An Optimal Best-First Search Algorithm for Solving Infinite Horizon DEC-POMDPs

Daniel Szer and François Charpillet

INRIA Lorraine - LORIA, MAIA Group,
54506 Vandœuvre-lès-Nancy, France
{szer, charp}@loria.fr
http://maia.loria.fr

Abstract. In the domain of decentralized Markov decision processes, we develop the first complete and optimal algorithm that is able to extract deterministic policy vectors based on finite state controllers for a cooperative team of agents. Our algorithm applies to the discounted infinite horizon case and extends best-first search methods to the domain of decentralized control theory. We prove the optimality of our approach and give some first experimental results for two small test problems. We believe this to be an important step forward in learning and planning in stochastic multi-agent systems.

1 Introduction

Efficient learning and planning algorithms for problems within distributed and only partially observable stochastic environments can be particularly useful in a large number of todays research areas, such as network traffic routing [1], decentralized supply chains [7], or the control of a robot team for space exploration [12] or humanitarian missions [11]. Formalizing the problem of optimal control in a rigorous way is an important part of the solution, and the theory of Markov Decision Processes (MDPs) has been shown to be particularly powerful in that context [16]. It is only recently however, that the Markov framework has been extended to problems of decentralized control [5], [3]. The major additional complexity in multi-agent decision making lies in the fact that agents may have different partial information about both the underlying system state and the local information held by the remaining agents. Reasoning about the potential private information of a teammate may in fact lead into an infinite loop of *"I believe that you believe"*-like assumptions. This is the reason why solving decentralized partially observable MDPs optimally is significantly harder, namely NEXP-complete [3], than solving their centralized counterparts.

While some important progress has been made in solving single-agent MDPs, we still lack in efficient algorithms for the multi-agent case. Depending on the problem constraints, different solution concepts are required, and for some of them, optimal non-trivial algorithms have not yet been established. Characterizing the optimal solution of a general decentralized MDP however constitutes

J. Gama et al. (Eds.): ECML 2005, LNAI 3720, pp. 389–399, 2005.

a crucial step toward both efficient approximation techniques and learning algorithms. We will focus in this paper on infinite horizon problems that can be solved using deterministic finite memory controllers, and we are able to present the first complete algorithm to solve this class of problems optimally. Our approach is an extension of best-first search techniques to decentralized control theory and shows to be very effective compared to existing solutions.

In the remainder of the paper, we will introduce the DEC-POMDP framework for decentralized decision making under uncertainty and expose some existing approaches within this problem family, before describing our search method and related experimental results.

2 Decentralized Markov Decision Processes

The family of Markov decision processes describes discrete stochastic systems that evolve under the influence of one or multiple controllers. With each transition of the system is associated a reward value, and the objective of the controller is to select precisely that sequence of actions that maximizes the collection of rewards in the long run. For the case of several distributed but cooperative controllers, their objective is to act selfishly as to maximize the reward collected by the team.

2.1 The DEC-POMDP Model

We base our work on the DEC-POMDP formalism introduced by [3], although alternative definitions are equally allowed.

Definition 1 (DEC-POMDP). *An n-agent DEC-POMDP is given as a tuple* $\langle S, \{A_i\}, P, R, \{\Omega_i\}, O, p_0 \rangle$*, where*

- S *is a finite set of states*
- A_i *is a finite set of actions, available to agent* i
- $P(s, a_1, \ldots a_n, s')$ *is a function of transition probabilities*
- $R(s, a_1, \ldots a_n, s')$ *is a reward function*
- Ω_i *is a finite set of observations for agent* i
- $O(s, a_1, \ldots a_n, o_1, \ldots o_n, s')$ *is a function of observation probabilities*
- p_0 *is the initial state distribution of the system*

Solving a DEC-POMDP can be seen as finding a set of n *policies*, one for each controller, that yield maximum reward when being executed synchronously. The optimization problem can therefore be stated as maximizing the following expectation value

$$E\left[\sum_{t=0}^{\infty} \gamma^t R\left(s_t, (a_1, \ldots a_n)_t, s_{t+1}\right) \;\middle|\; p_0\right] \quad \text{with} \quad 0 \leq \gamma < 1 \tag{1}$$

where γ is a discount factor to avoid infinite sums. We will denote q_i the policy associated with agent i. In order to be optimal, the Markov assumption requires a policy to depend on the whole information available to the

agent at time t, namely its complete history of past observations and actions: $(q_i)_t = q_i((a_i)_0, \ldots (a_i)_{t-1}, (o_i)_0, \ldots (o_i)_t \mid p_0)$. For infinite horizon problems however, this would require a controller to have infinite memory, which is not always possible. We will therefore specify the nature of the controller in more detail.

