Efficient Bidding Strategies for Simultaneous Cliff-Edge Environments

Ron Katz and Sarit Kraus
Department of Computer Science and The Gonda Brain Research Center
Bar-Ilan University, Ramat-Gan 52900, Israel, sarit@cs.biu.ac.il

Abstract

This paper proposes an efficient agent for competing in simultaneous substitutional Cliff-Edge (SCE) environments, which include simultaneous auctions and multi-player Ultimatum-Games. The agent competes in one-shot interactions repeatedly, each time against different human opponents, and its performance is evaluated based on all the interactions in which it participates. It learns the general pattern of the population's behavior and does not apply any examples of previous interactions in the environment, neither of other competitors nor of its own. Moreover, the agent rapidly adjusts to environments comprising a large number of optional decisions at each decision point. We propose a generic approach which competes in different substitutional environments under the same configuration, with no knowledge about the specific rules of each environment. The underlying mechanism of the proposed agent is the Simultaneous Deviated Virtual Reinforcement Learning (SDVRL) algorithm, which is an extension of an algorithm for non-simultaneous environments. In addition, we propose a heuristic for improving our agent's complexity. Experiments comparing the average payoff of the proposed algorithm with other possible algorithms reveal a significant superiority of the former. In addition, our agent performs better than human competitors executing the same tasks.

1. Introduction

In many auctions, substitutability exists between two or more items for sale [3, 8]. This situation is especially common in web-based auctions, where one can find many simultaneous auctions of similar popular items on different sites [11]. In such situations, a bidder bids on multiple items in order to obtain only one of them. Similarly, there are many economic and political situations in which one has to propose a resource allocation among multiple opponents, and to reach an agreement for the allocation from at least one of them [2, 7].

In this work we present an approach for repeatedly interacting in simultaneous substitutional first-price sealed-bid auctions and N-players Ultimatum-Game (UG). Both auctions and UG environments belong to the Cliff-Edge (CE) set of negotiation environments, which are characterized by an underlying conflict for the competitor between the desire to maximize profits and the risk of causing the entire deal to fall through [5]. Consider, for example, an agent that participates in substitutional auctions. On one hand, the agent strives to decrease its offers in order to pay less for the goods. On the other hand, if the offers are too low, the agent may lose the auctions. Moreover, the agent strives to win in no more than one auction, without ruining the probability of winning at least in one. In this paper, we focus on one-shot interactions, which are repeated with different opponents. Such situations occur in periodically repeated auctions with the same goods, which are very popular nowadays especially via the Internet [13, 3]. Similarly, sequential UG with different opponents, as well as N-players UG versions are under thorough investigation [1, 12, 2, 7].

The topic of simultaneous auctions for substitute goods has been drawing a lot of attention both in economical research [10] and in automated agents literature [9]. Economical works usually assume full-rationality of the players. However, when interacting with humans, the theoretical equilibrium strategy is not necessarily the optimal strategy since human subjects, inherently computationally restricted as well as rationally bounded, commonly do not behave according to perfect equilibrium. Thus, such studies (e.g. [10]) are not efficient for human-involved environments, as in our case. Some researchers have developed agents for competing in simultaneous continuous auctions, where other bids are visible [9]. Similarly, many agents have been developed for the well known Trading Agent Competition (TAC - http://tac.eecs.umich.edu) which includes substitutional auctions (e.g. [8]). In the TAC environment, however, the auctions are continuous and the bids of others are visible. Moreover, the auctions are second-price - and therefore there is no risk in overbidding, as in our environment. Furthermore, none of the above consider repeated interactions while changing human opponents.

Zhu and Wurman [13] proposed an agent for interacting with other agents or human bidders in simultaneous first-price sealed-bid auctions. Their agent, however, interacts repeatedly with the same opponent. In such problems, the general approach is to specifically model each of the opponents' behavior, and to adjust a strategy which optimally reacts to the opponent's predicted reaction. In our study, on the other hand, specific opponent modeling is not relevant, since the agent interacts with each opponent only once. Therefore, we propose using an agent which models the general pattern of the whole population of opponents. Jennings and his collogues, who presented several methods for bidding in multiple auctions of heterogeneous types (e.g. [11]) also did not consider the adaptation to the population

of bidders.

One possible approach for learning the behavior of the population of bidders in a certain environment is to observe previous samples of interactions within this environment [3]. This approach, however, requires a large number of historical examples of human behavior. Moreover, the option of using a historical database is not always possible. In a sealed-bid auction, for example, a bidder usually cannot obtain information about the bids which were offered in previous similar auctions. Thus, we propose a mechanism to develop an agent which does not depend on examples of previous interactions. Our agent performs *on-line learning* while interacting with others, and its performance is evaluated from the first interaction.

