Coco-Q: Learning in Stochastic Games with Side Payments

Eric Sodomka, Elizabeth M. Hilliard, Michael L. Littman, Amy Greenwald
SODOMKA@CS.BROWN.EDU, BETSY@CS.BROWN.EDU, MLITTMAN@CS.BROWN.EDU, AMY@CS.BROWN.EDU
Brown University, 115 Waterman Street, Providence, RI 02912-1910

Abstract

Coco ("cooperative/competitive") values are a solution concept for two-player normal-form games with transferable utility, when binding agreements and side payments between players are possible. In this paper, we show that coco values can also be defined for stochastic games and can be learned using a simple variant of Q-learning that is provably convergent. We provide a set of examples showing how the strategies learned by the Coco-Q algorithm relate to those learned by existing multiagent Q-learning algorithms.

1. Introduction

The field of reinforcement learning (Sutton & Barto, 1998) is concerned with agents that improve their behavior in sequential environments through interaction. One of the best known and most versatile reinforcement-learning (RL) algorithms is Q-learning (Watkins & Dayan, 1992), which is known to converge to optimal decisions in environments that can be characterized as Markov decision processes.

Q-learning is best suited for single-agent environments; nevertheless, it has been applied in multi-agent environments, with varying degrees of success (Sandholm & Crites, 1995; Gomes & Kowalczyk, 2009). Minimax-Q (Littman, 1994) is a version of Q-learning for two-agent zero-sum games, which converges to optimal (meaning, minimax) decisions for these games.

In general-sum games, however, the learning problem has proven much more challenging. Nash-Q (Hu & Wellman, 2003) is an attempt to use Q-learning in the general setting, but its update rule is inefficient and it lacks meaningful convergence guarantees (Bowling, 2000; Littman, 2001). Correlated-Q (Greenwald & Hall, 2003) is an improvement over Nash-Q in that, in exchange for access to a correlating device, its update rule is efficient. However, there are environments for which correlated-Q is unable to converge to stationary, optimal decisions (Zinkevich et al., 2005).

As in the correlated-Q work, we consider what can be achieved if agents are afforded extra power. Instead of a correlating device, we consider the impact of *binding agreements* on our ability to design learning algorithms that converge to stationary, optimal decisions.

In Section 2, we provide the necessary background on coco values as a solution concept for normal-form games. In Section 3, we generalize coco values to stochastic games, and prove that a variant of Q-learning based on this concept converges. Section 4 introduces a set of grid games and compares and contrasts the solutions that result from the coco concept to the aforementioned multi-agent learning algorithms.

2. Coco Values in Normal-Form Games

Coco values were introduced by Kalai & Kalai (2010) as a solution to two-player normal-form games where players have *transferable utility* (TU): a common currency equally valued by both players. The coco-value solution concept takes advantage of the extra powers players have when making binding agreements that specify a joint action and include a *side payment*—a transfer of utility from one player to the other.

Coco gets its name from the way it is calculated: by decomposing the game into a cooperative (or team) game and a competitive (or zero-sum) game, and then combining the solutions to these games.

Consider a scenario where you and a short, but strong, friend are picking bananas. Your friend cannot reach any bananas, and you can only reach two. If you try to climb on your friend to reach more bananas, you fail. However, if your friend is willing to give you a boost, you can climb and get two high bananas, as well as the two low ones. This game is depicted in Figure 1.

Proceedings of the 30th International Conference on Machine Learning, Atlanta, Georgia, USA, 2013. JMLR: W&CP volume 28. Copyright 2013 by the author(s).

		You	
		reach	climb
Friend	don't boost	$0, 2$	$0, 0$
	boost	$0, 2$	$0, 4$

Figure 1. A banana-picking game.

If utility is not transferable, your friend has no incentive to help you pick bananas. But, if it is, then offering her some bananas might encourage her to give you a boost. How many of the four bananas you pick should you offer her? One bite? All four? COCO values offer a focal point—you should offer her one banana for her efforts. The COCO values of this game are $(1, 3)$.

2.1. Operators on Values

A two-player normal-form game $\langle A, \overline{A}, U, \overline{U} \rangle$ is played by a player, *Ego*, and another player, *Alter*. Here, A is Ego's set of actions and $\overline{A}$ is Alter's. The values for the game are defined by U for Ego and $\overline{U}$ for Alter. For action $a \in A$ and $\overline{a} \in \overline{A}$, $U(a, \overline{a})$ is Ego's value and $\overline{U}(a, \overline{a})$ is Alter's.

We now define a few operators on values. The *maxmax* (or friend) operator finds the highest possible U value:

$$\text{maxmax}(U) = \max_{a \in A} \max_{\overline{a} \in \overline{A}} U(a, \overline{a}).$$

This operation is computationally straightforward: simply maximize over all the entries in U.

