Heuristics for Multiagent Reinforcement Learning in Decentralized Decision Problems

Martin W. Allen
and David Hahn
Computer Science Department
University of Wisconsin-La Crosse
La Crosse, Wisconsin
Email: mallen@cs.uwlax.edu

Douglas C. MacFarland
Computer Science Department
Worcester Polytechnic Institute
Worcester, Massachusetts
Email: dcmacfarland@cs.wpi.edu

Abstract—Decentralized partially observable Markov decision processes (Dec-POMDPs) model cooperative multiagent scenarios, providing a powerful general framework for team-based artificial intelligence. While optimal algorithms exist for Dec-POMDPs, theoretical and empirical results demonstrate that they are impractical for many problems of real interest. We examine the use of reinforcement learning (RL) as a means to generate adequate, if not optimal, joint policies for Dec-POMDPs. It is easily demonstrated (and expected) that single-agent RL produces results of little joint utility. We therefore investigate heuristic methods, based upon the dynamics of the Dec-POMDP formulation, that bias the learning process to produce coordinated action. Empirical tests on a benchmark problem show that these heuristics significantly enhance learning performance, even out-performing a hand-crafted heuristic in cases where the learning process converges quickly.

I. INTRODUCTION

Multiagent decision-making poses many challenges for AI research. Techniques that work well—even those that are provably optimal—in a single-agent context often do not scale to multiple agents, or simply fail outright. An example is the decentralized partially observable Markov decision problem (Dec-POMDP), which extends well-known mathematical models for single-agent AI to cooperative team contexts. While the single-agent cases have proven amenable to both dynamic programming and reinforcement learning (RL) approaches, such algorithmic techniques face considerable difficulties when applied to teams—even simply pairs of agents—who must coordinate their activities based on local information.

This work extends what is possible for solving such complex decision problems. We examine a pair of heuristics, generated automatically given a Dec-POMDP problem formulation, that bias RL to encourage agent coordination. The first of these examines the model’s state-transition dynamics, seeking state-action pairs for each agent that appear, heuristically, to offer the possibility of coordinated action; initial value estimates are then set for such pairs, before RL is used to learn the actual values as usual. The second heuristically identifies certain system states in which agents “call for help,” summoning other team-members to their location. Doing so means that situations where multiple agents are present in a single location repeat more often than would happen given the usual RL dynamics, and the reinforcement process can thus learn their true value. Empirical results show that these two heuristics, especially when combined, allow RL to significantly out-perform typical single-agent Q-learning. Even more important, they can in some cases out-perform a hand-crafted heuristic designed specifically to guide agents towards a known optimal joint policy. Together, these heuristic approaches allow learning in Dec-POMDPs to proceed in a much more coordinated fashion.

II. THE DEC-POMDP MODEL

Decentralized partially observable Markov decision processes (Dec-POMDPs) extend the well-known MDP [1] and POMDP [2] to domains in which agents work as a team in a stochastic environment, seeking to optimize a shared value function, based upon limited, local information. They provide a rigorous model for multiagent artificial intelligence, and a basis for control, planning, and learning algorithms.

Formally, a Dec-POMDP is given in terms of a tuple:

$$M = \langle \{\alpha_i\}, S, \{A_i\}, P, \{\Omega_i\}, O, R, T \rangle$$

with elements as follows:
- Each $\alpha_i$ is an agent.
- $S$ is a finite set of world states.
- $A_i$ is a finite set of actions available to $\alpha_i$.
- $P$ is a Markovian state transition probability function.
- $\Omega_i$ is a finite set of observations for $\alpha_i$.
- $O$ is the joint observation function for state transitions.
- $R$ is a global (thus cooperative) reward function.
- $T$ is the time-horizon of the problem (finite or not).

Dec-POMDP dynamics are governed by the transition, observation, and reward functions. At every time-step, agents each choose an action, after which the system state transitions according to function $P$ (in general stochastically, although the model also captures deterministic domains). Based on the eventuating transition, agents each receive individual observations with probabilities given by function $O$, and the group receives common reward according to function $R$.

Like single-agent POMDPs, partial observability arises when agent observations do not determine the underlying
The solution to a Dec-POMDP is a joint policy that describes what each agent will do in response to its history of observations. Given a stochastic problem, an optimal policy is one that maximizes expected joint reward, received via function $R$ over time $T$. As in single-agent models, when the horizon is infinite (i.e., the problem has no definite number of time-steps until termination), the solution sought maximizes expectation over discounted future reward.

For finite-horizon problems, dynamic programming methods can generate policies for each agent based on its observation history, with their union forming an optimal joint policy [3]. (For infinite-horizon Dec-POMDPs, this is not possible; they have single-agent POMDPs as a subcase, and optimal policies are thus generally non-computable [10].) The existence of such optimal methods is cold comfort, however, since Dec-POMDPs are NEXP-complete [11], a complexity class—nondeterministic exponential time—guaranteed to be exponential time (doubly exponential, unless P = NP). In general practice, then, Dec-POMDPs are too difficult to solve optimally, even in very simple domains, although a wide range of sophisticated algorithms have recently begun to extend the range of realistic problems that can be solved [12].