In addition, this study focuses on settings where in each interaction the agent is required to choose an action from a large set of possible options, which reflects commercial environments more realistically. The existence of a large set of options demands a construction of a fast and efficient screening procedure. Furthermore, the importance of quickness is emphasized in this study, since we examine short-term durations of only several dozens of interactions.

A recent work broadly discusses the situation of **non-simultaneous** repeated CE interactions with human opponents without using historical information [5]. In that study, the performance of several algorithms were evaluated using empirical data in various CE environments, such as single auctions and 2-players UG. The Deviated Virtual Reinforcement Learning (DVRL) algorithm was found to yield the highest performance in all the examined environments. In this paper we propose an approach for extending algorithms which are designated for non-simultaneous CE environments to function in simultaneous substitutional environments.[1] We show that the extended version of the DVRL algorithm, the Simultaneous DVRL (**SDVRL**), performs significantly better than other algorithms when competing against human opponents in 4 different environments. Likewise, it performs better than human competitors who face the same task. In addition, we propose an optimization of the SDVRL, namely the Fixed Success Probabilities algorithm (**FSP**), which uses a simple heuristic to find the optimal actions. It is noteworthy that the underlying mechanisms of the algorithms discussed here are all generic and suitable for various substitutional Cliff-Edge (SCE) environments. Moreover, in all the simulations presented herein, each algorithm was run with a fixed configuration setting of the basic parameters that was not changed from environment to environment.

In the next section, we formally describe the SCE set of environments. In section 3 we present the proposed SDVRL algorithm, and the FSP heuristic. In section 4 we survey other relevant algorithms, and compare their performance with the SDVRL's. In addition, we compare the latter with human performance executing the same tasks. We conclude and present directions for future work in section 5.

[1] Actually, our approach also suits simultaneous auctions for complementary goods. However, it is not discussed here due to space limitations.

2 The SCE Environments

The general pattern of one-shot SCE interactions considers an agent required to choose *K* offers $i_1,,,i_K$ which are all integers, $0 \leq i_j \leq N$, $1 \leq j \leq K$, where *N* is the maximal optional choice. Then, a positive reward *r* corresponding to the offers $i_1,,,i_K$ is determined, depending on whether the offers passed the acceptance thresholds set by the current opponents (in auctions we refer to the acceptance threshold of the auctioneer, which is the second highest bid proposed). Since the SCE set includes various environments, we detail the models of four environments upon which this paper focuses: substitutional auctions (**SA**), retractable substitutional auctions (**RSA**), one-accept multi-player UG (**MUG**) and retractable MUG (**RMUG**). The two latter environments are multi-player UG versions which reflect situations similar to the former two, respectively. Other SCE environments can be similarly modeled as well:

• In the **SA** model, *K* similar goods, each with a common value of *N* are simultaneously auctioned. The agent can offer *K* bids, $i_1,,,i_K$ which are all integers, $0 \leq i_j \leq N, 1 \leq j \leq K$. Given the bids, a reward *r* is determined according to the highest bids $b_1,,,b_K$ of all the other bidders (one or more) in each of the simultaneous auctions. If $1 \leq j \leq K$ exists s.t. $b_j \leq i_j$, then $r = N - \sum_{j|b_j \leq i_j} i_j$ (the value of attaining the item, subtracted by the winning bids), otherwise *r=0*. Similarly, in the **MUG**, the agent should divide an *N* amount among itself and other *K* players. It is required to choose integers $i_1,,,i_K$, $\sum_{j=1}^{K} i_j \leq N$, which are the amounts proposed to the other players. The reward *r* is determined by the opponents' acceptance thresholds $t_1,,,t_K$. If $1 \leq j \leq K$ exists s.t. $t_j \leq i_j$, then $r = N - \sum_{j|t_j \leq i_j} i_j$, otherwise *r=0*. Thus, the fewer the number of accepted offers (but at least 1), and the lower the offers - the higher the reward.

• In the **RSA** model, *K* similar goods, each with a common value of *N* are simultaneously auctioned. In another auction protocol which can be abstracted by this model, an item with a common value of *N* is auctioned in one auction, where each bidder can offer *K* bids. Thus, the agent can offer *K* bids, $i_1,,,i_K$ which are all integers,$0 \leq i_j \leq N, 1 \leq j \leq K$ and can retract any of its offers later. Given the bids, a reward *r* is determined according to the highest bids $b_1,,,b_K$ from all the other bids (one or more) in each of the simultaneous auctions. If $1 \leq j \leq K$ exists s.t. $b_j \leq i_j$, then $r = N - \min_{j|b_j \leq i_j} i_j$, which is the value of attaining the item, subtracted by the minimal bid that won the auction (we assume that the agent retracts all the other successful bids, since they incur higher costs), otherwise *r=0*. Similarly, in the **RMUG**, the agent should divide an *N* amount among itself and other *K* players. It is required to choose integers $i_1,,,i_K$ ($0 \leq i_j \leq N, 1 \leq j \leq K$), which are the amounts proposed to the other players, and later it can retract any offer, even if it was accepted. The reward *r* is determined by the opponents' acceptance thresholds $t_1,,,t_K$. If $1 \leq j \leq K$ exists s.t. $t_j \leq i_j$, $r = N - \min_{j|t_j \leq i_j} i_j$, otherwise *r=0*.

