observed.

Planning begins by approximating $\Delta(\Pi_{sa}, \Pi_c, v_{GV}, b_{GV})$ for every Π_c for the GV (unless previous computations combined with policy orderings guarantee that $\Delta(\Pi_{sa}, \Pi_c, v_{GV}, b_{GV}) < \gamma$) and generating a $\hat{\Pi}$ according to Eq. 4. Then, we optimize Eq. 1 over $\hat{\Pi}$ to generate a team macro-action policy π, which may include collaborative macro-actions. Team policies are executed as follows: first, each agent begins to execute its current macro-action. When any subset of agents finishes their current macro-actions, they update their belief and communicate their belief with each other and the central planner. The central planner replans, and the finished agents begin their next macro-actions. Additionally, non-finishing agents truncate their macro-actions as soon as possible (i.e., at the end of their current primitive action, or graph edge), communicate with the central planner to get the updated team belief, and replan. This process repeats until all agents reach their respective goals.

Metric ↓ better	**Oracle**	**Independent**	**Passive Comms**	**Comms + Approx**	**Comms +No Approx**
Avg % Regret	0	81.82	81.07	**4.00**	**4.00**
Avg Makespan	80.27	104.10	103.30	**82.27**	**82.27**
SEM Cost	3.86	3.08	3.02	3.86	3.86
Avg GV Cost	80.27	104.10	103.30	**82.27**	**82.27**
SEM GV Cost	3.86	3.08	3.02	3.86	3.86
Avg AV Cost	1.0	**1.0**	**1.0**	7.0	7.0
SEM AV Cost	0.0	0.0	0.0	0.0	0.0
Avg Time (s)	-	**1.18**	**1.18**	24.29	42.62
SEM Time (s)	-	0.01	0.01	0.06	0.15

Table 1: Toy Graph Results. We report the average (Avg) normalized percent regret from optimal, average team makespans, AV costs, and GV costs in simulation units (u), algorithm execution times, and standard errors of the mean (SEM). Our approach, *Comms+Approx*, generated lower makespan plans than the single agent planning approaches, *Independent* and *Passive Comms*, and was more time efficient than *Comms+No Approx*, which did not use approximations to generate a small policy class $\hat{\Pi}$.

Experiments

Experimental Setup

We benchmarked our informed, policy-class aware multiagent planner, *Comms+Approx*, against three multiagent planners, *Independent*, *Passive Comms*, and *Comms+No Approx*. In *Comms+Approx* planning, the multiagent team planned using the described approach, and the GV sensed all incident edges during plan execution, while the AV(s) sensed only requested edges. In *Independent* planning, each agent planned independently, and did not share information with the team; for this case, we assumed that the GV sensed all incident edges during plan execution, and the AV(s) did not sense. In *Passive Comms* planning, each agent planned independently, but the agents shared, updated, and planned using a joint belief space; for this case, we assumed that the GV sensed all incident edges during plan execution, and the AV(s) sensed all traversed edges during plan execution. In *Comms+No Approx*, the team planned using a modified version of *Comms+Approx* which did not generate approximations and planned over the full multiagent macro-action policy class, $\hat{\Pi} = \Pi_{sa} \cup \Pi_{ma}$. We also compared our approach to an oracle, *Oracle*, which planned using Dijkstra's algorithm (Dijkstra et al. 1959) on a fully observed graph.

