Dual Formulations for Optimizing Dec-POMDP Controllers

Akshat Kumar[†] and **Hala Mostafa**[‡] and **Shlomo Zilberstein**[*]
[†]School of Information Systems, Singapore Management University, [‡]United Technologies Research Center
[*]College of Information and Computer Sciences, University of Massachusetts Amherst
akshatkumar@smu.edu.sg, mostafh@utrc.utc.com, shlomo@cs.umass.edu

Abstract

Decentralized POMDP is an expressive model for multiagent planning. Finite-state controllers (FSCs)—often used to represent policies for infinite-horizon problems—offer a compact, simple-to-execute policy representation. We exploit novel connections between optimizing decentralized FSCs and the dual linear program for MDPs. Consequently, we describe a dual mixed integer linear program (MIP) for optimizing *deterministic* FSCs. We exploit the Dec-POMDP structure to devise a compact MIP and formulate constraints that result in policies executable in partially-observable decentralized settings. We show analytically that the dual formulation can also be exploited within the expectation maximization (EM) framework to optimize *stochastic* FSCs. The resulting EM algorithm can be implemented by solving a sequence of linear programs, without requiring expensive message passing over the Dec-POMDP DBN. We also present an efficient technique for policy improvement based on a weighted entropy measure. Compared with state-of-the-art FSC methods, our approach offers over an order-of-magnitude speedup, while producing similar or better solutions.

Introduction

Decentralized partially-observable MDPs (Dec-POMDPs) have emerged as a prominent framework for collaborative sequential decision making (Bernstein et al. 2002). Dec-POMDPs capture planning problems where agents act based on different partial information about the environment and about each other to maximize a global reward function. Applications of Dec-POMDPs include coordinating planetary rovers (Becker et al. 2004), target tracking by a team of sensors (Nair et al. 2005), and improving throughput in wireless networks (Pajarinen, Hottinen, and Peltonen 2014).

For *finite-horizon* Dec-POMDPs, numerous point-based algorithms have been developed (Seuken and Zilberstein 2007; Kumar and Zilberstein 2010b). An alternate policy representation that relies on the distribution over world states and joint histories of agents, also called *occupancy states*, offers sufficient statistic for decentralized planning (Dibangoye et al. 2013; Oliehoek 2013). However, unlike their point-based POMDP counterparts (Pineau, Gordon, and Thrun 2006), Dec-POMDP solvers suffer due to the lack of a compact belief state representation.

Finite-state controllers (FSCs) alleviate that problem and more easily generalize from POMDPs (Hansen 1998) to Dec-POMDPs (Amato, Bernstein, and Zilberstein 2010). Executing a FSC-based policy requires a simple lookup table, with no belief updates. This resource-efficient execution is desirable in many settings, such as battery-constrained mobile devices (Grzes et al. 2015) and wireless devices (Pajarinen, Hottinen, and Peltonen 2014). Grzes et al. provide real world energy efficiency comparisons between controller-based POMDP policies and point-based approaches for assistive healthcare applications on a mobile device, showing that FSCs are the most effective representation in terms of preserving battery life and being highly responsive w.r.t. the frequency of policy queries (point-based approaches require belief update, which may be time consuming, delaying response). In addition, controllers can provide significantly more semantic information about the policy than occupancy states (Dibangoye et al. 2013), which is crucial for human understanding of the policy, for example in assistive healthcare applications (Hoey et al. 2012). Similarly, controller-based policies have been recently used for multi-robot coordination under uncertainty where interpretable compact policies are desired (Amato et al. 2015).

Optimizing controllers for POMDPs and Dec-POMDPs is challenging. Several approaches have been developed to optimize deterministic (Amato and Zilberstein 2008) or stochastic (Bernstein et al. 2009; Amato, Bernstein, and Zilberstein 2010; Kumar and Zilberstein 2010a; Pajarinen and Peltonen 2011b) controllers. We present a mixed integer linear programming (MIP) formulation for optimizing deterministic controllers for Dec-POMDPs. Our approach provides a direct connection between planning for Dec-POMDPs and the dual LP formulation for MDPs (Puterman 1994). This *dual view* of multiagent planning allows for adaptations of techniques developed for large MDPs, including factored state and action spaces. The MIP formulation is advantageous over nonlinear programming (NLP) formulations (Amato, Bernstein, and Zilberstein 2010) and inference-based techniques (Kumar and Zilberstein 2010a) because most off-the-shelf MIP solvers provide an upper bound on solution quality that can be used to calculate the optimality gap. Given enough time, MIP solvers provide optimal fixed-size controllers, which cannot be guaranteed with non-convex programming solvers.

Beyond deterministic controllers, we show analytically that the dual formulation can benefit the expectation maximization (EM) framework to optimize stochastic FSCs (Kumar, Zilberstein, and Toussaint 2015). The resulting EM algorithm can be implemented by solving a sequence of linear programs, without expensive message passing over the Dec-POMDP DBN. Thus, the dual perspective on decentralized planning can further benefit existing approaches.

The benefits of using MIP and dual LP based formulations have been recognized by the planning community. For example, a MIP-based approach to obtain optimal policy for *finite-horizon* Dec-POMDP was developed by Aras and Dutech (2010). This approach works by incorporating the world state and all possible observation histories for every time step into an extended state space over which a MIP is defined. The main disadvantage of such approaches is that the size of the observation history increases exponentially with the horizon, severely impacting scalability. A dual LP-based approach was presented by Witwicki and Durfee (2007) for a subclass of Dec-MDPs, while Mostafa and Lesser (2011) presented a MIP-based approach for another Dec-MDP subclass. For POMDPs, a MIP-based FSC optimization was proposed by Kumar and Zilberstein (2015). In contrast to previous approaches, our approach is based on optimizing a FSC-based stationary policy for infinite-horizon Dec-POMDPs. We formulate constraints that yield policies that are executable in partially-observable decentralized settings. Furthermore, the size of our MIP is polynomial in the FSC size and is therefore far more scalable than previous MIP approaches (Aras and Dutech 2010).

In previous work on optimizing decentralized controllers (Amato, Bernstein, and Zilberstein 2010; Kumar and Zilberstein 2010a), it is often hard to determine the "right" size of the controller in terms of the number of nodes. Larger controllers better approximate the optimal policy but are more challenging to optimize. We use the recently developed notion of history-based controllers (Kumar and Zilberstein 2015), and extend it for the decentralized setting. We develop techniques that can find good controllers via an iterative process for adding controller nodes using a principled heuristic. Using MIP-based optimization and the node addition heuristics, we show that our approach can find high quality compact controllers for most standard problems, while providing more than an order-of-magnitude speedup over previous controller optimization approaches.

The Dec-POMDP Model

We define the two-agent Dec-POMDP as follows. Agents 1 and 2 select actions from the sets A and B, respectively, with $a \in A$ and $b \in B$ denoting their individual actions. The state transition probability $P(s'|s,a,b)$ depends upon the actions of both agents. Upon taking the joint-action $\langle a,b\rangle$ in state $s \in S$, the agents receive the joint-reward $R(s,a,b)$. Y is a finite set of observations for agent 1 and Z for agent 2. $O(yz|s,a,b)$ denotes the probability of agent 1 observing $y \in Y$ and agent 2 observing $z \in Z$ when the joint-action $\langle a,b\rangle$ results in state s. Future rewards are discounted by a factor $\gamma < 1$. The initial belief over world states is $b_0(s)$.