2.2 Policies for DEC-POMDPs

A widely accepted class of policies for single-agent POMDPs can be represented as *policy graphs*. A policy graph can be described by a set of nodes, which contain the actions to be executed, and a set of arcs, which are parametrized by the observations the agent gets. A step in policy execution consists of executing the action given by the current node, and transitioning to the next node, based on the observation signal that occurred. For finite horizon problems, an optimal policy graph can always be represented as a tree [10], whereas for the infinite horizon case, loops have to be allowed. A policy graph with loops is called a *finite state controller*:

Definition 2 (FSC). *A finite state controller (FSC) is a policy graph, defined as* $q = \langle N, \alpha, \eta, n_0 \rangle$*, where*

- N *denotes a set of nodes*
- $\alpha = \alpha(n)$ *is the action selected in node* n
- $\eta = \eta(n, o)$ *is the successor node when observation* o *is perceived in node* n
- n_0 *is the starting node*

An example of a 3-node FSC is given in Figure 1. For the case of decentralized problems with multiple controllers, the goal is it to find a set of FSCs, one for each agent, such that their concurrent execution maximizes the expectation value given in (1). We will call such a set a *policy vector*:

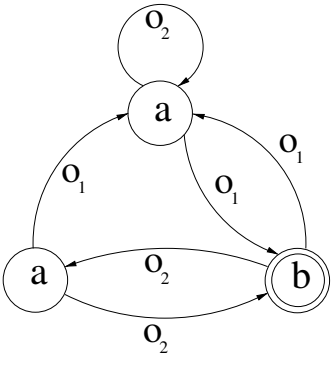

Fig. 1. A deterministic finite state controller with 3 nodes for a problem with 2 actions (a and b) and 2 observations (o_1 and o_2)

Definition 3 (Policy Vector). *A policy vector δ is defined as $\delta = (q_1, \ldots q_n)$, such that q_i constitutes a policy, in our case a FSC, assigned to agent i.*

As stated earlier, finite memory controllers are naturally limited in treating infinite horizon problems, and increasing the controller size will in general lead to higher rewards. We therefore state our optimization criterion as finding the best policy vector *for a given controller size*.

2.3 Related Work

Solving cooperative but decentralized Markov decision processes has only been recently addressed by the research community. After the establishement of the formal DEC-POMDP model by [3], and the alternative MTDP model by [17], the first optimal algorithm for finite horizon problems, based on dynamic programming, has been suggested in [9]. We recently proposed an alternative approach, based on heuristic search [19]. Furthermore, there exist several suboptimal solutions that adopt concepts from game theory, such as described in [6], [14], or that use local optimization techniques as described in [15]. Although these algorithms are often much easier to apply, the quality of their solution can be more or less unsatisfactory depending on the problem. A first attempt to solve general DEC-POMDPs with infinite horizon has been made by Bernstein et al. in [4]. Their algorithm is based on policy iteration for stochastic finite state controllers, and is therefore related to our approach, although it is not guaranteed to produce optimal controllers. We will indeed be able to show that, while we restrict ourselves to deterministic automata only, our algorithm outperforms their approach on the test problems we studied. There exist several algorithms that treat special subclasses of decentralized MDPs, such as transition independent DEC-MDPs, where agents do not interfere directly while execution [2].

3 Best-First Search for Infinite Horizon DEC-POMDPs

Solving Markov decision problems usually involves maximizing an evaluation function in either state space or policy space, with our approach being an example for the latter.

