Winning the Lemonade Stand Game: Cooperation or Defection?

Taha Rafiq
University of Waterloo
t2rafiq@uwaterloo.ca

Abstract

The Lemonade Stand Game (LSG) is a simple 12-action iterated constant sum game in which three agents compete to maximize their utility over a period of *n* rounds. The Lemonade Stand Game Tournament (LSGT) is an annual agent design competition based on the forementioned game. This paper examines various strategies employed by agents in the LSGT 2011 and proposes a new strategy that outperforms all the previous strategies by a significant margin (83% performance improvement over the top ranked agent from LSGT 2011). The fundamental questions addressed by this paper are whether cooperation is always the best strategy in LSG, and how can an agent maximize it's utility when other agents are not cooperative.

Introduction

Autonomous agent interaction is a growing area of research in the artificial intelligence community. In an autonomous environment, agents take actions that are a response to the environment and their actions themselves also form a part of the environment that other agents respond to. Because of the autonomous nature of the environment, agents may choose to take actions that are not entirely rational or based on game-theoretic principles. Herein lies an important challenge of multiagent systems research; there may be strategies that are provably superior considering a rational world, however agents in the real world rarely behave in this manner (Zinkevich 2011a).

How should an agent behave when this is the case? This paper looks to examine the question stated above using an empirical study of agent strategies in the Lemonade Stand Game. Previous competition results of LSGT demonstrate that simply using proven artificial intelligence methods such as approximating an equilibrium strategy or regret minimization does not result in successful agents. In fact, the best performing agents employ completely different strategies (Zinkevich 2011a). The reason behind this stems from the fact that the LSG isn't solvable, therefore traditional methods such as equilibrium computation and minimax do n't work. Furthermore, combining strategies from different equilibria can yield highly sub-optimal strategy profiles in the LSG (Zinkevich 2011a).

Overview of the Lemonade Stand Game

The Lemonade Stand Game is a 12-action constant sum game which is played between three agents. The LSG website (Zinkevich 2011b) describes the game as follows:

> It is summer on Lemonade Island, and you need to make some cash. You decide to set up a lemonade stand on the beach (which goes all around the island), as do two others. There are twelve places to set up around the island like the numbers on a clock. Your price is fixed, and all people go to the nearest lemonade stand.
>
> The game is repeated. Every night, everyone moves under cover of darkness (simultaneously) and in the morning, their locations are fixed. There is no cost to move. After 100 days of summer, the game is over.

Therefore each game consists of 100 rounds or iterations of the single-shot game. In the LSGT, all the participating agents take part in an equal number of games to make the tournament fair.

Utilities in the Lemonade Stand Game

In the LSG, there are 6, 12, or 18 customers at each of the 12 spots along the beach. The customers go to the nearest lemonade stand, with ties split randomly. An agent gets a dollar for each person that comes to its lemonade stand. The utility of the repeated game is the sum of the utilities of single-shot games.

Example Configuration

In order to aid understanding of the LSG, we present an example configurations of a single round of the LSG in Figure 1. The three different shapes represent three competing agents. The numbers outside the circle represent the number of customers at each spot on the island. The numbers in the center of the circle represent the utilities of the agents at the end of this round.

Related Work

The inspiration for the Lemonade Stand Game and the consequent tournament was taken from Alexrod's iterated prisoner's dilemma (Axelrod 1987).

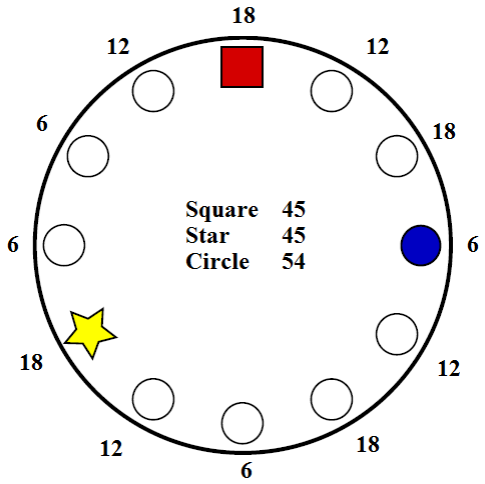

Figure 1: An example configuration of a single-shot round of the LSG.

