Auton Agent Multi-Agent Syst (2014) 28:290–336
DOI 10.1007/s10458-013-9227-z

Efficient bidding strategies for Cliff-Edge problems

Rina Azoulay · Ron Katz · Sarit Kraus

Published online: 30 April 2013

Abstract In this paper, we propose an efficient agent for competing in Cliff-Edge (CE) and simultaneous Cliff-Edge (SCE) situations. In CE interactions, which include common interactions such as sealed-bid auctions, dynamic pricing and the ultimatum game (UG), the probability of success decreases monotonically as the reward for success increases. This trade-off exists also in SCE interactions, which include simultaneous auctions and various multi-player ultimatum games, where the agent has to decide about more than one offer or bid simultaneously. Our agent competes repeatedly in one-shot interactions, each time against different human opponents. The agent learns the general pattern of the population's behavior, and its performance is evaluated based on all of the interactions in which it participates. We propose a generic approach which may help the agent compete against unknown opponents in different environments where CE and SCE interactions exist, where the agent has a relatively large number of alternatives and where its achievements in the first several dozen interactions are important. The underlying mechanism we propose for CE interactions is a new meta-algorithm, deviated virtual learning (DVL), which extends existing methods to efficiently cope with environments comprising a large number of alternative decisions at each decision point. Another competitive approach is the Bayesian approach, which learns the opponents'

Preliminary results of this research appear in [30] and [31].

R. Azoulay (✉)
Department of Computer Science, Jerusalem College of Technology,
Havaad Haleumi Street, POB 16031, Jerusalem 91160, Israel
e-mail: rrinaa@gmail.com

R. Katz · S. Kraus
Department of Computer Science and The Gonda Brain Research Center, Bar-Ilan University,
Ramat-Gan 52900, Israel
e-mail: katz@biu.013.net.il

S. Kraus
Institute for Advanced Computer Studies, University of Maryland,
College Park, MD 20742, USA
e-mail: sarit@cs.biu.ac.il

statistical distribution, given prior knowledge about the type of distribution. For the SCE, we propose the simultaneous deviated virtual reinforcement learning algorithm (SDVRL), the segmentation meta-algorithm as a method for extending different basic algorithms, and a heuristic called fixed success probabilities (FSP). Experiments comparing the performance of the proposed algorithms with algorithms taken from the literature, as well as other intuitive meta-algorithms, reveal superiority of the proposed algorithms in average payoff and stability as well as in accuracy in converging to the optimal action, both in CE and SCE problems.

Keywords Cliff-Edge problems · Human–agent interactions · Ultimatum game · Dynamic pricing · Reinforcement learning · Virtual learning

1 Introduction

An agent participating in real world situations often needs to make a decision about a certain offer or bid, where the probability of its success decreases monotonically as the expected reward increases. The attempt to maximize profits while preventing the entire deal from falling through is a great challenge in many games and economic interactions. This situation is somewhat similar to a person standing on the edge of a cliff, trying to see the panoramic view. The closer he comes to the edge, the better the view. However, one step too many may cause the viewer to fall off the cliff. Hence, interactions and games of this type are referred to as Cliff-Edge (CE) interactions.

We focus on an automated agent that repeatedly competes in one-shot CE interactions, each time against a different opponent. Since most of the participants in the CE interactions are either humans or agents who do not maximize their monetary payoffs [8,45], the agent is required to learn from its own experience how to interact with its future opponents.

We propose an innovative meta-algorithm which extends existing learning algorithms to efficiently compete in CE interactions. This meta-algorithm does not require any prior knowledge about people's behavior in each environment, and it can be used in any particular CE interaction domain.

We also consider simultaneous Cliff-Edge (SCE) environments, which include simultaneous auctions (for substitutes or complementary goods) and various multi-player games, and we propose learning meta-algorithms which efficiently compete in various SCE environments.

1.1 Examples of CE interactions

Repeated interactions occur, for example, in the important application of dynamic pricing. In the dynamic pricing domain, where prices change according to the reactions of consumers which change over time [16,23,35], the seller has to decide on the price of an item and the consumer needs to decide whether or not to agree to this price. If the consumer agrees to buy the item given its price, the transaction is performed and the seller obtains its profit, which is a positive function of the price. But, if the price is higher than the consumer's threshold, the buyer will not purchase the item and the seller's profit will be zero. Thus, a CE interaction exists from the seller's viewpoint: it may be beneficial to increase the price and thereby increase the seller's benefit from the trade, but once the price exceeds the unknown threshold the seller will "fall off the edge" and remain with no benefits at all.

CE interactions also exist in the first-price sealed-bid auction [10,19,53]. Sealed-bid auctions repeated periodically with the same goods are very popular nowadays, especially on the

internet (see [1,18,25]). Here the unknown threshold is the price suggested by the highest bid of the other bidders. Once the agent offers a bid higher than this threshold, it will win. As its bid is lowered but still remains higher than the threshold, the winner's utility increases, since it will have to pay less. But if the bid is lower than the threshold, then the bidder will not win at all and its utility will remain zero. Note that each interaction is performed against a different human opponent, thus no mechanism of signaling exists. However, all of the different human opponents come from the same population. Thus, the agent is able to learn the general pattern of the population's behavior.

