Scaling Up Optimal Heuristic Search in Dec-POMDPs via Incremental Expansion

Matthijs T.J. Spaan
Inst. for Systems and Robotics
Instituto Superior Técnico
Lisbon, Portugal
mtjspaan@isr.ist.utl.pt

Frans A. Oliehoek
CSAIL
MIT
Cambridge, MA 02139, USA
fao@csail.mit.edu

Christopher Amato
Aptima, Inc.
Woburn, MA 01801, USA
camato@aptima.com

ABSTRACT

Planning under uncertainty for multiagent systems can be formalized as a decentralized partially observable Markov decision process. We advance the state of the art for optimal solution of this model, building on the Multiagent A* heuristic search method. A key insight is that we can avoid the full expansion of a search node that generates a number of children doubly exponential in the node's depth. Instead we incrementally expand the children of a node only when a next child might have the highest heuristic value. We target a subsequent bottleneck by introducing a more memory-efficient representation for our heuristic functions. Proof is given that the resulting algorithm is correct and experiments demonstrate a significant speedup over the state of the art, allowing for optimal solutions over longer horizons for many benchmark problems.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent systems*

General Terms

Algorithms, Theory, Experimentation

Keywords

Planning under uncertainty, Cooperative multiagent systems, Decentralized POMDPs, Heuristic search

1. INTRODUCTION

Planning under uncertainty for multiagent systems is an important problem in artificial intelligence, as agents may often possess uncertain information while sharing their environment with other agents. Due to stochastic actions and noisy sensors, agents must reason about many possible outcomes and the uncertainty surrounding them. Because multiple agents are present, each agent must also reason about the choices of the others and how they may affect the environment. In cooperative systems, finding optimal joint plans is especially challenging when each agent must choose actions based solely on local knowledge due to nonexistent or noisy communication. Possible application domains include multi-robot teams, communication networks, load balancing, and many other problem settings in which agents need to coordinate under uncertain conditions.

The decentralized partially observable Markov decision process (Dec-POMDP) is a formal model for such planning problems. Unfortunately, optimal solution methods [3] and even bounded approximations (ϵ-optimal solutions) [12] suffer from doubly-exponential complexity (NEXP-Complete); the search space for horizon $h+1$ is exponentially larger than the one for horizon h. This means that algorithms with guarantees on solution quality have difficulties with at least some problems. Algorithms that are always efficient in runtime can have no guarantees on the solution quality.

In this paper we consider the optimal solution of Dec-POMDPs over a finite horizon. Even though the high worst-case complexity results preclude such methods from being applicable for some larger problems, there are several reasons to be interested in optimal solutions: 1) As approximate algorithms come with no guarantees, optimal methods are necessary as a tool to analyze the performance of approximate algorithms. 2) Most successful approximate algorithms (e.g., [4, 14, 17]) are based on optimal solution methods, so algorithmic improvements to the latter are likely to directly transfer to the former. 3) Optimal techniques can give insight in the nature of problems and their solutions. For instance, previous work on optimal methods generated the insight that certain properties of the BroadcastChannel problem make it easier to solve [11]. 4) They are of interest for solving small problems that arise naturally or as part of a decomposition. Moreover, many problem instances are much easier to solve than the worst-case complexity suggests [1], allowing optimal solutions to be practical.

We provide significant advances to the state of the art in optimal Dec-POMDP solution methods by extending Multiagent A* (MAA*) [16] —which performs an A* search through the tree of possible partial joint policies— and derived methods with a new technique for incremental expansion of search tree nodes. Expanding a node in this search tree entails generating all possible children, which is a major source of intractability since the number of such children is doubly exponential in the depth of the node. In practice, however, only a small number of the generated nodes may actually be queried during the search. Our key observation is that if a method is able to *incrementally* generate children in order of their heuristic value, it does not need to expand all of them at once. We exploit this insight, building upon recent advances in the solution of collaborative Bayesian games [9].

As with any A* method, our approach's performance de-

The Sixth Annual Workshop on Multiagent Sequential Decision-Making in Uncertain Domains (*MSDM-2011*), held in conjunction with *AAMAS-2011* on May 3, 2011 in Taipei, Taiwan.

pends on the tightness of the heuristic. In many problems the upper bound provided by the value function of the underlying MDP (Q_{MDP}) is not tight enough for heuristic search to be effective [10]. Other heuristics are tighter, such as those based on the underlying POMDP solution (Q_{POMDP}) or the value function resulting from assuming 1-step-delayed communication (Q_{BG}). However, they require storing values for all joint action-observation histories or representing them as a potentially exponential number of vectors. A crucial insight is that the number of values stored in a tree-based representation grows exponentially when moving forward in time, while the size of a vector-based representation grows in the opposite direction. We exploit this insight by introducing a hybrid representation that is more compact.

In this work, we integrate the incremental expansion idea in GMAA* with incremental clustering (GMAA*-IC), an MAA* extension that uses lossless history clustering for improved scalability [11]. The resulting algorithm is called GMAA*-ICE as it provides incremental clustering and expansion. We prove that GMAA*-ICE is correct and expands search nodes in the same order as the original method.

We show the efficacy of our methods on a suite of benchmark problems, demonstrating a significant speedup over the state of the art. In many cases GMAA*-ICE provides the optimal solution over longer horizons than those previously solved. In particular, incremental expansion provides leverage in those problem domains in which history clustering is less effective.

The rest of the paper is organized as follows. We begin in Sec. 2 with background on Dec-POMDPs, their representation as CBGs as well as on GMAA*-IC. Sec. 3 introduces GMAA*-ICE, and in Sec. 4 we prove its correctness. The hybrid representation is introduced in Sec. 5 and Sec. 6 presents experimental results. Lastly, Sec. 7 presents conclusions and future work.

2. BACKGROUND

Here we provide some background information on Dec-POMDPs and their optimal solution over a finite horizon.

2.1 Decentralized POMDPs

A *decentralized partially observable Markov decision process (Dec-POMDP)* consists of:

- A set of n agents.
- $\mathcal{S}$ is a finite set of states.
- $\mathcal{A} = \times_i \mathcal{A}_i$ is the set of joint actions, where $\mathcal{A}_i$ is the set of actions available to agent i. Every time step, one joint action $\mathbf{a} = \langle a_1,...,a_n \rangle$ is taken.
- T is the transition function, a mapping from states and joint actions to probability distributions over next states: $T : \mathcal{S} \times \mathcal{A} \to \mathcal{P}(\mathcal{S})$.
- R is the reward function that maps states and joint actions to real numbers: $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$.
- $\mathcal{O} = \times_i \mathcal{O}_i$ is the set of joint observations, with $\mathcal{O}_i$ the set of observations available to agent i. Every time step, one joint observation $\mathbf{o} = \langle o_1,...,o_n \rangle$ is received.
- O is the observation function, a mapping from joint actions and successor states to probability distributions over joint observations: $O : \mathcal{A} \times \mathcal{S} \to \mathcal{P}(\mathcal{O})$.
- h is the horizon, the number of time steps.
- $\boldsymbol{b}^0$ the initial state distribution.