Although there isn't a lot of published research on the Lemonade Stand Game, the game has received some attention in the last few years. (Zinkevich, Bowling, and Wunder 2011) outlines the objectives of the LSG and summarizes the results from the first two years of the competition. In (Sykulski et al. 2010) the strategy of the winning agent of the 2009 version of the competition is presented in detail. Furthermore, (Reitter et al. 2010) presents a study of the performance of a meta-cognitive strategy against other trivial and non-trivial hypothetical agents in the LSG.

Overview of LSGT 2011 Agents

In this section, we present an overview of the agents that competed in the Lemonade Stand Game Tournament in 2011. Then we present a categorization of the agents based on their design philosophy. Finally, we present the results of an experiment consisting of the 2011 agents to see their baseline performance. We present this information in order to develop a better understanding of the results from our experiments in subsequent sections, where we present our proposed agent and compare it's performance with the 2011 agents.

The 2011 agents are presented in the following subsections. Their descriptions are adapted from the original descriptions provided on the LSG website (Zinkevich 2011b).

1. BMJoe (BMJ)

This agent uses a simple strategy in which it only takes into account the last round. If the agent got its share of the customers (at least one-third of the total number of customers on the island), it will stay at its spot. Otherwise, it will move to a spot that will give him third of the customers (assuming the other agents won't move).

If the agent cannot find a place that will get it a fair share, it will try to punish one of the other agents by putting his stand next to, or at the same spot as that agent until a spot that will give him his fair share becomes available.

2. BowlingStrategy (BOW)

The strategy of this agent is to identify collusion-pairs, i.e., two spots which together would maximize the pairs' combined payoff. Pairs are then selected based on their collusion success (the minimum combined utility the agents can achieve) and their fairness (how different their utilities would be without the third agent). Pairs with the highest collusion success are chosen, and among those, pairs with the fairest allocation are preferred. Of those still tied, a random pair is chosen.

Given a chosen pair at the start of the game, the agent tracks the accumulated score of the two positions in the pair. The position with the highest cumulative score is chosen. An initial bonus is given to the position in the pair, which got the smaller (less fair) allocation. The agent takes the smaller allocation in the hope that this would induce a fight between the other agents for the larger allocation, whose payoff would be lower when divided among the two agents.

3. BrownBot (BWN)

The strategy employed by this agent consists of a concept of 'leaders' and 'followers'. Leaders are agents who seek to position themselves on the best spot on the island and hope that the other agents would cooperate by looking for other reasonable spots. Followers are agents that look to cooperate with the leader by selecting a spot that is mutually beneficial for the two.

If there are two leaders and one follower in the game, the follower will benefit as the leaders will fight over the best spot leaving the other spots to the follower. The opposite also holds true in the case of two followers and one leader. Taking this observation into account, this agent adopts a leader strategy initially, which is adapted to a follower strategy if the agent finds itself in a persistent conflict with another agent for several turns.

4. NewAcrossS (NAS)

The strategy employed by this agent is similar to the Bowling agent. It attempts to find spots where two agents can cooperate to achieve a higher payoff than the third agent. If this strategy doesn't work and the agent finds itself in the last place, it attempts to 'sandwich' an agent (to lower it's utility) in order to force it to move to another spot. If this doesn't work, it will simply play the best response to the last round's moves.

5. GTTA (GTA)

This agent uses a refinement to the set of Nash Equilibria to construct a set of strategies which correspond to a set of types that opponents can take. The types considered by this agent are: generous leader, fair leader, greedy leader, me follower, other follower and unknown. Based on an assessment of the other agents' types, the agent forms its strategy accordingly.

6. JerseyBeachcomber (JBC)

The strategy of this agent is based on the reasoning that it is likely that one or both of the opposing agents will choose

a spot close to the best one. It is also possible that the opposing agents are on the two best spots. If this is the case, the third agent would prefer picking a spot near the best, leaving the next-best agent with a net higher score as the two others fight over the best spot. The decision of when to move from this initial spot (or any current location) is calculated by keeping a tally of how many points the winning agent is receiving, and moving once the difference between the utilities gained by this agent and the winning one reaches a threshold.

7. MatchMateAdv (MMA)

This agent tries to find a pure strategy solution by looking for two spots that give the two agents the highest combined reward, and among those the agent chooses the pair of spots that is most fair for both agents. The agent then tries to find a partner to play this pair by occupying one of the spots and moving to the other spot if another agent is located at the same spot as it.

8. Pujara (PUJ)

This agent initially picks a spot that is above some measure of density (number of customers around that spot). The agent keeps a track of it's performance as the game progresses. The agent switches to another spot (chosen randomly) if it consistently performs badly (i.e. the utility is below some threshold value for a specific number of rounds).

