Chapter 7
Infinite-Horizon Planning Methods: Discounted Cumulative Reward

As discussed in Chapter 6, infinite-horizon planning methods focus on generating ε-optimal solutions or solutions with a fixed controller size. We first discuss policy iteration, which is an extension of dynamic programming for Dec-POMDPs to the infinite-horizon case and can produce ε-optimal solutions. We then describe some of the heuristic algorithms that generate fixed-size controllers. We describe these algorithms in terms of Moore controllers, but they can all use Mealy controllers (see Section 6.2) with minor extensions.

7.1 Policy Iteration

Policy iteration (PI) for Dec-POMDPs [Bernstein et al., 2009] is similar to the finite-horizon dynamic programming algorithm (Algorithm 4.1), but finite-state controllers are used as policy representations (like the policy iteration approach for POMDPs [Hansen, 1998]). In addition, there are two other main differences with the finite-horizon DP:

1. Instead of using a set of separately represented policies (i.e., policy trees), PI maintains a single controller for each agent and considers the value of beginning execution from any node of the controller. That is, starting at each (joint) node can be interpreted as an infinite-horizon (joint) policy and the set of policies can be considered to be the set of joint nodes. Pruning can then take place over nodes of these controllers to remove dominated policies.
2. Rather than initializing the iteration process with the (set of) one-stage-to-go policies as in DP, PI can start from any initial joint controller for each agent $m_0 = \langle m_{1,0}, \dots, m_{n,0} \rangle$. Exactly what this initial joint controller m_0 is does not matter: the controller is going to be subject to exhaustive backups, which will 'push' the initial controller to beyond the effective horizon (where the discount factor renders the value bounded by ε), as well as controller improvements.

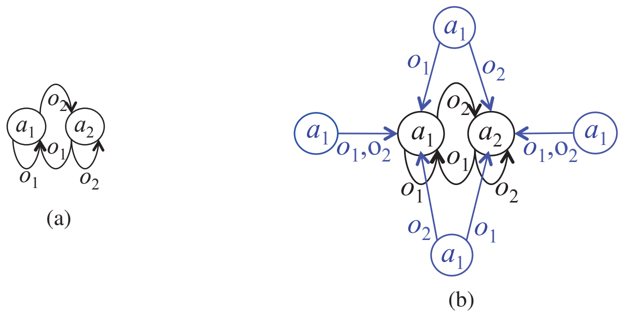

Fig. 7.1: A full backup for a single agent for action a_1 when starting with the controller in (a), resulting in the controller in (b).

The basic procedure of PI is that it continuously tries to improve the joint controller by improving the controller maintained for each agent. This per-agent improvement is done using an exhaustive backup operation (this is the same as in the finite-horizon DP case) by which nodes are added to the controller that intuitively correspond to all possible 'one-step longer' policies.[1] To counter the growth of the individual controllers, pruning can be conducted, which removes a node in an agent's controller if it is dominated (i.e., if it has equal or lower value than when beginning in a combination of nodes for all (s,I_{-i})-pairs). That is, the policy for agent i given by beginning in the node is dominated by some set of policies given by starting at other nodes in the controller. These exhaustive backups and pruning steps continue until the solution is provably within ε of an optimal solution. This algorithm can produce an ε-optimal policy in a finite number of steps [Bernstein et al., 2009]. The details of policy iteration follow.

The policy iteration algorithm is shown in Algorithm 7.1. The input is an initial joint controller, m_0, and a parameter ε. At each step, evaluation, backup and pruning occurs. The controller is evaluated using (6.3.1). Next, an exhaustive backup is performed to add nodes to each of the local controllers. Similarly to the finite-horizon case, for each agent i, $|\mathbb{A}_i||\mathbb{I}_i|^{|\mathbb{O}_i|}$ nodes are added to the local controller, where $|\mathbb{I}_i|$ is the number of nodes in the current controller. The exhaustive backup represents starting from each action and for each combination of observations transitioning to each of the nodes in the current controller. Repeated application of exhaustive backups amounts to a brute-force search in the space of deterministic policies, which will converge to ε-optimality, but is obviously quite inefficient.