In this paper, we consider environments with a large set of decision options, and set *N* to be 100.

Obviously, the basic CE interactions contain a trade-off

between the expected reward and the probability of success: choosing an offer which increases the expected reward, decreases the probability of success, and vice versa. In the simultaneous environment, in addition, the agent should consider increasing the total reward obtained in all the simultaneous interactions. To demonstrate the challenge of a competitor in SCE environments, let *P(i)* be the probability that amount offered *i* succeeds (if the offer is higher than the other bids in the case of an auction, or is accepted by the responder in the UG) . Considering the basic case of *K=2*, an efficient agent must find the optimal offers i_1, i_2 for the opponent population, that maximize the accumulative utility function:

• In SA and in the MUG:

(1.1) $U(i_1,i_2) = P(i_1)P(i_2)(N - i_1 - i_2) + P(i_1)(1 - P(i_2))(N - i_1) + (1 - P(i_1))P(i_2)(N - i_2)$

• In RSA and in the RMUG:

(1.2) $U(i_1,i_2) = P(i_1)P(i_2)(N - min(i_1,i_2)) + P(i_1)(1 - P(i_2))(N - i_1) + (1 - P(i_1))P(i_2)(N - i_2)$

3 The Proposed Approach

In this section we present a detailed description of the proposed algorithms for competing in SCE environments. As mentioned above, our approach extends basic algorithms which are designated for non-simultaneous environments. We assume that the basic non-simultaneous algorithms select their actions according to an evaluation of the success probability, *P(i)*, of each offer, *i*, provided it is chosen. The evaluation of the success probability is determined according to the results of previous interactions.

Algorithm 1 THE GENERAL APPROACH

1: **For** each interaction, **Do**
2: Select offers i_1, i_2 according to a SELECT procedure
3: Observe results of the 2 offers, calculate reward
4: Update vector P according to the UPDATE procedure
5: **For** l_1=0 to N, For l_2=l_1 to N, **Do**
6: Update $U(l_1, l_2)$ according to the appropriate utility function (**1.1** or **1.2**) and the current $P(l_1), P(l_2)$ values.

Therefore, each basic non-simultaneous algorithm consists of its own UPDATE and SELECT procedures. The UPDATE procedure determines how to update the success probabilities vector, *P*, after observing the successfulness of the latest action. The SELECT procedure determines how to select the next action (apparently according to both the current expected utility evaluation, and considerations of optimality exploration). In **Algorithm 1** we outline our general approach. For the simplicity of the code we present the solution for *K*=2, which can be easily adjusted to higher *K* values. The main idea is to maintain a table, *U*, containing the expected utility of each combination of offers i_1, i_2.

3.1 The SDVRL algorithm

In this paper we claim that the best algorithm for human environments, as we will demonstrate, is the extension of the Deviated Virtual Reinforcement Learning algorithm (DVRL) [5], as presented in **Algorithm 2**. The main challenge of an on-line learning algorithm is to efficiently balance between the need for *exploration* of new options, and the will to *exploit* current information in order to maximize payoffs. The SDVRL, unlike most algorithms, distorts observed information in a manner which induces exploration. On the other hand, it selects its actions greedily and thus efficiently exploits its current information. According to SDVRL's UPDATE procedure we increase the evaluation of the success probabilities, P-values, of all the offers higher than a successful offer, as well as several offers below this offer, as though all these offers were also (virtually) successfully offered (line 10). Similarly, after an offer fails, we reduce the P-values of all the offers below the actual offer and several offers above the actual offer, as described in line 8. The success probability of each offer is calculated by dividing the number of previous successes by the total number of previous interactions in which the offer was actually or virtually proposed (lines 8,10).

Algorithm 2 THE SDVRL ALGORITHM

Notation: *n(j)* denotes the number of previous interactions in which offer *j* was actually or virtually proposed. α, β denote the deviation rate. Below we present the configuration used in our environment.

