Autonomous Agents and Multi-Agent Systems (2020) 34:42
https://doi.org/10.1007/s10458-020-09467-6

Multi-agent active information gathering in discrete and continuous-state decentralized POMDPs by policy graph improvement

Mikko Lauri[1] · **Joni Pajarinen**[2,3] · **Jan Peters**[3,4]

Abstract
Decentralized policies for information gathering are required when multiple autonomous agents are deployed to collect data about a phenomenon of interest when constant communication cannot be assumed. This is common in tasks involving information gathering with multiple independently operating sensor devices that may operate over large physical distances, such as unmanned aerial vehicles, or in communication limited environments such as in the case of autonomous underwater vehicles. In this paper, we frame the information gathering task as a general decentralized partially observable Markov decision process (Dec-POMDP). The Dec-POMDP is a principled model for co-operative decentralized multi-agent decision-making. An optimal solution of a Dec-POMDP is a set of local policies, one for each agent, which maximizes the expected sum of rewards over time. In contrast to most prior work on Dec-POMDPs, we set the reward as a non-linear function of the agents' state information, for example the negative Shannon entropy. We argue that such reward functions are well-suited for decentralized information gathering problems. We prove that if the reward function is convex, then the finite-horizon value function of the Dec-POMDP is also convex. We propose the first heuristic anytime algorithm for information gathering Dec-POMDPs, and empirically prove its effectiveness by solving discrete problems an order of magnitude larger than previous state-of-the-art. We also propose an extension to continuous-state problems with finite action and observation spaces by employing particle filtering. The effectiveness of the proposed algorithms is verified in domains such as decentralized target tracking, scientific survey planning, and signal source localization.

Keywords Planning under uncertainty · Decentralized POMDP · Information gathering · Active perception

✉ Mikko Lauri
lauri@informatik.uni-hamburg.de

[1] Department of Informatics, University of Hamburg, Hamburg, Germany

[2] Tampere University, Tampere, Finland

[3] Intelligent Autonomous Systems, Technische Universität Darmstadt, Darmstadt, Germany

[4] Max Planck Institute, Tübingen, Germany

Published online: 10 June 2020

1 Introduction

Autonomous agents and robots can be deployed in information gathering tasks in environments where human presence is either undesirable or infeasible. Examples include monitoring of deep ocean conditions, or space exploration. It may be desirable to deploy a team of agents due to the large scope of the task at hand, resulting in a multi-agent active information gathering task. Such a task typically has a definite duration, after which the agents, the data collected by the agents, or both, are recovered. For example, underwater survey vehicles may be recovered by a surface vessel after completion of a survey mission, or the agents may periodically communicate their data back to a base station.

The overall task can be viewed as sequential decision-making. Each agent takes an action and then perceives an observation. The action taken is determined based on the history of the agents' past actions and observations. Action selection is repeated after perceiving each observation until the task ends. To maximize effectiveness of the team and the informativeness of the data collected, planning is required. Planning produces a policy that prescribes how each team member should act to maximize a shared utility. Utility in information gathering tasks is measured by information-theoretic quantities, such as negative entropy or mutual information. Planning should also take into account the possible non-determinism of action effects and observation noise. During the task, the maximum range of communication between agents and the amount of data that can be transmitted can vary, or there may be delays in communication. At one extreme, communication is entirely prevented during task execution.

We approach cooperative information gathering as a decentralized partially observable Markov decision process, or Dec-POMDP [7, 37]. The Dec-POMDP is a general model for sequential co-operative decision-making under uncertainty. It models uncertainty in the current state of the system, the effects of actions, and observation noise. Types of inter-agent communication, such as data-rate limited, delayed, or non-existent communication, can be explicitly modelled in the Dec-POMDP framework. This generality makes the Dec-POMDP an ideal choice for formalizing multi-agent information gathering problems.

Specifically, a Dec-POMDP models a sequential multi-agent decision making task. There is an underlying hidden system state, and a set of agents. Each agent has its own set of local actions it can execute, and a set of local observations it may perceive. Markovian state transition and observation processes conditioned on the agents' actions and the state determine the relative likelihoods of subsequent states and observations. A reward function determines the utility of executing any action in any state. In a Dec-POMDP, no implicit communication or information sharing between the agents during task execution is assumed. Communication may be explicitly modelled via the actions and observations. Each agent acts independently, without necessarily knowing what the other agents have perceived or how they have acted. The objective in a Dec-POMDP is to plan optimal local policies for each agent that maximize the expected sum of rewards over a finite horizon of time.

A decentralized information gathering task differs from other multi-agent control tasks by the lack of a goal state. It is not the purpose of the agents to execute actions that reach a particular state, but rather to observe the environment in a manner that provides the greatest amount of information while satisfying operational constraints. As the objective is information acquisition, the reward function depends on the joint belief of the agents.

In contrast to most prior work on Dec-POMDPs, in this paper we consider reward functions that are non-linear functions of the joint belief state. Convex functions of a probability

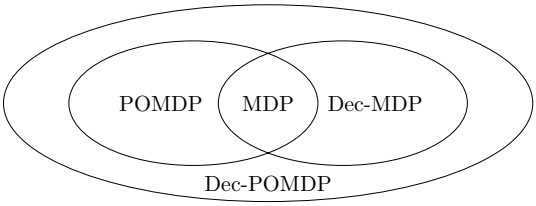


Fig. 1 Relationships of Markovian decision processes. Dec-ρ POMDPs investigated in this article are Dec-POMDPs with a reward that is a convex function of the joint belief state

mass function naturally model certainty [13], and have been proposed before in the context of single-agent POMDPs [3] and Dec-POMDPs [27]. However, to the best of our knowledge no heuristic or approximate algorithms for convex reward Dec-POMDPs have been proposed, and no theoretical results on the properties of such Dec-POMDPs exist in the literature.

1.1 Contributions

Information gathering in decentralized POMDPs (Dec-POMDPs) remains very much an unexplored topic in multi-agent research. According to our knowledge, the only paper prior to ours on information gathering Dec-POMDPs is [27] (same first author as in this paper). Linear reward Dec-POMDPs are solved in several works [7, 14, 19, 29, 33, 35, 38, 40, 42, 47] and Spaan et. al. gather information in fully centralized multi-agent POMDPs [50]. Our work in contrast focuses on Dec-POMDPs with a non-linear reward function. Information gathering for single-agent POMDPs is formalized by Araya-López et. al. [3] using a convex reward function. In this paper, we prove that the value function of information gathering Dec-POMDPs is convex. Lauri et. al. [27] apply the ideas presented in [3] to the Dec-POMDP setting by modifying an existing search tree based Dec-POMDP algorithm. We instead utilize our new proof of value function convexity by combining the idea of iterative improvement of a fixed-size policy represented as a graph [42], and reasoning about reachability of the graph nodes in a manner methodologically similar to the plan-time sufficient statistics of [35]. In experiments, our algorithm solves problems an order of magnitude larger than prior state-of-the-art. The Dec-POMDP generalizes other decision-making formalisms such as multi-agent POMDPs and Dec-MDPs [7]. Thus, our results also apply to these special cases, as illustrated in Fig. 1.

Specifically, the contributions of our article are:

- We prove that in Dec-POMDPs where the reward is a convex function of the joint belief, the value function of any finite horizon policy is convex in the joint belief.
- We propose the first heuristic anytime algorithm for Dec-POMDPs with a reward that is a function of the agents' joint state information. The algorithm is based on iterative improvement of the value of fixed-size policy graphs. We derive a lower bound that may be improved instead of the exact value, leading to computational speed-ups. We also propose an extension of the algorithm for problems with a continuous state space and finite action and observation spaces.
- We experimentally verify the feasibility and usefulness of our algorithm in Dec-POMDPs with non-linear rewards, in domains such as decentralized target tracking, scientific survey planning, and signal source localization.

This article is an extended version of our earlier conference paper [28] published in the Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS) 2019. The extensions compared to the conference paper are the following:

- The conference paper is restricted to discrete small state spaces. To improve the applicability of the proposed approach, we extend the algorithm to large discrete and continuous state spaces by utilizing particle filtering.
- We extend the experimental evaluation with a new benchmark task motivated by signal source localization and mobile robotics. In the benchmark task, the agents can move from a graph node to another and receive observations of a signal source depending on the distance to the source. For the discrete domains, we provide new data for the anytime performance of our proposed algorithm.
- We extend the related work section, and describe the algorithm in more detail. In particular, we clarify the backward pass. We also provide a complexity analysis of the proposed algorithm.

1.2 Outline

The article is organized as follows. We review related work in Sect. 2. In Sect. 3, we define the Dec-POMDP problem we consider and introduce notation and definitions. Section 4 derives the value of a policy graph node. In Sect. 5, we prove convexity of the value in a Dec-POMDP where the reward is a convex function of the state information. Section 6 introduces our policy improvement algorithm, while in Sect. 7 we present its extension to continuous-state problems with finite action and observation spaces. Experimental results are presented in Sect. 8, and concluding remarks are provided in Sect. 9.

2 Related work

In a wide range of applications, autonomous systems have controllable sensors or the ability to otherwise control data acquisition. This allows active information gathering and planned allocation of sensing resources. Active information gathering is known by several synonyms or near-synonyms depending on the context and subfield, such as sensor management [20] or active perception [6]. In the following, we first review approaches based on other models than Dec-POMDPs for multi-agent active information gathering. Secondly, we review the state-of-the-art in Dec-POMDPs and highlight the differences to the present paper.

2.1 Multi-agent active information gathering

In the following paragraphs, we review related work in multi-agent active information gathering in three major directions: distributed constraint optimization problems, application of sequential greedy maximization of submodular functions, and adaptation of single-agent POMDP methods to the multi-agent setting. Finally, we briefly review other related techniques such as applying open loop planning, gradient ascent based control, stochastic control under simplifying assumptions, and active distributed hypothesis testing.

A detailed review of the solution techniques for each of these related approaches is beyond the scope of the present paper. We refer the reader to relevant literature on the

respective topics, and instead focus on describing applications and modelling assumptions and contrasting them to the Dec-POMDP approach to multi-agent active information gathering. Each of the alternative approaches we review makes some simplifying assumptions about the active information gathering problem in contrast to Dec-POMDP based planning. For instance, action effects may be assumed to have known outcomes. The system state may be assumed to be static, or perfectly known to the agents, or it may be assumed to evolve deterministically. Finally, implicit local communication is often assumed to be possible, whereas in the Dec-POMDP communication is allowed only if explicitly modelled.

Distributed constraint optimization problems Information gathering can be framed as a distributed constraint optimization problem (DCOP). Instead of trying to find a policy for each agent that maximizes total expected information gain in a stochastic dynamic system over time, the standard DCOP finds a variable assignment for each agent such that, for example, sensors cover a large area. Compared to a full Dec-POMDP solution a DCOP approach is computationally less intensive but usually requires stronger assumptions or approximations to the underlying problem. There are also extensions to basic DCOP; we discuss DCOPs and these extensions in the context of information gathering next.

Distributed constraint optimization problems (DCOPs) are described by a set of agents and a set of variables along with a set of cost functions. Each variable is assigned to one of the agents. Each of the cost functions is defined on a subset of the variables it is affected by. The cost functions are real-valued, although an additional special value can indicate a violated constraint. A solution of a DCOP is an assignment of all variables that does not violate any of the constraints, with each agent only choosing the assignment of variables assigned to it. The objective is to find a solution that minimizes the sum of the individual cost functions such that no constraints are violated. With regards to communication, a typical setting in DCOPs is that each agent can communicate locally with its neighbouring agents to coordinate actions. The most relevant variants of DCOPs for active information gathering are dynamic DCOPs and probabilistic DCOPs [54], which we discuss next. For a detailed introduction to DCOPs and their solution algorithms, we refer the reader to the recent survey [16].

A dynamic DCOP (D-DCOP) extends the variables, assignments, and cost functions to depend on a discrete time step. A D-DCOP may be viewed as a sequence of DCOP problems, one for each time step. The DCOP at the current time step is assumed to be known by the agents, however the agents are unaware how the DCOP will evolve over future time steps. Zivan et. al. [57] present a D-DCOP variant for controlling a mobile sensor team tracking multiple targets. Each sensor is an agent that has a perfectly observable variable describing its location which may evolve locally over time. Each target is represented by a cost function whose value depends on how well the sensors cover the target. The coverage of a target is measured by a monotonic function of the number of agents within sensing range of the target, and no explicit probabilistic modelling of the sensing process is undertaken.

Markovian D-DCOPs [34] augment D-DCOPs by introducing an underlying state with Markovian dynamics depending on the variable assignments. The cost functions are dependent on the state that is fully observable. Unlike in the D-DCOP, changes in the Markovian D-DCOP are restricted to changes in the underlying state. The Markovian D-DCOP is demonstrated in a multi-sensor tracking environment, where the target state is described using the Markovian dynamics and the variable assignments correspond to assignments of sensors to targets.

Probabilistic DCOPs (P-DCOP) contain two types of variables: decision variables and random variables. Decision variables are analogous to variables in the DCOP, with

assignments selected by the agents. Random variables model events beyond the agents' control and assume values according to a specified probability distribution. In the P-DCOP actions have known outcomes. Uncertainty in environment dynamics is modelled by defining stochastic cost functions. The cost functions may depend on decision variables and random variables. The objective is to find an assignment of all decision variables such that, e.g., the expected utility of the assignments under the probability distribution of the random variables is maximized. P-DCOPs with partial agent knowledge [22, 53] are a variant especially relevant for active information gathering. They introduce a finite time horizon to the P-DCOP, and a solution is an assignment of decision variables for each time step. The objective is to maximize the cumulative utility over the time horizon. At each time step the agents acquire information about the cost functions and the available utility through exploration. P-DCOPs have been applied to a mobile wireless network problem, where the objective is to choose agent locations to maximize the signal strength of an ad-hoc communication relay network formed between the agents [22, 53].

Sequential greedy maximization of submodular functions Submodularity is a property of a set function often used for proving a bound on the suboptimality of greedy algorithms in information gathering. The tasks that we target in this paper are in general not approximately solvable by a greedy algorithm. Below we discuss how information gathering problems are formulated to allow use of greedy submodular maximization with suboptimality bounds. This will help the reader build a deeper understanding of the challenges this paper addresses.

The assignment or selection of sensors can be formulated as a selection of a subset among a larger set of possible choices. The selection corresponds to a choice of which sensors to apply, where to deploy sensors, or how to operate multiple sensors. The utility of an assignment is then measured by a set function that maps the selected subset to a real-valued utility. Several set functions relevant for active information gathering such as mutual information and entropy are submodular. Informally, the submodularity property of a set function encodes the intuition that the marginal benefit of deploying a new sensor is reduced as we deploy more and more sensors. A sequential greedy algorithm is often applied to approximately maximize submodular functions. Such an algorithm sequentially finds the item with the greatest marginal utility, and inserts it into the current subset of selected items. The selection is repeated until a subset with the required number of items has been obtained. Maximization of a submodular set function by this greedy approach results in an approximation within at most a factor of $(1 - 1/e)$ from an optimal solution. Consequently, this selection method has been applied to optimize sensor placements under a Gaussian process model of spatial phenomena [24]. In this approach, mutual information between the selected sensing locations and the rest of the space is maximized.

In multi-agent active information gathering, submodular function maximization under a matroid constraint is especially relevant. A matroid is a mathematical structure that generalizes the notion of linear independence from vector spaces to sets. In multi-agent active information gathering, the independent sets in a matroid correspond to possible joint actions or sensing strategies of each of the agents in a team. Approximation guarantees for the sequential greedy algorithm outlined above can also be proven for submodular maximization under a matroid constraint [10]. A distributed algorithm based on greedy maximization for multi-robot exploration is proposed in [12]. As the proposed algorithm runs in parallel on all of the agents, the required time of planning is reduced compared to sequential planning for each agent in turn, e.g., as proposed in [4]. To simplify the problem, Corah and Michael [12] plan over sequences of actions rather than over closed-loop

policies. Contrary to a Dec-POMDP model, a fully connected communication network and a shared belief state among all agents are assumed.

In distributed submodular optimization under a matroid constraint, the information an agent has about the actions of other agents affects the approximation quality obtained with the greedy algorithm. Gharesifard and Smith [17] assume each agent knows the strategies of its neighbouring agents, as defined by a communication graph. A local greedy algorithm is found to approximate an optimal solution within a factor inversely proportional to the clique number of the communication graph: the larger the maximum clique in the graph, the better the approximation factor.

Adapting single-agent POMDPs Single-agent POMDPs feature imperfect knowledge of the underlying true state in a system, uncertain action effects and noisy observations. If instantaneous communication and centralized control are assumed, multi-agent active perception may be treated as a POMDP under the control of a central controller that determines the actions of all individual agents [45, 50].

When strong centralization assumptions are infeasible, POMDP techniques may be adapted to multi-agent active information gathering by adding additional coordination mechanisms. A notable example is [11], where a decentralization scheme based on an auction mechanism is proposed. Each agent is assumed to play one of a finite set of roles or behaviours. Each role corresponds to a specific local reward function that depends on the local state and action of the agent playing the role. The complete reward function is defined as the sum of local reward functions and a joint reward function modelling agent cooperation. At run-time, each individual agent first solves a centralized multi-agent POMDP planning problem to compute a policy and its value for each of the potential roles. An auction algorithm is applied for task allocation where the cost of assigning a policy to an agent is equal to the negative value of the policy, resulting in high-value policies likely being assigned to each agent. The resulting approach requires some communication between the agents to facilitate the bidding, but can avoid the computational complexity of planning in a Dec-POMDP. The approach is demonstrated in two scenarios, environmental monitoring and cooperative tracking.