The goal of a Dec-POMDP is to find a decentralized deterministic joint policy $\boldsymbol{\pi} = \langle \pi_1, \ldots, \pi_n \rangle$. Here each deterministic individual policy π_i maps from local observation-histories (OH) $\vec{o}_i^t = \left(o_i^1,...,o_i^t\right)$ to actions: $\pi_i(\vec{o}_i^t) = a_i^t$. An individual policy π_i can be interpreted as a sequence of decision rules $\pi_i = (\delta_i^0, \delta_i^1, \ldots, \delta_i^{h-1})$, where δ_i^t maps from length-t OHs to actions. In the remainder of this paper, we will also consider action-observation histories (AOHs) $\vec{\theta}_i^t = \left(a_i^0,o_i^1,a_i^1,...,a_i^{t-1},o_i^t\right)$. An AOH for an agent i at stage t is denoted by $\vec{\theta}_i^t$. The optimal joint policy $\boldsymbol{\pi}^*$ maximizes the expected cumulative reward. Computing an optimal joint policy is provably intractable (NEXP-complete) [3].

The fact that an individual policy π_i depends only on the local information $\vec{o}_i$ available to an agent means that the on-line execution phase is truly decentralized; no communication is assumed to take place (other than specified via actions and observations). The planning itself however, may take place in an off-line phase and be centralized. This is the assumption we make in this work. For a more detailed introduction to Dec-POMDPs see, e.g., [10, 15].

2.2 Multiagent A*

We build upon GMAA*-Cluster [11], which in turn is based on Multiagent A* (MAA*) [16]. These methods perform a search over partial, or *past*, joint policies $\boldsymbol{\varphi}^t$ that specify the joint policy up to stage t: $\boldsymbol{\varphi}^t = (\boldsymbol{\delta}^0, \boldsymbol{\delta}^1, \ldots, \boldsymbol{\delta}^{t-1})$, where $\boldsymbol{\delta}^i$ is the joint decision rule for the i-th stage. For such a $\boldsymbol{\varphi}^t$, we can compute a heuristic value $\widehat{V}(\boldsymbol{\varphi}^t)$ by computing $V^{0\ldots(t-1)}(\boldsymbol{\varphi}^t)$, the actual expected reward over the first t stages, and adding a heuristic value $H(\boldsymbol{\varphi}^t)$ for the remaining stages. When the heuristic is *admissible*, i.e., a guaranteed overestimation, it is possible to perform standard A* search: select the node $q = \langle \boldsymbol{\varphi}^t, \hat{v} \rangle$ with the highest $\hat{v}$, which is initialized as $\hat{v} \leftarrow \widehat{V}(\boldsymbol{\varphi}^t)$, and expand it by generating all child nodes $\boldsymbol{\varphi}^{t+1} = \langle \boldsymbol{\varphi}^t \circ \boldsymbol{\delta}^t \rangle$ that can be formed by appending a joint decision rule $\boldsymbol{\delta}^t$ to $\boldsymbol{\varphi}^t$. We assume that there is a total ordering over nodes, such that ties in value $\hat{v}$ are broken in a consistent way.

A major source of complexity in MAA* is the full expansion of a search node; the number of $\boldsymbol{\delta}^t$ (that can be used to form the children of a node $\boldsymbol{\varphi}^t$ at depth t in the search tree) is doubly exponential in t, because the number of OHs grows exponentially with t. In an attempt to counter this problem, for the last stage $t = h-1$, MAA* generates the child nodes one by one until a node is found with value equal to its parent's heuristic value. If this happens, no other siblings will have to be generated.

Unfortunately, this method does not provide much leverage in practice, since it is unlikely that a child node will have the same heuristic value as its parent and, even if one does, there is no effective way to find such a child [15]. Also, this does not address the complexity of intermediate stages. Therefore Seuken and Zilberstein [15] argue that MAA* "can at best solve problems whose horizon is only 1 greater than those that can already be solved by naïve brute force search."

In this paper, we address these problems. That is, we provide efficient incremental expansion through a method that is able to select the highest ranked child at all stages, not just at the last stage. Moreover, we combine it with another method that has brought scaling to MAA*: clustering of histories.

Algorithm 1 GMAA*-IC [11]

```
1: v^GMAA ← −∞
2: φ^0 ←(), v̂ ← +∞, q^0 ←⟨φ^0, v̂⟩
3: P ←{q^0}
4: repeat
5:    q ← Select(P)                                   {q = ⟨φ^t, v̂⟩}
6:    P.pop(q)
7:    B(φ^{t−1}) ← φ^{t−1}.CBG        {Note φ^t = ⟨φ^{t−1} ∘ β^{t−1}⟩ }
8:    B(φ^t) ← ConstructExtendedCBG(B(φ^{t−1}),β^{t−1})
9:    B(φ^t) ← ClusterCBG(B(φ^t))
10:   Φ_Expand ← Expand(B^t)
11:   V̂(φ^{t+1}) ← V^{0…t−1}(φ^t) + V̂(β^t)
12:   if last stage t = h − 1 then
13:      if V(π) > v^GMAA then
14:         v^GMAA ← V(π)                    {found new lower bound}
15:         π* ← π
16:         P.prune(v^GMAA)
17:   else
18:      Q ←{⟨φ,V̂(φ)⟩ | φ ∈ Φ_Expand, V̂(φ) > v^GMAA}
19:      P.insert(Q)
20: until P is empty
```

2.3 Lossless incremental clustering

GMAA*-Cluster extends MAA* by 1) interpreting nodes in the search tree as *collaborative Bayesian games (CBGs)* and 2) clustering histories and thereby indirectly policies. In particular, we consider the version that performs incremental clustering (GMAA*-IC). For a complete introduction to GMAA* refer to [10], we concisely outline the main ideas.

As discussed, each node in the MAA* search tree corresponds to a $\boldsymbol{\varphi}^t$. This can be interpreted as corresponding to a CBG [10]: given state distribution $\boldsymbol{b}^0$, for each $\boldsymbol{\varphi}^t$ it is possible to construct a CBG $B(\boldsymbol{\varphi}^t)$, which consists of:

- the set of agents $\{1 \dots n\}$.
- $\mathcal{A}$ is the set of the joint actions.
- $\boldsymbol{\Theta}$, the set of their *joint types*. A joint type $\boldsymbol{\theta}$ specifies a *type* for each agent $\boldsymbol{\theta} = \langle \theta_1, \dots, \theta_n \rangle$.
- $\Pr(\cdot)$, a probability distribution over joint types.
- $\widehat{Q}$, a heuristic payoff function $\widehat{Q}(\boldsymbol{\theta}, \mathbf{a}) \to \mathbb{R}$.

A type θ_i of an agent i represents the private information it holds, so it corresponds to the history of actions and observations $\vec{\theta}_i^t$. This means that $\widehat{Q}$ should provide a heuristic estimate for each $(\vec{\boldsymbol{\theta}}^t, \mathbf{a})$-pair.