We would argue that this is not a defect in the Dec-POMDP model, but simply a fact about the intrinsic difficulty of the complex, multiagent problems they so effectively represent. Indeed, given that a number of other proposed models of such problems are formally equivalent to Dec-POMDPs, they too share worst-case NEXP complexity [9]. Furthermore, while it is possible to reduce complexity (only to NP-completeness, but still) by restricting agent interactions [13], this is only possible by ensuring almost complete independence between the agents—many models that try to simplify the problem without strong independence are also NEXP-complete [14].

The model thus retains active research interest. While the range of optimal solution algorithms has expanded in the past decade, much work is concerned rather with non-optimal techniques. We follow that path, examining ways in which reinforcement learning can be made effective for Dec-POMDPs, using heuristics that encourage agent coordination.

B. An Example Benchmark Dec-POMDP

Our initial inquiry utilizes a well-studied Dec-POMDP, the repeated box-pushing problem (Fig. 1). Variants have been used to benchmark a number solution techniques [7]. Here, the key elements of the problem are as follows:

**Agents:** there are two agents (circles in the diagram).

**States:** the environment is a $(5 \times 10)$ grid-world, and a global state consists of the positions of the two agents and three boxes, two small and one large. Boxes and agents begin in the positions shown. Boxes can only be moved upwards in their relevant columns, occupying one of 4 positions each; boxes may not overlap with agents, and agents may not overlap with one another. This yields a total of 132,480 states.

**Actions:** Each agent can choose one of five actions at each time-step, either attempting to move in one of the four cardinal directions, or waiting. By moving in the direction of a box from an adjacent square, an agent can attempt to push it.

**Transitions:** If an agent attempts to enter a square already occupied by another agent or leave the grid bounds, or if two agents try to enter the same square at once, the actions fail with certainty. Attempts to push a box in any direction but upward also fails. Pushing either small box upward succeeds with probability 0.8, unless the other agent is blocking the move. If a single agent pushes the large box upward, it fails, but if both agents, working side by side, push the large box, they succeed with probability 0.64. All other actions succeed with certainty. If a box is pushed into the top-most goal region, it will be removed from the grid on the next time-step, appearing immediately in its original location, unless an agent already occupies that location; if so, the box remains in the goal area and does not relocate until the space becomes unoccupied.

**Observations:** Each agent observes its own location in the grid and the contents of the four cardinally-adjacent locations (empty, agent, wall, small box, large box). Each agent has its own private information, and the environment is truly partially observable, since even the union of both agent-observations does not uniquely determine the global state.

**Rewards:** If a small box enters the goal area, joint reward is increased 50 units. If the large box reaches the goal, reward increases 1,000 units. All actions (waiting, successful, or failed moves) reduce reward 1 unit. Thus, for example, net gain when agents push the large box to the goal is 998 units, since 1 unit is subtracted for each agent’s movement. Movement of boxes towards, but not into, the goal area, adds nothing to the reward function—all payoff is delivered upon reaching the goal.

**Time-horizon:** The problem has no finite time-horizon, allowing agents to continue pushing boxes indefinitely.
C. Interesting Aspects of the Problem

The box-pushing domain has all key features of a Dec-POMDP: some stochastic action-outcomes, private information via separate observation functions, joint partial observability of the state, and shared reward. Further, this version of the problem has four important facets as far as this research is concerned: (1) it is complex enough that a straightforward optimal algorithm cannot solve it; (2) it is easy to see what the optimal policy should be, and so to evaluate the quality of solutions found; (3) the optimal policy requires cooperation between agents; and (4) single-agent reinforcement learning methods do not lead to an effective policy.

As to the first point, combinatorics of the state-action space make it practically impossible to solve the problem using dynamic programming, even with aggressive policy pruning [3], [15]. At the same time, although success is less probable when pushing the large box (0.64, versus 0.8 for a small box), the larger reward (1,000 versus 50 units) means that it is still beneficial for agents to work together and push it to the goal; furthermore, the 1-unit penalty for all actions means that the optimal policy is to take some shortest path to the large box each time it re-appears, then repeatedly push upwards until reaching the goal. Assuming agents are able to successfully push the box each time they try (which will happen with probability 0.645 = 0.262), each such sequence will consist of 8 moves—3 upward pushes and 5 moves back into position—for a net reward of 984 units, averaging 123 units per time-step. (Expected reward is somewhat less, given the possibility that pushing fails and the discount factor applied to future rewards, as is typical in indefinite-horizon problems.)

III. Heuristics for RL in Dec-POMDPs

As mentioned, basic reinforcement learning techniques are insufficient for the box-pushing problem. It is known that having each agent employ a single-agent learning algorithm in a domain requiring coordination does not lead to optimal joint policies, and may in fact be harmful [16], [17]. (Of course, that this holds in many other domains does not guarantee it holds in this one; as discussed in Section IV, however, empirical tests conclusively demonstrate that a single-agent algorithm fails here, too.) This has generated a lot of interest in specifically multiagent reinforcement learning (MARL) [18].

Our own work takes a relatively unique approach to the issue, deriving heuristics directly from the Dec-POMDP formulation in order to guide the MARL process in a cooperative direction.

