

The Cross-Entropy Method for Policy Search in Decentralized POMDPs

Frans A. Oliehoek and Julian F.P. Kooij
 Intelligent Systems Lab, University of Amsterdam
 Amsterdam, The Netherlands
 E-mail: {faolieho,jkooij}@science.uva.nl, www.science.uva.nl/~faolieho

Nikos Vlassis
 Dept. of Production Engineering and Management
 Technical University of Crete
 Greece
 E-mail: vlassis@dpem.tuc.gr, http://www.dpem.tuc.gr/vlassis

Keywords: multiagent planning, decentralized POMDPs, combinatorial optimization

Received: March 15, 2008

Decentralized POMDPs (Dec-POMDPs) are becoming increasingly popular as models for multiagent planning under uncertainty, but solving a Dec-POMDP exactly is known to be an intractable combinatorial optimization problem. In this paper we apply the Cross-Entropy (CE) method, a recently introduced method for combinatorial optimization, to Dec-POMDPs, resulting in a randomized (sampling-based) algorithm for approximately solving Dec-POMDPs. This algorithm operates by sampling pure policies from an appropriately parametrized stochastic policy, and then evaluates these policies either exactly or approximately in order to define the next stochastic policy to sample from, and so on until convergence. Experimental results demonstrate that the CE method can search huge spaces efficiently, supporting our claim that combinatorial optimization methods can bring leverage to the approximate solution of Dec-POMDPs.

Povzetek: Prispevek opisuje novo metodo multiagentnega načrtovanja.

1 Introduction

The construction of intelligent agents is one of the major goals in Artificial Intelligence. In the last two decades more and more research has concentrated on systems with multiple intelligent agents, or multiagent systems (MASs). In this article we focus on the recently proposed model of decentralized partially observable Markov decision processes (Dec-POMDPs) (7). A Dec-POMDP is a generalization to multiple agents of the well-known POMDP model for single-agent planning under uncertainty (19).

The Dec-POMDP model presents a decision theoretic formalization of multiagent planning under uncertainty. It models a team of cooperative agents that individually receive observations of their environment and cannot communicate. It is a very general model which has for instance been applied in the context of cooperative robotics (5; 14), communication networks (30), and sensor networks (26). The Dec-POMDP does not explicitly consider communication, but communication actions can be modeled as regular actions. Also, there are extensions that do explicitly incorporate communication (32), which can typically be used to share observations.

In this paper we focus on the regular Dec-POMDP setting in which the agents have to select their action based on their private observation histories. During an off-line plan-

ning phase we try to find a policy for a fixed number of time steps for each agent. The profile of all individual policies is collectively called a joint policy. The policies should be selected such that, when they are executed jointly during the on-line execution phase, the resulting behavior is (near-) optimal according to a predefined performance measure.

In the single-agent (i.e., POMDP) case, the history of observations results in a ‘belief’ (a probability distribution) over states, which forms a sufficient statistic for the history and can therefore be used to base the action choice on (19). In the multiagent case, however, no such simplification is possible, and the agents have to base their actions on their entire history. This means that the number of possible deterministic policies for a single agent is finite, but grows doubly exponentially with the planning horizon. Planning for a Dec-POMDP requires finding a profile of such policies, one for each agent, which defines a very challenging combinatorial optimization problem.

In this paper we examine the *Cross-Entropy (CE) method*, a recently introduced and promising method for combinatorial optimization problems (12) and its application to Dec-POMDPs. The CE method provides a framework for finding approximate solutions to combinatorial problems, and has gained popularity for various applications (24; 9; 1; 11), due to its ability to find near-optimal solutions in huge search spaces. Because of the combinato-

rial nature of the decentralized setting, the CE method may be a valuable tool in many algorithms for Dec-POMDPs.

In this paper we show how the CE-method can be applied in the setting of Dec-POMDPs using a relatively direct translation of the general CE optimization method. This, however, is not entirely trivial, so we discuss the difficulties and propose solutions. Our primary goal in this paper is not to provide a state-of-the art method for solving Dec-POMDPs. In fact, there are some algorithms which are more advanced than the one we present here (14; 35; 34; 3; 28). Rather, we show how the CE method can be used to tackle the combinatorial nature of the problem. This will allow for improvement of existing algorithms in two related ways. First, existing methods may rely on exhaustive evaluation of restricted sets of (partial) policies. The CE method may help to speed up these parts of algorithms. Second, such speed-up may allow the consideration of less restricted sets of policies, increasing the accuracy of the method.

1.1 Related work

In the last decade, planning for multiagent systems under uncertainty has received considerable attention. The Dec-POMDP model has been introduced by Bernstein et al. (6, 7), who also proved that finding a solution for a finite horizon Dec-POMDP is NEXP-complete. The infinite horizon problem is undecidable, as it is for the single agent case (i.e., solving a regular POMDP) (23). Pynadath and Tambe (32) introduced the multiagent team decision problem (MTDP), an equivalent model to the Dec-POMDP and a communicative extension, the COM-MTDP.

There are three main algorithms for the optimal solution of finite-horizon Dec-POMDPs apart from brute-force policy search. Hansen et al. (17) proposed dynamic programming (DP) for Dec-POMDPs, an algorithm that works by incrementally constructing policies for longer horizons. The second is multiagent A* (MAA*), a form of heuristic search (38). Finally Aras et al. (4) recently proposed a method based on mixed integer linear programming.

Due to the complexity results for Dec-POMDPs, however, the focus of much research has shifted to two other topics. First, people have tried to identify special instances of the problem that are less hard. For instance, a lot of research has focused on the case where there are special assumptions on observability and/or communication (10; 16; 21; 33; 36). Becker et al. (5) considered the special case of Dec-POMDPs where agents have their local state spaces and transitions and observations are independent. Similar assumptions are made by Nair et al. (26); Kim et al. (20); Varakantham et al. (39) who exploit resulting locality of interaction in the context of sensor networks. Other special cases are identified by Goldman and Zilberstein (15).

Second, research has focused on methods that find approximate solutions. For example, joint equilibrium search for policies (JESP) is a method that settles for a local optimum (25). Emery-Montemerlo et al. (13) proposed to con-

struct a policy by approximating the Dec-POMDP with a series of compressed Bayesian games (BGs). Both Szer and Charpillet (37) and Seuken and Zilberstein (35, 34) proposed approximate extensions of DP for Dec-POMDPs.

Except for JESP, all these approximate methods in one way or another restrict the search space to provide leverage. This reduced search space is then searched exhaustively to find the best approximate solution. In this work, rather than searching a constrained policy space exhaustively, we examine ways to search the entire space approximately. In particular, we show how the CE method can be used to directly search spaces of up to 10^{243} joint policies fairly effectively.

1.2 Overview of article

Section 2 provides a background on decentralized decision making and formally introduces the Dec-POMDP framework. In section 3 we treat the background of the CE method. Section 4 introduces direct CE policy search for Dec-POMDPs, detailing how we apply the CE method to Dec-POMDPs. Section 5 introduces an extension of our algorithm with approximate evaluation. In section 6 we give an experimental evaluation and finally section 7 concludes and discusses future work.

2 Decentralized POMDPs

Here we provide a formal background of decentralized POMDPs. We start by introducing the decentralized tiger (DEC-TIGER) problem, which is a standard benchmark. Next, we introduce the formal Dec-POMDP model. Finally, we formalize histories, policies and the optimality criterion and we discuss naive (i.e., brute force) policy search.

2.1 The decentralized tiger problem

In the DEC-TIGER problem, two agents standing in a hallway are confronted with two doors. Behind one door lies a treasure, while behind the other there is a tiger. As such, there are two possible states the world can be in: s_l , the tiger is behind the left door, and s_r , the tiger is behind the right door. Initially, these states have equal probability. The goal is to open the door that holds the treasure, which would result in a positive reward. But if at least one of them opens the other (wrong) door they receive a large penalty.

Both agents can take three actions, namely `OpenLeft` (a_{OL}) and `OpenRight` (a_{OR}) to open the left or right door, and `Listen` (a_{Li}) to try to observe carefully behind what door the tiger growls. The two possible observations `HearLeft` (o_{HL}) and `HearRight` (o_{HR}) indicate whether the agent heard the tiger behind the left or right door respectively. Observations are received at every time step but are random and uninformative unless *both* agents performed `Listen`, in which case they hear the tiger behind the correct door with high probability (each agent has a 85% chance of

getting the correct observation). The action Listen has a small cost associated with it, but can decrease the agents’ uncertainty about the state (which remains unchanged).

Furthermore, the rewards are specified such that it is always beneficial for the agents to act jointly: it is better to open the wrong door jointly than doing it alone. After one or both agents has opened a door, rewards are given and the situation is reset to a random state. More details on the DEC-TIGER problem are provided by Nair et al. (25).

2.2 The formal model

The *decentralized partially observable Markov decision process (Dec-POMDP)* describes a stochastic, partially observable environment for a set of cooperating agents.

Definition 2.1. A Dec-POMDP is a tuple $\langle Ag, S, \mathcal{A}, T, R, \mathcal{O}, O \rangle$ where:

- $Ag = \{1, \dots, n\}$ is the set of agents.
- S is a finite set of states.
- The set $\mathcal{A} = \times_i \mathcal{A}_i$ 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 $\mathbf{a} = \langle a_1, \dots, a_n \rangle$ is taken.¹
- T is the transition function, a mapping from states and joint actions to probability distributions over next states: $T : S \times \mathcal{A} \rightarrow \mathcal{P}(S)$.²
- R is the reward function, a mapping from states and joint actions to real numbers: $R : S \times \mathcal{A} \rightarrow \mathbb{R}$.
- $\mathcal{O} = \times_i \mathcal{O}_i$ is the set of *joint observations*, with \mathcal{O}_i the set of observations available to agent i . Every time step one joint observation $\mathbf{o} = \langle o_1, \dots, o_n \rangle$ is received.
- O is the observation function, a mapping from joint actions and successor states to probability distributions over joint observations: $O : \mathcal{A} \times S \rightarrow \mathcal{P}(\mathcal{O})$.

A Dec-POMDP is considered at a number of discrete time steps, or *stages*, t . At every such stage each agent i takes an individual action a_i . As a result of the taken *joint* action \mathbf{a} , the state then stochastically transitions from s to a new state s' according to T . At that point, the environment emits a joint observation \mathbf{o} with probability $P(\mathbf{o}|\mathbf{a}, s')$, as specified by O . From this \mathbf{o} each agent i observes its individual component o_i , selects a new action, etc.