We used Monte Carlo simulation to generate value function approximations for planning. As in Eyerich, Keller, and Helmert (2010), we use the same names to refer to algorithms and policies for planning, where executing an algorithm corresponds to optimizing the corresponding policy at each timestep, and a policy is generated online by selecting the macro-action which maximizes the value function of the

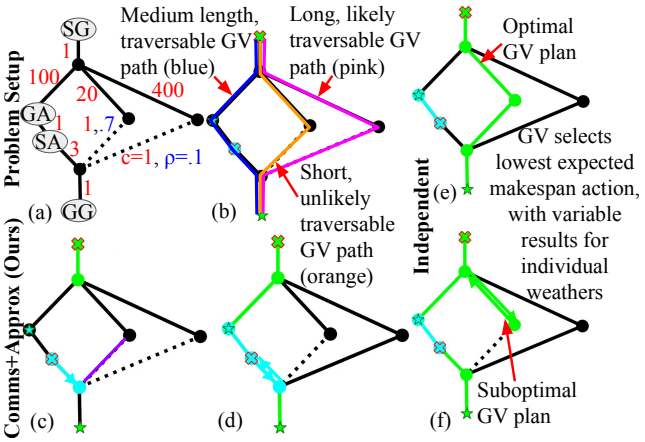

Figure 2: Visual Plan Comparison in the Toy Graph. Plans executed by the AV (blue) and GV (green) navigating from their respective starts (SA/SG) to goals (GA/GG) in the toy graph (a, b), where edge costs and blocking probabilities are labeled in red and blue, respectively. Solid lines represent traversable edges and dotted lines represent edges with uncertain traversability. In the *Independent* plan, the AV navigated directly to the goal, and the GV selected the plan which minimized the expected makespan over all weathers, resulting in highly variable planning performance in some weathers (e, f). In the *Comms+Approx* plan, the AV diverted to sense the unknown edge (purple) on the short path to the goal, while the GV combined waiting and moving and arrived at the fork between paths when the AV sensed the edge on the short path (c). Then, the GV took the shortest traversable path to the goal (d).

Metric ↓ better	**Oracle**	**Independent**	**Passive Comms**	**Comms+ Approx**
Avg % Regret	0	35.02	29.33	**12.81**
Avg Makespan	1151.34	1595.42	1519.35	**1306.94**
SEM Makespan	12.39	26.62	23.55	17.04
Avg GV Cost	1151.34	1595.42	1519.35	**1306.29**
SEM GV Cost	12.39	26.62	23.55	17.06
Avg AV Cost	181.89	**181.89**	**181.89**	1011.05
SEM AV Cost	1.04	1.04	1.04	15.7
Avg Time (s)	-	31.13	**27.26**	5203.42
SEM Time (s)	-	0.75	0.55	199.09

Table 2: Islands Domain Results. We report the average (Avg) normalized percent regret, average team makespans, AV costs, and GV costs in simulation units (u), algorithm execution times (s), and standard errors of the mean (SEM). Our approach, *Comms+Approx*, resulted in lower makespan plans than *Independent* and *Passive Comms*, often by diverting the AV to sense, resulting in larger AV costs but smaller GV costs, reducing team makespan.

macro-action *successor*, or the terminal state of the macro-action. Independent macro-actions were evaluated using the optimistic rollout policy (ORO, Eyerich, Keller, and Helmert 2010), which calculated successor value functions by simulating optimistic planning (assuming unknown edges were traversable) from each successor for r different weathers, where r is the number of rollouts used. We also modified ORO to approximate the sensing value functions by revealing information sensed by other agents during optimistic planning, and by triggering replanning when the team belief was updated. Waiting value functions were evaluated using a combined policy, Greedy Rollout - Greedy policy (GR-G), where value functions were calculated for all navigation-based successors and the current state, which is the result of taking the single timestep wait action. The GR-G policy approximated successor costs by averaging the costs of Greedy policy (G) executions for r rollout weathers; the Greedy policy (G) calculated macro-action values by simulating execution of the optimistic policy in a given rollout weather for each successor and the current state. We planned using $r = 100$ rollouts for all policies, as this provided a good trade-off between computation time and solution quality in preliminary experiments, and $\gamma = 1 \times 10^{-10}$. When evaluating value functions to calculate Δ, we used the same rollout weathers to evaluate each term in the approximation, as this reduced errors in Δ caused by rollout weather stochasticity. Note that because the value functions used to generate the bounds in the Approach section are approximated, we cannot guarantee that our implementation does not prune macro-actions which could lead to high-reward plans due to poor value function approximation. While the OPT, ORO, and GR-G rollout policies were selected due to their good performance for low r, a rollout policy with a regret bound, such as UCT (Kocsis and Szepesvári 2006), could be used to bound the regret of pruning macro-actions using the proposed value function bounds.