3.1 Searching in the Space of Policy Vectors

Forward search in the space of policy vectors can be considered as an incremental construction of an optimal policy based on evaluations of only partially completed policy stubs. In each step, the most promising stub is selected and further developed, hence the best-first approach. A section of such a search tree is shown in Figure 2. We recall that $\delta = (q_1, \ldots q_n)$ denotes a policy vector of FSCs. For a completely defined policy vector, we set $V_\delta(p_0)$ as the *value* of executing δ in p_0, which is nothing more than the expectation introduced in (1). We then state our maximization problem as follows:

$$\delta^* = \arg\max_{\delta} V_\delta(p_0) \tag{2}$$

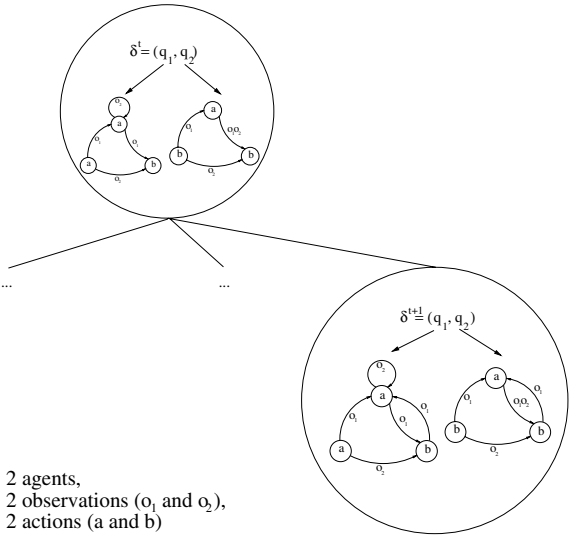

Fig. 2. A section of the multi-agent best-first search tree, showing a partially defined policy vector for 2 agents, and one of its stronger constrained child vectors

Evaluating policy vectors can be done using the model parameters P, R, and O of the DEC-POMDP. We will describe this in more detail in the following subsections.

3.2 Evaluating Partially Defined Policy Vectors

Because of the incremental nature of the search process, we will have to emphasize on what we understand by a policy stub. We recall that each FSC is defined by its number of nodes N, and its two functions $\alpha : N \to A$ and $\eta : N \times O \to N$. A policy stub is a FSC where either α or η or both are only *partially defined*. Obviously, any partially defined FSC can be completed easily at random by assigning actions and successor nodes at those points where α and η are not constrained. The crucial step however consists in estimating efficiently *all* possible completions of a policy stub, in order to determine whether or not to expand the corresponding leaf node of the search tree. Heuristic search methods such as A* have been shown to be very efficient in those cases where an *upper bound* estimate for the set of possible completions can be established. In order to show that a similar upper bound can indeed be defined in our case, we will introduce two mappings that specify the current constrainment of the FSCs:

$$\Lambda_i(n) = \begin{cases} \{\alpha_i(n)\}, \text{ if } \alpha_i \text{ is defined in } n \\ A, \qquad \text{otherwise} \end{cases}$$

$$\Pi_i(n, o) = \begin{cases} \{\eta_i(n, o)\}, \text{ if } \eta_i \text{ is defined for } n \text{ and } o \\ N, \qquad \text{otherwise} \end{cases}$$

Similarly, we define the multi-controller extensions $\Lambda(\mathbf{n}) = (\Lambda_1(n), \ldots \Lambda_n(n))$ and $\Pi(\mathbf{n}, \mathbf{o}) = (\Pi_1(n, o), \ldots \Pi_n(n, o))$.

It has been pointed out by Sondik and later by Hansen [8] that evaluating a policy represented as a FSC consists in solving a system of linear equations. In fact, the cross-product between a FSC and a POMDP constitutes itself a finite MDP [13]. We extend this result to the multi-agent case:

Definition 4 (Multi-agent cross-product MDP). *Given a DEC-POMDP* $\langle S, \{A_i\}, P, R, \{\Omega_i\}, O, p_0\rangle$ *and a policy vector* $\delta = \langle \{N_i\}, \{\alpha_i\}, \{\eta_i\}\rangle$*, we define a cross-product MDP* $\langle \overline{S}, \overline{A}, \overline{P}, \overline{R}\rangle$*, with*

- $\overline{S} = (\underset{i}{\times} N_i) \times S$
- $\overline{A} = \underset{i}{\times}(A_i \times N_i^{\Omega_i})$
- $\overline{P}((\mathbf{n}, s), (\mathbf{a}, \eta^{\mathbf{n}}), (\mathbf{n}', s')) = P(s, \mathbf{a}, s') \sum_{\substack{\mathbf{o} \in \times \mathbf{\Omega} \\ s.t.\ \eta^{\mathbf{n}}(\mathbf{o}) = \mathbf{n}'}} O(s, \mathbf{a}, \mathbf{o}, s')$
- $\overline{R}((\mathbf{n}, s), (\mathbf{a}, \eta^{\mathbf{n}}), (\mathbf{n}', s')) = R(s, \mathbf{a}, s')$

and where $\eta^{\mathbf{n}}$ *is a mapping that - given a vector of nodes* $\mathbf{n}$ *- determines a vector of successor nodes* $\mathbf{n}'$ *for each vector of observations* $\mathbf{o}$*,* $\eta^{\mathbf{n}} : \mathbf{\Omega} \to \mathbf{N}$*.*

