# Recursive Small-Step Multi-Agent A$^*$ for Dec-POMDPs

**Wietze Koops**, **Nils Jansen**, **Sebastian Junges** and **Thiago D. Simão**
Radboud University, Nijmegen, The Netherlands
{wietze.koops, nils.jansen, sebastian.junges, thiago.simao}@ru.nl

## Abstract

We present recursive small-step multi-agent A$^*$ (RS-MAA$^*$), an exact algorithm that optimizes the expected reward in decentralized partially observable Markov decision processes (Dec-POMDPs).

RS-MAA$^*$ builds on multi-agent A$^*$ (MAA$^*$), an algorithm that finds policies by exploring a search tree, but tackles two major scalability concerns. First, we employ a modified, small-step variant of the search tree that avoids the double exponential outdegree of the classical formulation. Second, we use a tight and recursive heuristic that we compute on-the-fly, thereby avoiding an expensive precomputation. The resulting algorithm is conceptually simple, yet it shows superior performance on a rich set of standard benchmarks.

## 1 Introduction

This paper considers finite-horizon decentralized decision-making under stochastic dynamics and partial observability. Markov decision processes (MDPs) are *the* formalism to capture decision-making under (stochastic) uncertainty. These models assume that a centralized controller can fully observe the state space. Partially observable MDPs (POMDPs) capture the natural restriction that a controller only observes features of the current state. However, they still assume a centralized controller that can collect information from all sensors. In decentralized control, we assume that various agents each locally observe features of the state space and must locally select actions to execute. Such problems are formally captured by decentralized partially observable Markov decision processes (Dec-POMDPs) [Oliehoek and Amato, 2016]. These models naturally capture, for instance, sensor networks or multi-robot navigation in settings where communication is impossible, ineffective, or lossy.

Finding an optimal (decentralized) controller (hereafter: a *policy*) is computationally challenging; the associated decision problem is NEXP-hard [Bernstein *et al.*, 2002]. Nevertheless, a series of advances resulted in a collection of effective algorithms that can find (1) provably optimal (exact) or (2) $\varepsilon$-optimal solutions. The focus of this paper is the former type, and we will refer to those as *exact algorithms*. In particular, we consider a family of exact algorithms referred to as multi-agent A$^*$ (MAA$^*$) [Szer *et al.*, 2005; Oliehoek *et al.*, 2008; Oliehoek *et al.*, 2013] based on A$^*$-style heuristic search that finds provably optimal policies.

**Multi-agent A$^*$.** At every time-step (*stage*) of the decision-making problem, a local policy must reason about the history of observations to decide upon an action. Such a mapping is commonly referred to as *decision rules*, and for all policies together, these rules are the *joint decision rules*. A (joint) *policy* is then described by a sequence of (joint) *decision rules*. MAA$^*$ employs a search tree where a path through the tree encodes this sequence of decision rules. Consequently, the leaves of this tree correspond to all policies. MAA$^*$ searches through the tree by incrementally fixing decision rules for the individual stages. Every node corresponds to a partial policy up to time step $t$. As standard in A$^*$, the order in which we explore the tree is steered by an *admissible heuristic* that, for every inner node of the tree, guarantees that we conservatively overapproximate the optimal reward obtained by a policy at the leaves of the subtree. Note that nodes reflect choices for the joint decision rules, i.e., we must choose responses for each agent and for each observation history. This representation yields a search tree with an outdegree that is double exponential in the stage $t$. Avoiding the double exponential blowup and providing a tight admissible heuristic are two primary concerns when developing variations to MAA$^*$.

**Our Approach: Small-step MAA$^*$.** In classical MAA$^*$, expanding nodes is cumbersome due to the double exponential outdegree of nodes. While incremental expansion [Spaan *et al.*, 2011] aims to alleviate the prohibitive outdegree, we prevent this massive outdegree by taking smaller steps in the decision-making process. In particular, we pick actions for each agent and each observation history successively. Compared to MAA$^*$, our search tree limits the outdegree of nodes from double exponential to constant, while increasing the height from linear to exponential. Paired with a tight heuristic, this significantly limits the number of nodes expanded. Furthermore, the depth of the tree is further restricted by using clustering [Oliehoek *et al.*, 2009], which is applicable off-the-shelf to small-step MAA$^*$.

**Admissible Heuristics.** A further important step towards efficient small-step MAA$^*$ is the efficient computation of a tight and admissible heuristic. We base our heuristic on the recursive heuristic from Szer *et al.* [2005]. In a nutshell, to

obtain a heuristic value for a node at stage $t$, we assume that, at stage $t$, the agents once communicate their local observations. This (generally unrealistic) assumption yields a reward value that overapproximates the actual value and thus yields an admissible heuristic. In particular, this assumption allows us to compute a distribution over the current state. We can then compute a heuristic value by solving a Dec-POMDP problem for horizon $h - t$ from the current state. We add two ingredients that make this heuristic effective. First, we can decide to assume communication at a stage $t' < t$, which often allows us to reuse earlier computations *and* provides even tighter results. Second, when computing the heuristic value, we can terminate the computation early and use the current heuristic value as a sound overapproximation of the true value. This means that even the aborted computation yields an admissible heuristic value. With these ingredients, the recursive heuristic has three benefits: 1) It is tighter than heuristics employed by the state-of-the-art. 2) It can be computed on-the-fly and avoids precomputing and storing heuristic values. 3) It avoids dependencies on other algorithms or formalisms, like POMDP solvers or Bayesian games.

**Contributions.** We present *recursive small-step multi-agent A** (RS-MAA*), a fast yet conceptually simple instantiation of MAA*[1]. In particular, RS-MAA* can be concisely implemented and relies neither on precomputed external heuristics nor solvers for cooperative Bayesian games. Its on-the-fly recursive heuristic only computes heuristic values for nodes in the search tree that are expanded. This alleviates a well-known weakness in the state-of-the-art GMAA*-ICE which relates to the memory consumption of storing the heuristic values [Spaan *et al.*, 2011]. Our formulation is compatible with clustering [Oliehoek *et al.*, 2009]. A prototype of our algorithm significantly outperforms the state-of-the-art optimal planners. In particular, to the best of our knowledge[2], RS-MAA* is the first to scale to horizon 12 on DECTIGER (an improvement from 6) and to horizon 1500 on RECYCLING ROBOTS (an improvement from 80).

### Related Work

We review extensions and improvements of MAA* and some recent trends. For a survey and in-depth treatment of various approximate and exact approaches, we refer to the detailed monograph by Oliehoek and Amato [2016].

Szer *et al.* [2005] introduce multi-agent A* (MAA*), investigating heuristics based on different approximations such as an MDP, a POMDP and a recursive MAA*. Thereafter, different heuristics have been proposed to reduce the number of nodes expanded during the search. Oliehoek and Vlassis [2007] introduce a heuristic that models interactions between the agents in one time step using Bayesian games. Oliehoek *et al.* [2008] introduce generalized MAA* (GMAA*), which uses a Bayesian game to determine which child of a given partial policy is the best, but solving the Bayesian game itself is also expensive. Oliehoek *et al.* [2009] propose the GMAA* with incremental clustering (GMAA*-IC) algorithm, which clusters observation histories while expanding the tree to reduce the number of nodes in the tree. This is the first approach to solve DECTIGER (see Example 1) for horizon 5. Finally, Spaan *et al.* [2011] introduce GMAA*-IC with incremental expansion (GMAA*-ICE), which, when expanding a node, finds the best child incrementally by a second (nested, but not recursive) A* search over partially specified Bayesian game policies. This was the first method to solve DECTIGER for horizon 6. We refer to Oliehoek *et al.* [2013] for an in-depth explanation of these approaches. In this paper, we propose a new variation of MAA* that improves the scalability even further, solving DECTIGER with horizon 12.

The state-of-the-art to generate $\epsilon$-optimal solutions reduces the Dec-POMDP to an MDP with continuous state space and uses an adaption of heuristic-search value-iteration for continuous MDPs [Dibangoye *et al.*, 2016]. Recent work has studied different subclasses of Dec-POMDPs to alleviate the complexity of the general method. Amato *et al.* [2019] adds macro actions (also known as option [Pateria *et al.*, 2022; Barto and Mahadevan, 2003]) to the Dec-POMDP, which allows an agent to perform multiple actions without having to coordinate with the remaining agents. Xie *et al.* [2020] study problems with two agents under one-sidedness, that is, where one of the agents observes the actions and observations of the other agent. Finally, Lauri *et al.* [2019; 2020] investigate the information gathering task where the reward function is based on the joint belief of the agents.