On a broadly conceptual level, our approach is distinct from that employed in much (not all) research. RL is mostly used when underlying system dynamics are unknown, and action-outcomes and rewards must be learned from experience. Our case is quite different: we assume agents know the complete system dynamics beforehand. Our motivation is the intractability of optimally solving Dec-POMDPs—rather than use RL to uncover system dynamics, we use it precisely because we do know them, and can use them to direct learning. Thus, rather than employing algorithms that are blind to the underlying system, we augment an existing algorithm using knowledge gleaned from the problem specification.

We begin with the familiar Q-learning (QL) algorithm [19], in which agents use observed rewards to modify a value function over state-action pairs, \(Q(s, a)\). While learning, agents follow an \(\epsilon\)-greedy policy, choosing actions to maximize \(Q\) in all but fraction \(\epsilon\) of cases, where \(\epsilon\) is initially large to encourage exploration, but is reduced systematically over time as the agent learns more, and exploits that knowledge.

At each step in the learning process, an agent observes current state \(s\), uses its \(\epsilon\)-greedy policy to choose an action \(a\), and then observes positive or negative reward \(r\) and ensuing state \(s'\). \(Q(s, a)\) is then updated using the equation:

\[
Q(s, a) \leftarrow Q(s, a) + \lambda(r + \gamma \max_a Q(s', a'))
\]

with components as follows:

- \(\lambda \in [0, 1]\): a step-size parameter, weighting updates. In this work, we use \(\lambda = 0.5\) when learning; to test learned value of a stable policy, \(\lambda\) is set to 0.
- \(\gamma \in (0, 1)\): a discount factor, to of reduce importance of far-future outcomes. As often the case, we set \(\gamma = 0.9\).
- \(a': Q(s, a)\) is updated using reward \(r\) and \(\max\) for resulting state \(s'\), assuming that real value of \((s, a)\) obtains if we take the best learned action so far.

Each agent in the box-pushing environment performs QL on its own, with separate tables of learned Q-values. We depart from the basic approach, suitable for MDPs, in two key ways. First, states are local, consisting only of individual partial observations of some agent, abstracting over numerous possible global system states. Second, agents apply heuristic bias in two distinct ways, as follows.

First off, we allow agents to alter initial values of state-action pairs. In basic QL, given no prior knowledge of the environment, initial Q-values may be arbitrary and uniform. Indeed, in most RL applications agents do not know the set of possible states ahead of time, and the Q-value tables are initially empty, with entries added incrementally while learning. In our approach, on the other hand, agents begin with the entire problem specification, and initialize Q-values to encourage possible coordinated action. To do so, they analyze the Dec-POMDP dynamics to isolate enabling actions, namely those they that may allow another agent to observe something that they cannot otherwise. Once such an action has been identified, an information-theoretic measure is used to evaluate how much it “matters,” in terms of the difference it makes to the another agent’s observations. This difference measure is defined so that it increases in direct proportion to the probability of observing any new possible state, and is used to make the associated action more likely during learning. Thus, agents begin with a bias towards actions that can make a difference to what others observe, since such differences may indicate states where coordination is possible.

Our second approach derives from the first. In the first heuristic, each agent assigns non-zero values to some entries in its Q-table. Of those pairs, some will have the special...
1: for all \((s, a_1) \in S \times A_1\) do  
2: \(E_{a_1}^s(s_2^t) \leftarrow 0, \forall s_2^t \in \Omega_2\)  
3: for all \(a_2 \in A_2\) do  
4: \(\mathbb{P}_{a_1, a_2}^s \leftarrow \{(s_1^t, p^t), \ldots, (s_m^t, p_m^t)\}\)  
5: \(\Omega_{a_1, a_2}^s \leftarrow \{(s_1^t, p^t), \ldots, (s_k^t, p_k^t)\}\)  
6: for all \((s_2^t, p^t) \in \Omega_{a_1, a_2}^s\) do  
7: \(E_{a_1}^s(s_2^t) \leftarrow E_{a_1}^s(s_2^t) + \frac{1}{|A_2|} p^t\)  
8: end for  
9: end for  
10: end for

Fig. 2. The first algorithm for computing the Q-value heuristic, for agent \(a_1\). Step 5 is generated in accord with equation (2). When complete, \(E_{a_1}^s(s_2^t)\) is the expectation that \(a_2\) observes \(s_2^t\), given local state \(s\) and action \(a_1 \in A_1\). 

feature that the local state involves observing another agent in a neighboring grid-cell. In spatially-defined problems like box-pushing, this provides another opportunity for coordination. In particular, we investigate what happens if agents are allowed to “call for help” from other agents, summoning them to a particular location to learn about possible joint outcomes. When an agent observes a local state in which another agent is not present, but there is some other local state that is identical except that the other is present, there is some probability of seeking aid. This probability grows in direct proportion to the heuristic value of the state featuring the other agent, and calling for help causes others to converge upon the same location, allowing for possible coordinated action at that point. 

### A. Heuristic 1: Biased Initial Q-values

Our first heuristic has two major components, as follows.