In the ultimatum game (UG), [5,8,11,15,32,45,50], the agent has to suggest how to divide an amount of money between it and its responder. If the responder agrees to the offer, the amount will be divided according to the agent's offer. Otherwise none of the agents will obtain any division. The UG clearly belongs to the CE interaction, since here as well the proposer suggests how to divide an amount of money between it and its responder. As it enlarges its own part while reducing its opponent's part, its benefits will increase, but if it goes beyond the threshold, which is the minimum amount its responder will agree to receive, then the proposer will fall off the edge: the proposer's benefits will become zero, since the responder will not agree to an offer which is less than its acceptance threshold.

Given the CE interactions studied in this paper, each particular type of game may often be played for only a few dozen interactions. For example, in the dynamic pricing domain, the threshold distribution is different for each type of item sold in the shop, so the pricing of each particular item should be considered a different game. Thus, an agent representing a small or medium-sized e-shop is required to use a learning strategy which will attain a high expected utility for a set of a few dozen interactions for each particular item.

Since CE interactions occur in many practical situations, it is very important and useful in today's commercial world to design efficient agents for repeated CE interactions. In all of the domains studied in our paper, complete information exists about the amount of money and about the responder's preferences. However, the CE game can be viewed as an imperfect information game, since the learning agent placing the offer does not know which acceptance threshold its opponent will choose to use.

1.2 Simultaneous Cliff-Edge environments

The second problem we discuss in this paper is an agent that participates in SCE environments. In many auctions, substitutability or complementarity exists between two or more items for sale [6,17,22,38]. This situation is especially common in web-based auctions, where one can find many simultaneous auctions of similar popular items on different sites [7,51]. In such situations, many bidders have substitute preferences, namely they bid on multiple items in order to obtain only one of them. Other bidders have complementary preferences, i.e., they would like to win all of the simultaneous auctions, as in the case of purchasing airline tickets and hotel reservations for a group vacation (where the whole group is supposed to be on the same flight and stay in the same hotel).

Similarly, there are many economic and political situations in which one has to propose resource allocation among multiple opponents and reach an agreement for the allocation from at least one of them or, in other cases, from all of them [15,32].

1.3 Human behavior versus theoretical prediction

The current research deals mainly with interactions between automated agents and humans. This differs from multi-agent environments of pure computerized agents, where most

competitors have high computational power and act as rational negotiators. An agent operating with rational negotiators can compute the theoretical equilibrium of the game, which is reached when all the participants act according to their rational strategy. According to the equilibrium concept if all of the participants behave according to their rational strategy, no agent or agent designer will be motivated to deviate and use another strategy.

In contrast, human subjects are inherently rationally bounded as well as computationally restricted, and commonly they do not even attempt to maximize their expected monetary payoffs. They look for a bounded range of solutions and their behavior can be updated and changed due to reasons that cannot be completely modeled. As a result, they do not follow the perfect equilibrium concept, as was demonstrated by [45]. For example, in the case of the UG, any player that tries to maximize his expected monetary payoffs should agree to any offer greater than zero in a one-shot game situation (which is the case whenever a new opponent is reached in each round). However, experiments show that small offers will be rejected even in one-shot games; this fact cannot be explained when assuming players maximize their expected monetary payoffs.

As a result, the behavior of a human cannot be completely predicted: Two humans facing the same situation and the same prior knowledge may behave completely differently. Thus, no theoretical equilibrium strategy can be proven to be the best strategy given human participants, or given automated agent participants which, for some reason, are not designed to maximize their expected monetary value [36,37,44].

1.4 Previous approaches

Several previous studies concern automated agents that interact repeatedly with a single human opponent. Niklasson et al. [42] presented an artificial player based on a neural network algorithm that successfully plays the 'rock, paper, scissors' game against human players. Zhu and Wurman [53] proposed an agent that interacts repeatedly with the same opponent in sequential first-price sealed-bid auctions. The general approach for such a problem is to model the opponent's behavior and to use the best response strategy given the opponent's predicted reaction. In our research we consider a different environment, in which the agent interacts with the same opponent only once. Thus, modeling a specific opponent is not relevant, and the agent is not concerned with the long-term effects of its actions regarding each individual opponent. Consequently, we propose an agent which models the general pattern of the whole population of opponents in order to improve its performance when working with humans from the same population.

Other studies have proposed automated agents that interact with a human opponent population by using samples of previous interactions within the population. Gal et al. [21] used a version of the EM algorithm [14] and the gradient descent technique in order to estimate the opponent parameters in the colored trails game [21].

Davidson et al. [13] suggested using neural networks to learn how to play poker. The training data for the neural network was based on log files of real games played on the IRC poker server. Consequently, the neural network was able to predict the actions of real opponents with at least 80 % accuracy.

Dumas et al. [17] developed a probabilistic bidding agent that participates in multiple auctions. The agent gathers the bidding histories for each relevant auction house in order to predict the probability of each bid to win. Finally, Ghose and Tran [23] used the Feed-Forward Neural Network to study information on prices set by its competitors.

In these studies, learning the best strategy was determined using historical data from the human population. A model representing the population was studied, and the learned

model was used by the agent for future interactions. This approach, however, requires a large number of historical examples of human–human interactions. Moreover, the option of using a historical database is not always possible.

For example, in a sealed-bid auction, a bidder usually cannot obtain information about the bids which were offered in previous similar auctions. Therefore, 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 based on its average performance in all of its interactions.

Previous studies on CE interactions [5,35,45] focus on settings in which an agent has to choose from a small set of alternatives (no more than ten options during each interaction), and their solutions are extensively described in Sect. 3. Nevertheless, in our study, we are interested in settings whereby the agent must choose from a large number of alternatives each interaction.