Other related work Multi-agent information gathering may be formulated as an open-loop planning problem, where the solution is a sequence of actions or a path to be traversed for each agent. These approaches ignore the adaptivity of Dec-POMDP solutions that allow agents to modify their behaviour conditional on the observations received at run-time. Instead, if communication is possible at some time after starting execution of the paths, replanning may be applied to update the solution based on the information acquired up to that time. An example of decentralized open-loop planning with periodic communication is the decentralized Monte Carlo tree search (Dec-MCTS) algorithm [9]. The agents are first provided with initial information of each others' plans. Each agent in a team runs an instance of MCTS to optimize its local actions, assuming the other agents will act according to their last known plan. The updated plans found via local MCTS instances are periodically communicated between the agents, and planning is restarted. Hollinger and Singh [21] formalize multi-agent exploration as path planning in a time-expanded graph. Instead of assuming communication links exist between agents, feasible paths are constrained in a way that agents will periodically be within communication range. While in communication range, the agents share information and may replan to update the subsequent paths.

Several works propose to derive a control policy for each individual agent that follows the gradient of mutual information [5, 23]. Joint belief states are updated via agent-to-agent communication. Communication links are described by a graph, and convergence of the state estimate to the true value is shown in the case of a connected graph.

The problem of planning policies for multi-agent active information gathering can be simplified if assumptions are made regarding the dynamics of the state transitions and the observation processes of the agents. Schlotfeldt et. al. [46] investigate a problem with a deterministic state transition model with a linear-Gaussian model of the observation process. The joint belief state in this case is Gaussian and may be tracked by a Kalman filter (see, for example, [44, Sect. 4.3]). In particular, the covariance matrix of the Gaussian is independent of the actual measurement values recorded. The covariance matrix only depends on the deterministic state trajectory of the agent. This allows planning a policy for active information gathering by solving an open loop control problem, analogous to similar approaches in classical control theory where a measurement subsystem can be controlled independently of the overall plant control task [30]. Schlotfeldt et. al. [46] search policies for each agent sequentially, assuming the other agents' plans are fixed. A distributed estimation scheme is proposed to enable decentralization, requiring a connected communication graph to guarantee state estimate convergence.

Distributed hypothesis testing [26] targets a scenario where a stationary hidden state (hypothesis) should be inferred from noisy observations recorded by a team of agents. The agents perform local Bayesian belief updates, and communicate the updates to neighbouring agents. A consensus algorithm is applied to fuse the updates from neighbours into an updated local belief with convergence guarantees. Active distributed hypothesis testing where each agent can choose among a finite set of sensing actions is considered by Lalitha and Javidi [25]. They characterize randomized action strategies that ensure that the maximum likelihood estimate of the hypothesis converges to the true hypothesis.

2.2 Decentralized POMDPs

In this subsection, we first provide a brief summary of the state-of-the-art in Dec-POMDPs with linear reward functions. The expected value of a reward function that depends on the hidden state and action is a linear function of the joint belief. These types of rewards are standard in Dec-POMDPs. Then, we describe related work in active information gathering in single-agent POMDPs and Dec-POMDPs that are especially relevant for the work presented in this paper. These works target problems with a reward function that is convex in the joint belief, allowing convenient modelling of information gathering tasks.

Dec-POMDPs with linear reward functions The computational complexity of finding an optimal decentralized policy for a finite-horizon Dec-POMDP is NEXP-complete [7], that is, double-exponential w.r.t. the planning horizon. It is also NEXP-complete to compute solutions with an absolutely bounded error [43].

Exact algorithms for Dec-POMDPs may apply either backwards in time dynamic programming [19], or forwards in time heuristic search [38, 52]. It can also be shown that a Dec-POMDP is equivalent to a special case of a POMDP that is completely unobservable [14, 29, 36]. The state space of this POMDP is the set of possible plan-time sufficient statistics [35], which are joint distributions over the hidden state and the histories of the agents' actions and observations given the past policies executed by the agents. The actions correspond to selecting the next decision rules for the agents. An optimal policy for such a non-observable POMDP is an optimal sequence of local decision rules for each agent, which corresponds to an optimal solution of the equivalent Dec-POMDP. This insight has allowed adaptation of POMDP algorithms for solving Dec-POMDPs [14].

Approximate and heuristic methods for solving Dec-POMDPs have been proposed, e.g., based on finding locally optimal "best response" policies for each agent [33], memory-bounded dynamic programming [47], cross-entropy optimization over the space of policies [40, 41], or monotone iterative improvement of fixed-size policies [42]. Algorithms for special cases such as goal-achievement Dec-POMDPs [2] and factored Dec-POMDPs [39] have also been proposed.

Structural properties, such as transition, observation, and reward independence between the agents, can also be leveraged and may even result in a problem with a lesser computational complexity [1]. Some Dec-POMDP algorithms [38] take advantage of plan-time sufficient statistics. The sufficient statistics provide a means to reason about possible distributions over the hidden state, also called joint beliefs, reached under a given policy.

No implicit communication between the agents is assumed in the Dec-POMDP framework. However, communication may be explicitly included into the Dec-POMDP model if desired, leading to the so-called Dec-POMDP-Com model [18]. Spaan et. al. [49] also include communication into their Dec-POMDP model but in addition split state features into local and shared global ones making belief tracking, as in POMDP models, possible. In another line of work, Wu et. al. [56] investigate online decision making in Dec-POMDPs. Their goal is to reduce the amount of communication needed in online Dec-POMDP planning: agents re-synchronize with other agents whenever the behavior of other agents changes sufficiently from the expected behavior.

Active Information Gathering in POMDPs and Dec-POMDPs In the context of single-agent POMDPs, Araya-López et al. [3] argue that information gathering tasks are naturally formulated using a reward function that is a convex function of the state information and introduce the ρPOMDP model with such a reward. This enables application of, for example, the negative Shannon entropy of the state information as a component of the reward function. Under a locally Lipschitz continuous reward function, an optimal value function of a ρPOMDP is Lipschitz-continuous [15] which may be exploited in a solution algorithm.

Spaan et. al. [50] introduce an alternative formulation for information gathering in single agent POMDPs that assumes problem specific state features and for each state feature a special action for committing to a specific value of the state feature. This allows for rewarding information gathering without changing the POMDP optimization process at the cost of growing the action space and the need for selecting features. Subsequently, it was proven that a ρPOMDP with a piecewise linear and convex reward function can be transformed into a standard POMDP with a linear reward function [45]. The key to making the transformation is to augment the action space of the POMDP by adding a prediction action for each of the linear components of the ρPOMDP reward function. The reward function for the new prediction actions is set to return the reward of the corresponding component of the ρPOMDP reward function. The conversion also allows an optimal solution for either problem to be transformed into an optimal solution of the other problem in polynomial time.

In this article, we present the first heuristic algorithm for Dec-POMDPs with rewards that depend non-linearly on the joint belief. Our algorithm is based on the combination of the idea of using a fixed-size policy represented as a graph [42] with plan-time sufficient statistics [35] to determine joint beliefs at the policy graph nodes. The local policy at each policy graph node is then iteratively improved, monotonically improving the value of the node. We show that if the reward function is convex in the joint belief, then the value

function of any finite-horizon Dec-POMDP policy is convex as well. This is a generalization of a similar result known for single-agent POMDPs [3]. Lauri et. al. [27] directly apply the ideas presented in [3] to the Dec-POMDP setting by modifying an existing search tree based Dec-POMDP algorithm that can solve small problems. However, our new approach allows us to derive a lower bound for the value of a policy. We empirically show that an algorithm maximizing this lower bound finds high quality solutions. Thus, compared to prior state-of-the-art in Dec-POMDPs with convex rewards [27], our algorithm is capable of handling problems an order of magnitude larger.

We do not require any communication between agents during task execution, and do not make any distributional assumptions about the state transition or observation models.

3 Decentralized POMDPs

We next formally define the Dec-POMDP problem we consider. Contrary to most earlier works, we define the reward as a function of *state information* and action. This allows us to model information acquisition problems. We choose the finite-horizon formulation to reflect the fact that a decentralized information gathering task should have a clearly defined end after which the collected information is pooled and subsequent inference or decisions are made.

A finite-horizon Dec-POMDP is a tuple $(T, I, S, \{A_i\}, \{Z_i\}, P^s, P^z, b^0, \{\rho_t\})$, where

- $T \in \mathbb{N}$ is the problem time horizon. In other words, time steps $t = 0, 1, \dots, T$ are considered in the problem.
- $I = \{1, 2, \dots, n\}$ is a set of agents.
- S is a finite set of hidden states. We write s^t for a state at time t.
- $\{A_i\}$ and $\{Z_i\}$ are the collections of finite local action and local observation sets of each agent $i \in I$, respectively. The local action and observation of agent i at time t are written as $a_i^t \in A_i$ and $z_i^t \in Z_i$, respectively. We write as A and Z the Cartesian products of all A_i or Z_i, respectively. Furthermore, the joint action and observation at time t are written as $a^t = (a_1^t, \dots, a_n^t) \in A$ and $z^{t+1} = (z_1^{t+1}, \dots, z_n^{t+1}) \in Z$, respectively.
- P^s is the state transition probability that gives the conditional probability $P^s(s^{t+1} \mid s^t, a^t)$ of the new state s^{t+1} given the current state s^t and joint action a^t.
- P^z is the observation probability that gives the conditional probability $P^z(z^{t+1} \mid s^{t+1}, a^t)$ of the joint observation z^{t+1} given the state s^{t+1} and previous joint action a^t.
- $b^0 \in \Delta(S)$ is the initial state distribution,[1] also called the joint belief, at time $t = 0$.
- $\rho_t : \Delta(S) \times A \to \mathbb{R}$ are the reward functions at times $t = 0, \dots, T-1$, while $\rho_T : \Delta(S) \to \mathbb{R}$ determines a final reward obtained at the end of the problem horizon.

Most works in Dec-POMDPs up to date define state-dependent reward functions of the form $R_t : S \times A \to \mathbb{R}$ and $R_T : S \to \mathbb{R}$ for time steps $t = 0, 1, \dots, T-1$ and $t = T$, respectively. The Dec-ρPOMDP with belief-dependent reward functions as defined above generalizes a Dec-POMDP with state-dependent rewards. The standard Dec-POMDP with a state-dependent reward function is recovered by choosing the belief-dependent reward to

[1] We denote by $\Delta(S)$ the space of probability mass functions over S.

equal the expected state-dependent reward, that is, $\rho_t(b, a) := \mathbb{E}_{s\sim b}[R_t(s, a)]$. We choose the symbol ρ_t for belief-dependent rewards to distinguish from state-dependent reward functions and to be consistent with [3] that in part inspired this work.

The Dec-POMDP starts from some state $s^0 \sim b^0$. Each agent $i \in I$ then selects a local action a_i^0, and the joint action $a^0 = (a_1^0, \dots, a_n^0)$ is executed. The time step is incremented, and the state transitions according to P^s, and each agent perceives a local observation z_i^1, where the likelihood of the joint observation $z^1 = (z_1^1, \dots, z_n^1)$ is determined according to P^z. The action selection and observation process repeats similarly until $t = T$ when the task ends.

Optimally solving a Dec-POMDP means to design a policy for each agent that encodes which action the agent should execute conditional on its past observations and actions; in a manner such that the expected sum of rewards collected is maximized. In the following, we make the notion of a policy exact, and determine the expected sum of rewards collected when executing a policy.

3.1 Histories and policies

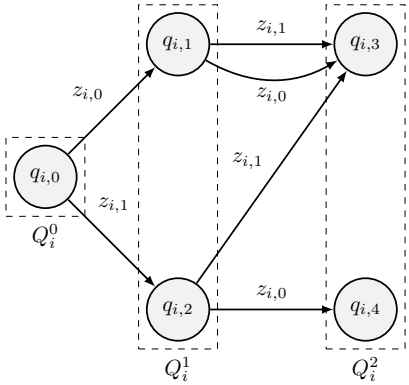
At any time step while the agents are executing actions in a Dec-POMDP, each agent only has knowledge of its own past local actions and observations and the initial state distribution b^0. Each agent may decide on its next local action based on this knowledge. We formalize the information available to agent i at time t as a local history. At the starting time $t = 0$ the information available to any agent i is completely described by a local history $h_i^0 = b^0$. On subsequent time steps $t \geq 1$, the local history h_i^t of any agent i belongs to the local history set

$$H_i^t = \{(b^0, a_i^0, z_i^1, \dots, a_i^{t-1}, z_i^t) \mid \forall k : a_i^k \in A_i, z_i^k \in Z_i\}.$$

Analogously, we define the joint history set $H^t = \{(b^0, a^0, z^1, \dots, a^{t-1}, z^t) \mid \forall k : a^k \in A, z^k \in Z\}$ for joint actions and observations. Given a joint history h^t, we may view its composite local histories as the tuple $(h_1^t, \dots, h_n^t)$, or vice versa. Both the local and joint histories satisfy the recursion of the form $h^t = (h^{t-1}, a^{t-1}, z^t)$.

In general, a local policy for agent i is a set of mappings $H_i^t \mapsto A_i$, one for every $t = 0, 1, \dots, T-1$. Such a local policy determines the next action the agent should take conditional on any possible local history up to the current time. The fact that local policies only depend on an agent's local actions and observations is a key feature of Dec-POMDPs which ensures that an agent can execute its local policy in a decentralized manner, without knowledge about the other agents' actions and observations. A solution of a Dec-POMDP is in turn a joint policy, which is the collection of all agents' local policies. An optimal solution is a joint policy that maximizes the expected sum of rewards obtained when each agents acts according to its local policy contained in the joint policy.

We define a local policy as a particular kind of finite state controller (FSC) following [42]. This choice allows a compact representation of policies by viewing them as FSCs with few nodes. Furthermore, as we will argue later, increasing the number of nodes allows retaining the generality of the definition given in the paragraph above.



Node q_i	Action $\gamma_i(q_i)$
$q_{i,0}$	$a_{i,0}$
$q_{i,1}$	$a_{i,1}$
$q_{i,2}$	$a_{i,2}$
$q_{i,3}$	$a_{i,1}$
$q_{i,4}$	$a_{i,0}$

Fig. 2 A local policy $\pi_i = (Q_i, q_{i,0}, \gamma_i, \lambda_i)$ for agent i with local action space $A_i = \{a_{i,0}, a_{i,1}, a_{i,2}\}$ and local observation space $Z_i = \{z_{i,0}, z_{i,1}\}$ represented as a directed acyclic graph. The shaded circles show the set of nodes Q_i, the starting node is $q_{i,0}$. The dashed boxes indicate the subsets $Q_i^t \subset Q_i$ of nodes reachable at time t. The table on the right shows the output function γ_i that determines the action to take in any node. The node transition function λ_i is indicated by the out-edges from each node: for example, $\lambda_i(q_{i,0}, z_{i,0}) = q_{i,1}$. Execution of the policy is equivalent to traversal of the graph starting from the initial node, taking actions indicated by γ_i, and transitioning to the next node conditional on the next observation according to λ_i

Definition 1 *(Local policy)* For agent i, a local policy is $\pi_i = (Q_i, q_{i,0}, \gamma_i, \lambda_i)$, where Q_i is a finite set of nodes, $q_{i,0} \in Q_i$ is a starting node, $\gamma_i : Q_i \to A_i$ is an output function that determines which action to take, and $\lambda_i : Q_i \times Z_i \to Q_i$ is a node transition function that determines the next node conditional on the last observation.

This definition coincides with that of a Moore machine [31]. Another useful way to view a local policy is as a directed acyclic graph, where policy execution is equivalent to graph traversal starting from the specified starting node. Figure 2 shows an example of such a local policy graph, along with a description of how the local policy is executed. The depicted local policy is suitable for a problem with a horizon $T = 3$.

An arbitrary node may be reached by zero (in case the node has no in-edges), one, or more than one local histories. This will be important in our subsequent analysis of the problem and the proposed solution algorithm. For example, in the case of Fig. 2, node $q_{i,3}$ is reachable by three local histories, namely $(b^0, a_{i,0}, z_{i,0}, a_{i,1}, z_{i,0})$, $(b^0, a_{i,0}, z_{i,0}, a_{i,1}, z_{i,1})$, and $(b^0, a_{i,0}, z_{i,1}, a_{i,2}, z_{i,1})$. The first two of these local histories traverse through the nodes $q_{i,0} \to q_{i,1} \to q_{i,3}$, while the last local history traverses through nodes $q_{i,0} \to q_{i,2} \to q_{i,3}$. Although the local histories are different, the action to be taken next after experiencing them is the same, $\gamma_i(q_{i,3}) = a_{i,1}$.

Next, we formally define the concept of a joint policy that is a collection of each agent's local policies.

Definition 2 *(Joint policy)* Given local policies $\pi_i = (Q_i, q_{i,0}, \gamma_i, \lambda_i)$ for all agents $i \in I$, the corresponding joint policy is the tuple $\pi = (Q, q_0, \gamma, \lambda)$, where Q is the Cartesian product of all Q_i, $q_0 = (q_{1,0}, \ldots, q_{n,0}) \in Q$ is the initial node, and for an arbitrary joint policy node $q = (q_1, \ldots, q_n) \in Q$ and joint observation $z = (z_1, \ldots, z_n) \in Z$, the output function $\gamma : Q \to A$ determines the joint action to take in a node according to $\gamma(q) = (\gamma_1(q_1), \ldots, \gamma_n(q_n))$, and the node transition function $\lambda : Q \times Z \to Q$ is defined such that $\lambda(q, z) = (\lambda_1(q_1, z_1), \ldots, \lambda_n(q_n, z_n))$.