Categorization of Agents

In this subsection, we present a categorization of agents on the basis of whether they are *cooperative* or *independent*. A cooperative agent seeks to find a colluding partner and cooperate with it to maximize the combined utility, while an independent agent simply seeks to maximize its own utility and does not adopt any mechanism which takes other agents into account.

Figure 2 presents the categorization. This categorization will be helpful when we present our proposed agent because it incorporates both categories in order to successfully compete against all of the LSGT 2011 agents.

Agent	Category
BMJ	Independent
BOW	Cooperative
BWN	Cooperative
NAS	Cooperative
GTA	Cooperative
JBC	Cooperative
MMA	Cooperative
PUJ	Independent

Figure 2: Categorization of the agents from LSGT 2011 into cooperative or independent agents.

Baseline Performance

We conducted an experiment to evaluate the baseline performance of the agents. The agents competed against each other in a tournament of 10,000 games. The games were selected in a manner that each agent participated in roughly the same number of games in order to make the tournament fair. The experiment was repeated 100 times to minimize the effect of variance. The performance of the agents is given in Figure 3.

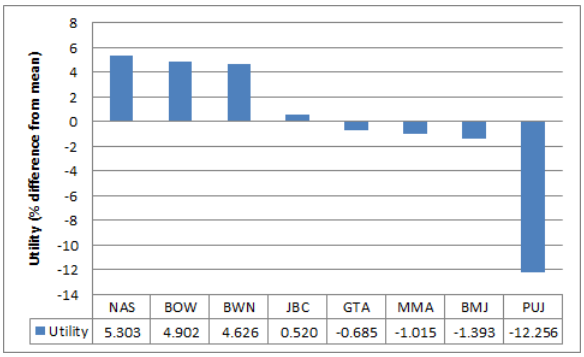

Figure 3: Performance comparison of agents from LSGT 2011. Confidence intervals have been omitted as they were too small to be visible.

Proposed Agent's Strategy

In this section we provide details of the proposed agent's strategy.

The strategy of our agent is based on the following observations from the previous section's experiment:

- Cooperative agents tend to perform better than independent ones.
- Most of the games are 'decided' long before all the rounds of a game are played; After the initial few rounds even the best agents become static in the action they played (regardless of their performance in the game). In a sense, agents *'run out of ideas'*.
- Some agents aren't designed to cooperate (or have different cooperation approaches) and hence attempts at cooperation with them fail, resulting in lower utility of the agent.

Taking these observations into account, we designed a cooperative agent which switches to an aggressive punishment strategy if attempts at cooperation fail. At any time during a game, our agent is in one of six following *states*:

1. **COLLUDE:** This is the starting state of the agent. In this state, the agent attempts to find a cooperating partner by identifying pairs of spots which maximize the combined utility of agents playing in those spots, and have a 'fair' division of utility among the agents. This state can be considered a slightly modified version of BrownStrategy.
2. **BEST_SPOT:** If attempts at collusion fail, the agent switches to this state. In this state, the agent simply finds

the most 'dense' contiguous region on the board, and positions itself at the spot in the middle of the dense region. The rationale behind this decision is that if the other agents aren't cooperative, perhaps they can be intimidated to move from the best spot by reducing it's value. The agent continues to play in this state if its utility from the previous round is at least better than one of the other agents.

3. **PUNISH:** If the agent (in state BEST_SPOT or CONSTANT) notices that it's utility from the previous round is lower than both the other agents' utilities, it switches to this state. In this state, the agent attempts to 'scare away' the nearest agent by playing at a spot that minimizes the utility of that agent (usually by placing its stall at the same spot as that agent). If the agent notices that the agent it is punishing isn't forced to move after a set number of rounds, it attempts to de-settle the other agent.
4. **CONSTANT:** While playing in the PUNISH state, the agent monitors the utility it would get if it played at another spot. If the expected utility is greater than the average utility for the game, the agent moves to that spot and switches to this state. In this state, the agent does nothing; it simply keeps playing in the same spot.
5. **MAXIMIZE:** If the agent finds that it can't find a colluding partner and neither of the other agents can be forced to move to another spot, it goes into this state. In this state, the agent simply seeks to maximize its utility given that the other agents' spots.

The actions of the agent can be configured by various parameters which affect the performance of the agent. The parameters along with their assigned values are given below.