The *minmax* (or foe) operator finds the highest possible worst-case value for Ego:

$$\begin{aligned} &\text{minmax}(U) \\ &\quad = \max_{\pi \in \Pi(A)} \min_{\overline{\pi} \in \Pi(\overline{A})} \sum_{a \in A} \sum_{\overline{a} \in \overline{A}} \pi(a) \overline{\pi}(\overline{a}) U(a, \overline{a}). \end{aligned}$$

Here, $\Pi(X)$ represents the space of probability distributions over a discrete set X. Note that $\text{minmax}(-U) = -\text{minmax}(U)$. This operator is also relatively straightforward to compute, as it can be calculated in polynomial time using linear programming.

The *Nash* operator selects a value for Ego according to a Nash equilibrium. Specifically,

$$\text{Nash}(U, \overline{U}) = \sum_{a \in A} \sum_{\overline{a} \in \overline{A}} \pi(a) \overline{\pi}(\overline{a}) U(a, \overline{a}),$$

where π and $\overline{\pi}$ are probability distributions that constitute some Nash equilibrium of the bimatrix game. This operator is more challenging to compute, as the problem of computing any Nash equilibrium is now known to be PPAD-complete (Chen & Deng, 2006) and NP-hard for the welfare-maximizing equilibrium (Gilboa & Zemel, 1989). Perhaps worse, the value of the operator is not well defined, as there can be multiple conflicting Nash equilibria.

The *CE* operator is analogous, but it selects its value using a correlated equilibrium. Specifically,

$$\text{CE}(U, \overline{U}) = \sum_{a \in A} \sum_{\overline{a} \in \overline{A}} \pi(a, \overline{a}) U(a, \overline{a}),$$

where π is a probability distribution over joint actions constituting a correlated equilibrium of the bimatrix game. In this paper, we restrict our attention to the correlated equilibrium operator that maximizes total welfare: that is, the sum of the players' values. Like minmax, this operator can be computed in polynomial time using linear programming. It need not produce a unique value, however, as there can be multiple correlated equilibria with the same total welfare but a different allotment of values for the two players.

2.2. Coco Value Operator

With this notation established, we can write the COCO value of a bimatrix game as an operator as follows:

$$\begin{aligned} &\text{Coco}(U, \overline{U}) \\ &\quad = \text{maxmax}((U + \overline{U})/2) + \text{minmax}((U - \overline{U})/2). \end{aligned}$$

For all other operators, the players' joint actions are chosen from the set that yields (one of) the operator's values. In the case of COCO, the players play a welfare-maximizing joint action:

$$(a^*, \overline{a}^*) \in \underset{(a, \overline{a})}{\text{argmax}} (U(a, \overline{a}) + \overline{U}(a, \overline{a})).$$

Then, to their respective values, they add a transfers (or side payment), which, for each player, amounts to the difference between its COCO value and its share of the welfare-maximizing values: The side payments P, received by Ego, and $\overline{P}$, received by Alter, are:

$$P = \text{Coco}(U, \overline{U}) - U(a^*, \overline{a}^*),$$

$$\overline{P} = \text{Coco}(U, \overline{U}) - \overline{U}(a^*, \overline{a}^*).$$

A positive P indicates a payee, and a negative P indicates a payer. Payments balance because $P = -\overline{P}$.

To solidify how exactly COCO values are computed, consider the following example (Figures 2 and 3).

The original game f (Figure 2) decomposes into a team game and zero-sum game (Figure 3). We say "decomposes" because the sums of the team and zero-sum

		Alter $\overline{a}$
Ego	a_1	$1, 0$
	a_2	$0, 4$

		Alter $\overline{a}$
Ego	a_1	$0, 0$
	a_2	$1, 5$

Figure 2. Left: Game f: $(U, \overline{U})$, a general-sum game. Right: Game f': another general-sum game.

		Alter $\overline{a}$
Ego	a_1	0.5
	a_2	2

		Alter $\overline{a}$
Ego	a_1	0.5
	a_2	-2

Figure 3. Left: $(U+\overline{U})/2$, a team game. Right: $(U-\overline{U})/2$, a zero-sum game. Jointly, a decomposition of game f.

game values match those for Ego in the original game and the differences match those for Alter. The COCO values are then the sum of the solutions of these two games, namely $(2, 2)+(.5, -.5) = (2.5, 1.5)$. To achieve these values, the agents play $(a_2, \overline{a})$ and receive values $(0, 4)$, and then Alter pays Ego 2.5. Ego wields a great deal of power in this game, and only agrees to play a_2 because of the promise of substantial side payments.

Note that it would be rational, and indeed a Nash equilibrium, for Ego to accept a smaller value, say .5, or for Alter to offer a larger side payment, say 3.5. COCO values offer a "recommended focal point" in games where side payments and binding agreements support many Nash equilibria (Kalai & Kalai, 2012).

Because COCO values are based on side payments, they also allow players to threaten each other. For example, if Ego's values were reversed for a_1 and a_2, making it rational for Ego to play a_2, the COCO values would be $(2.5, 2.5)$. Alter would still need to pay Ego to safeguard against Ego's incredible threat of playing a_1.