The idea of splitting up steps in the A* search tree into smaller steps to limit the outdegree of the tree was proposed before by Cazenave [2010], and applied to multi-agent path finding and multiple sequence alignment. However, it has not been applied before to solving Dec-POMDPs.

## 2 Problem Statement

$\Delta(X)$ denotes distributions over finite sets $X$.

**Definition 1** (Dec-POMDP)**.** *A* Dec-POMDP *is a tuple* $\langle \mathcal{D}, \mathcal{S}, \boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{O}}, b, T, R, O \rangle$ *with a set* $\mathcal{D} = \{1, \dots, n\}$ *of* $n$ agents*, a finite set* $\mathcal{S}$ *of* states*, a set* $\boldsymbol{\mathcal{A}} = \times_{i \in \mathcal{D}} \mathcal{A}_i$ *of* joint actions*, where each* $\mathcal{A}_i$ *is the finite set of* local actions *of agent* $i$*, and a set* $\boldsymbol{\mathcal{O}} = \times_{i \in \mathcal{D}} \mathcal{O}_i$ *of* joint observations*, where each* $\mathcal{O}_i$ *is the finite set of* local observations *of agent* $i$*. The* transition function $T\colon \mathcal{S} \times \boldsymbol{\mathcal{A}} \to \Delta(\mathcal{S})$ *defines the* transition probability $\Pr(s' \mid s, \mathbf{a})$*,* $b \in \Delta(\mathcal{S})$ *is the* initial belief*,* $R\colon \mathcal{S} \times \boldsymbol{\mathcal{A}} \to \mathbb{R}$ *is the* reward *function, and the* observation function $O\colon \boldsymbol{\mathcal{A}} \times \mathcal{S} \to \Delta(\boldsymbol{\mathcal{O}})$ *defines the* observation probability $\Pr(\mathbf{o} \mid \mathbf{a}, s')$.

A Dec-POMDP describes a system that initially starts in a state $s^0$ with probability $b(s^0)$. At each time step $t$, each agent $i$ selects a local action $a_i^t$. Together, this yields a joint action $\mathbf{a}^t = \langle a_1^t, \dots, a_n^t \rangle$ and a reward $r^t = R(s^t, \mathbf{a}^t)$. The system evolves to the next state $s^{t+1}$ with probability $\Pr(s^{t+1} \mid s^t, \mathbf{a}^t)$. The observation $\mathbf{o}^{t+1}$ is obtained with probability $\Pr(\mathbf{o}^{t+1} \mid \mathbf{a}^t, s^{t+1})$, and each agent $i$ receives the local observation $o_i^{t+1}$. We call the sequence of joint observations $\boldsymbol{\tau} = \mathbf{o}^1\mathbf{o}^2 \dots \mathbf{o}^t$ a *joint observation history* (JOH) and the sequence of local observations $\tau_i = o_i^1 o_i^2 \dots o_i^t$ of agent $i$

[1] Proofs and source code are given in the supplementary material available at https://zenodo.org/record/7949016

[2] Based on masplan.org and our experiments.

a *local observation history* (LOH). We denote the first $d$ observations of $\boldsymbol{\tau}$ by $\mathsf{pre}(\boldsymbol{\tau}, d)$. We denote the set of all LOHs of length exactly $\ell$ and at most $\ell$ by $\mathcal{O}_i^{\ell}$ and $\mathcal{O}_i^{\leq \ell}$, respectively.

We are interested in offline planning: How should the individual agents act locally such that, jointly, they optimize the cumulative reward up until horizon $h$? A *local policy* for agent $i$ maps LOHs for that agent to a local action, formally: $\pi_i \colon \mathcal{O}_i^{\leq h-1} \to \mathcal{A}_i$. A *joint policy* is a tuple of local policies $\boldsymbol{\pi} = \langle \pi_1, \cdots, \pi_n \rangle$, and $\Pi$ denotes the set of all joint policies.

Given a joint policy $\boldsymbol{\pi}$, we can define the value of executing this policy in the initial belief $b$ over a horizon $h$ as:

$$V_{\boldsymbol{\pi}}(b, h) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{h-1} R(s^t, \mathbf{a}^t) \middle| s^0 \sim b\right],$$

where $s^t$ and $\mathbf{a}^t$ are the state and joint action at time step $t$.

**Problem statement:** Given a Dec-POMDP and a horizon $h$, find a joint policy $\boldsymbol{\pi}$ that maximizes $V_{\boldsymbol{\pi}}(b, h)$.

**Example 1** (Decentralized tiger problem)**.** *The* DECTIGER *problem [Nair* et al.*, 2003] is a multi-agent variation of the Tiger problem [Kaelbling* et al.*, 1998]. In this problem, two agents are in front of two doors. A tiger is behind the left or the right door, which indicates the state of the environment (*TL *or* TR*), and a treasure is behind the other. Each agent has three actions: listen (*Li*), open the left door (*OL*), or open the right door (*OR*). Listening gives a noisy observation regarding the location of the tiger (*HL *or* HR*). In the multi-agent variation, the dynamics depend on the joint action. The state resets if any agent opens a door, in which case both agents get an uninformative observation (uniform distribution over the observations). If only one agent opens the door with the treasure, they receive a positive reward. But, if both agents find the treasure, they receive a larger reward. A large penalty is given for opening the door with the tiger.*

The DECTIGER problem (Example 1) illustrates the challenges in cooperating to solve a Dec-POMDP. In this case, besides finding a policy to gather information, the agents must also coordinate in the offline phase deciding the right moment to open a door. Without such coordination, the actions of one agent could compromise the belief of the remaining agents.

## 3 Small-step Multi-agent A*

In this section, we frame the search for an optimal policy by an incremental heuristic search over partial policies.

**Ordered LOHs.** For our algorithm, we explore LOHs in a fixed order. Intuitively, we order the LOHs of all agents as follows: first by stage (length of the observation history), then by agent, and then lexicographically by observation history. This ordering is formalized in the following definition.

**Definition 2.** *Given a total order $\preceq_i$ on $\mathcal{O}_i$, we define a total order $\preceq$ on $\bigcup_{i=1}^{n} \mathcal{O}_i^{\leq h-1}$ such that $\tau_i^t \preceq \tau_j^{t'}$ if*

$$\big(t < t'\big) \text{ or } \big(t = t' \wedge i < j\big) \text{ or } \big(t = t' \wedge i = j \wedge \tau_i^t \preceq_i^{lex} \tau_j^{t'}\big).$$

Throughout the paper, we use this fixed order for LOHs.

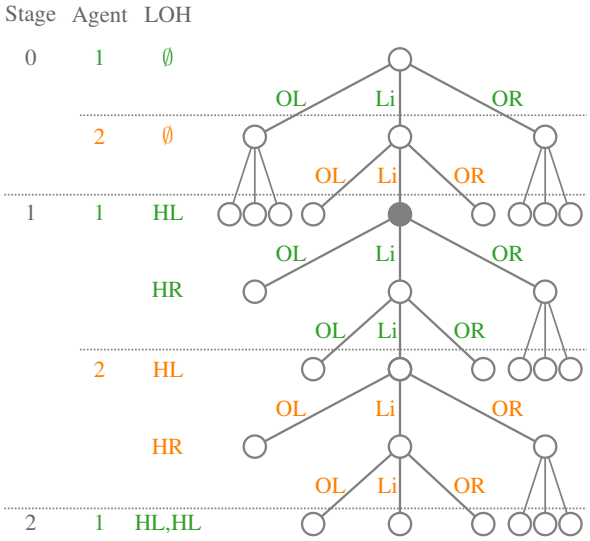


Figure 1: A partial tree of the DECTIGER problem at stage 1, where we expanded all nodes in stage 0 but only a subset of the nodes in stage 1 after expanding the gray node (filled).

