Q-Decomposition for Reinforcement Learning Agents

Stuart Russell RUSSELL@CS.BERKELEY.EDU
Andrew L. Zimdars ZIMDARS@CS.BERKELEY.EDU
Computer Science Division, University of California, Berkeley, Berkeley CA 94720-1776 USA

Abstract

The paper explores a very simple agent design method called *Q-decomposition*, wherein a complex agent is built from simpler subagents. Each subagent has its own reward function and runs its own reinforcement learning process. It supplies to a central arbitrator the Q-values (according to its own reward function) for each possible action. The arbitrator selects an action maximizing the sum of Q-values from all the subagents. This approach has advantages over designs in which subagents recommend actions. It also has the property that if each subagent runs the Sarsa reinforcement learning algorithm to learn its local Q-function, then a globally optimal policy is achieved. (On the other hand, local Q-learning leads to globally suboptimal behavior.) In some cases, this form of agent decomposition allows the local Q-functions to be expressed (or approximated) by much-reduced state and action spaces. These results are illustrated in two domains that require effective coordination of behaviors.

1. Introduction

A natural approach to developing agents for complex tasks is to decompose the monolithic agent architecture into a collection of simpler subagents and provide an arbitrator that combines the outputs of these subagents. The principal architectural choices in such a design concern the nature of the information communicated between the arbitrator and the subagents and the method by which the arbitrator selects an action given the information it receives.

We will illustrate the various architectural choices using a very simple environment (Figure 1). The agent starts in state S_0 and can attempt to move $Left$, Up, or $Right$, or it can stay put. With probability ϵ, each movement action has no effect; otherwise, the agent reaches a terminal state with rewards of dollars and/or euros as shown. If we assume rough parity between dollars and euros, then the

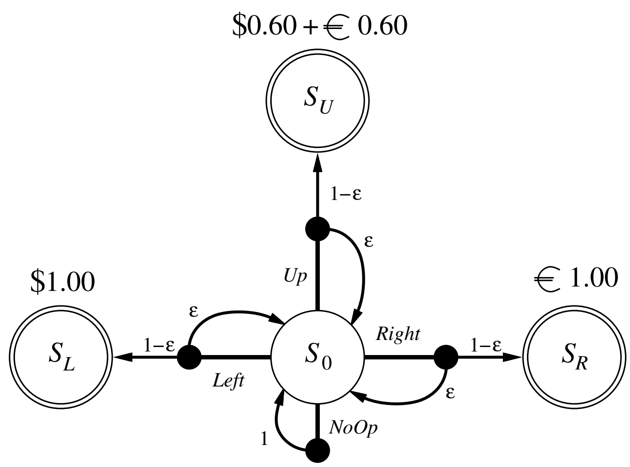

Figure 1. A simple world with initial state S_0 and three terminal states S_L, S_U, S_R, each with an associated reward of dollars and/or euros. The discount factor is $\gamma \in (0, 1)$.

optimal policy is clearly to go Up. The question is how to achieve this with a distributed architecture in which one subagent cares only for dollars and the other only for euros.

One very common design, called *command arbitration*, requires each subagent to recommend an action to the arbitrator. In the simplest such scheme, the arbitrator chooses one of the actions and executes it (Brooks, 1986; Maes, 1989). The problem with this approach is that each subagent may suggest an action that makes the other subagents very unhappy; there is no way to find a "compromise" action that is reasonable from every subagent's viewpoint. In our example, the dollar-seeking subagent will suggest $Left$ whereas the euro-seeking subagent will suggest $Right$. Whichever action is chosen by command arbitration, the agent is worse off than it would be if it went Up.

To overcome such problems, some have proposed *command fusion*, whereby the arbitrator executes some kind of combination (such as an average) of the subagents' recommendations (Saffiotti et al., 1995; Ogasawara, 1993; Lin, 1993; Goldberg et al., in press). Unfortunately, fusing the subagents' actions may be disastrous. In our example, averaging the direction vectors for *Left* and *Right* yields *NoOp*, which is the worst possible choice. Furthermore, command fusion is often inapplicable—as, for example, when two chess-playing subagents recommend a knight move and a bishop move respectively.

The weaknesses of command arbitration have been pointed out previously by proponents of *utility fusion* (Rosenblatt, 2000; Pirjanian, 2000). In a utility-fusion agent, each subagent calculates its own outcome probabilities for actions and its own utilities for the outcome states. The arbitrator combines this information to obtain a global estimate of the utility of each action. Although the semantics of probability combination is somewhat unclear, the method does make it possible to produce meaningful compromise actions. Rosenblatt reports much-improved performance for an autonomous land vehicle, compared to command arbitration. Unfortunately, the paper gives no indication as to the semantics or the origin of the local utility functions. We will see below that global optimality imposes nontrivial conditions on the local utilities if fusion is to work.