A large number of alternatives reflects commercial environments more realistically, where the number of possible decisions is commonly much higher than ten (e.g. [16,19]). Unfortunately, as we show in this study, the performance of existing algorithms when choosing from a larger number of alternatives, for example 100, is insufficient. The existence of a larger set of alternatives demands construction of a fast and efficient screening procedure. Furthermore, since this study focuses on short-term interactions with only several dozen opponents, the importance of fast convergence is emphasized.

1.5 Our approach

The general idea of the proposed algorithm is to reinforce options which may be more profitable than the best option currently considered. Thus, in the conflict between exploration and exploitation of current information, we compromise with a small deviation from the current best option. This method, named deviated virtual learning (DVL), is extensively described in Sect. 4.

Due to the importance of CE interactions, several approaches for automatically competing in CE situations have been previously proposed [35,48,52]. In this paper we extensively survey these approaches and experimentally compare their performance when competing against human opponents in several variations of CE interactions.

We show that both the extension of the DVL meta-algorithm over the basic reinforcement learning (RL) algorithm [47] called the deviated virtual reinforcement learning algorithm (DVRL), and the Bayesian approach [12] under normal distribution perform better than previous approaches in most of the examined environments. In particular, in the auction domain, the deviated Gittins algorithm (DVG) performed the best. The DVG performed the best in the auction, the DVRL performed the best in the UG variant and in the other games the Bayesian approach under normal distribution performed the best, with results slightly better than the DVRL.

Performance was evaluated according to three criteria: the obtained reward, the stability and consistency in different situations and the accuracy in converging to the optimal action.

The algorithms discussed here are all generic and suitable for the different types of CE interactions, without taking into account the specific characteristics of each environment. Thus, the algorithms can be used by different agents participating in different kinds of CE games, without any need to provide the specific type of game (auction, UG, etc.).

For the SCE interactions, we propose two meta-algorithms: (a) the simultaneous deviated virtual reinforcement learning algorithm (SDVRL), which is an extension of the DVL

algorithm and (b) the segmentation meta-algorithm as a method for extending different basic algorithms. Finally, in order to reduce the complexity of the SDVRL algorithm, we provide a heuristic called fixed success probabilities (FSP).

The paper is organized as follows. In Sect. 2, we formally describe the CE interaction. In Sect. 3 we present different approaches for the Cliff-Edge environments and in Sect. 4 we present our proposed approaches, based on the deviation principle, which we have found to be beneficial in most of the CE environments. In Sect. 5 we describe comparisons of the performance of the different approaches given simulations which are based on real data obtained from humans, and we provide a theoretical analysis of the simulation results.

Thereafter we discuss the SCE problems. In Sect. 6 we introduce and formally define these problems, and we provide examples of SCE environments. In Sect. 7 we present the proposed algorithm for the SCE interactions and in Sect. 8 we compare the different approaches and analyze the results. Our conclusion and suggestions for future work are given in Sect. 9.

2 The Cliff-Edge model

In the following section we formally describe and provide some real-world examples of the CE problem. Recall that the group of CE interactions includes real world situations in which an agent needs to decide about a certain offer or bid, given a new opponent in each interaction. Given the new opponent, the agent knows that as it proposes an offer which is better for itself, its reward may increase, but one step too many may cause it to lose all it has gained hitherto. This situation is somewhat similar to a person standing on the edge of a cliff, as explained in the introduction. When considering that the threshold of the opponent is unknown, a trade-off exists between the probability of success and reward: the probability of the agent's success gradually decreases monotonically as the expected reward increases.[1]

The general pattern of one-shot CE interactions considers an agent that is required to choose an offer i, which is an integer $0 \leq i \leq N$, where N is the maximal choice. Then a positive reward $R(i)$ corresponding to the offer i is determined, depending on whether the offer passed a certain acceptance threshold τ set by the opponent. In the rest of the paper we use the following notations: $S(i)$ denotes the reward of an agent when its offer i wins, $F(i)$ signifies the rewards of an agent when its offer loses, and $P(i)$ is the probability that an offer i will succeed.

Since the CE set includes various environments, we detail the model of each of the three environments mentioned in this paper: auction, UG and dynamic pricing. Other CE environments can be similarly modeled.

- In the UG, the agent suggests how to divide an amount of N between itself and its opponent. It chooses an integer i, $0 \leq i \leq N$, which is the amount proposed to the opponent. We assume that each responder has an acceptance threshold τ, where it will accept any offer equal to or higher than τ. Based on this assumption, the reward r is determined as follows:

$$R(i) = \begin{cases} N - i & \tau \leq i \\ 0 & otherwise \end{cases}$$

[1] Note that each interaction is a one-shot game and is performed against a different human opponent; consequently no opponent has learning opportunities.

Table 1 Attributes of different CE interactions

Environment	Meaning of N	Success probability	Winner's reward $S(i)$	Loser's reward $F(i)$
UG	Amount to divide	$P(\tau \leq i)$	$N-i$	0
First price auction	Common value	$P(\tau \leq i)$	$N-i$	0
All-pay auction	Amount to divide	$P(\tau \leq i)$	$N-i$	$-i$
Dynamic pricing	Common value	$P(\tau \geq i)$	i	0

- In the sealed-bid first-price auction model an amount N is auctioned.[2] The agent is required to choose its bid, which is an integer i, $0 \leq i \leq N$. Given the bid, a reward $R(i)$ is determined according to the highest bid, τ, among all of the bidders in the current auction excluding the bid of the agent.[3] In other words, the opponent of the agent is considered to be the bidder with the highest bid excluding the agent's bid.