**Partial policies.** A *local partial policy* for agent $i$ is a partial function $\varphi_i \colon \mathcal{O}_i^{\leq h-1} \to \mathcal{A}_i$. A *partial policy* is a tuple of local partial policies $\boldsymbol{\varphi} = \langle \varphi_1, \ldots, \varphi_n \rangle$. The length of a partial policy is $\ell$, if every local partial policy $\varphi_i$ is defined on only the first $\ell$ LOHs (of any agent) according to the order in Definition 2. Hence, a partial policy of length $\ell$ is defined on $\ell$ LOHs in total, summed over all local partial policies. The *stage* $\sigma(\boldsymbol{\varphi})$ of a partial policy $\boldsymbol{\varphi}$ is $t$, if the policy is defined for all LOHs of length $t-1$, but not for all LOHs of length $t$. The set of all partial policies is $\Phi$. A partial policy $\boldsymbol{\varphi}'$ *extends* a partial policy $\boldsymbol{\varphi}$, denoted by $\boldsymbol{\varphi} <_E \boldsymbol{\varphi}'$ if $\boldsymbol{\varphi}'$ agrees with $\boldsymbol{\varphi}$ on all LOHs for which $\boldsymbol{\varphi}$ specifies the action. Formally,

$$\boldsymbol{\varphi} <_E \boldsymbol{\varphi}' \iff \forall i \in \mathcal{D}. \forall \tau_i \in \mathcal{O}_i^{\leq h-1}. \varphi_i(\tau_i) \in \{\bot, \varphi_i'(\tau_i)\},$$

where $\varphi_i(\tau_i) = \bot$ denotes that $\varphi_i$ is not defined at $\tau_i$. The *extensions of* $\boldsymbol{\varphi}$ are the fully-specified policies extending $\boldsymbol{\varphi}$:

$$E(\boldsymbol{\varphi}) = \big\{\boldsymbol{\pi} \in \Pi \mid \boldsymbol{\varphi} <_E \boldsymbol{\pi}\big\}.$$

**Search tree.** We define a search tree with an empty policy at the root and fully specified policies at the leaves.

**Definition 3** (Small-step search tree)**.** *In a small-step search tree for a Dec-POMDP, the nodes are the partial policies, the root node is the empty policy, and the children of a partial policy $\boldsymbol{\varphi}$ of length $\ell$ are exactly the partial policies $\boldsymbol{\varphi}'$ such that $\boldsymbol{\varphi} <_E \boldsymbol{\varphi}'$ and the length of $\boldsymbol{\varphi}'$ is $\ell + 1$.*

**Example 2.** *Consider the (partial) tree in Figure 1. For the highlighted node, we add three children, each reflecting the choice of an action by agent 1 after observing HL. In this tree, we can extract the policy of a node by backtracking. For example, the policy in the bottom-left node is $\langle \{\emptyset \mapsto Li, HL \mapsto Li, HR \mapsto Li\}, \{\emptyset \mapsto Li, HL \mapsto Li, HR \mapsto OL\}\rangle$.*

**A*.** Small-step MAA* applies A* on the small-step search tree. A* successively expands the nodes in the tree [Russell

and Norvig, 2020]. To guide the expansion of nodes, a heuristic $Q\colon \Phi \to \mathbb{R}$ is used. The A$^*$ algorithm keeps a list of open nodes, which initially only contains the root. In each step, the open node with the highest heuristic value is selected, this node is removed from the list of open nodes, and all children are added to the list of open nodes. Once a leaf is selected as node with the highest heuristic value, that node is returned as the final solution. The algorithm is exact if the heuristic is *admissible*, i.e. if $Q$ is an overapproximation:

$$Q(\boldsymbol{\varphi}) \geq \max_{\boldsymbol{\pi} \in E(\boldsymbol{\varphi})} V_{\boldsymbol{\pi}}(b, h).$$

**Difference to classical MAA$^*$.** Classical MAA$^*$ constructs a stage-$(t+1)$ policy in one step from a stage-$t$ policy, specifying actions for all length-$t$ LOHs for all $n$ agents. As a result, each stage-$t$ policy has $O\left(|\mathcal{A}_*|^{n|\mathcal{O}_*|^t}\right)$ children in the A$^*$ search tree, where $|\mathcal{A}_*|$ and $|\mathcal{O}_*|$ denote the sizes of the largest local action and observation sets, respectively. In small-step MAA$^*$, this is split up in $n|\mathcal{O}_*|^t$ levels, where each node has up to $|\mathcal{A}_*|$ children and just adds an action for one agent for one particular LOH.

**Example 3.** *In the tree from Fig. 1, RS-MAA$^*$ reaches only 12 nodes in stage 1 and 6 nodes in stage 2 by expanding only 6 nodes below the gray node in stage 1. By contrast, MAA$^*$ would have reached 81 nodes at stage 2 after expanding the gray node.*

Compared to MAA$^*$, our search tree limits the outdegree of the nodes from double exponential to constant, at the cost of increasing the tree height from linear to exponential.

## 4 Admissible Recursive Heuristics

Next, we introduce two families of admissible heuristics.

**Generalized policies.** To define admissible heuristics, we introduce more general types of policies. First, we allow local policies $\pi_i$ for each agent to depend on the joint observation history. A *generalized local policy* $\pi_i\colon \boldsymbol{\mathcal{O}}^{\leq h-1} \to \mathcal{A}_i$ maps JOHs to a local action. A *generalized (joint) policy* is a tuple of generalized local policies (one for each agent). Let $\Pi_{\mathsf{gen}}$ denote the set of all generalized policies. As before, a generalized policy $\boldsymbol{\pi}$ *extends* a partial policy $\boldsymbol{\varphi}$, denoted by $\boldsymbol{\varphi} <_E \boldsymbol{\pi}$, if $\pi_i$ agrees with $\varphi_i$ on all OHs corresponding to an LOH for which $\varphi_i$ specifies the action. Formally,

$$\boldsymbol{\varphi} <_E \boldsymbol{\pi} \iff \forall i \in \mathcal{D}. \forall \boldsymbol{\tau} \in \boldsymbol{\mathcal{O}}^{\leq h-1}.\ \varphi_i(\tau_i) \in \{\bot, \pi_i(\boldsymbol{\tau})\}.$$

The generalized extensions of a partial policy $\boldsymbol{\varphi}$ are the set of all fully specified generalized policies agreeing with $\boldsymbol{\varphi}$:

$$E_{\mathsf{gen}}(\boldsymbol{\varphi}) = \left\{\boldsymbol{\pi} \in \Pi_{\mathsf{gen}} \mid \boldsymbol{\varphi} <_E \boldsymbol{\pi}\right\}.$$

**Fixed-depth heuristics.** First, we write the Dec-POMDP optimization problem over generalized policies as

$$\begin{aligned} &\max_{\boldsymbol{\pi} \in \Pi_{\mathsf{gen}}} V_{\boldsymbol{\pi}}(b, h) \\ &\text{subject to} \quad \tau_i = \widetilde{\tau}_i \implies \pi_i(\boldsymbol{\tau}) = \pi_i(\widetilde{\boldsymbol{\tau}}) \\ &\text{for all } i \in \mathcal{D},\ \boldsymbol{\tau}, \widetilde{\boldsymbol{\tau}} \in \boldsymbol{\mathcal{O}}^v \text{ with } v \in \{0, \dots, h-1\}. \end{aligned} \tag{1}$$

Intuitively, we optimize over the generalized policies but add constraints that ensure agents have to take equal actions on *joint* OHs that they cannot distinguish, therefore any feasible solution encodes a decentralized policy. By relaxing some of the constraints, we achieve optimization problems that describe an upper bound on the optimal Dec-POMDP value, thus forming admissible heuristics.

**Definition 4.** *Let $\boldsymbol{\varphi}$ be a partial policy. The Dec-POMDP heuristic $Q_{\mathsf{Dec},d}(\boldsymbol{\varphi})$ of depth $d$ is defined as*

$$\begin{aligned} &\max_{\boldsymbol{\pi} \in E_{\mathsf{gen}}(\boldsymbol{\varphi})} V_{\boldsymbol{\pi}}(b, h) \quad \textit{subject to} \\ &\quad (\mathsf{pre}(\boldsymbol{\tau}, d) = \mathsf{pre}(\widetilde{\boldsymbol{\tau}}, d) \wedge \tau_i = \widetilde{\tau}_i) \implies \pi_i(\boldsymbol{\tau}) = \pi_i(\widetilde{\boldsymbol{\tau}}) \\ &\quad \textit{for all } i \in \mathcal{D},\ \boldsymbol{\tau}, \widetilde{\boldsymbol{\tau}} \in \boldsymbol{\mathcal{O}}^v \textit{ with } v \in \{d, \dots, h-1\}. \end{aligned}$$

The heuristic $Q_{\mathsf{Dec},d}$ allows local policies to condition their choices after stage $d$ on the local observation of other agents up to stage $d$. Notice that for $d = 0$, the Dec-POMDP heuristic corresponds to the original optimization problem (1).