1. **COLLUDE_THRESH:** This parameter defines a threshold for the max number of rounds the agent waits to find a colluding partner. We set the default value of this parameter to 20.
2. **DENSITY_CONST:** This parameter defines the number of spots considered when determining the density of a region (used for selection of the best spot). We set the default value of this parameter to 4.
3. **PUNISH_THRESH:** This parameter defines a threshold for the max number of rounds the agent attempts to force another agent away from its spot. We set the default value of this parameter to 8.

Performance Evaluation

In this section we provide details of experiments performed to evaluate the performance of our proposed agent in comparison with agents from LSGT 2011.

Experimental Setup

The experiments were performed on a Lenovo ThinkPad X220 laptop computer with Ubuntu 11.10 (Oneric). The system has 8 GB of RAM and features a Intel Core i5 processor. The code for running the LSG was downloaded from the tournament website (Zinkevich 2011b). Version 4.0.3 of the code was used.

Experiments

For our experiments, we considered three variations of our agent in order to determine which strategy performs the best: always cooperate, always rebel, or a mix of both cooperation and rebellion depending on the situation of the game. The details of the modifications for each variation are given below.

1. **CooperatorAgent (COP):** For this agent, we set the value of COLLUDE_THRESH to 100. This means that the agent tries to collude with another agent for the entire duration of the game.
2. **RebellerAgent (REB):** For this agent, we set the initial state to BEST_SPOT. This means that this agent never goes into the COLLUDE state. It always occupies the best spot and attempts to punish other agents attempting to occupy the best spot.
3. **CombinedAgent (COM):** This default values and states are used for this agent, as described in the preceding section. Hence, the agent attempts to cooperate with another agent for the first 20 rounds, after which it changes to an aggressive punishment strategy if cooperation isn't working.

We looked at these variations because we believe these relate strongly to the categorization of agents that we provided earlier. The CooperatorAgent is a cooperative agent, the RebellerAgent agent is an independent one, while the CombinedAgent can be considered a mix of both categories of agents.

We tested each of the three variations of our agent against the agents from LSGT 2011 in three separate experiments (i.e. the three variations of our agent weren't evaluated against each other). In each experiment, one of our variations competed with all the agents from LSGT 2011 and in a tournament of 10,000 games (each agent played in approximately 3333 games). Each experiment was repeated 100 times to minimize the effect of variance.

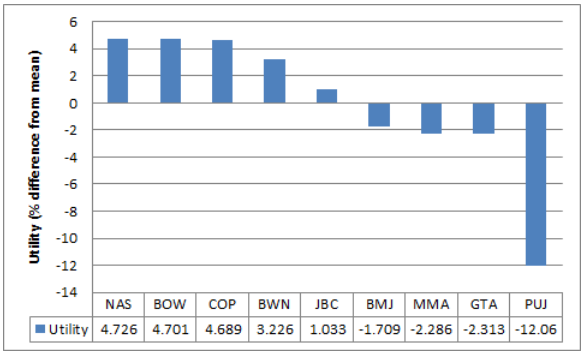

Figure 4: Performance of CooperatorAgent against agents from LSGT 2011. Confidence intervals have been omitted as they were too small to be visible.

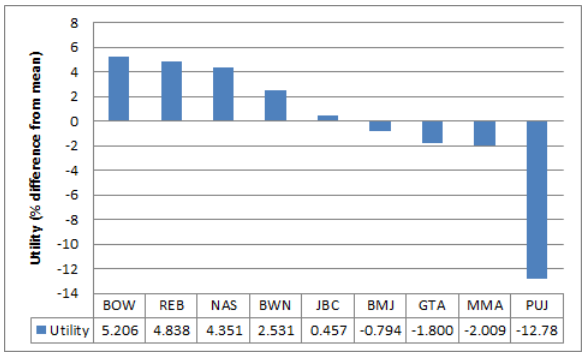

Figure 5: Performance of RebellerAgent against agents from LSGT 2011. Confidence intervals have been omitted as they were too small to be visible.

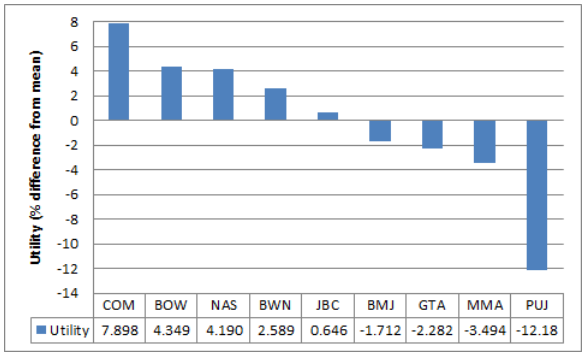

Figure 6: Performance of CombinedAgent against agents from LSGT 2011. Confidence intervals have been omitted as they were too small to be visible.