1) Finding Enabling Actions: The heuristic seeks to bias agents towards those that can not do alone. To do so, agent \(a_1\) first finds a set of enabling actions for each global state \(s\). The process begins with Dec-POMDP state-transition function \(P^s\), for each global state \(s \in S\) and joint action-pair \(\pi = (a_1, a_2)\), a distribution over possible next states is given as set:

\[
\mathbb{P}_{\pi}^s = \{(s_1^t, p^t), (s_2^t, p_2^t), \ldots, (s_m^t, p_m^t)\}
\]

with each component consisting of some next state \(s^k \in S\) and a probability value \(p^k\), so (a) \(|\mathbb{P}_{\pi}^s| = m \leq |S|\), (b) \(\forall k, p^k > 0\), and (c) \(\sum_k p^k = 1\). Furthermore, we assume a deterministic observation function, as with the box-pushing problem; that is, for each agent \(a_i\) and global state \(s\) we have a fixed resulting local state: \(O_i(s) = s_i^t \in \Omega_i\) such that \(O_i(s_i^t | s) = 1\), where \(O_i\) is the marginal distribution over observations for \(a_i\).

Each agent generates heuristic initial Q-values. The first algorithm used (Fig. 2) is given for agent \(a_1\), and is easily extended to \(a_2\) by permuting subscripts. Here, \(a_1\) iterates over pairs of global states \(s \in S\) and its own actions \(a_1 \in A_1\) to produce, for each local state \(s_2^t\) of agent \(a_2\), a probability value \(E_{a_1}^s(s_2^t)\). This value is computing by considering all possible action-pairs, \((a_1, a_2)\), and converting global probability-set

1: for all \((s, (a_1, a_2)) \in S \times (A_1 \times A_2)\) do  
2: \(\mathbb{P}_{a_1, a_2}^s \leftarrow \{(s_1^t, p^t), \ldots, (s_m^t, p_m^t)\}\)  
3: for all \((s^t, p^t) \in \mathbb{P}_{a_1, a_2}^s\) do  
4: \(\hat{p} \leftarrow E_{a_1}^s(O_2(s^t))\)  
5: if \(\hat{p} \neq 0\) and \(\exists a'_1 \in A_1, E_{a_1}^s(O_2(s^t)) = 0\) then  
6: \(r \leftarrow R(s, (a_1, a_2), s^t)\)  
7: \(\delta \leftarrow KD(\hat{p})\)  
8: \(Q(O_1(s), a_1) \leftarrow Q(O_1(s), a_1) + \delta(r + |\max R|)\)  
9: end if  
10: end for  
11: end for

Fig. 3. The second algorithm for computing the Q-value heuristic, for agent \(a_1\). When complete, each pair \((s_1, a_1)\) such that \(s_1\) is a local state of \(a_1\) and \(a_1\) is an action by \(a_1\) that enables local states of \(a_2\) is assigned a heuristic Q-value, using the KD Divergence measure (equations (3), (4)), weighted by one-step reward \(r\) and maximal reward magnitude (equation (5)).

\[
\mathbb{P}_{a_1, a_2}^s \text{ to a probability-set over local states of } a_2:
\]

\[
\Omega_{a_1, a_2}^s = \{(s_2^t, p^t) | \exists (s^k, p^k) \in \mathbb{P}_{a_1, a_2}^s, s_2^t = O_2(s_2^k) \wedge p^t = \sum (s_2^t, p^t) \in \mathbb{P}_{a_1, a_2}^s, s_2^t = O_2(s_2^k)\}.
\] (2)

This set is used to incrementally compute \(E_{a_1}^s(s_2^t)\), the expectation that agent \(a_2\) observes local state \(s_2^t\), given that \(a_1\) takes action \(a_1\) and \(a_2\) chooses actions uniformly at random.

2) Rating Enabling Actions: In the prior step, each agent \(a_i\) computes sets of enabling actions \(a_i\) for every global state \(s\), those for which expectation \(E_{a_1}^s(s_2^t)\) is not zero, for some local state \(s_2^k\) of other agent, \(a_k\). This information, along with reward function \(R\), is then used to bias initial value \(Q(O_1(s), a_i)\) (Fig. 3). Note that the heuristic is computed only for those actions \(a_i\) that are genuinely enabling in the sense that not every other action also enables the local state—in other words, the heuristic is only applied when choosing \(a_i\) for state \(s\) allows \(a_k\) to experience local state \(s_2^k\) that may not otherwise be possible. If action \(a_i\) qualifies, non-zero probability \(\hat{p} = E_{a_1}^s(s_2^t)\) is used to calculate an information-theoretic divergence measure.

The heuristic calculation is designed only to rank enabling actions based upon how likely (or not) they make local state \(s_2^k\), and so the particular divergence used is not strictly important, so long as it is well-defined. Here, we use Lin’s K-Divergence [20] for distributions \(P_1\) and \(P_2\) over X:

\[
K(P_1, P_2) = \sum_{x \in X} P_1(x) \log_2 \frac{P_1(x)}{\frac{1}{2}(P_1(x) + P_2(x))}.
\] (3)

Other divergence measures are certainly possible, including Lin’s L and Jensen-Shannon measures. Unlike some, e.g., Kullback-Leibler [21] or Jeffrey’s [22], \(K\) has the advantage of being well-defined if \(\exists x \in X, (P_1(x) = 0 \lor P_2(x) = 0)\).

