Developing a Repeated Multi-Agent Constant-Sum Game Algorithm Using Human Computation

Christopher G. Harris
Informatics Program
The University of Iowa
Iowa City, Iowa USA
christopher-harris@uiowa.edu

***Abstract*— In repeated multi-agent constant-sum games, each player's objective is to maximize control over a finite set of resources. We introduce *Tenspotter*, an easy-to-use publicly-available game designed to allow human players to compete as agents against a machine algorithm. The algorithm learns play strategies from humans, reduces them to nine basic strategies, and uses this knowledge to build and adapt its collusion strategy. We use a tournament format to test our algorithm against human players as well as against other established multi-agent algorithms taken from the literature. Through these tournament experiments, we demonstrate how learning techniques adapted using human computation – information obtained from both human and machine inputs – can contribute to the development of an algorithm able to defeat two well-established multi-agent machine algorithms in tournament play.**

Keywords- intelligent agents; multi-agent games; constant-sum games; Tenspotter; human computation; Android-based games

I. INTRODUCTION

The objective of a multi-agent, constant-sum game is for players to coordinate a series of moves in order to control a finite set of resources. Take an example of two competing retailers located along a busy shopping street, each selling an identical product at a fixed price. Each retailer wants to maximize market share by drawing the largest number of customers. Assuming a uniform distribution of customers along the street and assuming that each customer will always choose the nearest shop, Hotelling's law [5] predicts that a street with two shops will optimally find both shops right next to each other at the same halfway point, each capturing the customers on their respective side of the halfway point. Thus, each retailer will serve half the market. With three shops, the optimal (stable) situation is to find each store at locations one-sixth, half-way, and five-sixth along the length of the same street. Now consider how this balance might change if the street was circular in shape or if store locations could change frequently without the advanced knowledge of the other retailers?

The Lemonade Game (LG) is a tournament introduced by Zinkevich *et. al.* [15] in 2009 based on the game of the same name. The game scenario is set up as follows. It is summer on a circular island and vendors decide to set up a lemonade stand on the beach, which extends around the island's perimeter. You compete with two other vendors. In LG there are twelve possible lemonade stand locations around the island, arranged like a clock dial. The price of lemonade is fixed for all vendors, and customers always visit the lemonade stand closest to them. Each night, all vendors move simultaneously under the cover of darkness to a new location. No costs are associated with moving to a new spot. After 100 days of summer (each day is considered a round), the game is over. The total utility for each vendor of this repeated game is the cumulative sum of the utilities of each round. The vendor with the largest total utility is considered the game's winner.

Many three-agent negotiation algorithms, such as those used in LG, work by coordinating a series of moves with one of the two opponents in an effort to exploit the third opponent. These algorithms often rely on identifying and predicting each opponent's strategy, determining the ideal opponent for collusion, and exploiting the identified opportunity. However, there are numerous potential strategies involved with collusion attempts – some are mutually compatible while some are not. We discuss two well-known multi-agent algorithms later in this paper.

To illustrate and test these collusion attempts, we introduce *Tenspotter*, a multi-agent game designed to examine how both human players and algorithms recognize collusive behavior and adapt. A screenshot is shown in Fig. 1. Although easy to learn and play, *Tenspotter* is a challenge to master: state-of-the art algorithms fail when matched against even a simple strategy. Probability theory and Nash Equilibrium theory do not provide a single solution to this game [9].

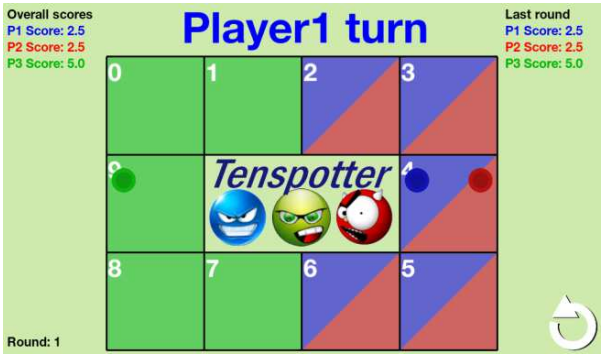

Figure 1. The *Tenspotter* game interface

Our contributions are as follows. First we introduce and describe *Tenspotter*, a three–player repeated constant–sum game available online[1]. Second, we describe the genesis of our algorithm from nine basic strategies, which represent those discussed in the literature and through our own empirical evaluation of human play. Third, we test our algorithm in two different experiments, each matched against opponents with different playing styles and characteristics: first, against human opponents and second, against leading repeated multi-agent constant-sum algorithms. We evaluate our performance against each.