An agent's individual policy maps its local action-observation history to the next action. However, an explicit representation of this mapping is ill-suited for infinite-horizon problems. Hence, we represent an agent's policy using a finite-state controller (FSC). The FSC for an agent (say agent 1 w.l.o.g.) is specified by parameters $\theta = \langle \mathcal{P}, \pi, \lambda \rangle$, where $\mathcal{P}$ is a set of memory nodes or controller nodes for the agent. The action mapping for each node is denoted using $\pi : \mathcal{P} \to \Delta A$, where ΔA is the set of all probability distributions over A. The node mapping $\lambda : \mathcal{P} \times Y \to \Delta \mathcal{P}$ encodes the transition function, specifying the new node given current node p and observation received y. We optimize deterministic controllers. Therefore, each $\pi(a|p)$ is a 0-1 variable such that $\sum_{a\in A} \pi(a|p) = 1 \; \forall p \in \mathcal{P}$. The deterministic node transition function $\lambda(p'|p,y)$ is defined analogously.

We denote nodes of agent 1's controller by p and agent 2's nodes by q. The value for starting the joint-controller in nodes $\langle p,q\rangle$ at state s is given by:

$$V(p,q,s) = \sum_{a,b} \pi(a\,|\,p)\pi(b\,|\,q) \Big[R(s,a,b) + \gamma \sum_{s'} P(s'\,|\,s,a,b)$$
$$\sum_{y,z} O(yz\,|\,s',a,b) \sum_{p'q'} \lambda(p'\,|\,p,y)\lambda(q'\,|\,q,z)V(p',q',s') \Big] \quad (1)$$

Assuming that execution starts at nodes $\langle p_0, q_0\rangle$ at time zero, the value of a joint-controller for the initial belief b_0 is $V(b_0;\boldsymbol{\theta}) = \sum_s b_0(s) V(p_0,q_0,s;\boldsymbol{\theta})$, where $\boldsymbol{\theta}$ denotes the FSC parameters for both agents. Our goal is to find an optimal joint-policy $\boldsymbol{\theta}^\star$ such that $V(b_0;\boldsymbol{\theta}^\star)$ is maximized. Optimizing fixed-size deterministic controllers is NP-Hard. In fact, even optimizing a reactive controller with one node per observation is NP-Hard (Littman 1994).

The Dual MIP for Dec-POMDPs

Our dual MIP formulation for optimizing controllers is based on the dual LP formulation for optimizing MDP policies. Therefore, we first describe relevant aspects of dual LP for MDPs. The linear program for finding an optimal MDP policy is often represented as follows (Puterman 1994):

$$\max_{\{x(\cdot,\cdot)\}} \sum_s \sum_a R(s,a)x(s,a) \quad (2)$$

$$\sum_a x(j,a) = \sum_s \sum_a \gamma \; P(j|s,a)x(s,a) + b_0(j) \; \forall j \in S \quad (3)$$

where the variable $x(s,a)$, also known as *occupancy measure*, intuitively denotes the total discounted amount of time the environment state is s and action a is taken. Therefore, the total reward corresponding to being in state s and taking action a is $R(s,a)x(s,a)$. The constraints enforce the flow conservation principle.

Definition 1. *The variable $x(s,a)$ is defined as follows (Puterman 1994):*

$$x(s,a) = \sum_{j\in S} b_0(j) \sum_{t=1}^{\infty} \gamma^{t-1} P(s_t = s, a_t = a \mid s_1 = j) \quad (4)$$

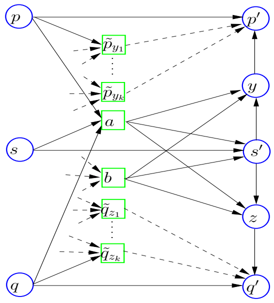


Figure 1: CP-MDP DBN

Cross-Product MDP for Dec-POMDP

We develop a mathematical programming formulation analogous to the dual LP formulation of MDPs to optimize decentralized FSCs. First, we show that a Dec-POMDP can be viewed as an MDP over cross-product states of the form (p, q, s), where s is a world state and $\langle p, q\rangle$ is a joint-controller state. While cross-product MDP (CP-MDP) has been previously used (Seuken and Zilberstein 2007), we develop the *dual formulation for CP-MDPs*, thereby making the following contributions:

- A naive adaptation of dual formulation to CP-MDP leads to exponential number of variables in the number of observations. We exploit the CP-MDP's DBN structure (Fig. 1) to reduce the program size to polynomial in the Dec-POMDP model parameters.
- We formulate a new set of constraints that enable the dual formulation for CP-MDP to find a *valid* policy executable in partially-observable decentralized settings.

Dual Formulation for CP-MDP

Our first contribution is the compact dual formulation for CP-MDP shown in Table 1. $x(a|p)$, $x(b|q)$, $x(p'|p, y)$, and $x(q'|q, z)$ are binary policy variables (π, λ). All other variables are continuous occupancy measures under different marginalizations. In what follows, we formally define these variables and the necessary associated constraints.

Variables: $x(p, q, s, a, b),\ x(p, q, s, a, b, p'_y, q'_z)$
$x(p, a), x(q, b), x(p), x(q), x(p, p'_y), x(q, q'_z)$
$x(a|p), x(b|q), x(p'|p, y), x(q'|q, z)\ \forall\ p, q, s, a, b, y, z, p', q'$

Maximize: $$\sum_{p,q,s}\sum_{a,b} R(s, a, b)x(p, q, s, a, b) \qquad (5)$$

Subject to:

$$\sum_{a,b} x(p', q', s', a, b) = \eta_0(p', q', s') + \gamma \sum_{p,q,s}\sum_{a,b,y,z} O(yz \mid s', a, b)$$
$$P(s' \mid s, a, b)x(p, q, s, a, b, \tilde{p}_y = p', \tilde{q}_z = q')\forall(p', q', s') \qquad (6)$$
$$x(p, q, s, a, b) = \sum_{p',q'} x(p, q, s, a, b, p'_y, q'_z)\forall(p, q, s, a, b, y, z) \qquad (7)$$
$$x(p, a) = \sum_{q,s,b} x(p, q, s, a, b)\forall(p, a) \qquad (8)$$
$$x(q, b) = \sum_{p,s,a} x(p, q, s, a, b)\forall(q, b) \qquad (9)$$
$$x(p) = \sum_{a} x(p, a)\forall p \qquad (10)$$
$$x(q) = \sum_{b} x(q, b)\forall q \qquad (11)$$
$$x(p, p'_y) = \sum_{q,s,a,b,q'} x(p, q, s, a, b, p'_y, q'_z)\forall(p, y, z, p') \qquad (12)$$
$$x(q, q'_z) = \sum_{p,s,a,b,p'} x(p, q, s, a, b, p'_y, q'_z)\forall(q, y, z, q') \qquad (13)$$
$$x(p) - x(p, a) \le \frac{1 - x(a|p)}{1-\gamma}\forall(p, a) \qquad (14)$$
$$x(q) - x(q, b) \le \frac{1 - x(b|q)}{1-\gamma}\forall(q, b) \qquad (15)$$
$$x(p) - x(p, p'_y) \le \frac{1 - x(p'|p, y)}{1-\gamma}\forall(p, y, p') \qquad (16)$$
$$x(q) - x(q, q'_z) \le \frac{1 - x(q'|q, z)}{1-\gamma}\forall(q, z, q') \qquad (17)$$
$$\sum_{a} x(a|p) = 1\ \forall p;\ \sum_{b} x(b|q) = 1\ \forall q \qquad (18)$$
$$\sum_{p'} x(p'|p, y) = 1\ \forall(p, y)\ \ ; \sum_{q'} x(q'|q, z) = 1\ \forall(q, z) \qquad (19)$$
$$x(a|p) \in \{0, 1\}, x(b|q) \in \{0, 1\}, \qquad (20)$$
$$x(p'|p, y) \in \{0, 1\}, x(q'|q, z) \in \{0, 1\} \qquad (21)$$