In this paper we are searching for plans that specify actions for a fixed number of stages h . That is, we assume a finite *planning horizon* of h time steps. Furthermore, we assume that there is a distribution over states at the initial stage $t = 0$, also called the initial ‘belief’ $b^0 \in \mathcal{P}(S)$.³

¹Unless stated otherwise, subscripts denote agent indices.
²We use $\mathcal{P}(X)$ to denote the infinite set of probability distributions over the finite set X .
³Unless stated otherwise, superscripts denote time indices.

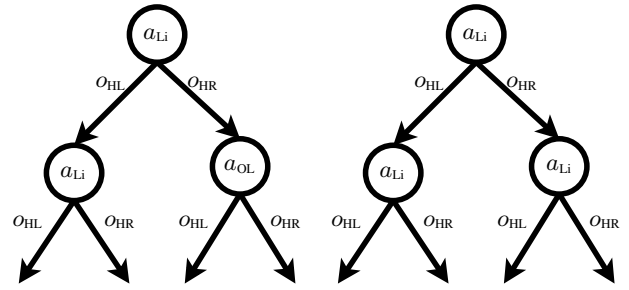


Figure 1: A part of an arbitrary joint policy for DEC-TIGER. Left shows to policy for agent 1, right for agent 2. The figure shows the actions taken at stages 0, 1 and shows the observations received at stages 1, 2.

2.3 Histories and policies

In a Dec-POMDP, an agent i only knows its own taken actions a_i and observations o_i . As a result it has to base its actions on the histories of those. Therefore, before we formalize the notion of a policy, we first formalize these histories.

Definition 2.2. The *action-observation history for agent i* $\vec{\theta}_i^t$ is the sequence of actions taken and observations received by agent i until time step t :

$$\vec{\theta}_i^t = (a_i^0, o_i^1, a_i^1, \dots, a_i^{t-1}, o_i^t). \quad (2.1)$$

The *joint action-observation history* is a tuple with the action-observation history for all agents $\vec{\theta}^t = \langle \vec{\theta}_1^t, \dots, \vec{\theta}_n^t \rangle$. The set of all action-observation histories for agent i at time t is denoted $\vec{\Theta}_i$.

Definition 2.3. The *observation history for agent i* is the sequence of observations an agent has received:

$$\vec{o}_i^t = (o_i^1, \dots, o_i^t), \quad (2.2)$$

\vec{o}^t denotes a joint observation history and $\vec{\mathcal{O}}_i$ denotes the set of all observation histories for agent i .

Now we are ready to formalize policies.

Definition 2.4 (deterministic policy). A *pure or deterministic policy* π_i for agent i in a Dec-POMDP is a mapping from observation histories to actions, $\pi_i : \vec{\mathcal{O}}_i \rightarrow \mathcal{A}_i$. A pure joint policy $\pi = \langle \pi_1 \dots \pi_n \rangle$ is a tuple containing a pure policy for each agent. Π is the set of all pure joint policies.

One could expect that the most general definition of a policy is as a mapping from action-observation histories to actions. This is indeed the case for *stochastic policies* where an action is chosen at each action-observation history with a particular probability. For a deterministic policy, however, this is unnecessary because an agent can infer the actions it took from its observation history.

A pure policy can be visualized as a tree, as is illustrated in Figure 1, which shows a joint policy for the decentralized tiger problem. In this figure, each node marks a point

where an agent has to decide upon an action. A path leading to such a node defines the observation history, and the depth corresponds to the stage at which the decision should be taken. A pure policy assigns to each node one action from \mathcal{A}_i , thereby defining what action the agent should take given some observation history.

Solving a Dec-POMDP amounts to finding an optimal joint policy π^* such that when the agents act accordingly, their expected shared reward is maximal. The quantity that we want the agents to maximize is the expected cumulative reward:

$$\pi^* = \arg \max_{\pi} E_{\pi} \left(\sum_{t=0}^{h-1} R(s, \mathbf{a}) \right). \quad (2.3)$$

Bernstein et al. (7) have shown that optimally solving a Dec-POMDP is NEXP-complete, implying that any optimal algorithm will be doubly exponential in the horizon. This becomes apparent when realizing that the number of pure joint policies is:

$$O \left[\left(|\mathcal{A}_*| \frac{(|\mathcal{O}_*|^h - 1)}{|\mathcal{O}_*| - 1} \right)^n \right], \quad (2.4)$$

where $|\mathcal{A}_*|$ and $|\mathcal{O}_*|$ denote the largest individual action and observation sets.

The naive way of going about is to enumerate each of these joint policies and evaluate their expected cumulative reward, or *value*. The value of a specific (state, joint observation history) pair under a joint policy π is given by:

$$V_{\pi}(s^t, \bar{\mathbf{o}}^t) = R(s^t, \pi(\bar{\mathbf{o}}^t)) + \sum_{s^{t+1}} P(s^{t+1} | s^t, \pi(\bar{\mathbf{o}}^t)) \sum_{\mathbf{o}^{t+1} \in \mathcal{O}} P(\mathbf{o}^{t+1} | s^{t+1}, \pi(\bar{\mathbf{o}}^t)) V_{\pi}(s^{t+1}, \bar{\mathbf{o}}^{t+1}) \quad (2.5)$$

where $\bar{\mathbf{o}}^{t+1}$ is the new joint observation history at stage $t+1$: $\bar{\mathbf{o}}^{t+1} = (\bar{\mathbf{o}}^t, \mathbf{o}^{t+1})$. The total expected reward $V(\pi)$, with respect to the initial state distribution b^0 is then given by

$$V(\pi) = \sum_s V_{\pi}(s, \bar{\mathbf{o}}^0) b^0(s), \quad (2.6)$$

where $\bar{\mathbf{o}}^0$ is the initial (empty) joint observation history. For one joint policy this calculation requires evaluation of (2.5) for each of the $\sum_{t=0}^{h-1} |\mathcal{O}|^t = \frac{|\mathcal{O}|^h - 1}{|\mathcal{O}| - 1}$ joint observation histories and $|S|$ states, leading to a total cost of:

$$O \left(|S| \cdot \frac{|\mathcal{O}|^h - 1}{|\mathcal{O}| - 1} \right). \quad (2.7)$$

3 Cross-entropy optimization

de Boer, Kroese, Mannor, and Rubinstein (12) described the *Cross-Entropy (CE)* method as a general framework to both rare event estimation and combinatorial optimization. We will focus only on the application to optimization. In

particular, it has been illustrated how the CE method can be adapted to find good policies for a particular class of Markov Decision Processes (MDPs) (24; 12). In this section, we first provide a brief introduction to the CE method for optimization, followed by a description of the mentioned application to MDPs.

3.1 General CE Optimization

The cross entropy method can be used for optimization in cases where we want to find a—typically large—vector x from a hypothesis space \mathcal{X} that maximizes some *performance function* $V : \mathcal{X} \rightarrow \mathbb{R}$. That is, when we are looking for

$$x^* = \arg \max_{x \in \mathcal{X}} V(x). \quad (3.1)$$

The CE method associates an estimation problem with this optimization. It maintains a probability distribution f_{ξ} over the hypothesis space, parametrized by a vector ξ . In particular, CE estimates the probability that the performance $V(\mathbf{x})$ of a sample \mathbf{x} drawn according to f_{ξ} is higher than some γ :

$$P_{\xi}(V(\mathbf{x}) \geq \gamma) = \sum_{x \in \mathcal{X}} I(V(x), \gamma) f_{\xi}(x), \quad (3.2)$$

where $I(V(x), \gamma)$ is an indicator function:

$$I(V(x), \gamma) = \begin{cases} 1 & , V(x) \geq \gamma \\ 0 & , V(x) < \gamma. \end{cases} \quad (3.3)$$

Let γ^* be the optimal value, i.e., $\gamma^* \equiv V(x^*)$. Considering $\gamma = \gamma^*$ and a uniform distribution f_{ξ} in (3.2) gives the correspondence to the optimization problem (3.1). Also observe that $P_{\xi}(V(\mathbf{x}) \geq \gamma^*)$ is most likely very small under a uniform f_{ξ} , which explains the link to rare event estimation.

The core of the CE method is an iterative two-phase process:

1. Generate a set of samples \mathbf{X} according to f_{ξ} .
2. Select the best N_b of samples \mathbf{X}_b , and use those to update the parameter vector ξ .⁴

The first step is rather trivial. The second step, however, deserves some explanation. Let $\gamma^{(j)}$ be defined as the minimum performance within the selected set of samples of the j -th iteration. I.e.,

$$\gamma^{(j)} \equiv \min_{\mathbf{x} \in \mathbf{X}_b} V(\mathbf{x}). \quad (3.4)$$

The CE method requires that this lower bound performance is not allowed to decrease over time: $\gamma^{(j+1)} \geq \gamma^{(j)}$. This implies that \mathbf{X}_b can contain less than N_b samples because

$$\forall_{\mathbf{x} \in \mathbf{X}_b} V(\mathbf{x}) \geq \gamma^{(j)} \quad (3.5)$$

⁴The number of best samples is often characterized as a fraction $0 \leq \rho \leq 1$ of the set of samples \mathbf{X} . I.e., $N_b = \text{round}(\rho \cdot N)$.

is a hard requirement. The set \mathbf{X}_b is then used to create $\xi^{(j+1)}$, a maximum-likelihood estimate of the parameters. These new parameters can be smoothed using a learning rate $0 \leq \alpha \leq 1$ by interpolating with $\xi^{(j)}$ the parameter vector of the previous iteration:

$$\xi^{(j+1)} = \alpha \xi^{(j+1)} + (1 - \alpha) \xi^{(j)}. \quad (3.6)$$

This reduces the probability that some components of the parameter vector will be 0 or 1 early in the CE process, which could cause the method to get stuck in local optima.

Usually, the iterative process is stopped when $\gamma^{(j)}$ has not improved over some predefined number of steps. But other conditions such as a time limit or a fixed number of iterations can be used. When the stop condition is finally met, the best sample \mathbf{x} found in the entire process is returned as an approximation of x^* .