$$R(i) = \begin{cases} N-i & \tau \leq i \\ 0 & otherwise \end{cases}$$

- In the all-pay auction, all of the bidders must pay their bids, even if they do not win the auction [33]:

$$R(i) = \begin{cases} N-i & \tau \leq i \\ -i & otherwise \end{cases}$$

- In the dynamic pricing task, an agent is required to select a price i for an item being sold to a certain consumer (i actually represents the profit of the seller, beyond the cost). In our model, i is an integer $0 \leq i \leq N$, where N is the maximum reasonable price for the item. After a decision is made about the price, a reward, $R(i)$, is determined according to the current consumer's acceptance threshold τ.

$$R(i) = \begin{cases} i & \tau \geq i \\ 0 & otherwise \end{cases}$$

Table 1 summarizes the attributes of the four considered models, given a threshold value τ. We can observe that in most of the CE interactions, the point at which the offer equals the unknown threshold is the CE point. The exact behavior of the reward function depends on the model. It can be linear until the threshold and then zero after the threshold (as in the dynamic pricing task), or it can be zero until the threshold and linear after the threshold (as in the UG), or it always can have a negative slope (as in the all-pay auction). But, in all cases, the threshold is the private knowledge of the opponent and is not known to the agent offering the bid. As observed in Table 1, there is an obvious 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. Similarly, in the all-pay auction, which is the only case where $F(i)$ can be negative and not equal to zero, choosing an offer which decreases the expected loss, increases the probability of failure. Figure 1 demonstrates the different payoff functions of the UG, the all-pay auction and the dynamic pricing game (the payoff function of the auction game appears to be equal to that of the ultimatum game).

[2] In order to avoid differences in the valuation of the item sold to the bidders and incomplete information about it, in this case the item auctioned is an amount of money [19].

[3] For reasons of compatibility with other environment models, equal bids yield a gain for the learning agent.

Fig. 1 The agent's payoff as a function of the agent's bid in CE interactions. The x axes denote the agent's offer, and the y axes denote the agent's payoff for a particular threshold of 0.5

For demonstration purposes, in all cases the threshold was set to 0.5. We can see that a CE exists at this point, and the agent's challenge is to increase its benefits without falling off the edge.

Although most of the problems have a similar structure concerning the agent's point of view, we should emphasize that the opponent in each problem has different motivations and a different pattern of desired actions. Consequently, although the same learning methods may be used for the different CE environments, the behavior of the learning process, as well as its performance, may be different for each particular environment.

3 Prior proposed algorithms for CE interactions

In this section, we survey several prior approaches used for CE interactions. We describe the general attributes, the advantages and the disadvantages of each approach. Additional details of the algorithms can be found in Appendix A.

3.1 The Monte-Carlo approach

The naive approach for reinforcement learning problems is to apply the Monte-Carlo (MC) method (see [47], Chap. 5). According to this method, each option is examined many times, in a serial or random order, and finally (the later the better) a model is constructed from the success probability (P) distribution of the opponent population. The MC method is considered to be a naive approach since it does not attempt to speed up the learning process, but instead tries all possible bids many times until a model is learned. Clearly, after a large number of trials, the success probability distribution will be correctly learned. However, the learning process is extremely time-consuming. Moreover, the MC method does not eliminate inefficient actions. For example, an efficient agent will quickly realize that most responders in the UG would accept a share of 50 %, and thus a proposal of more than 50 % would not be advised. Therefore, we would rather use a less naive approach in which directed exploration is conducted. Nevertheless, for a very large number of interactions, the MC method may provide a more precise model of the real success probability distribution in the population than algorithms which use directed exploration.

In this study we are more concerned with finding fast and efficient approaches for the first several dozen opponents. Handling the initial interactions when there is lack of experience with previous results is obviously the most difficult stage. Moreover, considering a larger number of interactions, the severity of the accuracy problem caused by directed exploration can be reduced by combining a sophisticated algorithm with a naive one. A method such as MC may replace a more sophisticated algorithm in the progressive interactions, based on previous interactions conducted by the latter.

3.2 The Bayesian approach

Strens [46] suggests combining dynamic programming with reinforcement learning using the following process: the process parameters' posterior distribution is represented, and a greedy behavior is used with respect to a sample from this posterior. The posterior probability density is obtained from the prior density and the new action's results by applying Bayes' theorem. Their model is also applicable to MDP models with multiple states and different expected results for each state.

Conitzer and Garera [12] propose using Bayesian inference to deal with the online principal–agent problem, which also belongs to the CE set. In a principal–agent problem, a principal seeks to motivate an agent to take certain actions which are beneficial to the principal, while spending as little as possible on the reward. The underlying assumption of the authors is that the opponent's population is drawn from a fixed distribution. Moreover, the authors assume that the principal has a prior belief regarding this distribution (whether it is uniform or exponential).

In each round the principal chooses the reward that maximizes her expected utility, given her beliefs regarding the agents' distributions. The principal then updates these beliefs based on the observed result, using Bayes' rule. So, if the principal assumes that the type of distribution is exponential with parameter λ, she can learn the parameter's value during the interactions. In Conitzer and Garera's paper, the principal initially believed that the parameter λ is equally likely to be any member among $0.00, 0.01, \ldots, 10.00$. The general Bayesian approach is described in Appendix A.3.