1: N=100, α=10, β=15, τ=0 **For** j=0 to N, **Do** P(j)=1, n(j)=0
2: **For** each interaction, **Do**
3: **If** t=0 **then** select i_1, i_2 uniformly, $0 \le i_1 \le i_2 \le N$
4: **Else** offers i_1, i_2=$\arg\max_{l_1,l_2} U(l_1, l_2)$
5: Observe results of the 2 offers, calculate reward
6: **For** each offer $i_v, 1 \le v \le 2$, **Do**
7: **If** offer i_v has failed **then** **For** j=0 to $(i_v + \alpha)$, **Do**
8: P(j) = $\frac{P(j)n(j)}{n(j)+1}$, n(j)=n(j)+1
9: **If** offer i_v has succeeded **then** **For** j=$(i_v - \beta)$ to N, **Do**
10: P(j) = $\frac{P(j)n(j)+1}{n(j)+1}$, n(j)=n(j)+1
11: τ = τ+2
12: $\alpha = \frac{10}{\lfloor \tau/10 \rfloor + 1}$ $\beta = \frac{15}{\lfloor \tau/10 \rfloor + 1}$
13: **For** l_1=0 to N, **For** $l_2 = l_1$ to N , **Do**
14: Update $U(l_1, l_2)$ according to the appropriate utility function (**1.1**) or (**1.2**) and the current $P(l_1), P(l_2)$ values

The SELECT procedure simply selects the offers with the current maximal U-value (line 4). The deviation principle underlying the UPDATE procedure enables a greedy SELECT procedure, since it induces fast exploration of the optimal offer. For simplicity's sake, consider the basic non-simultaneous case. If the agent, for example, offered 70% of the amount *N* to its UG opponent in the first interaction, and its offer was accepted, it would offer 55% (for β=15) in the next interaction. The agent continues to decrease its offer until it is rejected. This fast exploration process can be very efficient in SCE environments, as well. However, the "inaccurate" updating of the *P* values according to the deviation principle (lines 7-10) may cause misevaluations of the *U* values (line 14), which crucially rely on *P* and *(1-P)* values. Therefore, we examine the efficiency of the deviation principle in simultaneous environments by comparing

it with more guarded algorithms, as described in the next section.

In line 12 we gradually decreased the values of α and β since the model comes closer to the real distribution of the opponents population during the learning process. It is important to note that in this study SDVRL was run in all the environments with the same α and β configurations as detailed in lines 1 and 12. We assume that these values can be proportionally adjusted to other SCE environments with different *N* values.

The SDVRL method is simple and guarantees finding the optimal offers i_1, i_2 ($i_1 \leq i_2$) according to the current success probabilities evaluation, in each interaction. This is because it actually checks every possible combination of offers. However, this methodology is expensive: since in each interaction we compare $O(N^K)$ possible combinations. The total complexity equals $O(I \cdot N^K)$, where *I* denotes the number of interactions.[2] Thus, the naive algorithm is applicable for only a few simultaneous interactions, which is usually sufficient in common real-world applications. For high *K* values we may use heuristics or search algorithms, as described in the following section.

3.2 The FSP heuristic

In this section we propose a heuristic, termed *Fixed Success Probabilities* (FSP), which improves the complexity of SDVRL to $O(N^k)$. This heuristic is based on the observation that when an SDVRL agent interacts repeatedly in SCE environments, the **success probabilities** (P-values) of the chosen offers are almost the same during all the interactions. Note that the chosen offers themselves might be noticeably modified from one interaction to another, due to the updating of the P-vector. However, the corresponding P-values of the chosen offers are quite stable. A good demonstration of this phenomena can be viewed in **Figure 1**, where the amounts of the offers during 34 interactions in the SA environment (left figure) and their corresponding success probabilities, the P-values (on the right) are plotted. As can be clearly seen in the right figure, the P-values are almost constant (at around 0.005 and 0.98) from the very preliminary stages (unlike the offers on the left). Thus, rather than examine all the possible combinations of offers and calculate their utility at each interaction, we can simply choose 2 offers whose current P-values are 0.005 and 0.98. Therefore, the FSP algorithm (**Algorithm 3**) obeys the SDVRL method only in the first interactions (in all the SCE environments examined in our simulations we waited 5 interactions - lines 1-4, 18-19). After that (line 20) we calculate the P-values c_1, c_2 of the 2 offers chosen in the 5^{th} interaction (when $\frac{\tau}{2} = 5$, since τ is incremented by 2 at each interaction - Algorithm 2, line 11). From then on, the FSP chooses 2 offers with the same P-values, according to the current P-vector at each interaction (lines 6-9), with a complexity of O(N).

[2]It can be easily proved that in the retractable environments, the upper offer i_K should always maximize $P(i_K)(N-i_K)$ independently of other offers (for $K = 2$, for example, if $i_1 \leq i_2, U(i_1, i_2) = P(i_1)(N-i_1) + (1-P(i_1))P(i_2)(N-i_2)$).Thus, for these environments the complexity actually equals $O(I \cdot N^{k-1})$.