We treat observation as a binary-valued random variable, since agent \(a_k\) either observes \(s_2^k\) with probability \(\hat{p}\), or observes another local state with probability \((1 - \hat{p})\), and
compare that distribution to the 0 probability of observing $s_k^j$ under any non-enabling action:

$$\text{KD}(\hat{p}) = K((\hat{p}, 1 - \hat{p}), (0, 1))$$

$$= \hat{p} \log_2 \frac{\hat{p}}{\frac{1}{2}(\hat{p} + 0)} + (1 - \hat{p}) \log_2 \frac{1 - \hat{p}}{\frac{1}{2}(1 - \hat{p}) + 1} \tag{4}$$

Thus, $\delta$ is non-negative, $0 \leq \delta \leq 1$, increasing monotonically with $\hat{p}$, so actions with a higher enabling probability receive a larger heuristic value. Finally, we use $\delta$ as a weighting term on reward to get heuristic value as follows:

$$\delta(r + |\max R|), \tag{5}$$

where $r$ is the one-step reward for the related global state transition and $|\max R|$ denotes the magnitude of the largest reward-value (positive or negative) in the Dec-POMDP. This has the effect that initial heuristic values are set so that they will not immediately be overwhelmed by rewards actually encountered during learning.

3) Discussion of the Heuristic: Together, these computations examine the state-transition model of the Dec-POMDP, seeking state-transitions that can only occur if agent actions coordinate in some fashion, and ignoring cases where no coordination occurs possible. As an example, consider the problem’s initial state $s^0$ (Fig. 1), where the two agents are extensively physically separated. When, for example, agent $\alpha_1$ considers this state during the first heuristic algorithm (Fig. 2), then no matter which action of $\alpha_2$ it considers, possible outcomes are not affected at all, since nothing $\alpha_1$ might do can affect what $\alpha_2$ observes on the next time-step. (Recall that an agent observes only its own location and the four adjacent cells, and that agents move only a single square at a time.) Thus, when $\alpha_1$ comes to the condition of line 5 in the second algorithm (Fig. 3), any observation by $\alpha_2$ will either remain impossible, in the sense that it already has probability 0 under observation function $O$ and so $\hat{p} = 0$ as well, or will be possible no matter what $\alpha_1$ does: $\forall a' \in A_1, E_{\alpha_1}^o(O_2(s^0)) \neq 0$.

In such a state, no heuristic value will accrue to any action.

Fig. 4 shows a more interesting example, when the agents are poised to push the large box into the goal. In this case, the move_up action of $\alpha_1$, combined with the same action by $\alpha_2$, enables $\alpha_2$ to observe something not otherwise possible, namely occupying the region just below the goal, with the large box above it and $\alpha_1$ to its left. Given that the probability of this occurring is 0.64 when both agents work together, the expected value $E_{\text{up,up}}(O_2(s^1)) = 0.64/5 = 0.128$, under the assumption that $\alpha_2$ will in fact choose its action uniformly at random, and divergence is thus $\delta = KD(0.128) \approx 0.039$.

Then, since the reward $r$ obtained for entering this state is 998 units, which is also the maximum value possible, heuristic value $(0.039 \cdot 2 \cdot 998) \approx 77.6$ is added to $Q(s_1, a_1)$, where $a_1$ is the move_up action, and $s_1 = O_1(s)$ is $\alpha_1$’s own observation of the state pictured in Fig. 4.

Finally, we note that not every state-action pair picked out by the heuristic is actually part of the optimal joint policy— it is a heuristic, after all, and agents cannot pre-compute the actual usefulness of actions before they even begin learning. Fig. 5 gives an example. In this state, $\alpha_1$ has pushed a box into the goal region, which would normally cause the box to relocate to its original position in the next time-step. However, $\alpha_2$ currently occupies that position; this means that if both agents choose wait actions, $\alpha_1$ can observe something not otherwise possible, namely the box remaining in the goal region rather than vanishing from view. For our heuristic, this counts as potential coordination, since $\alpha_2$ waiting enables something no other actions allow (if $\alpha_2$ moves in any direction, the box will relocate). Thus, positive heuristic value is attached, initially incentivizing $\alpha_2$ to wait rather than move away.

As remarked, this is the nature of a heuristic—not every decision it makes will be the right one. In fact, we consider this something of a conceptual strength. Rather than hand-craft a heuristic guaranteed to assign value only to actions from a known optimal policy, we have designed ours so it requires no such special knowledge. Heuristic computation is based entirely upon identifying state-action pairs in a Dec-POMDP where some coordination seems possible, without needing to know whether or not that coordination is genuine (or indeed even benign). Since the heuristic value only sets initial $Q$-values, the RL process that follows can be used to discover which “coordination” opportunities are truly beneficial.

B. Heuristic 2: Calling for Help

After computing heuristic $Q$-values, each agent $\alpha_i$ iterates over those $(s_i^1, a_i)$ pairs non-zero initial heuristic, and if local state $s_i^1$ involves observing another agent in a neighboring location, generates a new probability:

$$H_i(s_i^1) = \frac{\kappa}{|s_i^1|} \sum_{a_i \in A_i} Q(s_i^1, a_i) \sum_{a_i \in A_i} Q(s_i^1, a_i) \tag{6}$$

where $\kappa$ is some upper limit on probabilities (0.8 in this research), and $s_i^1$ is the local state identical to $s_i^1$, but containing no observation of the other agent in an adjacent location.