2.3. Coco Value Properties

Though relatively simple, the COCO operator has some remarkable properties. First, since it is a combination of the maxmax and minmax operators, both of which can be computed efficiently, COCO too can be computed efficiently. Further, since each of their values is unique, so too is COCO's.

Next, observe that the sum of the players' COCO values is the maximum joint value possible:

$$\begin{aligned}
&\text{Coco}(U, \overline{U}) + \text{Coco}(\overline{U}, U) \\
&= \text{maxmax}((U + \overline{U})/2) + \text{minmax}((U - \overline{U})/2) + \\
&\quad \text{maxmax}((\overline{U} + U)/2) + \text{minmax}((\overline{U} - U)/2) \\
&= \text{maxmax}(U + \overline{U}). \qquad (1)
\end{aligned}$$

On the other hand, the difference between the two COCO values is the minmax of the difference:

$$\begin{aligned}
&\text{Coco}(U, \overline{U}) - \text{Coco}(\overline{U}, U) \\
&= (\text{maxmax}((U + \overline{U})/2) + \text{minmax}((U - \overline{U})/2)) - \\
&\quad (\text{maxmax}((\overline{U} + U)/2) + \text{minmax}((\overline{U} - U)/2)) \\
&= \text{minmax}(U - \overline{U}). \qquad (2)
\end{aligned}$$

The COCO value is also known (Kalai & Kalai, 2012) to be the only value that satisfies a set of five desirable axioms, which includes Pareto efficiency.

Next, we show how the COCO operator can be applied to stochastic games.

3. Coco Values in Stochastic Games

A two-player stochastic game (Shapley, 1953) $\langle S, A, \overline{A}, T, R, \overline{R}, \gamma \rangle$ is defined by a state space S, action spaces A and $\overline{A}$ for Ego and Alter, a transition function T mapping states and joint actions to probability distributions over states, reward functions R and $\overline{R}$ mapping states and joint actions to rewards for the two players, and a discount factor γ.

If the number of actions available to Alter is one ($|\overline{A}| = 1$), the environment is a Markov decision process (MDP) (Puterman, 1994) and Ego's objective is to maximize its expected discounted future reward.

3.1. Generalized Q-learning in Games

Q-learning (Watkins & Dayan, 1992) learns behavior from experience while acting in an MDP. We define a generalized Q-learning algorithm for 2-player games. Each player keeps an estimate (Q values) of the expected future discounted reward starting in state s and taking joint action $(a, \overline{a})$. Ego's estimate is written $Q_s(a, \overline{a})$ or Q_s, and Alter's, $\overline{Q}_s(a, \overline{a})$ or $\overline{Q}_s$. This estimate is updated based on the agent's experience in its environment.

Let $\langle s, a, \overline{a}, r, \overline{r}, s' \rangle$ be an experience tuple that reflects that the players observe a transition from state s to state s' after Ego takes action a and Alter $\overline{a}$. The players receive values r and $\overline{r}$, respectively. Generalized Q-learning updates Q and $\overline{Q}$ as follows:

$$\begin{aligned}
Q'_s &= Q_s + \alpha(r + \gamma \otimes (Q_{s'}, \overline{Q}_{s'}) - Q_s), \\
\overline{Q}'_s &= \overline{Q}_s + \alpha(\overline{r} + \gamma \otimes (\overline{Q}_{s'}, Q_{s'}) - \overline{Q}_s).
\end{aligned}$$

Q' and $\overline{Q}'$ represent the updated versions of Q and $\overline{Q}$, α is a learning rate, and γ a discount factor.

In general, it is desirable for the Q values to converge to the solution of the following system of equations,

which we call the *solution of the game*. Given operator $\otimes$, for all s, a, and $\overline{a}$,

$$Q_s(a,\overline{a}) = R_s(a,\overline{a}) + \gamma \sum_{s'} T(s,a,\overline{a},s') \otimes (Q_{s'}, \overline{Q}_{s'}), \tag{3}$$

and likewise for $\overline{Q}$.

Depending on the choice of $\otimes$, different solutions, and correspondingly, different learning algorithms, result. In an MDP, $\otimes$ = max recovers the original Q-learning algorithm, which converges to the corresponding solution of the game. In a zero-sum game ($R = -\overline{R}$), $\otimes$ = minmax, resulting in an algorithm called minimax-Q or foe-Q. In a team game ($R = \overline{R}$), $\otimes$ = maxmax, resulting in an algorithm called friend-Q (Littman, 2001). Each of friend-Q and foe-Q converge to the corresponding solution of the game.

For general-sum games, the situation is more complex. Nash-Q uses $\otimes$ = Nash and CE-Q uses $\otimes$ = CE. While both definitions are plausible, the Q function alone does not contain enough information to identify a solution of the game (Zinkevich et al., 2005).