In a CBG, each agent uses a BG-policy β_i that maps individual types to actions: $\beta_i(\theta_i) = a_i$. A joint policy for the CBG $\boldsymbol{\beta}$ corresponds to a joint decision rule: $\boldsymbol{\beta} \equiv \boldsymbol{\delta}^t$ with heuristic value given by

$$\widehat{V}(\boldsymbol{\beta}) = \sum_{\vec{\boldsymbol{\theta}}^t} \Pr(\vec{\boldsymbol{\theta}}^t | \boldsymbol{\varphi}^t, \boldsymbol{b}^0) \widehat{Q}(\vec{\boldsymbol{\theta}}^t, \boldsymbol{\beta}(\vec{\boldsymbol{\theta}}^t)), \tag{1}$$

where $\boldsymbol{\beta}(\vec{\boldsymbol{\theta}}^t) = \langle \beta_i(\vec{\theta}_i^t) \rangle_{i=1\dots n}$ denotes the joint action that results from application of the individual BG-policies to the individual AOH $\vec{\theta}_i^t$ specified by $\vec{\boldsymbol{\theta}}^t$.

From this CBG perspective, when expanding a node all $\boldsymbol{\beta}$ are returned and appended to $\boldsymbol{\varphi}^t$ to form the set of all children:

$$\Phi_{\text{Expand}} = \left\{ \langle \boldsymbol{\varphi}^t \circ \boldsymbol{\beta} \rangle | \boldsymbol{\beta} \text{ is a joint BG policy of } B(\boldsymbol{\varphi}^t) \right\}.$$

The valuation of such a child $\boldsymbol{\varphi}^{t+1} = \langle \boldsymbol{\varphi}^t \circ \boldsymbol{\beta} \rangle$ is given by

$$\widehat{V}(\boldsymbol{\varphi}^{t+1}) = V^{0\dots(t-1)}(\boldsymbol{\varphi}^t) + \widehat{V}(\boldsymbol{\beta}), \tag{2}$$

where now the expected immediate reward for stage t is represented within the heuristic $\widehat{V}(\boldsymbol{\beta})$. It can be shown when the heuristic $\widehat{Q}$ faithfully represents the expected immediate reward, this reformulation is exactly equal to MAA* [10].

This reformulation of MAA* to work on CBGs does not directly gain any computational advantage. GMAA*-IC leverages the CBG representation of MAA* by clustering individual types in a CBG in such a way that the solution of the clustered CBG corresponds to a solution of the original CBG. Clustering also gives an effective way to eliminate histories with zero probability. Clustering does not always reduce the CBG's size, but when it does, it will result in great computational savings, since the number of $\boldsymbol{\beta}$ is exponential in the number of types. In particular, it is possible to perform incremental clustering by bootstrapping from the clustered CBG for the previous stage [11].

Algorithm 1 shows pseudo-code for GMAA*-IC. At every iteration, `Select` returns the best-ranked $q = \langle \boldsymbol{\varphi}^t, \hat{v} \rangle$ from the open list P. Subsequently, a CBG is constructed, clustered and used to generate all child nodes $\boldsymbol{\varphi}^{t+1}$. This process continues until a full policy is found with value higher than the upper bounds of any remaining partial policies.

3. INCREMENTAL EXPANSION

Recently, new methods for solving CBGs have been developed [7, 9] that can provide speedups of multiple orders of magnitude over brute force search (enumeration). Unfortunately, MAA* has not been able to profit from these methods: in order to guarantee optimality, it relies on expansion of *all* (child nodes corresponding to all) joint BG-policies $\boldsymbol{\beta}$ for the intermediate stages.[1] However, many of the expanded child nodes may never be selected for further expansion. The key observation is the following:

Observation 1. If we have a way to generate the children in increasing heuristic order and that heuristic is admissible, we do not have to expand all the children.

We discuss this in more detail below, starting with a formalization of the relative heuristic values of two child nodes.

LEMMA 1. *Given two joint BG policies $\boldsymbol{\beta}, \boldsymbol{\beta}'$ for a CBG $B^t(\boldsymbol{\varphi}^t)$, if $\widehat{V}(\boldsymbol{\beta}) \geq \widehat{V}(\boldsymbol{\beta}')$, then for the corresponding child nodes $\widehat{V}(\boldsymbol{\varphi}^{t+1}) \geq \widehat{V}(\boldsymbol{\varphi}^{t+1\prime})$.*

PROOF. *This holds directly by the definition of $\widehat{V}(\boldsymbol{\varphi}^t)$*

$$\begin{aligned} \widehat{V}(\boldsymbol{\varphi}^{t+1}) &= V^{0\dots(t-1)}(\boldsymbol{\varphi}^t) + \widehat{V}(\boldsymbol{\beta}) \\ &\geq V^{0\dots(t-1)}(\boldsymbol{\varphi}^t) + \widehat{V}(\boldsymbol{\beta}') = \widehat{V}(\boldsymbol{\varphi}^{t+1\prime}), \end{aligned}$$

as given by (2). □

It follows directly that, if for $B^t(\boldsymbol{\varphi}^t)$ we use a CBG solver that can generate a sequence of policies $\boldsymbol{\beta}, \boldsymbol{\beta}', \dots$ such that

$$\widehat{V}(\boldsymbol{\beta}) \geq \widehat{V}(\boldsymbol{\beta}') \geq \dots$$

then, for the sequence of corresponding children

$$\widehat{V}(\boldsymbol{\varphi}^{t+1}) \geq \widehat{V}(\boldsymbol{\varphi}^{t+1\prime}) \geq \dots.$$

[1] For the last stage, clearly it is possible to only generate the best child node of $\boldsymbol{\varphi}^{h-1}$ by appending the optimal solution of the CBG.

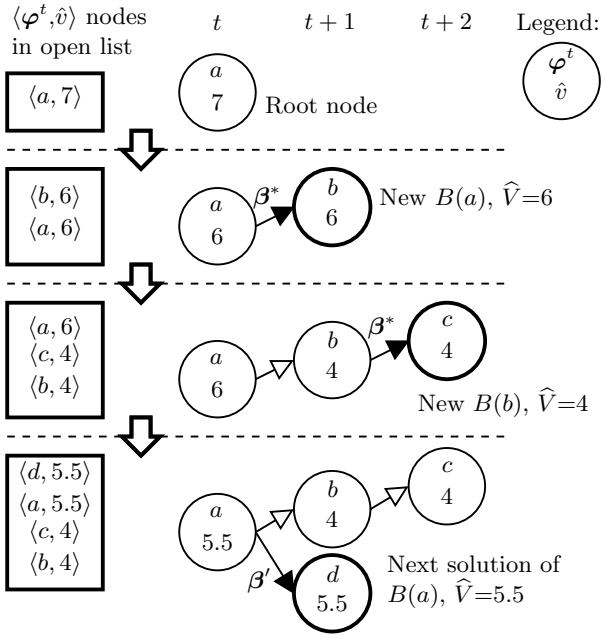


Figure 1: Illustration of incremental expansion. On the left we show the evolution of the open list. Past joint policies $\boldsymbol{\varphi}^t$ are indexed by letters.

Exploiting this knowledge, we can expand only the first child $\boldsymbol{\varphi}^{t+1}$, compute its $\widehat{V}(\boldsymbol{\varphi}^{t+1})$ and set the value of the parent node to $q.\hat{v} \leftarrow \widehat{V}(\boldsymbol{\varphi}^{t+1})$, since we know that all the unexpanded siblings will have $\widehat{V}$ lower or equal to that. As such, we can reinsert q into P to act as a *placeholder* for all its non-expanded children. To ensure that children are expanded before their parents, we break ties in a consistent manner, ranking nodes for earlier stages t higher in case of equal value.[2] Fig. 1 illustrates incremental expansion.