[1] Available for download at http://www.irgames.org/tenspotter/

II. Related Work

Using repeated games to examine normal-form (also called constant-sum) decision models has drawn considerable attention recently in fields such as computer science, economics and engineering, particularly because they provide insight into collusion between agents - each employing different strategies - in an attempt to maximize utility. However, relatively little research has been done on the intersection of game-based normal-form decision–making models and autonomous agents, except for repeated games.

Although first described by Axelrod in [2], studies involving the iterated prisoner's dilemma have been described extensively in the literature, (e.g., [1, 3]). The prisoner's dilemma competition differs from the autonomous agent-based games, such as *Tenspotter*, in a few important ways. In a prisoner's dilemma competition, the act of cooperation between agents is clearly identified. Moreover, since most repeated prisoner's dilemma games typically only involve two agents, the target of an agent's cooperation is obvious.

There have been numerous studies that examine repeated ultimatum games, (e.g., [7, 12]) and repeated negotiation games, (e.g., [6, 8]). Each of these studies examines the underlying importance of two issues: detecting each opponent's strategy and adapting one's own strategy based on this detection.

Perhaps the best-known repeated game involving autonomous agents is the Lemonade Game (LG), offered as annual tournament since 2009 [15]. Several participants have described the algorithms used in their approaches. In [13], Sykulski *et. al.* describe an algorithm they used to defeat other algorithms in the initial LG challenge. They detect the best opponent to collude with based on two conditions – "sticking" or "shadowing" the chosen opponent. To detect the best partner for collusion, their algorithm compares the strategy of each opponent with that of an "optimal" opponent. Wunder *et. al.* [14] groups the LG strategies into levels of complexity, and applies a cognitive hierarchy approach to examine how these levels might be exploited. Likewise, Reitter *et. al.*[10] investigated three techniques: "stick", "random" and "roll", using a metacognitive approach. In each of these three approaches, the authors illustrate portions of their algorithms, which we recreate later in this paper.

III. Tenspotter

Tenspotter is a repeated-sum game offered as an Android application, designed to allow human players to play as an agent against algorithms built on these strategies. As the name *Tenspotter* implies, there are ten spots that can occupied by three participating agents. Agents independently select one of the ten spots in each round. Total utility for each round is 10 points and can be divided into fractional units, (shown in Fig. 2 and 3 as partially-shaded spots). Scoring in *Tenspotter* is done in the following way. Spots occupied by one or more players are split equally between each occupant. For the remaining unoccupied spots, we calculate the Euclidean distance to each of the three player-occupied spots and assign one point to the nearest player (in the case where the minimum distance is shared by more than one player, we divide the points equally between the players). This is referred to as a "sandwich" since Player 1 is sandwiched between the other two players and thus receives the minimum possible score of 1. The other two players earn a score of 4.5 points each. In the screenshots shown in Fig. 2 and 3, we use color shading for each spot. On the right of Fig. 2, all three players occupy the same spot (spot 2). In this case, all receive an equal payout of 3.33 points. On the left of Fig. 3, Players 1 (blue) and 2 (red) occupy spot 7 and Player 3 (green) occupies spot 1. Utility in spots 4 and 9 are split 0.33/0.33/0.33 each, since the two spots are equidistant from all three players. In LG, the utility is split differently: 0.25/0.25/0.50. On the right of Fig. 3, Players 1 and 2 occupy spot 4 and Player 3 occupies spot 9 – directly across the board from spot 4. In this case, utility is assigned in a 0.25/0.25/0.50 ratio, similar to LG. Utility for each player in each round is in the range (1, 5). Thus, a player's maximum utility is observed when we have a single player occupying a spot directly opposite the other two players, the first player receiving a utility of 5 while the two opponents each receive a utility of 2.5. The worst case is to be sandwiched between two players (see Fig. 2, left), resulting in a utility of 1 while the two opponents receive a utility of 4.5 each. Stable competition occurs when all players achieve a minimum utility ≥ 2.5.