For stage-$t$ partial policies with $d \leq t$, this heuristic is easier to compute than the Dec-POMDP value itself. Since $d \leq t$, the policy for the first $d$ stages is specified by $\boldsymbol{\varphi}$, and we can compute the realized reward $V_{\boldsymbol{\varphi}}(b, d)$ up to stage $d$ as $V_{\boldsymbol{\varphi}}(b, d) = \mathbb{E}_{\boldsymbol{\varphi}}\left[\sum_{t=0}^{d-1} r^t\right]$. Revealing the first $d$ joint observations $\boldsymbol{\tau} \in \boldsymbol{\mathcal{O}}^d$ provides the agents with a new joint belief $b_{\boldsymbol{\tau}}$ after $d$ stages, so for each length-$d$ JOH we solve a Dec-POMDP with horizon $h - d$ and the new belief $b_{\boldsymbol{\tau}}$:

$$Q_{\mathsf{Dec},d}(\boldsymbol{\varphi}) = V_{\boldsymbol{\varphi}}(b, d) + \sum_{\boldsymbol{\tau} \in \boldsymbol{\mathcal{O}}^d} \Pr(\boldsymbol{\tau}) \cdot \max_{\boldsymbol{\pi} \in E(\boldsymbol{\varphi}|_{\boldsymbol{\tau}})} V_{\boldsymbol{\pi}}(b_{\boldsymbol{\tau}}, h-d), \tag{2}$$

where $\boldsymbol{\varphi}|_{\boldsymbol{\tau}}$ is the shortened partial policy for which $\boldsymbol{\varphi}$ has already specified an action: $(\boldsymbol{\varphi}|_{\boldsymbol{\tau}})_i(\widetilde{\tau}) = \varphi_i(\tau_i \widetilde{\tau})$. Note that $\boldsymbol{\varphi}$ may have already specified actions for LOHs of length larger than $d$. These actions are then also specified in problems with horizon $h - d$ through the shortened partial policy $\boldsymbol{\varphi}|_{\boldsymbol{\tau}}$.

When computing $Q_{\mathsf{Dec},d}$, we allow each agent to also base their decision on these first $d$ joint observations. The smaller $d$, the less information agents have, so the tighter the heuristic is. This provides the intuition for the following lemma.

**Lemma 1.** *Let $\boldsymbol{\varphi}$ be a partial policy. If $d' \leq d$, then $Q_{\mathsf{Dec},d'}(\boldsymbol{\varphi}) \leq Q_{\mathsf{Dec},d}(\boldsymbol{\varphi})$.*

*Proof.* The definitions of $Q_{\mathsf{Dec},d'}$ and $Q_{\mathsf{Dec},d}$ only differ in the condition used in the antecedent of the constraint. Since the condition $\mathsf{pre}(\boldsymbol{\tau}, d) = \mathsf{pre}(\widetilde{\boldsymbol{\tau}}, d)$ is stronger than $\mathsf{pre}(\boldsymbol{\tau}, d') = \mathsf{pre}(\widetilde{\boldsymbol{\tau}}, d')$, the maximization problem for $Q_{\mathsf{Dec},d}$ effectively removes some constraints from the maximization problem for $Q_{\mathsf{Dec},d'}$.

Since a less constrained maximization problem has a higher (or equal) optimal value, the result follows. □

**Theorem 1.** *$Q_{\mathsf{Dec},d}$ is admissible.*

*Proof.* We can write $\max_{\boldsymbol{\pi} \in E(\boldsymbol{\varphi})} V_{\boldsymbol{\pi}}(b)$ as

$$\begin{aligned} &\max_{\boldsymbol{\pi} \in E_{\mathsf{gen}}(\boldsymbol{\varphi})} V_{\pi}(b, h) \quad \text{subject to} \\ &\quad \tau_i = \widetilde{\tau}_i \implies \pi_i(\boldsymbol{\tau}) = \pi_i(\widetilde{\boldsymbol{\tau}}) \\ &\quad \text{for all } i \in \mathcal{D},\ \boldsymbol{\tau}, \widetilde{\boldsymbol{\tau}} \in \boldsymbol{\mathcal{O}}^s \text{ with } s \in \{0, \dots, h-1\}. \end{aligned}$$

Since the condition $\mathsf{pre}(\boldsymbol{\tau}, 0) = \mathsf{pre}(\widetilde{\boldsymbol{\tau}}, 0)$ is empty, it follows that $\max_{\boldsymbol{\pi} \in E(\boldsymbol{\varphi})} V_{\boldsymbol{\pi}}(b) = Q_{\mathsf{Dec},0}(\boldsymbol{\varphi})$. By Lemma 1, it follows that $Q_{\mathsf{Dec},d}(\boldsymbol{\varphi}) \geq Q_{\mathsf{Dec},0}(\boldsymbol{\varphi}) = \max_{\boldsymbol{\pi} \in E(\boldsymbol{\varphi})} V_{\boldsymbol{\pi}}(b)$ for all partial policies $\boldsymbol{\varphi}$, so $Q_{\mathsf{Dec},d}$ is admissible. □

**Variable-depth heuristics.** The smaller $d$, the harder the heuristic $Q_{\mathsf{Dec},d}$ is to compute, since (by Eq. 2) we have to solve Dec-POMDPs with a larger horizon. On the other hand, we do not want to compute $Q_{\mathsf{Dec},d}(\boldsymbol{\varphi})$ if $\sigma(\boldsymbol{\varphi}) < d$, because then Eq. 2 cannot be used. We suggest to use $Q_{\mathsf{Dec},d}$ unless $\sigma(\boldsymbol{\varphi}) < d$, in which case we use $Q_{\mathsf{Dec},\sigma(\boldsymbol{\varphi})}(\boldsymbol{\varphi})$.

Towards this idea, for some $\boldsymbol{\varphi}$ we would like to use heuristics $Q_{\mathsf{Dec},\sigma(\boldsymbol{\varphi})}(\boldsymbol{\varphi})$. However, this heuristic is suboptimal in the following sense: Along the search tree, whenever we increase the stage, we switch to a looser heuristic. Consequently, it is possible that the heuristic value of a node is larger than that of its parent, despite more actions being fixed. To fix this weakness, we may simply take the minimum with the heuristic value of the parent. Formally, let the parent $\boldsymbol{\varphi}^-$ of a partial policy $\boldsymbol{\varphi}$ of length $\ell > 0$ be the partial policy of length $\ell - 1$ such that $\boldsymbol{\varphi}^- <_E \boldsymbol{\varphi}$.

**Definition 5.** *Let $\boldsymbol{\varphi}$ be a partial policy. The heuristic $Q_d$ is recursively defined by $Q_d(\boldsymbol{\varphi}) = \infty$ if $\sigma(\boldsymbol{\varphi}) = 0$, and if $\sigma(\boldsymbol{\varphi}) > 0$ we define*

$$Q_d(\boldsymbol{\varphi}) = \min\{Q_{\mathsf{Dec},\min\{\sigma(\boldsymbol{\varphi}),d\}}(\boldsymbol{\varphi}), Q_d(\boldsymbol{\varphi}^-)\}.$$

We can also set $d = \infty$; then $\min\{\sigma(\boldsymbol{\varphi}), d\} = \sigma(\boldsymbol{\varphi})$.

**Corollary 1.** *$Q_d$ is admissible for $d \geq 1$.*

This follows from Thm. 1 and $Q_d(\boldsymbol{\varphi}) \geq Q_{\mathsf{Dec},1}(\boldsymbol{\varphi})$, $d \geq 1$.

**Comparing heuristics.** The recursive Dec-POMDP heuristic proposed by Szer *et al.* [2005] can be written as $Q_{\mathsf{Dec},\sigma(\boldsymbol{\varphi})}(\boldsymbol{\varphi})$. It does exhibit the weakness mentioned above that we fix in the definition of $Q_d$. Two other commonly used heuristics in the literature [Oliehoek and Vlassis, 2007; Oliehoek *et al.*, 2013] are the $Q_{\mathsf{POMDP}}$ and $Q_{\mathsf{BG}}$ heuristic. For the $Q_{\mathsf{POMDP}}$ heuristic, it is assumed that the agents can communicate and hence have access to all joint observations when making their decisions. For the $Q_{\mathsf{BG}}$ heuristic, it is assumed that agents can communicate with one step of delay. The $Q_{\mathsf{POMDP}}$ and $Q_{\mathsf{BG}}$ heuristics can be captured as instances of the generalized policy optimization problem. This allows to compare our heuristics to $Q_{\mathsf{POMDP}}$ and $Q_{\mathsf{BG}}$.