We integrate incremental expansion in GMAA*-IC resulting in GMAA* *with incremental clustering and expansion* (GMAA*-ICE). It performs an A* search over nodes $q = \langle \boldsymbol{\varphi}^t, \hat{v}, \text{PH} \rangle$, where PH is a boolean indicating whether the node is a placeholder. At every iteration, the heuristically highest ranked q is selected from an open list P and expanded. When a new best full joint policy is found, the lower bound $\underline{\text{v}}^{GMAA}$ is updated. Each time a new CBG is constructed, it is built by extending the CBG for the parent node and then applying lossless clustering. However, rather than expanding all children, GMAA*-ICE requests only the next solution $\boldsymbol{\beta}$ of an *incremental CBG solver*. This next CBG solution $\boldsymbol{\beta}$ is then used to construct a single child $\boldsymbol{\varphi}^{t+1} = \langle \boldsymbol{\varphi}^t \circ \boldsymbol{\beta} \rangle$.

For the incremental CBG solver, we use the BaGaBaB algorithm [9], which performs a second (nested) A* search, but now over (partially specified) CBG policies.[3] The BaGaBaB solver for $\boldsymbol{\varphi}^t$ is initialized with lower bound

$$\underline{\text{v}}^{CBG} = \underline{\text{v}}^{GMAA} - V^{0\ldots(t-1)}(\boldsymbol{\varphi}^t), \tag{3}$$

[2]Furthermore, experiments confirmed that the overhead of potentially expanding a parent first is negligible.

[3]In principle GMAA*-ICE can use any CBG solver that is able to incrementally deliver all $\boldsymbol{\beta}$ in descending order of $\widehat{V}(\boldsymbol{\beta})$. However, there are not many such CBG solvers that can avoid enumerating all $\boldsymbol{\beta}$ before providing the first result.

Algorithm 2 GMAA*-ICE

1: $\underline{\text{v}}^{GMAA} \leftarrow -\infty$
2: $\boldsymbol{\varphi}^0 \leftarrow ()$, $\hat{v} \leftarrow +\infty$, PH $\leftarrow$ **false**, $q^0 \leftarrow \langle \boldsymbol{\varphi}^0, \hat{v}, \text{PH} \rangle$
3: $\text{P}^{\text{IE}} \leftarrow \{q^0\}$
4: **repeat**
5: $q \leftarrow$ `Select`(P^{IE}) {$q = \langle \boldsymbol{\varphi}^t, \hat{v}, \text{PH} \rangle$}
6: P^{IE}.pop(q)
7: **if** PH **then**
8: $B(\boldsymbol{\varphi}^t) \leftarrow \boldsymbol{\varphi}^t$.CBG {reuse stored CBG}
9: **else**
10: ConstructExtendedBGandSolver($\boldsymbol{\varphi}^t$) {(omitted)}
11: $\underline{\text{v}}^{CBG} = \underline{\text{v}}^{GMAA} - V^{0\ldots(t-1)}(\boldsymbol{\varphi}^t)$
12: $\bar{\text{v}}^{CBG} = +\infty$
13: **if** last stage $t = h - 1$ **then**
14: $\bar{\text{v}}^{CBG} = \widehat{V}(\boldsymbol{\varphi}^{h-1}) - V^{0\ldots(h-2)}(\boldsymbol{\varphi}^{h-1})$
15: $\langle \boldsymbol{\beta}^t, \widehat{V}(\boldsymbol{\beta}^t) \rangle \leftarrow B(\boldsymbol{\varphi}^t)$.Solver.NextSolution($\underline{\text{v}}^{CBG}$,$\bar{\text{v}}^{CBG}$)
16: **if not** $\boldsymbol{\beta}^t$ **then**
17: {fully expanded: no solution s.t. $V(\boldsymbol{\beta}^{h-1}) \geq \underline{\text{v}}^{CBG}$}
18: delete q and continue {(i.e., goto line 5)}
19: $\boldsymbol{\varphi}^{t+1} \leftarrow \langle \boldsymbol{\varphi}^t \circ \boldsymbol{\beta}^t \rangle$
20: $\widehat{V}(\boldsymbol{\varphi}^{t+1}) \leftarrow V^{0\ldots t-1}(\boldsymbol{\varphi}^t) + \widehat{V}(\boldsymbol{\beta}^t)$
21: **if** last stage $t = h - 1$ **then**
22: **if** $V(\boldsymbol{\pi}) > \underline{\text{v}}^{GMAA}$ **then**
23: $\underline{\text{v}}^{GMAA} \leftarrow V(\boldsymbol{\pi})$ {found new lower bound}
24: $\boldsymbol{\pi}^\star \leftarrow \boldsymbol{\pi}$
25: P^{IE}.prune($\underline{\text{v}}^{GMAA}$)
26: **else**
27: $q' \leftarrow \langle \boldsymbol{\varphi}^{t+1}, \widehat{V}(\boldsymbol{\varphi}^{t+1}), \textbf{false} \rangle$
28: P^{IE}.insert(q')
29: $q \leftarrow \langle \boldsymbol{\varphi}^t, \widehat{V}(\boldsymbol{\varphi}^{t+1}), \textbf{true} \rangle$ { Update parent node q }
30: P^{IE}.insert(q)
31: **until** P^{IE} is empty

and, in case of the last stage $t = h - 1$, upperbound

$$\bar{\text{v}}^{CBG} = \widehat{V}(\boldsymbol{\varphi}^{h-1}) - V^{0\ldots(h-2)}(\boldsymbol{\varphi}^{h-1}), \tag{4}$$

since

$$\widehat{V}(\boldsymbol{\varphi}^h) - V^{0\ldots(h-2)}(\boldsymbol{\varphi}^{h-1}) = \widehat{V}(\boldsymbol{\beta})$$
$$\widehat{V}(\boldsymbol{\varphi}^{h-1}) - V^{0\ldots(h-2)}(\boldsymbol{\varphi}^{h-1}) \geq \widehat{V}(\boldsymbol{\beta}) = V(\boldsymbol{\delta}^{h-1}|\boldsymbol{b}^0, \boldsymbol{\varphi}^{h-1}).$$

This can be used to stop expanding when we find a lower bound equal to the upper bound $\bar{\text{v}}^{CBG} = V(\boldsymbol{\beta})$, as in the original A*. Note that $\widehat{V}(\boldsymbol{\beta})$ is only a bound on values when solving the last stage: the last equality holds if $\widehat{Q}(\vec{\boldsymbol{\theta}}^{h-1}, \mathbf{a}) = R(\vec{\boldsymbol{\theta}}^{h-1}, \mathbf{a})$, i.e., the heuristic payoff function for the CBG reflects the actual expected reward. This means that the upper bound can only be used in solving the last-stage CBGs. Also note that each time when asking BaGaBaB for a next solution, $\underline{\text{v}}^{CBG}$ is reset by re-evaluating (3), because $\underline{\text{v}}^{GMAA}$ may have changed since the last solution was delivered. Then it continues searching (by selecting the heuristically best-ranked node from its own internal open list and proceeding as normal).