Thus, for the state in Fig. 4, agent $\alpha_1$ will assign some positive heuristic value to action move_up in local state $s_i^1$.
consisting of the given location, with \( \alpha_2 \) to the right, and the box above. Based on this fact, it will assign a probability \( H_1(s_1^t) \) to the state \( s_1^t \) that is identical to \( s_2^t \), except that \( \alpha_2 \) is not present and \( \alpha_1 \) observes only empty space to its right. This probability increases with heuristic value: states with maximal heuristic value receive probability \( \kappa \), and all others receive proportionally lesser probability.

Now, if \( \alpha_1 \) enters state \( s_1^t \) during the learning process, it will, with probability \( H_1(s_1^t) \), engage in the following procedure:

1. \( \alpha_1 \) pauses learning, memorizes its current location, and begins “calling for help,” moving uniformly at random.
2. When in close proximity to agent \( \alpha_2 \) (here, within 3 moves in any direction), \( \alpha_1 \) communicates its memorized location to \( \alpha_2 \), which also pauses learning.
3. Both agents move immediately to the memorized location (with \( \alpha_2 \) in some available neighboring location), and begin using the \( Q \)-learning algorithm again.

Thus, when an agent identifies a location with some heuristic value associated with apparently coordinated activity, involving the presence of the other agent, it will occasionally and probabilistically decide to summon the other agent to that location during learning. Again this heuristic privileges opportunities for coordination, and again it is imperfect, since there can be many cases where agents summon each others to locations at which no important joint action is truly available.

C. Time Cost to Calculate Heuristic Values

Space does not permit detailed analysis of the complexity of heuristic computation, but we note that each is at worst quadratic in the size of the state-set, \( S \), since transition function \( P \) may involve non-zero transitions between every possible state-pair. If \( S \) is very large and the specification of \( P \) is rendered in some compact (perhaps functional) form, this can be quite costly. In practice, however, and especially if \( P \) is given in plain tabular format, the entire process can be handled in time linear in the size of the Dec-POMDP specification. In addition, many internal iterations in the algorithms are eliminated by proper use of data structures. In our experience, heuristic computation takes a small fraction of the time required for the learning stage, especially in cases where policy exploration is extensive, and convergence to a stable policy takes many time-steps. (We do note that when using the “call for help” heuristic, learning takes longer, since steps during the process of seeking out the other agent are not counted against the learning process.)

IV. EMPIRICAL RESULTS

We performed a range of experimental runs, varying heuristics used (if any), in the following 6 combinations:

1. Basic \( Q \)-learning, where each agent simply employed its own single-agent algorithm.
2. \( Q \)L with each agent using biased initial \( Q \)-values, as in Section III-A.
3. \( Q \)L with a hand-crafted heuristic: agents always and only summoned help when alone under the large box (with and without biased initial \( Q \)-values).

(5, 6) \( Q \)L where agents called for help based upon computed heuristic probabilities, as in Section III-B (with and without biased initial \( Q \)-values).

Of these combinations, the first is intended only to provide a baseline, with no expectation of finding effective coordinated policies. The third and fourth combinations, where agents always called for help when under the large box, serves as a point of comparison between our more principled heuristics and a hand-crafted one that is based upon our prior knowledge of the actual optimal policy (since it is only under the large box that coordinated action is truly necessary and valuable).

For each of these combinations, learning was done under 5 different rates of convergence, reducing random exploration parameter \( \epsilon \) every \( M \in \{5, 10, 50, 100, 500\} \) thousand time-steps, terminating when \( \epsilon < 0.01 \); thus, e.g., runs with \( M = 10 \) take twice as long to converge as ones with \( M = 5 \). (Reduction is achieved by multiplying \( \epsilon \) by \( 1/2, 1/3, 1/4, \ldots \), successively every \( M \) time-steps). Each combination of algorithm and \( \epsilon \)-schedule was run to completion 20 times, for 600 total tests, tracking accumulated reward over time. Reported values were averaged over all runs to mitigate stochastic effects.

Our results show that the heuristics can improve the outcome of RL in our sample Dec-POMDP domain, especially where less exploration is allowed. Final values for total accumulated reward over time are shown in Table I. As expected, basic \( Q \)-learning fails to produce coordination in the multiagent box-pushing domain. While remarkably consistent, use of the single-agent algorithms by itself results in far from optimal results, and when output policies are inspected after the fact, it is revealed that each agent only learns to push a single smaller box, on its own, with no coordinated movement of the large box ever arising, no matter how long agents randomly explore. Domain dynamics are too complex for agents to randomly explore the behavior of jointly pushing the large box very often—certainly not often enough so to reinforce its long-term utility. For the small boxes, on the other hand, exploring agents attempt pushing enough times to reliably discover the reward available, and each thus learns a policy of repeatedly doing so.