Solving the cross-product MDP can be done through common dynamic programming techniques, leading to a value function over the augmented state space $\overline{S}$ and the following fixed point:

$$\overline{V}_\delta(\mathbf{n}, s) = \max_{\mathbf{a} \in \Lambda(\mathbf{n})} \left\{ \sum_{s', \mathbf{o}} P(s', \mathbf{o} | s, \mathbf{a}) \Big[R(s, \mathbf{a}, s') + \gamma \max_{\mathbf{n}' \in \Pi(\mathbf{n}, \mathbf{o})} \overline{V}_\delta(\mathbf{n}', s') \Big] \right\} \quad (3)$$

This value function is the multi-agent extension of the one given in [13].

Lemma 1. *For any policy vector* δ' *that can be obtained from* δ *by adding further constraints on* Λ *or* Π*,* $\overline{V}_\delta \geq \overline{V}_{\delta'}$*.*

Proof. The lemma states that $\overline{V}_\delta$ is indeed an upper bound for all policy vectors that might result from δ by further constraining it. This is true since constraining Λ or Π will simply result in reducing the set of options under the max-operators $\max_{\mathbf{a} \in \Lambda(\mathbf{n})}$ and $\max_{\mathbf{n}' \in \Pi(\mathbf{n}, \mathbf{o})}$ in (3), and the value function can therefore never increase.

If the vector of FSCs is completely defined, the cross-product MDP degenerates to a simple Markov chain, and the upper bound coincides with the true value of the policy vector. We can evaluate an upper bound for the value of any partially defined policy vector and start state distribution p_0:

$$\overline{V}_\delta(p_0) = \max_{\mathbf{n}} \sum_{s_i} p_0(s_i) \overline{V}_\delta(\mathbf{n}, s_i) \quad (4)$$

3.3 An Optimal Heuristic Search Algorithm for Decentralized POMDPs

Theorem 1. *The heuristic best-first search algorithm in* [Algorithm 1] *is complete and returns the optimal solution for the given controller size.*

Proof. The search process will eventually terminate in the worst case after enumerating all possible policy vectors, which means after constructing the complete search tree. The leaf node with the highest value then contains an optimal solution to the problem. If the search terminates earlier and returns a policy vector δ, we can guarantee by the "best-first" property that no other active leaf node presents a higher evaluation. Since the evaluation function itself constitutes an upper bound for the value of any further constrained policy vector, we know that all unvisited child nodes will present values that fall below this bounded. This excludes the existence of any policy vector with a higher value, and thus guarantees the optimality of the solution.

Algorithm 1. Best-first search for infinite horizon DEC-POMDPs

Require: D_0 initialized with the skeleton of an unconstrained policy vector
1: **repeat**
2: Select $\delta^* \in D_i$ such that $\forall \delta \in D_i$: $\overline{V}_{\delta}(p_0) \leq \overline{V}_{\delta^*}(p_0)$
3: Construct $\delta^{*'}$, the next child of δ^*
4: **if** $\delta^{*'}$ is an improved suboptimal solution **then**
5: Report $\delta^{*'}$
6: **for** all $\delta \in D_i$ **do**
7: **if** $\overline{V}_{\delta}(p_0) \leq \overline{V}_{\delta^{*'}}(p_0)$ **then**
8: $D_i \leftarrow D_i \setminus \delta$
9: **end if**
10: **end for**
11: **end if**
12: $D_i \leftarrow D_i \cup \delta^{*'}$
13: **if** δ^* is fully expanded **then**
14: $D_i \leftarrow D_i \setminus \delta^*$
15: **end if**
16: **until** $\exists \delta^* \in D_i$ s.t. δ^* is *complete* and $\forall \delta \in D_i$: $\overline{V}_{\delta}(p_0) \leq \overline{V}_{\delta^*}(p_0) = V_{\delta^*}(p_0)$

4 Experimental Results

We tested the heuristic search approach on two problems that have already been studied before in [4], namely a broadcast channel problem, and a 2-robot navigation task. The discount rate for all problems is $\gamma = 0.9$.