Algorithm 2 shows the pseudo-code for GMAA*-ICE. The main differences with Algorithm 1 are seen from line 7 to line 19. In this section, the algorithm first determines if a placeholder is being used and either reuses the current CBG solver or constructs a new one. Then, new bounds are calculated and the next solution is obtained, removing the node when all children with value above the lower bound have been expanded. Lastly, only a single child is generated rather than expanding all children as in Algorithm 1.

4. THEORETICAL GUARANTEES

We shall now prove some properties of GMAA*-ICE. We say that two search algorithms are *search-equivalent* if they select exactly the same set of nodes to expand in the search tree. That is, that they `Select` the same q for expansion on line 5 of Algorithm 1 and 2 (but the set of expanded nodes can be different). We will show that the IC and ICE variants are search-equivalent. To do so, we will talk about equivalence of the open lists maintained. The open list P maintained by IC only contains non-expanded nodes q. That of ICE, $\mathrm{P^{IE}}$, contains both non-expanded nodes q and placeholders (previously expanded nodes), $\bar{q}$. We use Q and $\bar{Q}$ to denote the respective (ordered) subsets of $\mathrm{P^{IE}}$. We think of these open lists as ordered sets of heuristic values and their associated nodes.

Definition 1. P and $\mathrm{P^{IE}}$ are *equivalent*, $\mathrm{P} \equiv \mathrm{P^{IE}}$, when:

1. $Q \subseteq \mathrm{P}$.
2. The q's have the same ordering: $\mathrm{P.remove}(\mathrm{P} \setminus Q) = Q$. ($A$.remove($B$) removes the elements of B from A without changing A's ordering.)
3. Nodes not present in $\mathrm{P^{IE}}$ instead have a placeholder, $\forall q = \langle \boldsymbol{\varphi}^t, \hat{v}_q, \text{false} \rangle \in (\mathrm{P} \setminus Q) : \exists \bar{q} = \langle \boldsymbol{\varphi}^{t-1}, \hat{v}_{\bar{q}}, \text{true} \rangle \in \bar{Q}$ such that: $\bar{q}$ is the parent of q ($\boldsymbol{\varphi}^t = \langle \boldsymbol{\varphi}^{t-1} \circ \boldsymbol{\beta} \rangle$), and $\bar{q}$ is more highly ranked: $\hat{v}_{\bar{q}} \geq \hat{v}_q$.[4]
4. There are no other placeholders.

Let us write IT-IC(P) and IT-ICE($\mathrm{P^{IE}}$) for one iteration of the respective algorithms. Let IT-ICE* denote the operation that repeats IT-ICE as long as a placeholder was selected (so it ends when a q is expanded).

LEMMA 2. *If $P \equiv P^{IE}$, then executing IT-IC(P) and IT-ICE*(P^{IE}) will lead to new open lists that again are equivalent: $P' \equiv P^{IE\prime}$.*

PROOF. *When IT-ICE* selects a placeholder $\bar{q}$, it will generate child q' that was already present in P (due to property 3 and 4 of def. 1) and insert it at the proper location, thereby preserving properties 1 and 2.*[5] *If there are remaining unexpanded children of $\bar{q}$, IT-ICE* will reinsert $\bar{q}$ with an updated heuristic value $\bar{q}.\hat{v} \leftarrow q'.\hat{v}$ which is guaranteed to upper bound the value of unexpanded siblings q'' since $q'.\hat{v} = \widehat{V}(q'.\boldsymbol{\varphi}) \geq \widehat{V}(q''.\boldsymbol{\varphi}) = q''.\hat{v}$ (preserving properties 3 and 4).*

When IT-ICE finally selects a non-placeholder q, it is guaranteed to be the same q as selected by IT-IC (due to property 1 and 2). Expansion in ICE will generate 1 child q' (again, inserted at the same relative location as in IC) and insert placeholder $\bar{q} = \langle q.\boldsymbol{\varphi}, q'.\hat{v}, true \rangle$ for the other siblings q'' (again preserving properties 3 and 4).* □

THEOREM 1. GMAA*-ICE *and* GMAA*-IC *are search-equivalent.*

PROOF. *This follows directly from the proof of Lemma 2: Both algorithms initialize with the same (equivalent) open list and therefore maintain equivalent open lists throughout search. At each point IT-ICE(P^{IE}) will either select a $\bar{q} = \langle \boldsymbol{\varphi}, \hat{v}, true \rangle$—then IC also expanded a node for $\boldsymbol{\varphi}$—or a q. In the last case, because of property (2) of def. 1 we know that the same q is selected by IT-IC(P).* □

Note that Theorem 1 does not mean that the run time and space requirements of GMAA*-ICE and GMAA*-IC are identical: for each expansion, GMAA*-ICE will only generate one child node to be stored on the open list versus a number of child nodes that is, in the worst case, doubly exponential in the depth of the selected node.[6] On the other hand, GMAA*-ICE may select a placeholder for further expansion (in the worst case all child nodes will still have to be generated).

We say that a search algorithm is complete if it searches until it finds an optimal solution.

COROLLARY 1. *When using a heuristic of the form*

$$\widehat{Q}(\vec{\boldsymbol{\theta}}^t, \mathbf{a}) = \mathbf{E}[R(s, \mathbf{a}) \mid \vec{\boldsymbol{\theta}}^t] + \mathbf{E}[\widehat{V}(\vec{\boldsymbol{\theta}}^{t+1}) \mid \vec{\boldsymbol{\theta}}^t, \mathbf{a}], \qquad (5)$$

where $\widehat{V}(\vec{\boldsymbol{\theta}}^{t+1}) \geq Q_{\boldsymbol{\pi}^*}(\vec{\boldsymbol{\theta}}^{t+1}, \boldsymbol{\pi}^*(\vec{\boldsymbol{\theta}}^{t+1}))$ is an overestimation of the value of an optimal joint policy $\boldsymbol{\pi}^*$, GMAA*-ICE *is complete.*

PROOF. *Under the stated conditions,* GMAA*-IC *is complete [10, 11]. Since* GMAA*-ICE *is search equivalent to* GMAA*-IC*, it is also complete.* □

5. HEURISTIC REPRESENTATION

As with any heuristic search method, the effectiveness of MAA* and variations depends on a high-quality admissible heuristic function. First we will briefly review existing heuristics, after which we introduce new, more scalable, representations.