Table 1: MIP for the CP-MDP of a 2-agent Dec-POMDP

We first describe the CP-MDP for a Dec-POMDP. The *state space* is the cross-product of the joint-controller state and the environment state $(p, q, s) \in \mathcal{P} \times \mathcal{Q} \times S$. The *action space* has two components. The first consists of the actions a and b of both agents. The controller transition parameters $\lambda(\cdot)$ for both agents are also included in the action space, since our goal in solving the CP-MDP is to find the action mapping for each node as well as decide which node to transition to for each possible observation. This is achieved by creating a random variable $\tilde{p}_{y_i}$ for each observation y_i of agent 1 and variable $\tilde{q}_{z_i}$ for each observation z_i of agent 2. The domain of $\tilde{p}_{y_i}$ (resp. $\tilde{q}_{z_i}$) is the set of all controller nodes $\mathcal{P}$ (resp. $\mathcal{Q}$) to which control can transfer upon receiving observation y_i (resp. z_i). In our formulation, the observation y_i is not a variable, rather the *variable* is which node the control transitions to *given* y_i. These variables encode the transition function $\lambda(p'|p, y)$. The joint-action for the underlying cross-product MDP is therefore $\langle a, b, \tilde{p}_{y_1}, \ldots, \tilde{p}_{y_k}, \tilde{q}_{z_1}, \ldots, \tilde{q}_{z_k}\rangle$, or in short $\langle a, b, \langle\tilde{p}_y\rangle, \langle\tilde{q}_z\rangle\rangle$. The *transition function* for the CP-MDP is:

$$P(p', q', s' \,|\, p, q, s, a, b, \langle\tilde{p}_y\rangle, \langle\tilde{q}_z\rangle) = \sum_{y_r \in Y, z_l \in Z} P(p' \,|\, p, \langle\tilde{p}_y\rangle, y_r)$$
$$P(q' \,|\, q, \langle\tilde{q}_z\rangle, z_l)O(y_r z_l \,|\, s', a, b)P(s' \,|\, s, a, b) \qquad (22)$$

where the distribution $P(p'|p, \langle\tilde{p}_y\rangle, y_r)$ is defined as:

$$P(p' \,|\, p, \langle\tilde{p}_y\rangle, y_r) = \begin{cases} 1 & \text{if } \tilde{p}_{y_r} = p' \\ 0 & \text{otherwise} \end{cases} \qquad (23)$$

$P(q'|q, \langle\tilde{q}_z\rangle, z_l)$ is defined analogously. Intuitively, these distributions encode the fact that a complete assignment to variables $\langle\tilde{p}_y\rangle$ and $\langle\tilde{q}_z\rangle$ defines a deterministic controller transition function. We now extend the MDP dual formulation to CP-MDP. Analogous to the $x(s, a)$ variables in the MDP case, we define variables $x(p, q, s, a, b, \langle\tilde{p}_y\rangle, \langle\tilde{q}_z\rangle)$ using the state and joint-action of the CP-MDP.

Definition 2. *The variable $x(p, q, s, a, b, \langle\tilde{p}_y\rangle, \langle\tilde{q}_z\rangle)$ is defined as:*

$$\sum_{j \in S, k \in \mathcal{P}, l \in \mathcal{Q}} \eta_0(j, k, l) \sum_{t=1}^{\infty} \gamma^{t-1}$$
$$P_t(p, q, s, a, b, \langle\tilde{p}_y\rangle, \langle\tilde{q}_z\rangle \mid s_1 = j, p_1 = k, q_1 = l) \qquad (24)$$

where $\eta_0(\cdot)$ is the (given) initial distribution over world states and the joint-controller nodes. Next, we adapt the MDP flow constraint (3) to CP-MDPs. Note that the variables $x(p, q, s, a, b, \langle\tilde{p}_y\rangle, \langle\tilde{q}_z\rangle)$ can be considered a distribution. To reduce the exponential number of these variables, we exploit the structure of the DBN in Fig. 1.
The LHS of Eq. 3 can be written for a CP-MDP as:

$$\sum_{a,b} x(p', q', s', a, b) = \sum_{a,b,\langle\tilde{p}_y\rangle,\langle\tilde{q}_z\rangle} x(p', q', s', a, b, \langle\tilde{p}_y\rangle, \langle\tilde{q}_z\rangle) \quad (25)$$

The RHS for the flow constraint is:

$$\eta_0(p', q', s') + \gamma \sum_{p,q,s,a,b,\langle\tilde{p}_y\rangle,\langle\tilde{q}_z\rangle} P(p', q', s' | p, q, s, a, b, \langle\tilde{p}_y\rangle, \langle\tilde{q}_z\rangle)$$
$$x(p, q, s, a, b, \langle\tilde{p}_y\rangle, \langle\tilde{q}_z\rangle) \qquad (26)$$

We simplify the inner summation term in the above equation by further marginalizing over joint-observations $\langle y, z\rangle$:

$$\sum_{p,q,s,a,b,y_r,z_l,\langle\tilde{p}_y\rangle,\langle\tilde{q}_z\rangle} P(p' | p, y_r, \langle\tilde{p}_y\rangle) P(q' | q, z_l, \langle\tilde{q}_z\rangle)$$
$$P(s', y_r, z_l | p, q, s, a, b, \langle\tilde{p}_y\rangle, \langle\tilde{q}_z\rangle) x(p, q, s, a, b, \langle\tilde{p}_y\rangle, \langle\tilde{q}_z\rangle) \quad (27)$$

We note that variables (s', y_r, z_l) in the DBN of Fig. 1 are independent of $(p, q, \langle\tilde{p}_y\rangle, \langle\tilde{q}_z\rangle)$ given (s, a, b). Also, distributions $P(p' | p, y_r, \langle\tilde{p}_y\rangle)$ and $P(q' | q, z_l, \langle\tilde{q}_z\rangle)$ are deterministic and are 1 only when $\tilde{p}_{y_r} = p'$ and $\tilde{q}_{z_l} = q'$. Therefore, we can simplify the quantity in (27) to:

$$\sum_{p,q,s,a,b,y_r,z_l} P(s', y_r, z_l | s, a, b)$$
$$\sum_{\langle\tilde{p}_y\rangle \backslash p'_{y_r}, \langle\tilde{q}_z\rangle \backslash q'_{z_l}} x(p, q, s, a, b, p'_{y_r}, q'_{z_l}, \langle\tilde{p}_y\rangle \backslash p'_{y_r}, \langle\tilde{q}_z\rangle \backslash q'_{z_l}) \quad (28)$$

Analogous to the marginalization of probability distributions, we can simplify the above expression to:

$$\sum_{p,q,s,a,b,y_r,z_l} P(s', y_r, z_l | s, a, b) x(p, q, s, a, b, p'_{y_r}, q'_{z_l}) \qquad (29)$$

To summarize, Eq. (25) forms the basis for the LHS of flow constraint (6) in Table 1. Eqs. (26) and (29) form the basis for the RHS of this flow constraint. Notice that using proper marginalization, the number of $x(\cdot)$ variables used in constraint (6) is polynomial in problem parameters; we do not use exponentially many $x(p, q, s, a, b, \langle\tilde{p}_y\rangle, \langle\tilde{q}_z\rangle)$ variables.