Results

We present the results of our experiments in this subsection. Figure 4, 5 and 6 present the performance of CooperatorAgent, RebellerAgent and CombinedAgent respectively against the LSGT 2011 agents. CooperatorAgent and RebellerAgent perform almost as good as the best performing agent from LSGT 2011. However, as Figure 6 indicates, CombinedAgent outperforms all the agents from LSGT 2011 by a large margin. CombinedAgent attains 83% more utility than the second placed agent, BowlingStrategy.

CombinedAgent outperforms all the other agents because of its ability to cooperate when other agents are willing to cooperate and defect when cooperation isn't an effective strategy. This makes it a very difficult adversary to evaluate and adapt to, and hence it performs better than the other two variations.

In Figure 7, we present the performance of the three variations of our agent against each of the other LSGT 2011 agents individually (taking only those games into account in which the compared agent was participating). CooperatorAgent and RebellerAgent perform better than average against all the agents, but their performance against some of the agents is just slightly above average. However, CombinedAgent consistently performs well above the average against all the agents.

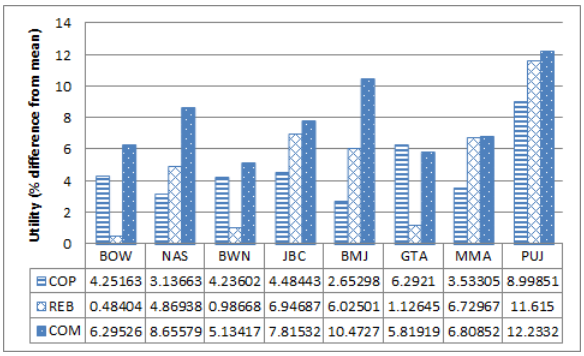

Figure 7: Performance of our agent variations against individual agents from LSGT 2011.

Conclusion

In this paper, we examined the strategies employed by LSGT 2011 agents, and proposed a new strategy which outperforms all the 2011 agents by a significant margin. Our agent's performance is 83% better than the best agent from 2011 according to our experiments.

Our approach when designing this strategy was based on a number of observations, such as the success of cooperative agents, and the static behavior of most agents after the first few rounds. Hence, we engineered our agent keeping these observations into account and were able to create an agent that was substantially better than the ones from the last years competition.

We believe our experiments provide further insight into the domain of autonomous multiagent interaction. Our work was inspired by the desire to learn more about how agents should react in an environment in which the seemingly 'best' action isn't always the right one. Our results show that an agent that incorporates a number of different strategies and can adapt to the changes in the environment successfully is likely to be better off than other, comparatively more static agents.

Acknowledgments

We would like to acknowledge the participants of LSGT 2011 who have provided the source code of their agents and descriptions of the employed strategies on the LSGT website (Zinkevich 2011b). We would not have been able to perform these experiments without them. We would further like to thank Martin Zinkevich of Yahoo! Research who runs the LSGT annually and maintains the code for the tournament.

References

Axelrod, R. 1987. The evolution of strategies in the iterated prisoners dilemma. *Genetic algorithms and simulated annealing* 32–41.

Reitter, D.; Juvina, I.; Stocco, A.; and Lebiere, C. 2010. Resistance is futile: Winning lemonade market share through

metacognitive reasoning in a three-agent cooperative game. In *Proceedings of the 19th Behavior Representation in Modeling & Simulation (BRIMS)*.

Sykulski, A. M.; Chapman, A.; Munoz de Cote, E.; and Jennings, N. R. 2010. ea^2: The winning strategy for the inaugural lemonade stand game tournament. In *Proceedings of the 2010 European Conference on Artificial Intelligence*.

Zinkevich, M.; Bowling, M. H.; and Wunder, M. 2011. The lemonade stand game competition: solving unsolvable games. *SIGecom Exchanges* 10(1):35–38.

Zinkevich, M. 2011a. The future of multiagent learning. Website: http://martin.zinkevich.org/lemonade/future.php.

Zinkevich, M. 2011b. The lemonade stand game tournament website. Website: http://martin.zinkevich.org/lemonade/.