Our proposal is the Q-*decomposition* method, which requires each subagent to indicate a value, from its perspective, for every action. That is, agent j reports its action values $Q_j(s,a)$ for the current state s to the arbitrator; the arbitrator then chooses an action maximizing the *sum* of the Q_j values.[1] In this way, an ideal compromise can be found.

The primary theoretical assumption underlying Q-decomposition is that the agent's overall reward function $r(s,a,s')$ can be additively decomposed into separate rewards $r_j(s,a,s')$ for each subagent—that is, $r(s,a,s') = \sum_j r_j(s,a,s')$. Thus, for a mobile robot, one subagent might be concerned with obstacle avoidance and receive a negative reward for each bump, while another subagent might be concerned with navigation and receive a positive reward for making progress towards a goal; the agent's overall reward function must be a sum of these two kinds of rewards. Of course, additive decomposition can *always* be achieved by choosing the right subagent reward functions. Heuristically speaking, we are interested in decompositions that meet two criteria. First, we want to be able to arrange the "physical" agent design so that each subagent receives just its own reward r_j. Second, each subagent's action-value function $Q_j(s,a)$, which predicts the sum of r_j rewards the agent expects to receive over time, ought to be simpler to express, and easier to learn, than the global Q-function for the whole agent. In many cases, it should be possible to design subagents so that they need sense only a subset of the state variables and need express preferences only over a subset of the components that describe the global actions.

[1]The idea is a natural one and was sketched previously in unpublished comments by the first author (Russell, 1991).

We would like to ensure that the agent's behavior is globally optimal, even if it results from a distributed deliberation process. We show that this is achieved if each subagent's Q_j-function correctly reflects its own future r_j rewards *assuming future decisions are made according to the global arbitration policy*. The next question is how to arrange for each subagent to learn the right Q_j-function using a local reinforcement learning procedure, ideally one that does not need to access the Q_j-functions or rewards of the other subagents.

We show in Section 3.1 that if each agent uses the conventional Q-learning algorithm (Watkins, 1989), global optimality is *not* achieved. Instead, each subagent learns the Q_j values that would obtain if that subagent were to make all future decisions for the agent. This "illusion of control" means that the subagents converge to "selfish" estimates that overestimate the true values of their own rewards with respect to a globally optimal policy. For our dollar/euro example, local Q-learning leads in some cases to a global policy that chooses $NoOp$ in state S_0.

The principal result of the paper (Section 3.2) is the simple observation that global optimality is achieved by local reinforcement learning with the Sarsa algorithm (Rummery & Niranjan, 1994), provided that on each iteration the arbitrator communicates its decision to the subagents. This information allows the subagents to become realistic, rather than optimistic, about their own future rewards.

Sections 4 and 5 investigate Q-decomposition in worlds that are somewhat less trivial than the dollar/euro example. The first is the well-known "racetracks" problem with two subagents: one wants to make progress and the other wants to avoid crashes. The second example is a simulated fishery conservation problem, in which several subagents (fishing boats) must learn to cooperate to extract the maximum sustainable catch. This example illustrates how conflicts between selfish actors can lead to a "tragedy of the commons".

2. Background and definitions

From a "global" viewpoint, we will assume a standard Markov decision process $\langle \mathcal{S}, \mathcal{A}, P, r, \gamma \rangle$, with (finite) state space $\mathcal{S}$, primitive actions $\mathcal{A}$, transition measure P, bounded reward function $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$, and discount factor $\gamma \in (0,1]$. From a "local" viewpoint, however, we assume that the reward signal is decomposed into an n-element vector $\mathbf{r}$ of bounded reward components r_j, each defined over the full state and action space, such that subagent j receives r_j and such that $r = \sum_{j=1}^{n} r_j$.

Denote policies by $\pi : \mathcal{S} \rightarrow \mathcal{A}$, and associate with each reward component r_j and policy π the expected discounted future value $Q_j^\pi : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ of a (state, action) pair:

$$Q_j^\pi(s,a) = E\left[\sum_{a \in \mathcal{A}} r_j(s,a,s') + \gamma Q_j^\pi(s', \pi(s'))\right]$$

From the additive decomposition of the global reward function, it follows immediately that the action-value function Q^π for the entire system, given policy π, is the sum of the subagents' action-value functions:

$$\begin{aligned} Q^\pi(s,a) &= E\left[r(s,a,s') + \gamma Q^\pi(s',\pi(s'))\right] \\ &= \sum_{j=1}^{n} Q_j^\pi(s,a) \end{aligned}$$

It also follows that an arbitrator that has to evaluate the global action-value function for a particular state s_i only needs to receive the vector $Q_j(s_i,\cdot)$ from each subagent.