Existing proofs of convergence of variants of Q-learning make use of the following result. If $\otimes$ is a *non-expansion*, then Equation 3 has a unique solution and Equation 3 converges to it (Littman & Szepesvári, 1996). For $\otimes$ to be a non-expansion, we need the following to be true for all functions f and f',

$$|\otimes f - \otimes f'| \leq \text{maxmax}\,|f - f'|.$$

That is, the summaries of two functions should be no further apart than the functions themselves. Operators max, maxmax, and minmax are all non-expansions, and convergence is guaranteed. Operators Nash and CE are not non-expansions and convergence is not guaranteed.

The Coco operator is not a non-expansion. To see why, compare game f in Figure 2(Left) to game f' in Figure 2(Right). For this pair of games, we have $\text{maxmax}\,|f-f'| = 1$; the values between corresponding values in the two games differ by at most one. In Section 2.2, we show that Coco $f = (2.5, 1.5)$. By the same process, Coco $f' = (3, 3)$. The largest difference between COCO values is 1.5, so the Coco operator is *not* a non-expansion.

Although, the Coco operator is not a non-expansion, Coco-Q, the Q-learning algorithm that arises from defining $\otimes$ = Coco, still converges! We prove this result next. Then, in the remainder of the paper, we describe experimental results that demonstrate that Coco-Q learns sound policies.

3.2. Convergence of Coco-Q

Let Q and $\overline{Q}$ be a pair of initial functions.[1] We will maintain two auxiliary functions, Z (zero-sum) and C (common interest/team), defined as $Z_s = (Q_s - \overline{Q}_s)/2$ and $C_s = (Q_s + \overline{Q}_s)/2$.

Let $\langle s, a, \overline{a}, r, \overline{r}, s' \rangle$ be an experience tuple, which we will use to update the Q values as follows:

$$Q'_s = Q_s + \alpha_{a,\overline{a}}(r + \gamma\,\text{Coco}(Q_{s'}, \overline{Q}_{s'}) - Q_s);$$

$$\overline{Q}'_s = \overline{Q}_s + \alpha_{a,\overline{a}}(\overline{r} + \gamma\,\text{Coco}(\overline{Q}_{s'}, Q_{s'}) - \overline{Q}_s);$$

$$Z'_s = Z_s + \alpha_{a,\overline{a}}((r-\overline{r})/2 + \gamma\,\text{minmax}(Z_{s'}) - Z_s);$$

$$C'_s = C_s + \alpha_{a,\overline{a}}((r+\overline{r})/2 + \gamma\,\text{maxmax}(C_{s'}) - C_s).$$

We claim that the following property holds. If $Z_s = (Q_s - \overline{Q}_s)/2$ and $C_s = (Q_s + \overline{Q}_s)/2$, then $Z'_s = (Q'_s - \overline{Q}'_s)/2$ and $C'_s = (Q'_s + \overline{Q}'_s)/2$. That is, the updates maintain this relationship between Q, $\overline{Q}$, Z, and C. The relationship for Z'_s holds because

$$\begin{aligned}
&(Q'_s - \overline{Q}'_s)/2 \\
&\quad = (Q_s + \alpha_{a,\overline{a}}(r + \gamma\,\text{Coco}(Q_{s'}, \overline{Q}_{s'}) - Q_s))/2 - \\
&\qquad (\overline{Q}_s + \alpha_{a,\overline{a}}(\overline{r} + \gamma\,\text{Coco}(\overline{Q}_{s'}, Q_{s'}) - \overline{Q}_s))/2 \\
&\quad = (Q_s - \overline{Q}_s)/2 + \alpha_{a,\overline{a}}((r-\overline{r})/2 + \\
&\qquad \gamma\,\text{minmax}((Q_{s'} - \overline{Q}_{s'})/2) - (Q_s - \overline{Q}_s)/2) \\
&\quad = Z_s + \alpha_{a,\overline{a}}((r-\overline{r})/2 + \gamma\,\text{minmax}(Z_{s'}) - Z_s) \\
&\quad = Z'_s.
\end{aligned}$$

(The second equality follows from Equation 2.) The relationship for C' holds by analogous reasoning.

Given this relationship, it is not necessary to explicitly maintain all four functions. We could keep track of Q and $\overline{Q}$ and, at any time, produce $Z_s = (Q_s - \overline{Q}_s)/2$ and $C_s = (Q_s + \overline{Q}_s)/2$. Or, we could keep track of Z and C and, at any time, produce and $Q_s = C_s + Z_s$ and $\overline{Q}_s = C_s - Z_s$. Although the natural implementation is the former, the latter is the one for which convergence is evident. In particular, note that the Z update is precisely minimax-Q on the reward function $(R_s - \overline{R}_s)/2$, which converges. Similarly, the C update is precisely friend-Q on the reward function $(R_s + \overline{R}_s)/2$, which also converges. Since the sum and difference of convergent sequences converge, Coco-Q converges.