Producing a Valid Dec-POMDP Policy

Note that the flow constraint (6) in Table 1 alone will not result in a *valid policy* as different FSC parameters may depend on the unobservable world state. In a valid Dec-POMDP policy, each agent's controller parameters depend only on its local observables, which in our case are the last local observation received and the agent's local controller state. Hence, if a deterministic policy transitions to node $p^\star$ given current controller state p and observation y ($\lambda(p^\star | p, y) = 1$), then enforcing the occupancy measure condition $x(p) = x(p, p^\star_y)$ results in a valid policy. Intuitively, this condition says that whenever the control is in state p ($= x(p)$), then *given* observation y, the action is to transition to $p^\star$ ($= x(p, p^\star_y)$). This condition *prevents* dependence on other agent's parameters or the unobservable world-state.

Theorem 1. *The MIP constraint* (16) *along with the integrality constraints of* (21) *guarantee a valid node mapping in a partially observable setting. That is,*

$$x(p, p^\star_y) = x(p) \;\; \textit{iff}\; x(p^\star | p, y) = 1, \; \forall y \in Y, \forall p \in \mathcal{P} \quad (30)$$
$$x(p, p'_y) = 0 \qquad \forall p' \in \mathcal{P} \backslash p^\star, \; \forall y \in Y, \forall p \in \mathcal{P} \quad (31)$$

Proof. We know from constraint (12):

$$x(p, p'_y) = \sum_{qsabq'} x(p, q, s, a, b, p'_y, q'_z) \qquad (32)$$

$$\sum_{p'} x(p, p'_y) = \sum_{p'} \sum_{qsabq'} x(p, q, s, a, b, p'_y, q'_z) \qquad (33)$$

Using constraint (7), we can simplify the RHS as follows:

$$\sum_{p'} x(p, p'_y) = \sum_{qsab} x(p, q, s, a, b) \qquad (34)$$

Using constraint (8), we further simplify the RHS:

$$\sum_{p'} x(p, p'_y) = \sum_{a} x(p, a) \qquad (35)$$

Using constraint (10), we get:

$$\sum_{p'} x(p, p'_y) = x(p) \qquad (36)$$

Using the above relation, we deduce that:

$$x(p) \geq x(p, p'_y) \; \forall y, p', p \qquad (37)$$

Let us now focus on constraint (16). Using the normalization constraint (19) and integrality constraints (21), we know that for each (p, y) pair, there will be exactly one $p^\star$ such that $x(p^\star | p, y) = 1$. Constraint (16) for this $p^\star$ is:

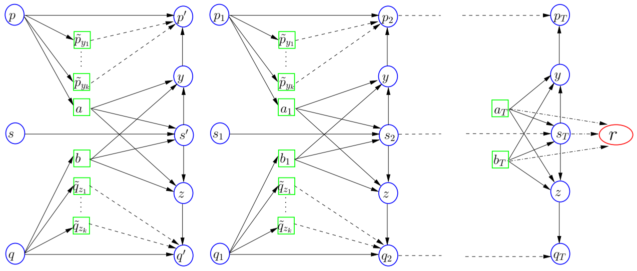
$$x(p) - x(p, p^\star_y) \leq 0 \qquad (38)$$
$$x(p) \leq x(p, p^\star_y) \qquad (39)$$

Using relations (37) and (39), we get that:

$$x(p) = x(p, p^\star_y) \qquad (40)$$

Using Eq. (36), we also deduce that $x(p, p'_y) = 0 \; \forall p' \in \mathcal{P} \backslash p^\star$. This proves the theorem statement. Finally, constraint (16) also holds for all $p' \in \mathcal{P} \backslash p^\star$, resulting in the valid inequality $x(p) \leq \frac{1}{1-\gamma}$. □

We can similarly show that $x(p) = x(p, a^\star)$ resulting in a valid action selection policy that depends only on controller state. Assuming m nodes for each agent's controller, the MIP has $O(m^4 |S||A|^2 |Y|^2)$ variables, $2 \times (m|A| + m^2|Y|)$ of which are binary, and $O(m^2 |S||A|^2 ||Y|^2)$ constraints. Binary variables, whose number greatly influences the complexity of solving MIPs, correspond only to the policy parameters π and λ. Our MIP for Dec-POMDPs is therefore an efficient encoding, containing a minimal number of binary variables.



(a) (b)

Figure 2: (a) two time-slice DBN for a fixed policy, and (b) the T-step DBN for the reward obtained at time step T

Optimizing Stochastic Controllers

We have shown how the dual perspective results in a mixed-integer program to optimize *deterministic* controllers. In this section, we show that a similar dual perspective can also be used to optimize *stochastic* controllers within the expectation maximization (EM) framework (Dempster, Laird, and Rubin 1977). The well known EM algorithm has been extended to optimize stochastic Dec-POMDP controllers. Several variants of EM have been developed for different variations of the Dec-POMDP model from two-agent general Dec-POMDPs (Kumar and Zilberstein 2010a; Pajarinen and Peltonen 2013) to larger factored Dec-POMDPs (Pajarinen and Peltonen 2011a; Kumar, Zilberstein, and Toussaint 2015). The core computational challenge in the E-step of the EM algorithm is to perform forward-backward message passing on the DBN with the structure shown in Fig. 2(b), similar to message passing in hidden Markov models.

The message passing used in existing EM implementations is computationally challenging and inaccurate as the number of time steps for message propagation is not known a priori. Based on the discount factor γ, a suitable cutoff time is computed on-the-fly when γ^T is small enough. Such a cutoff time may be large depending on the value of γ, and may cause numerical inaccuracies. Another drawback is that for every different variation of Dec-POMDPs (such as transition/observation independence or factored models), the forward-backward message passing needs to be analytically derived to reflect the changes in the DBN structure.

To avoid these issues, we show that the quantities computed by the message-passing approaches can be extracted directly by solving a simple *linear program* (LP) based on the dual framework. Consequently, the entire EM algorithm can be easily implemented by solving a series of LPs, one for each iteration. Using this LP also removes the dependence of EM's runtime and numerical accuracy on the discount factor γ. Furthermore, modifying the DBN structure changes some constraints in the LP, eliminating the need to re-derive the message-passing update equations.