For any suitable transition measure P, there is a unique fixed point $Q^* : \mathcal{S}\times\mathcal{A}\to\mathbb{R}$ satisfying the Bellman equation

$$Q^*(s,a) = E\left[\sum_{j=1}^{n} r_j(s,a,s') + \max_{a'\in\mathcal{A}} \gamma Q^*(s',a')\right] \quad (1)$$

The policy $\pi^* : \mathcal{S}\to\mathcal{A}$ corresponding to this action-value function is the optimal policy for the MDP.

3. Local reinforcement learning schemes

3.1. Local Q learning: The illusion of control

Suppose that each subagent's action-value function Q_j is updated under the assumption that the policy followed by the agent will also be the optimal policy with respect to Q_j. In this case, the value update is the usual Q-learning update (Watkins, 1989), but relies only on value information local to each subagent. In detail,

$$\begin{aligned} Q_j(s_t,a_t) \leftarrow (1-\alpha_j^{(t)})Q_j(s_t,a_t) + \\ \alpha_j^{(t)}\left[r_j(s_t,a_t,s_{t+1}) + \gamma \max_{a\in\mathcal{A}} Q_j(s_{t+1},a)\right] \end{aligned} \quad (2)$$

($\alpha_j^{(t)}$ is a learning rate that decays to 0 over time.) Observe that a_t is used only to associate the reward signal r_j with a particular action; the learner does not require a_t to evaluate the discounted future value $\gamma \max_{a\in\mathcal{A}} Q_j(s_{t+1},a)$.

For each j, denote the fixed point of this update procedure by $\tilde{Q}_j$ and the corresponding policy by $\tilde{\pi}_j$.

3.1.1. Convergence

Even though $\tilde{\pi}_j$ may not optimize the sum of rewards over all subagent values, and even though the arbitrator may never execute this policy, the following theorem demonstrates that this sort of off-policy update leads to the convergence of the Q_j estimates to a collection of locally greedy ("selfish") estimates.

Theorem 1. *(Theorem 4 in (Tsitsiklis, 1994).) Suppose that each $(s,a)\in\mathcal{S}\times\mathcal{A}$ is visited infinitely often. Under the update scheme described in equation (2), each Q_j will converge a.s. to a $\tilde{Q}_j$ satisfying*

$$\tilde{Q}_j(s,a) = \sum_{s'\in\mathcal{S}} P(s'|s,a)\left[R(s,a,s') + \gamma \max_{a'\in\mathcal{A}} \tilde{Q}_j(s,a)\right] \quad (3)$$

Tsitsiklis sets out several technical assumptions, all of which are satisfied in the current setting, and all of which are omitted for brevity. This update is analogous to that proposed by Stone and Veloso (1998), who assume that the action space may be partitioned into subspaces, one for each subagent, and that rewards realized by each subagent are independent given only the local action of each subagent.

3.1.2. Q learning for dollars

Suppose an agent in the dollars-and-euros world revises its action-value estimates according to the selfish Q update of equation (2). Let $\tilde{Q}_d$ denote the action-value function for the dollars subagent, and $\tilde{Q}_e$ the action value function for the euros subagent. The policy $s_0 \mapsto \textit{Left}$ optimizes $\tilde{Q}_d$:

$$\begin{aligned} \tilde{Q}_d(s_0,\textit{Left}) &= (1-\epsilon)\sum_{k=1}^{\infty}\epsilon^k\gamma^k \\ &= \frac{1-\epsilon}{1-\gamma\epsilon} \\ \tilde{Q}_d(s_0,\textit{NoOp}) &= \gamma\tilde{Q}_d(s_0,\textit{Left}) \\ &= \frac{(1-\epsilon)\gamma}{1-\epsilon\gamma} \\ \tilde{Q}_d(s_0,\textit{Right}) &= \epsilon\gamma\tilde{Q}_d(s_0,\textit{Left}) \\ &= \frac{\gamma\epsilon}{1-\epsilon\gamma} \\ \tilde{Q}_d(s_0,\textit{Up}) &= 0.6(1-\epsilon) + \gamma\epsilon\tilde{Q}_d(s_0,\textit{Left}) \\ &= 0.6(1-\epsilon) + \frac{\gamma\epsilon}{1-\epsilon\gamma} \end{aligned}$$

The values are symmetric for $\tilde{Q}_e$, with the optimal action for $\tilde{Q}_e$ from s_0 being $\textit{Right}$.