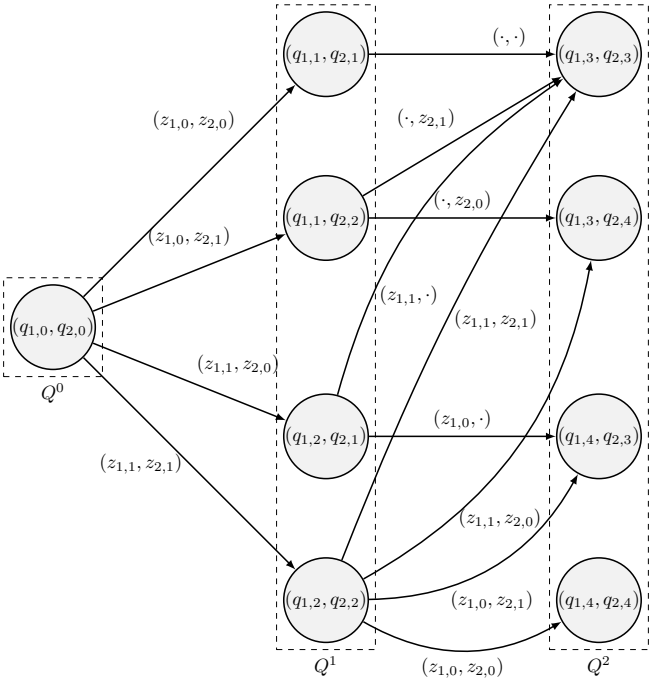


Fig. 3 A joint policy $\pi = (Q, q_0, \gamma, \lambda)$ composed of two local policies π_1 and π_2 that are identical to the one shown in Fig. 2. For clarity, we denote existence of parallel edges by single edges with labels such as $(\cdot, z_{2,1})$, which indicates there are parallel edges for joint observations where the local observation of agent 1 can assume any value, and the local observation of agent 2 is $z_{2,1}$

Figure 3 shows an example of a joint policy constructed from two local policies using the definition above. The nodes $q \in Q$ in the joint policy graph are tuples of nodes in the local policy graphs: $Q = Q_1 \times Q_2$. For example, the node $q = (q_{1,3}, q_{2,3})$ indicates that both agents i are currently executing their respective local policies π_i at node $q_{i,3}$. The dashed boxes indicate the subsets $Q^t \subset Q$ of nodes reachable at time t. The node output function γ and node transition function λ of the joint policy are constructed from the corresponding function γ_i and λ_i of the local policies, respectively, as described in Definition 2. From each node $q \in Q$, there is now an out edge for each *joint* observation $z \in Z$. As is the case for local policies, also in a joint policy a given node may be reached by zero, one, or more than one joint history. Since a joint policy is constructed from local policies that are decentrally executable, the joint policy is also decentrally executable.

The generality of the policy representation is maintained, as any finite horizon local policy can be represented by a local policy graph with sufficiently many nodes. If the size of the local policy graph is grown sufficiently a graph where each node is reachable by a single unique local history can be created. Such a policy graph is in fact a tree and can represent any possible mapping from local histories to local actions and thus any possible local policy. A joint policy composed of such trees is a so-called pure deterministic policy for a Dec-POMDP in the sense that an agent's local observation history is sufficient to determine the agent's next action. It is known that there exists an optimal pure policy in

every Dec-POMDP [40]. Since a tree can represent an optimal policy, the policy graph representation we choose can also represent an optimal policy given enough nodes. Consequently, our main theoretical results hold in the general case, and are not limited to our particular policy representation. Our emphasis in this paper is on compact local policies that are not trees and are not guaranteed to be able to represent an optimal policy. We refer the reader to [40] for further discussion of possible types of policies for Dec-POMDPs.

To close this subsection, we give a technical condition for policy graphs that constrains the structure of local policies. The condition, called temporal consistency, ensures that each node can be identified with a unique time step.

Definition 3 *(Temporal consistency)* A local policy $\pi_i = (Q_i, q_{i,0}, \gamma_i, \lambda_i)$ is temporally consistent if there exists a partition $Q_i = \bigcup_{t=0}^{T-1} Q_i^t$ where Q_i^t are pairwise disjoint and non-empty, $Q_i^0 = \{q_{i,0}\}$, and for any $t = 0, \ldots, T-2$, for $q_i^t \in Q_i^t$, for all $z_i \in Z_i$, $\lambda_i(q_i^t, z_i) \in Q_i^{t+1}$.

In a temporally consistent policy, at a node in Q_i^t the agent has $(T-t)$ decisions left until the end of the problem horizon. Temporal consistency guarantees that exactly one node in each set Q_i^t can be visited, and that after visiting a node in Q_i^t, the next node will belong to Q_i^{t+1}. Temporal consistency naturally extends to joint policies, such that there exists a partition of Q by pairwise disjoint sets Q^t. In Figs. 2 and 3, the sets Q_i^t and Q^t are indicated by the dashed boxes. Throughout the rest of the article, we assume temporal consistency holds for all policies. The assumption of temporal consistency does not restrict our proposed method, it merely allows us to refer to a subset of nodes reachable at a particular time step.

3.2 Bayes filter

While planning policies for information gathering, it is useful to reason about the joint belief of the agents given some joint history. This can be done via recursive Bayesian filtering, see [44] for a general overview of the topic. Recursive Bayesian filtering is a process by which the probability mass function over the hidden state of the system is estimated given a joint history and the state transition and observation probability models of the Dec-POMDP. Recall that during policy execution agents only perceive their own local actions and observations. Thus, Bayesian filtering typically cannot be achieved online by an individual agent as it lacks the necessary information. However, if the local policies of each agent are planned centrally as is the usual case in Dec-POMDPs, Bayesian filtering may be applied during planning to reason about the possible joint beliefs the system may reach.

We now describe the discrete Bayesian filter we use to track the joint belief. The initial joint belief b^0 is a function of the state at time $t = 0$, and for any state $s^0 \in S$, $b^0(s^0)$ is equal to the probability $P(s^0 \mid h^0)$. When a joint action a^0 is executed and a joint observation z^1 is perceived, we find the posterior belief $P(s^1 \mid h^1)$ where $h^1 = (h^0, a^0, z^1)$ applying a Bayes filter. For notational convenience, we drop the explicit dependence of b^t on the joint history in the following. In general, given any current joint belief b^t corresponding to some joint history h^t, and a joint action a^t and joint observation z^{t+1}, the posterior joint belief is calculated by

$$b^{t+1}(s^{t+1}) = \frac{P^z(z^{t+1} \mid s^{t+1}, a^t) \sum_{s^t \in S} P^s(s^{t+1} \mid a^t, s^t) b^t(s^t)}{\eta(z^{t+1} \mid b^t, a^t)}, \tag{1}$$

where

$$\eta(z^{t+1} \mid b^t, a^t) = \sum_{s^{t+1} \in S} P^z(z^{t+1} \mid s^{t+1}, a^t) \sum_{s^t \in S} P^s(s^{t+1} \mid a^t, s^t) b^t(s^t) \tag{2}$$

is the normalization factor equal to the prior probability of observing z^{t+1}. Given b^0 and any joint history $h^t = (b^0, a^0, z^1, \cdots, a^{t-1}, z^t)$, repeatedly applying Eq. (1) yields a sequence $b^0, b^1, \ldots, b^t$ of joint beliefs. We shall denote the application of the Bayes filter (Eq. (1)) by the shorthand notation

$$b^{t+1} = \zeta(b^t, a^t, z^{t+1}). \tag{3}$$

Furthermore, we shall denote the filter that recovers b^t given h^t by repeated application of ζ by a function $\tau : H^t \to \Delta(S)$, that is,

$$b^t = \tau(h^t) = \underbrace{\zeta(\zeta(\ldots \zeta(\zeta}_{t \text{ times}}(b^0, a^0, z^1), a^1, z^2) \ldots)). \tag{4}$$

The innermost application of ζ recovers b^1 given b^0, a^0, and z^1. The output is the input to the next application of ζ together with a^1 and z^2. This process is repeated until the tth application of ζ which outputs b^t.

3.3 Value of a policy

The value of a joint policy $\pi = (Q, q_0, \gamma, \lambda)$ is equal to the expected sum of rewards collected when acting according to the policy, starting from the initial joint belief. To characterize the value of a policy, we define value functions $V_t^\pi : \Delta(S) \times Q^t \to \mathbb{R}$ for $t = T, T-1, \ldots, 0$ using a backwards in time dynamic programming principle. Each $V_t^\pi(b, q)$ gives the expected sum of rewards when following policy π until the end of the horizon when t decisions have been taken so far, for any joint belief $b \in \Delta(S)$ and any policy node $q \in Q^t$. Then, $V_0^\pi(b^0, q_0)$ is the value of the policy.

We start with time step $t = T$ which is a special case when all actions have already been taken, and the value function only depends on the joint belief and is equal to the final reward: $V_T(b) = \rho_T(b)$.

For $t = T - 1$, one decision remains, and the remaining expected sum of rewards of executing policy π is equal to

$$V_{T-1}^\pi(b, q) = \rho_{T-1}(b, \gamma(q)) + \sum_{z \in Z} \eta(z \mid b, \gamma(q)) V_T(\zeta(b, \gamma(q), z)), \tag{5}$$

that is, the sum of the immediate reward and the expected final reward at time T. From above, we define V_t^π iterating backwards in time for $t = T-2, \ldots, 0$ as

$$V_t^\pi(b, q) = \rho_t(b, \gamma(q)) + \sum_{z \in Z} \eta(z \mid b, \gamma(q)) V_{t+1}^\pi(\zeta(b, \gamma(q), z), \lambda(q, z)). \tag{6}$$

The objective is to find an optimal policy $\pi^* \in \operatorname{argmax}_\pi V_0^\pi(b^0, q_0)$ whose value is greater than or equal to the value of any other policy.

4 Value of a policy node

Executing a policy corresponds to a stochastic traversal of the policy graphs (Fig. 2) conditional on the observations perceived. In this section, we first answer two questions related to this traversal process. First, given a history, when is it consistent with a policy, and which nodes in the policy graph will be traversed (Sect. 4.1)? Second, given an initial state distribution, what is the probability of reaching a given policy graph node, and what are the relative likelihoods of histories if we assume a given node is reached (Sect. 4.2)? With the above questions answered, we define the value of a policy graph node both in a joint and in a local policy (Sect. 4.3). These values will be useful in designing a policy improvement algorithm for Dec-POMDPs. Probabilities of joint histories conditioned on the local information of one agent have been derived earlier for reasoning what one agent can know about the experiences of other agents [33]. Oliehoek [35] introduces plan-time sufficient statistics describing the joint distribution over hidden states and joint observation histories. Our derivation here is methodologically similar, but we consider explicitly the setting where a policy is represented as a graph and reason about reachability probability of a policy graph node that summarizes multiple joint histories, rather than the reachability probability of a particular joint history.

4.1 History consistency

A history is consistent with a policy when executing the policy could have resulted in the given history. In the context of policy graphs defined in Sect. 3.1, consistency means that the actions in the history are equal to those that would be taken by the policy conditional on the observations in the history.

Definition 4 *(History consistency)* We are given for all $i \in I$ the local policy $\pi_i = (Q_i, q_{i,0}, \gamma_i, \lambda_i)$, and the corresponding joint policy $\pi = (Q, q_0, \gamma, \lambda)$.

1. A local history $h_i^t = (b_0, a_i^0, z_i^1, \ldots, a_i^{t-1}, z_i^t)$ is consistent with π if the sequence of nodes $(q_i^0, q_i^1, \ldots, q_i^t)$ where $q_i^0 = q_{i,0}$ is the initial node and $q_i^k = \lambda_i(q_i^{k-1}, z_i^k)$ for $k = 1, \ldots, t$ satisfies: $a_i^k = \gamma_i(q_i^k)$ for every k. We say h_i^t ends at $q_i^t \in Q_i^t$ under π.
2. A joint history $h^t = (h_1^t, \ldots, h_n^t)$ is consistent with π if for all $i \in I$, h_i^t is consistent with π and ends at q_i^t. We say h^t ends at $q^t = (q_1^t, \ldots, q_n^t) \in Q^t$ under π.

Due to temporal consistency, any $h_i^t \in H_i^t$ consistent with a policy will end at some $q_i^t \in Q_i^t$. Similarly, any $h^t \in H^t$ ends at some $q^t \in Q^t$.

4.2 Node reachability probabilities

Above, we have defined when a history ends at a particular node. Using this definition, we now derive the joint probability mass function (pmf) $P(q^t, h^t \mid \pi)$ of policy nodes and joint histories given that a particular policy π is executed.

We note that $P(q^t, h^t \mid \pi) = P(q^t \mid h^t, \pi) P(h^t \mid \pi)$ and first consider $P(h^t \mid \pi)$. The unconditional a priori probability of experiencing the joint history $h^0 = (b^0)$ is $P(h^0) = 1$. For $t \geq 1$, the unconditional probability of experiencing h^t is obtained recursively by $P(h^t) = \eta(z^t \mid \tau(h^{t-1}), a^{t-1}) P(h^{t-1})$. Conditioning $P(h^t)$ on a policy yields $P(h^t \mid \pi) = P(h^t)$ if

h^t is consistent with π and 0 otherwise. Next, we have $P(q^t \mid h^t, \pi) = \prod_{i \in I} P(q_i^t \mid h_i^t, \pi)$, with $P(q_i^t \mid h_i^t, \pi) = 1$ if h_i^t ends at q_i^t under π and 0 otherwise.

Combining the above, the joint pmf is defined as

$$P(q^t, h^t \mid \pi) = \begin{cases} P(h^t) & \text{if } h^t \text{ ends at } q^t \text{ under } \pi \\ 0 & \text{otherwise} \end{cases}.$$

Marginalizing over h^t, the probability of ending at node q^t under π is

$$P(q^t \mid \pi) = \sum_{h^t \in H^t} P(q^t, h^t \mid \pi), \tag{7}$$

and by definition of conditional probability,

$$P(h^t \mid q^t, \pi) = \frac{P(q^t, h^t \mid \pi)}{P(q^t \mid \pi)}. \tag{8}$$

We now find the probability of ending at q_i^t under π. Let Q_{-i}^t denote the Cartesian product of all Q_j^t *except* Q_i^t, that is,

$$Q_{-i}^t = \prod_{\substack{j \in I \\ j \neq i}} Q_j^t.$$

Then $q_{-i}^t \in Q_{-i}^t$ denotes the nodes for all agents except i. We have $(q_{-i}^t, q_i^t) \in Q^t$. The probability of ending at q_i^t under π is

$$P(q_i^t \mid \pi) = \sum_{q_{-i}^t \in Q_{-i}^t} P\big((q_{-i}^t, q_i^t) \mid \pi\big), \tag{9}$$

where the sum terms are determined by Eq. (7). Again, by definition of conditional probability,

$$P(q_{-i}^t \mid q_i^t, \pi) = \frac{P\big((q_{-i}^t, q_i^t) \mid \pi\big)}{P(q_i^t \mid \pi)}, \tag{10}$$

where the term in the numerator is obtained from Eq. (7).

4.3 Value of policy nodes

We define the values of a node in a joint policy and an individual policy.

Definition 5 *(Value of a joint policy node)* Given a joint policy $\pi = (Q, q_0, \gamma, \lambda)$, the value of a node $q^t \in Q^t$ is defined as

$$V_t^\pi(q^t) = \sum_{h^t \in H^t} P(h^t \mid q^t, \pi) V_t^\pi(\tau(h^t), q^t),$$

where $P(h^t \mid q^t, \pi)$ is defined in Eq. (8) and $\tau(h^t)$ is the joint belief corresponding to history h^t as given in Eq. (4).

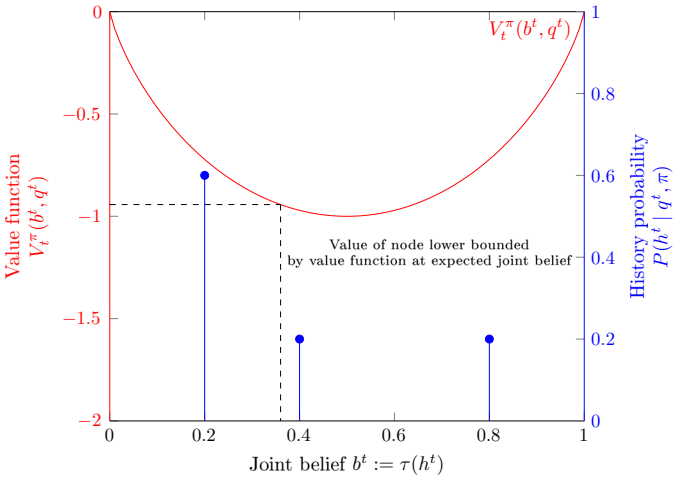


Fig. 4 An example of the value function of a node and its lower bound. The horizontal axis denotes the joint belief in a two-state Dec-POMDP as a real number indicating the probability of the first state. On the red left-hand vertical axis the value function $V_t^\pi(b^t, q^t)$ of the policy is drawn as a convex function of the joint belief. The blue right-hand vertical axis denotes the probability of joint histories, and the three lines with blue circle markers denote an example of a distribution $P(h^t \mid q^t, \pi)$ of joint belief states at a joint policy graph node. The exact value $V_t^\pi(q^t)$ of the node is calculated as the expectation of the value function $V_t^\pi(b^t, q^t)$ under $P(h^t \mid q^t, \pi)$. A lower bound for the value of the node is found by instead taking the value function at the expected joint belief state as indicated by the dashed lines

To give intuition about the definition above, consider the joint policy π in Fig. 3, and suppose $q^t = (q_{1,3}, q_{2,4})$. This node can be reached by three joint histories that correspond to the following three sequences of joint observations, listed as tuples of local observations: $\big((z_{1,0}, z_{2,1}), (z_{1,0}, z_{2,0})\big)$, $\big((z_{1,0}, z_{2,1}), (z_{1,1}, z_{2,0})\big)$, and $\big((z_{1,1}, z_{2,1}), (z_{1,1}, z_{2,0})\big)$. Note that the out-edge of q^t labeled with $(\cdot, z_{2,0})$ corresponds to two joint observations as described in the figure caption. The probabilities of joint histories are quantified by $P(h^t \mid q^t, \pi)$, which can be computed as outlined in Sect. 4.2. This probability can be non-zero only for the three joint histories that end in q^t. The value of the node, $V_t^\pi(q^t)$ is obtained by taking the expectation of $V_t^\pi(\tau(h^t), q^t)$ under the pmf of histories.