In each EM iteration, for a fixed policy for each agent (computed from previous iteration), message passing takes place on the DBN shown in Fig. 2(a). This DBN is analogous to the CP-MDP DBN in Fig. 1, expect for the fact that given a fixed decentralized policy, we can remove the incoming influence from the underlying state s and the other agent's controller q to agent 1's policy (and vice versa). We focus on updating the action parameter $\pi^\star(a|p)$ for agent 1 (node transition update is similar). EM's update is therefore:

$$\pi^\star_{ap} = \frac{\pi_{ap}}{C_p} \sum_{qs} \hat{\alpha}(p,q,s) \Big[\sum_b \hat{R}_{sab}\pi_{bq} + \frac{\gamma}{1-\gamma} \sum_{p'q's'y'z'} \hat{\beta}(p',q',s') \\ \lambda_{p'py'}\lambda_{q'qz'} \sum_b O_{y'z's'ab}\pi_{bq}P_{s'sab} \Big] \quad (41)$$

where $\hat{\alpha}(p,q,s)$ and $\hat{\beta}(p,q,s)$ are computed using message passing; $\pi^\star(\cdot)$ denotes the new estimate of the action parameter. For brevity, different parameters (e.g., $\pi(a|p)$) are denoted using shorthand (as in π_{ap}). For a fixed joint-policy $\boldsymbol{\theta}$, using the occupancy measure in Definition 2, let

$$x(p,q,s) = \sum_{a,b,\langle\tilde{p}_y\rangle,\langle\tilde{q}_z\rangle} x(p,q,s,a,b,\langle\tilde{p}_y\rangle,\langle\tilde{q}_z\rangle) \quad (42)$$

For the sake of readability we removed the dependence of $x(\cdot)$ on the joint-policy $\boldsymbol{\theta}$. We show the following relation.

Theorem 2. *For a fixed joint-policy $\boldsymbol{\theta}$, the following holds:*

$$\hat{\alpha}(p,q,s) = x(p,q,s).$$

Proof. The quantity $\hat{\alpha}(p,q,s)$ computed using message passing over the Dec-POMDP DBN is defined as (Kumar, Zilberstein, and Toussaint 2015):

$$\hat{\alpha}(p,q,s) = \sum_{t=1}^{\infty} \gamma^{t-1} P_t(p,q,s) \quad (43)$$

where $P_t(p,q,s)$ denotes the probability that at time step t, the joint controller and world state is $\langle p,q,s\rangle$ as per the given policy $\boldsymbol{\theta}$. Using Eq. (42) and Definition 2, we have:

$$x(p,q,s) = \sum_{t=1}^{\infty} \gamma^{t-1} \sum_{a,b,\langle\tilde{p}_y\rangle,\langle\tilde{q}_z\rangle} \sum_{j\in S,k\in\mathcal{P},l\in\mathcal{Q}} \eta_0(j,k,l) \\ P_t(p,q,s,a,b,\langle\tilde{p}_y\rangle,\langle\tilde{q}_z\rangle \mid s_1 = j, p_1 = k, q_1 = l) \quad (44)$$

$$= \sum_{t=1}^{\infty} \gamma^{t-1} \sum_{a,b,\langle\tilde{p}_y\rangle,\langle\tilde{q}_z\rangle} P_t(p,q,s,a,b,\langle\tilde{p}_y\rangle,\langle\tilde{q}_z\rangle) \quad (45)$$

where we marginalize over the initial joint-controller and the world state $\langle k,l,j\rangle$ in Eq. (44) to get (45). We can further marginalize the probability in Eq. (45) to get:

$$x(p,q,s) = \sum_{t=1}^{\infty} \gamma^{t-1} P_t(p,q,s) \quad (46)$$

The theorem follows from Eqs. (43) and (46). □

The above result shows that the quantity $\hat{\alpha}$ is the same as the occupancy measures $x(\cdot)$ that are computed using the dual framework. Before we describe the connections of the second quantity $\hat{\beta}(p,q,s)$ with the dual framework, we show in Table 2 the LP that computes the $x(p,q,s)$ measures for a given joint-policy $\boldsymbol{\theta}$.

It is also instructive to analyze the dual of the LP in Table 2, shown in Table 3. Notice that the LP in Table 3 simply evaluates the given joint-policy $\boldsymbol{\theta}$. A key observation

Variables: $x(p,q,s)\ \forall\ p,q,s$

Maximize:
$$\sum_{p,q,s}\sum_{a,b} R(s,a,b)x(p,q,s)\pi(a|p)\pi(b|q) \tag{47}$$

Subject to:
$$x(p',q',s') = \eta_0(p',q',s') + \gamma \sum_{p,q,s}\sum_{a,b,y,z} O(yz \mid s',a,b)P(s'|s,a,b)$$
$$\pi(a|p)\pi(b|q)\lambda(p'|p,y)\lambda(q'|q,z)x(p,q,s)\ \ \forall p',q',s' \tag{48}$$

$$x(p,q,s) \geq 0 \quad \forall\ p,q,s \tag{49}$$

Table 2: LP to compute the $x(p,q,s)$ variables. Controller parameters $\theta = \langle \pi, \lambda, \nu \rangle$ for each agent are fixed.

Variables: $V(p,q,s)\ \forall\ p,q,s$

Minimize:
$$\sum_{p,q,s} \eta_0(p,q,s)V(p,q,s) \tag{50}$$

Subject to:
$$V(p,q,s) \geq \sum_{a,b} \pi(a \mid p)\pi(b \mid q)\Big[R(s,a,b) + \gamma \sum_{s'} P(s' \mid s,a,b)$$
$$\sum_{y,z} O(yz \mid s',a,b) \sum_{p'q'} \lambda(p' \mid p,y)\lambda(q' \mid q,z)V(p',q',s')\Big] \ \ \forall p,q,s$$

Table 3: The dual of the LP in table 2.

is that when the LP of Table 2 is solved, then most standard LP solvers would also return the optimal dual variables $V(p,q,s)$. Therefore, in practice we do not need to solve the LP in Table 3 to get the joint-policy values $V(p,q,s)$. We next show the connections of $V(p,q,s)$ and the quantity $\hat{\beta}(p,q,s)$ required by the EM algorithm.

Theorem 3. *For a fixed joint-policy* $\boldsymbol{\theta}$*, let* $V(p,q,s)$ *denote the value of starting the joint-controller in nodes* $\langle p,q \rangle$ *at state s, then we have:* $\hat{\beta}(p,q,s) \propto V(p,q,s)$.

Proof. The value $V(p,q,s)$ is defined as:

$$V(p,q,s) = \sum_{T=1}^{\infty} \gamma^{T-1} \mathbb{E}\big[R(s_T,a_T,b_T) | \langle p_1,q_1,s_1 \rangle = \langle p,q,s \rangle\big]$$

where $R(\cdot)$ is the reward obtained at time step T given the initial controller and the world state is $\langle p,q,s \rangle$.