To increase the efficiency of the algorithm, pruning takes place. Because planning takes place offline, the controllers for each agent are known at each step, but agents will not know which node of their controller any of the other agents will be in during execution. As a result, pruning must be completed over the multiagent belief

[1] Essentially, doing (infinitely many) exhaustive backups will generate all (infinitely many) policies.

Algorithm 7.1 Policy iteration for Dec-POMDPs.

```
Input: An initial joint controller $m_0 = \langle m_{1,0}, \ldots, m_{n,0} \rangle$.
Output: An $\varepsilon$-optimal joint controller $m^*$.
 1: $\tau \leftarrow 0$
 2: $m_\tau \leftarrow m_0$
 3: repeat
 4:    {Backup and evaluate:}
 5:    for $i = 1$ to $n$ do
 6:       $m_{i,\tau} \leftarrow$ExhaustiveBackup$(m_{i,\tau-1})$
 7:    end for
 8:    Compute $V^{m_\tau}$                              {Evaluate the controllers}
 9:    {Prune dominated nodes until none can be removed:}
10:    while some nodes have been pruned do
11:       for $i = 1$ to $n$ do
12:          $m_{i,\tau} \leftarrow$Prune$(i, m_{-i,\tau}, m_{i,\tau})$
13:          UpdateController$(m_{i,\tau})$   {Remove the pruned nodes and update links accordingly}
14:          Compute $V^{m_\tau}$                        {Evaluate updated controllers}
15:       end for
16:    end while
17:    $\tau \leftarrow \tau + 1$
18: until $\frac{\gamma^{\tau+1}|R_{\max}|}{1-\gamma} \leq \varepsilon$
19: return $m^* \leftarrow m_\tau$
```

space (using a linear program that is very similar to that described for finite-horizon dynamic programming in Section 4.3). That is, a node for an agent's controller can only be pruned if there is some combination of nodes that has higher value for all states of the system and at all nodes of the other agents' controllers. Unlike in the finite-horizon case, edges to the removed node are then redirected to the dominating nodes. Because a node may be dominated by a distribution of other nodes, the resulting transitions may be stochastic rather than deterministic. The updated controller is evaluated, and pruning continues until no agent can remove any further nodes.

Convergence to ε-optimality can be calculated based on the discount rate and the number of iterations of backups that have been performed. Let $|R_{\max}|$ be the largest absolute value of any immediate reward in the Dec-POMDP. Then the algorithm terminates after iteration t if $\frac{\gamma^{t+1}|R_{\max}|}{1-\gamma} \leq \varepsilon$. At this point, due to discounting, the value of any policy after step t will be less than ε.

7.2 Optimizing Fixed-Size Controllers

Like optimal finite-horizon approaches, methods for producing ε-optimal infinite-horizon solutions are intractable for even moderately sized problems. This results in optimal algorithms not converging to any reasonable bound of the optimal solution in practice. To combat this intractability, approximate infinite-horizon algorithms

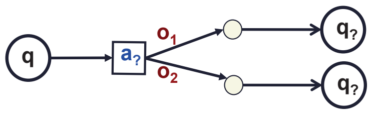

Fig. 7.2: Choosing actions and transitions in each node of a fixed-size controller.

have sought to produce a high-quality solution while keeping the controller sizes for the agents fixed.

That is, the concept behind these approaches is to choose a controller size $|\mathbb{I}_i|$ for each agent and then determine action selection and node transitions parameters that produce high values. However, because these methods fix the size of the agents' controllers, the resulting size may be smaller than is needed for even an ε-optimal solution. This means that, while some of the approaches in this section can produce a controller that has the highest value *given* the fixed size, that value may be arbitrarily far from the optimal solution.