Another example for an arbitrary node q^t of a policy π is shown in Fig. 4. Here, the horizontal axis indicates the joint belief state in a Dec-POMDP with two underlying hidden states. The joint belief in this case is represented by a single real number denoting the probability of the first of these two states. The red curve depicts the value function $V_t^\pi(b^t, q^t)$ of the policy. The blue circle markers corresponding to the blue right-hand vertical axis depict the pmf $P(h^t \mid q^t, \pi)$ of histories at the node. The value $V_t^\pi(q^t)$ above is defined as the expectation of the value function (red curve) under the pmf of histories (blue circle markers).

Definition 6 *(Value of a local policy node)* For $i \in I$, let $\pi_i = (Q_i, q_{i,0}, \gamma_i, \lambda_i)$ be the local policy and let $\pi = (Q, q_0, \gamma, \lambda)$ be the corresponding joint policy. For any $i \in I$, the value of a local node $q_i^t \in Q_i^t$ is

$$V_t^{\pi}(q_i^t) = \sum_{q_{-i}^t \in Q_{-i}^t} P(q_{-i}^t \mid q_i^t, \pi) V_t^{\pi}\left((q_{-i}^t, q_i^t)\right),$$

where $P(q_{-i}^t \mid q_i^t, \pi)$ is defined in Eq. (10).

The value of a local node q_i^t is equal to the expected value of the joint node (q_{-i}^t, q_i^t) under $q_{-i}^t \sim P(q_{-i}^t \mid q_i^t, \pi)$. Suppose we wish to quantify the value of a local node q_i^t of agent i. From the perspective of agent i, it is not aware of which local policy nodes the other agents in their respective local policies are located at. However, since agent i is aware of the local policies of all agents, it can deduce which local policy nodes are possible for the other agents. These are the local policy nodes of agents other than i at the same time step as q_i^t, that is, the nodes in Q_{-i}^t. As an example, consider the case depicted in Fig. 3 and suppose we wish to compute the value of $q_1^t = q_{1,3}$ for agent 1. The second agent's node q_2^t belongs to $Q_2^t = \{q_{2,3}, q_{2,4}\}$. The probability of the second agent being in either of these two nodes is obtained from $P(q_2^t \mid q_1^t, \pi)$ as outlined in Sect. 4.2. According to the definition above, the value of $q_{1,3}$ is then computed as $P(q_{2,3} \mid q_{1,3}, \pi) V_t^{\pi}((q_{1,3}, q_{2,3})) + P(q_{2,4} \mid q_{1,3}, \pi) V_t^{\pi}((q_{1,3}, q_{2,4}))$.

5 Convex-reward Dec-POMDPs

In this section, we prove several results for the value function of a Dec-POMDP whose reward function is convex in $\Delta(S)$. Convex rewards are of special interest in information gathering. This is because of their connection to so-called uncertainty functions [13], which are non-negative functions concave in $\Delta(S)$. Informally, an uncertainty function assigns large values to uncertain beliefs, and smaller values to less uncertain beliefs. Negative uncertainty functions are convex and assign high values to less uncertain beliefs, and are suitable as reward functions for information gathering. Examples of uncertainty functions include Shannon entropy, generalizations such as Rényi entropy, and types of value of information, for example, the probability of error in hypothesis testing.

The following theorem shows that if the immediate reward functions are convex in the joint belief, then the finite horizon value function of any policy is convex in the joint belief.

Theorem 1 *If the reward functions $\rho_T : \Delta(S) \to \mathbb{R}$ and $\rho_t : \Delta(S) \times A \to \mathbb{R}$ are convex in $\Delta(S)$, then for any policy π, $V_T : \Delta(S) \to \mathbb{R}$ is convex and $V_t^{\pi} : \Delta(S) \times Q^t \to \mathbb{R}$ is convex in $\Delta(S)$ for any t.*

Proof Let $\pi = (Q, q_0, \gamma, \lambda)$, and $b \in \Delta(S)$. We proceed by induction ($V_T(b) = \rho_T(b)$ is trivial). For $t = T - 1$, let $q^{T-1} \in Q^{T-1}$, and denote by $a := \gamma(q^{T-1})$ the joint action taken according to π at this node. From Eq. (5), $V_{T-1}^{\pi}(b, q^{T-1}) = \rho_{T-1}(b, a) + \sum_{z \in Z} \eta(z \mid b, a) V_T(\zeta(b, a, z))$. We recall from above that V_T is convex, and by Eq. (1), the Bayes filter $\zeta(b, a, z)$ is a linear function of b. The composition of a linear and convex function is convex, so $V_T(\zeta(b, a, z))$ is a convex function of b. The non-negative weighted sum of convex functions is also convex, and by assumption ρ_{T-1} is convex in $\Delta(S)$, from which it follows that V_{T-1}^{π} is convex in $\Delta(S)$.

Now assume V^{π}_{t+1} is convex in $\Delta(S)$ for some $0 \leq t \leq T-1$. By the definition in Eq. (6) and the same argumentation as above, it follows that V^{π}_{t} is convex in $\Delta(S)$. □

Since a sufficiently large policy graph can represent any policy, we conclude that the value function of an optimal policy is convex in a Dec-POMDP with a reward function convex in the joint belief.

The following corollary gives a lower bound for the value of a policy graph node.

Corollary 1 *Let $g^t : H^t \to [0,1]$ be a probability mass function over the joint histories at time t. If the reward functions $\rho_T : \Delta(S) \to \mathbb{R}$ and $\rho_t : \Delta(S) \times A \to \mathbb{R}$ are convex in $\Delta(S)$, then for any time step t, and for any policy π and $q^t \in Q^t$,*

$$\mathbb{E}_{h^t \sim g(h^t)}\left[V^{\pi}_{t}(\tau(h^t), q^t)\right] \geq V^{\pi}_{t}\left(\mathbb{E}_{h^t \sim g(h^t)}\left[\tau(h^t)\right], q^t\right).$$

Proof By Theorem 1, $V^{\pi}_{t} : \Delta(S) \times Q^t \to \mathbb{R}$ is convex in $\Delta(S)$. The claim immediately follows applying Jensen's inequality. □

Applied to Definition 5, the corollary says the value of a joint policy node q^t is lower bounded by the value of the expected joint belief at q^t. An illustration of this lower bound is shown by the dashed lines in Fig. 4. Applied to Definition 6, we obtain a lower bound for the value of a local policy node q^t_i as

$$V^{\pi}_{t}(q^t_i) \geq \mathbb{E}_{q^t_{-i} \sim P(q^t_{-i} | q^t_i, \pi)}\left[V^{\pi}_{t}\left(\mathbb{E}_{h^t \sim P(h^t | q^t, \pi)}\left[\tau(h^t)\right], q^t\right)\right],$$

where inside the inner expectation we write $(q^t_{-i}, q^t_i) = q^t$. A lower bound for the value of any local node $q^t_i \in Q^t_i$ is found by finding the values $V^{\pi}_{t}(q^t)$ of all joint nodes $q^t \in Q^t$ and then taking the expectation of $V^{\pi}_{t}(q^t)$ where $q^t = (q^t_{-i}, q^t_i)$ under $P(q^t_{-i} \mid q^t_i, \pi)$.

As Corollary 1 holds for any pmf over joint histories, it could be applied also with pmfs other than $P(h^t \mid q^t, \pi)$. For example, if it is expensive to enumerate the possible histories and beliefs at a node, one could approximate the lower bound through importance sampling [32, Ch. 23.4].

Since a linear function is both convex and concave, rewards that are state-dependent and rewards that are convex in the joint belief can be combined on different time steps in one Dec-POMDP and the lower bound still holds.

In standard Dec-POMDPs, the expected reward is a linear function of the joint belief. Then, Corollary 1 holds with equality, as shown by the following.

Corollary 2 *Consider a Dec-POMDP where the reward functions are defined as $\rho_T(b) = \sum_{s \in S} b(s) R_T(s)$ where $R_T : S \to \mathbb{R}$ is a state-dependent final reward function, and for $t = 0, 1, \ldots, T-1$, $\rho_t(b,a) = \sum_{s \in S} b(s) R_t(s,a)$, where $R_t : S \times A \to \mathbb{R}$ are the state-dependent reward functions. Then, the conclusion of Corollary 1 holds with equality.*

Proof Let $\pi = (Q, q_0, \gamma, \lambda)$ and $b \in \Delta(S)$. First note that $V_T(b) = \rho_T(b) = \sum_{s \in S} b(s) R_T(s)$. Consider then $t = T-1$, and let $q^{T-1} \in Q^{T-1}$, and write $a := \gamma(q^{T-1})$. Then from the definition of V^{π}_{T-1} in Eq. (5), consider first the latter sum term which equals

$$\sum_{z\in Z}\eta(z\mid b,a)\sum_{s'\in S}\zeta(b,a,z)(s')R_T(s')$$
$$=\sum_{s'\in S}\left[\sum_{z\in Z}\sum_{s\in S}P^z(z\mid s',a)P^s(s'\mid a,s)b(s)\right]R_T(s')$$

which follows by replacing $\zeta(b,a,z)$ by Eq. (1), canceling out $\eta(z\mid b,a)$, and rearranging the sums. The above is clearly a linear function of b, and by definition, so is ρ_t, the first part of V^π_{T-1}. Thus, $V^\pi_{T-1}:\Delta(S)\times Q^{T-1}\to\mathbb{R}$ is linear in $\Delta(S)$. By an induction argument, it is now straightforward to show that V^π_t is linear in $\Delta(S)$ for $t=0,1,\ldots,T-1$. Finally,

$$\mathbb{E}_{h^t\sim g(h^t)}\left[V^\pi_t(\tau(h^t),q^t)\right]=V^\pi_t\left(\mathbb{E}_{h^t\sim g(h^t)}\left[\tau(h^t)\right],q^t\right)$$

for any pmf g over joint histories by linearity of expectation. □

Corollary 1 indicates that the value of a node is lower bounded by the value of the expected joint belief in the node. This result has applications in policy improvement algorithms that iteratively improve the value of a policy by modifying the output and node transition functions at each local policy node. Instead of directly optimizing the value of a node, the lower bound can be optimized. We present one such algorithm in the next section. As shown by Corollary 2, such an algorithm will also work for Dec-POMDPs with a state-dependent reward function.

6 The nonlinear policy graph improvement algorithm

The Policy Graph Improvement (PGI) algorithm [42] was originally introduced for Dec-POMDPs with a standard state-dependent reward function that is linear in the joint belief. PGI monotonically improves policies by locally modifying the output and node transition functions of the individual agents' policies. The policy size is fixed, such that the worst case computation time for an improvement iteration is known in advance. Moreover, due to the limited size of the policies the method produces compact, understandable policies.

We extend PGI to the non-linear reward case, and call the method non-linear PGI (NPGI). Contrary to tree based Dec-POMDP approaches the policy does not grow double-exponentially with the planning horizon as we use a fixed size policy. NPGI may improve the lower bound of the values of nodes (Corollary 1). The lower bound is tight when each policy graph node corresponds to only one history suggesting we can improve the quality of the lower bound by increasing policy graph size.

NPGI is shown in Algorithm 1. At each improvement step, NPGI repeats two steps: the forward pass and the backward pass. In the forward pass, the current best joint policy is applied to find the set B of expected joint beliefs at every policy graph node. In the backward pass, we iterate over all local policy graph nodes optimizing the policy parameters for each of them, that is, the output function and the node transition function. As output from the backward pass, we obtain an updated policy π^+ using the improved output and node transition functions γ^+ and λ^+, respectively. As NPGI

optimizes a lower bound of the node values, we finally check if the expected sum of rewards for the improved policy, $V_0^{\pi^+}(b^0, q_0)$, is greater than the value of the current best policy, and update the best policy if necessary. Similarly as PGI, NPGI is an anytime algorithm. It may be terminated at any point in time, and the best joint policy recovered so far may be returned.

In the following, we first give details of the forward pass. Then, we discuss in detail the most important part of the algorithm, the so-called backward pass. Finally, we close the section with a discussion of some implementation details.

6.1 The forward pass

Given a joint policy $\pi = (Q, q_0, \gamma, \lambda)$ and an initial joint belief b^0, the forward pass calculates the set $B = \{b_q \mid q \in Q\}$ of expected beliefs b_q for each of the joint policy graph nodes q. We implement the forward pass in two stages, first enumerating the local and joint histories, and then applying the Bayes filter to find the corresponding belief states along with their relative likelihoods.

From the local policy graphs $\pi_i = (Q_i, q_{i,0}, \gamma_i, \lambda_i)$ of each agent i (see Fig. 2), we first enumerate the sets of local histories ending at each of the nodes $q_i \in Q_i$. This corresponds to enumeration of all paths in the graph π_i. Next, for a given joint policy graph node $q = (q_1, \ldots, q_n) \in Q$, we look up the set of local histories for each q_i. The set of joint histories at q is then obtained by enumerating all combinations of local histories. That is, if there are m_i local histories that end at q_i, there are $\prod_{i=1}^{n} m_i$ unique combinations of local histories, each of which corresponds to one joint history that ends at q.

Now we have enumerated the set of joint histories that ends at q. Recursively using the Bayes filter, Eq. (4), we obtain for each joint history h a corresponding belief state $\tau(h)$. As described in Sect. 4.2, we also obtain the relative likelihoods $P(h \mid q, \pi)$ of the joint histories. The expected belief state at q is then obtained by $b_q = \sum_h P(h \mid q, \pi)\tau(h)$.

Algorithm 1 NPGI

Input: Policy $\pi = (Q, q_0, \gamma, \lambda)$, initial belief b^0
Output: Improved policy π

```
while not converged and time limit not exceeded do
    B ← ForwardPass(π, b^0)
    π^+ ← BackwardPass(π, B)
    if V_0^{π^+}(b^0, q_0) ≥ V_0^π(b^0, q_0) then π ← π^+
    end if
end while
return π
```

Algorithm 2 Backward pass of NPGI.

Input: Local policies $\pi_i = (Q_i, q_{0,i}, \gamma_i, \lambda_i)$ for each $i \in I$, expected beliefs $B = \{b_q \mid q \in Q\}$
Output: Improved local policies $\pi_i^+ = (Q_i, q_{0,i}, \gamma_i^+, \lambda_i^+)$ for each $i \in I$

for $i \in I$ **do**
  $\gamma_i^+ \leftarrow \gamma_i, \lambda_i^+ \leftarrow \lambda_i$ ▷ Initialization
end for
for $t = T-1, \ldots, 0$ **do**
  for $i \in I$ **do**
    $W_i^t \leftarrow \emptyset$
    for $q_i^t \in Q_i^t$ **do**
      if $t = T-1$ **then** ▷ Last time step: improve local action
        Solve Eq. (11), assign $\gamma_i^+(q_i^t)$
      else ▷ Other time steps: improve local action and node transitions
        Solve Eq. (12), assign $\gamma_i^+(q_i^t)$ and $\lambda_i^+(q_i^t, z_i) \forall z_i$
      end if
      if $\exists w_i^t \in W_i^t$: SamePolicy(w_i^t, q_i^t) **then** ▷ Remove duplicate local policies
        Redirect(q_i^t, w_i^t)
        Randomize(q_i^t)
        $B \leftarrow$ ForwardPass(π^+, b^0)
      end if
      $W_i^t \leftarrow W_i^t \cup \{q_i^t\}$ ▷ Update set of improved nodes
    end for
  end for
end for
return $\{(Q_i, q_{i,0}, \gamma_i^+, \lambda_i^+)\}_{i \in I}$
procedure SamePolicy(q_i^t, w_i^t)
  if $q_i^t = w_i^t$ **then return** True **end if**
  if $\gamma_i^+(q_i^t) \neq \gamma_i^+(w_i^t)$ **then return** False **end if**
  if $t < T-1$ **then**
    for $z_i \in Z_i$ **do**
      $q_i^{t+1} \leftarrow \lambda_i^+(q_i^t, z_i)$
      $w_i^{t+1} \leftarrow \lambda_i^+(w_i^t, z_i)$
      if not SamePolicy(q_i^{t+1}, q_i^{t+1}) **then return** False **end if**
    end for
  end if
  return True
end procedure
procedure Redirect(q_i^t, w_i^t)
  for $(x, z_i) \in \{(x, z_i) \in Q_i^{t-1} \times Z_i \mid \lambda_i^+(x, z_i) = q_i^t\}$ **do**
    $\lambda_i^+(x, z_i) = w_i$
  end for
end procedure
procedure Randomize(q_i^t)
  $\gamma_i^+(q_i^t) \sim \text{Uniform}(A_i)$
  if $t \neq T-1$ **then**
    $\forall z_i \in Z_i : \lambda_i^+(q_i^t, z_i) \sim \text{Uniform}(Q_i^{t+1})$
  end if
end procedure

6.2 The backward pass

We denote the improved joint policy as $\pi^+ = (Q, q_0, \gamma^+, \lambda^+)$. The parameters γ^+ and λ^+ of this policy will be incrementally updated throughout the backward pass. This is done by iterating over all nodes in the local policy graphs, and solving optimization problems to maximize the node values. The maximization is done over possible values of the output

function and node transition function at the node. At time step t for agent i, for each node $q_i^t \in Q_i^t$, we maximize either the value $V_t^{\pi^+}(q_i^t)$ or its lower bound with respect to the local policy parameters. In the following, we present the details for maximizing the lower bound. The algorithm for the exact value can be derived analogously, then we store all belief states possible at a node $q \in Q$ instead of the expected belief in the forward pass.

