Combining Correlated-Q Learning and Coco Values

Adam Abeshouse, Elizabeth Hilliard, Eric Sodomka

1 Introduction

Reinforcement learning predominantly addresses single-agent learning. There have been, however, some efforts to apply the techniques to multiagent scenarios. When multiple players interact, game theory is commonly used to model how agents make decisions about how to optimize their actions. We combined an algorithm that applies Q-learning to multiagent settings, Coorelated-Q Learning [2](CE-Q), with a new game theoretic cooperative solution concept, coco values [6].

Cooperative and competitive, or coco, values [4, 5, 6] are a solution concept for cooperative games that are proven to produce equilibrium strategies with properties such as pareto optimality, payoff dominance and monotonicity in strategies when applied to traditional bimatrix games. We combined ideas from correlated-Q learning [2] and repeated stochastic games [1] but instead used coco values as a solution concept for multiple agents playing grid games.

2 Implementation

We had the implementation of the original CE-Q paper in matlab, but discovered that the grid game implementation was too rigid and that it was not easily extendible to the correct algorithm for VI. We therefore decided to implement a more general stochastic game solver in Java and a coco value specific simulator in Python.

Our solver performs value iteration on a general N-player stochastic game with (potentially) transferable utility. At each iteration, the solver loops over each state and creates a normal-form game whose payoffs are derived from Bellman's equation: that is, the payoffs for player i in state s when joint actions $\vec{a}$ are taken is i's immediate reward for the joint actions plus its discounted expected future reward from the value function. The normal form game is then itself solved with any normal form game solver, and the payoffs for each player are put into the new value function at state s. We currently have two solvers for normal-form games: Nash and coco values; other normal-form solvers such as correlated equilibria or friend/foe are also possible. The python solver follows the same algorithm but is specific to coco values.

We also implemented a more general framework for creating grid games. The CE-Q paper's implementation hardcoded where the barriers could be, did not allow for multiple goals for an agent and could not support arbitrarily shaped boards or an arbitrary number of agents. Our implementation allows for arbitrarily shaped boards, configurable placement of barriers and semi-walls and multiple goals for an agent. Both the Python and Java implementations use the same input file system.

3 Future Work

We foresee interesting results when these grid games are played with more than two agents who are allowed to coordinate their actions. Our game implementation allows for an arbitrary number of agents but solving large normal form games is computational expensive and Coco values for more than two players are currently undefined. Another interesting exploration would be to develop more and larger grid games.

We would be interested in seeing further investigation of the relationship between various parameters of the grid and the agents' policies. It would be interesting to see, given some fixed configuration, what are the "thresholds" for relative goal values which cause the optimal policy to switch from one which favors one agent to one which favors the other.

A different direction would be to conduct experiments on how humans play some of the grid games. It could be informative to examine how allowing, disallowing, encouraging or enforcing side payments would effect the strategies humans choose to adopt and compare the humans' learned behavior and final payoff to those of the coco agents.

4 Original Table

Figure 2. Convergence in the grid games: all algorithms are converging. The CE-Q algorithm shown is uCE-Q.

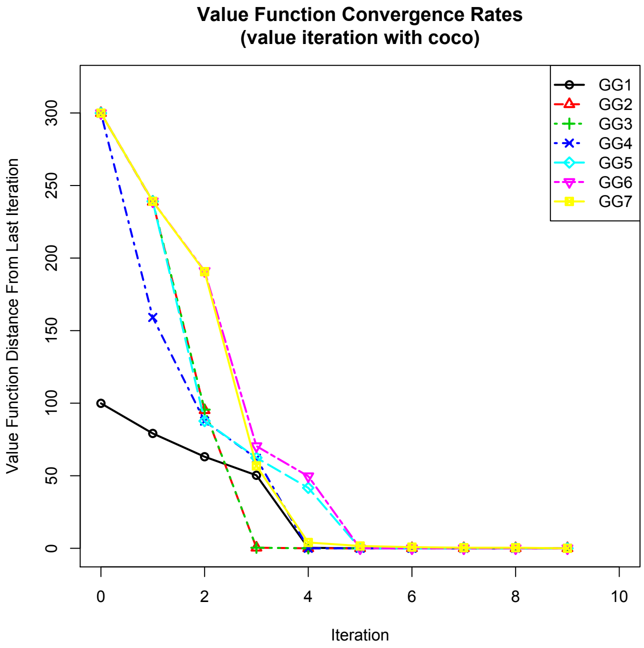
Grid Games	GG1		GG2		GG3	
Algorithm	Score	Games	Score	Games	Score	Games
Q	100,100	2500	49,100	3333	100,125	3333
Foe-Q	0,0	0	67,68	3003	120,120	3333
Friend-Q	$-10^4, -10^4$	0	$-10^4, -10^4$	0	$-10^4, -10^4$	0
uCE-Q	100,100	2500	50,100	3333	116,116	3333
eCE-Q	100,100	2500	51,100	3333	117,117	3333
rCE-Q	100,100	2500	100,49	3333	125,100	3333
lCE-Q	100,100	2500	100,51	3333	$-10^4, -10^4$	0