When the two subagent value functions are combined, a perverse thing happens. For certain values of ϵ and γ, the "optimal" behavior is $\textit{NoOp}$, even though an agent that never tries to escape can never achieve a reward:

$$\begin{aligned} \tilde{Q}(s_0,\textit{Left}) &= \tilde{Q}(s_0,\textit{Right}) \\ &= \frac{\epsilon+(1-\epsilon)}{1-\epsilon\gamma} \\ &= \frac{1}{1-\epsilon\gamma} \\ \tilde{Q}(s_0,\textit{NoOp}) &= \frac{2(1-\epsilon)\gamma}{1-\epsilon\gamma} \end{aligned}$$

When the discount factor is sufficiently large ($\gamma > 0.5$), the sum of selfish action-value estimates indicate that the agent is better off doing nothing than settling on one of the absorbing states. Informally, the dollars subagent prefers *Left*, but assigns a fairly high value to $\textit{NoOp}$ because it can go *Left* at the next time step. The euros agent likewise prefers *Right*, but assigns a fairly high value to $\textit{NoOp}$ because it can go *Right* after that. As a result, the expected value of receiving both the dollars and the euros on the next time step dominates the value of receiving one or the other on the current time step, even though only one or the other can occur. The "illusion of control" leads to an incorrect policy.

3.2. Local Sarsa: Global realism

In equation (2), each subagent updated its estimate of (discounted) future rewards by assuming that it could have exclusive control of the system. While this may be a useful approximation (as when value functions depend on subspaces of $\mathcal{S} \times \mathcal{A}$ with small overlap), it does not in general guarantee that $\tilde{Q}$ will have the same values, or yield the same policy, as Q^*. The relationship between $\tilde{Q}$ and Q^* is discussed below. For now, consider an alternative that gives a collection of Q_j functions whose sum converges to Q^*.

Let $Q_j^* : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ denote the contribution of the jth reward function to the optimal value function Q^* defined in Equation (1):

$$Q_j^*(s,a) = E\left[r_j(s,a,s') + \gamma Q_j^*(s', \pi^*(s'))\right] \qquad (4)$$

To converge to Q_j^*, the updates executed by each subagent must reflect the globally optimal policy. Update schemes that do this must replace the locally selfish updates described above with updates that are asymptotically greedy with respect to Q^*.

The Sarsa algorithm (Rummery & Niranjan, 1994), which requires on-policy updates, suggests one approach. Rather than allowing each subagent to choose the successor action it uses to compute its action-value update, each subagent uses the action a_{t+1} executed by the arbitrator in the successor state s_{t+1}:

$$\begin{aligned} Q_j(s_t,a_t) \leftarrow{} & (1-\alpha_j^{(t)})Q_j(s_t,a_t) + \\ & \alpha_j^{(t)}\left[r_j(s_t,a_t,s_{t+1}) + \gamma Q_j(s_{t+1},a_{t+1})\right] \end{aligned} \qquad (5)$$

This requires that the arbitrator inform each subagent of the successor action it actually followed, but the communication overhead for this is linear in the dimension of $\mathcal{A}$.

3.2.1. Convergence

Rummery and Niranjan establish that the Sarsa update enforces convergence to Q^* in the case when an agent maintains a single action-value function.

Lemma 1. *(Rummery & Niranjan, 1994) Suppose that an agent receives a single reward signal $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$, and maintains a corresponding Q function by the update*

$$\begin{aligned} Q(s_t,a_t) \leftarrow{} & (1-\alpha(t))Q(s_t,a_t) + \\ & \alpha(t)\left[r(s_t,a_t,s_{t+1}) + \gamma Q(s_{t+1},a_{t+1})\right] \end{aligned} \qquad (6)$$

If all $(s,a) \in \mathcal{S} \times \mathcal{A}$ are visited infinitely often, $\mathcal{S}$ and $\mathcal{A}$ are finite, and the policy pursued by the agent is greedy in the limit of infinite exploration, then under the update scheme of equation (6)*, Q will converge a.s. to Q^* as defined in equation* (1)*.*

The following theorem demonstrates that this update procedure yields estimates converging to Q_j^*, as defined in equation (4), when the arbitrator asymptotically chooses the optimal action.

Theorem 2. *Suppose that all $(s,a) \in \mathcal{S} \times \mathcal{A}$ are visited infinitely often and the policy pursued by the arbitrator is greedy in the limit of infinite exploration. Suppose also that $\mathcal{S}$ and $\mathcal{A}$ are finite. Under the update scheme of equation (5), each Q_j will converge a.s. to a Q_j^* satisfying equation (4).*

In order to guarantee that the local Sarsa update provides convergence to the global optimum, it suffices to observe that the local Sarsa update is just the monolithic Sarsa update in algebraic disguise:

$$\begin{aligned} \sum_{j=1}^{n} Q_j(s_t,a_t) \leftarrow{} & \sum_{j=1}^{n}(1-\alpha_j^{(t)})Q_j(s_t,a_t) + \\ & \sum_{j=1}^{n}\alpha_j^{(t)}\left[R_j(s_t,a_t,s_{t+1}) + \gamma Q_j(s_{t+1},a_{t+1})\right] \\ Q(s_t,a_t) \leftarrow{} & (1-\alpha^{(t)})Q(s_t,a_t) + \\ & \alpha^{(t)}\left[R(s_t,a_t,s_{t+1}) + \gamma Q(s_{t+1},a_{t+1})\right] \end{aligned}$$

The individual Q_j converge to Q_j^* because the policy under which they are updated converges to π^*, so that in the limit, they are updated under a fixed policy.