The quantity $\hat{\beta}(p,q,s)$ required by EM updates is defined recursively using backward messages on the DBN with the structure as in Fig. 2(b). Consider a T-step DBN shown in Fig. 2(b). This T-step DBN shows the evolution of joint-controller state and the world state for T time steps. At the end of T-time steps, a binary reward variable r is introduced whose conditional probability simulates normalized rewards of the Dec-POMDP model. That is, we have:

$$P(r=1|s_T=s, a_T=a, b_T=b) = \frac{R(s,a,b) - R_{min}}{R_{max} - R_{min}} \tag{51}$$

For a T-step DBN (as in figure 2(b)), Kumar, Zilberstein, and Toussaint (2015) show that:

$$P(r=1;T) = \frac{\mathbb{E}\big[R(s_T,a_T,b_T)\big] - R_{min}}{R_{max} - R_{min}} \tag{52}$$

Algorithm 1: EM For Stochastic FSC Optimization

```
Initialize: $\boldsymbol{\theta}$ ←RandomInit
it ← 0
while $it \leq$ Max-Iter do
    solveDualLP($\boldsymbol{\theta}$)
    Set $V(p,q,s) \leftarrow$ dual variables from LP solution $\forall p,q,s$

    $\hat{\alpha}(p,q,s) \leftarrow x(p,q,s)\ \forall p,q,s$
    $\hat{\beta}(p,q,s) = \frac{V(p,q,s) - \frac{1}{1-\gamma}R_{min}}{R_{max} - R_{min}}\ \forall p,q,s$
    Update $\pi^\star(a|p)$ using (41) $\forall a,p$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}^\star$
    it ← it + 1
```

The quantity $\hat{\beta}(p,q,s)$ required by EM updates is (Kumar, Zilberstein, and Toussaint 2015):

$$\hat{\beta}(p,q,s) = \sum_{\tau=1}^{\infty} \gamma^{\tau-1} \beta_\tau(p,q,s) \tag{53}$$

$$\beta_\tau(p,q,s) = P(r=1|p_{T-\tau+1}=p, q_{T-\tau+1}=q, s_{T-\tau+1}=s, T) \tag{54}$$

Semantically, $\beta_\tau(p,q,s)$ denotes the probability of reward variable $r=1$ when the current joint-controller and world state is $\langle p,q,s \rangle$ and there are exactly τ time steps to go. Kumar *et al.* (2015) showed that the value of β_τ is the same for each DBN of length $T \geq \tau$ as all such DBNs share the same tail. If we substitute $T=\tau$ in Eq. (54), we get:

$$\beta_\tau(p,q,s) = P(r=1|p_1=p, q_1=q, s_1=s, \tau) \tag{55}$$

Using the relation in (52), we get:

$$\beta_\tau(p,q,s) = \frac{\mathbb{E}\big[R(s_\tau,a_\tau,b_\tau)|\langle p_1,q_1,s_1 \rangle = \langle p,q,s \rangle\big] - R_{min}}{R_{max} - R_{min}}$$

That is, $\beta_\tau(p,q,s)$ denotes the normalized expected reward obtained after time step τ. Substituting this result back into the definition of $\hat{\beta}$ in (53), we get:

$$\hat{\beta}(p,q,s) = \frac{V(p,q,s) - \frac{1}{1-\gamma}R_{min}}{R_{max} - R_{min}} \tag{56}$$

which concludes the proof. □

To summarize, using the dual LP shown in Table 2, the EM algorithm can be implemented via a sequence of LPs, as outlined in Algorithm 1. The LP in Table 2 is solved for previous iteration's joint-policy $\boldsymbol{\theta}$. Using the output of this LP, $\hat{\alpha}$ and $\hat{\beta}$ can be extracted. This LP forms the E-step of the EM algorithm. New estimates of policy parameters (such as $\pi^\star(\cdot)$) can be found by the standard EM updates.

Iteratively Increasing Controller Size

A key practical issue with optimizing controllers is that it is often hard to determine the "right" size of each controller. Larger controllers can potentially represent better policies, but they are hard to optimize within the MIP framework. Recently, a number of techniques have been proposed for POMDPs that start with optimizing a small controller and iteratively increase its size (Grzes and Poupart 2015;

Kumar and Zilberstein 2015). We use the notion of history-based controllers (HBCs) introduced by Kumar and Zilberstein (2015) to iteratively add nodes. In a generic controller, the optimization problem determines both what aspect of the observation history each node should memorize and what action to take, which are hard to optimize simultaneously. HBCs associate semantic interpretation (e.g., memorizing the last action, observation or both) with each node, thereby providing highly interpretable controllers and keeping the size of the dual MIP small to aid the optimization engine. We use the following definition of HBCs.

Definition 3 (Kumar and Zilberstein 2015)**.** *A HBC consists of a set of nodes $\{n_s\} \cup \mathcal{N}$, where n_s denotes a unique start node. The set $\mathcal{N}$ is partitioned according to observations: $\mathcal{N} = \cup_{y \in Y} N_y$, where N_y represents the set of nodes n_y encoding the fact that the last observation received was y. The deterministic action and node mappings are defined as: $\pi : \{n_s\} \cup \mathcal{N} \to A$, and $\lambda_y : (\{n_s\} \cup \mathcal{N}) \times \{y\} \to N_y \ \forall y \in Y$.*

Intuitively, the node mapping λ_y for a HBC determines the target set N_y based on the last received observation y. Fig. 3(a) shows an example of a HBC with three observations. Incoming edges to each node in N_{y_i} are labeled with the corresponding observation y_i. Optimizing a HBC-based policy is NP-Hard since a reactive controller (which is NP-Hard) is an instance of HBC with $|N_y| = 1 \ \forall y$. Nonetheless, using HBC aids the MIP solver by reducing the number of binary variables; if the maximum controller size for any agent is m, then the node mapping for a standard FSC formulation requires $O(m^2|Y|)$ binary variables, whereas for HBC it requires only $O(m'm|Y|)$, where m' is the largest size of any set N_y. In all our experiments, $m' \ll m$, significantly reducing the complexity of solving the dual MIP.

We now detail the procedure for iteratively optimizing decentralized FSCs, extending the approach of Kumar and Zilberstein (2015). We start with optimizing a default-size controller (e.g., a reactive controller). Then, using the entropy-based heuristic described below, we choose a node n_y from the set of nodes for agents 1 and 2 ($n_y \in \mathcal{P} \cup \mathcal{Q}$). Next we *split* the node n_y to create a single copy n'_y. That is, if node n_{y_2} in Fig. 3(a) is chosen for splitting, we add the node n'_{y_2} to the original controller $\mathcal{C}$ resulting in the larger controller $\mathcal{C}'$ as in Fig. 3(b). Instead of optimizing the larger controller $\mathcal{C}'$ using the dual MIP from scratch, we use a warm-start strategy. Intuitively, only parameters that are affected by the addition of the new node n'_{y_2} are made variables to be re-optimized using the dual MIP. Such parameters are:

- The action mapping π for nodes n_{y_2} and n'_{y_2}
- All incoming transitions to the node n_{y_2} in the original controller $\mathcal{C}$ become variable in the new controller with two choices n_{y_2} and n'_{y_2}. This is shown using dotted arrows from the node n_{y_1} to n_{y_2} and n'_{y_2} in Fig. 3(b)
- The node mapping λ for both nodes n_{y_2} and n'_{y_2} becomes variable to be re-optimized, as shown using the outgoing dotted arrows from n_{y_2} and n'_{y_2} in Fig. 3(b).

Once the structure of C' is set using the above rules, then we re-optimize C' using the MIP in Table 1. We perform such iterative optimization for a maximum number of iterations or until the new controller $\mathcal{C}'$ encodes the same policy as the previous controller $\mathcal{C}$ for every possible node split.

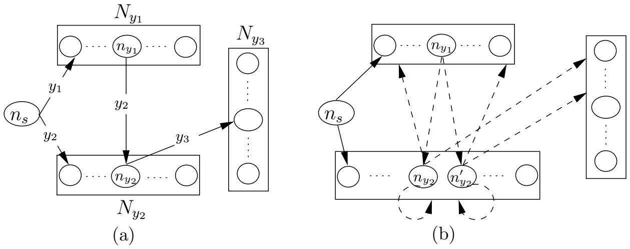


Figure 3: Part (a) shows an example of a HBC with three observations. Part (b) shows the warm-start strategy with dotted lines corresponding to variables to be optimized.