Toy Environment

We demonstrated our approach by planning for a 1-GV/1-AV team for 100 trials in the toy scenario in Fig. 2-a-b with GV start SG and goal GG and AV start SA and goal GA for all trials. A weather was drawn randomly from the distribution over weathers $p(w)$ at the start of each trial, and the GV and AV navigated at the same speed. The GV had three possible paths to the goal: one long, likely traversable path, one medium-length, traversable path, and one short-length, likely untraversable path. The non-collaborative planners selected a GV plan without knowledge of the state of the unknown edge on the short path to the goal, which resulted in highly variable GV planning performance given the weather. The performance in (e) matched what the planner expected, but in (f) the unknown edge on the short path was in fact not traversable and the GV had to backtrack in a way the planner did not anticipate, leading to higher overall cost. In contrast, the collaborative planners predicted the potential cost of the unknown edge and used the AV to sense it, while the GV combined waiting and movement macro-actions to arrive at the decision point between paths when the AV sensed and shared the traversability of the edge (c). This enabled the GV to select the shortest path to the goal for every weather (d). While the sensing and waiting actions incurred small constant GV and AV costs in all trials, they resulted in more consistent, ultimately lower makespan team

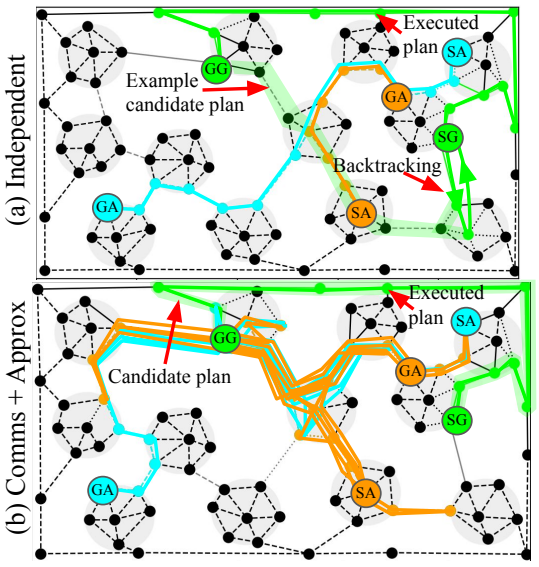

Figure 3: Visual Plan Comparison in the Islands Domain. Example plans executed by the GV (green) and AVs (blue/orange) navigating from their respective starts (SG/SA) to goals (GG/GA) in the islands domain. Dotted, solid, and dashed lines represent untraversable, traversable, and unknown edges at trial termination. (a) The *Independent* planner selected an untraversable GV path (highlighted green), and the GV backtracked after reaching the dead end. (b) In the *Comms+Approx* plan, the AVs navigated to and sensed uncertain edges, and determined the candidate path in (a) was untraversable. The GV waited for information about the candidate path, then navigated to the goal. Note that the AVs did considerable additional sensing work, traversing a large part of the graph to sense edge traversability for the GV, which drove down the team makespan (e.g., by 26.8% here).

plans across weathers. Additionally, the AV did not sense the uncertain edge on the long GV path to the goal, indicating that our approach distinguished between task-relevant and task-irrelevant team sensing macro-actions.

Quantitatively, we showed that our approach found plans with 21.0% and 20.4% lower average makespans as compared to the single agent baselines, *Independent* and *Passive Comms*, respectively, indicating that collaborative planning improved navigation outcomes in the environment. We also showed that our approach found similar makespan plans to *Comms+No Approx* while planning for 43.0% less time on average, demonstrating that the proposed approximations increased planning efficiency without reducing plan quality.