In fact, viewed from another perspective, it is not at all surprising that Coco-Q converges. A stochastic game is just a succinct representation of a normal-form game in which the two players select policies. As such, we have every reason to expect that the COCO operator

[1] Note that $\overline{Q}$ can come from Alter's estimates, or Ego can maintain its own independent copy.

would apply to stochastic games and retain its formal properties. Indeed, the COCO values of a general-sum stochastic game decompose into the sum and difference of the solutions of the corresponding team and zero-sum stochastic games, analogously to the normal-form case discussed by Kalai & Kalai (2010).

4. Coco Values in Grid Games

As shown above, Coco-Q converges to the set of values defined by Equation 3. We now introduce a set of sample stochastic games and analyze their corresponding values. The results we present use value iteration instead of Q-learning to derive the Q functions, as the former converges faster and with less noise, making it a useful tool for understanding the behavior of Coco-Q.

4.1. Grid Game Specification

A grid game is played on a grid of $m \times n$ squares. Each agent has associated with it a starting square on the grid and a (possibly empty) set of goal squares, where it receives rewards. Agents can observe their own and others' positions in the grid, as well as walls and semi-walls, which impede movement to varying degrees. At each time step, all agents simultaneously choose an action from the set {up, down, left, right, stick}. Every action except stick incurs a step cost, even if the agent is unable to move as intended.

If an agent's selected move is unimpeded, the agent moves in the direction specified by that move. If an agent tries to move through a wall, or to a square that is already occupied by an agent who sticks, the agent remains in its current square. If an agent tries to move through a semi-wall, it will do so with probability p; otherwise, it remains in its current square. If both agents try to move into the same square, including goals, at the same time, one of the agents is chosen—uniformly at random—to do so; the other remains in its current square.

The game ends once either agent reaches one of its goal squares. If multiple agents reach their respective goal squares simultaneously, they all receive their respective rewards. In our experiments, unless otherwise specified, we set goal rewards = 100, step costs = -1, and $p = 0.5$. Initially, we set the discount factor $\gamma = 1$, for ease of interpretation of the values; later, we set $\gamma = 0.95$ to illustrate what can happen when agents prefer to reach their goal sooner than later.

A *policy* π is tuple of mappings, one per player, from states to actions. A *trajectory* for a joint policy π is a possible sequence of states and actions that could arise when agents play π. We denote by V_i^π player i's expected discounted reward at its start state under joint policy π, dropping the superscript π when it is clear from context.

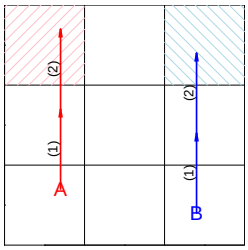
Figure 4(a) shows a sample grid game with agent trajectories overlaid. In this and all subsequent games, agents are depicted on the grid by A and B. Agent goals are shown as squares with diagonal lines passing through them (from the lower-left to upper-right corner for A's, and from the upper-left to lower-right corner for B's). If both agents have a goal in the same square, both sets of diagonal lines are shown. Trajectories are shown as a sequence of arrows pointing from the agent's current square to its next square. Any time an agent moves to a new square, the corresponding arrow of the trajectory is labeled with the time step. A stick action is shown as a circle. If the agent attempts to move to a square but cannot because of an obstacle, an arrow is shown in the direction the agent attempted to move, but the arrow stops at the square at which the agent was stopped. Further, if side payments occur, underneath the grid we show the payment that was made to A at each step, so that positive values mean A received a positive payment. (B's side payment is the negative of this number.) Concretely, in the game shown in Figure 4(a), each agent has a single goal two squares above its starting square. If both agents play up twice, then both agents will reach their goals at the same time, each receiving a goal reward of 100 minus a step cost of 2 (assuming $\gamma = 1$).

4.2. Example Games

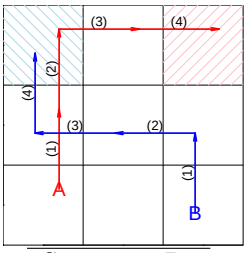
We now present some specific grid games designed to illustrate properties of COCO values. We compare the COCO policies to those of Correlated-VI, the result of solving Equation 3 with $\otimes$ = CE. Specifically, we use the *utilitarian* variant of CE (Greenwald & Hall, 2003), which, in the case of multiple correlated equilibria, chooses an equilibrium that maximizes the sum of the agents' rewards.

Coordination In Coordination (Figure 4(b)), A and B have to cross paths without colliding to reach their goals, which are diagonally across the grid from their starting square. While Q-learning has been shown to have poor performance in this game, Nash-Q and CE-Q, which don't use transfers, have been shown to coordinate to reach the efficient outcome (Hu & Wellman, 2003; Greenwald & Hall, 2003), in which $V_A = V_B = 96$ (assuming $\gamma = 1$).

COCO also takes actions that bring agents directly to their goals (and, hence, yield rewards of 96 for both), making interesting side payments along the way. In the particular trajectory shown in Figure 4(b), in step



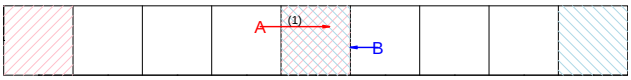
(a) An example trajectory for an example game



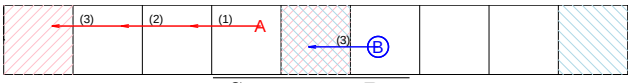
Step	P_A
1	0.00
2	−0.50
3	0.50
4	0.00

(b) COCO on Coordination. $\gamma = 1$.