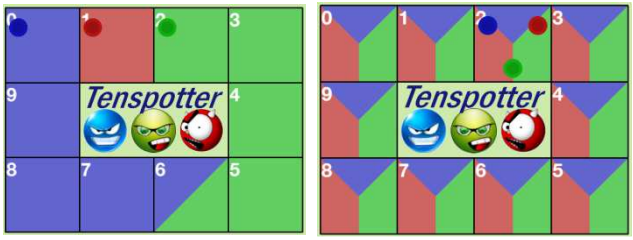

Figure 1. Tenspotter screenshots showing a "sandwich" move on the red player (left); three players occupying the same spot and therefore equally dividing utility (right).

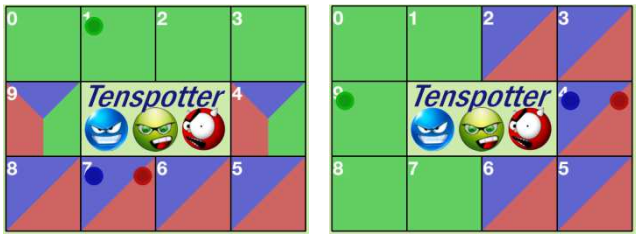

Figure 2. Tenspotter screenshots showing two players (red and blue) occupying the same spot, but not directly opposite the green player (left); directly opposite the green player (right).

A. Basic Strategies

The nine basic strategies considered here are either based on commonly-used strategies found in previously-described LG algorithms from the literature or are strategies that demonstrated a compelling value through early empirical examination. The nine strategies are as follows.

Random – generates a random number in the range (0,9) and occupies the corresponding spot until the next round.

Stick – uses Random for the first round, then "sticks", or retains that same spot for the duration of the game.

Stick10 –similar to Stick, this strategy obtains a randomly-assigned spot number in the initial round and keeps the same spot for the next nine rounds. Every ten

rounds thereafter, it randomly determines a new spot and sticks for the subsequent nine rounds

Shadow – uses Random for the first round, Then, in round *n* ($n > 1$), it "shadows", or moves to the same spot number of the cumulative point leader determined in the *n-1*th round. It repeats this action in every subsequent *round*. This is the same as the successful TIT-FOR-TAT described by Axelrod in [2], but here the opponent shadowed changes as the cumulative point leader changes.

Shadow+5 – similar to the Shadow strategy, but instead it moves directly across the board (i.e., 5 spots away) from the cumulative point leader.

Minimum – uses Random for the initial spot. In subsequent rounds, it moves to the spot of the opponent that earned the *lowest* number of points in the *n-1*th round. If >1 player received the lowest score, it randomly determines which of the spots to occupy (the strategy also considers itself in the lowest-score evaluation). Note that unlike the shadow, which relies on the *cumulative* point leader, this strategy examines the score each player received in the *immediately-preceding* (*n-1*th) round only.

Maximum – takes the same approach as the Minimum strategy, but instead uses the player with the maximum point score from the previous round.

Average – uses Random for the initial spot. In subsequent rounds, it determines the average of the two opponents' spot numbers from the immediately-preceding (*n-1*th) round and moves to that position, rounded up to the nearest integer.

Average+5 – similar to the Average strategy, but instead moves to the spot directly across the board (i.e., five spots away) from the average of the two opponents' locations as determined in the immediately preceding (*n-1*th) round.

TABLE I. WIN PERCENTAGE FOR EACH BASIC STRATEGY PLAYED, ALONG WITH BEST AND WORST STRATEGIES PLAYED BY AN OPPONENT.