3.2 The CE Method for MDPs

In their work, Mannor et al. (24) show how the CE method can be applied to shortest paths MDPs, a class of MDPs for which the optimal value function is stationary, i.e., the expected value of taking a particular action in a particular state is not dependent on the stage. The optimal policy for such a MDP is a mapping from states to actions $\pi_{\text{MDP}} : \mathcal{S} \rightarrow \mathcal{A}$, which can be represented as an $|\mathcal{S}|$ -vector. As in section 3.1, the goal is to find the vector that maximizes a performance function, in this case the expected total reward. So rewriting (3.1), we are looking for

$$\pi_{\text{MDP}}^* = \arg \max_{\pi_{\text{MDP}}} V(\pi_{\text{MDP}}), \quad (3.7)$$

where the performance function now is the value of the MDP-policy π_{MDP} . The CE method tackles this problem by maintaining a parameter vector $\xi = \langle \xi_{s_1}, \dots, \xi_{s_{|S|}} \rangle$, where each ξ_s is a probability distribution over actions. Using these probabilities it is possible to sample N trajectories: starting from some start state actions are randomly selected according to the probabilities as described by ξ until the goal state is reached. Using the N_b best (highest total reward) trajectories \mathbf{X}_b , the parameter vector can be updated as follows:

$$P(a|s) = \frac{\sum_{\mathbf{x} \in \mathbf{X}_b} I(\mathbf{x}, s, a)}{\sum_{\mathbf{x} \in \mathbf{X}_b} I(\mathbf{x}, s)}, \quad (3.8)$$

where $I(\mathbf{x}, s, a)$ is an indicator function that indicates that action a was performed at state s in trajectory \mathbf{x} , and $I(\mathbf{x}, s)$ indicates whether s was visited in trajectory \mathbf{x} .

After updating the parameter vector ξ , a new set \mathbf{X} of trajectories can be sampled, etc. Empirical evaluation shows that this process converges to (near-) optimal policy in only a few iterations (24).

4 Direct CE policy search for Dec-POMDPs

In this section we propose an adaptation of the CE method for Dec-POMDP policy search, which we dub direct CE (DICE) policy search for Dec-POMDPs because it directly searches the space of joint policies without first restricting it.

DICE is a direct translation of the ideas presented in the previous section. Still, there are some problems when trying to apply the procedure as outline in the previous section to Dec-POMDPs: First, because we consider finite horizon Dec-POMDPs, there is no stationary value function. Second, the policies of the agents are not defined over states, but over their individual observation histories \vec{o}_i^t , and these are not a Markovian signal. Third, there is no clear way to sample traces and use those to update the distribution.

In the Dec-POMDP case, the hypothesis space is the space of deterministic joint policies Π . In order to apply the CE method, we are required to define a distribution over this space and an evaluation function for sampled policies. Also, we show how the distribution can be adapted using the best policies found in each iteration. First, we will present two methods to define the distribution over the joint policy space. After that we describe how the parameter updates are performed. Finally, a we give a summary and complexity analysis of the DICE algorithm.

4.1 Policy distributions

In the case of Dec-POMDPs f_ξ denotes a probability distribution over pure joint policies, parametrized by ξ . We will represent this probability as the product of probability distributions over individual pure joint policies:

$$f_\xi(\pi) = \prod_{i=1}^n f_{\xi_i}(\pi_i). \quad (4.1)$$

Here ξ_i is the vector of parameters for agent i , i.e., $\xi = \langle \xi_1, \dots, \xi_n \rangle$.

The question is how to represent the probability distributions over individual pure policies. One clear solution is to enumerate all the pure policies for an agent i and to maintain an explicit discrete probability distribution over this set of policies. I.e., the distribution is represented as a mixed policy (31). However, this approach suffers from two drawbacks. First, the number of pure individual policies π_i might be huge, making such an explicit distribution impractical to represent. Second, this representation is hard to parametrize in a meaningful way using some vector ξ_i , as it gives no access to the internals of the policies: parameters would specify probabilities for entire pure policies, rather than specifying behavior for particular observation histories as in figure 1.

Therefore, rather than using a mixed policy representation, we will use a behavioral- (31) or stochastic policy (22) description: a mapping from decision points to probability

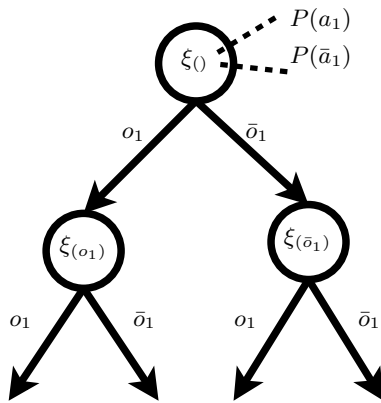


Figure 2: A part of a stochastic policy for an agent of a fictitious Dec-POMDP.

distributions over actions. Note that this is similar to the approach for MDPs, as described in section 3.2, where the states are the decision points.

4.1.1 Observation history based

The simplest way to represent a policy distribution is to make a direct translation from CE for MDPs: instead of maintaining a simple probability distribution over actions for each state, we now maintain one for each observation history (OH). Figure 2 illustrates such a representation of the policy distribution. It shows that for each observation history \vec{o}_i^t a parameter $\xi_{\vec{o}_i^t}$, that specifies the distribution over actions, is maintained:

$$\forall_{a_i} \xi_{\vec{o}_i^t}(a_i) \equiv P(a_i|\vec{o}_i^t). \quad (4.2)$$

Note that the distribution differs only from a pure policy (Figure 1) by keeping distributions over actions instead of a single action at the nodes of the tree. Consequently, the parameter vector for agent i is defined as $\xi_i \equiv \langle \xi_{\vec{o}_i^t} \rangle_{\vec{o}_i^t \in \vec{\mathcal{O}}_i}$, and the probability of a particular policy π_i for agent i is as

$$f_{\xi_i}(\pi_i) = \prod_{\vec{o}_i^t \in \vec{\mathcal{O}}_i} \xi_{\vec{o}_i^t}(\pi_i(\vec{o}_i^t)). \quad (4.3)$$

We refer to this policy distribution representation as the OH-based representation.

4.1.2 Action-observation history based

Defining the parameters as in section 4.1.1 is the most straightforward approach, but does not take into account the action history: the choice for action $\pi_i(\vec{o}_i^t)$ has no influence on the choice for the action at the next time step $\pi_i(\vec{o}_i^{t+1})$. As explained in section 2, in general a stochastic policy does take into account the taken action. However, we know that there is at least one deterministic joint policy for a Dec-POMDP (which consists of individual policies that are a mapping from observation histories to actions). Moreover, in previous research we did investigate

an action-observation history (AOH) based representation, but the influence appeared to be minor (27). Therefore we will not consider this further in this work.

4.2 Sampling and Evaluation

Unlike the MDP case, in the setting of Dec-POMDPs there is no trivial way to sample some trajectories given the joint policy distribution f_{ξ} and use that to update the distribution. Rather we propose to sample complete joint policies and use those for the parameter update.

Selecting a random sample of joint policies from the distribution is straightforward. For all the observation histories \vec{o}_i^t of an agent i an action can be sampled from action distribution $\xi_{\vec{o}_i^t}$. The result of this process is a deterministic policy for agent i . Repeating this procedure for each agent samples a deterministic joint policy. The evaluation of a joint policy can be done using (2.6).

4.3 Parameter update

We described how to represent and sample from the probability distribution over policies. This section describes how the set of best policies \mathbf{X}_b sampled from the previous distribution $f_{\xi^{(j)}}$, can be used to find new parameters $\xi^{(j+1)}$.

Let $I(\pi_i, \vec{o}_i^t, a)$ be an indicator function that indicates whether $\pi_i(\vec{o}_i^t) = a$. In the OH-based distribution the probability of agent i taking action $a^t \in \mathcal{A}_i$ after having observed \vec{o}_i^t can be re-estimated as:

$$\xi_{\vec{o}_i^t}^{(j+1)}(a^t) = \frac{1}{|\mathbf{X}_b|} \sum_{\pi \in \mathbf{X}_b} I(\pi_i, \vec{o}_i^t, a^t), \quad (4.4)$$

where $|\mathbf{X}_b|$ normalizes the distribution since

$$\forall_{\vec{o}_i^t} \sum_{a \in \mathcal{A}_i} \sum_{\pi \in \mathbf{X}_b} I(\pi_i, \vec{o}_i^t, a) = |\mathbf{X}_b|. \quad (4.5)$$

Note that the thus computed new parameter vector $\xi^{(j+1)}$ will afterward be smoothed using the learning rate α according to (3.6).

4.4 Summary and complexity analysis

Algorithm 1 summarizes the DICE policy search method. To start it needs I , the number of iterations, N , the number of samples taken at each iteration, N_b , the number of samples used to update ξ , and α , the learning rate. The outer loop of lines 3–17 covers one iteration. The inner loop of lines 5–13 covers sampling and evaluating one joint policy. Lines 14–16 perform the parameter update. Because the CE method can get stuck in local optima, one typically performs a number of restarts. We have not incorporated these in the algorithm itself, however.

Now we consider the complexity of this algorithm. For each iteration we draw N joint policies. The sampling of

Algorithm 1 The DICE policy search algorithm

Require: CE parameters: I, N, N_b, α

- 1: $V_b \leftarrow -\infty$
 - 2: initialize $\xi^{(0)}$ {typically uniform random}
 - 3: **for** $i \leftarrow 0$ to I **do**
 - 4: $\mathbf{X} \leftarrow \emptyset$
 - 5: **for** $s \leftarrow 0$ to N **do**
 - 6: sample π from $f_{\xi^{(i)}}$
 - 7: $\mathbf{X} \leftarrow \mathbf{X} \cup \{\pi\}$
 - 8: $V(\pi) \leftarrow \text{Evaluate}(\pi)$
 - 9: **if** $V(\pi) > V_b$ **then**
 - 10: $V_b \leftarrow V(\pi)$
 - 11: $\pi_b \leftarrow \pi$
 - 12: **end if**
 - 13: **end for**
 - 14: $\mathbf{X}_b \leftarrow$ the set of N_b best $\pi \in \mathbf{X}$
 - 15: Compute $\xi^{(i+1)}$ {using (4.4) }
 - 16: $\xi^{(i+1)} \leftarrow \alpha \xi^{(i+1)} + (1 - \alpha) \xi^{(i)}$
 - 17: **end for**
 - 18: **return** π_b
-

such a joint policy involves, for each agent, selecting an action for each of its observation histories and has complexity

$$O(n \cdot |\mathcal{A}_*| \cdot |\vec{\mathcal{O}}_*|) = O\left(n \cdot |\mathcal{A}_*| \cdot \frac{|\mathcal{O}_*|^h - 1}{|\mathcal{O}_*| - 1}\right),$$

where $*$ denotes the agent index with the largest observation and action set respectively. The complexity of updating ξ is similar, but includes the term N_b (rather than being performed N times). Evaluation (i.e., computing the total expected reward) of a policy $V(\pi)$, is performed by evaluating equation (2.6) and (2.5) from the last stage $h - 1$ up to the first 0. The complexity of this calculation for a single pure joint policy scales exponentially with the planning horizon, as explained in section 2. Because $|\mathcal{O}| = O(|\mathcal{O}_*|^n)$, we can rewrite (2.7) as

$$O\left(|\mathcal{S}| \cdot \frac{|\mathcal{O}_*|^{nh} - 1}{|\mathcal{O}_*|^n - 1}\right),$$