4.1 A Broadcast Channel Problem

The first setting simulates a simplified multi-access broadcast channel, where agents are situated at the nodes of the system. Each agent has to decide whether

or not to send one of the messages from its message buffer. Sending is exclusive, which means that only one message can go through a channel at each time. If both agents try to send a message at the same time over the same channel, a collision occurs, and messages will remain in the buffer. The problem is partially observable and hence decentralized, since agents can only observe the state of their own message buffers but do not know whether or not any other agent has something to send as well. The common goal of all participating agents is to maximize the throughput of the system, with a reward of 1.0 given for any message that has been transmitted. In the experiments we conducted, there are 2 agents and 2 possible actions for each one of them (`send`, `not send`). The buffer size is 1, and the number of global states thus is 4, namely the cross-product of the local buffer states. There are 6 possible observations, characterized by the local buffer state (`empty`, `full`) and a status flag of the channel from the previous time step (`idle`, `active`, `collision`). New messages arrive with a rate of $p_1 = 0.9$ for agent 1, and $p_2 = 0.1$ for agent 2.

The highest possible discounted sum of rewards that can be attained for this problem assuming full global observability is $\sum_t \gamma^t(1.0) = 9.0$, which would mean that a message could be transmitted at each time step. Surprisingly, we can see in Figure 5 that this value is almost attained in our case with just a deterministic single-node FSC, althought the problem is now decentralized and only partially observable. The bounded policy iteration approach for stochastic controllers produces less competitive policies.

4.2 A Robot Navigation Task

In the second problem, two agents navigate on a two by two grid-world, without interfering with each other. Their goal is to stay both on the same grid cell - which produces a reward of 1.0 - but their observation capabilities are limited so they don't see each other, and they also have limited sensing capabilities concerning the environment: There are only 4 observations, indicating whether there is a wall to their left and/or to their right. Each agent has 5 actions to move in any direction or stay on its current cell, but transitions are stochastic as given by Figure 4: The agent only moves with probability 0.6 to its intended direction. The problem has 16 states. Figure 5 shows that

$\mathbf{Buffer}_t$	Action	$\mathbf{Buffer}_{t+1}$	Prob
`empty`	(any)	`empty`	$1.0 - p_i$
		`full`	p_i
`full`	`send`	`empty`	$1.0 - p_i$
		`full`	p_i
	`not send`	`empty`	0.0
		`full`	1.0

Fig. 3. Channel problem: Transition probabilities for each one of the buffers

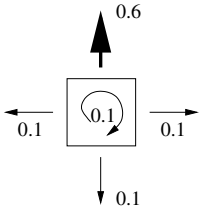

Fig. 4. Navigation task: Transition probabilities for action `North`

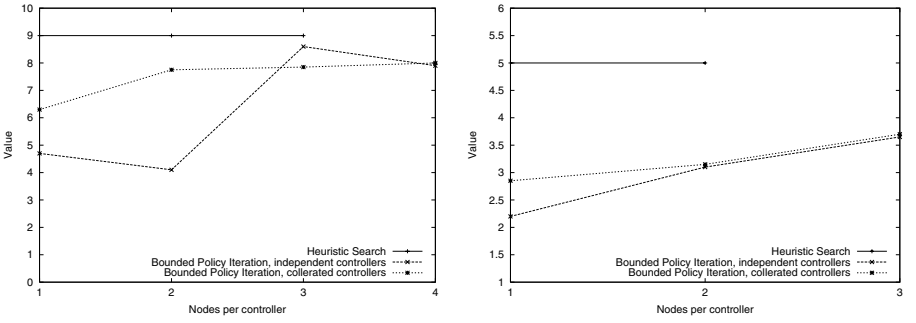

Fig. 5. Value of optimal deterministic policy vector for the heuristic approach, and average value per trial run for two versions of bounded policy iteration on stochastic controllers. Left: Channel problem - Right: Robot problem.

the search algorithm is again more competitive than the policy iteration approach. However, it takes more time to converge and may in the end run out of memory.

The experimental results show that the advantage of using stochastic controllers, which have the theoretical ability to produce higher average rewards than deterministic ones [18], might be more than consumed by the local optimality of the algorithms that compute them. In addition, the deterministic controllers with more than one node are degenerated versions of the one controller case: In the given example problems, having a larger memory does not necessary help, which is why the value of the deterministic controllers do not increase with increasing size.