Islands Environment

Next, we demonstrated our approach for a 1-GV/2-AV team navigating in a simulated island environment, modeled as a CTP graph with 90 vertices and 166 edges with euclidean distance edge costs, shown in Fig. 3. We generated 100 environments by sampling edge blocking probabilities uniformly from an allowable range based on edge type. We modeled four different types of edges: highway and on-ramp edges with $\rho \in [0.0, 0.001)$, and local road and bridge edges with $\rho = 0.5$. We generated 10 trials per environment by randomly sampling a start and goal in every graph, then sampling 10 individual trial weathers per graph and start/goal pair, discarding and re-sampling if the start and goal were not connected in the weather. We also discarded and re-sampled start/goal pairs if the GV trajectory generated by the first weather was not at least 8 edges long to remove short, high traversal probability trajectories from our dataset which were unlikely to benefit from collaborative planning. We also assumed that the AVs moved 8x faster than the GVs.

We quantitatively assessed the *Independent*, *Passive Comms*, and *Comms+Approx* planners in Table 2; results for the *Comms+No Approx* approach were omitted due to computational limitations. We also compared to an oracle planner which ran a shortest path algorithm on the fully observed graph for each agent. Our approach found 18.1% and 14.0% lower makespan plans than the *Independent* and *Passive Comms* planners on average. Our approach frequently resulted in shorter GV plans and longer AV plans, indicating that it used AV sensing actions to reduce GV planning uncertainty, which resulted in shorter GV plans and shorter team makespans overall. However, as our approach solved a multiagent POMDP, rather than three independent single agent POMDPs, it was less time efficient: 167x and 191x slower than *Independent* and *Passive Comms*, respectively.

Example plans generated by the *Independent* and *Comms+Approx* approaches are shown in Fig. 3. In *Independent* planning, the GV attempted to reach the goal using an untraversable candidate plan and had to backtrack, while the AVs navigated directly to the goals. In *Comms+Approx* planning, the GV waited while the AVs sensed various candidate paths in the environment, then took a high-quality path to the goal, resulting in a 26.8% lower team makespan.

Conclusion

We presented a novel method for improving collaborative team navigation in stochastic, unknown environments by quickly identifying high-quality collaborative team macro-action policies and approximating their value for planning. Our approach reduced team navigation cost as compared to agents which planed independently, and increased planning efficiency as compared to teams which optimized team policies over an uninformed set of collaborative multiagent policies. In future work, we would like to explore efficient approximations for other team compositions while removing the burden on the user to hand define both the space of candidate collaborative macro-actions and the functions for approximating their values. We are interested in using learned models (e.g., graph neural networks) to generate candidate macro-actions and approximations for more complex teams. We would also like to extend our approach to teams and environments with limited communication (e.g., due to communication delays, dropouts, or dead zones) by incorporating more complex communication models into the MADMOMDP observation function and modifying the belief communication and policy truncation functions.

Acknowledgements

This paper is based upon work supported under the DCIST CRA by the Army Research Laboratory under Cooperative Agreement Number W911NF-17-2-0181 and by the Singapore Defence Science and Technology Agency (DSTA)-MIT Master Research Agreement. The MIT SuperCloud and Lincoln Laboratory Supercomputing Center provided HPC resources that contributed to the results reported within this paper. Their support is gratefully acknowledged.

References

Amato, C.; Konidaris, G.; Anders, A.; Cruz, G.; How, J. P.; and Kaelbling, L. P. 2016. Policy search for multi-robot coordination under uncertainty. *The International Journal of Robotics Research*, 35(14): 1760–1778.

Amato, C.; Konidaris, G. D.; and Kaelbling, L. P. 2014. Planning with macro-actions in decentralized POMDPs. In *Proceedings of the Thirteenth International Conference on Autonomous Agents and Multiagent Systems*, 1273–1280. Association for Computing Machinery (ACM).