7.2.1 Best-First Search

A way to compute the best *deterministic* joint controller of given size is via heuristic search. One such method, referred to as *best-first search for Dec-POMDPs* [Szer and Charpillet, 2005], generalizes the MAA* technique from Section 4.2.2 to the infinite horizon. Rather than filling templates of trees (cf. Figure 5.1b) the method here fills templates of controllers; it searches through the possible actions that can be taken at each agent's controller nodes and the possible transitions that result from each observation in each node.

In more detail, the algorithm searches through the possible deterministic transition mappings $\iota_i : \mathbb{I}_i \times \mathbb{O}_i \to \mathbb{I}_i$ and the deterministic action function, $\pi_i : \mathbb{I}_i \to \mathbb{A}_i$, for all agents. The controller nodes are ordered and a forward search is conducted that specifies the action selection and node transition parameters for all agents, one node at a time. The heuristic value determines an upper bound value for these partially specified controllers (again, in a way that is similar to MAA*) by assuming centralized control is used for unspecified nodes. In this way, an approximate value for the controller is calculated given the currently specified deterministic parameters and the algorithm fills in the remaining nodes one at a time in a best-first fashion.

This process continues until the value of a set of fully specified controllers is greater than the heuristic value of any partially specified controllers. Since this is an instance of heuristic search applied with an admissible heuristic, this technique is guaranteed to find the best deterministic joint finite-state controller of a given size.

7.2.2 Bounded Policy Iteration

Because stochastic controllers with the same number of nodes are able to produce higher-valued policies than deterministic controllers, researchers have also explored optimizing stochastic controllers (a difficult problem by itself; see the overview and complexity results by Vlassis et al. 2012). These approaches seek to find probabilistic parameters for action selection and node transition. That is, for agent i, the algorithms find the probability of taking action a_i in node I_i, $\Pr(a_i|I_i)$, and the probability of transitioning to node I_i' in node I_i after action a_i is taken and observation o_i made, $\Pr(I_i'|I_i,a_i,o_i)$.

One method to produce stochastic controllers is by using a set of linear programs. In particular, *bounded policy iteration for decentralized POMDPs (Dec-BPI)* [Bernstein et al., 2005], is such a method that extends the BPI algorithm for POMDPs [Poupart and Boutilier, 2003] to the multiagent setting. This approach iterates through the nodes of each agent's controller, attempting to find an improvement for that node. That is, it tries to improve a single local controller, assuming that the controllers of the other agents are fixed, and thus is conceptually similar to JESP, described in Section 5.2.1. In contrast to JESP, however, this improvement for an agent i cannot be found using plain dynamic programming over a tree or directed acyclic graph (DAG) of reachable 'JESP beliefs' $b_i(\langle s,\bar{o}_{-i,t}\rangle)$ (even when replacing histories by internal states leading to a belief of the form $b_i(\langle s,I_{-i}\rangle)$ there would be infinitely many in general) or enumeration of all controllers m_i (there are uncountably many stochastic controllers). Instead, Dec-BPI uses linear programming to search for a new node to replace the old node.

Specifically, this approach iterates through agents i, along with nodes for agent I_i. Then, the method assumes that the current controller will be used from the second step on, and tries to replace the parameters for I_i with ones that will increase value for just the first step. That is, it attempts to find parameters satisfying the following inequality:

$$\forall s_t \in S, \forall I_{-i} \in \mathbb{I}_{-i}$$

$$V^m(s,I) \leq \sum_a \pi(a|I) \left[R(s,a) + \gamma \sum_{s,o',I'} \Pr(I'|I,a,o')\Pr(s',o'|s,a)V^m(s',I') \right].$$