The main drawback of the Bayesian approach is the assumption about the opponent's distribution. Prior knowledge about the type of distribution of the population is not guaranteed in many domains, especially when interacting with merely dozens of opponents. In the Cliff-Edge domains, the thresholds are assumed to be drawn from a group of thresholds, with a certain distribution. However, assuming a specific type of distribution, such as normal or exponential distribution, limits the ability to learn the accurate threshold distribution if it is not the type assumed in the algorithm.

Nevertheless, we decided to empirically examine the performance of the Bayesian-based agent and compare it with the other learning methods. We present the results of our comparison, in Sect. 5. Our findings indeed show that the Bayesian approach performs poorly under the exponential distribution assumption.

Next, we implemented the Bayesian approach under the normal distribution assumption. The general approach in our implementation is the same as that in [12].[4] In our algorithm framework, the SELECT procedure chooses the offer which maximizes the agent's benefits under the current beliefs about the distribution parameters and the UPDATE procedure updates the agent's beliefs given the last result, using Bayes' rule, but the underlying distribution was normal.

In fact, the threshold's distribution is considered to be discrete, so its distribution is not Gaussian, but it is relatively close to the Gaussian distribution in the different domains we considered. Thus, we use the Bayesian algorithm under the normal distribution as a heuristic, in order to be able to update the probability value of each possible threshold.

[4] In the work of [12], the exact threshold of the worker is not observed, and the worker simply decides whether or not to work, given the offer (salary) it obtained from its principal. Consequently, the principal's beliefs are updated given the result of the success or failure of the offer. Similarly, in our case, the user threshold is not observed and the result of an offer is a success (acceptance of an offer) or a failure of the offer. Hence the beliefs about the society's distribution are in line with the Bayesian approach given this Boolean result.

When running our simulation over this normal distribution with the Bayesian approach, we discovered that the results of implementation of the Bayesian approach when the underlying threshold distribution is normal are competitive with the best heuristic methods. Our detailed results are presented in Sect. 5 and details of the normal distribution Bayesian algorithm are provided in Appendix A.4.

3.3 Directional learning theory

Directional learning theory is a qualitative reasoning process in which the agent considers ex-post whether it was able to obtain a higher payoff by choosing a different offer, and then it is able to change its strategy in order to improve its results. According to the directional learning theory, if an offer i is rejected at interaction t, then at interaction $t+1$ the proposer will offer its opponent a higher offer. If offer i is accepted at interaction t, then at interaction $t+1$ the offer will be lowered.

This simple method, which was used in Brenner and Vriend's experiments [8], ignores all the data accumulated before the previous interaction. If all of the opponents behave in exactly the same way, then this method will reach the optimal solution. But, with dynamic opponents, it performs relatively poorly. Simulations of the directional learning method in our environment did not display impressive performance either.[5] On the other hand, in the algorithms proposed in the current study, all of the previous interactions are taken into account, and the model comes closer to the real distribution of the opponent's population during the learning process.

3.4 Roth-Erev reinforcement learning algorithm

Concerning the UG, Roth and Erev [45] designed a human proposer model based on basic reinforcement learning (RL) [47]. Reinforcement learning algorithms are extensively used in artificial intelligence, and are based on the idea that learning is done during interaction with the world. The agent has to choose between several options, and it has to discover which option is the best one by trying them. The RL algorithm suggests which option to choose each time, and which grade to assign to each option. The general idea of Roth and Erev is that each offer is chosen with a probability according to its current Q-value—the larger an offer's Q-value, the higher the probability it will be chosen. After each interaction, provided that the last offer succeeded, the probability of it being chosen again is increased by increasing its Q-value by the reward this offer yields. The algorithm's details are presented in Appendix A.2.

Roth and Erev showed that despite the fact that the UG is characterized by a significant gap between empirical human behavior and perfect equilibrium predictions, the RL algorithm can, with a few additional modifications, successfully model typical human empirical behavior in the UG. The most noticeable modification, based on the persistent local environment principle, uses a "generalization" parameter which prevents the probability of an option receiving a score of zero if it is adjacent to a successful option.

The criterion of adjacency was configured in [45] at -1 and $+1$. Thus, if the agent offers four and this offer is accepted by the opponent, the agent reinforces the Q-values of offers 3,

[5] A similar approach, namely the derivative-following strategy, was proposed for the dynamic pricing problem [16]. However, each round (one commerce day) was considered to be a set of several interactions with a number of opponents, which totaled thousands of interactions.

4 and 5. Their assumption, which is valid in all of the CE interactions, was that an adjacent option of an optimal solution will usually be a good solution as well.

Following Roth and Erev, the relevance of options close to the chosen option is considered also by the DVL meta-algorithm, as described in Sect. 4.

3.5 A modified version of Zhong, Wu and Kimbrough's (ZWK) RL algorithm

Zhong et al. [52] developed an agent that plays the UG against various simulated opponents, using a version of RL which is different from the Roth-Erev algorithm (R&E). The UG agent designed by these authors used an ϵ-greedy selection procedure whereby the agent, with a probability of (1-ϵ), selects the offer with the current maximal Q-value, and with a probability of ϵ it uniformly selects an offer for exploration purposes.

The advantage of this selection procedure over a selection according to the Q-value's distribution as in R&E is that it usually chooses the best option. The disadvantage is that the exploration is not weighted by the estimation of expected utility considerations, as in R and E but, rather, is completely random. In order to decrease the effect of this disadvantage, we modified the selection procedure in our experiment to take into account the generalization principle of R&E. Thus, in the case where the offer with the current maximal Q-value is not selected, our ZWK algorithm selects an offer giving higher priority to offers which are adjacent to the offer with the current maximal Q-value, rather than selecting them uniformly.