**Theorem 2.** *Let $\boldsymbol{\varphi}$ be a partial policy. Then $Q_\infty(\boldsymbol{\varphi}) \leq Q_{\mathsf{POMDP}}(\boldsymbol{\varphi})$ and $Q_{\mathsf{Dec},\sigma(\boldsymbol{\varphi})-1}(\boldsymbol{\varphi}) \leq Q_{\mathsf{BG}}(\boldsymbol{\varphi})$.*

*Proof.* The (centralized) POMDP policy optimization problem is the unconstrained version of the optimization problem (1), from which the first inequality follows.

The one-step delay communication problem used to compute $Q_{\mathsf{BG}}$ corresponds constraints of the form

$$(\mathsf{pre}(\boldsymbol{\tau}, s{-}1) = \mathsf{pre}(\widetilde{\boldsymbol{\tau}}, s{-}1) \wedge \tau_i = \widetilde{\tau}_i) \implies \pi_i(\boldsymbol{\tau}) = \pi_i(\widetilde{\boldsymbol{\tau}}).$$

The antecedent of the constraint $\mathsf{pre}(\boldsymbol{\tau}, t{-}1) = \mathsf{pre}(\widetilde{\boldsymbol{\tau}}, t{-}1) \wedge \tau_i = \widetilde{\tau}_i$ used in $Q_{\mathsf{Dec},t-1}(\boldsymbol{\varphi})$ is weaker for $s \geq t$ (leading to a more constrained problem), whereas all actions corresponding to OHs of length $s \leq t - 1$ are already fixed (and satisfy the Dec-POMDP constraint, so certainly also the constraint for $Q_{\mathsf{BG}}$). This shows the second inequality. □

## 5 Abandoning Heuristic Computations Early

In this section, we consider a cheap variation of the recursive heuristic that provides an upper bound on $Q_d$ and is thus admissible. We algorithmically obtain this heuristic by early termination of the computation of the recursive heuristic.

Assume we are computing a heuristic value for the partial policy $\boldsymbol{\varphi}$. By Eq. 2, this can be done by solving Dec-POMDPs with a smaller horizon, e.g., using an A$^*$ algorithm with an admissible heuristic $Q$. Due to the nature of A$^*$, the highest heuristic value of an open node is an upper bound for the expected reward of an optimal policy. Hence, we can abandon a computation after exploring $M$ nodes and return the highest heuristic value of an open node at that point. We call this value $L_M(\boldsymbol{\varphi}, Q)$. If the heuristic is such that the heuristic value of node $\boldsymbol{\varphi}$ is always at most the value of its parent $\boldsymbol{\varphi}^-$, then the A$^*$ algorithm explores nodes in order of decreasing values. $L_M(\boldsymbol{\varphi}, Q)$ will be the $M$th largest value in the tree, or, if A$^*$ terminates within $M$ steps, the value of the fully specified policy. Next, we formalize $L_M(\boldsymbol{\varphi}, Q)$.

**Definition 6.** *Consider a search tree with heuristic $Q\colon \Phi \to \mathbb{R}$ and some fixed partial policy $\boldsymbol{\varphi}$. We define the set of candidate policies $\Phi_c = \{\boldsymbol{\varphi} <_E \boldsymbol{\varphi}' \wedge \sigma(\boldsymbol{\varphi}') \geq 1\}$. Let $w$ be an ordered list over $\Phi_c$ such that $Q(w_i) \geq Q(w_j)$ for $i < j$, and $k^* = \min\{k \mid w_k \in \Pi\}$ is the smallest index where $w$ contains a fully specified policy. With $K = \min\{k^*, M\}$, we define $L_M(\boldsymbol{\varphi}, Q) = Q(w_K)$.*

Indeed, $w_{k^*} = \arg\max_{\boldsymbol{\pi} \in \Phi_c \cap \Pi_{\mathsf{gen}}} Q(\boldsymbol{\pi})$, i.e., the fully specified policy in $\Phi_c$ with the largest heuristic value is the first fully specified policy in the list. Furthermore, we know that $Q(w_K) \geq Q(w_M)$. Note that we only count partial policies with stage larger than 0, as we do not want to compute heuristic values for partial policies with stage 0; formally, we assign these nodes an infinite heuristic value.

Recall that the heuristic $Q_d$ can be written in terms of optimal values of Dec-POMDPs with a lower horizon. For the heuristic $Q_{M,d}$, we substitute the precise values with the values obtained after early termination.

**Definition 7.** *First, $Q_{M,d}(\boldsymbol{\varphi}) = \infty$ for $\sigma(\boldsymbol{\varphi}) = 0$. For $\sigma(\boldsymbol{\varphi}) \geq 1$, we define $Q_{M,d}(\boldsymbol{\varphi})$ inductively over the horizon. For $h = 1$, partial policies with $\sigma(\boldsymbol{\varphi}) \geq 1$ are fully specified and $Q_{M,d}(\boldsymbol{\varphi}) = V_{\boldsymbol{\varphi}}(b, 1)$. For $h = \ell + 1$, $Q_{M,d}(\boldsymbol{\varphi})$ is*

$$\min\left\{V_{\boldsymbol{\varphi}}(b,t) + \sum_{\boldsymbol{\tau} \in \boldsymbol{\mathcal{O}}^t} \left[\Pr(\boldsymbol{\tau}) L_M(\boldsymbol{\varphi}|_{\boldsymbol{\tau}}, Q_{M,d})\right], Q_{M,d}(\boldsymbol{\varphi}^-)\right\},$$

*where $t = \min\{d, \sigma(\boldsymbol{\varphi})\}$.*

Note that the $Q_{M,d}$ values used for computing $L_M(\boldsymbol{\varphi}|_{\boldsymbol{\tau}})$ are values with horizon $\ell + 1 - t \leq \ell$, so these are defined already. Hence, the recursion is well-founded. Naturally aborting earlier leads to values that are less tight:

**Lemma 2.** *Let $\boldsymbol{\varphi}$ be a partial policy. Then*

$$Q_d(\boldsymbol{\varphi}) \leq Q_{M',d}(\boldsymbol{\varphi}) \leq Q_{M,d}(\boldsymbol{\varphi}) \text{ for all } M' \geq M.$$

A proof of this lemma is given in the supplementary material.

## 6 Recursive Small-Step Multi-Agent A$^*$

In a nutshell, Recursive Small-step Multi-Agent A$^*$ (RS-MAA$^*$) applies the $Q_{M,d}$ heuristic with suitable $M$ and $d$ on the search tree in Def. 3. Here, we discuss the choice for $M$ and $d$, as well as other improvements that help scalability.

**Clustering.** Following Oliehoek *et al.* [2009], we employ incremental lossless clustering. Clustering computes groups of LOHs for which the same action choice is provably optimal. We use incremental clustering to compute the clustered LOHs for stage $t+1$ once a policy is expanded where each LOH corresponding of length $t$ is assigned an action, but no LOH of length $t+1$. Each edge in the small-step search tree now corresponds to assigning an action to each LOH in a clustered set of LOHs.

**Storing heuristics.** A key feature of RS-MAA$^*$ is that heuristics are computed on-the-fly, in contrast to precomputing them. Recall that we solve the Dec-POMDP for smaller horizons and different initial beliefs. The combination of horizon and initial belief (and initial part of the policy) can appear in multiple computations. To avoid recomputing those heuristic values, we memoize all values. To enable finding values that we already computed, we characterize each heuristic as a tuple containing 1) the horizon, 2) the initial belief, 3) an index describing the partial policy, and 4) a list of indices describing the clustering structure at each stage. We additionally reuse computations for which we completed the computation of $Q_d$, i.e., for which we did not terminate early. Those values can be reused when evaluating extended policies. In particular, whenever we do not terminate early, we not only find a heuristic value, but also a fully specified policy $\boldsymbol{\pi}$. Then, for all $\boldsymbol{\varphi}'$ satisfying $\boldsymbol{\varphi} <_E \boldsymbol{\varphi}' <_E \boldsymbol{\pi}$ we have $Q(\boldsymbol{\varphi}') = V(\boldsymbol{\pi}, \sigma(\boldsymbol{\pi}))$. Hence, we can also store the heuristic value for all such $\boldsymbol{\varphi}'$ without explicitly computing them.[3]

**Dynamically abandoning heuristic computations early.** In Section 5, we have shown how heuristic computations can be terminated early based on a fixed maximum number of iterations $M$. We can go further and also terminate the heuristic computation of $Q(\boldsymbol{\varphi})$ when the highest heuristic value of an open node is much smaller than the heuristic value of the parent $u = Q(\boldsymbol{\varphi}^-)$, which is an upper bound for $Q(\boldsymbol{\varphi})$. Specificially, we terminate the heuristic computation after $M$ iterations or if the new heuristic value $v$ satisfies $v \leq u - \alpha \max\{|u|, 1\}$, where $\alpha$ is a parameter, called the *threshold*. The intuition for this is that a large drop in computing the heuristic value is sufficient to avoid expanding that node again (at least for a long time), so spending more time computing heuristics for this node is likely a waste of time.