3.2.2. Sarsa for dollars

The local Sarsa update yields an intuitively correct policy for the dollars-and-euros world. The optimal policy is to choose *Up* from s_0, and the net value of *NoOp* in s_0 vanishes because the update method of equation (5) requires that all agents assume a single policy.

$$\begin{aligned} Q_d^*(s_0, \mathit{Left}) &= Q_e^*(s_0, \mathit{Right}) \\ &= (1-\epsilon) + \frac{0.6\gamma\epsilon(1-\epsilon)}{1-\gamma\epsilon} \\ Q_d^*(s_0, \mathit{Right}) &= Q_e^*(s_0, \mathit{Left}) \\ &= \frac{0.6\gamma\epsilon(1-\epsilon)}{1-\gamma\epsilon} \\ Q_d^*(s_0, \mathit{Up}) &= Q_e^*(s_0, \mathit{Up}) \\ &= \frac{0.6\epsilon}{1-\gamma\epsilon} \\ Q_d^*(s_0, \mathit{NoOp}) &= Q_e^*(s_0, \mathit{NoOp}) \\ &= \frac{0.6\gamma\epsilon(1-\epsilon)}{1-\gamma\epsilon} \end{aligned}$$

It is easily shown from these values that *Up* is optimal for all $\epsilon \in [0,1]$ and all $\gamma \in (0,1)$. In the undiscounted case, the supervisor incurs no penalty for repeatedly choosing *NoOp* for a very long time, but it will never achieve a reward if it does so forever.[2]

3.3. Remarks

Note that $\tilde{Q}_j$ provides an optimistic estimate of Q_j^* by definition of the selfish action-value function. The sum of selfish components is therefore an optimistic estimate of the optimal value over all components. This can lead to overestimates of future value, as in the case of the three-state gridworld. For equality of the selfish and globally-optimal policies, we require

$$Q^*(s,a) = \tilde{Q}(s,a)$$

because equivalent policies will converge to identical action-value estimates.

4. The racetrack world

4.1. Description

An agent seeking to circulate around a racetrack must trade off his speed against the cost in time and money of damaging his equipment by colliding with the wall or with other racers. However, these goals are opposed to one another: the safest race car driver is the one who never starts his first lap, and the fastest one looks to win at all costs.

Define a racetrack as a rectangular gridworld with an excluded region (the "infield") in the center, so that the surrounding open spaces (the "track") are of uniform width. Represent the state of an agent by its position (the (x, y) index of the grid square it currently occupies) and velocity (in squares per unit of time). At each time step, the agent may alter each component of its velocity by -1, 0, or $+1$ unit/second, giving a total of nine actions. Actions succeed with probability 0.9; the agent accelerates $45°$ to the left or right of the desired vector with probability 0.03 each, the agent accelerates $90°$ to the left or right of the desired vector with probability 0.01 each, and the agent does nothing with probability 0.02. If an agent collides with a wall, its position is projected back onto the track, and each component of its velocity is reduced by 1. If the agent does not accelerate away from the wall, it will continue to "slip" parallel to the wall, but will not cross into the infield.

An agent receives a reward of 10 for completing one lap, and a penalty of -1 for each collision with the wall. An agent also receives a shaping reward (Ng et al., 1999) proportional to the measure of the arc swept by its action.

4.2. Implementation

This paper evaluates a single racer on a 2-unit-wide track with a 4×3 infield. Training consisted of 4000 episodes, with fifty test episodes occurring after every fifty training episodes. To provide an incentive for finishing quickly, experiments assumed a discount factor of $\gamma = 0.9$. Exploration occurred uniformly at random; the exploration rate decreased from 0.25 to 0.0625 over the first 2500 episodes, and remained constant thereafter. The update step size α diminished from 0.3 to 0.01 over the 4000 episodes. Both training and test episodes were truncated at 200 steps; preliminary tests indicated that an agent could get stuck by hitting a wall at low speed, then choosing not to accelerate in any direction.

4.3. Results

Figure 2 compares the values achieved by trained using local Q and local Sarsa updates. This is not an ideal domain to illustrate the suboptimality of local Q updates, because the dynamics of the racetrack world couple the objectives of completing a lap quickly and avoiding collisions: not only does a racer suffer a negative reward when it collides with a wall, but it loses speed, which diminishes the discounted value of its eventual completion reward. However, the local Sarsa learner outperformed the local Q learner by almost 1 standard deviation after 4000 training episodes.