Entropy-based Heuristic The intuition behind choosing a node to split is that better policies are encoded by controllers whose states are strongly indicative of the underlying world state. We therefore quantify the uncertainty about the world state associated with every HBC node for each agent's controller. To measure uncertainty, we use the $x(\cdot)$ variables computed by solving the dual MIP:

$$x(p,q,s) = \sum_{a,b} x(p,q,s,a,b) \text{ , and } x(s|p,q) = \frac{x(p,q,s)}{x(p,q)}$$

$$H(p) = -\sum_{q \in \mathcal{Q}} x(q) \sum_{s} x(s|p,q) \ln x(s|p,q)$$

$H(\cdot)$ is analogous to *conditional entropy* (Kumar and Zilberstein 2015). The higher a node's entropy, the more "confused" the agent is about the world state and the greater the need to split it and re-optimize the more refined policy. A node corresponding to a unique world state has 0 entropy. Another refinement we found useful for a more focused node addition is to give higher splitting priority to nodes where the HBC spends more time. We incorporate this by defining the notion of weighted entropy as $H_w(p) = x(p) \cdot H(p)$. Therefore, in each iteration we compute $H_w(p)$ for each node of both agents, and choose the one with the highest weighted entropy $H_w(p)$ to split. Empirically, we found that this strategy provided excellent empirical performance. Furthermore, the weighted entropies can be computed very efficiently as a byproduct of the dual MIP with minimal extra computation.

Experiments

We compare our dual MIP with node addition approach ('dualMIP') to the state-of-the-art controller optimization algorithms, including the periodic FSC optimization ('Peri') (Pajarinen and Peltonen 2011b), EM applied to periodic FSC structure (Pajarinen and Peltonen 2011b; Kumar and Zilberstein 2010a) and the mealy NLP approach (Amato, Bonet, and Zilberstein 2010). We also compare against an optimal branch-and-bound solver to optimize FSCs (Amato and Zilberstein 2008) and a non-FSC-based approach (Dibangoye, Buffet, and Charpillet 2014). We used

	dualMIP		Peri		Peri-EM		NLP	
Problem	Qual.	Time	Qual.	Time	Qual.	Time	Qual.	Time
BroadcastChannel $\|S\|$:4, $\|A\|^i$:2, $\|Y\|^i$:2 FB-HSVI = 9.2	9.1	0.05	–	–	–	–	9.1	1
Dec-Tiger $\|S\|$:2, $\|A\|^i$:3, $\|Y\|^i$:2 FB-HSVI = 13.4	13.4	4.2	13.4	202	13.4	6540	-1.49	1
RecyclingRobots $\|S\|$:4, $\|A\|^i$:3, $\|Y\|^i$:2 FB-HSVI = 31.9	31.9	1.1	31.8	77	31.8	272	31.9	1
Grid3×3 $\|S\|$:81, $\|A\|^i$:5, $\|Y\|^i$:9 FB-HSVI = 5.8	5.8	4.4	4.64	9714	–	–	4.9	270
BoxPushing $\|S\|$:100, $\|A\|^i$:4, $\|Y\|^i$:5 FB-HSVI = 224	181.2	6.2	148.6	5675	106.6	7164	143.1	1
MarsRover $\|S\|$:256, $\|A\|^i$:6, $\|Y\|^i$:8 FB-HSVI = 26.9	23.8	20.2	24.1	6088	18.1	7132	19.6	396
Wireless $\|S\|$:64, $\|A\|^i$:2, $\|Y\|^i$:6 FB-HSVI = -144	-184.1	18.2	-181.2	6492	-218	3557	-296	270

Table 4: Quality and runtime (sec) of dualMIP against different controller optimization approaches. '$-$' indicates result unavailable in the literature. Time comparisons are approximate as other approaches may have used different platforms.

CPLEX 12.6 as the MIP solver on a 2.8GHz machine with 4GB RAM. All problem domains are publicly available[1].

Our dualMIP algorithm first optimizes a default reactive controller and then adds nodes according to our entropy-based heuristics. Table 4 shows time and quality results, including the solution quality provided by FB-HSVI (Dibangoye, Buffet, and Charpillet 2014), which converts the infinite-horizon problem to a finite-horizon problem and uses finite-horizon planning to generate a non-stationary policy. Thanks to reward discounting, this provides a good approximation, given $\gamma = 0.9$ for most problems. As Table 4 shows, dualMIP provides similar or better quality than previous FSC approaches and is multiple orders-of-magnitude faster than previous best FSC optimization approaches 'Peri' and 'Peri-EM'. For all the instances, dualMIP terminates within 30 sec, which is a significant improvement over previous controller optimization approaches. For example, for 'BoxPushing', dualMIP provides a higher quality ($\approx$181) than 'Peri' ($\approx$148) with a significant speedup (6 sec for dual MIP vs. 5675 sec for 'Peri'). The NLP approach is faster than 'Peri' for some instances, but sometimes provides drastically lower quality (e.g., for 'Dec-Tiger' and 'Wireless').

Controller size One notable drawback of previous controller optimization approaches is that an initial controller size must be specified for each problem (required by 'Peri', 'Peri-EM' and 'NLP'). Additionally, it is often hard to associate any semantic information with controller nodes. Our dualMIP approach along with HBC-based policy addresses both of these issues. Table 5 shows the size of the reactive controller and the final controller size after node addition by dualMIP for different problems. Starred ($\star$) entries denote problems where dualMIP added nodes; for others, dualMIP terminates without adding nodes as node additions led to duplicates of the reactive controller. Our approach was able to optimize reactive controllers provably optimally for all the benchmarks. Table 5 shows that for 4 out of 7 benchmarks, optimal reactive controller provided similar quality to previous approaches. For the three problems where dualMIP added nodes, the final controller size was still very compact. In contrast to the compact HBCs optimized by dualMIP, 'Peri' uses a minimum controller size of $(180, 180)$ for 'RecyclingRobots' and a maximum of $(1500, 1500)$ for 'Wireless'. These results show that the HBC-based policy optimized using dualMIP is both highly interpretable and compact as opposed to the controllers produced by other FSC optimization approaches.

Problem	Reactive	Final	Problem	Reactive	Final
Broadcast	(3, 3)	R	Grid3×3	(10, 10)	R
Recycling	(3, 3)	R	Wireless*	(7, 7)	(8,7)
Dec-tiger*	(3, 3)	(7, 7)	BoxPushing*	(6, 6)	(7,8)
			MarsRover	(9, 9)	R

Table 5: No. of controller nodes per agent; dualMIP has $(|Y|+1)$ nodes in the initial reactive controller before adding nodes; 'R' denotes the final controller was also reactive

For two benchmarks ('BoxPushing' and 'Wireless'), the non-FSC based approach of Dibangoye, Buffet, and Charpillet (2014) provides better quality than dualMIP. These settings motivate more insightful node addition strategies analogous to the ones developed for growing POMDP controllers. Nonetheless, our approach has an advantage over non-FSC based approaches as it provides highly compact controllers ideal in resource constrained settings, in contrast to occupancy measure (Dibangoye, Buffet, and Charpillet 2014) based policies that require maintaining distributions over the world-state and joint-observation histories, which may increase exponentially with the planning horizon.