P1 strategy played	P1 strategy win rate	Best strategy for P2 (P2 win rate vs. P1 strategy)	Worst strategy for P2 (P2 win rate vs. P1 strategy)
Random	0.505	Shadow+5 (0.476)	Stick10 (0.173)
Stick	0.259	Average+5 (0.763)	Minimum (0.073)
Stick10	0.328	Average+5 (0.714)	Minimum (0.049)
Shadow	0.266	Random (0.735)	Maximum (0.067)
Shadow+5	0.466	Shadow (0.512)	Random (0.166)
Minimum	0.131	Average+5 (0.852)	Average (0.146)
Maximum	0.272	Average+5 (0.754)	Shadow (0.097)
Average	0.175	Average+5 (0.883)	Minimum (0.049)
Average+5	0.593	Maximum (0.349)	Shadow+5 (0.103)

Table I shows the probability of winning, given player P1 and player P2 each choose one of the nine basic strategies. This is based on simulation of 24,300 games of 100 rounds each (2.43 million rounds). Table I also shows the best and worst strategy for player P2 to play against player P1. From this we can make some key observations. First, the Average+5 strategy is the most robust against the other eight strategies with an overall long-term win percentage of 0.593. However, should another player, aware that player P1 is playing the Average+5 strategy, decide to play the Maximum strategy, that second player has a 0.349 probability of winning – only slightly better than random (0.333). This winning percentage rate is also affected by the strategy chosen by the third player, but here we assume a long-run evaluation over a large number of rounds and that our third player, P3, randomly chooses to play one of the nine basic strategies in each round.

We also notice some natural collaborations occur, such as between Average and Average+5, and between Shadow and Shadow+5. There is no single-best strategy – even the robust Average+5 strategy is defeated using a Maximum strategy. Fortunately, many of the successful LG algorithms, two of which we will consider later in this paper, focus on only three of these nine strategies – Random, Stick and Shadow.

B. Advanced Strategies

Using information gained from 2.43 million simulations and another 163,400 human participant interactions with our nine basic strategies, we create an algorithm to address the ordering of each strategy. Our algorithm is then refined through play against human and other collusion-seeking algorithms, using the probability for a move in round *n* based on the opponent locations in round *n-1*. To build this algorithm, we determine whether our agent should lead or lag a targeted opponent. *Leading* an opponent means a player's chosen move in round *n* will affect the targeted opponent's location in round *n+1*. *Lagging* an opponent means that in round *n* we predict a targeted opponent's move for round *n+1*, and we move to an optimal position in round *n+1* to exploit that opponent.

When we are *leading* an opponent, as with Minimum, Maximum, Average, Average+5 (and with Stick or Stick10 if we are the cumulative point leader), we use the information from the immediately-preceding round to influence our move in the current round. With the Shadow and Shadow+5 strategies, if we are the current cumulative point leader in the game, we can perform a *lag* approach and can therefore anticipate the opponents move regardless of our current move. We can therefore move to a spot that maximizes the points received for that round.

Few other algorithms stick to a single basic strategy throughout a game, but instead change frequently to adapt to the moves of its opponents. However, all algorithms play *a strategy* in every round – as we have discovered, this is typically a Stick, Shadow or a Random strategy.

IV. EXPERIMENT ON BASIC STRATEGIES

Agent algorithms often form complex models using simple inputs as components. To address this, we perform a second online simulation to determine which of the nine basic strategies work best with leading, lagging, or switching to another opponent. In this second simulation, we conducted 24,300 games of 100 rounds each, evaluating our algorithm's approach against every combination of basic strategies. We evaluated all strategy combinations to determine a probabilistic value for each of our approaches, which are summarized in Table II.

TABLE II. THE WIN PERCENTAGE FOR OUR ALGORITHM AGAINST EACH OF THE NINE BASIC STRATEGIES.