Figure 4. A possible COCO trajectory in the Coordination grid game. This trajectory corresponds to one of many possible COCO policies, all of which have the same values.



(a) Correlated-VI on Prisoner



Step	P_A
1	50.00
2	−49.00
3	0.00

(b) COCO on Prisoner. $\gamma = 1$.

Figure 5. A Correlated-VI trajectory and the unique COCO trajectory in the Prisoner grid game.

1, both agents move up with no transfers, but in step 2, A pays B a small amount for B to move left while A moves up onto B's goal. B is being (temporarily) compensated for assuming a vulnerable position: In the resulting state, A can guarantee B never reaches its goal, while B cannot do the same to A. Once A reaches a square where it is no longer a threat to B, it gets its side payments back from B.

Prisoner Figure 5 depicts Prisoner, a grid game version of the well-known normal-form game, the prisoners' dilemma. Each agent has its own goal at one end of the grid, and there is a shared goal in the center.

This grid game resembles the prisoners' dilemma because moving to the shared goal ("defect") is a dominant strategy for each agent: if A is moving to the shared goal, B prefers to also move to the shared goal, and possibly win the collision tiebreaker, since otherwise the game ends before B can reach either goal. If A moves toward its own goal ("cooperate"), B still prefers to move to the shared goal since that way it immediately reaches a goal without incurring any additional step costs or wasting any time. However, the agents each receive higher expected value when they both cooperate than when they both defect.

Figure 5(a) depicts the Correlated-VI policy. Both agents play their dominant strategy, and each gets into the shared goal with probability 0.5, so that $V_A = V_B = 49 = (100 - 2)(0.5)$ (assuming $\gamma = 1$).

Figure 5(b) shows the unique COCO policy in this game.[2] Under this policy, B sticks in place for two steps while A proceeds to its goal, and then both agents walk into their goals simultaneously. The sum of the agents' values is 196 instead of 98. Note that this solution is analogous to what is computed by FolkEgal (Munoz de Cote & Littman, 2008).

The transfer payments made by the COCO strategy are of particular interest. First, B pays A to take a step to the left while B sticks. B has to pay A so that A will move into a more vulnerable position. A is now two steps from a goal, while B is only one step away. Second, A pays B 49 to stick instead of moving into the shared goal, giving A time to move next to its own goal. Finally, when A and B are both able to step into their respective goals, no side payments are made. The final values are 98 for A and 98 for B.

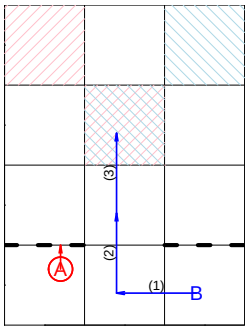
It is noteworthy that the players' final values are equal in this game. Even though the two players adopt different roles—one approaching the near goal and one moving to a distant goal—their expected discounted rewards end up the same. Values are not equal for the two players under FolkEgal or Correlated-VI. To achieve equal values under the policies they produce, it would be necessary to average across policies, with the two players switching roles. Although space does not permit a full proof here, this desirable property is general to COCO values—players in symmetric games have symmetric values.

Turkey In Turkey[3] (Figure 6), A and B have their own goals, in the top left and top right corners, respectively, and a shared goal near the center of the grid. The thick dashed lines on the grid represent semi-walls.

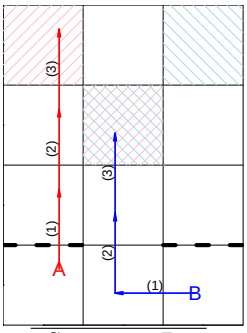
By way of comparison, Figure 6(a) shows a possible Correlated-VI trajectory for one possible Correlated-VI policy (with values $V_A = 43.20$ and $V_B = 87.40$). In

[2] Modulo exchanging roles.

[3] a variant of Chicken

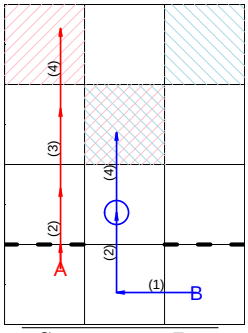


(a) Correlated-VI on Turkey. $V_A = 43.2$, while $V_B = 87.4$.