The backward pass of NPGI is shown in Algorithm 2. We first consider time step $t = T - 1$. We loop over each agent $i \in I$, and over each local policy graph node q_i^{T-1}. Since after this step no subsequent actions will be taken, we find an optimal local action and assign it to $\gamma_i^+(q_i^{T-1})$. An optimal local action is such that it maximizes the sum of the expected immediate and final reward. For clarity, in the following we drop explicit notations of the time step from the notation of beliefs, policy graph nodes, actions, and observations. Recall that $P(q_{-i} \mid q_i, \pi)$ gives the pmf over the local policy graph nodes of agents other than i. To simplify notation, we use the shorthand b_{-i} for the expected joint belief at a joint policy graph node $q = \left(q_i, q_{-i}\right)$. Finally, we write $a = (\gamma_1^+(q_1), \cdots, a_i, \cdots, \gamma_n^+(q_n))$ as the joint action where local actions of all other agents except i are fixed to those specified by the current output function γ^+. We solve

$$\max_{a_i \in A_i} \sum_{q_{-i} \in Q_{-i}} P(q_{-i} \mid q_i, \pi) \left[\rho_{T-1}(b_{-i}, a) + \sum_{z \in Z} \eta(z \mid b_{-i}, a) V_T(\zeta(b_{-i}, a, z)) \right] \tag{11}$$

and assign $\gamma_i^+(q_i)$ equal to the local action that maximizes Eq. (11).

Next, consider time step $t \leq T - 1$. There are now actions remaining after the current one, so we consider both the current local action and which node to traverse to next via the node transition function. We find an optimal local action and assign it to $\gamma_i^+(q_i^t)$, and find an optimal configuration for out edges of q_i^t and assign values of $\lambda_i(q_i^t, z_i)$ accordingly for each $z_i \in Z_i$. As earlier, we shall drop all explicit notations of the time step for clarity. We use the same notation for b_{-i} and a as above. Additionally, for any joint observation $z = (z_1, \ldots, z_n) \in Z$, define

$$f(z) = \left(\lambda_1^+(q_1, z_1), \ldots, q_i^{z_i}, \ldots, \lambda_n^+(q_n, z_n)\right)$$

as the next joint policy node to transition to when the transitions of all other agents except i are fixed to those specified by λ^+, and agent i transitions to $q_i^{z_i}$. We solve

$$\begin{aligned} \max_{\substack{a_i \in A_i \\ \forall z_i \in Z_i : q_i^{z_i} \in Q_i}} & \sum_{q_{-i} \in Q_{-i}} P(q_{-i} \mid q_i, \pi) \cdot \\ & \left[\rho_t\left(b_{-i}, a\right) + \sum_{z \in Z} \eta(z \mid b_{-i}, a) V_{t+1}^{\pi^+}\left(\zeta(b_{-i}, a, z), f(z)\right) \right], \end{aligned} \tag{12}$$

and assign $\gamma_i^+(q_i)$ and $\lambda_i^+(q_i, \cdot)$ to their respective maximizing values.

Line 13 of Algorithm 2 checks if there exists a node w_i^t that we have already optimized that has the same local policy as the current node q_i^t. If such a node exists, we redirect all of the in-edges of q_i^t to w_i^t to avoid having nodes with identical local policies. The redirection may change the expected beliefs at the nodes at time steps $t, t+1, \ldots, T-1$. To ensure that the expected joint beliefs and node reachability probabilities remain valid, we recompute the forward pass (Line 16).

If we redirect the in-edges of q_i^t to w_i^t, on Line 15 we randomize the local policy of the now useless node q_i^t that has no in-edges, in the hopes that it may be improved on

subsequent backward passes. To randomize the local policy of a node $q_i^t \in Q_i^t$, we sample new local policies until we find one that is not identical to the local policy of any other node in Q_i^t. Likewise, when randomly initializing a new policy in our experiments we avoid including in any Q_i^t nodes with identical local policies.

In Algorithm 2 the loop at Line 7 may sometimes encounter a node q_i^t that is unreachable. That is, there are no in-edges to q_i^t, or the probability of every joint history ending at q_i^t is equal to zero. In such cases, we randomize the local policy at q_i^t by calling the subroutine RANDOMIZE on it.

6.3 Computational complexity

We next derive the computational complexity of one improvement iteration for NPGI (Algorithm 1) when using the lower bound. We first consider the forward pass. The forward pass requires computation of all joint belief states reachable over the horizon of T decisions under the current joint policy π. For any history of past joint actions and observations, the joint policy π always specifies a single joint action to take. Therefore, the number of possible joint belief states at each time step increases by a factor of $|Z|$. The number of joint belief states evaluated in the forward pass is $O(|Z|^T)$.

The backward pass (Algorithm 2) iterates over all T time steps, and all of the n agents. For each agent i, all its controller nodes Q_i are iterated over. There are a total of $m = \sum_{i=1}^{n} |Q_i|$ controller nodes. There are Tnm iterations, and in the worst case at each iteration the optimization problem from Eq. (12) is solved.

To determine the complexity of solving Eq. (12), we determine the size of the feasible set of the optimization problem. At any node q_i, we may set the output function $\lambda_i(q_i)$ to equal any of the $|A_i|$ local actions. Likewise, the node transition function $\gamma_i(q_i, \cdot)$ for any of the $|Z_i|$ possible local observations may be set to equal any of the $O(|Q_i|)$ possible successor nodes. To determine the worst case complexity, let $|A_*| = \max_{i \in I} |A_i|$, $|Z_*| = \max_{i \in I} |Z_i|$, and $|Q_*| = \max_{i \in I} |Q_i|$. There are $O(|A_*||Z_*|^{|Q_*|})$ elements in the feasible set of Eq. (12).

To evaluate each feasible solution, a sum over the possible joint policy graph nodes q_{-i} of agents other than i is computed. Each of the other agents is in one of at most $|Q_*|$ local policy graph nodes. Thus, there are $O(|Q_*|^{n-1})$ possible node configurations for the other agents. Each sum term corresponds to a joint policy graph node (q_i, q_{-i}) and is equal to the expected value of the node starting at its expected joint belief, weighted by the reachability probability of the node. The policy is executed until the end of the problem horizon. By a similar argument as made above for the forward pass, evaluating this value requires computation of at most $O(|Z|^T)$ joint belief states and the corresponding rewards. The reachability probabilities $P(q_{-i} \mid q_i, \pi)$ are computed and cached already during the forward pass.

In summary, the computational complexity of one iteration in the while loop of Algorithm 1 is

$$O(|Z|^T + Tnm|A_*||Z_*|^{|Q_*|}|Q_*|^{n-1}|Z|^T). \tag{13}$$

The time complexity of an improvement step in NPGI is exponential in the number of agents n, the number of nodes $|Q_*|$ in the largest local policy graph. The complexity is exponential in the planning horizon T, unlike tree based Dec-POMDP approaches with doubly-exponential complexity. The complexity is linear in the number of actions.

6.4 Implementation details

We discuss how to initialize policies, and mention some techniques we use to escape local maxima. The source code of our implementation of NPGI is available online at https://github.com/laurimi/npgi.

Policy initialization. We initialize a random policy for each agent $i \in I$ with a given policy graph width $\left|Q_i^t\right|$ for each t as follows. For example, for a problem with $T = 3$ and $\left|Q_i^t\right| = 2$, we create a policy similar to Fig. 2 for each agent, where there is one initial node $q_{i,0}$, and 2 nodes at each time step $t \geq 1$. The action determined by the output function $\gamma_i(q_i)$ is sampled uniformly at random from A_i. For each node $q_i^t \in Q_i^t$ for $0 \leq t \leq T-1$, we sample a next node from Q_i^{t+1} uniformly at random for each observation $z_i \in Z_i$ and assign the node transition function $\lambda_i(q_i, z_i)$ accordingly. At the last time step, it is only meaningful to have $\left|Q_i^T\right| \leq |A_i|$. In our experiments if $\left|Q_i^T\right| > |A_i|$, we instead set $\left|Q_i^T\right| = |A_i|$.

Escaping local maxima. NPGI repeats the forward and backward passes several times. During our experiments we observed that sometimes NPGI gets stuck in a local maxima, and is unable to find improved policy parameters that yield a higher value policy. To mitigate this, we follow the well known ϵ-greedy approach [51] in reinforcement learning: select the best action according to a specified probability, and, for exploration, select a suboptimal action otherwise. In practice, we added a randomization such that at each local node q_i^t, with probability 0.5, instead of optimizing the policy parameters as discussed above we perform the following heuristic improvement step. First, we sample a single joint history that ends in q_i^t. We then solve the optimization problems of Eqs. (12) for the joint belief that corresponds to the sample joint history. We empirically observed that this added randomness helped NPGI to more effectively search the space of policies and to reach higher quality solutions.

7 Non-linear policy graph improvement for continuous-state problems

We describe how NPGI, introduced in the previous section, can be extended to Dec-POMDPs where the state space S is continuous. The main modifications we make are to represent belief states approximately as sets of weighted particles, and to apply particle filtering (c.f. [44]) to replace the exact Bayes filter. This allows us to handle belief states without assuming them to have any particular parametric representation such as a Gaussian distribution. We require that samples can be drawn from the initial belief, and from the state transition model P^s and observation model P^z. Furthermore, we require that we can evaluate probabilities $P^z(z^{t+1} \mid s^{t+1}, a^t)$.

7.1 Forward pass

Given a Dec-POMDP and a joint policy $\pi = (Q, q_0, \gamma, \lambda)$, in the forward pass we track a joint distribution $P(s^t, q^t \mid \pi)$ over states and joint policy graph nodes. We first approximate this distribution at $t = 0$ by a set of $N \in \mathbb{N}$ weighted particles as

$$P(s^0, q^0 \mid \pi) \approx \sum_{j=1}^{N} w^{0,(j)} \delta(s^0 - s^{0,(j)}) \delta(q^0 - q^{0,(j)})$$

where δ is the Dirac delta function, and we sample each particle $j = 1, 2, \ldots, N$ according to

$$s^{0,(j)} \sim b^0(s^0), \quad q^{0,(j)} \sim \delta(q^0 - q_0),$$

with initial weight $w^{0,(j)} = 1/N$.

Given the particle approximation at time t, we find the approximation at time $(t+1)$ by the particle update process in Algorithm 3. We use this algorithm to calculate the sets of particles for all time steps $t = 1, 2, \ldots, T$. For each particle, we sample the next hidden state and the observation, and set the next policy graph node accordingly. No weight updates are required, as the effect of the observation probabilities are already included when drawing samples from P^z. We therefore omit the weights from the algorithm description, but note that they are still required for subsequent steps in the backward pass.

Algorithm 3 Update of particles from t to $t+1$.

Input: Particles $\{s^{t,(j)}, q^{t,(j)}\}_{j=1}^N$
Output: Particles $\{s^{t+1,(j)}, q^{t+1,(j)}\}_{j=1}^N$
1: **for** $j \in \{1, 2, \ldots N\}$ **do**
2: $a^t \leftarrow \gamma(q^{t,(j)})$ ▷ Joint action to take at node
3: $s^{t+1,(j)} \sim P^s(s^{t+1} \mid s^{t,(j)}, a^t)$ ▷ Draw next state sample
4: $z^{t+1} \sim P^z(z^{t+1} \mid s^{t+1,(j)}, a^t)$ ▷ Draw next observation
5: $q^{t+1,(j)} \leftarrow \lambda(q^{t,(j)}, z^{t+1})$ ▷ Next node
6: **end for**
7: **return** $\{s^{t+1,(j)}, q^{t+1,(j)}\}_{j=1}^N$

After running Algorithm 3 for all time steps, we have a particle approximation of $P(s^t, q^t \mid \pi)$ for all $t = 0, \ldots, T$ as a set of particles $\{s^{t,(j)}, q^{t,(j)}, w_q^{t,(j)}\}_{j=1}^N$. For any time step t and joint policy graph node $q \in Q^t$, we obtain an approximation of $P(s^t \mid q, \pi)$ as follows. Let $J_q \subseteq \{1, 2, \ldots, N\}$ the subset of indices j of particles for which $q^{t,(j)} = q$. Then the set

$$\{s^{t,(j)}, q^{t,(j)}, w_q^{t,(j)} | j \in J_q\}$$

of particles where

$$w_q^{t,(j)} = \frac{w^{t,(j)}}{\sum_{l \in J_q} w^{t,(l)}}$$

are the normalized weights approximates $P(s^t \mid q, \pi)$. The normalization term $\sum_{l \in J_q} w_q^{t,(l)}$ approximates $P(q \mid \pi)$, the probability of reaching q under π.

7.2 Backward pass

The backward pass requires no modifications in terms of the main optimization problems, Eqs. (11) and (12) that are to be solved. However, the value of a policy has to be

calculated using the particle approximation of the belief. We estimate the expected sum of rewards obtained when following π until the end of the horizon by using policy rollouts [8, 55].

When evaluating the value of a node q_i^t in a policy π during the backward pass, we approximate the objective function of Eqs. (11) and (12) by Algorithm 4. We loop over all joint policy graph nodes q^t where agent i is at local policy graph node q_i^t. On Lines 3-4 we first get the particle approximation of $P(s^t \mid q^t, \pi)$, the expected belief state at q^t, and an estimate of $P(q^t \mid \pi)$. We then average the return of K policy rollouts starting from the expected belief to obtain an estimate $\hat{v}_q$ of the expected sum of rewards from q^t until the end of the horizon. The value is weighted by the prior probability to reach q^t and added to the estimate of sum of rewards (Line 10).

In each rollout, we select actions according to the policy and sample a joint observation (Line 20). We then apply a particle filter to find the posterior belief at the next time step (Line 21). Particle filtering approximates the optimal Bayes filter in cases where computing the filtering equations exactly is infeasible, for example due to non-linearity of the state transition or observation model, or because the joint belief state cannot be represented exactly. In principle any particle filtering algorithm may be applied. We describe in the following briefly the sequential importance resampling (SIR) particle filter we use in our experiments. More details on SIR and approximate Bayesian filtering may be found, e.g., in [44].

We drop the subscript q and assume we have N particles – with weights normalized to sum to one – that we wish to update using SIR. We are given a particle set $\{s^{k,(j)}, w^{k,(j)}\}_{j=1}^N$ and a joint action a^k and joint observation z^{k+1}. First, SIR updates each particle by drawing a sample from the state transition model according to

$$s^{k+1,(j)} \sim P^s(s^{k+1} \mid s^{k,(j)}, a^k) \tag{14}$$

for every particle j. Second, the weights of all particles are updated according to

$$w^{k+1,(j)} \propto w^{k,(j)} P^z(z^{k+1} \mid s^{k+1,(j)}, a^k). \tag{15}$$

After updating weights, they are renormalized to sum to one. The weight updates tend to have an effect of increasing the relative weights of state particles most likely to correspond to the perceived joint observation. However, repeating the updates often leads to a situation where almost all particles have zero or close to zero weights. Resampling is applied to avoid this degenerate situation. If the effective number of particles $1/\sum_{j=1}^{N}(w^{k+1,(j)})^2$ falls below $N/10$, the particles are resampled after the weight update. In resampling, the weights $w^{k+1,(j)}$ are considered as probabilities in a discrete distribution on the particles. A set of N samples are drawn from this discrete distribution, and the old particle set is replaced by the sampled one. Weights are reset to all be equal to $1/N$.

During the rollout, we approximate the per-step reward function ρ_t by the procedure EstimateReward. At the final time step, we approximate the final reward function ρ_T using the procedure EstimateFinalReward. These methods apply the current particle approximation of the belief state to evaluate the per-step reward function, and their details depend on the selected reward function. In Sect. 8.2, we describe a possible implementation in a signal source seeking domain.

Algorithm 4 Estimate expected sum of rewards by policy rollouts when agent i starts at node q_i^t.

Input: Node q_i^t, joint policy π, particle approximation $\{s^{t,(j)}, q^{t,(j)}, w^{t,(j)}\}_{j=1}^N \sim P(s^t, q^t \mid \pi)$, number of rollouts K
Output: Estimate $\hat{v}$ of the expected sum of rewards when following π, conditional on agent i being in local policy graph node q_i^t.

$\hat{v} \leftarrow 0$
for $q^t = (q_1^t, \ldots, q_n^t) \in Q^t$ where agent i is at q_i^t **do**
  Get particle approximation $P_q^t = \{s^{t,(j)}, w_q^{t,(j)} \mid j \in J_q\}$ ▷ $P(s^t \mid q^t, \pi)$
  $p \leftarrow \sum_{j \in J_q} w^{k,(j)}$ ▷ $P(q^t \mid \pi)$
  $\hat{v}_q \leftarrow 0$
  for $k = 0, 1, \ldots, K-1$ **do**
    $\hat{r} \leftarrow \textsc{Rollout}(q^t, P_q^t, \pi)$
    $\hat{v}_q \leftarrow \frac{\hat{r} + k \cdot \hat{v}_q}{k+1}$ ▷ Update moving average
  end for
  $\hat{v} \leftarrow \hat{v} + p \cdot \hat{v}_q$
end for
return $\hat{v}$
procedure $\textsc{Rollout}(q^t, P_q^t, \pi)$
  $\hat{r} \leftarrow 0$
  for $k = t, t+1, \ldots, T-1$ **do**
    $a^k \leftarrow \gamma(q^k)$ ▷ Action at node q^k
    $\hat{r} \leftarrow \hat{r} + \textsc{EstimateReward}(P_q^k, a^k)$
    Sample state s^k from P_q^k
    $s^{k+1} \sim P^s(s^{k+1} \mid s^k, a^k)$
    $z^{k+1} \sim P^z(z^{k+1} \mid s^k, a^k)$
    Find updated P_q^{k+1} using particle filtering on P_q^k with a^k, z^{k+1}
    $q^{k+1} \leftarrow \lambda(q^k, z^{k+1})$ ▷ Update policy graph node
  end for
  $\hat{r} \leftarrow \hat{r} + \textsc{EstimateFinalReward}(P_q^T)$
  return $\hat{r}$
end procedure

Besides the number of improvement iterations, in the continuous-state variant of NPGI the trade-off between algorithm runtime and policy quality is affected by the number of rollouts K. Algorithm 4 provides an unbiased estimate $\hat{v}$ of the expected sum of rewards, but the variance of the estimate depends on K. As described in [55], with an estimate error tolerance of ϵ and a confidence threshold of ξ, the required number of rollouts is

$$K(\epsilon, \xi) = \frac{(v_{max} - v_{min})^2 \ln \frac{1}{\xi}}{2\epsilon^2}, \tag{16}$$

where v_{max} and v_{min} are the greatest and smallest possible expected sums of rewards, respectively. Note that from the forward pass we obtain a particle approximation of the expected belief state at a particular node. Therefore, in the backward pass we optimize local policies to maximize the lower bound similarly as described in Sect. 6. The number of required rollouts concerns approximation quality of the lower bound.