In addition, since the exploration is much more important in the initial interactions, we gradually decreased the probability of random selection during the learning process. These two modifications were examined and were found (in preliminary testing) to be significantly efficient in our environment.

The UPDATE procedure in ZWK's algorithm is also different from that of R&E in that it takes into account the previous failures of offer i, rather than only the previous positive rewards. The details of this algorithm are presented in Appendix A.5.

3.6 Virtual learning (VL)

A reasonable approach for the UPDATE procedure in CE interactions is *Virtual Learning* (VL) [43,49]. The idea of virtual learning is that instead of learning only on the basis of actions and payoffs actually experienced, the agent can also learn by reasoning from the chosen actions and other actions. That is, given the results of the foregone actions and payoffs, the agent can conclude about other close actions that were not chosen yet, and it can reinforce its propensity to choose actions as if it had actually tried those actions and experienced the corresponding payoffs [49].

For example, in the UG—according to the VL principle, the agent treats all of the offers (not actually proposed) higher than a successful offer as if they were (virtually) successfully proposed as well. Similarly, it considers all of the offers lower than an unsuccessful offer as unsuccessfully proposed. The rationale behind this principle is that if some offer x was accepted by a particular user, then clearly each offer that is better for the user would also be accepted. Thus, if we consider the UG, we have to reinforce the Q-values of all of the offers higher than the actual offer, provided it was successful, and reduce the Q-values of all of the offers lower than the actual offer, provided it was unsuccessful.[6]

[6] The same rationale is behind the auction environment. However, in the pricing environment the opposite is true: all of the offers lower than a successful offer are considered successful, since the lower the price of an item, the higher the probability it will be purchased by the consumer.

A crucial problem with the VL approach is *information asymmetry* [49], which causes intensified reinforcement of Q-values higher than the optimal option's Q-value. The reason for this phenomenon, as demonstrated in [49], is the *information asymmetry problem*: If offer x is accepted, the agent can reason about offers higher than x (with a higher probability of being accepted), but it does not receive information about what will happen with offers lower than x (with a lower probability of being accepted). Similarly, if offer x is rejected, the agent can reason that offers lower than x would also be rejected, but it has no information about offers higher than x.

As a result, whenever the optimal offer is successfully chosen, offers which are higher than the optimal offer are reinforced as well. However, when the optimal offer fails, it is punished. The offers slightly above do not suffer any penalty, giving them the appearance of having a much higher probability of success than their true probability.

3.7 Todd and Borger's virtual RL algorithm (VRL)

Despite the information asymmetry in VL found by [49], Todd and Borgers [48] have shown that the addition of a virtual learning mechanism to the RL algorithm when playing the UG may be very efficient. However, the efficiency depends on several configurations of the learning procedure, such as the selection procedure (R&E's or ZWK's) and the Q-values update procedure.

After examining the performance of different VRL variants suggested by Todd and Borgers, the best performing version of the algorithm is presented in Appendix A.6.

Note that according to the classic virtual learning approach, we can ignore the offers which are not expected to change in the same direction as the selected offer (i.e., alternatives which are much lower than a successful selected offer and alternatives which are much higher than an unsuccessful selected offer). However, we found in our empirical results that it is better to consider these offers and to change them in a manner opposite the results, in order to be able to concentrate on the range around the successful offer.

4 The deviated virtual learning principle

In order to overcome the information asymmetry problem described in Sect. 3.6, we propose the *Deviated Virtual Learning* (DVL) principle. The Q-values in the DVL algorithms describe the quality of each choice, as learned by the system, where the quality is influenced both by the results of the choice itself and by the results of its neighbors. The Q-values are updated according to the following concepts:

Whenever an offer i succeeds, the following updates are performed: (a) its quality value Q increases; (b) the quality value of each offer with a higher success probability than offer i, increases; (c) the quality value of each offer j which is close enough to offer i increases (also including offers with a lower winning probability than that of offer i itself);

Whenever an offer i fails, the following updates are performed: (a) its quality value decreases. (b) the quality value of each offer with a lower probability of success than offer i, decreases. (c) the quality of each offer j which is close enough to offer i decreases, including those with a higher winning probability than offer i itself. The size of the neighborhood which has to be considered depends on the round number, which decreases over time. Thus, for each offer i, the number of offers j which are considered close enough to offer i decreases over time.

Parts (a) and (b) of the strategy are obtained from the VL technique. Part (c) is our own extension, which uses the proximity heuristic, i.e., if an offer is accepted, then the offers which are close to it will probably be accepted as well.

4.1 DVL meta-algorithm

The DVL principle can be implemented in various basic algorithms, thereby producing a general meta-algorithm, as presented in Algorithm 1. According to the DVL principle, we need to deviate from the strict rationale underlying the VL principle and extend the range of offers updated as successful/unsuccessful after each interaction. Thus, after successfully offering amount i, we increase the Q-values of all of the offers higher than the actual offer i, as well as $\lfloor \frac{i}{round+1} \rfloor$ offers **below** the actual offer, as described in line ten.

At the same time, we reduce the Q-values of all of the offers lower than this new threshold (line nine). Similarly, after an unsuccessful interaction, we reduce the Q-values of all of the offers above the actual offer i, and we also reduce the Q-values of some offers **above** i, up to $\lfloor \frac{N-i}{round+1} \rfloor$ (line 13). Furthermore, all of the offers above this new threshold are reinforced (line 14).[7]

Algorithm 1 THE DVL META- ALGORITHM

Notation: N is the upper bound of possible offers. SELECT is the procedure which selects an offer i. UPDATE is the updating procedure which is performed after observing the results of offer i. These procedures are implemented individually by each DVL-based algorithm.