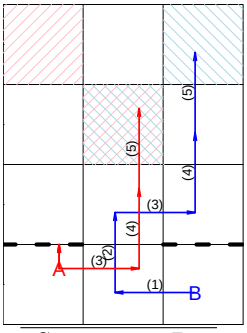
Step	P_A
1	22.10
2	0.00
3	0.00

(b) COCO on Turkey, trajectory 1



Step	P_A
1	22.10
2	0.50
3	−49.00
4	0.00

(c) COCO on Turkey, trajectory 2



Step	P_A
1	22.10
2	0.50
3	−49.50
4	0.00
5	0.00

(d) COCO on Turkey, trajectory 3

Figure 6. The three possible trajectories for COCO in grid game Turkey. $\gamma = 0.95$.

this trajectory, B takes the shortcut around the semi-wall, while A tries to pass through the other, but fails. After one failed attempt, A sticks, since B will reach the shared goal and the game will end before A can reach its own goal. Had A been successful at passing through the semi-wall, it would have gone directly to its goal and reached it at the same time as B.

Interestingly, there are exactly three possible trajectories for COCO. Figure 6(b) shows the first such trajectory (which occurs with probability p). This trajectory is the same as what happens in Correlated-VI when A succeeds at passing through the semi-wall: both players march to their goals. The difference between COCO and Correlated-VI is that, in COCO, a transfer payment is made from B to A to compensate A for the riskier route. When A happens to make it through the semi-wall, no further transfer payments are made, and A receives more total value than is possible in a game without transfer payments (at B's expense).

Figure 6(c) shows what happens when A fails to make it through the semi-wall after the first attempt: unlike Correlated-VI, which sticks, under COCO, A makes another attempt to pass through the semi-wall. In this trajectory, it succeeds the second time (this happens with probability p, so this trajectory occurs with probability p^2), and then makes a transfer to B so that B waits one additional turn for A to catch up, at which point they walk into their goals simultaneously without further side payments. Note that B actually pays a small amount to A to make that second attempt at passing through the semi-wall. It does so because the alternative of A moving right on step two would necessitate that B pursue its own goal instead of the shared goal, which would incur additional step costs for B.

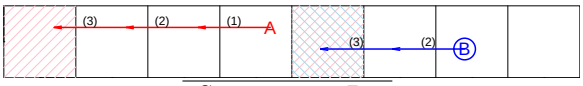
In the third and final possible trajectory for COCO (Figure 6(d)), A does not succeed at getting through the wall on the first or second attempt. At that point, B decides to pursue its own goal, and let A pursue the shared goal, rather than risk another block from the semi-wall. For B's added step cost, A pays B a slightly higher amount than it did when it passed through the semi-wall successfully on the second step. This trajectory occurs with probability $1 - p - p^2$.

Table 1 derives V_A and V_B for Turkey, by calculating an expected value of V_A and V_B across trajectories. The column labeled *Probability* gives the probability of each trajectory, and the columns labeled A and B, respectively, give the values of V_A and V_B for each player for each trajectory. Once again, we see a symmetric game, even in the face of asymmetric roles, resulting in symmetric values.

Trajectory	Probability	A	B
(b)	0.5	109.5	65.3
(c)	0.25	60.4	104.6
(d)	0.25	54.8	99.0
Expected Value	—	83.55	83.55

Table 1. Details of the V_A and V_B computation in Turkey when agents play COCO.

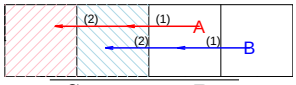
Friend or Foe Friend or Foe (Figure 7) is an asymmetric game in which not all goals have equal value. Specifically, A's individual goal on the extreme left side of the grid has a much higher reward than the shared goal near the center (a reward of 1,000 versus 100).



Step	P_A
1	−355.73
2	0.00
3	0.00

(a) COCO on Friend or Foe

Figure 7. The COCO trajectory for the unique deterministic policy in the Friend or Foe grid game. $\gamma = 0.95$.



Step	P_A
1	7.43
2	450.97

(a) COCO on Incredible

Figure 8. The COCO trajectory for the unique deterministic policy in the Incredible grid game. $\gamma = 0.95$.

Friend or Foe is distinct from the other games in that, without considering transfer payments, the game has no deterministic equilibrium. In the first round, if A moves left, B's best response is to also move left, since B is then guaranteed to reach its goal as quickly as possible. But, if B moves left, A has no hope of getting its large-valued goal, so it would be better off immediately moving to the shared goal on the right. If A immediately moves right, B might as well stick and not waste step costs. But, if B sticks on the first round, A would be better off moving left in pursuit of its large-valued goal.

Correlated-VI does not converge to a stationary policy in this instance, but rather it "converges" to a policy cycle of length two (Zinkevich et al., 2005). The agents' joint action at the start state oscillates.

COCO, in contrast, converges to a stationary policy. Under the COCO policy (see Figure 7), A pays B a share of the large goal value to stick as A moves left. Both agents are then two steps from a goal, and they move to them without making further side payments.

Incredible One potential issue with the COCO solution concept in normal-form games discussed by Kalai & Kalai (2012) is that players may not be incentivized to abide by a COCO policy. Figure 8 depicts the game Incredible, which illustrates this issue. In this game, B receives a larger value for reaching its goal than A receives for reaching its goal (a reward of 1,000 versus 100), but B's movement towards its goal is impeded by A. If A sticks, B is stuck with at most value 0.