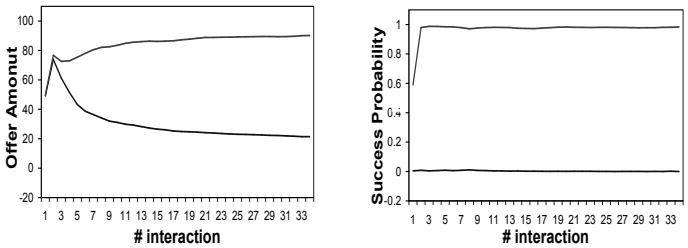

Figure 1. Amounts of offers (on the left) and their corresponding success probabilities (on the right) during 34 interactions in the SA environment with K=2 (average of 100 runnings)

Algorithm 3 THE FSP SDVRL ALGORITHM

1-3: As in **Algorithm 2**
4: **If** $\frac{\tau}{2} \leq 5$ **then** offers i_1, i_2=arg $\max_{l_1,l_2} U(l_1, l_2)$
5: **Else** j=0
6: **While** P(j)$< c_1$ **Do** j=j+1
7: offer $i_1 = j$
8: **While** P(j)$< c_2$ **Do** j=j+1
9: offer $i_2 = j$
10-17: As in **Algorithm 2** lines 5-12
18: **If** $\frac{\tau}{2} < 5$ **then** **For** l_1=0 to N, **For** l_2=l_1 to N , **Do**
19: Update $U(l_1, l_2)$ according to the appropriate utility
· function (**1.1**) or (**1.2**) and the current $P(l_1), P(l_2)$ values
20: **If** $\frac{\tau}{2}$=5 **then** c_1=$P(i_1)$,c_2=$P(i_2)$

It is worthy to note that the high complexity in the SDVRL stems from the need to search the K-dimensional combination of offers that maximizes the utility function during each interaction. This problem can also be solved at a lower cost using general search algorithms, such as the Nelder-Mead Simplex search, Brent's method and the Steepest Descent as well as discrete methods such as the Genetic Search and Hill-Climbing. However, in this limited framework we mainly discuss methods which concern the uniqueness of SCE environments. Moreover, the FSP heuristic can be integrated with any of these general optimization methods, which will save the need to perform searches in progressive interactions.

4 Experimental design and analysis

In this section we examine the performance of the proposed SDVRL algorithm competing in various SCE environments with human opponents, and compare it to the performance of other algorithms. A broad survey of the algorithms examined is presented, followed by a description of the experiment design and the results. After analyzing the results, we compare the performance of SDVRL to human negotiators' performance.

4.1 Comparative Algorithms description

In this section we survey several algorithms which were successfully used in repeated CE environments, and extend

them to the SCE environments. As detailed in the introduction, we could not find any algorithm which directly deals with the SCE environments considered here.

4.1.1 Gittins' index strategy

Some researchers have observed CE environments as a special case of the multi-armed bandit problem [1]. In the latter, every period a decision maker has to decide on which one of *n* slot machines he wants to play, given each machine has different gain probabilities (unknown a priori). These researchers used Gittins' indices strategy [4], which finds the optimal choice of action at each step in multi-armed bandit problems, for playing UG, considering each optional offer as an arm: For each action, consider the total number of times it has been chosen, *n*, and the number of times it has been successful, *s*. For certain discount factors of the expected reward, there are published look-up tables of indices, *G(s,n-s)* for each pair of *s* and *n-s*.[3] Each index represents a comparative measure of the combined value of the expected payoff of action *i* (given its history of payoffs) and the value of the information that we would attain by choosing it. However, the CE problems are different from the classical multi-armed bandit problem since the arms are independent, while in CE environments the success probability of an offer is depended on the size of the offer (i.e. the higher the offer the higher the probability). This difference may become more critical when there is a large number of options rather than the 10 optional arms problems considered in [1]. Thus, Katz and Kraus (2006) extended the basic Gittins' method using the deviation principle. Here, the difference from the SDVRL is in the updating of the P-values, which are calculated according to Gittins' indices. Thus, the P-values update in lines 8 and 10 of **Algorithm 2** should be changed to:

(2) *P(j) = G(s(j),n(j)-s(j))*

In addition, after successful offers, in line 10, we should increment s(j)=s(j)+1. We denote Gittins' version of SDVRL as Simultaneous Deviated Virtual Gittins - **SDVG**.

4.1.2 Segment-based approach

The segment-based approach [5] was found to be very efficient in non-simultaneous CE environments, though not as much as the DVRL. However, since it is more guarded and conservative than the deviation approach we decided to examine its performance in the SCE environments. In **Algorithm 4** we present the Simultaneous Segment-based extension of Virtual Reinforcement Learning (SSVRL).

In order to perform exploration, we use the ϵ-greedy version of Reinforcement Learning (RL) [12]. According to this version, at each interaction with a probability of $(1-\epsilon)$ we choose offers which supposedly yield the highest profits, according to the estimation of the current U-table. In order to explore new options, with a probability of ϵ, we select our offers uniformly. In addition, according to the *virtual* RL principle (without deviation), we increase the success probability evaluation, P-values, of all the offers higher than the actual offer, provided it was successful (line 9). Similarly, we should reduce the P-values of all the offers lower than the actual offer (line 7), provided it was unsuccessful (while with the deviation principle we also increase (reduce) several offers below (above) the actual offer).