1: *round=1*
2: **For** j=0 to N, **Do** Q(j)=1
3: **For** each interaction, **Do**
4: **If** *round=1* **then** select i uniformly, $0 \leq i \leq N$
5: **Else** Select offer i according to a SELECT procedure
6: Observing opponent's move, calculate reward
7: **If** offer i has succeeded **Then**
8: **For** j=0 to N, **Do**
9: **If** j $\geq$(i **-** $\lfloor \frac{i}{round+1} \rfloor$) increase Q(j) according to an UPDATE procedure
10: **Else** reduce Q(j) according to an UPDATE procedure
11: **If** offer i has failed **Then**
12: **For** j=0 to N, **Do**
13: **If** j <(i **+** $\lfloor \frac{N-i}{round+1} \rfloor$) reduce Q(j) according to an UPDATE procedure
14: **Else** increase Q(j) according to an UPDATE procedure
15: *round=round+1*

Note that a major advantage of the DVL principle is the fact that it does not require any prior knowledge about the population distribution or about the probability of the particular offers to be accepted, given that offer i was accepted or rejected. Thus, it can be used generally, in any particular CE interaction domain.

There are two main motivations for the deviation principle underlying the DVL approach for the reward selection problem. First, it is not necessarily a mistake to consider an offer which is slightly lower than a successful offer to be successful as well. Using the UG terminology, it is quite reasonable to assume that if proposal i was accepted (rejected) by an opponent, she would have also accepted (rejected) a slightly lower (higher) proposal. The

[7] It is worth noting that all of the VL-based algorithms in this paper are presented in a version suitable for auctions and for the UG. In the pricing environment, where increasing the price decreases the acceptance probability, the update ranges in lines 8 and 9 (as well as 12 and 13) need to be swapped.

chance that the proposal would exactly hit the acceptance threshold of the opponent is low, especially for a large number of alternatives. However, we assume that during the learning process the evaluated model comes closer to the real distribution of the opponent's population, and therefore the deviation extents (the size of neighborhood of each proposal) are gradually decreased (by positioning *round + 1* in the denominators in lines 9 and 13). The exact functions used for updating Q and for the neighborhood of i that is updated in the case of success or failure of i are heuristics, which take into consideration the number of the round. These heuristics ensure that as the agent becomes more experienced, the changes caused after each interaction become more focused.

The second, and even more important motivation, is related to the issue of "exploration vs. exploitation". 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 DVRL distorts observed information in a manner which actually outlines a direction of the optimal solution searching, as opposed to the random trial-and-error approach that underlies conventional methods, such as RL. A DVRL agent that offers 80 % of the cake (N) to its UG opponent in the first interaction, which is accepted, will offer 40 % in the next interaction. The agent will continue to decrease the offer until it is rejected. A similar pattern of this directed searching was observed in human learning, which is compatible with the directional learning theory (see [26]).

According to the virtual learning approach, after each step the learning agent not only learns about the offer that was suggested, but about other offers as well. In particular, it can reason about offers with a higher probability of winning (w.r.t. the chosen offer) in case of winning, and it can reduce offers with a lower probability of winning (w.r.t. the chosen offer) in case of losing. The deviation version also adds the ability to reason about offers from both sides (offers with a higher or a lower probability of winning) of the neighborhood of the chosen offer.

4.2 Deviated reinforcement learning algorithm (DVRL)

As mentioned in Sect. 4.1, the DVL meta-algorithm can extend any basic algorithm according to its SELECT and UPDATE procedures. However, we will demonstrate that the best algorithm for human environments, among the algorithms examined here, is the DVL extension of the reinforcement learning algorithm [47] (DVRL), as presented in Algorithm 2.

The SELECT procedure for the DVRL is called a greedy policy: this selection rule simply selects the action (or one of the actions) with the highest estimated action value [47]. Thus, the SELECT procedure merely chooses the offer, i, with the highest Q-value (line 5).

The UPDATE procedure (lines 9, 10, 13 and 14 of Algorithm 2) specifies how the Q-values of all possible offers will be changed in each stage. In particular, it specifies the value of the offers which need to be changed in the same order as the offer chosen (and succeeds or fails), as well as the values of the other offers (below the range of a successful offer, or above the range of an offer that failed). As the set of prior interactions (rounds) increases, the range of the neighborhood decreases.

For example, in a case where offer i is successful, if offer j is above or in the neighborhood of offer i, then $Q(j)$ will be updated as follows:

$$Q_{new}(j) = \frac{Q_{old}(j)(round - 1) + S(j)}{round}$$

Algorithm 2 THE DVRL ALGORITHM

Notation: N is the upper bound of possible offers. S(j) denotes the corresponding reward for a successful offer j and F(j) is the corresponding reward for offer j when it fails.