5.1 Existing Heuristics

One way to obtain a heuristic $\widehat{Q}(\boldsymbol{\theta}, \mathbf{a})$ is via solving the underlying MDP, known as $\mathrm{Q_{MDP}}$ [8]. Similar to the underlying MDP, one can define the underlying POMDP of a Dec-POMDP and its solution can be used as a heuristic, called $\mathrm{Q_{POMDP}}$ [13, 16]. $\mathrm{Q_{POMDP}}$ computes a value $Q^t_{\mathrm{P}}(\boldsymbol{b}^{\vec{\boldsymbol{\theta}}^t}, \mathbf{a})$ which can directly be used as a heuristic:

$$\widehat{Q}_{\mathrm{P}}(\vec{\boldsymbol{\theta}}^t, \mathbf{a}) \equiv Q^t_{\mathrm{P}}(\boldsymbol{b}^{\vec{\boldsymbol{\theta}}^t}, \mathbf{a}). \qquad (6)$$

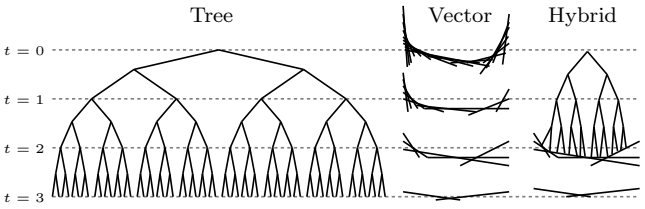
For a finite horizon, there are two approaches to computing $\mathrm{Q_{POMDP}}$. First, it is possible to construct the 'belief MDP tree': the tree of all joint beliefs (induced by all joint AOHs, illustrated in Fig. 2(a)(left)). This is conceptually simple: starting with $\boldsymbol{b}^0$ corresponding to the empty joint action-observation history $\vec{\boldsymbol{\theta}}^{t=0}$, for each $\mathbf{a}$ and $\mathbf{o}$ compute the resulting $\vec{\boldsymbol{\theta}}^{t=1}$ and corresponding belief $\boldsymbol{b}^{\vec{\boldsymbol{\theta}}^1}$ and continue recursively. Given this tree, it is possible to compute values for all the nodes by standard dynamic programming.

Second, it is possible to apply vector-based POMDP techniques (Fig. 2(a)(middle)) [6]. The Q-value function for a stage $Q^t_{\mathrm{P}}(\boldsymbol{b}, \mathbf{a})$ can be represented using a set of vectors for

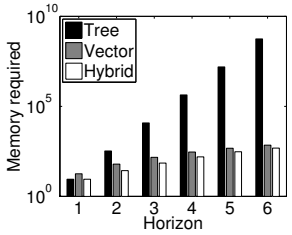
[4] Again, we assume a total ordering on the nodes such that ties in value are broken consistently (and, in this particular case, such that on equality $\hat{v}_{\bar{q}} = \hat{v}_q$, the child is ranked higher to ensure it is expanded before its parent).

[5] This is the same location as IT-IC, as heuristic values are independent of the search process, and ties are dealt with consistently: two $\boldsymbol{\varphi}^t$ with same $\hat{v}$ are always ordered the same (in the open list and by the node expansion).

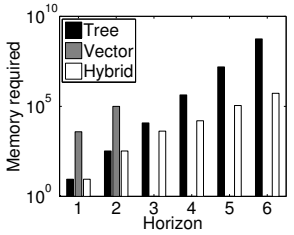
[6] When a problem allows clustering, the number of child nodes grows less dramatically.



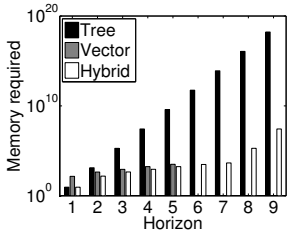
(a) Comparison of Q representations.



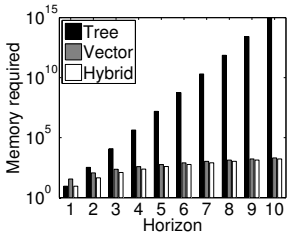
(b) Dec-Tiger.



(c) FireFighting.



(d) Hotel 1.



(e) Recycling Robots.

Figure 2: Hybrid representations. (a) Comparison of different representations of heuristic Q functions. (b)-(e) The number of real numbers stored for different representations of $\mathbf{Q_{BG}}$.

each joint action $\mathcal{V}^t = \{\mathcal{V}^t_1, \ldots, \mathcal{V}^t_{|\mathcal{A}|}\}$. $Q^t_{\mathrm{P}}(\boldsymbol{b},\mathbf{a})$ is then defined as the maximum inner product:

$$Q^t_{\mathrm{P}}(\boldsymbol{b},\mathbf{a}) \equiv \max_{v^t_{\mathbf{a}} \in \mathcal{V}^t_{\mathbf{a}}} \boldsymbol{b} \cdot v^t_{\mathbf{a}}.$$

Given $\mathcal{V}^{h-1}$, the vector representation of the last stage (one vector for each joint action), it is possible to compute $\mathcal{V}^{h-2}$, etc. In order to limit the growth of the number of vectors needed to represent the stages, dominated vectors may be pruned. It is well-known that $\mathrm{Q_{MDP}}$ is an upper bound to the POMDP value function. Therefore, $\mathrm{Q_{POMDP}}$ provides a tighter upper bound to Q^* than $\mathrm{Q_{MDP}}$. However, it is also more costly to compute and store: both approaches may need to store a number of values exponential in h.

Finally, a third heuristic commonly used in MAA* is $\mathrm{Q_{BG}}$: the value function that results from assuming 1-step-delayed communication. Such value functions can also be represented using vectors [5, 10], so the same two methods of computation apply here. However, note that $\mathrm{Q_{BG}}$ is tighter than $\mathrm{Q_{POMDP}}$: $Q^* \leq \mathrm{Q_{BG}} \leq \mathrm{Q_{POMDP}} \leq \mathrm{Q_{MDP}}$ [10].

5.2 Hybrid Representations

Previous research indicated that the upper bound provided by $\mathrm{Q_{MDP}}$ is often too loose for effective heuristic search in MAA* [10]. However, for tighter heuristics such as $\mathrm{Q_{BG}}$ or $\mathrm{Q_{POMDP}}$ the space needed to store these heuristics grows exponentially with the horizon, as explained before.

In practice, we experienced that the exponential space requirements to compute the heuristics become a bottleneck

Algorithm 3 Compute Hybrid $\widehat{Q}$ with minimum size.

```
1: Q^{h-1} ← {R^1, ..., R^{|A|}}   {vector representation of last stage}
2: z ← |A| × |S|                    {the size of the |A| vectors}
3: for t = h − 2 to 0 do
4:    y ← |Θ⃗^t| × |A|               {size of AOH representation}
5:    if z < y then
6:       V ← VectorBackup(Q^{t+1})
7:       V' ← Prune(V)
8:       Q^t ← V'
9:       z ← |V'| × |S|
10:   else
11:      Q^t ← TreeBackup(Q^{t+1})   {From now on z ≥ y}
```

regarding the problems we can solve. To mitigate this problem we introduce heuristics with a hybrid representation, as illustrated in Fig. 2(a)(right). The key insight is that the exponential growth of the discussed representations is in opposite directions. Therefore we can use the low-space-complexity side of both representations: the later stages use a vector-based representation (and later stages have fewer vectors), while the earlier stages use a history-based representation (and earlier stages have fewer histories). Algorithm 3 shows how a minimally-sized representation can easily be computed.

Fig. 2(b)-(e) illustrate the power of combining vector with tree-based representations, by plotting the memory requirements (in terms of number of parameters) of the "Tree", the "Vector" ($\mathrm{Q_{POMDP}}$), and the "Hybrid" representation for $\mathrm{Q_{BG}}$, where missing "Vector" bars indicate those representations grew beyond limits. The hybrid representation is computed following Algorithm 3, and the vector-based $\mathrm{Q_{BG}}$ representation is computed using a variation of Incremental Pruning. The pruning performance depends on the problem and the complexity of the value function, which can increase suddenly, as for instance happens in Fig. 2(d). We see that for several benchmark problems the hybrid representation allows for very significant savings in memory space, allowing us to compute tight heuristics for longer horizons.