Here, $\mathbb{I}_{-i}$ is the set of controller nodes for all agents besides i. The search for new parameters can be formulated as a linear program in Figure 7.3. Note that a 'combined' action selection and node transition probability,

$$\Pr(I_i',a_i|I_i,o_i) \triangleq \Pr(I_i'|I_i,a_i,o_i)\Pr(a_i|I_i),$$

is used to ensure the improvement constraints are linear; naive inclusion of the right-hand side product would lead to quadratic improvement constraints. Instead, we introduce more variables that lead to a linear formulation. The second probability constraint in Figure 7.3 ensures that the action selection probabilities can be recovered (i.e., that $y(a_i,o_i,I'_i)$ does not encode an invalid distribution). Note that the first and second probability constraints together guarantee that

$$\forall o_i \quad \sum_{I'_i,a_i} y(a_i,o_i,I'_i) = 1.$$

Given: I_i the node for which we want to test improvement.
variables:
ε, the value gap that we try to maximize ($\varepsilon > 0$ indicates an improvement),
$x(a_i) = \pi_i(a_i|I_i)$, action probability variables,
$y(a_i,o_i,I'_i) = \Pr(I'_i,a_i|I_i,o_i)$, combined action and next-stage node probabilities.
maximize: ε.
subject to:
Improvement constraints, $\forall s,I_{-i}$:

$$V^m(s,I) + \varepsilon \leq \sum_a \pi_{-i}(a_{-i}|I_{-i}) \left[x(a_i)R(s,a) + \gamma \sum_{s,o',I'} y(a_i,o_i,I'_i) \Pr(I'_{-i},s',o'|s,a) V^m(s',I') \right]$$

Probability constraints:

$$\begin{aligned} & \sum_{a_i} x(a_i) = 1 \\ \forall a_i,o_i \quad & \sum_{I'_i} y(a_i,o_i,I'_i) = x(a_i) \\ \forall a_i \quad & x(a_i) \geq 0 \\ \forall a_i,o_i,I'_i \quad & y(a_i,o_i,I'_i) \geq 0 \end{aligned}$$

Fig. 7.3: The linear program (LP) to test for improvement in Dec-BPI. The LP determines if there is a probability distribution over actions and transitions from node I_i that improves value when assuming the current controller will be used from the second step on. Note that $\Pr(I'_i,a_i|I_i,o_i)$ is the combined action and transition probability which is made consistent with the action selection probability $\pi_i(a_i|I_i)$ in the probability constraints. This form is needed to ensure the objective function is linear.

This linear program is polynomial in the sizes of the Dec-POMDP and the joint controller, but exponential in the number of agents. Bernstein et al. allow each agent's controller to be correlated by using shared information in a *correlation device* (as discussed in Section 6.2.4). This may improve solution quality while requiring only a limited increase in problem size [Bernstein et al., 2005, 2009].

7.2.3 Nonlinear Programming

Because Dec-BPI can often become stuck in local maxima, nonlinear programming (NLP) has also been used [Amato et al., 2007a, 2010]. The formulation seeks to optimize the value of a set of fixed-size controllers given an initial state distribution. The variables for this problem are the action selection and node transition probabilities for each node of each agent's controller as well as the value of the resulting policy starting from a set of controller nodes.

More formally, the NLP maximizes the expected value of the initial controller node for each agent at the initial belief subject to the Bellman constraints. To this end, let us translate the value of a joint controller (from Equation 6.3.1) in terms of variables that will be used for optimization:

$$z(I,s) = \sum_a \Big[\prod_i x(I_i,a_i)\Big] \left(R(s,a) + \gamma \sum_{s',o} \Pr(s',o|s,a) \sum_{I'} \Big[\prod_j y(I_j,a_i,o_i,I'_j)\Big] z(I',s')\right). \quad (7.2.1)$$

As shown in Figure 7.4, $z(I,s)$ represents the value, $V(I,s_t)$, of executing the controller starting from nodes I and state s, while $x(I_i,a_i)$ is the action selection probability, $\Pr(a_i \mid I_i)$, and $y(I_i,a_i,o_i,I'_i)$ is the node transition probability, $\Pr(I'_i \mid I_i,a_i,o_i)$. Note that to ensure that the values are correct given the action and node transition probabilities, these nonlinear constraints must be added to the optimization which represent the Bellman equations given the policy determined by the action and transition probabilities. We must also ensure that the necessary variables are valid probabilities in a set of probability constraints. This approach differs from DEC-BPI in that it explicitly represents the node values as variables, thus allowing improvement and evaluation to take place simultaneously. An optimal solution of this nonlinear program represents an optimal fixed-size solution to the Dec-POMDP, but as this is often intractable, approximate solvers have been used in practice [Amato et al., 2010].