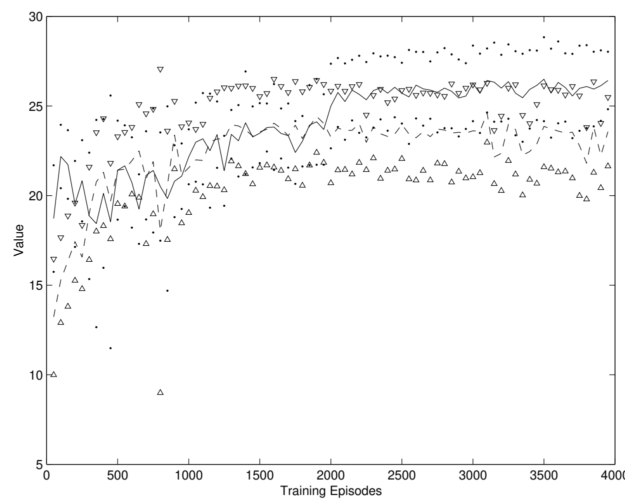

Figure 2. Values realized by local Q (dashed line) and local Sarsa (solid line) updates. Arrowheads indicate one standard deviation for local Q; dots indicate one standard deviation for local Sarsa. Episodes were truncated at 200 steps.

5. The fisheries problem

5.1. Description

Consider the problem of allocating resources in a commercial fishery. A commercial fishing fleet wishes to maximize the aggregate discounted value of its catch over time, which requires that it show at least some concern for the sustainability of the fish population. Individual fishermen, however, may choose to act selfishly and maximize their

[2] In MDP jargon, a supervisor that chooses to stay forever is pursuing an "improper" policy. Convergence still holds under certain restrictions on the reward functions; cf. (Bertsekas & Tsitsiklis, 1996).

own profit, assuming that others in the fleet will reduce their catch for the viability of the fishery. If all fishermen follow the selfish policy, the result is a "tragedy of the commons": fish stocks collapse and the fishery dries up.

To model this effect, and compare the performance of the local Q and Sarsa algorithms, consider a fishery with n boats that alternates between a "fishing" season and a "mating" season. Assume that a fish population of size $f(t)$ reproduces according to a density-dependent model (Ricker, 1954):

$$f(t+1) = f(t)\exp\left[R\left(1 - \frac{f(t)}{f_{\max}}\right)\right]$$

where R for a fish population without immigration or emigration is the difference between the birth rate and the death rate, and $f_{\max}$ is the "carrying capacity" of the environment (the population at which growth diminishes to 0). At the beginning of each fishing season, the fish are assigned to one of n regions with equal probability. Let $f_j(t)$ denote the number of fish in the jth region at time t, with $\sum_j f_j(t) = f(t)$. The "fisheries commissioner" (arbitrator) selects the proportion of the season $a_j(t)$ that each boat will fish, based on the total fish population $f(t)$. Let $\eta \in (0,1)$ denote the efficiency of each fishing boat. The number of fish caught by the jth boat, $c_j(t)$, is distributed $\mathrm{Poisson}(\eta a_j(t) f_j(t))$ and is capped at $f_j(t)$.[3] The reward realized by boat j at time t is given by

$$r_j(c_j(t), a_j(t)) = c_j(t) - \zeta a_j^2(t)$$

for some constant ζ. The cost of fishing increases quadratically with a_j to reflect the increase in crew and equipment fatigue over the course of a season.

5.2. Implementation

This paper evaluates $n = 10$ fishing boats, $f(0) = 1.5 \times 10^5$ fish, a population growth rate of $R = 0.5$, a carrying capacity of $f_{\max} = 2 \times 10^5$ fish, an efficiency of $\eta = 0.98$, and a maximum fishing cost of $\zeta = 10^3$. Experiments proceeded over 800 episodes, with each episode terminating when fewer than 200 fish remained, or when the fishery had survived 100 years.[4] With a discount factor of $\gamma = 0.9$, the contribution of the 100th episode to the initial value was reduced by a factor of 2.65×10^{-5}. The step size α for learning updates was 0.1 for the first 100 episodes, and 0.05 thereafter. Exploration occurred uniformly at random; the exploration rate decreased from 0.4 to 0.05 over the first 400 training episodes, and remained constant thereafter.

Each action-value estimate Q_j depended on three features: the current population $f(t)$, the proportion $a_j(t)$ fished by the jth boat, and the total proportion $a_{-j}(t) = \sum_{i \neq j} a_i(t)$ fished by the other boats, so that Q_j was defined over a 3-dimensional space, rather than an $(n+1)$-dimensional space.[5] A radial basis function approximator represented the action-value estimates Q_j, with values updated by bounded gradient steps.