Comparisons with other branch-and-bound solvers We also compared the dualMIP approach to a branch-and-bound search strategy to find optimal attribute-based controllers (Amato and Zilberstein 2008). These attributes are typically handcrafted. For 2 benchmarks (Dec tiger, BoxPushing), we used the same controller structure as in (Amato and Zilberstein 2008) and compared the time dualMIP takes to globally optimize these controllers against their approach. Our approach was more than an order-of-magnitude faster. For Dec tiger, dualMIP required 4 sec versus 127 sec required by Amato and Zilberstein's approach, similarly 5 sec. for BoxPushing (dualMIP) versus 4974 sec.

Conclusion

Finite state controllers offer a compact, simple-to-execute policy representation for infinite-horizon problems. Previ-

[1] http://rbr.cs.umass.edu/camato/decpomdp/

ous FSC optimization approaches for Dec-POMDPs suffer from lack of scalability and solution quality guarantees, large controller size, susceptibility to getting stuck in local optima and the need to fix the controller size a priori. By adopting the dual view previously used for MDPs, we present a MIP formulation for optimizing a FSC-based stationary policy for infinite-horizon Dec-POMDPs. We devise constraints that guarantee that the policy is executable in partially-observable decentralized settings. In addition, we show analytically that the dual perspective can further benefit existing approaches for optimizing stochastic controllers, such as the EM algorithm. We also propose an entropy-based node addition heuristic to iteratively refine the policy, thereby providing competitive solution qualities using compact controllers with up to an order-of-magnitude speedup.

Acknowledgments This work was supported in part by the School of Information Systems research center at Singapore Management University and the NSF grant IIS-1405550.

References

Amato, C., and Zilberstein, S. 2008. What's worth memorizing: Attribute-based planning for Dec-POMDPs. In *ICAPS Workshop on Multiagent Planning*.

Amato, C.; Bernstein, D. S.; and Zilberstein, S. 2010. Optimizing fixed-size stochastic controllers for POMDPs and decentralized POMDPs. *Autonomous Agents and Multi-Agent Systems* 21(3):293–320.

Amato, C.; Konidaris, G.; Anders, A.; Cruz, G.; How, J. P.; and Kaelbling, L. P. 2015. Policy search for multi-robot coordination under uncertainty. In *Robotics: Science and Systems XI*.

Amato, C.; Bonet, B.; and Zilberstein, S. 2010. Finite-state controllers based on Mealy machines for centralized and decentralized POMDPs. In *AAAI Conf. on Artificial Intelligence*, 1052–1058.

Aras, R., and Dutech, A. 2010. An investigation into mathematical programming for finite horizon decentralized POMDPs. *Journal of Artificial Intelligence Research* 37:329–396.

Becker, R.; Zilberstein, S.; Lesser, V.; and Goldman, C. V. 2004. Solving transition independent decentralized Markov decision processes. *Journal of Artificial Intelligence Research* 22:423–455.

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

Bernstein, D. S.; Amato, C.; Hansen, E. A.; and Zilberstein, S. 2009. Policy iteration for decentralized control of Markov decision processes. *Journal of Artificial Intelligence Research* 34:89–132.

Dempster, A. P.; Laird, N. M.; and Rubin, D. B. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical society, Series B* 39(1):1–38.

Dibangoye, J. S.; Amato, C.; Buffet, O.; and Charpillet, F. 2013. Optimally solving Dec-POMDPs as continuous-state MDPs. In *Int'l Joint Conf. on Artificial Intelligence*, 90–96.

Dibangoye, J. S.; Buffet, O.; and Charpillet, F. 2014. Error-bounded approximations for infinite-horizon discounted decentralized POMDPs. In *ECML PKDD*, 338–353.

Grzes, M., and Poupart, P. 2015. Incremental policy iteration with guaranteed escape from local optima in POMDP planning. In *Int'l Conf. on Autonomous Agents and Multiagent Systems*, 1249–1257.

Grzes, M.; Poupart, P.; Yang, X.; and Hoey, J. 2015. Energy efficient execution of POMDP policies. *IEEE Trans. on Cybernetics* 45(11):2484–2497.

Hansen, E. A. 1998. Solving POMDPs by searching in policy space. In *Uncertainty in Artificial Intelligence*, 211–219.

Hoey, J.; Yang, X.; Quintana, E.; and Favela, J. 2012. LaCasa: Location and context-aware safety assistant. In *Int'l Conf. on Pervasive Computing Technologies for Healthcare*, 171–174.

Kumar, A., and Zilberstein, S. 2010a. Anytime planning for decentralized POMDPs using expectation maximization. In *Uncertainty in Artificial Intelligence*, 294–301.

Kumar, A., and Zilberstein, S. 2010b. Point-based backup for decentralized POMDPs: Complexity and new algorithms. In *Int'l Conf. on Autonomous Agents and Multiagent Systems*, 1315–1322.

Kumar, A., and Zilberstein, S. 2015. History-based controller design and optimization for partially observable MDPs. In *Int'l Conf. on Automated Planning and Scheduling*, 156–164.

Kumar, A.; Zilberstein, S.; and Toussaint, M. 2015. Probabilistic inference techniques for scalable multiagent decision making. *Journal of Artificial Intelligence Research* 53:223–270.

Littman, M. 1994. Memoryless policies: Theoretical limitations and practical results. In *Simulation of Adaptive Behavior*, 238–245.

Mostafa, H., and Lesser, V. R. 2011. Compact mathematical programs for DEC-MDPs with structured agent interactions. In *Uncertainty in Artificial Intelligence*, 523–530.

Nair, R.; Varakantham, P.; Tambe, M.; and Yokoo, M. 2005. Networked distributed POMDPs: A synthesis of distributed constraint optimization and POMDPs. In *AAAI Conf. on Artificial Intelligence*, 133–139.

Oliehoek, F. A. 2013. Sufficient plan-time statistics for decentralized POMDPs. In *Int'l Joint Conf. on Artificial Intelligence*, 302–308.

Pajarinen, J., and Peltonen, J. 2011a. Efficient planning for factored infinite-horizon DEC-POMDPs. In *Int'l Joint Conf. on Artificial Intelligence*, 325–331.

Pajarinen, J., and Peltonen, J. 2011b. Periodic finite state controllers for efficient POMDP and DEC-POMDP planning. In *Advances in Neural Information Processing Systems*, 2636–2644.

Pajarinen, J., and Peltonen, J. 2013. Expectation maximization for average reward decentralized POMDPs. In *European Conf. on Machine Learning*, 129–144.

Pajarinen, J.; Hottinen, A.; and Peltonen, J. 2014. Optimizing spatial and temporal reuse in wireless networks by decentralized partially observable Markov decision processes. *IEEE Trans. on Mobile Computing* 13(4):866–879.

Pineau, J.; Gordon, G.; and Thrun, S. 2006. Anytime point-based approximations for large POMDPs. *Journal of Artificial Intelligence Research* 27:335–380.

Puterman, M. L. 1994. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. New York, NY, USA: John Wiley & Sons, Inc., 1st edition.

Seuken, S., and Zilberstein, S. 2007. Memory-bounded dynamic programming for DEC-POMDPs. In *Int'l Joint Conf. on Artificial Intelligences*, 2009–2015.

Witwicki, S. J., and Durfee, E. H. 2007. Commitment-driven distributed joint policy search. In *Int'l Conf. on Autonomous Agents and Multiagent Systems*, 480–487.