Opponent strategy played	Our win rate vs. opponent strategy played	Opponent strategy played	Our win rate vs. opponent strategy played
Average	0.856	**Stick10**	0.702
Minimum	0.841	**Shadow+5**	0.566
Stick	0.763	**Random**	0.541
Shadow	0.745	**Average+5**	0.441
Maximum	0.724		

From Table II, we see that our algorithm is the winner nearly 70 percent of the time, on average, against the basic strategies. The lowest average win percentage for our algorithm is against the Average+5 opponent; however, it shows a win rate of 0.441, an improvement over the best of the nine basic strategies. This illustrates the merits of our algorithm over using the nine basic strategies alone. When compared to the average win percentage (column 2 of Table 1), we find that the average win percentage for our algorithm is a significant improvement [paired two-tailed t-test, t(8) = 3.5791, p = 0.0072].

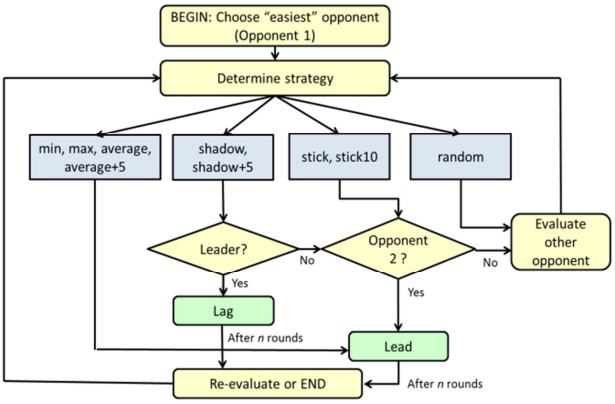

Figure 4. A simplified flow diagram of our model's algorithm

A flow diagram of our model's algorithm is provided in Fig. 4. We begin by choosing the "easiest" opponent (call it Opponent 1) and determine which of the nine basic strategies they are using, while we play an Average+5 strategy. The detection of the opponent's strategy typically takes 3 to 5 rounds. We have three options: we can choose to lead, to lag, or to switch to the other opponent (Opponent 2) to evaluate. Once a strategy is chosen, it is periodically re- evaluated every *n* rounds. The value of *n* may be known (e.g., 10 for Stick10) or may vary depending on if our model remains the cumulative point leader.

In these simulations, our model is the only one to use an algorithm to make strategy decisions, and is therefore not a true tournament. The Random strategy was determined to be the most difficult opponent for our algorithm to collude with, while the Stick strategy was the easiest. We observe a weak positive correlation between strategy collusion suitability with our model and overall win percentage, [$r = 0.38$, $p = \leq 0.001$, $R^2 = 0.148$].

V. Experiments on Advanced Strategies

We conducted two sets of experiments to test our algorithm. In the first, we examine its abilities against human players; in the second, we test it against some of the leading algorithms used in repeated symmetric three–player constant–sum finite horizon games.

For the first set of experiments, we hired human agents to play *Tenspotter* and serve as the two opponents of our algorithm. We test our algorithm with human opponents to examine its ability to adjust to tactics that do not follow any of our nine prescribed basic strategies. A total of 140 players were solicited using Amazon Mechanical Turk and paired at random to play a web-based version of the game. Players were told in advance which of the opponents was human and which player was represented by our model. We paid crowdworkers $0.02 to play the through a game of 100 rounds. To provide an additional incentive to the crowdworkers, we provided them an additional bonus of $0.05 if they won a game. An earlier empirical experiment using the same game showed that a small financial incentive improved player scores significantly [two-tailed t-test, t(138) = 22.16, p < 0.001]. We only offered compensation for the first game; however, nearly 31 percent of players played additional games with no expectation of compensation [2], demonstrating the enjoyment factor of the game. In 300 games, human players won 16 percent of the time, which was significantly different than random, [χ^2 (4, N = 300) = 60.06, p $\leq$ 0.001]. Thus, our algorithm demonstrated its ability to handle opponents that did not follow one of the nine basic strategies.

Examining data from games that our algorithm did not win provided some interesting insights. First, despite no ability to verbally communicate with each other, nearly all winning human players were able to successfully collude with the other human opponent and exploit our model. Our model was unable to intervene if that collusion occurred. In order to win, we observe that the collusion needed to be established early in the game – successful human players demonstrated this action within the first 15 rounds, with the highest game scorers establishing this action within 7 rounds. No human player was consistently able to collude against our model, although some human players were able to do so a majority of the time, despite being paired with different human opponents each time.