This latter term typically dominates the complexity of sampling and updating ξ , therefore the total time complexity of DICE is given by

$$O\left[I \cdot N \cdot \left(|\mathcal{S}| \frac{|\mathcal{O}_*|^{nh} - 1}{|\mathcal{O}_*|^n - 1}\right)\right].$$

5 Approximate Evaluation

The complexity analysis showed that the time required for DICE scales exponentially with the planning horizon, and that policy evaluation forms the bottleneck of our algorithm. In order to alleviate this bottleneck, we examine the application of approximate evaluation. The idea is that rather than computing the expected value, we sample this

value by simulating r episodes, or *traces*, and using the average of outcomes as an estimate $\tilde{V}(\pi)$ for the actual value $V(\pi)$. We will refer to the resulting method as DICE policy search with approximate evaluation (DICE-A).

Clearly, this approximation might introduce errors. Notice, however, that the CE method does not discriminate between policies within the set \mathbf{X}_b of best samples. Therefore, as long as the relative ordering is preserved, the same policies are used to update the policy distribution, yielding the same results. In fact, only when the ranking of policies is disturbed near the cut-off threshold (around the N_b -th joint policy), will approximate evaluation influence the distribution updating process.

There is a second potential source of error, though. When the fraction of best samples \mathbf{X}_b is used to update γ using (3.4), the new γ might in fact be an over-estimation. This could make it very difficult to sample new instances with a higher (approximate) value. In previous work, we therefore also considered a version of our algorithm that did not use the hard threshold γ , but rather always used the best N_b samples (27). The results, however, did not show a significant improvement, nor did we encounter any such difference in further experiments we performed. Therefore we will not consider this further in this paper.

5.1 Complexity

Simulating one trace of a joint policy involves looking up the actions for each of the n agents and sampling one of $|\mathcal{S}|$ successor states and one of $|\mathcal{O}| = O(|\mathcal{O}_*|^n)$ joint observations at each of the h stages. Such a simulation is performed r times, so the time complexity of performing approximate evaluation of a single joint policy is:

$$O(r \cdot h \cdot n \cdot |\mathcal{O}_*|^n \cdot |\mathcal{S}|). \tag{5.1}$$

DICE-A performs such an evaluation for each of the N sampled policies in each of the I iterations. Therefore, total time complexity of DICE with approximate evaluation is given by

$$O(I \cdot N \cdot r \cdot h \cdot n \cdot |\mathcal{O}_*|^n \cdot \mathcal{S}), \tag{5.2}$$

as long as approximate evaluation dominates the time needed to sample a policy and update the parameter vector ξ .

5.2 Error bounds

The estimated value $\tilde{V}(\pi)$ is only an approximation of the true value $V(\pi)$. However, we are able to establish bounds on this error. In particular, we know that $V(\pi)$ is bounded when the immediate reward function is bounded. Let us write R_{min}, R_{max} for the lower and upper bound of the reward function, that is, $\forall_{s, \mathbf{a}} R(s, \mathbf{a}) \in [R_{min}, R_{max}]$. Then the value of a policy is bounded by

$$V(\pi) \in [hR_{min}, hR_{max}].$$

Wassily Hoeffding (18) proved that the probability that the sum of r independent random variables X_1, \dots, X_r , each bounded $X_i \in [a_i, b_i]$, exceeds the expectation (of the sum) with $r\epsilon$ or more is bounded by

$$P((X_1 + \dots + X_r) - E[X_1 + \dots + X_r] \geq r\epsilon) \leq \exp\left(-\frac{2r^2\epsilon^2}{\sum_{i=1}^r (b_i - a_i)^2}\right),$$

for any $\epsilon > 0$.

In our setting, each X_i denotes the outcome of the simulation of the i -th episode, and is an unbiased estimate of $V(\pi)$. Also, in our setting we are interested in a two-sided error bound, and all X_i respect the same bound $X_i \in [hR_{min}, hR_{max}]$. Therefore we can write

$$\begin{aligned} &P(|(X_1 + \dots + X_r) - E[X_1 + \dots + X_r]| \geq r\epsilon) \\ = &P\left(\left|\frac{(X_1 + \dots + X_r)}{r} - \frac{E[X_1 + \dots + X_r]}{r}\right| \geq \epsilon\right) \\ = &P\left(\left|\frac{1}{r} \sum_{i=1}^r X_i - E(X)\right| \geq \epsilon\right) \\ = &P\left(\left|\tilde{V}(\pi) - V(\pi)\right| \geq \epsilon\right) \\ \leq &2 \exp\left(-\frac{2r^2\epsilon^2}{r(hR_{max} - hR_{min})^2}\right). \end{aligned}$$

Using this result we can control the lower bound on the probability that an error of size less than ϵ is made. Suppose we want to approximate $V(\pi)$ with accuracy ϵ with at least probability δ . I.e., $\delta = \lfloor P(\text{error} < \epsilon) \rfloor$ is the lower bound on the probability of an error smaller than ϵ , yielding

$$\begin{aligned} P(\text{error} \geq \epsilon) &= 1 - P(\text{error} < \epsilon) \\ P(\text{error} \geq \epsilon) &\leq 1 - \lfloor P(\text{error} < \epsilon) \rfloor \\ &= 1 - \delta. \end{aligned}$$

Then we must have that

$$\begin{aligned} &P\left(\left|\frac{1}{r} \sum_{i=1}^r X_i - E(X)\right| \geq \epsilon\right) \leq \\ &2 \exp\left(-\frac{2r^2\epsilon^2}{r(hR_{max} - hR_{min})^2}\right) \leq 1 - \delta. \end{aligned}$$

Solving the right-hand side for r goes as follows:

$$\begin{aligned} -\frac{2r^2\epsilon^2}{r(b-a)^2} &\leq \ln\left(\frac{1-\delta}{2}\right) \\ 2r\epsilon^2 &\geq -(b-a)^2 \ln\left(\frac{1-\delta}{2}\right) \\ r &\geq \frac{(b-a)^2}{2\epsilon^2} \ln \frac{2}{1-\delta}. \end{aligned}$$

with

$$(b-a)^2 = h^2 (R_{max} - R_{min})^2.$$

So, to guarantee an error smaller than ϵ (with probability δ), the required number of traces grows only quadratically with the horizon.

5.3 Post-evaluation

When using DICE with approximate evaluation, the end result is a joint policy and its estimated value. In order to know the true quality of the joint policy, an exact evaluation can be started at this point. However, due to the exponential complexity of such an exact evaluation, this is not always feasible. In settings where it is not, we propose to do a more accurate sample based estimation of the value of the joint policy.

Of course, it may happen that the new exact (or more accurately determined) value of the joint policy is less than the previous estimate, and perhaps also less than the estimated value of other joint policies encountered during DICE. To prevent this, one may keep a list of the best k joint policies encountered. At the end of the procedure, one can then exactly evaluate all these k joint policies and select the best one. Alternatively, the CE process can be augmented with one additional iteration, where all sampled policies are evaluated exactly (or more accurately).

6 Experiments

Here we give an empirical evaluation of the proposed algorithm. The implementation of DICE follows the description without any further remarks. In our DICE-A implementation we have not implemented an additional evaluation iteration or list of k best policies as suggested. We only apply post-evaluation to the best ranked joint policy. This post-evaluation consists of a more accurate evaluation of 20,000 runs when the value function consists of more than 20,000 (s, \vec{o}) -pairs, and exact evaluation otherwise.

First we examine the influence of all parameters of the CE optimization procedure. Next, we briefly discuss some benchmark problems and investigate the performance on these benchmark problems of different horizons and compare it to dynamic programming JESP. Finally, we investigate how DICE scales with respect to the number of agents, again comparing to JESP.

6.1 Different CE parameters

The CE method has quite a few configurable parameters: the learning rate α , the number of iterations I , the number of samples drawn per iteration N , the fraction of best samples kept for update ρ and the induced number of samples used for this update $N_b = \rho \cdot N$. We have empirically investigated different settings of these parameters for DICE policy search. Additionally, for DICE-A we investigate the parameter r for the number approximation simulations.

The results reported in this section are all obtained on the DEC-TIGER problem. The non-varying parameters in these experiments were set as follows $\alpha = 0.2$, $I = 30$, $N = 50$, $N_b = 5$.

First we examine the influence of the learning rate α . Figure 3 shows that low α (0.2) results in low variance, but if too low ($\alpha = 0.1$) the CE process cannot converge

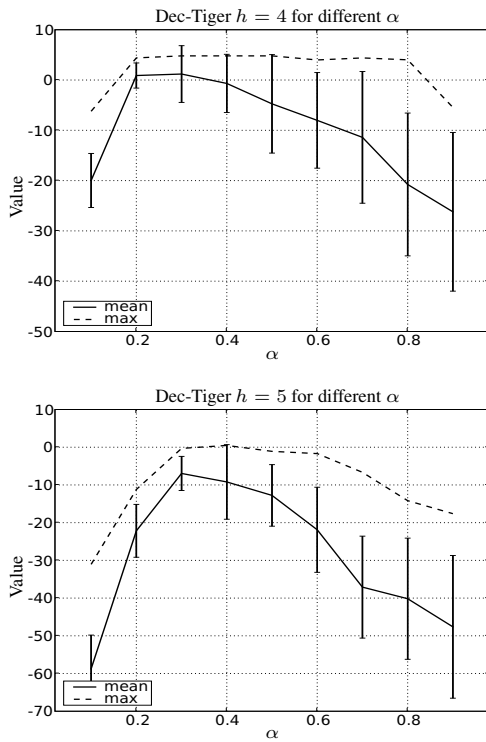


Figure 3: Performance for varying α . Top: horizon 4. Bottom: horizon 5.

to a good solution within a limited number of iterations. Also shown is that DICE on DEC-TIGER with 30 iterations $\alpha = 0.3 - 0.4$ gives the best trade-off between learning and convergence.