7.2.4 Expectation Maximization

As an alternative method for determining the parameters of stochastic controllers, expectation maximization (EM) has been used [Kumar and Zilberstein, 2010b, Kumar et al., 2011]. Again a fixed-size controller is assumed, but rather than determining the controller parameters using optimization, the problem is reformulated as a likelihood maximization problem and EM is used. This *planning as inference* technique is an extension of similar methods for POMDPs [Toussaint et al., 2006].

variables for each agent i **:**
$x(I_i,a_i) = \Pr(a_i \mid I_i)$, action probability variables,
$y(I_i,a_i,o_i,I'_i) = \Pr(I'_i \mid I_i,a_i,o_i)$, next-stage node probabilities,
$z(I,s_t) = V(I,s_t)$, the values.
maximize:

$$\sum_{s_0} b_0(s_0) z(I_0,s_0).$$

subject to:
Bellman constraints:

$$\forall I,s \qquad z(I,s) = \sum_a \Big[\prod_i x(I_i,a_i)\Big] \left(R(s,a) + \gamma \sum_{s',o} \Pr(s',o|s,a) \sum_{I'} \Big[\prod_j y(I_j,a_i,o_i,I'_j)\Big] z(I',s') \right). \tag{7.2.2}$$

Probability constraints for each agent i:

$$\begin{aligned} \forall I_i,a_i &\quad \sum_{a_i} x(I_i,a_i) = 1 \\ \forall I_i,a_i,o_i &\quad \sum_{I'_i} y(I_i,a_i,o_i,I'_i) = 1 \\ \forall I_i,a_i &\quad x(I_i,a_i) \geq 0 \\ \forall I_i,a_i,o_i,I'_i &\quad y(I_i,a_i,o_i,I'_i) \geq 0 \end{aligned}$$

Fig. 7.4: The nonlinear program (NLP) representing the optimal fixed-size solution for the problem. The action selection, $\Pr(a_i|I_i)$, and node transition probabilities, $\Pr(I'_i|I_i,a_i,o_i)$, are optimized for each agent i to maximize the value of the controllers. This optimization is performed for the given initial belief b_0 and a given (arbitrarily selected) tuple of the initial nodes, $I_0 = \langle I_{1,0},\ldots,I_{n,0}\rangle$.

The basic idea is that the problem can be represented as an infinite mixture of dynamic Bayesian networks (DBNs), which has one component for each time step t. The DBN responsible for a particular t covers stages $0,\ldots,t$ and represents the 'probability' that the 'maximum reward' is received at its last modeled stage (i.e., at t). The intuition is that the probability of achieving the maximum reward can be considered as a substitute for the value of the controller. We give a brief formalization of this intuition next; for details we refer the reader to the papers by Toussaint et al. [2006], Kumar and Zilberstein [2010b], and Kumar et al. [2011].

First, the formalization is based on binary reward variables, r, for each stage t that provide probability via $\Pr(r=1|s_t,a_t) \triangleq \frac{R(s_t,a_t)-R_{min}}{R_{max}-R_{min}}$, where R_{min} and R_{max} are the smallest and largest immediate rewards. This probability encodes the 'chance of getting the highest possible reward' at stage t. This can be used to define $\Pr(r,Z|t;\theta)$ with $Z = \langle s_0,a_0,s_1,o_1,I_1,a_1,\ldots,a_{t-1},s_t,o_t,I_t\rangle$ the entire histories of states, actions, observations and internal states, and with $\theta = \{\Pr(a|I),\Pr(I'|I,o),\Pr(I)\}$ the parameter vector that specifies the action selection, node transition and initial node probabilities. Now these probabilities constitute a mixture probability via:

$$\Pr(r,Z,t;\theta) = \Pr(r,Z|t;\theta)P(t),$$

with $P(t) = \gamma^t(1-\gamma)$ (used to discount the reward that is received for each time step t). This can be used to define an overall likelihood function $L^\theta(r=1;\theta)$, and it can be shown that maximizing this likelihood is equivalent to optimizing value for the Dec-POMDP (using a fixed-size controller). Specifically, the value function of controller θ can be recovered from the likelihood as $V(b_0) = \frac{(R_{max}-R_{min})L^\theta}{1-\gamma} + \sum_t \gamma^t R_{min}$ [Kumar and Zilberstein, 2010b].

As such, maximizing the value has been cast as the problem of maximizing likelihood in a DBN, and for this the EM algorithm can be used [Bishop, 2006]. It iterates by calculating occupancy probabilities—i.e., the probability of being in each controller node and problem state—and values given fixed controller parameters (in an E-step) and improving the controller parameters (in an M-step). The likelihood and associated value will increase at each iteration until the algorithm converges to a (possibly local) optima. The E-step calculates two quantities. The first is $P_t^\theta(I,s_t)$, the probability of being in state s_t and node I at each stage t. The second quantity, computed for each stage-to-go, is $P_\tau^\theta(r=1|I,s)$, which corresponds to the expected value of starting from I,s and continuing for τ steps. The M-step uses the probabilities calculated in the E-step and the previous controller parameters to update the action selection, node transition and initial node parameters.

After this EM approach was introduced, additional related methods were developed. These updated methods include EM for Dec-POMDPs with factored state spaces [Pajarinen and Peltonen, 2011a] and factored structured representations [Pajarinen and Peltonen, 2011b], and EM using simulation data rather than the full model [Wu et al., 2013].

7.2.5 Reduction to an NOMDP

Similarly to the transformation of finite-horizon Dec-POMDPs into NOMDPs described in Section 4.3, infinite-horizon Dec-POMDPs can also be transformed into (plan-time) NOMDPs. The basic idea here is to replace the observation histories by (a finite number of) information states such as nodes of an FSC. Let $I = \langle I_1,\dots,I_n\rangle$ denote a joint information state. This allows us to redefine the plan-time sufficient statistic as follows:

$$\sigma_t(s,I) \triangleq \Pr(s,I|\delta_0,...,\delta_{t-1}).$$

Again, this statistic can be updated using Bayes' rule. In particular $\sigma'(s',I')$ is given by

$$\forall_{(s',I')} \quad [U_{ss}(\sigma,\delta)](s',I') = \sum_{(s,I)} \Pr(s',I'|s,I,\delta(I))\sigma(s,I), \qquad (7.2.3)$$

where—using $\iota(I'|I,a,o) = \prod_{i\in\mathbb{D}} \iota_i(I'_i|I_i,a_i,o_i)$ for the joint information state update probability—the probability of transitioning to (s',I') is given by

$$\Pr(s',I'|s,I,a) = \Pr(s'|s,a) \sum_o \iota(I'|I,a,o) \Pr(o|a,s'). \tag{7.2.4}$$

It is easy to show that, *for a given set* $\{\iota_i\}$ *of information state functions*, one can construct a plan-time NOMDP analogous to Definition 23 in Section 4.3.3, where augmented states are tuples $\bar{s} = \langle s,I\rangle$. However, as discussed before, in the infinite-horizon setting, the selection of those information state functions becomes part of the problem.