Finding the right opponent for collusion and sticking with that opponent paid off – humans who tried to collude with the other human opponent and then switched to collude with our algorithm (or vice versa) were rarely successful. Second, many players appeared to play with no preset strategy, mimicking the random approach – a difficult strategy for our algorithms to handle. Our algorithm has only a slightly better than even chance of success against random approaches.

The second experiment approximated two successful LG algorithms described in the literature. These algorithms are modified for use with *Tenspotter*. The first algorithm (EA^2) by Sykulsi *et. al*. [13], involved the winner of the initial LG tournament. The second algorithm, described by Wunder *et. al.* [14], uses a Parameterized Interactive Partially Observable Markov Decision Process (PI-POMDP) approach. We chose these algorithms because they have been successful against other opponents and are described in sufficient detail to apply to *Tenspotter*. We adjusted them slightly for the differences in the number of spots available for play in *Tenspotter* as well as for the difference in scoring methods. We conducted a tournament of 60 games using the three agents (EA^2, PI-POMDP, and our algorithm). These results, reported in Table III, are significantly different than random, [χ^2 (4, N = 60) = 36.8, p $\leq$ 0.001]. Examining Table III, we see that the mean number of points per game for each algorithm is significantly different at the p=0.05 level [F(2, 176) = 22.86, p$\leq$ 0.001].

[2] In the additional games played without compensation, each player was still eligible for the bonus if they won.

TABLE III. THE WIN PERCENTAGE FOR EACH OF THE THREE MOST COMMONLY PLAYED STRATEGIES AGAINST THE AVERAGE+5 STRATEGY.

Algorithm	Avg. utility earned per game	# of 1st place finishes	# of 2nd place finishes	# of 3rd place finishes
Our model	358.0	36	13	11
PI-POMDP	332.1	18	24	18
EA^2	309.9	6	23	31

Post-hoc analysis using a Sidak test [11] indicated that our algorithm significantly outperformed the other two algorithms. Overall, our algorithm is able to win 36 of the 60 games of the tournament. The EA^2 algorithm is trained to classify its opponents by their proximity to playing either a Stick or Follow strategy, determined on their previous actions, manifested in a weighting factor. Because our algorithm adapts based on previous actions, a strong pattern of Stick or Follow is rarely observed. The EA^2 algorithm is therefore rarely able to achieve meaningful weights for its parameters in response to our model. The EA^2 default for poor weighting is to initiate a stick approach; in this scenario, our algorithm will move to the other opponent; likewise, EA^2, our model chooses actions that demonstrate low stick and follow indices, and will be ignored by EA^2 as a collaborative partner.

The PI-POMDP algorithm [4] uses techniques similar to our own model; specifically, it takes a cognitive hierarchy approach, allowing for a distribution of agent types to represent each level of complexity. Thus, it can lead to multiple best responses, leaving a challenge for it to choose from among the most appropriate responses. One observed weakness in the PI-POMDP algorithm is that the set of included opponent policies is not clearly specified and therefore the solution breaks down when it encounters an unfamiliar policy or strategy. PI-POIMP mitigates this by working with a range of solutions. However, with our model, we can lead with one of nine strategies, six of which are not matched by PI-POIMP, exploiting this weakness.

In each of the six games our advanced algorithm did not win, opponents chose to collude with each other. This points out a weakness of our model – it does not try to "steal away" one of the two opponents when they are in the process of collaborating. We note that our model is built upon recognizing strategy changes in our opponents. Since opposing algorithms categorize each of their opponents in one of two basic categories (either 'collude' or 'exploit'), our algorithm can easily take advantage of this subtle flaw. However, as more sophisticated algorithms are introduced, our model will need to adapt to the additional sophistication they present.