For longer time-horizons, the introduction of the biased \( Q \)-values into the picture \( (QL + BQ) \) does produce a modest improvement in overall value, suggesting more rapid convergence upon valuable actions. The difference is negligible, however, and observed policies failed to push the large box to the goal, calling into question the validity of the observed effect.

More interesting results arise when comparing the use of a hand-crafted fixed policy for summoning help with heuristic and probabilistic decisions about when to do so. Over longer time-horizons—when \( \epsilon \)-randomness is reduced every 50,000 or more steps—agents do considerably better when always and only summoning aid in the presence of the large box \( (LB) \). This is unsurprising, since the hand-crafted heuristic in some sense “gives the game away,” focusing attention specifically upon those states in which coordination is actually needed. Thus, agents repeatedly visit the large box together, allowing reinforcement toward optimal policy. Even with such
TABLE I

TOTAL REWARD ACCUMULATED DURING LEARNING, RELATIVE TO $\epsilon$-REDUCTION SCHEDULE. QL: SINGLE-AGENT Q-LEARNING; BQ: HEURISTICALLY BIASED INITIAL Q-VALUES; LB: ALWAYS CALL FOR HELP UNDER LARGE BOX; HH: USE HEURISTIC TO DECIDE WHEN TO CALL FOR HELP. MAXIMAL ACCUMULATED VALUES MARKED IN BOLD FOR EACH ROW. (ALL RESULTS AVERAGED OVER 20 RUNS.)

<table>
<thead>
<tr>
<th>$\epsilon$-red. (K)</th>
<th>QL</th>
<th>QL + BQ</th>
<th>LB</th>
<th>LB + BQ</th>
<th>HH</th>
<th>HH + BQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>$3.06 \times 10^6$</td>
<td>$2.8 \times 10^6$</td>
<td>$3.48 \times 10^6$</td>
<td>$5.41 \times 10^6$</td>
<td>$1.63 \times 10^6$</td>
<td>$1.43 \times 10^7$</td>
</tr>
<tr>
<td>10</td>
<td>$6.12 \times 10^6$</td>
<td>$6.09 \times 10^6$</td>
<td>$8.62 \times 10^6$</td>
<td>$1.92 \times 10^7$</td>
<td>$6.19 \times 10^6$</td>
<td>$2.62 \times 10^7$</td>
</tr>
<tr>
<td>50</td>
<td>$3.06 \times 10^7$</td>
<td>$3.24 \times 10^7$</td>
<td>$2.21 \times 10^8$</td>
<td>$2.44 \times 10^8$</td>
<td>$7.38 \times 10^7$</td>
<td>$1.04 \times 10^8$</td>
</tr>
<tr>
<td>100</td>
<td>$6.12 \times 10^7$</td>
<td>$6.52 \times 10^7$</td>
<td>$4.52 \times 10^8$</td>
<td>$4.92 \times 10^8$</td>
<td>$1.63 \times 10^8$</td>
<td>$1.36 \times 10^8$</td>
</tr>
<tr>
<td>500</td>
<td>$3.06 \times 10^8$</td>
<td>$3.28 \times 10^8$</td>
<td>$2.15 \times 10^9$</td>
<td>$2.15 \times 10^9$</td>
<td>$5.64 \times 10^8$</td>
<td>$6.34 \times 10^8$</td>
</tr>
</tbody>
</table>

(a) Reducing $\epsilon$-randomness every 5,000 steps.  
(b) Reducing $\epsilon$-randomness every 500,000 steps.

Fig. 6. Average reward per time-step for two extremal $\epsilon$-randomness reduction schedules. (Approaches are as described in Table I.)

a pronounced built-in advantage, however, agents are still aided by use of the biased initial $Q$-value heuristic ($LB + BQ$), doing somewhat better over the middle time-horizons. For shorter time horizons, when learning converges much more quickly, however, we see an entirely opposite and altogether more interesting phenomenon: in these cases, while the hand-crafted fixed policy still outperforms one in which agents use heuristic probability only to determine when to ask for help ($HH$), the best results actually come from combining heuristics ($HH + BQ$). Not only does this approach outperform normal $Q$-learning by an order of magnitude in every case, it is also an order of magnitude better than even the hand-crafted approach for the shortest time horizon, when $\epsilon$ is reduced every 5,000 steps. When we reduce $\epsilon$ every 10,000 steps, the advantage is not considerable, although agents do accumulate $7 \times 10^8$ more units of value on average.

These results are emphasized by the shapes of the graphs in Fig. 6, which show average reward per time-step when using the single-agent algorithm ($QL$), the hand-crafted fixed policy for seeking help ($LB, LB + BQ$), and the approach that combines both of our heuristics ($HH + BQ$). As shown there, the simple fixed policy, with or without biased initial values, works best over the longest horizon (Fig. 6b), accumulating considerably more reward per time-step than the dual-heuristic approach. Although the latter accumulates reward at a higher rate than simple single-agent $Q$-learning, especially earlier in the learning process, its use of sophisticated heuristic calculations is mostly trumped by the hand-crafted policy.