The idea of the segment-based approach is to hierarchically divide all the options into segments, and in the initial interactions to activate the learning method on these segments, rather than on specific discrete options (line 3). After several interactions the resolution can be increased by focusing on smaller segments. This process continues gradually towards the last segmentation hierarchy with the smallest segments, i.e. specific discrete options, where final fine tuning is performed (line 4).[4] This algorithm is based on the assumption of locality, i.e. adjacent options yield similar average profits. The advantage of this approach is the

Algorithm 4 THE SSVRL ALGORITHM

Notation: H denotes the number of segmentation hierarchies configured by the user according to the size of the options set (the order of magnitude of the expected number of interactions, *I*, must be considered when setting this parameter), S_h is the size of each segment in the h^{th} segmentation hierarchy, and T_h is the serial number of the last interaction of the h^{th} segmentation hierarchy. P_h and U_h are the P-vector and the Utility table corresponding to the h^{th} segmentation hierarchy, respectively. ϵ denotes the probability for random selection.

1: For h=1 to H, **Do** For j=0 to $\lceil \frac{N}{S_h} \rceil$, **Do** P_h(j)=1,
2: For each interaction, **Do**
3: **If** $0 < \tau \leq T_1$: c = 1, $m_1, m_2 = \arg\max_{k_1,k_2} U_1(k_1,k_2)$
With a probability of (1-ϵ) select segments $j_1, j_2 = m_1, m_2$
With a probability of ϵ, select segments j_1, j_2 (from the 1^{st} hierarchy) uniformly
Select offers i_1, i_2 uniformly from segments j_1, j_2
...
4: **If** $T_{H-1} < \tau \leq T_H$: c=H,
$m_1, m_2 = \arg\max_{k_1,k_2} U_H(k_1,k_2)$
With a probability of (1-ϵ) select segments $j_1, j_2 = m_1, m_2$
With a probability of ϵ, select segments j_1, j_2 (from the H^{th} hierarchy) uniformly
Select offers i_1, i_2 uniformly from segments j_1, j_2
5: **For** each offer $i_v, 1 \leq v \leq 2$ **Do**
6: **If** offer i_v has failed **then**
7: **For** h=c to H, **For** $w = 0$ to $\lfloor \frac{j}{S_h} \rfloor$ **Do**
$P_h(w) = \frac{P_h(w)n(w)}{n(w)+1}$, $n(w) = n(w)$+1
8: **If** offer i_v has succeeded **then**
9: **For** h=c to H, **For** $w = \lfloor \frac{j}{S_h} \rfloor$ to $\lceil \frac{N}{S_h} \rceil$ **Do**
$P_h(w) = \frac{P_h(w)n(w)+1}{n(w)+1}$, $n(w) = n(w)$+1
10: $\tau = \tau$+1
11: **For** h=c to H, **For** l_1=0 to N, **For** l_2=l_1 to N , **Do**
12: Update $U_h(l_1, l_2)$ according to the appropriate utility function (**1.1**) or (**1.2**) and the current values of $P_h(l_1), P_h(l_2)$

[3] In our finite simulations we used the Bernoulli reward process table with a discount factor of 0.99 (see [4] p. 237), as was used in [1].

[4] In our simulations we used H=3 hierarchies. In the first T_1=5 interactions we focused on 5 segments of S_1=20 integers each. Then, till the 10^{th} interaction (T_2=10), we focused on 20 segments of S_2=5 integers each. From then on we focused on 100 segments of single offers (S_3=1).

gradual filtering of the optimal solution. A common problem in conventional RL, for instance, is that an option might have a high P-value in a progressive stage of the learning process, although it is far from the optimal option. With the segment-based approach, this situation is prevented already in preliminary stages, by weakening the P-value of the entire range surrounding this non-beneficial solution.

The segment-based approach can also extend Gittins' strategy, by modifying the bottom expressions in lines 7 and 9 according to expression **(2)**. We call this version, Simultaneous Segment-based Virtual Gittins - **SSVG** algorithm.

4.2 Experiment Description

In order to evaluate the performance of the proposed SDVRL algorithm, and to compare it with the other algorithms described above, we experimentally examined agents interacting with human opponents in the 4 SCE environments mentioned above: SA, RSA, MUG and RMUG. The examination of different environments guarantees generality and robustness of the results. It is important to mention that each algorithm was run with fixed parameter configurations (such as deviation rates α, β in the SDVRL and SDVG algorithms) for all the different environments. Though specific configurations for each environment could yield better performance, we preferred the generality of the algorithms over a variety of SCE environments, ensuring that no environment specific characterization would be used.