Fig. 5 Arrangement of locations in the MAV domain (left) and rovers domain (right)

8 Experiments

We evaluate the performance of NPGI on information gathering Dec-POMDPs. We discuss discrete domains in Sect. 8.1 and a continuous-state source seeking domain in Sect. 8.2. In both subsections, we introduce the problem domains, the experimental setup, and then present the results. The source code of our implementation of NPGI and the experimental domains are available online at https://github.com/laurimi/npgi.

8.1 Discrete domains

We run experiments on the micro air vehicle (MAV) domain of [27] and propose an information gathering rovers domain inspired by the Mars rovers domain of [2]. In both tasks the objective of the agents is to maximize the expected sum of rewards collected minus the entropy of the joint belief at the end of the problem horizon. We provide a brief summary of the domains in the following, and a complete description in the "Appendix".

MAV domain. A target moves between four possible locations, l_i in Fig. 5. The target is either friendly or hostile; a hostile target moves more aggressively. Two MAVs, MAV1 and MAV2 in the figure, are tasked with tracking the target and inferring whether it is friendly or hostile. The MAVs can choose to use either a camera or a radar sensor to sense the location of the target. An observation from either sensor corresponds to a noisy measurement of the target's location. The camera is more accurate if the target is close, whereas the radar is more accurate when the target is farther away. The Manhattan distance is applied, i.e., at l_0 the target is at distance 0 from MAV1 and at distance 3 from MAV2. If both MAVs apply their radars simultaneously, accuracy decreases due to interference.

Using the camera has zero cost, and using the radar sensor has a cost of 0.1, and an additional cost of 1 or 0.1 if the target is at distance 0 or 1 to the MAV, respectively, to model the risk of revealing the MAVs own location to the (potentially hostile) target. To model information gathering, we set the final reward equal to the negative Shannon entropy of the joint belief, i.e., $\rho_T(b) = \sum_{s \in S} b(s) \log_2 b(s)$. The initial belief is uniform over all states. This problem has 8 states; 4 target locations and a binary variable for friendly/hostile, and 2 actions and 4 observations per agent.

Information gathering rovers. Two rovers are collecting information on four sites l_i of interest arranged as shown in Fig. 5. Each site is in one of two possible states which remains fixed throughout. The agents can move north, south, east, or west. Movement fails with probability 0.2, in which case the agent remains at its current location. The agents always fully observe their own location. Additionally, the agents can choose to conduct measurements of the site they are currently at. A binary measurement of the site status is recorded with false positive and false negative probabilities of 0.2 each. If the agents measure at the same location at the same time, the false positive and false negative probabilities are significantly lower, 0.05 and 0.01, respectively. Movement has

zero cost, while measurement has a cost of 0.1. The final reward is equal to the negative entropy. The initial belief is such that one agent starts at l_0, the other at l_3, with a uniform belief over the site status. The problem has 256 states, and 5 local actions and 8 local observations per agent.

8.1.1 Experimental setup

We compare NPGI to one exact algorithm and two heuristic algorithms. The exact method we employ is the Generalized Multi-Agent A* with incremental clustering and expansion, or GMAA*-ICE [38], with the QPOMDP search heuristic. According to [38] a vector representation of the search heuristic, analogous to the representation of an optimal POMDP value function by a set of so-called α-vectors [48], can help scale up GMAA*-ICE to larger problems. However, since the vector representation only exists if the reward function is linear in the joint belief, we represent the search heuristic as a tree. The two heuristic methods are joint equilibrium based search for policies, JESP [33], and direct cross-entropy policy search, DICEPS [40].

All of the methods above are easily modified to our domains where the final reward is equal to the negative Shannon entropy. However, applicability of NPGI is wider as it allows the reward at *any* time step to be a convex function of the joint belief. There are also other algorithms such as FB-HSVI [14] and PBVI-BB [29] that have demonstrated good performance on many benchmarks. However, these algorithms rely on linearity of the reward to achieve compression of histories and joint beliefs, and non-trivial modifications beyond the scope of this work would be required to extend them to Dec-POMDPs with non-linear rewards.

As baselines, we report values of a greedy open loop policy that executes a sequence of joint actions that has the maximal expected sum of rewards under the initial belief, and the best blind policy that always executes the same joint action.

We run NPGI using both the exact value of nodes and the lower bound from Corollary 1. The number of policy graph nodes $\left|Q_i^t\right|$ at each time step t is 2, 3, or 4. For each run with NPGI we run 30 backward passes, starting from randomly sampled initial policies. For all methods, we report the averages over 100 runs. If a run does not finish in 2 hours, we terminate it.

8.1.2 Results

Tables 1 and 2 show the average policy values in the MAV and information gathering rovers problems, respectively. NPGI is indicated by "Ours" when the lower bound (LB) was used, and as "Ours (No LB)" when exact evaluation of node values was applied. Results are reported as function of the problem horizon T, and for NPGI also as function of the policy graph size $\left|Q_i^t\right|$. Where applicable, the standard error obtained as the ratio of the sample standard deviation and the square root of the number of samples is reported. The symbol "-" indicates missing results due to exceeding the cut-off time.

GMAA*-ICE finds an optimal solution, but similarly to [27] we find that it does not scale beyond $T = 3$ in either problem. Considering $T = 2$ and $T = 3$, the average values of our method are very close to the optimal value in both problems. In these cases, we found that NPGI finds an optimal policy in about 60% of the runs.

In the MAV problem (Table 1), performance of our method is consistent for varying policy graph size $\left|Q_i^t\right|$ and horizon T. This indicates that even a small policy suffices to

Table 1 Average policy values in the MAV domain ($|S| = 8, |A_i| = 2, |Z_i| = 4$)

Method	$\lvert Q_i^t \rvert$	$T = 2$	$T = 3$	$T = 4$	$T = 5$
Ours	2	$-1.931\ (2 \times 10^{-3})$	$-1.833\ (3 \times 10^{-4})$	$\mathbf{-1.768}\ (6 \times 10^{-5})$	$-1.725\ (1 \times 10^{-4})$
	3	$-1.931\ (2 \times 10^{-3})$	$-1.833\ (3 \times 10^{-4})$	$\mathbf{-1.768}\ (6 \times 10^{-5})$	$\mathbf{-1.724}\ (5 \times 10^{-6})$
	4	$-1.931\ (2 \times 10^{-3})$	$-1.832\ (3 \times 10^{-4})$	$\mathbf{-1.768}\ (2 \times 10^{-5})$	$\mathbf{-1.724}\ (5 \times 10^{-6})$
Ours (No LB)	2	$-1.930\ (2 \times 10^{-3})$	$-1.832\ (2 \times 10^{-3})$	$\mathbf{-1.768}\ (8 \times 10^{-5})$	$-1.725\ (7 \times 10^{-5})$
	3	$-1.930\ (2 \times 10^{-3})$	$-1.833\ (3 \times 10^{-3})$	$\mathbf{-1.768}\ (5 \times 10^{-5})$	$\mathbf{-1.724}\ (5 \times 10^{-6})$
	4	$-1.930\ (2 \times 10^{-3})$	$-1.832\ (2 \times 10^{-3})$	$\mathbf{-1.768}\ (2 \times 10^{-5})$	$\mathbf{-1.724}\ (6 \times 10^{-6})$
DICEPS		$-1.925\ (1 \times 10^{-3})$	$-1.937\ (3 \times 10^{-3})$	$-1.926\ (1 \times 10^{-3})$	$-1.940\ (2 \times 10^{-3})$
JESP		$-1.953\ (2 \times 10^{-3})$	$-1.859\ (2 \times 10^{-3})$	$-1.794\ (4 \times 10^{-4})$	$-1.750\ (4 \times 10^{-4})$
GMAA*-ICE		$\mathbf{-1.919}$	$\mathbf{-1.831}$	–	–
Greedy		−2.156	−2.044	−1.978	−1.932
Blind		−1.945	−1.904	−1.909	−1.932

Best values for each planning horizon are bolded. Numbers in parentheses indicate standard error where applicable

Table 2 Average policy values in the information gathering rovers domain ($|S| = 256, |A_i| = 5, |Z_i| = 8$)

Method	$\lvert Q_i^t \rvert$	$T = 2$	$T = 3$	$T = 4$	$T = 5$
Ours	2	$-3.495\ (3 \times 10^{-3})$	$-3.192\ (2 \times 10^{-3})$	$-3.036\ (2 \times 10^{-3})$	$-2.981\ (2 \times 10^{-3})$
	3	$-3.495\ (3 \times 10^{-3})$	$-3.190\ (1 \times 10^{-3})$	$\mathbf{-3.034}\ (0)$	$\mathbf{-2.975}\ (1 \times 10^{-3})$
	4	$-3.506\ (4 \times 10^{-3})$	$-3.192\ (2 \times 10^{-3})$	$-3.037\ (2 \times 10^{-3})$	$-3.041\ (8 \times 10^{-3})$
Ours (No LB)	2	$-3.495\ (3 \times 10^{-3})$	$-3.190\ (1 \times 10^{-3})$	$\mathbf{-3.034}\ (0)$	$-3.010\ (6 \times 10^{-3})$
	3	$-3.496\ (3 \times 10^{-3})$	$-3.190\ (1 \times 10^{-3})$	$\mathbf{-3.034}\ (0)$	–
	4	$-3.506\ (4 \times 10^{-3})$	$-3.190\ (1 \times 10^{-3})$	$-3.106\ (1 \times 10^{-2})$	–
DICEPS		$-3.482\ (1 \times 10^{-3})$	$-3.535\ (1 \times 10^{-2})$	$-3.825\ (2 \times 10^{-2})$	$-4.792\ (3 \times 10^{-2})$
JESP		$-3.483\ (1 \times 10^{-3})$	$-3.536\ (6 \times 10^{-3})$	–	–
GMAA*-ICE		$\mathbf{-3.479}$	$\mathbf{-3.189}$	–	–
Greedy		−3.844	−4.031	−3.877	−3.818
Blind		$\mathbf{-3.479}$	−3.412	−3.418	−3.472

Best values for each planning horizon are bolded. Numbers in parentheses indicate standard error where applicable

reach a high value in this problem. We also see that applying the lower bound does not reduce the quality of the policy found by our approach. In the rover problem (Table 2), a compact policy with as few as 2 nodes per time step in the policy graph can lead to a high value. The situation is potentially different in domains with many more actions and observations. Given a fixed policy graph size, in such large domains there are more possible joint belief states per policy graph node than in domains with few actions and observations. As it is possible that every joint belief state requires a different action as a response to reach a high expected sum of rewards, the likelihood to discover suboptimal policies is increased in large domains. The number of joint belief states per policy graph node also affects the quality of the lower bound (Corollary 1). Informally speaking, the greater the number of possible joint belief states, the greater the chance that the bound is lower than the exact value.

Table 3 Average NPGI backward pass duration (in seconds) with or without lower bound (LB)

	MAV		Rovers	
T	With LB	No LB	With LB	No LB
2	0.002	0.002	0.04	0.04
3	0.04	0.05	0.26	0.34
4	1.20	2.74	1.43	4.40
5	31.02	55.34	32.23	158.7

A higher standard error indicates more variation in the results, and is applicable for stochastic methods such as NPGI, DICEPS, and JESP. From Tables 1 and 2 we find all of the methods perform consistently, with a low standard error. Notably, for policy graph size parameter $|Q_i^t|$ equal to 2 or 3 with $T = 4$, NPGI without lower bound always converges to a policy with the same value. In the rovers domain (Table 2) we observe the standard error to increase for $T = 5$. As fewer improvement iterations can be completed within the cut-off time of 2 hours, the effect of the randomly initialized starting policy is still seen.

Table 3 shows the average duration of one backward pass of Algorithm 1 as function of the problem horizon T with $\left|Q_i^t\right| = 2$, with or without using the lower bound (LB). The lower runtime requirement when applying the lower bound is seen clearly for $T \geq 4$. The runtime of NPGI is dominated by the backward pass and solving the local policy optimization problems, Eqs. (11) and (12), which applying the lower bound help reduce. As indicated by the results in Tables 1 and 2, applying the lower bound also does not degrade the quality of the policies found. However, as noted above, the lower bound is also potentially looser in domains with a greater number of actions and observations.

Our method outperforms the greedy and blind baselines except for $T = 2$ in the rover problem where a blind policy of always measuring is optimal. In several cases, JESP and DICEPS return policies with a value lower than one or both of the baselines.

The size of the policy graph in NPGI must be specified before calculating the policy. As shown by our experiments, fixing the policy graph size effectively limits the space of policies to be explored and can produce compact and understandable policies. However, a potential weakness is that optimizing over fixed-size policies excludes the possibility to find a larger but potentially better policy.

Execution of NPGI may be terminated after any number of improvement steps and the best joint policy found so far will be returned. This allows the user to control the trade-off between the quality of the returned policy and the runtime of the algorithm. To investigate this trade-off and its dependence on the policy graph size and whether the lower bound is applied or not, we recorded the value of the best joint policy after each improvement iteration in each of the 100 runs. In Fig. 6, we report the average, greatest, and smallest values of the best joint policies for the MAV domain with planning horizon $T = 5$. We display results only for the first 30 improvement steps, as there were no significant changes over the remaining 20 improvement steps.

When the lower bound is *not* applied, i.e., the value function is evaluated exactly, on average NPGI improves the value of the joint policy slightly faster. This is seen by comparing the solid lines in the left and right hand plots of Fig. 6 during the initial 10 improvement steps. However, after a sufficient number of improvement steps (in this case more than 20), NPGI with the lower bound also tends to achieve joint policies of equal quality, as also shown in Table 1.

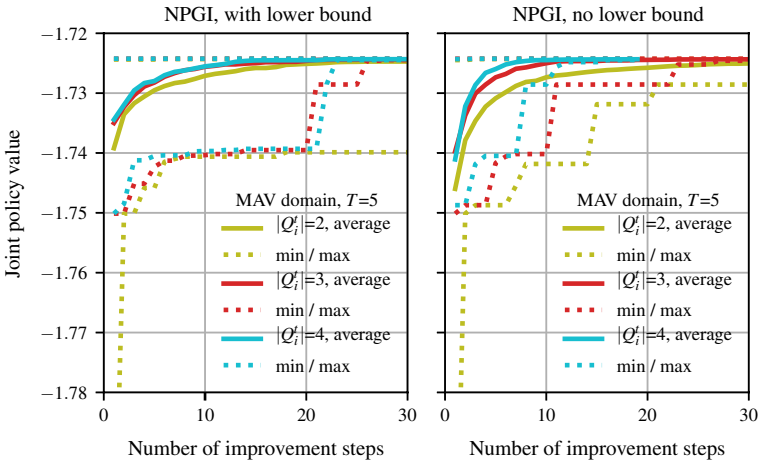


Fig. 6 Joint policy value in the MAV domain with planning horizon $T = 5$ as a function of the number of improvement steps for NPGI with the lower bound (left) and without the lower bound (right). Solid lines indicate the average value of all best policies over the 100 repetitions. The dotted lines indicate the greatest and smallest values of the best policies found over all repetitions. Note that the maxima are identical such that only the cyan dotted line is visible. The color of the lines indicates the size parameter $|Q_i^t|$ of the policy graphs. NPGI may be terminated after any number of improvement steps

Figure 6 indicates that the average value of the best joint policy over all repetitions converges towards the overall best value policy found among all repetitions (indicated by the upper dotted line). In most cases NPGI is able to find good quality policies given sufficient improvement steps. However, the worst cases indicated by the lower dotted lines are indicative of a weakness of NPGI, namely that it is still possible that a poor initialization leads NPGI to get stuck at a poor local maximum. This issue is more severe the smaller the policy graph: with a poorly initialized and small policy graph, there are few opportunities to improve the policy. This can be seen by comparing the lower dotted lines in Fig. 6 that show the smallest value of the best joint policy found over all the 100 repetitions of the experiment. When using the lower bound with size parameter $|Q_i^t| = 2$, the joint policy cannot be improved above the value of -1.74 in some repetitions. As is also seen from Fig. 6, worst-case performance is improved when exact value computation is used instead of the lower bound, and tends to also be better when larger policy graph sizes are considered.

We examined the results also for other planning horizons in the MAV domain. For planning horizons $T = 2$ and 3, the average, greatest, and smallest values were similar regardless of whether the lower bound was used or not. As discussed in Sect. 6, the lower bound is tighter the less histories map to each policy graph node. With shorter planning horizons, there are less histories possible at a particular policy graph node, which explains why the results are similar for both NPGI and NPGI without the lower bound. For planning horizon $T = 4$, the results not using the lower bound resulted in improvements of the average value over fewer improvement steps, similarly as in Fig. 6. However, the difference in the average values again decreases when more improvement steps are completed.