6. EXPERIMENTS

We performed an empirical evaluation of GMAA*-ICE by comparing to GMAA*-IC. This way we are able to assess the impact of the proposed incremental expansion without additional differences. Moreover, GMAA*-IC is currently (one of) the fastest optimal solvers for finite-horizon Dec-POMDPs.[7] Unless noted otherwise, we used $\mathrm{Q_{BG}}$ with a hybrid representation. We tested on a suite of benchmark problems from literature [11], using discount factor $\gamma = 1.0$ for all problems.[8] GMAA*-IC uses a brute-force solver that enumerates and evaluates all solutions (as in the original MAA*), while GMAA*-ICE uses BaGaBaB [9] (with joint types ordered according to increasing probability). Experiments were run on an Intel iCore5 CPU running Linux, and we limited each process to 2Gb of RAM and a maximum computation time of 3,600s. Reported CPU-times are averaged over 10 independent runs and have a resolution of 0.01s. They concern only the MAA* search process, since computation of the heuristic is the same for both methods

[7]The method proposed in [2] effectively focuses on state space reachability in problem structure.

[8]All problem definitions are available at `http://www.isr.ist.utl.pt/~mtjspaan/decpomdp`.

h	V^*	T_{IC}(s)	T_{ICE}(s)
Dec-Tiger			
2	−4.000000	≤ 0.01	≤ 0.01
3	5.190812	≤ 0.01	≤ 0.01
4	4.802755	0.27	≤ 0.01
5	7.026451	21.03	0.02
6	**10.381625**	–	46.43
7		–	∗
FireFighting $\langle n_h = 3, n_f = 3 \rangle$			
2	−4.383496	≤ 0.01	≤ 0.01
3	−5.736969	0.11	0.10
4	−6.578834	950.51	1.00
5	**−7.069874**	–	4.40
6	**−7.175591**	0.08	0.07
7		#	#
GridSmall			
2	0.910000	≤ 0.01	≤ 0.01
3	1.550444	0.10	≤ 0.01
4	2.241577	1.77	≤ 0.01
5	**2.970496**	–	0.02
6	**3.717168**	–	0.04
7		#	#

h	V^*	T_{IC}(s)	T_{ICE}(s)
Hotel 1			
2	10.000000	≤ 0.01	≤ 0.01
3	16.875000	≤ 0.01	≤ 0.01
4	22.187500	≤ 0.01	≤ 0.01
5	**27.187500**	≤ 0.01	≤ 0.01
6	**32.187500**	≤ 0.01	≤ 0.01
7	**37.187500**	≤ 0.01	≤ 0.01
8	**42.187500**	≤ 0.01	≤ 0.01
9	**47.187500**	0.02	≤ 0.01
10		#	#
Recycling Robots			
5	16.486000	≤ 0.01	≤ 0.01
15	47.248521	≤ 0.01	≤ 0.01
18	**56.479290**	≤ 0.01	≤ 0.01
20	**62.633136**	≤ 0.01	≤ 0.01
30	**93.402367**	0.08	0.05
40	**124.171598**	0.42	0.25
50	**154.940828**	2.02	1.27
60	**185.710059**	9.70	6.00
70	**216.479290**	–	28.66
80		–	–

h	V^*	T_{IC}(s)	T_{ICE}(s)
Cooperative Box Pushing			
2	17.600000	≤ 0.01	≤ 0.01
3	66.081000	0.11	≤ 0.01
4	98.593613	∗	313.07
5		#	#
BroadcastChannel			
5	4.790000	≤ 0.01	≤ 0.01
10	9.290000	≤ 0.01	≤ 0.01
20	18.313228	≤ 0.01	≤ 0.01
25	22.881523	≤ 0.01	≤ 0.01
30	**27.421850**	≤ 0.01	≤ 0.01
50	**45.501604**	≤ 0.01	≤ 0.01
53	**48.226420**	≤ 0.01	≤ 0.01
100	**90.760423**	≤ 0.01	≤ 0.01
250	**226.500545**	0.06	0.07
500	**452.738119**	0.81	0.94
600	**543.228071**	11.63	13.84
700	**633.724279**	0.52	0.63
800		–	–
900	**814.709393**	9.57	11.11
1000		–	–

Table 1: Experimental results comparing the computation times of GMAA*-IC (T_{IC}) and GMAA*-ICE (T_{ICE}), using the hybrid $\mathbf{Q_{BG}}$ representation. Memory limit violations are indicated by "–" while time limit overruns are shown as "∗". Furthermore, "#" indicates computing the heuristic exceeded memory or time limits. Bold entries highlight results for which no previous solution was known in literature. Boxed entries in the T_{IC} and T_{ICE} column indicates the maximum planning horizon that can be solved by GMAA*-IC resp. GMAA*-ICE when using the $\mathbf{Q_{MDP}}$ heuristic, given identical memory and time limits.

and can be amortized over multiple runs.[9]

The main results are listed in Table 1. It clearly shows that incremental expansion combined with the hybrid representations allows for significant improvements over the state of the art: for the vast majority of problems tested we provide results for longer horizons than any previously known (the bold entries). Thus, incorporating the hybrid representation into GMAA*-IC greatly increases its scalability, while adding the incremental expansion of GMAA*-ICE results in even more performance improvements. When comparing against GMAA*-IC, for Dec-Tiger we see that for $h = 5$ GMAA*-ICE achieves a speedup of 3 orders of magnitude, and it is also able to compute a solution for $h = 6$, unlike GMAA*-IC. For GridSmall we see a large speedup for $h = 4$ and very fast solutions for $h = 5, 6$, where GMAA*-IC runs out of memory. Similar positive results are obtained for Cooperative Box Pushing and FireFighting. An interesting counter-intuitive behavior can be observed for FireFighting, $h = 6$, which could be solved much faster than $h = 5$. Analysis reveals that the CBG instances encountered during the $h = 6$ search happen to cluster much better than the CBGs in the $h = 5$ search, which is possible because the heuristics vary with the horizon. Also for BroadcastChannel we can see that the search process is not necessarily monotonic in the planning horizon.

Due to the hybrid representation we can compute $\mathrm{Q_{BG}}$ heuristics for all these problems and horizons, and as a consequence our results, *also for GMAA*-IC*, are much better. Previous work often had to resort to $\mathrm{Q_{MDP}}$ for high horizons and/or large problems [11]. For instance, for Hotel 1 $h = 5$ a tree-based $\mathrm{Q_{BG}}$ representation (as used in [11]) would already occupy 29Gb. The boxed entries in Table 1 show the limits of running GMAA*-IC and GMAA*-ICE using $\mathrm{Q_{MDP}}$ instead of $\mathrm{Q_{BG}}$: in most of these problems we can reach longer horizons with $\mathrm{Q_{BG}}$. Only for FireFighting GMAA*-ICE with $\mathrm{Q_{MDP}}$ can compute solutions for higher h than possible with $\mathrm{Q_{BG}}$ (hence the missing box). In fact, the V^* for $h > 6$ are equal to the one for $h = 6$, as the optimal joint policy is guaranteed to extinguish all fires in 6 time steps, after which no non-zero rewards can be accrued.