In the first experiment human participants were used as responders in the MUG and RMUG games, and as bidders in the auctions. Each person participated only once in each environment, while the automated agents interacted serially against different human opponents. Evaluating agents that were designed for human-involved environments by examining their performance with real human data is necessary. As mentioned above, human competitors do not obey subgame perfect equilibrium, and thus their behavior cannot be a priori simulated. Additionally, human behavior cannot be accurately statistically modeled, especially in small populations as in this case. Another benefit from empirical experiments is the ability to provide a concrete algorithm, with a concrete configuration of parameters, which successfully competes against human opponents. Thus, this agent can be immediately applied in real applications, at least as a starting point.

In the first stage of the experiment we surveyed 34 students who participated in non-simultaneous auctions, 13 students who participated in both SA and in RSA 2-offer auctions (in random order), and 34 students who played both MUG and RMUG (in random order). The participants were students at Bar-Ilan University, aged 20-30, who were not experts in negotiation strategies nor in economic theories directly relevant to the experiment (e.g. game theory, decision theory). In both 2-responder ultimatum games the participants were required to determine their acceptance threshold, i.e. the minimal offer they would accept as responders of the total NIS 100 to be divided among the 3 players (without knowing how much the other responder would be offered). In the auctions the participants were required to propose a bid, which could be any integer from 0 to NIS 100. The winner gained a virtual NIS 100. At the end of the experiment, each participant was paid between 15 to NIS 30, proportionally to her earnings in the interactions in which she participated.

After extracting the 34 simple auction bids, the 13 SA bids, the 13 RSA bids, the 34 acceptance thresholds in MUG and the 34 thresholds in RMUG, we constructed sets of opponents' reactions for each environment. For *k*=2 we constructed 2 sets of 34 integers, for *k*=3 3 sets of 34 integers, etc. In the UG environments each set was a random permutation of the 34 original thresholds. In the auctions each set contained random bids selected from all the auctions' environments. With this, we simulated a realistic auction environment, where part of the bidders bid simultaneously in several auctions, and part of them participated in only one auction. Therefore, we could use the same sets for the SA and for the RSA simulations. At this stage we examined the performance of each of the algorithms detailed above, which were run serially against the sets of opponents' data. Thus, each algorithm had one interaction with each of the human opponents in each of the four environments, without knowing in advance the number of interactions. Since there is importance to the order of the opponents, we constructed 100 random permutations of the human decisions series, for each environment, and compared the average payoffs of the different algorithms for each permutation. In addition, these algorithms were run against an artificial series of 50 auction opponents, constructed randomly according to a normal distribution of N(71,10). In this manner, we examined the performance of the algorithms with a theoretical population which distributes normally, though there is no evidence of such a distribution in any CE environment.

4.3 Experimental results

Table 1 presents the average payoffs for each algorithm, competing in the environments mentioned above, with *K=2*. The average payoffs were calculated based on the data of all the 100 permutations. Due to the fact that the algorithms' random factors cause a variation in the results, we ran each algorithm repeatedly for 30 times for each permutation.

Environment	SSVG	SSVRL	SDVG	SDVRL	FSP
SA	81.08	88.43	88.68	**94.5**	**95.21**
RSA	22.19	24.5	26.47	**27.12**	25.84
MOUG	47.49	**50.99**	46.79	49.6	48.78
RMUG	58.14	62.35	60.79	**64.3**	64.13
Normal distribution auction	86.8	89.96	94.92	**100.67**	**100.93**
Retractable normal distribution auction	16.99	19.94	20.45	**21.16**	19.79

Table 1. Average payoff of the various algorithms against human opponents in several SCE environments with *K=2*

The results show that the SDVRL algorithm was almost always the most profitable algorithm among the 4 basic algorithms. A non-parametric Friedman test revealed significant differences in the ranking of the algorithms ($p < .001$). Further pairwise Wilcoxon tests showed that the most efficient algorithm was significantly the SDVRL algorithm,

except for the MOUG, where SSVRL was significantly better (though the differences were not substantial, compared to the outstanding advantage of SDVRL in other environments). The FSP heuristic was found to be very efficient as well, and it even performed better than the basic SDVRL in the SA auction. In the normal distribution auction and in the RMUG no significant difference was found between the basic SDVRL and the FSP-SDVRL.

Environment	SSVG	SSVRL	SDVG	SDVRL	FSP
SA	173.89	176.15	159.3	**183.59**	**183.99**
RSA	32.62	33.26	32.54	**33.27**	32.56
MOUG	52.57	**53.05**	48.95	51.15	51.06
RMUG	63.12	68.59	68.99	**69.7**	69.65
Normal distribution auction	185.64	187.63	173.38	**194.43**	**194.45**
Retractable Normal distribution auction	24.39	24.57	23.45	**24.78**	23.59

Table 2. Average payoff of various algorithms against human opponents in several SCE environments with *K=3*

In **Table 2** we present the average payoffs in the same environments with *K=3*, i.e. 3-responder MUG and RMUG, and 3 simultaneous auctions. The opponents' responses were based on the same data of the *K=2* simulations. Consistent with the results of *K=2*, the SDVRL algorithm was usually the most profitable algorithm (Wilcoxon, $p < .001$). However, in the MOUG, SSVRL was significantly better, while in the retractable normal distribution auction and in the RSA no significant difference was found between SSVRL and SDVRL. The SDVRL using the FSP heuristic was found to be very efficient as well, and except for the RSA and the retractable normal distribution auctions no significant difference was found between the basic SDVRL and the FSP-SDVRL.