We also examined the results in the information gathering rovers as a function of the number of improvement steps, and observed similar trends as in the MAV domain. Average joint policy value over all repetitions for NPGI both with or without using the lower

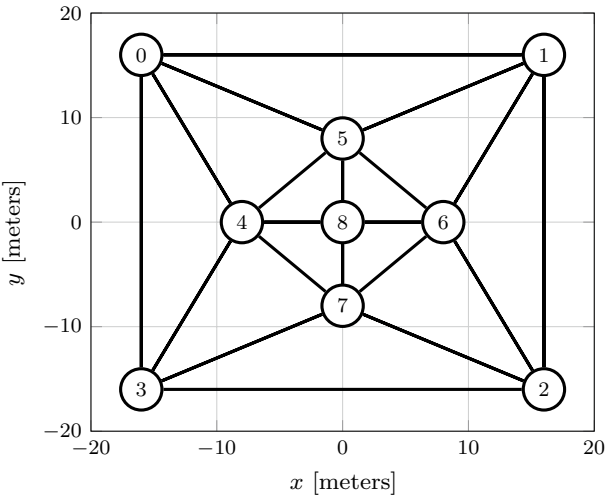


Fig. 7 A signal source is located in the 40-by-40 meter region depicted. Two agents can move between the nodes along the edges of the undirected graph. The number i inside each node indicates the location l_i it corresponds to. The closer an agent is to the signal source, the stronger the received signal strength (RSS) it records tends to be. The agents infer the source location based on their RSS observations

bound converges towards the maximum joint policy value among all repetitions. Convergence tends to require fewer iterations when not using the lower bound. In the case $T = 5$ the comparison could not be done, since without using the lower bound only at most 4 improvement steps could be completed within the cutoff time for $|Q_i^t| = 2$, and none for the larger policy graph sizes.

8.2 Continuous-state source seeking

We consider a signal source seeking problem where two agents infer the location of a signal source based on noisy data. The agents can move around the area and observe the received signal strength (RSS) from the source. RSS tends to be lower the farther away the source is from the receiving agent. We use NPGI to design local policies for the agents with the objective of minimizing the uncertainty of the source location estimate.

There is a stationary signal source at some unknown location l_s in the two dimensional space $\mathcal{L} = [-20, 20] \times [-20, 20]$. The two agents can move between a finite subset $L = \{l_0, l_1, \ldots, l_8\} \subset \mathcal{L}$ of locations along the edges of an undirected graph, as illustrated in Fig. 7. The possible local actions of each agent are to either deterministically move to any location adjacent to their current location, or stay where they are.

After moving to location l, the agent records the RSS from the source. We model RSS similarly as [5] by

$$\mathrm{RSS}(l, l_s) = P_{tx} + G_{tx} - L_{tx} + G_{rx} - L_{rx} - L_{fs}(l, l_s) - R, \tag{17}$$

where $P_{tx} = 18$ dBm is the power transmitted by the source, $G_{tx} = 1.5$ dBi is the transmitter antenna gain, $L_{tx} = 0$ dB is the transmitter loss, $G_{rx} = 1.5$ dBi is the receiver antenna gain, $L_{rx} = 0$ dB is the receiver loss, L_{fs} is the free space loss, and R is the noise due to signal fading. The free space loss in dB is

$$L_{fs}(l, l_s) = -27.55 + 20\log_{10}(\nu) + 20\log_{10}(d(l, l_s)), \tag{18}$$

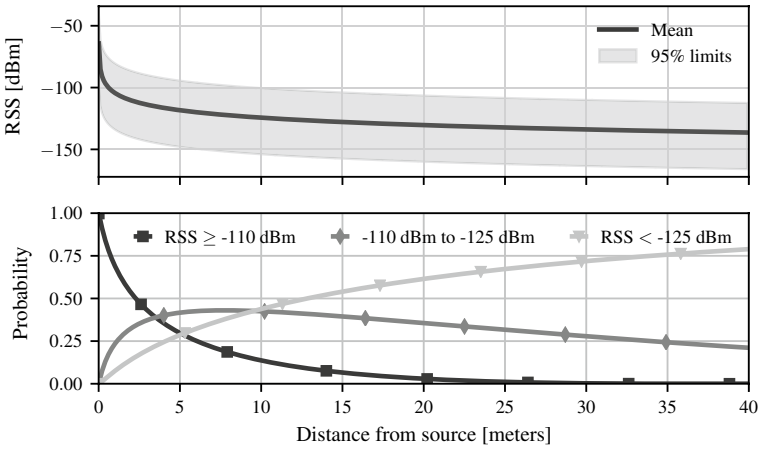


Fig. 8 The observation model with Rician fading in the source seeking task. Above: The mean of received signal strength (RSS) and its 95% limits under Rician fading as a function of distance. Below: the probabilities of the three discretized observations as a function of distance

where $\nu = 2.4$ GHz is the transmitted frequency, and $d(l, l_s)$ is the Euclidean distance between the source and receiver. We assume Rician fading, which fits scenarios where one signal path is dominant over the others. We set $R \sim \text{Rice}(\mu, \sigma)$ with $\mu = 20$ dB, $\sigma = 4$ dB. The distribution of the received signal strength with fading as a function of distance from the source is illustrated in Fig. 8.

For planning, we discretize the perceived RSS into three discrete observations: high (RSS ≥ -110 dBm), medium (-110 dBm $>$ RSS $\geq$ -125 dBm), and low (RSS $<$ -125 dBm). The probabilities of the respective discretized observations as a function of distance from the source are illustrated in Fig. 8.

We implement the procedure EstimateFinalReward from Algorithm 4 by first evaluating the weighted sample covariance, and then computing its log determinant. Denote the final set of weighted particles as $\{w^{T,(j)}, s^{T,(j)}\}_{j=1}^{N}$, where each state particle $s^{T,(j)} = (l_1^{T,(j)}, l_2^{T,(j)}, l_s^{T,(j)})$ is a tuple consisting of the locations $l_1^{T,(j)}, l_2^{T,(j)} \in L$ of the two agents and the source location $l_s^{T,(j)} \in \mathcal{L} \subset \mathbb{R}^2$. We first compute the weighted average source location

$$\hat{l}_s^T = \sum_{j=1}^{N} w^{T,(j)} l_s^{T,(j)}. \tag{19}$$

The 2-by-2 weighted sample covariance matrix Σ_T is then obtained by

$$\Sigma_T = \frac{1}{1 - \sum_{j=1}^{N} (w^{T,(j)})^2} \sum_{j=1}^{N} w^{T,(j)} (l_s^{T,(j)} - \hat{l}_s^T)(l_s^{T,(j)} - \hat{l}_s^T)^{\mathrm{T}}, \tag{20}$$

where $(\cdot)^{\mathrm{T}}$ denotes transpose. The procedure finally returns $-\log\det \Sigma_T$, the log determinant of the weighted sample covariance matrix of the state particles.

This choice of reward function is suitable since the weighted covariance of the particles is smaller the more tightly the particles are clustered together. Unimodal density estimates

where covariance is small are also preferred over multimodal estimates where the overall covariance is larger even if the modes themselves are strongly concentrated. This makes the reward function suitable for tasks where a single location estimate should be produced. A technical justification for the reward function choice is that the differential entropy (a measure of uncertainty for continuous random variables) of a multivariate Gaussian distribution is proportional to the log determinant of the covariance matrix of the Gaussian.

Since all other rewards in the source seeking problem are equal to zero, the procedure EstimateReward from Algorithm 4 simply always returns zero. We briefly outline how other, more complicated reward functions could be implemented in EstimateReward. Nonlinear rewards $\rho_t : \Delta(S) \times A \to \mathbb{R}$ could be estimated by applying a similar strategy as for the procedure EstimateFinalReward above, by implementing a suitable function that maps a joint action and a belief state represented as set of weighted particles to a real-valued reward. Given a weighted set of particles $\{w^{t,(j)}, s^{t,(j)}\}_{j=1}^{N}$ and a joint action a^t, a standard state and action based reward could easily be implemented by returning the weighted sum $\sum_{j=1}^{N} w^{t,(j)} R(s^{t,(j)}, a^t)$, where R is an appropriate state-based reward function.

For numerical stability, we clamped the maximum value returned by EstimateFinalReward to 50.0. The smallest and greatest possible sum of rewards in any rollout by Algorithm 4 are $v_{min} = 0.0$ and $v_{max} = 50.0$, respectively.

8.2.1 Experimental setup

We apply the NPGI variant for continuous-state problems with particle filtering to find a joint policy for the agents in the source seeking problem. We investigate planning horizons $T = 3, 4, \ldots, 7$ with randomly initialized policy graphs of width $|Q_i^t| = 2$ or 3. We use 10000 particles to represent $P(s^t, q^t \mid \pi)$. During the backward pass, we apply $K = 100$ rollouts to evaluate the policy values. Assuming a confidence threshold $\xi = 0.05$, and given the smallest and greatest possible sums of rewards $v_{min} = 0.0$ and $v_{max} = 50.0$ returned in any rollout this value corresponds to an error tolerance of $\epsilon \approx 6$ according to Eq. (16). We experimented with 10 rollouts, giving an error tolerance of approximately 20 which we found to make the algorithm unstable, with a large number of changes in the local policies due to the high variance of rollout value estimates. On the other hand, experiments with 1000 rollouts with corresponding error tolerance of approximately 1.9 did not significantly improve the outcomes while increasing the runtime. We run 50 improvement steps (forward and backward passes) in each case. A cut-off time of 2 hours is applied for policy graph width 2, and 6 hours for policy graph width 3. We repeat the NPGI policy computation 100 times using different random seeds.

We compare NPGI to two baselines: a random policy where each agent chooses a valid action uniformly at random, and a hand-designed patrolling policy. The patrolling policy is designed such that the agents cover the maximum amount of locations in the graph (see Fig. 7). In the patrolling policy, agent 1 follows the path $(l_1, l_2, l_6, l_5, l_8, l_7, l_2)$, and agent 2 follows the path $(l_0, l_3, l_4, l_7, l_8, l_5, l_1)$.

We run 1000 simulations on each policy. The initial belief of the signal source location is uniform over $\mathcal{L}$, and we sample the true location of the signal source uniformly at random. Both agents start at l_0. We record the average error in the final estimate of the source location. For state estimation during the simulations, we use a particle filter on the continuous observed RSS values.

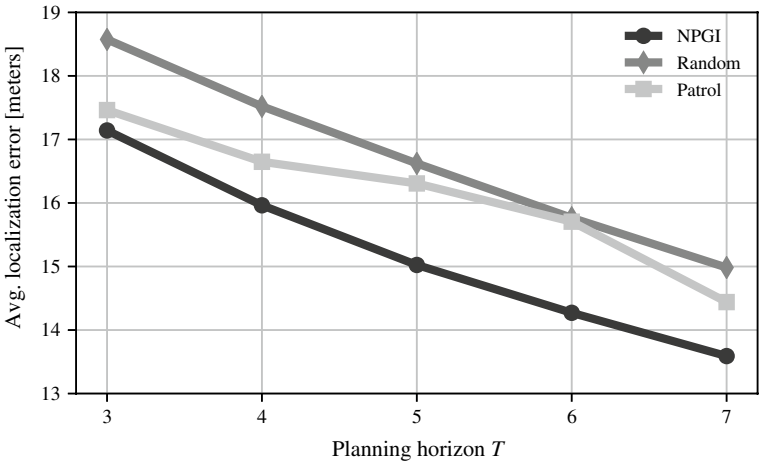


Fig. 9 Average error of the source location estimate (lower is better) as a function of the planning horizon T. Our NPGI planning approach outperforms random and patrolling policies

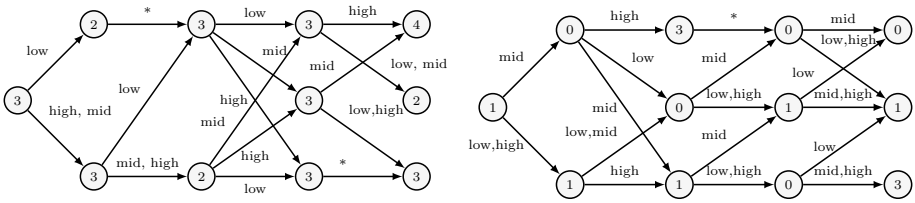


Fig. 10 An example of policies found by NPGI. Left: local policy for agent 1. Right: local policy for agent 2. Nodes are labeled by the movement action to be taken. The edge labels indicate the discretized observation conditional on which the edge is traversed. An asterisk (*) indicates unconditional traversal of the edge. Unreachable nodes are not drawn

8.2.2 Results

The average source localization error for the random, patrolling, and policies found by NPGI are shown in Fig. 9. For each planning horizon T, we report the final average error obtained by simulating the corresponding policy for T steps. For NPGI, we report only the result with policy graph width of $|Q_i^t| = 2$, as we found the average to not differ significantly from $|Q_i^t| = 3$. The standard error for all policies is very small, in the order of 10^{-2} meters, and is not visible in Fig. 9.

As expected, the policy of randomly selecting locations from which to observe the RSS performs worst. For $T = 3$, the patrolling policy that attempts to visit as many locations as possible and our NPGI policies perform almost equally well. However, for $T \geq 4$, NPGI policies that decide the next location based on the history of RSS measurements are able to reach a lower average error than the patrolling policy. The performance difference between patrolling policy and NPGI diminishes for $T = 7$. When all locations l_i have been visited, it is likely that the agents have perceived a high RSS in at least one of them which allows accurate estimation of the source location.

Figure 10 shows an example of one of the policies found by NPGI for $T = 5$ with policy graph width $|Q_i^t| = 3$. This policy reached an average localization error of 15.1 meters. The policy of agent 1 is shown on the left side of the figure, and it indicates the agent mostly stays in the bottom half of the environment around nodes 2, 3, and 4 (see Fig. 7). The policy of agent 2, shown on the right of Fig. 10, indicates it in turn remains in the upper half of the environment, mainly at nodes 0 and 1, sometimes 3.

The compact policies produced by NPGI are easily interpretable and understandable. We see that the two agents cooperate by concentrating their search efforts on the bottom and top halves of the environment (Fig. 7), respectively. Interestingly, the second to last action for agent 1 is always to move to location 3. However, conditional on how the agent arrives there (through which path in the policy graph), and what its final observation is, the agent's final location may be either one of 2, 3, or 4. Another observation we can make is that both agents avoid the center of the environment, never moving to location 8. RSS measurements recorded at the other surrounding nodes (0,1,2,3) suffice to estimate the source location.

We have demonstrated that NPGI can be applied to continuous-state domains, and that it can outperform random and simple heuristic policies. One further advantage that planning methods such as NPGI have but we do not investigate here in detail is that plans can be generated for particular initial information simply by re-executing the planner. In contrast, heuristic policies potentially require redesign by hand for each new initial information state.

9 Conclusion and future work

We showed that if the reward function in a finite-horizon Dec-POMDP is convex in the joint belief, then the value function of any policy is convex in the joint belief. Rewards that are convex in the joint belief are important in information gathering problems. We applied the result to derive a lower bound for the value, and empirically demonstrated that it improves the run-time of a heuristic anytime planning algorithm without degrading solution quality. We empirically showed that our algorithm outperforms existing heuristic solvers that are applicable to convex reward Dec-POMDPs. Furthermore, we derived an extension of our algorithm to continuous-state information gathering Dec-POMDPs with finite action and observation spaces and demonstrated its performance in a cooperative source seeking problem.

The work we presented here offers a principled formulation of model based multi-agent active information gathering, viewing it as a Dec-POMDP with an information-theoretic reward function. There are several potential directions of future research into decentralized information gathering problems.

We considered reward functions that are convex in the joint belief, since they are well-justified for information-gathering problems. However, NPGI with exact value function evaluation is applicable with rewards that depend on the joint belief in an arbitrary way. Investigation of other types of reward functions is a potential future direction.

Single-agent POMDPs with piecewise linear and convex (PWLC) reward functions can be shown to be equivalent to POMDPs with linear rewards through an augmentation of the action space with prediction actions [45]. Whether such an equivalence can be established for multi-agent Dec-ρPOMDPs we studied in this article remains an open question. As a convex function can be arbitrarily well approximated by a PWLC function, finding such

an equivalence can potentially lead to an approximation algorithm for Dec-ρPOMDPs with bounded suboptimality.

The heuristic anytime algorithm we present offers no guarantees on the performance of policies found. We empirically found the performance to be close to optimal in the discrete domains we examined and outperforming all baselines in the continuous-state domain. It remains to be investigated whether these findings translate to domains with a greater number of actions and observations as well. Up to date, state-of-the-art solvers for Dec-POMDPs with linear rewards (e.g., [14, 29]) can address problems larger than the Dec-ρPOMDPs with non-linear rewards we addressed here. Scaling up to larger problems remains an open challenge for Dec-ρPOMDPs.

Acknowledgements Open Access funding provided by Projekt DEAL. Jan Peters was supported by European Research Council under Grant No. 640554 (SKILLS4ROBOTS) and Joni Pajarinen by German Research Foundation project PA 3179/1-1 (ROBOLEAP)

Full description of the discrete experimental domains

In this appendix, we provide a full description of the two discrete experimental domains.

Micro air vehicle

In the micro air vehicle (MAV) domain there are two agents such that $I = \{1, 2\}$. A state is a pair (l, f) consisting of the location l and the type f of the target. The location takes a value in $L = \{l_0, l_1, l_2, l_3\}$ where the arrangement of locations is illustrated in Fig. 5 (left). The type takes a value in $F = \{\text{friendly}, \text{hostile}\}$. The state space is $S = L \times F$. Agent 1 is located at l_0, and agent 2 is located at l_4 (see Fig. 5 left). Each agent has local action space $A_i = \{\text{camera}, \text{radar}\}$ and local observation space $Z_i = \{z_0, z_1, z_2, z_3\}$, where each z_k corresponds to a Manhattan distance of k from the agent.