Next, we examine the number of CE iterations. The complexity analysis shows that this parameter should affect the running time linearly and this was experimentally confirmed. Figure 4 shows the value obtained by the algorithm for different numbers of iterations. It indicates that the algorithm converges after a fixed number of iterations. As we might expect, convergence requires less iterations for a smaller horizon (compare top and bottom figure). However, in both cases we see that the performance grows very rapidly when first increasing the number of iterations and then levels out. This indicates that even with a limited number of iterations, CE might be able to obtain fairly good results fast.

Figure 5 shows the values obtained by varying N , the number of joint policies sampled per iteration, which also increases the complexity linearly. The results were obtained using a fixed update fraction $\rho = 0.1$. Here too, we see that improvement is strong initially, and then flattens out later. Also note that the higher variance for $h = 5$ can be explained by looking at the performance for $I = 30$ in Figure 4.

The number of samples used to update the parameter vector ξ only marginally influences the run-time, and we were not able to determine this empirically. Also, using a larger fraction ρ decreases the performance. In this case,

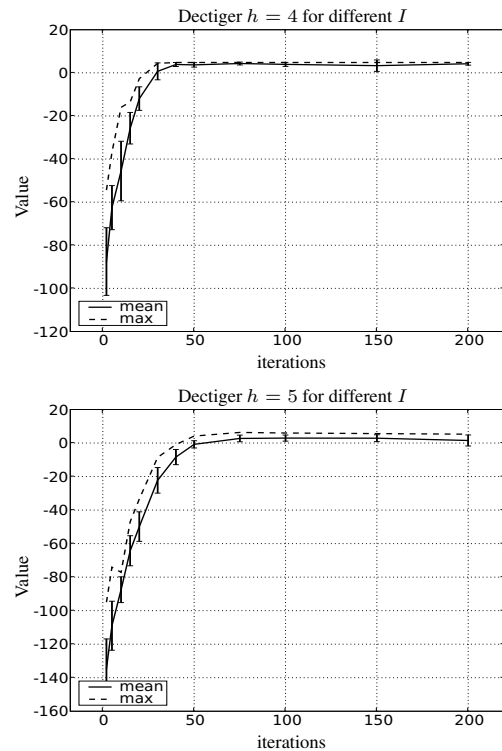


Figure 4: Performance for varying number of CE iterations. Top: horizon 4. Bottom: horizon 5.

the quality of the sampled joint policies used to re-estimate the distribution parameters degenerates and CE will not converge towards good solutions. The results in figure 6 indicate optimal performance for ρ between 0.05 and 0.1. We omitted the nearly identical graph for $h = 5$.

In case DICE-A is used for policy search, the number of approximation runs influences the run time linearly. In figure 7 we can clearly see that the quality of the approximate evaluation converges quickly for horizon 4. For horizon 5 the size of the policies increases exponentially and more runs are needed to maintain approximation quality, but also here we see no improvement beyond $r = 1000$.

6.2 Results on different problems

Here we report the results of the performance of DICE on different Dec-POMDP problems and different horizon lengths. In particular we consider the following problems: BROADCAST CHANNEL, DEC-TIGER, MEETING ON A GRID and FACTORED FIREFIGHTING. The DEC-TIGER problem was discussed section 2.1. For the other problems we now provide a brief description.

The BROADCAST CHANNEL problem involves two agents that control a broadcast channel. Each agent decides at every stage whether or not to send a message across it. When both agents send a message at the same time a collision occurs. When a message is successfully transmitted, the agents get a reward of +1. More information can be found in (30; 17).

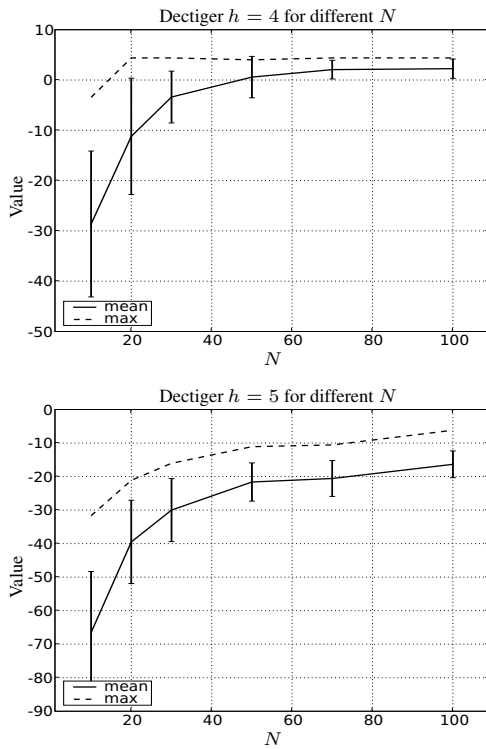


Figure 5: Performance under varying number of samples per CE iteration. Top: horizon 4. Bottom: horizon 5.

MEETING ON A GRID was introduced by Bernstein et al. (8) and considers two robots on a 2x2 grid. Their goal is to occupy the same square which gives them a +1 reward, but they have imperfect sensors and actuators which complicates the task. We consider the version with 2 observations per agent (2).

Finally, FACTORED FIREFIGHTING is a problem with 3 agents that have to fight fires at 4 houses (29). Each agent has 2 different houses it can go to (the sets are overlapping, but not equal). Every stage each agent should choose which of its 2 associated houses it wants to fight fire at. For each house that is burning, the agents receive a penalty of -1 or -2, depending on the level of fire.

Before showing results of the proposed CE approaches,

	Dec-Tiger	Broadcast	Grid	FFF
n	2	2	2	3
$ S $	2	4	16	81
$ A_i $	3	2	5	2
$ O_i $	2	2	2	2
2	7.290e02	6.400e01	1.563e04	5.120e02
3	4.783e06	1.638e04	6.104e09	2.097e06
4	2.059e14	1.074e09	9.313e20	3.518e13
5	3.815e29	4.612e18	2.168e43	9.904e27
6	1.310e60	8.507e37	1.175e88	7.846e56
7	1.545e121	2.895e76	3.454e177	4.925e114
8	2.147e243	3.352e153	Inf	1.941e230

Table 1: The number of joint policies for different problems and horizons. Inf denotes a value beyond double machine precision.

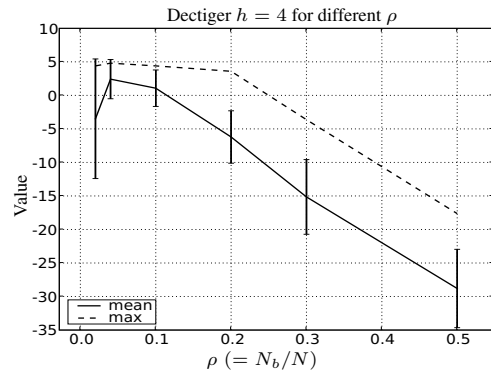


Figure 6: The fraction of samples ρ used to update the distribution f_ξ on horizon 4.

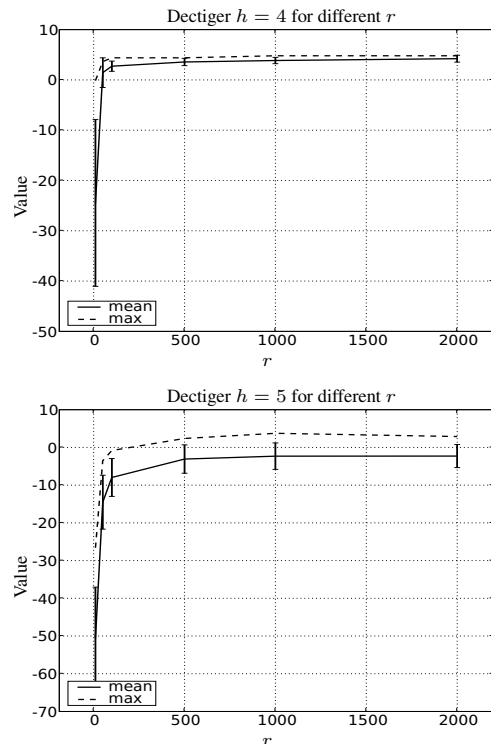


Figure 7: Performance for varying number of approximate evaluation runs. Top: horizon 4. Bottom: horizon 5.

we first report the size of the joint policy space for different considered problems in Table 1. Clearly we are dealing with huge search spaces here. In fact for $h = 8$ MEETING ON A GRID the number of joint policies was not representable by a double precision float (the maximum representable being $1.7977e+308$). We emphasize that DICE directly searches these spaces, without first restricting them.

We report the results of DICE and DICE-A on these different test problems. We also compare against dynamic programming JESP which was proposed by Nair et al. (25). This method starts with a random joint policy and then iterates over the agents, computing a best-response policy for the selected agent while keeping the other agents fixed. The term “dynamic programming” indicates that this best-response is computed by solving an augmented POMDP for the selected agent. The reason to compare against JESP is that, as mentioned in the introduction, it is the only other approximate method that does not constrain the search space in any way.

The settings for DICE used in these experiments are: a learning rate of $\alpha = 0.2$, $N = 50$ joint policies per iteration, using the $N_b = 5$ best for update (i.e. $\rho = 0.1$). For DICE-A we used the same settings and the approximation was based on 1000 simulations. The results in this section are averaged over 100 random initializations, or restarts, of each solution method. However, due to the time needed for higher horizons, we have not always restarted 100 times for the highest horizon considered. The numerical results and the number of restarts over which they are obtained are listed in the appendix. The reported timing results are cpu-user times and obtained on an Intel Xeon 3.4 GHz machine with 2GB memory running Debian linux.

The results for the DEC-TIGER problem are shown in Figure 8. Note that the run-time results in the right figure use a log scale. Our evaluation shows that for low horizons ($h < 6$) DICE outperforms JESP but has taken, under these settings, more time to finish. However, the run-time of JESP increases much faster for larger horizons: for horizon 8 it is about ten times slower than DICE. The run-time of DICE-A is affected least by the value of the horizon.