5.3. Results

Presented in Figure 3 are the values achieved by the selfish and optimal learner in the fishery problem. Every ten training episodes, ten test episodes were executed and the values averaged. As anticipated, selfish updates quickly fell victim to the "tragedy of the commons" (Figure 4). Each boat exhausted the fish stocks in its region because the fishery had computed the value of this selfish policy without regard for the actions of the other boats. As a result, the fish population crashed within a couple of years. Concurrent Sarsa's "realistic" updates led to a sustainable policy. Each boat only harvested a fraction of the fish in its region, so enough remained for the population to recover in the next mating cycle.

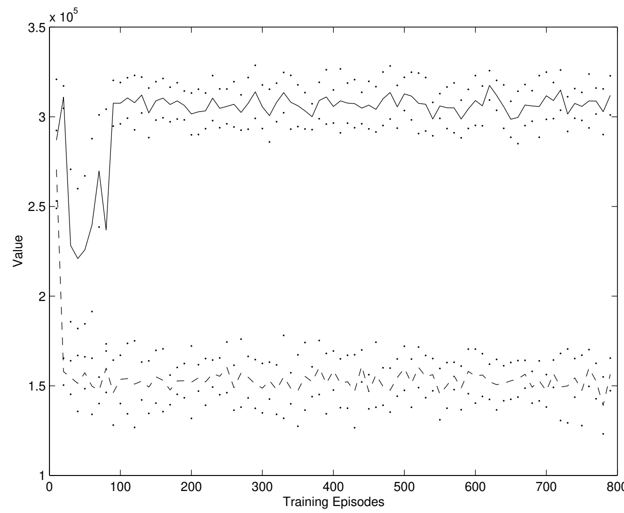

Figure 3. Values realized by local Q (dashed line) and local Sarsa (solid line) for the fishery world. Dots indicate one standard deviation.

6. Related work and conclusions

Q-decomposition extends the monolithic view of reinforcement learning in two directions: it identifies a natural decomposition of action-value estimates in terms of additive reward signals, and considers the tasks of action selection and value-function updates in terms of communication between an arbitrator and several subagents. Both directions provide broad avenues for future work.

[3] One can imagine fishing to be a queuing process wherein the fish line up to be caught.

[4] In early experiments, local Sarsa discovered a policy that had sustained the fishery for 20000 years, and showed no signs of dying out!

[5] Multinomial sampling to construct bins and Poisson sampling to simulate fishing both justify this aggregation.

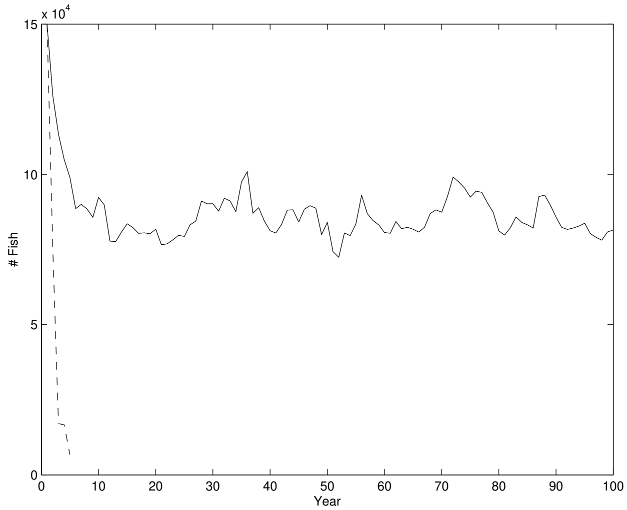

Figure 4. Characteristic depletion of fish stocks over one episode by local Q (dashed line) and local Sarsa (solid line).

The concept of Q-decomposition, and the corresponding notion of "subagents," suggests a particular method for representing value functions, or for aligning the interests of multiple subagents. Q-decomposition only requires that value function *updates* assume a particular form to guarantee optimal behavior of a reinforcement learning agent. In some cases, as in the case of the fishery world, this additive decomposition results in a more compact value function representation.

Other authors (Koller & Parr, 1999) have explored value function approximations that represent the true value function as a linear combination of a collection of basis functions, with each basis function defined over a small collection of state variables. In order to maintain these approximations, value function updates must be projected back onto the basis at every time step, because the evolution of the MDP over time can introduce dependencies not represented in the basis. However, some subagent rewards may be unconditionally independent of some components of the state, or may depend only on aggregate values, as in the fishery world. It may also be possible to realize representational savings by combining Q-decomposition with graphical representations of conditional utilities (Bacchus & Grove, 1995; Boutilier et al., 2001).

Devolution of value function updates also moves monolithic reinforcement learning into the "multi-body" setting, and suggests a spectrum of learning methods distinguished by the degree of communication between agents and a central arbitrator. At one extreme, traditional RL methods assume closely-coupled components: a single value function defining a monolithic policy. Q-decomposition allows for multiple value functions, each residing in a subagent, but still requires each subagent to report its value estimates to an arbitrator, and receive in turn the action that the arbitrator chooses. Further relaxations of the communications requirements of Q-decomposition include action decomposition and partial observability of actions. In the former case the action space $\mathcal{A}$ may be partitioned into subspaces $\mathcal{A}_j$ corresponding to subagent reward components; in the latter, subagents maintain histories of observations $\langle o(1), \ldots, o(t-1), o(t)\rangle$ from which they must estimate a_t to compute value updates.