VI. CONCLUSION

This paper has introduced a repeated multi-agent constant sum game called *Tenspotter*, and has described how this game varies from other established repeated games. We also described nine basic strategies, which were either previously used in other algorithms or empirically determined have value in repeated games. We used the crowd to tune these nine basic strategies to develop an algorithm specifically designed for collusion in repeated games. Our algorithm was tested against each of the nine basic strategies, and demonstrated it could be successful a large majority of the time. We then tested it against human players and against two algorithms that showed the ability to win in other repeated multi-agent game tournaments. Against human players, our model was successful 84% of the time; against the two algorithms, we were able win in 60% of the games. Therefore, we see considerable promise for our algorithm.

In future work, we anticipate using human computation methods once again to examine how to steal collusion partner in the process of forming with the other opponent. We will look at how to realign our collusion strategy to initiate simultaneous collaboration with both opponents. We also plan to categorize opponent moves into meta-strategies. We are aware that other multi-agent researchers are also improving their algorithms as well. This is an 'arms race' that we believe will lead to overall improvements in autonomous multi-agent efficiency.

REFERENCES

[1] Andreoni, J. and Miller, J. H. Rational cooperation in the finitely repeated prisoner's dilemma: Experimental evidence. *The economic journal*, vol 103 issue 418, 1993, pp. 570-585.

[2] Axelrod, R. and Hamilton, W. D. The evolution of cooperation. *Science*, vol 211, issue 4489, 1981, pp 1390-1404.

[3] Boyd, R. and Lorberbaum, J. P. No pure strategy is evolutionarily stable in the repeated Prisoner's Dilemma game, *Nature* **327**, pp. 58-59 (07 May 1987); doi:10.1038/327058a0.

[4] Gmytrasiewicz, P. and Doshi, P. A framework for sequential planning in multiagent settings. *Journal of Artificial Intelligence Research*, vol 24, issue 1, 2005, pp. 49-79

[5] Hotelling, H. Stability in competition. *The economic journal*, vol. 39, issue 153, 1929, pp. 41-57

[6] Jennings, N. R., Faratin, P., Lomuscio, A. R., Parsons, S., Wooldridge, M. J. and Sierra, C. Automated negotiation: prospects, methods and challenges. *Group Decision and Negotiation*, vol 10, issue 2, 2001, pp. 199-215.

[7] Lin, H. and Sunder, S. Using experimental data to model bargaining behavior in ultimatum games. *Experimental Business Research. Dordrecht: Kluwer*, 2002. Working paper, Available at: http://www.som.yale.edu/Faculty/sunder/research.html

[8] Littman, M. and Stone, P. Implicit negotiation in repeated games. *Intelligent Agents VIII*, 2002, pp. 393-404

[9] Nash, J. Non-cooperative games. *The Annals of Mathematics*, vol 54, issue 2, 1951, pp. 286-295.

[10] Reitter, D., Juvina, I., Stocco, A. and Lebiere, C. Resistance is futile: Winning lemonade market share through metacognitive reasoning in a three-agent cooperative game. In *Procedings of 19th Conference of Behavior Representation in Modeling and Simulation (BRIMS)*, Charleston, SC, 2010.

[11] Sidak, Z. Rectangular confidence regions for the means of multivariate normal distributions. *Journal of the American Statistical Association*, 1967, pp. 626-633

[12] Slembeck, T. Reputations and fairness in bargaining-experimental evidence from a repeated ultimatum game with fixed opponents. *Technical report, EconWPA (1999)* Available at: http://ideas.repec.org/p/wpa/wuwpex/9905002.html.

[13] Sykulski, A. M., Chapman, A., Munoz De Cote Flores Luna, J. E. and Jennings, N. EA^2: The Winning Strategy for the Inaugural Lemonade Game Tournament, 2010.

[14] Wunder, M., Kaisers, M., Littman, M. and Yaros, J. R. *A cognitive hierarchy model applied to the lemonade game*, In *AAAI Workshop on Interactive Decision Theory and Game Theory (IDTGT)*. 2010

[15] Zinkevich, M. A., Bowling, M. and Wunder, M. The Lemonade Game competition: solving unsolvable games. *SIGecom Exch.*, vol 10, issue 1, 2011, pp. 35-38.