One idea to address this, dating back to Meuleau et al. [1999a], is to make searching the space of deterministic information state functions part of the problem by defining a cross-product MDP in which “a decision is the choice of an action and of a next node”. That is, selection of the ι_i function (in a POMDP with protagonist agent i) can be done by introducing $|\mathbb{O}_i|$ new action variables (say, $a_i^\iota = \{a_i^{\iota,1},\ldots,a_i^{\iota,|\mathbb{O}_i|}\}$) that specify, for each observation $o_i \in \mathbb{O}_i$, to what next internal state to transition. This approach is extended to Dec-POMDPs by MacDermed and Isbell [2013] who introduce the *bounded belief Dec-POMDP*[2] *(BB-Dec-POMDP)*, which is a Dec-POMDP that encodes the selection of optimal $\{\iota_i\}$ by splitting each stage into two stages: one for selection of the domain-level actions and one for selection of the a_i^ι. We omit the details of this formulation and refer the reader to the original paper. The main point that the reader should note is that by making ι part of the (augmented) joint action, the probability $\Pr(s',I'|s,I,a)$ from (7.2.4) no longer depends on external quantities, which means that *it is possible to construct an NOMDP formulation analogous to Definition 23 that in fact does optimize over (deterministic) information state functions*.

This is in fact what MacDermed and Isbell [2013] propose; they construct the NOMDP (to which they refer as a ‘belief-POMDP’) for a BB-Dec-POMDP and solve it with a POMDP solution method. Specifically, they use a modification of the point-based method Perseus [Spaan and Vlassis, 2005] to solve the NOMDP. The modification employed is aimed at mitigating the bottleneck of maximizing over (exponentially many) decision rules in $V^*(\sigma) = \max_\delta Q^*(\sigma,\delta)$. Since the value function is PWLC, the next-stage value function can be represented using a set of vectors $v \in \mathscr{V}$, and we can write

$$\begin{aligned} V^*(\sigma) &= \max_\delta \sum_{(s,I)} \sigma(s,I) \left(R(s,\delta(I)) + \max_{v\in\mathscr{V}} \sum_{(s',I')} \Pr(s',I'|s,I,\delta) v(s',I') \right) \\ &= \max_{v\in\mathscr{V}} \max_\delta \sum_{(s,I)} \sigma(s,I) \underbrace{\left(R(s,\delta(I)) + \sum_{(s',I')} \Pr(s',I'|s,\delta(I)) v(s',I') \right)}_{v_\delta(s,I)}. \end{aligned}$$

[2] The term ‘bounded belief’ refers to the finite number of internal states (or ‘beliefs’) considered.

The key insight is the in the last expression, the bracketed part only depends on $\delta(I) = \langle \delta_1(I_1), \ldots, \delta_1(I_1) \rangle$, i.e., on that part of δ that specifies the actions for I only. As such it is possible to rewrite this value as the maximum of solutions of a collection of collaborative Bayesian games (cf. Section 5.2.3), one for each $v \in \mathscr{V}$:

$$\begin{aligned} V^*(\sigma) &= \max_{v \in \mathscr{V}} \max_{\delta} \sum_{I} \sigma(I) \sum_{s} \sigma(s|I) v_\delta(s,I) \\ &= \max_{v \in \mathscr{V}} \left[\max_{\delta} \sum_{I} \sigma(I) Q^v(I, \delta(I)) \right]. \end{aligned}$$

For each $v \in \mathscr{V}$, the maximization over δ can be interpreted as the solution of a CBG (5.2.2), and therefore can be performed more effectively using a variety of methods [Oliehoek et al., 2010, Kumar and Zilberstein, 2010a, Oliehoek et al., 2012a]. MacDermed and Isbell [2013] propose a method based on the relaxation of an integer program. We note that the maximizing δ directly induces a vector v_δ, which is the result of the point-based backup. As such, this modification can also be used by other point-based POMDP methods.

It is good to note that a BB-Dec-POMDP is just a special case of an (infinite-horizon) Dec-POMDP. The fact that it happens to have a bounded number of information states is nothing new compared to previous approaches: those also limited the number of information states (controller nodes) to a finite number. The conceptual difference, however, is that MacDermed and Isbell [2013] pose this restriction as part of the *model definition*, rather the solution method. This is very much in line with, and a source of inspiration for, the more general definition of multiagent decision problems that we introduced in Section 2.4.4.