The COCO policy prescribes that both players move left into their goals, but that B pay A to move left. However, even if B made no side payment, A should still move left towards its goal. B is essentially paying A over the incredible threat that A will stick.

4.3. Summary of Experiments

Figure 9 shows each player's values and the total values for the Nash-VI, Correlated-VI, and COCO policies for all of the grid games we discussed. As COCO values are welfare maximizing, no learning algorithm can achieve higher total values.

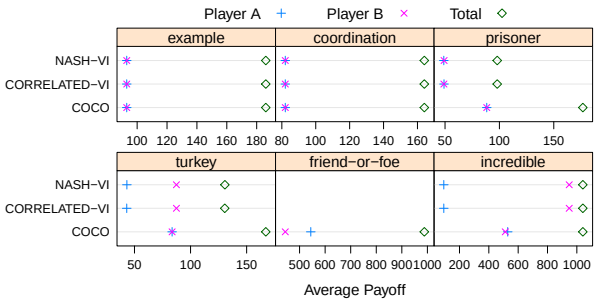


Figure 9. Expected values for A, B, and their combined value when playing (stationary) Nash-VI, Correlated-VI, or COCO policies, in the games described in this paper.

Our experiments illustrate both the intelligent transfers made by COCO agents, and the interesting properties that arise from playing COCO strategies. In Prisoner and Turkey, we illustrate that COCO agents accrue symmetric values in symmetric games. In Friend or Foe, we illustrate that, unlike COCO, Nash-VI and Correlated-VI are not guaranteed to converge. In Incredible, we illustrate that the COCO policy is not always individually rational, in the sense that an agent can achieve a higher value by not abiding by it.

5. Conclusions and Future Work

We introduced a new algorithm, Coco-Q, that is convergent and produces interesting solutions to challenging stochastic games when utility is transferable and binding agreements are possible.

Coco-Q, like COCO values in normal-form games, is not defined for games with three or more players. It is an open problem to generalize to the ideas discussed herein to a wider class of games.

It remains to be seen whether it is reasonable to expect agents (including people) to be rational about side payments in stochastic settings, but the strategies that Coco-Q exhibits appear sound.

References

Bowling, Michael. Convergence problems of general-sum multiagent reinforcement learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pp. 89–94, 2000.

Chen, Xi and Deng, Xiaotie. Settling the complexity of two-player Nash equilibrium. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 261–272, 2006.

Gilboa, I. and Zemel, E. Nash and correlated equilibria: Some complexity considerations. *Games and Economic Behavior*, 1:80–93, 1989.

Gomes, Eduardo Rodrigues and Kowalczyk, Ryszard. Dynamic analysis of multiagent Q-learning with e-greedy exploration. In *Proceedings of the 2009 International Conference on Machine Learning*, 2009.

Greenwald, Amy and Hall, Keith. Correlated-Q learning. In *Proceedings of the Twentieth International Conference on Machine Learning*, pp. 242–249, 2003.

Hu, Junling and Wellman, Michael P. Nash q-learning for general-sum stochastic games. *The Journal of Machine Learning Research*, 4:1039–1069, 2003.

Kalai, Adam and Kalai, Ehud. Cooperation in strategic games revisited*. *The Quarterly Journal of Economics*, 2012.

Kalai, Adam Tauman and Kalai, Ehud. Cooperation and competition in strategic games with private information. In *Proceedings of the 11th ACM conference on Electronic commerce*, EC '10, pp. 345–346, New York, NY, USA, 2010. ACM.

Littman, Michael L. Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the Eleventh International Conference on Machine Learning*, pp. 157–163, 1994.

Littman, Michael L. Friend-or-foe Q-learning in general-sum games. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pp. 322–328. Morgan Kaufmann, 2001.

Littman, Michael L. and Szepesvári, Csaba. A generalized reinforcement-learning model: Convergence and applications. In Saitta, Lorenza (ed.), *Proceedings of the Thirteenth International Conference on Machine Learning*, pp. 310–318, 1996.

Munoz de Cote, Enrique and Littman, Michael L. A polynomial-time Nash equilibrium algorithm for repeated stochastic games. In *24th Conference on Uncertainty in Artificial Intelligence (UAI'08)*, 2008.

Puterman, Martin L. *Markov Decision Processes—Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., New York, NY, 1994.

Sandholm, Tuomas W. and Crites, Robert H. Multiagent reinforcement learning in the iterated prisoner's dilemma. *Biosystems*, 37:144–166, 1995.

Shapley, L.S. Stochastic games. *Proceedings of the National Academy of Sciences of the United States of America*, 39:1095–1100, 1953.

Sutton, Richard S. and Barto, Andrew G. *Reinforcement Learning: An Introduction*. The MIT Press, 1998.

Watkins, Christopher J. C. H. and Dayan, Peter. Q-learning. *Machine Learning*, 8(3):279–292, 1992.

Zinkevich, Martin, Greenwald, Amy R., and Littman, Michael L. Cyclic equilibria in Markov games. In *Advances in Neural Information Processing Systems 18*, 2005.