In order to compare the performance of the FSP-SDVRL algorithm with the performance of human competitors we asked 26 participants to compete iteratively against a series of other human opponents, exactly in the same manner the automated agents had competed. Thirteen participants were proposers in 2-responder MUG and RMUG, and the other 13 participated in SA and RSA auctions with *k*=2, in random order. After each decision, the participants were informed of the success of their 2 offers by the "current" opponents. Actually, these proposers were expected to learn the distribution of the responders' population, exactly as the agents had learned. All the learners, including the FSP agent, competed with the same series of opponents.

The results, which are not presented due to space constraints, showed a clear advantage of the FSP SDVRL agent (usually the naive SDVRL even performed better) over the human learners in all the environments. The average payoff of our agent was in average above the human's average payoff, with about 0.8 standard deviation. Thus, our agent succeeded in outperforming human natural intuitions and life experience in this context. Note that the automated agent was run with the same parameters for all the environments, and it did not apply domain specific knowledge.

5 Conclusion and future work

We have presented a new algorithm, namely SDVRL, which efficiently competes in various multi-interaction SCE environments with different human opponents. Our experimental findings show that the SDVRL performs better than humans and other algorithms surveyed in this paper. An optimization heuristic, which runs in most interactions in linear time was found to yield a high performance, as well.

In future work we intend to extend the approaches discussed in this paper to more sophisticated classes of interactions. Particularly, we would like to examine the repeated version of SCE interactions, in which several negotiation rounds can be conducted against the same opponents. When competing repeatedly against the same opponents, a specific modeling of each opponent must be done, in addition to the generic modeling of the population. Moreover, in contrast to the one-shot version, a current move may influence the future behavior of the opponents, a fact that must be taken into account. Therefore, we intend to design an agent that develops several optional models of typical opponents in the population, and matches the appropriate model to each opponent with which it interacts.

In addition, we intend to model a human's decision-making process when interacting in SCE environments, as was done for non-simultaneous CE environments [6].

Acknowledgements: This work is supported in part by NSF # IIS0222914 and ISF #1211-04. Kraus is affiliated with UMIACS.

References

[1] P. Bourgine and B. Leloup. May learning explain the ultimatum game paradox ? Technical Report GRID Working Paper No. 00-03, Ecole Polytechnique, 2000.

[2] D. Diermeier and R. Morton. Proportionality versus perfectness: Experiments in majoritarian bargaining. In D. Austen-Smith and J. Duggan, editors, *Social choice and strategic behavior*, pages 157–196. Berlin: Springer, Forthcoming.

[3] M. Dumas, L. Aldred, G. Governatori, and A. ter Hofstede. Probabilistic automated bidding in multiple auctions. *Electronic Commerce Research*, 5(1):25–49, 2005.

[4] J. Gittins. *Multiarmed Bandits Allocation Indices*. Wiley, New York, 1989.

[5] R. Katz and S. Kraus. Efficient agents for cliff-edge environments with a large set of decision options. In *AAMAS'06*.

[6] R. Katz and S. Kraus. Modeling human decision making in cliff-edge environments. In *AAAI'06*.

[7] M. Knez and C. Camerer. Outside options and social comparison in a three-player ultimatum game experiments. *Games and Economic Behavior*, 10:65–94, 1995.

[8] R. L. Milidiu, T. Melcop, F. T. S. Liporace, and C. J. P. Lucena. Simple - a multi-agent system for simultaneous and related auctions. In *IAT'03*.

[9] C. Preist, A. Byde, and C. Bartolini. Economic dynamics of agents in multiple auctions. In *Agents 2001*, pages 545–551.

[10] B. Szentes and R. W. Rosenthal. Three-object two-bidder simultaneous auctions: Chopsticks and tetrahedra. *Games and Economic Behavior*, 44:114–133, 2003.

[11] D. Yuen, A. Byde, and N. R. Jennings. Heuristic bidding strategies for multiple heterogeneous auctions. In *ECAI'06*.

[12] F. Zhong, D. Wu, and S. Kimbrough. Cooperative agent systems: Artificial agents play the ultimatum game. *Group Decis. Negot.*, 11(6):433–447, 2002.

[13] W. Zhu and P. R. Wurman. Structural leverage and fictitious play in sequential auctions. In *AAAI'02*, pages 385–390.