The type of the target remains fixed throughout the task. Target motion is stochastic. If the target is friendly (hostile), it will stay at its current location with probability 0.85 (0.6). Otherwise, it will move to an adjacent location uniformly at random.

Reliability of measurements depends on the distance between each agent and the target. Furthermore, reliability is degraded due to interference if both agents operate their radar concurrently. As hostile targets are actively avoiding detection, measuring their location is less reliable. If an agent applies its camera, the measurement model is described by the two matrices

$$O_{c,f} = \begin{bmatrix} 0.9961 & 0.2547 & 0.2173 & 0.2426 \\ 0.0039 & 0.4419 & 0.2561 & 0.2493 \\ 0.0000 & 0.2547 & 0.2705 & 0.2534 \\ 0.0000 & 0.0487 & 0.2561 & 0.2547 \end{bmatrix}, \quad O_{c,h} = \begin{bmatrix} 0.6945 & 0.2616 & 0.2396 & 0.2468 \\ 0.2855 & 0.2957 & 0.2520 & 0.2497 \\ 0.0198 & 0.2616 & 0.2564 & 0.2515 \\ 0.0002 & 0.1811 & 0.2520 & 0.2520 \end{bmatrix} \tag{21}$$

where $O_{c,f}$ concerns the case where the target is friendly, and $O_{c,h}$ the case where the target is hostile. From top to bottom, each row of the matrix corresponds to the measurements z_1, z_2, z_3, and z_4. From left to right, each column of the matrix corresponds to the true Manhattan distance between the agent and the target. Each matrix element prescribes the conditional probability of perceiving the corresponding local observation given the true distance to the target.

In case only one of the agents uses its radar, detection reliability is improved. The measurement model is described by the two matrices

$$O_{r,f} = \begin{bmatrix} 1.0000 & 0.0404 & 0.0224 & 0.0703 \\ 0.0000 & 0.9192 & 0.2336 & 0.1867 \\ 0.0000 & 0.0404 & 0.5104 & 0.3354 \\ 0.0000 & 0.0000 & 0.2336 & 0.4076 \end{bmatrix}, \quad O_{r,h} = \begin{bmatrix} 0.9220 & 0.2493 & 0.1658 & 0.2004 \\ 0.0780 & 0.4623 & 0.2634 & 0.2431 \\ 0.0000 & 0.2493 & 0.3074 & 0.2729 \\ 0.0000 & 0.0391 & 0.2634 & 0.2836 \end{bmatrix} \tag{22}$$

interpreted similarly as above. If both agents concurrently apply their radar, measurement quality degrades due to inference and both agents' measurement model is described by the matrices

$$O_{\hat{r},f} = \begin{bmatrix} 0.5705 & 0.2662 & 0.1799 & 0.1733 \\ 0.3460 & 0.3418 & 0.2618 & 0.2369 \\ 0.0772 & 0.2662 & 0.2965 & 0.2857 \\ 0.0063 & 0.1258 & 0.2618 & 0.3041 \end{bmatrix} \quad O_{\hat{r},h} = \begin{bmatrix} 0.5000 & 0.2625 & 0.2173 & 0.2212 \\ 0.3533 & 0.3013 & 0.2561 & 0.2466 \\ 0.1247 & 0.2626 & 0.2705 & 0.2632 \\ 0.0220 & 0.1736 & 0.2561 & 0.2690 \end{bmatrix} \tag{23}$$

interpreted as above.

On time steps $t < T$, a standard state-dependent reward of the form $R(s, a)$ is used. For each agent that applies its radar a reward of -0.1 is obtained, but otherwise rewards are zero for all state and action pairs. At step $t = T$, the reward is set equal to the negative Shannon entropy of the joint belief state. The initial state distribution is uniform over all states.

Information gathering rovers

In the information gathering rovers domain there are two agents such that $I = \{1, 2\}$. A state is a tuple $(s_1, s_2, m_1, m_2, m_3, m_4)$, where s_i denotes the location of agent i and m_k denotes the state of location k. The location s_i assumes values in $L = \{l_0, l_1, l_2, l_3\}$, where the locations are arranged as shown in Fig. 5 (right). The state of each location is a binary variable $m_k \in \{0, 1\}$. In total, the state space is $S = L \times L \times \{0, 1\}^4$. The action space of both agents is $A_i = \{\text{north}, \text{south}, \text{east}, \text{west}, \text{measure}\}$. Each agent records an observation that is a pair $(\hat{l}, \hat{m})$ of a measured location $\hat{l}$ and a measured state $\hat{m}$ of the current location. The possible observations of both agents are $Z_i = L \times \{0, 1\}$.

The state of each location remains fixed throughout the task. The movement actions may change the agents' locations. If an illegal movement action is chosen (e.g., attempting to move south at location l_1), the agent's location remains unchanged. Otherwise, the movement succeeds with probability 0.8, and with probability 0.2 the agent remains at its current location.

Regardless of the action taken by an agent, it will always correctly observe its own current location. The second part of the measurement, the state of the current location, is subject to uncertainty. If an agent takes a movement action, it will observe the state as either 0 or 1 uniformly at random. If one agent i alone takes a measurement action at a location l_k, the measurement probability is conditional on the state m_k according to

$$p_{alone}(\hat{m} = 0 \mid m_k) = \begin{cases} 0.80 \text{ if } m_k = 0 \\ 0.15 \text{ if } m_k = 1 \end{cases}, \tag{24}$$

and $p_{alone}(\hat{m} = 1 \mid m_k) = 1 - p_{alone}(\hat{m} = 0 \mid m_k)$. In case both agents are at the same location l_k and select the measurement action, the measurement reliability of both agents is improved, and governed by

$$p_{together}(\hat{m} = 0 \mid m_k) = \begin{cases} 0.99 \text{ if } m_k = 0 \\ 0.05 \text{ if } m_k = 1 \end{cases} \tag{25}$$

and $p_{together}(\hat{m}_i = 1 \mid m_k) = 1 - p_{together}(\hat{m}_i = 0 \mid m_k)$.

On time steps $t < T$, a state-dependent reward $R(s, a)$ is used. For each agent that takes a measurement action a reward of -0.1 is obtained, otherwise rewards are zero. At step $t = T$, the reward is set equal to the negative Shannon entropy of the joint belief state. The initial state distribution is such that agent 1 starts at location l_0 and agent 2 at location l_3, and the belief over location states m_k is uniform.

References

1. Allen, M., & Zilberstein, S. (2009). Complexity of decentralized control: Special cases. In *Advances in neural information processing systems* (pp. 19–27).
2. Amato, C., & Zilberstein, S. (2009). Achieving goals in decentralized POMDPs. In *Autonomous agents and multiagent systems (AAMAS)* (pp. 593–600).
3. Araya-López, M., Buffet, O., Thomas, V., & Charpillet, F. (2010). A POMDP extension with belief-dependent rewards. In *Advances in neural information processing systems* (pp. 64–72).
4. Atanasov, N., Le Ny, J., Daniilidis, K., & Pappas, G. J. (2015). Decentralized active information acquisition: Theory and application to multi-robot SLAM. In *IEEE International conference on robotics and automation (ICRA)* (pp. 4775–4782).
5. Atanasov, N. A., Le Ny, J., & Pappas, G. J. (2015). Distributed algorithms for stochastic source seeking with mobile robot networks. *Journal of Dynamic Systems, Measurement, and Control*, *137*(3), 031004.
6. Bajcsy, R., Aloimonos, Y., & Tsotsos, J. K. (2018). Revisiting active perception. *Autonomous Robots*, *42*(2), 177–196.
7. Bernstein, D. S., Givan, R., Immerman, N., & Zilberstein, S. (2002). The complexity of decentralized control of Markov decision processes. *Mathematics of Operations Research*, *27*(4), 819–840.
8. Besse, C., & Chaib-draa, B. (2008). Parallel rollout for online solution of Dec-POMDPs. In *21st international florida artificial intelligence research society conference (FLAIRS)* (pp. 619–624).
9. Best, G., Cliff, O. M., Patten, T., Mettu, R. R., & Fitch, R. (2019). Dec-MCTS: Decentralized planning for multi-robot active perception. *The International Journal of Robotics Research*, *38*(2–3), 316–337.
10. Calinescu, G., Chekuri, C., Pal, M., & Vondrák, J. (2011). Maximizing a monotone submodular function subject to a matroid constraint. *SIAM Journal on Computing*, *40*(6), 1740–1766.
11. Capitan, J., Spaan, M. T., Merino, L., & Ollero, A. (2013). Decentralized multi-robot cooperation with auctioned POMDPs. *The International Journal of Robotics Research*, *32*(6), 650–671.
12. Corah, M., & Michael, N. (2019). Distributed matroid-constrained submodular maximization for multi-robot exploration: Theory and practice. *Autonomous Robots*, *43*(2), 485–501.
13. DeGroot, M. H. (2004). *Optimal statistical decisions*. Hoboken: Wiley. Wiley Classics Library edition.
14. Dibangoye, J. S., Amato, C., Buffet, O., & Charpillet, F. (2016). Optimally solving Dec-POMDPs as continuous-state MDPs. *Journal of Artificial Intelligence Research*, *55*, 443–497.

15. Fehr, M., Buffet, O., Thomas, V., & Dibangoye, J. (2018). rho-POMDPs have Lipschitz-continuous epsilon-optimal value functions. In *Advances in neural information processing systems* (pp. 6933–6943).
16. Fioretto, F., Pontelli, E., & Yeoh, W. (2018). Distributed constraint optimization problems and applications: A survey. *Journal of Artificial Intelligence Research*, *61*, 623–698.
17. Gharesifard, B., & Smith, S. L. (2017). Distributed submodular maximization with limited information. *IEEE Transactions on Control of Network Systems*, *5*(4), 1635–1645.
18. Goldman, C. V., & Zilberstein, S. (2003). Optimizing information exchange in cooperative multi-agent systems. In *Autonomous agents and multiagent systems (AAMAS)* (pp. 137–144).
19. Hansen, E. A., Bernstein, D. S., & Zilberstein, S. (2004). Dynamic programming for partially observable stochastic games. In *AAAI conference on artificial intelligence* (pp. 709–715).
20. Hero, A. O., & Cochran, D. (2011). Sensor management: Past, present, and future. *IEEE Sensors Journal*, *11*(12), 3064–3075.
21. Hollinger, G. A., & Singh, S. (2012). Multirobot coordination with periodic connectivity: Theory and experiments. *IEEE Transactions on Robotics*, *28*(4), 967–973.
22. Jain, M., Taylor, M., Tambe, M., & Yokoo, M. (2009). DCOPs meet the real world: Exploring unknown reward matrices with applications to mobile sensor networks. In it International joint conference on artificial intelligence (IJCAI).
23. Julian, B. J., Angermann, M., Schwager, M., & Rus, D. (2012). Distributed robotic sensor networks: An information-theoretic approach. *The International Journal of Robotics Research*, *31*(10), 1134–1154.
24. Krause, A., Singh, A., & Guestrin, C. (2008). Near-optimal sensor placements in Gaussian processes: Theory, efficient algorithms and empirical studies. *Journal of Machine Learning Research*, *9*(Feb), 235–284.
25. Lalitha, A., & Javidi, T. (2017). Learning via active hypothesis testing over networks. In *IEEE information theory workshop (ITW)* (pp. 374–378).
26. Lalitha, A., Javidi, T., & Sarwate, A. D. (2018). Social learning and distributed hypothesis testing. *IEEE Transactions on Information Theory*, *64*(9), 6161–6179.
27. Lauri, M., Heinänen, E., & Frintrop, S. (2017). Multi-robot active information gathering with periodic communication. In *IEEE international conference on robotics and automation (ICRA)* (pp. 851–856).
28. Lauri, M., Pajarinen, J., & Peters, J. (2019). Information gathering in decentralized POMDPs by policy graph improvement. In *Autonomous agents and multiagent systems (AAMAS)* (pp. 1143–1151).
29. MacDermed, L. C., & Isbell, C. L. (2013). Point based value iteration with optimal belief compression for Dec-POMDPs. In *Advances in neural information processing systems* (pp. 100–108).
30. Meier, L., Peschon, J., & Dressler, R. (1967). Optimal control of measurement subsystems. *IEEE Transactions on Automatic Control*, *12*(5), 528–536.
31. Moore, E. F. (1956). Gedanken-experiments on sequential machines. *Automata Studies*, *34*, 129–153.
32. Murphy, K. (2012). *Machine learning: A probabilistic perspective*. Cambridge: MIT Press.
33. Nair, R., Tambe, M., Yokoo, M., Pynadath, D., & Marsella, S. (2003). Taming decentralized POMDPs: Towards efficient policy computation for multiagent settings. In *International joint conference on artificial intelligence (IJCAI)* (pp. 705–711).
34. Nguyen, D. T., Yeoh, W., Lau, H. C., Zilberstein, S., & Zhang, C. (2014). Decentralized multi-agent reinforcement learning in average-reward dynamic DCOPs. In *AAAI conference on artificial intelligence*.
35. Oliehoek, F. A. (2013). Sufficient plan-time statistics for decentralized POMDPs. In *International joint conference on artificial intelligence (IJCAI)* (pp. 302–308).
36. Oliehoek, F. A., & Amato, C. (2014). Dec-POMDPs as non-observable MDPs. IAS technical report IAS-UVA-14-01, Intelligent Systems Lab, University of Amsterdam, Amsterdam, The Netherlands.
37. Oliehoek, F. A., & Amato, C. (2016). *A concise introduction to decentralized POMDPs*. Berlin: Springer.
38. Oliehoek, F. A., Spaan, M. T., Amato, C., & Whiteson, S. (2013). Incremental clustering and expansion for faster optimal planning in Dec-POMDPs. *Journal of Artificial Intelligence Research*, *46*, 449–509.
39. Oliehoek, F. A., Spaan, M. T., Whiteson, S., & Vlassis, N. (2008). Exploiting locality of interaction in factored Dec-POMDPs. In *Autonomous agents and multiagent systems (AAMAS)* (pp. 517–524).
40. Oliehoek, F. A., Spaan, M. T. J., & Vlassis, N. (2008). Optimal and approximate q-value functions for decentralized POMDPs. *Journal of Artificial Intelligence Research*, *32*(1), 289–353.

41. Omidshafiei, S., Agha-Mohammadi, A. A., Amato, C., Liu, S. Y., How, J. P., & Vian, J. (2017). Decentralized control of multi-robot partially observable Markov decision processes using belief space macro-actions. *The International Journal of Robotics Research*, *36*(2), 231–258.
42. Pajarinen, J. K., & Peltonen, J. (2011). Periodic finite state controllers for efficient POMDP and DEC-POMDP planning. In *Advances in neural information processing systems* (pp. 2636–2644).
43. Rabinovich, Z., Goldman, C. V., & Rosenschein, J. S. (2003). The complexity of multiagent systems: The price of silence. In *Autonomous agents and multiagent systems (AAMAS)* (pp. 1102–1103).
44. Särkkä, S. (2013). *Bayesian filtering and smoothing*. Cambridge: Cambridge University Press.
45. Satsangi, Y., Whiteson, S., Oliehoek, F. A., & Spaan, M. T. (2018). Exploiting submodular value functions for scaling up active perception. *Autonomous Robots*, *42*(2), 209–233.
46. Schlotfeldt, B., Thakur, D., Atanasov, N., Kumar, V., & Pappas, G. J. (2018). Anytime planning for decentralized multirobot active information gathering. *IEEE Robotics and Automation Letters*, *3*(2), 1025–1032.
47. Seuken, S., & Zilberstein, S. (2007). Memory-bounded dynamic programming for DEC-POMDPs. In *International joint conference on artificial intelligence (IJCAI)* (pp. 2009–2015).
48. Smallwood, R. D., & Sondik, E. J. (1973). The optimal control of partially observable Markov processes over a finite horizon. *Operations Research*, *21*(5), 1071–1088.
49. Spaan, M. T., Gordon, G. J., & Vlassis, N. (2006). Decentralized planning under uncertainty for teams of communicating agents. In *Autonomous agents and multiagent systems (AAMAS)* (pp. 249–256).
50. Spaan, M. T., Veiga, T. S., & Lima, P. U. (2015). Decision-theoretic planning under uncertainty with information rewards for active cooperative perception. *Autonomous Agents and Multi-Agent Systems*, *29*(6), 1157–1185.
51. Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. Cambridge: MIT Press.
52. Szer, D., Charpillet, F., & Zilberstein, S. (2005). MAA*: A heuristic search algorithm for solving decentralized POMDPs. In *Uncertainty in artificial intelligence (UAI)* (pp. 576–583).
53. Taylor, M. E., Jain, M., Jin, Y., Yokoo, M., & Tambe, M. (2010). When should there be a "me" in a "team"?: Distributed multi-agent optimization under uncertainty. In *Autonomous agents and multiagent systems (AAMAS)* (pp. 109–116).
54. Taylor, M. E., Jain, M., Tandon, P., Yokoo, M., & Tambe, M. (2011). Distributed on-line multi-agent optimization under uncertainty: Balancing exploration and exploitation. *Advances in Complex Systems*, *14*(03), 471–528.
55. Wu, F., Zilberstein, S., & Chen, X. (2010). Rollout sampling policy iteration for decentralized POMDPs. In *Uncertainty in artificial intelligence (UAI)* (pp. 666–673).
56. Wu, F., Zilberstein, S., & Chen, X. (2011). Online planning for multi-agent systems with bounded communication. *Artificial Intelligence*, *175*(2), 487–511.
57. Zivan, R., Yedidsion, H., Okamoto, S., Glinton, R., & Sycara, K. (2015). Distributed constraint optimization for teams of mobile sensing agents. *Autonomous Agents and Multi-Agent Systems*, *29*(3), 495–536.