**Dynamically increasing depth.** While Lemma 1 tells us that $Q_d(\boldsymbol{\varphi}) \leq Q_{d'}(\boldsymbol{\varphi})$ if $d \leq d'$, the inequality $Q_{M,d}(\boldsymbol{\varphi}) \leq Q_{M,d'}(\boldsymbol{\varphi})$ does not necessarily hold, i.e., taking the lowest depth does not always lead to the tightest heuristic when abandoning heuristic computations early. Hence, we actually compute $Q_{d'}(\boldsymbol{\varphi})$ for all $d' \geq d$. Computing the heuristic is much more expensive for lower depths, the additional runtime cost of computing heuristics for higher depths is negligible. Each time we expand a node $\boldsymbol{\varphi}$, we update $d$ to $\arg\min_{d'} Q_{d'}(\boldsymbol{\varphi})$ (taking the largest value of $d'$ in case of a tie) for all descendants of that node.

**Last stage.** We treat nodes almost at the bottom of the search tree differently. To compute actions for the last agent for the last stage, we can compute the best fully specified policy that is a descendant of this policy directly. For each LOH of the last agent, we can compute the best action to take independently by computing the expected reward of each action according to the joint multi-agent belief.

**Storing distributions.** To compute policy values and the clustering structure, we need to work with the full joint distribution $\Pr(\boldsymbol{\tau}, \mathbf{o}, s)$ and its conditional distributions. It turns out to be more efficient to instead store the marginal probabilities $\Pr(\boldsymbol{\tau}, \mathbf{o})$ and indices of conditional distributions over states $\Pr(s \mid \boldsymbol{\tau}, \mathbf{o})$, where the map between indices of state distributions and the corresponding distributions is stored globally. While in the worst case this takes more space than simply storing the joint distribution $\Pr(\boldsymbol{\tau}, \mathbf{o}, s)$, in practice many of the conditional distributions $\Pr(s \mid \boldsymbol{\tau}, \mathbf{o})$ are equal, thereby saving space. Moreover, using the indices allows for easy caching.

**Choosing parameters.** The most important parameter to set is $d$. Lower values of $d$ give tighter heuristics but are more expensive to compute. Setting the number of iterations $M$ too low leads to an overly loose heuristic. However, setting $M$ high leads to generally higher computation times. Due to the recursive nature of the algorithm, however, the time can scale as badly as $O(M^h)$. Typically, the computational effort is much less, in particular due to the tighter heuristic and the fact that recursive heuristics at later stages are easier to compute. Furthermore, some heuristics already find a complete policy before $M$ iterations and most heuristics do not nest $h$ layers deep (except if $d = 1$). The effect of $\alpha$ is similar to the effect of $M$, but harder to quantify. In particular, higher values of $\alpha$ yield tighter heuristics and are harder to compute.

## 7 Empirical Evaluation

This section provides an empirical evaluation of RS-MAA$^*$ on a set of standard benchmarks, in comparison with GMAA$^*$-ICE [Oliehoek *et al.*, 2013], the current state-of-the-art exact solver for Dec-POMDPs. Furthermore, we provide an ablation study, showing the effects of clustering, computing the best response for the last agent in the last stage directly, the choice of heuristic ($Q_d$), and abandoning the computation of the recursive heuristic early based on a maximum number of iterations ($M$) or a threshold ($\alpha$).

**Setup.** We implemented a stand-alone prototype of RS-MAA$^*$ in Python 3 that does not have any dependencies. We execute it using PyPy.[4] For GMAA$^*$-ICE, we use the C++ implementation in the MADP toolbox [Oliehoek *et al.*, 2017].

[3] We only store it for policies $\boldsymbol{\varphi}'$ of the same stage as clustering in later stages may depend on the observations that are revealed.

[4] The supplementary material contains the source code and scripts to reproduce the empirical results.

We use two versions of the $Q_{\mathsf{BG}}$ heuristic[5] using a hybrid representation and using tree-based incremental pruning. We also show results for $Q_{\mathsf{MDP}}$. Both implementations use floating point numbers and a threshold of $10^{-12}$ for comparing probabilities and rewards. We run RS-MAA$^*$ in two configurations, outlined in the paragraph below. For RS-MAA$^*$, we report the total running time, for GMAA$^*$-ICE, we give the 'overall time' as reported by solver. Reported timings are an average over 3 runs. We did not observe significant variations in the timings. All experiments were ran on 3.7GHz Intel Core i9 CPU running Ubuntu 22.04.1 LTS. We limited each process to 16GB of memory and 1 hour of CPU time.

**Hyperparameter selection.** Our algorithm has three hyperparameters, $d, M, \alpha$, discussed in Sec. 6. For most problems, $Q_1$ and $Q_2$ are too expensive to compute, but $Q_3$ is a good compromise. For easier problems, $Q_\infty$ also works well. In a preliminary evaluation using $M \in \{100, 200, 400\}$ and $\alpha \in \{0.1, 0.2\}$, we observed that $M = 100$ sometimes expands much more nodes than $M = 200$, but increasing $M$ to 400 did not reduce the number of nodes (significantly) compared to $M = 200$, and hence typically increased running times. Hence, we set $M = 200$. The threshold $\alpha$ had a small effect, so we conservatively set it to 0.2.

**Benchmarks.** We used the standard benchmarks from the literature: DECTIGER [Nair *et al.*, 2003], FIREFIGHTING [Oliehoek *et al.*, 2008] (3 fire levels, 3 or 4 houses), GRID with two observations [Amato *et al.*, 2006], BOXPUSHING [Seuken and Zilberstein, 2007], GRID3X3 [Amato *et al.*, 2009], MARS [Amato and Zilberstein, 2009], HOTEL [Spaan and Melo, 2008], RECYCLING [Amato *et al.*, 2007], and BROADCAST [Hansen *et al.*, 2004].

## Results

Table 1 summarizes the results for the benchmarks with different horizons $h$. We show the optimal value and timing results for GMAA$^*$-ICE (for three heuristics) and for RS-MAA$^*$ (for two heuristics). In summary, we observe that RS-MAA$^*$ scales to larger horizons on DECTIGER,FIREFIGHTING4, GRID, GRID3X3, MARS, and RECYCLING. On HOTEL, scalability is comparable, but RS-MAA$^*$ is faster. On BROADCAST and FIREFIGHTING3, GMAA$^*$-ICE scales better. In line with the computational complexity, horizons $\leq 10$ are already challenging on many benchmarks. However, HOTEL, RECYCLING and BROADCAST have a constant number of clustered LOHs which allows to scale to much larger horizons [Oliehoek *et al.*, 2013].

RS-MAA$^*$ scales much further on on DECTIGER and MARS. On DECTIGER, the tightness of the heuristic is essential. The $Q_{\mathsf{BG}}$ heuristic reveals too much information which leads to unrealistically high heuristic values. On $Q_3$, however, only information from the first two stages is shared, which under an optimal policy becomes irrelevant in later stages. This leads to a very tight heuristic in the later stages.