On the other hand, when using the shortest learning horizon (Fig. 6a), rapidly reducing $\epsilon$-randomness to converge upon a learned policy, the dual-heuristic approach significantly outdoes even the specially crafted fixed policy. In this case, the average reward accumulated by the fixed policy without heuristics is nearly indistinguishable from that received when doing simple single-agent $Q$-learning, suggesting that without extensive exploration, the reinforcement process is still unable to generate genuine coordination, even with a strong helping hand. Using heuristically biased initial $Q$-values along with the fixed help-policy improves things somewhat, but average reward is not nearly that accumulated by the dual-heuristic approach. Over the shorter term, then, with less time to explore, the heuristic combination can effectively and quickly guide agents towards profitable cooperative strategies.

Finally, Table II shows a smooth inversion as the length of time given for exploration increases. As the interval between $\epsilon$-reductions grows, reward per time-step for the dual-heuristic approach steadily decreases (although it is always at least twice as effective as elementary $Q$-learning). Things are nearly as smooth for the combination of the fixed policy for summoning help with heuristically biased initial $Q$-values, which increases in value steadily before losing some ground (albeit while remaining dominant) at the longest time-horizon.
The heuristics investigated have been shown to profitably bias reinforcement learning processes, encouraging agents to coordinate actions, producing more valuable outcomes. Since they are based solely upon the formulation of the problem as a decentralized POMDP, they can be applied to nearly any domain. In addition, although the heuristics are based upon the particular value-formulation used by $Q$-learning, they can be straightforwardly adapted for use with other RL algorithms.

When learning is allowed to converge quickly, the combined heuristic approach outperforms even a hand-crafted policy designed especially to lead to pre-computed optimal behavior. This is of substantial import. When we reduce $\epsilon$-randomness every 5,000 iterations, and our dual heuristic approach dominates, the entire learning process takes $4.55 \times 10^5$ total steps, and can be completed in a matter of a few minutes. On the other hand, when randomness is reduced every 500,000 iterations, the total number of steps likewise increases by a factor of 100. In that additional number of time-steps, simply following the dual-heuristic, short-horizon policy and accruing value at its average rate (Table II) would very nearly make up for all the gains to be had by pursuing the hand-crafted approach. We also note that total compute time for the longer learning runs (including non-learning steps during the process of summoning aid) extends much further again, with some runs taking more than a day’s time in total. Further, such a fixed policy is only possible when we already know which states are profitable for optimal coordination—precisely when learning is not in fact required! In contrast, our heuristic approaches allow agents to discover their own ways to coordinate when faced with problems that we do not already know how to solve.

Together, then, these results open the door to further investigation. Our ongoing work involves extending the heuristic techniques to other domains for comparison, while also modifying and developing the heuristics themselves. We are particularly interested in the mathematical analysis of the heuristics, and of their relation to outcome policy value. The largest open question is about generality, namely when we can expect such an approach to work. In box-pushing, purely local observations and rewards still function as a guide to joint success, something that will not be true in all domains. In fact, it should be possible to construct problems specifically designed to foil such a heuristic approach. We are currently investigating such domains, and looking into the problem features that might indicate whether or not our approach would usefully apply. Much more remains to be said.

### V. Conclusions

The heuristics investigated have been shown to profitably bias reinforcement learning processes, encouraging agents to coordinate actions, producing more valuable outcomes. Since they are based solely upon the formulation of the problem as a decentralized POMDP, they can be applied to nearly any domain. In addition, although the heuristics are based upon the particular value-formulation used by $Q$-learning, they can be straightforwardly adapted for use with other RL algorithms.

When learning is allowed to converge quickly, the combined heuristic approach outperforms even a hand-crafted policy designed especially to lead to pre-computed optimal behavior. This is of substantial import. When we reduce $\epsilon$-randomness every 5,000 iterations, and our dual heuristic approach dominates, the entire learning process takes $4.55 \times 10^5$ total steps, and can be completed in a matter of a few minutes. On the other hand, when randomness is reduced every 500,000 iterations, the total number of steps likewise increases by a factor of 100. In that additional number of time-steps, simply following the dual-heuristic, short-horizon policy and accruing value at its average rate (Table II) would very nearly make up for all the gains to be had by pursuing the hand-crafted approach. We also note that total compute time for the longer learning runs (including non-learning steps during the process of summoning aid) extends much further again, with some runs taking more than a day’s time in total. Further, such a fixed policy is only possible when we already know which states are profitable for optimal coordination—precisely when learning is not in fact required! In contrast, our heuristic approaches allow agents to discover their own ways to coordinate when faced with problems that we do not already know how to solve.

Together, then, these results open the door to further investigation. Our ongoing work involves extending the heuristic techniques to other domains for comparison, while also modifying and developing the heuristics themselves. We are particularly interested in the mathematical analysis of the heuristics, and of their relation to outcome policy value. The largest open question is about generality, namely when we can expect such an approach to work. In box-pushing, purely local observations and rewards still function as a guide to joint success, something that will not be true in all domains. In fact, it should be possible to construct problems specifically designed to foil such a heuristic approach. We are currently investigating such domains, and looking into the problem features that might indicate whether or not our approach would usefully apply. Much more remains to be said.

### ACKNOWLEDGMENTS

The authors are grateful for research support provide by the University of Wisconsin-La Crosse, the College of Science and Health, and the Computer Science Department.

### REFERENCES