Table 2. Grid Games played repeatedly, allowing 10^4 moves. Average scores are shown. The number of games played varied with the agents' policies: some move directly to the goal, while others digress.

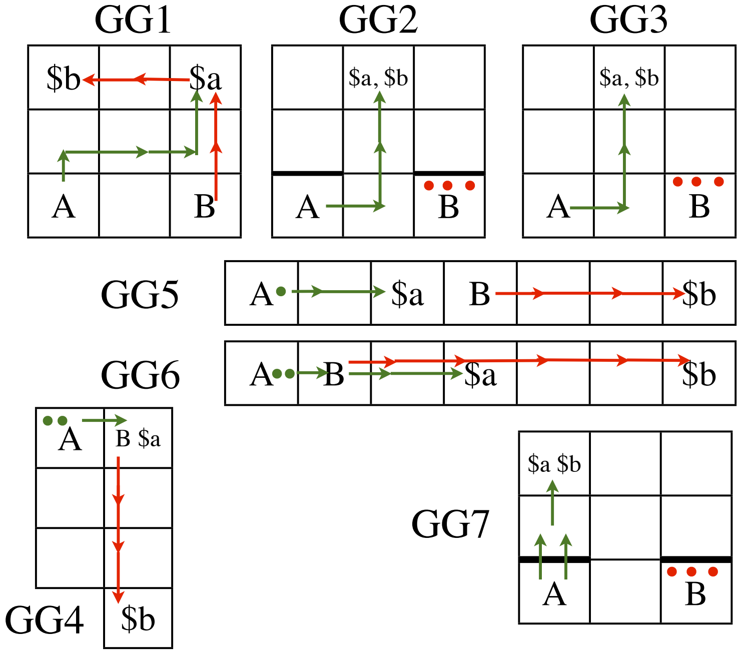
5 Data from Rerun

Grid Games	GG1		GG2		GG3	
Algorithm	Score	Games	Score	Games	Score	Games
Friend-Q	$-10^4, -10^4$	0	$-10^4, -10^4$	0	$-10^4, -10^4$	0
uCE-Q	100,100	2500	50,100	3333	117,117	3333
eCE-Q	100,100	2500	100,50	3333	117,117	3333
rCE-Q	100,100	2500	49,100	3333	100,125	3333
lCE-Q	100,100	2500	52, 100	3333	$-10^4, -10^4$	0

6 Convergence



7 Coco Agent Policies



References

[1] Enrique Munoz de Cote and Michael L. Littman. A polynomial-time nash equilibrium algorithm for repeated stochastic games. *CoRR*, abs/1206.3277, 2012.

[2] Amy Greenwald and Keith Hall. Correlated-q learning. In *In AAAI Spring Symposium*, pages 242–249. AAAI Press, 2003.

[3] Amy Greenwald, Keith Hall, and Martin Zinkevich. Correlated q-learning, 2005.

[4] Adam Kalai and Ehud Kalai. Cooperation in two person games, revisited. *SIGecom Exch.*, 10(1):13–16, March 2011.

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

[6] Adam Tauman Kalai and Ehud Kalai. A cooperative value for bayesian games, 2010.