On MARS, $Q_{\mathsf{BG}}$ struggles as its run time complexity includes a factor $O\left(|\mathcal{A}_*|^{n|\mathcal{O}_*|}\right)$, which is $6^{16}$ for MARS due to

[5]The exact configurations used by [Oliehoek *et al.*, 2013] do not seem to be available in the MADP toolbox, we use `QBGhybrid` and `QBGTreeIncPrune`.

| | | GMAA$^*$-ICE | | | RS-MAA$^*$ | |
|---|---|---|---|---|---|---|
| $h$ | value | QMDP | prune | hybrid | $Q_3$ | $Q_\infty$ |
| DECTIGER | | | | | | |
| 5 | 7.026451 | 40 | **<1** | 1 | **<1** | **<1** |
| 6 | 10.381625 | MO | 16 | 18 | **<1** | 17 |
| 7 | 9.993568 | MO | TO | MO | **2** | MO |
| 8 | 12.217263 | MO | TO | MO | **2** | MO |
| 9 | 15.572437 | MO | TO | MO | **9** | MO |
| 10 | 15.184380 | MO | TO | MO | **628** | MO |
| 11 | 17.408076 | MO | TO | MO | **249** | MO |
| 12 | 20.763250 | MO | TO | MO | **762** | MO |
| FIREFIGHTING ($\langle n_h = 3, n_f = 3\rangle$) | | | | | | |
| 3 | −5.736969 | 2 | TO | 4 | **1** | **1** |
| 4 | −6.578834 | 29 | TO | 181 | **3** | **3** |
| 5 | −7.069874 | 1730 | TO | 221 | **24** | 41 |
| 6 | −7.175591 | **57** | TO | TO | 281 | 281 |
| 7 | −7.175591 | **33** | TO | TO | MO | MO |
| FIREFIGHTING ($\langle n_h = 4, n_f = 3\rangle$) | | | | | | |
| 3 | −11.135643 | 567 | TO | 989 | **8** | **8** |
| 4 | −14.106819 | TO | TO | TO | **137** | 138 |
| GRID | | | | | | |
| 3 | 1.550444 | **<1** | TO | 7 | **<1** | **<1** |
| 4 | 2.241577 | 5 | TO | 288 | **2** | **2** |
| 5 | 2.970496 | MO | TO | TO | **9** | **9** |
| 6 | 3.717168 | MO | TO | TO | 97 | **96** |
| BOXPUSHING | | | | | | |
| 3 | 66.081000 | **<1** | TO | 2535 | **<1** | **<1** |
| 4 | 98.593613 | 131 | TO | TO | 10 | **9** |
| 5 | 107.729851 | TO | TO | TO | MO | MO |
| GRID3X3 | | | | | | |
| 4 | 0.432900 | **1** | MO | TO | **1** | **1** |
| 5 | 0.895656 | 3 | MO | TO | **2** | **2** |
| 6 | 1.492987 | MO | MO | TO | **12** | **12** |
| MARS | | | | | | |
| 4 | 10.180800 | 8 | MO | TO | **1** | **1** |
| 5 | 13.266538 | MO | MO | TO | **2** | **2** |
| 6 | 18.623165 | MO | MO | TO | **5** | **5** |
| 7 | 20.900724 | MO | MO | TO | 21 | **10** |
| 8 | 22.478798 | MO | MO | TO | **40** | MO |
| 9 | 24.320398 | MO | MO | TO | **160** | MO |
| HOTEL | | | | | | |
| 5 | 27.187500 | MO | 3 | 19 | **<1** | **<1** |
| 100 | 502.187500 | MO | 69 | MO | 201 | **21** |
| 250 | 1252.187500 | MO | 126 | MO | 595 | **57** |
| 500 | 2502.187500 | MO | 230 | MO | 1317 | **119** |
| 1500 | 7502.187500 | MO | 629 | MO | TO | **397** |
| 2000 | 10002.187500 | MO | 832 | MO | TO | **557** |
| RECYCLING | | | | | | |
| 8 | 25.709878 | **<1** | **<1** | MO | 1 | **<1** |
| 20 | 62.633136 | 52 | **<1** | MO | 4 | 1 |
| 70 | 216.479290 | MO | 14 | MO | 209 | **2** |
| 80 | 247.248521 | MO | 68 | MO | 292 | **3** |
| 100 | 308.786982 | MO | MO | MO | 467 | **4** |
| 500 | 1539.556213 | MO | MO | MO | TO | **21** |
| 1500 | 4616.479290 | MO | MO | MO | TO | **113** |
| BROADCAST | | | | | | |
| 10 | 9.290000 | **<1** | **<1** | MO | **<1** | **<1** |
| 100 | 90.760423 | MO | **32** | MO | 37 | 36 |
| 250 | 226.500545 | MO | **149** | MO | 718 | 720 |
| 500 | 452.738119 | MO | **385** | MO | 3496 | 3016 |
| 800 | 724.214207 | MO | MO | MO | TO | TO |

Table 1: Results and running times (seconds). TO and MO denote timeout (>3600s) and memout (>16GB).

| $M$ | 200 | $\infty$ | 200 | $\infty$ | 200 | 200 | no clustering | no last stage |
|---|---|---|---|---|---|---|---|---|
| $Q_d$ | $Q_3$ | $Q_3$ | $Q_3$ | $Q_3$ | $Q_1$ | $Q_2$ | | |
| $\alpha$ | 0.2 | 0.2 | $\infty$ | $\infty$ | 0.2 | 0.2 | | |
| $h=6$ | **<1** | **<1** | **<1** | **<1** | 3 | 1 | 1 | 1 |
| $h=7$ | **2** | **2** | **2** | **2** | 293 | 5 | MO | **2** |
| $h=8$ | **2** | 31 | 3 | 40 | 3355 | 15 | MO | **2** |
| $h=10$ | **628** | MO | 630 | TO | TO | TO | MO | 2453 |
| $h=12$ | **762** | MO | 1129 | TO | TO | TO | MO | 1296 |

Table 2: Ablation study for DECTIGER, showing computation times in seconds. The last two columns are computed using the baseline parameters $M = 200$, heuristic $Q_3$, $\alpha = 0.2$.

the eight different observations. Hence, computing $Q_{\mathsf{BG}}$ is infeasible (for every horizon). $Q_{\mathsf{MDP}}$ can be computed, but is not sufficiently tight. Likewise, the tighter heuristic $Q_3$ outperforms $Q_\infty$ for sufficiently large horizons.

GRID is a prime example where the on-the-fly computation of the heuristic values is an essential benefit, as the $Q_{\mathsf{BG}}$ heuristic cannot finish the computation of the heuristic values.

On HOTEL and BROADCAST, the $Q_{\mathsf{BG}}$ heuristic is already tight, which means that the more expensive computation of $Q_d$ does not significantly reduce the number of nodes that are explored. This also explains why $Q_\infty$ outperforms $Q_3$.

FIREFIGHTING3's structure makes $h = 5$ expensive (no potential for clustering). The recursive nature of RS-MAA$^*$ suffers from this when computing values for $h \geq 6$.

In short, RS-MAA$^*$ performs generally better on domains where agent knowledge about the state is relatively valuable. Note that the recursive heuristics are tighter than other known competitive heuristics because they share less information about observations (and hence about the state). Hence, if knowledge about the state is valuable, then the additional tightness of the recurive heuristic is more essential for limiting the number of nodes expanded in the search tree.

RS-MAA$^*$ is also better on domains with many clustered observations, as the speed-up provided by using the small-step search tree is more significant in this case. By contrast, GMAA$^*$-ICE seems better on domains where there are very few clustered observations, although in these cases RS-MAA$^*$ can sometimes still outperform GMAA$^*$-ICE.

**Ablation study.** For DECTIGER, we provide additional results using alternative configurations in Table 2. The first configuration reflects the baseline. The next 5 configurations change the three hyperparameters. In particular, setting $M = \alpha = \infty$ disables abandoning computations as described in Sec. 5. We additionally used the baseline parameters but without clustering or the last-stage optimization from Sec. 6. Clustering, early termination, and avoiding low-depth heuristics are essential components for the performance. Terminating early based on a threshold and the last-stage optimization are less essential, but still provide a significant speedup.

**Benchmark-specific parameters.** Although the parameters $M = 200$, $d = 3$, and $\alpha = 0.2$ work well for a wide range of benchmarks, some benchmarks benefit from different parameters. For example, BOXPUSHING is solved faster when using the tighter $Q_2$ heuristic, namely in respectively 3 and 6 seconds for $h = 4$ and $h = 5$. HOTEL can be solved in 44 seconds for $h = 2000$ using a less tight $M = 25$ and $Q_\infty$. Finally, BROADCAST can be solved in respectively 250, 465 and 1342 seconds for $h = 500$, $h = 800$, and $h = 2000$ using a less tight $M = 25$ and $Q_\infty$. Notably, it seems that for most benchmarks, parameter choices that are good for a short horizon are also a good choice on larger horizons.