In terms of quality of the found policy, DICE outperforms JESP for lower horizons: although all methods found (near-)optimal solutions for $h = 2, 3, 4$ within the 100 restarts, the variance of JESP is much higher. Unfortunately, the joint policies found by DICE for $h > 5$ are not very good, as the performance drops below JESP. However, this exposes the potential of approximate evaluation: which allows for much more iterations in less time than regular DICE. We also ran DICE-A with 200 iterations (20 restarts). While using approximation with the same settings does not seem to significantly change the results of the CE method by itself (DICE and DICE-A with 50 iterations perform roughly equal), it does allow for more iterations, leading to good overall results while keeping the run-time acceptable. Only for $h = 8$, JESP really seems to achieve a better result, but with high variance, and high run-times.

For BROADCAST CHANNEL the results are shown in

Figure 9. It shows that CE achieves a higher mean value and less variance with again little difference between DICE and DICE-A. The run-time of DICE-A on the other hand increases again much slower than the other two approaches, which eventually is more beneficial.

As indicated by Table 1, the MEETING ON A GRID problem is somewhat larger than the previous problems. This quickly makes exact evaluation problematic. For example, to compute the value of a horizon 6 joint policy $2.18e4 (s, \vec{o})$ -pairs have to be evaluated. Figure 10 shows that while JESP requires much more time than DICE, it does not result in better performance. The rightmost plot shows that the run-time of DICE-A is significantly lower from horizon 5 on. However, starting at $h = 6$ DICE-A seems to get trapped in a local optimum. Still, it is the only method to compute a policy for horizons 7 and 8.

The last problem, FACTORED FIREFIGHTING, is even larger. Because there are now 3 agents, the number of joint observation histories, and thus the number of entries to compute for exact evaluation and for JESP grows much faster. This is reflected by the results as the graphs in Figure 11 show. DICE and JESP can only find solutions for at maximum $h = 3$ and $h = 4$ respectively. Again this demonstrates the power of DICE-A, which does not encounter any problems computing results up to $h = 8$.

In general DICE and DICE-A seem to perform quite well in comparison to JESP. Although JESP’s maximum value is usually equal or greater than the maximum value found by the DICE methods, its mean value is lower and the standard deviation is high. This indicates the need for many restarts in order to find a good solution and performing a lot of restarts becomes problematic for higher horizons, because of the exponential increase in run-time. The results of DICE, however, have a much lower standard deviation, indicating that less restarts are necessary. It still shares the burden of exponentially increasing run-times, though. DICE-A has proven to be a very powerful method. Even though the standard deviations are somewhat greater than regular DICE, it is able to trade off run-time and accuracy and thus achieves reasonable results even for higher horizons, when the other methods fail.

6.3 Varying the number of agents

In the previous section we investigated the ability of DICE to scale with respect to the horizon. Here we investigate the scaling behavior with respect to the number of agents.

So far, almost all available Dec-POMDP problems involve only two agents. Two exceptions are the FIREFIGHTING problem introduced by Oliehoek et al. (28), and the FACTORED FIREFIGHTING already mentioned. Here we will use the former (non-factored) version, since this allows us to vary the number of agents, while keeping the number of houses (and thus the number of states) constant.

Again we examined the performance of DICE, DICE-A and JESP for this problem, now varying the number of agents. We did this at two chosen horizons $h = 3$ and

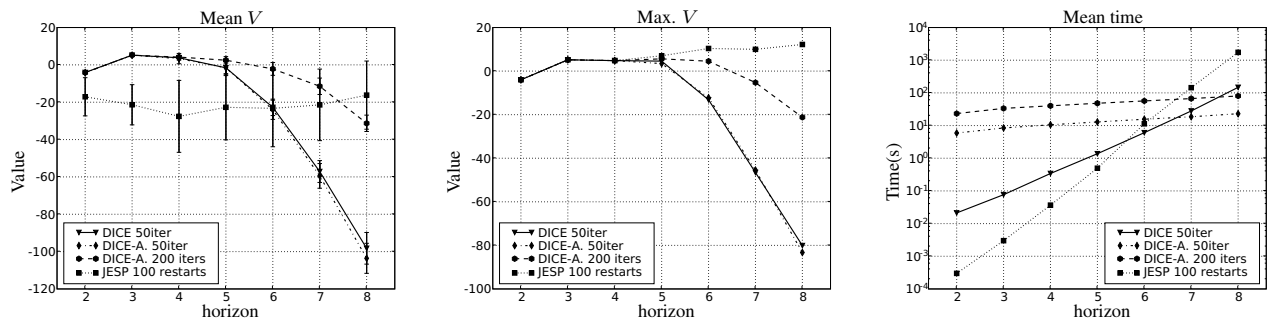


Figure 8: The DEC-TIGER problem. Left: average value. Middle: maximum value. Right: average run-time.

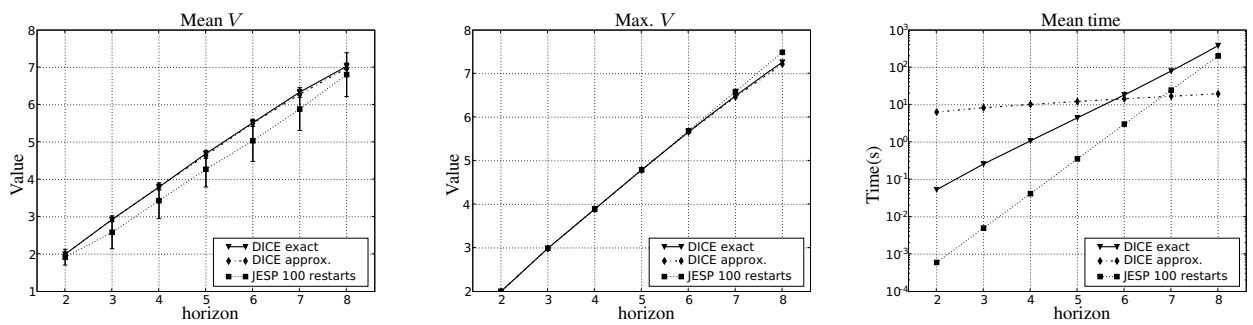


Figure 9: The BROADCAST CHANNEL problem. Left: average value. Middle: maximum value. Right: average run-time.

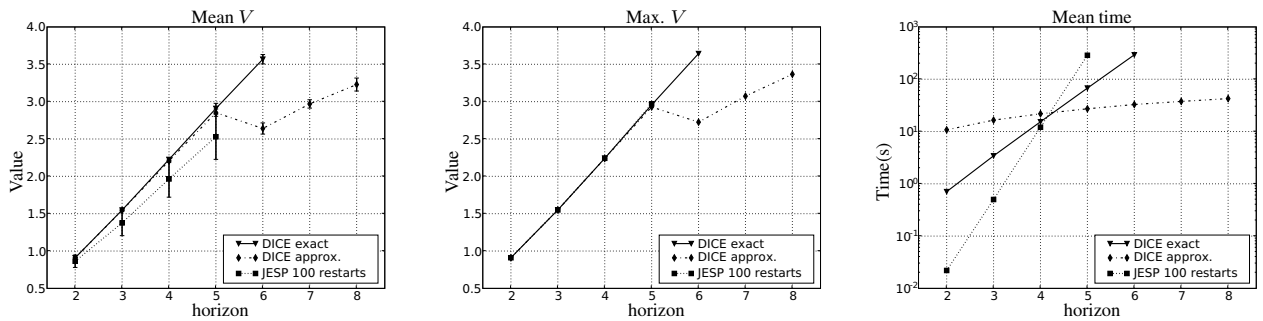


Figure 10: The MEETING ON A GRID problem. Left: average value. Middle: maximum value. Right: average time.

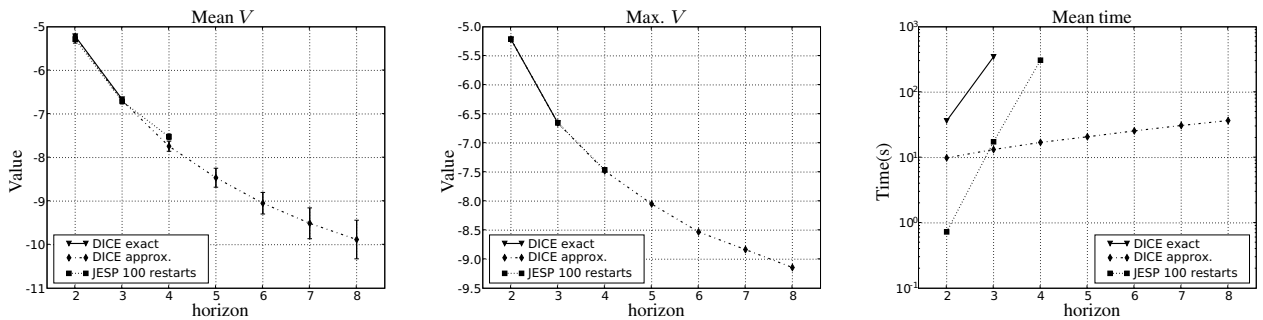


Figure 11: The FACTORED FIREFIGHTING problem with several restarts. Left: average value. Middle: maximum value. Right: average run-time.

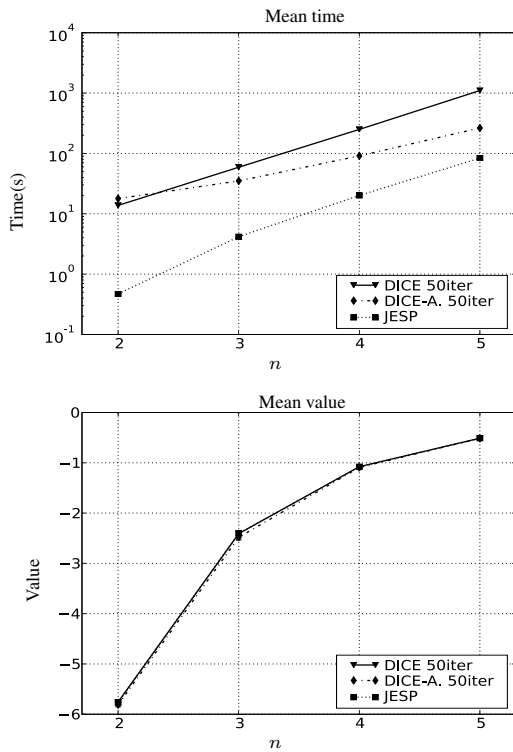


Figure 12: Performance for varying number of agents for $h = 3$ of the FIREFIGHTING problem.