The multiagent learning problem eliminates the central arbitrator, making optimality more difficult to achieve, but similar issues of communication between participants arise. Traditional game theory considers the uncommunicative extreme of this spectrum, where participants do not share policies or value functions. Claus and Boutilier (1998) have proposed the "individual learner" and "joint action learner" concepts to distinguish between agents in cooperative games that choose actions to maximize individual rewards, and agents that choose actions to maximize joint rewards. A joint action learner observes the actions of its peers, and maintains belief state about the strategies they follow, with the goal of maximizing the joint reward. An independent learner ignores the actions of its peers when "optimizing" its policy, analogous to a local Q learner. There is still no central arbitration mechanism, but inverse reinforcement learning techniques (Ng & Russell, 2000) might be used to deduce the policies of other agents and bridge the communications gap.

In treating only local Q learning and local Sarsa, this paper has evaluated two points in the continuum of possible representations. These fairly simple-minded approaches nonetheless provide compelling evidence of the value of Q-decomposition, and suggest a variety of future work.

References

Bacchus, F., & Grove, A. (1995). Graphical models for preference and utility. *Proceedings of the Eleventh International Conference on Uncertainty in Artificial Intelligence*. Montreal.

Bertsekas, D. P., & Tsitsiklis, J. N. (1996). *Neuro-dynamic programming*. Optimization and Scientific Computing. Belmont, MA: Athena Scientific.

Boutilier, C., Bacchus, F., & Brafman, R. I. (2001). UCP-networks: A directed graphical representation of conditional utilities. *Proceedings of the Seventeenth International Conference on Uncertainty in Artificial Intelligence*. Seattle.

Brooks, R. A. (1986). *Achieving artificial intelligence through building robots* (Technical Report A. I. Memo 899). MIT Artificial Intelligence Laboratory, Cambridge, MA.

Claus, C., & Boutilier, C. (1998). The dynamics of reinforcement learning in cooperative multiagent systems. *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*. Madison, WI.

Goldberg, K., Song, D., & Levandowski, A. (in press). Collaborative teleoperation using networked spatial dynamic voting. *Proceedings of the IEEE*.

Koller, D., & Parr, R. (1999). Computing factored value functions for policies in structured MDPs. *Proceedings of the 16th International Joint Conference on Artificial Intelligence* (pp. 1332–1339). Stockholm.

Lin, L.-J. (1993). Scaling up reinforcement learning for robot control. *Proceedings of the Tenth International Conference on Machine Learning*.

Maes, P. (1989). How to do the right thing. *Connection Science Journal*, *1*, 291–323.

Ng, A. Y., Harada, D., & Russell, S. J. (1999). Policy invariance under reward transformations: Theory and application to reward shaping. *Proceedings of the Sixteenth International Conference on Machine Learning* (pp. 278–287). Bled, Slovenia.

Ng, A. Y., & Russell, S. (2000). Algorithms for inverse reinforcement learning. *Proceedings of the Seventeenth International Conference on Machine Learning*.

Ogasawara, G. H. (1993). *RALPH-MEA: A real-time, decision-theoretic agent architecture*. Doctoral dissertation, University of California, Berkeley.

Pirjanian, P. (2000). Multiple objective behavior-based control. *Journal of Robotics and Autonomous Systems*.

Ricker, W. E. (1954). Stock and recruitment. *Journal of the Fisheries Research Board of Canada*, *11*, 624–651.

Rosenblatt, J. K. (2000). Optimal selection of uncertain actions by maximizing expected utility. *Autonomous Robots*, *9*, 17–25.

Rummery, G. A., & Niranjan, M. (1994). *On-line Q-learning using connectionist systems* (Technical Report CUED/F-INFENG/TR 166). Cambridge University Engineering Department.

Russell, S. J. (1991). An architecture for bounded rationality. AAAI Symposium on Integrated Architectures for Intelligent Agents.

Saffiotti, A., Konolige, K., & Ruspini, E. (1995). A multivalued logic approach to integrating planning and control. *Artificial Intelligence*, *76*, 481–526.

Stone, P., & Veloso, M. (1998). Team-partitioned, opaque-transition reinforcement learning. *Proceedings of the Fifteenth International Conference on MAchine Learning*.

Tsitsiklis, J. N. (1994). Asynchronous stochastic approximation and Q-learning. *Machine Learning*, *16*, 185–202.

Watkins, C. J. (1989). *Learning from delayed rewards*. Doctoral dissertation, University of Cambridge, England.