**Comparison to $\epsilon$-optimal Algorithms**

It is interesting to understand the performance of RS-MAA$^*$ compared to less strict $\epsilon$-optimal algorithms, such as FB-HSVI [Dibangoye *et al.*, 2016]. These algorithms provide an upper bound and a lower bound for the optimal value that are at most $\epsilon$ apart. As there is no implementation of FB-HSVI available at the time of writing, we provide a short qualitative comparison based on the data from Dibangoye *et al.* [2016]. For GRID, GRID3X3, MARS and BOXPUSHING, FB-HSVI is able to provide $\epsilon$-optimal solutions for horizon 10, thereby outperforming RS-MAA$^*$. For BROADCAST, it is is likely that RS-MAA$^*$ outperforms FB-HSVI, as computing the solution for horizon 100 already took 473 seconds for FB-HSVI. For RECYCLING, both algorithms perform very well and to see which algorithm performs better, results for FB-HSVI beyond horizon 100 are needed. All run times above allow an $\epsilon = 0.01$ optimality gap.

Based on this qualitative comparison, some aspects remain unclear: It is unclear from which $\epsilon$ onwards RS-MAA$^*$ outperforms FB-HSVI, and it is also unclear in which circumstances the upper bounds obtained with RS-MAA$^*$ within a given timelimit are stricter than the ones found by FB-HSVI.

## 8 Conclusion

We presented a novel method for finding optimal solutions for finite-horizon Dec-POMDP problems. Our approach showed competitive, and often superior, performance compared to the state-of-the-art. In particular, for several benchmarks, we extend the highest horizon for which an optimal solution is known. The contributions that led to this advancement are a small-step variant of the search tree for MAA$^*$, as well as a novel heuristic that is computed on-the-fly. In the future, we will investigate supporting nonlinear objectives, relevant, e.g., for rewards that relate the joint belief of agents, as in Lauri *et al.* [2019] and Lauri *et al.* [2020]. Finally, towards further scalability, we will integrate results by approximate Dec-POMDP solvers in the heuristic computation.

## Acknowledgements

We would like to thank the anonymous reviewers for their useful comments. This work has been partially funded by the ERC Starting Grant 101077178 (DEUCE) and the NWO grant NWA.1160.18.238 (PrimaVera).

## Contribution Statement

Wietze Koops is the primary designer of the algorithm and implemented it and evaluated the algorithms. The other authors contributed discussions, ideas, and shaped the description of the algorithm.

## References


[Amato and Zilberstein, 2009] Christopher Amato and Shlomo Zilberstein. Achieving goals in decentralized POMDPs. In *AAMAS*, pages 593–600, 2009.

[Amato *et al.*, 2006] Christopher Amato, Daniel S. Bernstein, and Shlomo Zilberstein. Optimal fixed-size controllers for decentralized POMDPs. In *AAMAS MSDM workshop*, 2006.

[Amato *et al.*, 2007] Christopher Amato, Daniel S. Bernstein, and Shlomo Zilberstein. Optimizing memory-bounded controllers for decentralized POMDPs. In *UAI*, pages 1–8, 2007.

[Amato *et al.*, 2009] Christopher Amato, Jilles Steeve Dibangoye, and Shlomo Zilberstein. Incremental policy generation for finite-horizon DEC-POMDPs. In *ICAPS*, 2009.

[Amato *et al.*, 2019] Christopher Amato, George Dimitri Konidaris, Leslie Pack Kaelbling, and Jonathan P. How. Modeling and planning with macro-actions in decentralized POMDPs. *JAIR*, 64:817–859, 2019.

[Barto and Mahadevan, 2003] Andrew G. Barto and Sridhar Mahadevan. Recent advances in hierarchical reinforcement learning. *Discret. Event Dyn. Syst.*, 13(4):341–379, 2003.

[Bernstein *et al.*, 2002] Daniel S. Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. The complexity of decentralized control of Markov decision processes. *Mathematics of operations research*, 27(4):819–840, 2002.

[Cazenave, 2010] Tristan Cazenave. Partial move A*. In *22nd IEEE International Conference on Tools with Artificial Intelligence*, volume 2, pages 25–31. IEEE, 2010.

[Dibangoye *et al.*, 2016] Jilles Steeve Dibangoye, Christopher Amato, Olivier Buffet, and François Charpillet. Optimally solving Dec-POMDPs as continuous-state MDPs. *JAIR*, 55:443–497, 2016.

[Hansen *et al.*, 2004] Eric A. Hansen, Daniel S. Bernstein, and Shlomo Zilberstein. Dynamic programming for partially observable stochastic games. In *AAAI*, pages 709–715, 2004.

[Kaelbling *et al.*, 1998] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and Acting in Partially Observable Stochastic Domains. *Artif. Intell.*, 101(1-2):99–134, 1998.

[Lauri *et al.*, 2019] Mikko Lauri, Joni Pajarinen, and Jan Peters. Information gathering in decentralized POMDPs by policy graph improvement. In *AAMAS*, pages 1143–1151. IFAAMAS, 2019.

[Lauri *et al.*, 2020] Mikko Lauri, Joni Pajarinen, and Jan Peters. Multi-agent active information gathering in discrete and continuous-state decentralized POMDPs by policy graph improvement. *Auton. Agents Multi Agent Syst.*, 34(2):42, 2020.

[Nair *et al.*, 2003] Ranjit Nair, Milind Tambe, Makoto Yokoo, David Pynadath, and Stacy Marsella. Taming decentralized POMDPs: Towards efficient policy computation for multiagent settings. In *IJCAI*, pages 705–711, 2003.

[Oliehoek and Amato, 2016] Frans A. Oliehoek and Christopher Amato. *A Concise Introduction to Decentralized POMDPs*. Briefs in Intelligent Systems. Springer, 2016.

[Oliehoek and Vlassis, 2007] Frans A. Oliehoek and Nikos Vlassis. Q-value heuristics for approximate solutions of Dec-POMDPs. In *AAAI Spring Symposium: Game Theoretic and Decision Theoretic Agents*, pages 31–37, 2007.

[Oliehoek *et al.*, 2008] Frans A. Oliehoek, Matthijs T. J. Spaan, and Nikos Vlassis. Optimal and approximate Q-value functions for decentralized POMDPs. *JAIR*, 32:289–353, 2008.

[Oliehoek *et al.*, 2009] Frans A. Oliehoek, Shimon Whiteson, and Matthijs T. J. Spaan. Lossless clustering of histories in decentralized POMDPs. In *AAMAS*, pages 577–584, 2009.

[Oliehoek *et al.*, 2013] Frans A. Oliehoek, Matthijs T. J. Spaan, Christopher Amato, and Shimon Whiteson. Incremental clustering and expansion for faster optimal planning in Dec-POMDPs. *JAIR*, 46:449–509, 2013.

[Oliehoek *et al.*, 2017] Frans A. Oliehoek, Matthijs T. J. Spaan, Bas Terwijn, Philipp Robbel, and João V. Messias. The MADP toolbox: An open source library for planning and learning in (multi-)agent systems. *JMLR*, 18:1–5, 2017.

[Pateria *et al.*, 2022] Shubham Pateria, Budhitama Subagdja, Ah-Hwee Tan, and Chai Quek. Hierarchical reinforcement learning: A comprehensive survey. *ACM Comput. Surv.*, 54(5):109:1–109:35, 2022.

[Russell and Norvig, 2020] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach (4th Edition)*. Pearson, 2020.

[Seuken and Zilberstein, 2007] Sven Seuken and Shlomo Zilberstein. Memory-bounded dynamic programming for DEC-POMDPs. In *IJCAI*, pages 2009–2015, 2007.

[Spaan and Melo, 2008] Matthijs T. J. Spaan and Francisco S. Melo. Interaction-driven Markov games for decentralized multiagent planning under uncertainty. In *AAMAS*, pages 525–532, 2008.

[Spaan *et al.*, 2011] Matthijs T. J. Spaan, Frans A. Oliehoek, and Christopher Amato. Scaling up optimal heuristic search in Dec-POMDPs via incremental expansion. In *IJCAI*, 2011.

[Szer *et al.*, 2005] Daniel Szer, François Charpillet, and Shlomo Zilberstein. MAA*: A heuristic search algorithm for solving decentralized POMDPs. In *UAI*, 2005.

[Xie *et al.*, 2020] Yuxuan Xie, Jilles Dibangoye, and Olivier Buffet. Optimally solving two-agent decentralized POMDPs under one-sided information sharing. In *ICML*, volume 119 of *Proceedings of Machine Learning Research*, pages 10473–10482, 2020.