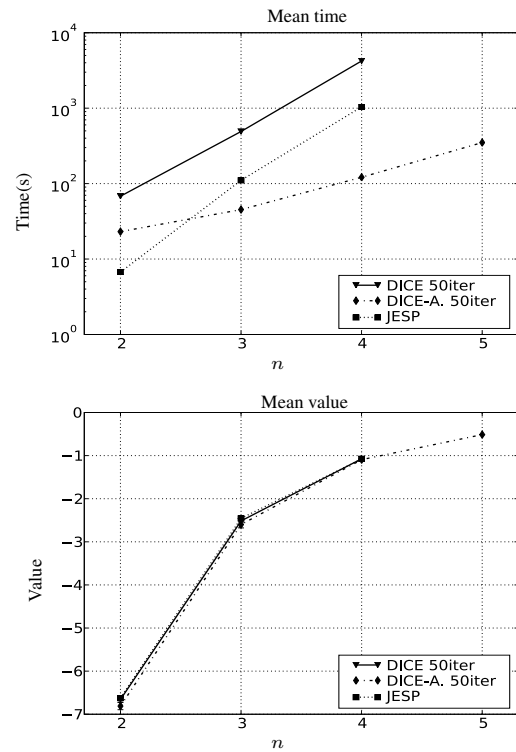


Figure 13: Performance for varying number of agents for $h = 4$ of the FIREFIGHTING problem.

$h = 4$, the results for which are shown in respectively Figure 12 and Figure 13. We used the same settings for the DICE algorithms as before. The number of restarts for all methods was set to 20.

The figures show that, for this problem, all methods performed very well in terms of achieved value. The maximum found value of all policies coincided (therefore these plots are omitted), which may indicate that these are true global optima. More interesting are the run time results. For $h = 3$, we see that JESP outperforms the DICE algorithm for all number of agents. However, its increase of run time when increasing the number of agents is higher than for DICE and particularly DICE-A. This is emphasized by the results for $h = 4$, that clearly show that run time for JESP, but also for exact DICE, grow so fast that they are unable to compute results for 5 agents within reasonable time.⁵ The run times of the two different settings of DICE-A, however, grow much slower and as a consequence these methods are able to find a good solution for 5 agents. Note that in accordance with (5.2), the run time still increases exponentially in the number of agents, simply because simulating an episode (sampling joint observations) takes time exponential in the number of agents. In problems that exhibit observation independence (such as FIREFIGHTING), it is possible to work around this. We have not considered this efficiency increasing measure further, but stress

⁵Each algorithm was given 8 hours to compute all results for a given horizon.

that the efficiency of DICE-A could be further improved for such problems.

7 Discussion and future work

This article has focused on decentralized decision making formalized in the Dec-POMDP framework. We argued that planning in such settings in principle is a complex combinatorial decision process. We demonstrated how to apply the CE-method, a recently introduced method for combinatorial optimization, for policy search in Dec-POMDPs.

We detailed the resulting algorithm, direct CE (DICE) policy search for Dec-POMDPs, and performed a complexity analysis, which identified the exact evaluation of sampled joint policies as a bottleneck. Consequently we proposed DICE-A which performs an approximate evaluation, and showed that, under some assumption, its time complexity is polynomial in the CE parameters. We also presented a formalization of the error bounds for this approximate evaluation.

We presented an empirical evaluation of the influence of the different CE parameters on policy search and also tested performance on different test problems from literature, over different horizons. In these latter experiments we compared against JESP, which, to our knowledge, is the only other approximate planning method for Dec-POMDPs that does not restrict the search space in any way. The results of this comparison were generally favorable for CE.

In particular, a nice feature of CE is that by adjusting the parameters, one is able to control the run-time. On the other hand, because JESP has no parameters, it is somewhat more easy to apply. In a final comparison we investigated how well the mentioned algorithms scale with respect to the number of agents. Although still exponential, DICE-A outperforms the other methods for larger problems.

However, this work does not intend to present a new state-of-the-art Dec-POMDP solver: we compare against JESP, which is one of the older Dec-POMDP algorithms, and more advanced methods have since been proposed. Rather our work shows that viewing these problems as combinatorial optimization problems and applying corresponding methods (such as CE optimization) can bring leverage to the planning process.

An interesting direction for future work is the application of the CE method (and potentially other methods for combinatorial optimization) in more recent algorithms. For instance, a Dec-POMDP can be approximated by solving for each stage t a pruned Bayesian game (BG) (13) or a compressed BG (14). The optimal solution of such BGs, however, involves enumeration of all the corresponding joint policies. CE might prove to be very effective to find approximate solutions for the BGs fast. In particular, it might turn out that the pruning/compression is no longer necessary for the same horizon when applying CE, and that when combining pruning/compression and CE, the algorithm can scale to higher h .

Another example is the method proposed by Seuken and Zilberstein (34). This method is an extension of dynamic programming for Dec-POMDPs that fixes the maximum number of policy trees that are retained in each iteration to a parameter *maxTrees*. The authors report that:

“Even for a small problem 2 actions and 5 observations, setting *maxTrees*=5 would be prohibitive because $(2 \cdot 5^2)^2 = 39,062,500$ policy tree pairs would have to be evaluated.”

It might be possible to apply CE to this smaller policy search problem that occurs in each iteration of the DP process. This could lead to improved efficiency, or the space could be less restricted in order to find a better approximate solution.

Other directions for future research would involve improving the efficiency of the CE method itself. One idea for this would be to use crude value approximation in the first iterations to quickly increase the probabilities of promising policies. In the course of the process, evaluation can be performed more accurately. Exact evaluation can most likely be accelerated by caching (intermediate) evaluation results of (parts of) joint policies. Also, the joint policies resulting from CE search might be improved by using those as a starting point for JESP, leading to a hybrid optimization scheme for multiagent settings.

Finally, and somewhat related, the success of approximate evaluation raises the question whether it is necessary to sample complete joint policies if they are only par-

tially inspected during approximate evaluation. The CE approach could benefit from a construction that samples parts of (joint) policies.

Acknowledgments

We would like to thank Matthijs T.J. Spaan for his help and advice. The research reported here is part of the Interactive Collaborative Information Systems (ICIS) project, supported by the Dutch Ministry of Economic Affairs, grant nr: BSIK03024.

References

- [1] G. Alon, D. Kroese, T. Raviv, and R. Rubinstein. Application of the cross-entropy method to the buffer allocation problem in a simulation-based environment. *Annals of Operations Research*, 134(1):137–151, 2005.
- [2] C. Amato, D. S. Bernstein, and S. Zilberstein. Optimal fixed-size controllers for decentralized POMDPs. In *Proc. of the AAMAS Workshop on Multi-Agent Sequential Decision Making in Uncertain Domains (MSDM)*, May 2006.
- [3] C. Amato, A. Carlin, and S. Zilberstein. Bounded dynamic programming for decentralized POMDPs. In *Proc. of the AAMAS Workshop on Multi-Agent Sequential Decision Making in Uncertain Domains (MSDM)*, May 2007.
- [4] R. Aras, A. Dutech, and F. Charpillet. Mixed integer linear programming for exact finite-horizon planning in decentralized POMDPs. In *The International Conference on Automated Planning and Scheduling*, 2007.
- [5] R. Becker, S. Zilberstein, V. Lesser, and C. V. Goldman. Solving transition independent decentralized Markov decision processes. *Journal of Artificial Intelligence Research (JAIR)*, 22:423–455, December 2004.
- [6] D. S. Bernstein, S. Zilberstein, and N. Immerman. The complexity of decentralized control of Markov decision processes. In *Proc. of Uncertainty in Artificial Intelligence*, pages 32–37, 2000.
- [7] D. S. Bernstein, R. Givan, N. Immerman, and S. Zilberstein. The complexity of decentralized control of Markov decision processes. *Math. Oper. Res.*, 27(4): 819–840, 2002.
- [8] D. S. Bernstein, E. A. Hansen, and S. Zilberstein. Bounded policy iteration for decentralized POMDPs. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI)*, 2005.

- [9] Z. Botev and D. P. Kroese. Global likelihood optimization via the cross-entropy method with an application to mixture models. In *WSC '04: Proceedings of the 36th conference on Winter simulation*, pages 529–535, 2004.
- [10] C. Boutilier. Planning, learning and coordination in multiagent decision processes. In *TARK '96: Proceedings of the 6th conference on Theoretical aspects of rationality and knowledge*, pages 195–210, San Francisco, CA, USA, 1996. Morgan Kaufmann Publishers Inc. ISBN 1-55860-417-9.
- [11] I. Cohen, B. Golany, and A. Shtub. Managing stochastic finite capacity multi-project systems through the cross-entropy method. *Annals of Operations Research*, 134(1):183–199, 2005.
- [12] P.-T. de Boer, D. P. Kroese, S. Mannor, and R. Y. Rubinstein. A tutorial on the cross-entropy method. *Annals of Operations Research*, 134(1):19–67, 2005.
- [13] R. Emery-Montemerlo, G. Gordon, J. Schneider, and S. Thrun. Approximate solutions for partially observable stochastic games with common payoffs. In *Proc. of Int. Joint Conference on Autonomous Agents and Multi Agent Systems*, pages 136–143, 2004.
- [14] R. Emery-Montemerlo, G. Gordon, J. Schneider, and S. Thrun. Game theoretic control for robot teams. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1175–1181, 2005.
- [15] C. V. Goldman and S. Zilberstein. Decentralized control of cooperative systems: Categorization and complexity analysis. *Journal of Artificial Intelligence Research (JAIR)*, 22:143–174, 2004.
- [16] C. Guestrin, D. Koller, and R. Parr. Multiagent planning with factored MDPs. In *Advances in Neural Information Processing Systems 14*, pages 1523–1530, 2002.
- [17] E. A. Hansen, D. S. Bernstein, and S. Zilberstein. Dynamic programming for partially observable stochastic games. In *Proc. of the National Conference on Artificial Intelligence*, pages 709–715, 2004.
- [18] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, Mar. 1963.
- [19] 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.
- [20] Y. Kim, R. Nair, P. Varakantham, M. Tambe, and M. Yokoo. Exploiting locality of interaction in networked distributed POMDPs. In *Proceedings of the of the AAAI Spring Symposium on Distributed Plan and Schedule Management*, 2006.
- [21] J. R. Kok and N. Vlassis. Using the max-plus algorithm for multiagent decision making in coordination graphs. In *RoboCup-2005: Robot Soccer World Cup IX*, Osaka, Japan, July 2005.
- [22] D. Koller and A. Pfeffer. Representations and solutions for game-theoretic problems. *Artificial Intelligence*, 94(1-2):167–215, 1997.
- [23] O. Madani, S. Hanks, and A. Condon. On the undecidability of probabilistic planning and infinite-horizon partially observable Markov decision problems. In *Proc. of the National Conference on Artificial Intelligence*, pages 541–548, 1999.
- [24] S. Mannor, R. Rubinstein, and Y. Gat. The cross entropy method for fast policy search. In *Proc. of the International Conference on Machine Learning*, pages 512–519, 2003.
- [25] R. Nair, M. Tambe, M. Yokoo, D. V. Pynadath, and S. Marsella. Taming decentralized POMDPs: Towards efficient policy computation for multiagent settings. In *Proc. of the Int. Joint Conf. on Artificial Intelligence*, pages 705–711, 2003.
- [26] R. Nair, P. Varakantham, M. Tambe, and M. Yokoo. Networked distributed POMDPs: A synthesis of distributed constraint optimization and POMDPs. In *Proc. of the National Conference on Artificial Intelligence*, pages 133–139, 2005.
- [27] F. A. Oliehoek, J. F. Kooij, and N. Vlassis. A cross-entropy approach to solving Dec-POMDPs. In *International Symposium on Intelligent and Distributed Computing*, pages 145–154, Oct. 2007.
- [28] 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.
- [29] F. A. Oliehoek, M. T. J. Spaan, S. Whiteson, and N. Vlassis. Exploiting locality of interaction in factored Dec-POMDPs. In *Proc. of Int. Joint Conference on Autonomous Agents and Multi Agent Systems*, pages 517–524, 2008.
- [30] J. M. Ooi and G. W. Wornell. Decentralized control of a multiple access broadcast channel: Performance bounds. In *Proc. 35th Conf. on Decision and Control*, 1996.
- [31] M. J. Osborne and A. Rubinstein. *A Course in Game Theory*. The MIT Press, July 1994.
- [32] D. V. Pynadath and M. Tambe. The communicative multiagent team decision problem: Analyzing teamwork theories and models. *Journal of AI research (JAIR)*, 16:389–423, 2002.