The efficacy of a hybrid representation can be clearly seen for problems like GridSmall, Cooperative Box Pushing, FireFighting and Hotel 1 (for the latter two see Fig. 2(c) resp. 2(d)), where neither the tree nor the vector representation is able to provide a compact $\mathrm{Q_{BG}}$ heuristic for longer horizons. Apart from FireFighting, for these problems computing and storing $\mathrm{Q_{BG}}$ (or another tight heuristic) for longer horizons forms the bottleneck for scaling further.

As a final note regarding Table 1, we point out that only on the BroadcastChannel problem GMAA*-IC is (slightly) faster than GMAA*-ICE. Because this problem exhibits clustering to a single joint type [11], the overhead of incremental expansion does not pay off (cf. footnote 6).

Summarizing our main results, we can conclude that 1) GMAA*-ICE outperforms GMAA*-IC leading to solutions of longer horizons in many problems, 2) both methods benefit from the improved heuristic representation, 3) in several problems computation and representation of the heuristic is the bottleneck that prevents from scaling further. The last point implies that our method may scale even further when

[9]The heuristics' computation time ranges from less than a second to many hours (for high h in some difficult problems).

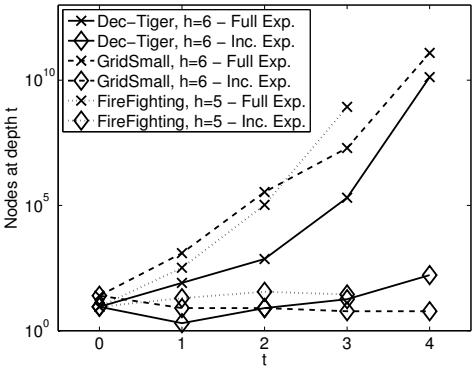


Figure 3: Number of expanded partial joint policies φ^t for intermediate stages $t = 0, \ldots, h-2$.

the computation of the heuristic is further improved.

Finally, we also investigated the impact of incremental expansion in terms of the number of nodes that are actually expanded for intermediate stages $t = 0, \ldots, h-2$. Fig. 3 shows the number of nodes expanded in GMAA*-ICE and the number that would be expanded for GMAA*-IC (which can be easily computed as they are search-tree equivalent). There is a clear relation between the results from Fig. 3 and Table 1. For example, it clearly illustrates why GMAA*-IC runs out of memory on GridSmall $h = 6$. The plots confirm our initial hypothesis that in practice only a small number of child nodes are being queried.

7. CONCLUSIONS & FUTURE WORK

Decentralized POMDPs offer a rich model for multiagent coordination under uncertainty. Optimal solution methods for Dec-POMDPs are of great interest; they are of practical value for smaller or decomposable problems and lie at the basis for most successful approximate methods [4, 14, 17]. In this paper, we advance the state of the art by introducing an effective method for *incremental expansion* of nodes in the search tree. We proved that the resulting algorithm, GMAA*-ICE, is search-equivalent to GMAA*-IC and therefore complete. A new bottleneck, the amount of space needed for representation of the heuristic, was addressed by introducing representations that are a hybrid between tree-based and vector-based representations.

We demonstrated our approach experimentally with and without incremental expansion, showing that its effect is complementary to clustering of histories. With just the new heuristic representation, optimal plans could be found for larger horizons than any known previous work for four benchmarks. In one case, horizons that are over an order of magnitude larger could be reached. By exploiting incremental expansion, GMAA*-ICE achieves further improvements in scalability. The combination of the hybrid representation and incremental expansion provides a powerful method for optimally solving DEC-POMDP over longer horizons.

Some possible extensions of this work the following. First, to quickly compute good lower bounds GMAA*-ICE may use weighted heuristics, which were of little practical value in the original MAA* as expanding single nodes was too expensive [15]. Second, we may consider improving the current CBG solver or try to adapt other CBG solvers, e.g., [7]. Third, incremental solvers for graphical CBGs may allow for further scaling of optimal solutions of Dec-POMDPs with multiple agents. Finally, future work should further consider improved heuristics and methods of computation, which can allow GMAA*-ICE to scale even further.

Acknowledgments

We would like to thank Anthony Cassandra for his pomdp-solve code (used for vector pruning), and the reviewers for their insightful suggestions. This work was funded in part by Fundação para a Ciência e a Tecnologia (ISR/IST pluriannual funding) through the PIDDAC Program funds and was supported by projects PTDC/EEA-ACR/73266/2006 and CMU-PT/SIA/0023/2009 (the latter under the Carnegie Mellon-Portugal Program). Research supported in part by AFOSR MURI project #FA9550-09-1-0538.

8. REFERENCES

[1] M. Allen and S. Zilberstein. Agent influence as a predictor of difficulty for decentralized problem-solving. In *AAAI*, 2007.

[2] C. Amato, J. Dibangoye, and S. Zilberstein. Incremental policy generation for finite-horizon DEC-POMDPs. In *ICAPS*, 2009.

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

[4] R. Emery-Montemerlo, G. Gordon, J. Schneider, and S. Thrun. Approximate solutions for partially observable stochastic games with common payoffs. In *AAMAS*, 2004.

[5] K. Hsu and S. Marcus. Decentralized control of finite state Markov processes. *IEEE Transactions on Automatic Control*, 27(2):426–431, Apr. 1982.

[6] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1-2):99–134, 1998.

[7] A. Kumar and S. Zilberstein. Point-based backup for decentralized POMDPs: Complexity and new algorithms. In *AAMAS*, 2010.

[8] M. L. Littman, A. R. Cassandra, and L. P. Kaelbling. Learning policies for partially observable environments: Scaling up. In *ICML*, 1995.

[9] F. A. Oliehoek, M. T. J. Spaan, J. Dibangoye, and C. Amato. Heuristic search for identical payoff Bayesian games. In *AAMAS*, 2010.

[10] F. A. Oliehoek, M. T. J. Spaan, and N. Vlassis. Optimal and approximate Q-value functions for decentralized POMDPs. *Journal of Artificial Intelligence Research*, 32:289–353, 2008.

[11] F. A. Oliehoek, S. Whiteson, and M. T. J. Spaan. Lossless clustering of histories in decentralized POMDPs. In *AAMAS*, 2009.

[12] Z. Rabinovich, C. V. Goldman, and J. S. Rosenschein. The complexity of multiagent systems: the price of silence. In *AAMAS*, 2003.

[13] M. Roth, R. Simmons, and M. Veloso. Reasoning about joint beliefs for execution-time communication decisions. In *AAMAS*, pages 786–793, 2005.

[14] S. Seuken and S. Zilberstein. Memory-bounded dynamic programming for DEC-POMDPs. In *IJCAI*, 2007.

[15] S. Seuken and S. Zilberstein. Formal models and algorithms for decentralized decision making under uncertainty. *Autonomous Agents and Multi-Agent Systems*, 17(2):190–250, 2008.

[16] D. Szer, F. Charpillet, and S. Zilberstein. MAA*: A heuristic search algorithm for solving decentralized POMDPs. In *UAI*, 2005.

[17] F. Wu, S. Zilberstein, and X. Chen. Online planning for multi-agent systems with bounded communication. *Artificial Intelligence*, 175(2):487–511, 2011.