Arjona-Medina, J. A.; Gillhofer, M.; Widrich, M.; Unterthiner, T.; Brandstetter, J.; and Hochreiter, S. 2019. Rudder: Return decomposition for delayed rewards. *Advances in Neural Information Processing Systems*, 32.

Bernstein, D. S.; Givan, R.; Immerman, N.; and Zilberstein, S. 2002. The complexity of decentralized control of Markov decision processes. *Mathematics of Operations Research*, 27(4): 819–840.

Bnaya, Z.; Felner, A.; Fried, D.; Maksin, O.; and Shimony, S. 2011. Repeated-task Canadian traveler problem. In *Proceedings of the Fourth International Symposium on Combinatorial Search*, 24–30.

Bnaya, Z.; Felner, A.; and Shimony, S. E. 2009. Canadian traveler problem with remote sensing. In *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence*, 437–442.

Bonet, B. 2009. Deterministic POMDPs revisited. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, 59–66.

Dijkstra, E. W.; et al. 1959. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1): 269–271.

Everett, M.; Miller, J.; and How, J. P. 2019. Planning beyond the sensing horizon using a learned context. In *Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1064–1071.

Eyerich, P.; Keller, T.; and Helmert, M. 2010. High-quality policies for the Canadian traveler's problem. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*.

Fried, D.; Shimony, S. E.; Benbassat, A.; and Wenner, C. 2013. Complexity of Canadian traveler problem variants. *Theoretical Computer Science*, 487: 1–16.

Guan, T.; He, Z.; Song, R.; Manocha, D.; and Zhang, L. 2022. TNS: Terrain traversability mapping and navigation system for autonomous excavators. In *Proceedings of Robotics: Science and Systems XVIII*.

He, R.; Brunskill, E.; and Roy, N. 2011. Efficient planning under uncertainty with macro-actions. *Journal of Artificial Intelligence Research*, 40: 523–570.

Kocsis, L.; and Szepesvári, C. 2006. Bandit based monte-carlo planning. In *Proceedings of the Seventeenth European Conference on Machine Learning*, 282–293.

Kurniawati, H. 2022. Partially observable Markov decision processes and robotics. *Annual Review of Control, Robotics, and Autonomous Systems*, 5(1): 253–277.

Lee, Y.; Cai, P.; and Hsu, D. 2021. MAGIC: Learning macro-actions for online POMDP planning. In *Proceedings of Robotics: Science and Systems XVII*.

Lim, Z.; Sun, L.; and Hsu, D. 2011. Monte Carlo value iteration with macro-actions. *Advances in Neural Information Processing Systems*, 24.

Nikolova, E.; and Karger, D. R. 2008. Route planning under uncertainty: the Canadian traveller problem. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*, 969–974.

Papadimitriou, C. H.; and Yannakakis, M. 1991. Shortest paths without a map. *Theoretical Computer Science*, 84(1): 127–150.

Russell, S. J.; and Norvig, P. 2021. *Artificial Intelligence: A Modern Approach*. Pearson, 4 edition.

Silver, D.; and Veness, J. 2010. Monte-Carlo planning in large POMDPs. *Advances in Neural Information Processing Systems*, 23.

Somani, A.; Ye, N.; Hsu, D.; and Lee, W. S. 2013. DESPOT: Online POMDP planning with regularization. *Advances in Neural Information Processing Systems*, 26.

Sunberg, Z. N.; and Kochenderfer, M. J. 2018. Online algorithms for POMDPs with continuous state, action, and observation spaces. In *Proceedings of the Twenty-Eighth International Conference on Automated Planning and Scheduling*.

Theocharous, G.; and Kaelbling, L. 2003. Approximate planning in POMDPs with macro-actions. *Advances in Neural Information Processing Systems*, 16.