5 Discussion

We have presented a new optimal algorithm to solve a particular class of decentralized POMDPs with infinite horizon. It is able to compute better controllers than a very recent policy iteration algorithm on two test problems, although it still suffers from the proved complexity of this class of problems. It should be a valuable step forward in establishing more efficient algorithms and approximation techniques. There are other possible ways of tackling problems of decentralized control: Instead of constraining the controller size, one could for example impose a bound on the solution quality. It remains an open problem, whether decentralized policy vectors can also be learned in an efficient way when the environment is only partially observable.

Acknowledgments

We thank Shlomo Zilberstein and the anonymous reviewers for their helpful comments.

References

1. Eitan Altman. Applications of Markov Decision Processes in Communication Networks: A Survey. Technical Report RR-3984, INRIA Sophia-Antipolis, 2000.
2. Raphen Becker, Shlomo Zilberstein, Victor Lesser, and Claudia V. Goldman. Solving Transition Independent Decentralized Markov Decision Processes. *Journal of Artificial Intelligence Research*, 22:423–455, 2004.
3. Daniel S. Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. The Complexity of Decentralized Control of Markov Decision Processes. *Mathematics of Operations Research*, 27(4):819–840, 2002.
4. Daniel S. Bernstein, Eric A. Hansen, and Shlomo Zilberstein. Bounded Policy Iteration for Decentralized POMDPs. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence*, 2005.
5. Craig Boutilier. Sequential Optimality and Coordination in Multiagent Systems. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, pages 478–485, 1999.
6. Iadine Chadès, Bruno Scherrer, and François Charpillet. A Heuristic Approach for Solving Decentralized-POMDP: Assessment on the Pursuit Problem. In *Proceedings of the 2002 ACM Symposium on Applied Computing*, 2002.
7. Fangruo Chen. Decentralized supply chains subject to information delays. *Management Science*, 45:1076–1090, 1999.
8. Eric A. Hansen. An Improved Policy Iteration Algorithm for Partially Observable MDPs. In *Proceedings of the 10th Conference on Neural Information Processing Systems*, 1997.
9. Eric A. Hansen, Daniel S. Bernstein, and Shlomo Zilberstein. Dynamic Programming for Partially Observable Stochastic Games. In *Proceedings of the 19th National Conference on Artificial Intelligence*, 2004.
10. Leslie Pack Kaelbling, Michael L. Littman, and Anthony Cassandra. Planning and Acting in Partially Observable Stochastic Domains. *Artificial Intelligence*, 101:99–134, 1998.
11. Hiroaki Kitano and Satoshi Tadoroko. Robocup-rescue: A Grand Challenge for Multiagent and Intelligent Systems. *AI Magazine*, 22(1):39–51, 2001.
12. Maja J. Mataric and Gaurav S. Sukhatme. Task-allocation and Coordination of Multiple Robots for Planetary Exploration. In *Proceedings of the 10th International Conference on Advanced Robotics*, 2001.
13. Nicolas Meuleau, Kee-Eung Kim, Leslie Pack Kaelbling, and Anthony Cassandra. Solving POMDPs by Searching the Space of Finite Policies. In *Proceedings of the 15th Annual Conference on Uncertainty in Artificial Intelligence*, 1999.
14. R. Nair, M. Tambe, M. Yokoo, D. Pynadath, and S. Marsella. Taming Decentralized POMDPs: Towards Efficient Policy Computation for Multiagent Settings. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, 2003.
15. Leonid Peshkin, Kee-Eung Kim, Nicolas Meuleau, and Leslie Kaelbling. Learning to Cooperate via Policy Search. In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, 2000.
16. Martin L. Puterman. *Markov Decision Processes – Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 1994.

17. David V. Pynadath and Milind Tambe. The Communicative Multiagent Team Decision Problem: Analyzing Teamwork Theories and Models. *Journal of Artificial Intelligence Research*, pages 389–423, 2002.
18. Satinder P. Singh, Tommi Jaakkola, and Michael I. Jordan. Learning Without State-Estimation in Partially Observable Markovian Decision Processes. In *Proceedings of the 11th International Conference on Machine Learning*, 1994.
19. Daniel Szer, François Charpillet, and Shlomo Zilberstein. MAA*: A heuristic search algorithm for solving decentralized POMDPs. In *Proceedings of the 21th Conference on Uncertainty in Artificial Intelligence*, 2005.