- [33] M. Roth, R. Simmons, and M. Veloso. Exploiting factored representations for decentralized execution in multi-agent teams. In *Proc. of Int. Joint Conference on Autonomous Agents and Multi Agent Systems*, pages 467–463, May 2007.
- [34] S. Seuken and S. Zilberstein. Improved memory-bounded dynamic programming for decentralized POMDPs. In *Proc. of Uncertainty in Artificial Intelligence*, July 2007.
- [35] S. Seuken and S. Zilberstein. Memory-bounded dynamic programming for DEC-POMDPs. In *Proc. of the Int. Joint Conf. on Artificial Intelligence*, pages 2009–2015, 2007.
- [36] M. T. J. Spaan and F. S. Melo. Interaction-driven Markov games for decentralized multiagent planning under uncertainty. In *Proc. of Int. Joint Conference on Autonomous Agents and Multi Agent Systems*, pages 525–532, 2008.
- [37] D. Szer and F. Charpillet. Point-based dynamic programming for DEC-POMDPs. In *Proc. of the National Conference on Artificial Intelligence*, 2006.
- [38] D. Szer, F. Charpillet, and S. Zilberstein. MAA*: A heuristic search algorithm for solving decentralized POMDPs. In *Proc. of Uncertainty in Artificial Intelligence*, 2005.
- [39] P. Varakantham, J. Marecki, Y. Yabu, M. Tambe, and M. Yokoo. Letting loose a SPIDER on a network of POMDPs: Generating quality guaranteed policies. In *Proc. of Int. Joint Conference on Autonomous Agents and Multi Agent Systems*, 2007.

Appendix

Tables 2 to 5 give a numerical overview of the results presented in section 6.2 and 6.3.

DEC-TIGER				
horizon	avg. value	std. dev. value	max. value	avg. time
DICE (50 iterations)				
2	-4.08	0.78	-4.00	0.02
3	5.19	0.00	5.19	0.08
4	3.81	1.28	4.80	0.34
5	-1.58	4.15	4.58	1.37
6	-22.75	4.40	-13.20	6.05
7	-57.11	5.85	-46.56	27.93
8	-98.23	8.47	-80.09	147.73
DICE-A (50 iterations)				
2	-4.19	0.70	-4.00	5.90
3	5.19	0.00	5.19	8.43
4	3.35	2.76	4.80	10.55
5	-1.56	3.11	3.45	12.90
6	-24.05	5.14	-12.22	15.60
7	-59.45	6.61	-45.43	18.58
8	-103.59	8.04	-83.27	23.02
DICE-A (200 iterations)				
2	-4.14	0.60	-4.00	23.26
3	5.19	0.00	5.19	33.54
4	4.16	0.87	4.80	40.20
5	2.50	1.93	5.63	48.10
6	-2.17	3.42	4.53	56.69
7	-11.46	4.39	-5.28	66.94
8	-31.30	4.38	-21.21	80.40
JESP				
2	-17.09	10.22	-4.00	0.00
3	-21.36	10.76	5.19	0.00
4	-27.58	19.26	4.80	0.04
5	-22.72	17.44	7.03	0.49
6	-23.28	20.57	10.38	11.32
7	-21.42	19.16	9.99	144.21
8	-16.20	18.26	12.22	1741.39

Table 2: Results for the DEC-TIGER problem. Statistics over 100 restarts, except for DICE-A with $I = 200$ (which we performed with 20 restarts) and horizon 8 JESP (which only completed 30 restarts).

FACTORED FIREFIGHTING				
horizon	avg. value	std. dev. value	max. value	avg. time
DICE (50 iterations)				
2	-5.21	0.00	-5.21	36.45
3	-6.66	0.01	-6.65	347.89
DICE-A (50 iterations)				
2	-5.22	0.02	-5.21	9.89
3	-6.69	0.03	-6.65	13.27
4	-7.74	0.12	-7.48	17.05
5	-8.46	0.22	-8.05	20.77
6	-9.05	0.25	-8.53	25.57
7	-9.51	0.35	-8.83	30.92
8	-9.88	0.44	-9.14	36.81
JESP				
2	-5.28	0.09	-5.21	0.73
3	-6.71	0.06	-6.65	17.41
4	-7.52	0.07	-7.46	308.36

Table 3: Results for the FACTORED FIREFIGHTING problem over 100 restarts.

BROADCAST CHANNEL				
horizon	avg. value	std. dev. value	max. value	avg. time
DICE (50 iterations)				
2	2.00	0.00	2.00	0.05
3	2.93	0.05	2.99	0.26
4	3.80	0.08	3.89	1.08
5	4.69	0.09	4.79	4.45
6	5.52	0.09	5.67	18.37
7	6.34	0.08	6.48	79.89
8	7.03	0.09	7.26	384.23
DICE-A (50 iterations)				
2	2.00	0.00	2.00	6.38
3	2.92	0.05	2.99	8.32
4	3.80	0.07	3.89	10.20
5	4.65	0.09	4.79	12.26
6	5.50	0.08	5.66	14.47
7	6.28	0.08	6.46	16.93
8	6.98	0.11	7.22	19.74
JESP				
2	1.92	0.21	2.00	0.00
3	2.59	0.44	2.99	0.01
4	3.43	0.48	3.89	0.04
5	4.27	0.47	4.79	0.36
6	5.04	0.55	5.69	3.03
7	5.88	0.57	6.59	24.71
8	6.81	0.59	7.49	202.46

Table 4: Results for the BROADCAST CHANNEL problem over 100 restarts.

MEETING ON A GRID				
horizon	avg. value	std. dev. value	max. value	avg. time
DICE (50 iterations)				
2	0.91	0.00	0.91	0.71
3	1.55	0.01	1.55	3.43
4	2.23	0.02	2.24	15.37
5	2.91	0.07	2.96	67.60
6	3.57	0.06	3.64	292.88
DICE-A (50 iterations)				
2	0.91	0.00	0.91	10.77
3	1.54	0.01	1.55	16.51
4	2.21	0.02	2.24	21.86
5	2.85	0.05	2.93	27.19
6	2.64	0.07	2.73	32.86
7	2.97	0.06	3.07	37.66
8	3.23	0.09	3.37	42.63
JESP				
2	0.86	0.08	0.91	0.02
3	1.38	0.17	1.55	0.50
4	1.97	0.24	2.24	12.11
5	2.53	0.30	2.97	287.02

Table 5: Results for the MEETING ON A GRID problem over 100 restarts.

FIREFIGHTING				
n	avg. value	std. dev. value	max. value	avg. time
DICE (50 iterations)				
2	-5.76	0.02	-5.74	13.77
3	-2.41	0.04	-2.39	59.07
4	-1.08	0.01	-1.07	249.88
5	-0.51	0.00	-0.51	1101.54
DICE-A (50 iterations)				
2	-5.80	0.04	-5.74	17.86
3	-2.48	0.05	-2.39	35.05
4	-1.09	0.02	-1.07	91.02
5	-0.51	0.01	-0.51	263.92
JESP				
2	-5.76	0.04	-5.74	0.47
3	-2.41	0.03	-2.39	4.17
4	-1.08	0.01	-1.07	20.15
5	-0.51	0.00	-0.51	83.61

Table 6: Results for the $h = 3$ FIREFIGHTING problem with varying number of agents over 20 restarts.

FIREFIGHTING				
n	avg. value	std. dev. value	max. value	avg. time
DICE (50 iterations)				
2	-6.66	0.06	-6.58	68.03
3	-2.52	0.02	-2.45	490.58
4	-1.07	0.00	-1.07	4227.08
DICE-A (50 iterations)				
2	-6.81	0.08	-6.67	23.07
3	-2.59	0.08	-2.45	45.22
4	-1.09	0.02	-1.07	121.37
5	-0.51	0.00	-0.50	350.04
JESP				
2	-6.62	0.03	-6.58	6.71
3	-2.45	0.03	-2.44	110.57
4	-1.08	0.01	-1.07	1040.75

Table 7: Results for the $h = 4$ FIREFIGHTING problem with varying number of agents over 20 restarts. For $n = 4$ is DICE shows an average over 4 completed runs and JESP over 18 completed runs.