1: *round=1*
2: **For** j=0 to N, **Do** Q(j)=1
3: **For** each interaction, **Do**
4: **If** *round=1* **then** select i uniformly, $0 \leq i \leq N$ *line 4-5: SELECT procedure*
5: **Else** offer i=arg max$_j$ $Q(j)$
6: Observing opponent's move, calculate reward
7: **If** offer i has succeeded **Then** *line 7-14: UPDATE procedure*
8: **For** j=0 to N, **Do**
9: **If** j $\geq$(i - $\lfloor \frac{i}{round+1} \rfloor$) $Q(j) = \frac{Q(j)(round-1)+S(j)}{round}$
10: **Else** $Q(j) = \frac{Q(j)(round-1)+F(j)}{round}$
11: **If** offer i has failed **Then**
12: **For** j=0 to N, **Do**
13: **If** j $<$(i + $\lfloor \frac{N-i}{round+1} \rfloor$) $Q(j) = \frac{Q(j)(round-1)+F(j)}{round}$
14: **Else** $Q(j) = \frac{Q(j)(round-1)+S(j)}{round}$
15: *round=round+1*

(see line 9 in Algorithm 2). The above formula can also be expressed in the general form of reinforcement learning [47]:

$$Q_{new}(j) = Q_{old}(j) + \frac{1}{round} * (S(j) - Q_{old}(j)).$$

where $1/round$ decreases as the number of rounds increases.

The UPDATE procedure given in Algorithm 2 updates the neighbors of an offer according to the result of the offer. In the case of a successful offer j, the algorithm will also increase the Q values of some neighboring offers with a lower probability of winning; in the case of a failed offer j, the algorithm will also reduce the Q values of some neighboring offers with a higher probability. The ratio of the neighborhood to be updated is $\frac{1}{round+1}$. Note that as the $round$ increases, a small set of neighbors will be changed given the result of each offer. Different values can also be considered. Note also that different neighborhood factors can be considered rather than $\frac{1}{round+1}$, and in Sect. 5.3 we compare the algorithm's results for different possible neighborhood factors.

4.3 Gittins' strategy and deviated-Gittins' strategy (DVG)

Some researchers have considered CE interactions as a special case of the multi-armed bandit problem [2,5,12,35]. In every period of this domain, a decision-maker has to decide on which one of n slot machines he wants to play, given that each machine has different gain probabilities (unknown a *priori*).

For the multi-armed bandit problem, different approaches have been suggested. Auer et al. [2] suggested some heuristics to deal with it. Other researchers [3,5,35] used Gittins' indices strategy [24],[8] for the multi-armed bandit problem, to model the behavior of proposers in the UG and the behavior of sellers in the pricing problem, considering each optional offer or price as an arm. According to Gittins' strategy, for each optional offer we consider the total number of times it has been chosen, n, and the number of times it has been successful,

[8] The difference between the Bayesian approach and Gittins' strategy is that Gittins' model is based on some particular properties which are not required in the general Bayesian approach. Some of these properties do not hold, in fact, in the CE domain, so it is used as a heuristic method.

x. For certain discount factors of the expected reward, look-up tables of indices, *G(x,n-x)*, are published for each pair of x and n-x.[9]. $G(x, n-x)$ is an index value, computed by Gittins [24], for calculating the value of choosing an arm in a multi-armed bandit problem, where the arm has previous x successful events and $n - x$ failure events. As the value of G is higher, the benefit of choosing this arm is also higher. 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.

The CE interactions discussed in this paper are different from the classical multi-armed bandit problem. In the bandit problem the arms are independent, while in CE interactions the different options depend on each other in their results, since the success probability of an offer is based on its size. This difference may become more critical when there are multiple possible decisions. Thus, Brenner and Vriend [8] tried to adjust the interdependent arms in the UG by adding the VL mechanism to Gittins' strategy.

In this paper we present the performance of both the basic Gittins' strategy (GS) (as in [3,5,35]) and the deviated virtual extension of GS (DVG). The main difference between DVG and the DVRL is the updating of the Q-values, which are calculated according to Gittins' indices, as detailed in Algorithm 3 (lines 7, 8, 11 and 12).

Algorithm 3 THE DVG ALGORITHM

Notation: N is the upper bound of possible offers. G(x,y) denotes Gittins' index for x successful events and y unsuccessful events. S(j) denotes the corresponding reward for a successful offer j.

```
1:  round=1      For j=0 to N, Do Q(j)=1,X(j)=0
2:  For each interaction, Do
3:      If round=1 then select i uniformly, 0 ≤ i ≤ N          line 3-4: SELECT procedure
4:      Else offer i=arg max_j Q(j)
5:       Observing opponent's move, calculate reward
6:       If offer i has succeeded Then
7:            For j=0 to N, Do
8:                 If j ≥(i - ⌊ i/(round+1) ⌋ )                line 8-15: UPDATE procedure
9:                      X(j)=X(j)+1,    Q(j) = G(X(j), round-X(j))*S(j)
10:                Else
11:                     Q(j) = G(X(j),round-X(j))S(j)
12:      If offer i has failed Then
13:           For j=0 to N, Do
14:                If j<(i + ⌊ (N−i)/(round+1) ⌋ )     Q(j) = G(X(j), round-X(j))*S(j)
15:                Else     X(j)=X(j)+1,    Q(j) = G(X(j), round-X(j))*S(j)
16:      round=round+1
```

The process described in Algorithm 3 is as follows: we save a Q-value (quality value) which contains the Gittins' value multiplied by the reward of the offer if it succeeds. The multiplying function is a heuristic, which takes into consideration the following three arguments for each offer: (a) the probability it will win; (b) the reward in case it wins; and (c) the possible gains from exploring the offer. In each step, we choose an offer i with the highest Q-value (i.e., the highest value of the heuristic function) and observe its result. Then we update the Q-value of offer i using the updated Gittins' value, including this additional event. Note that Algorithm 3 is a heuristic and does not promise optimality, but in our experiments

[9] In our finite simulations we used the Bernoulli reward process table with a discount factor of 0.99 (see [24], p. 237), as in [5,35].

it was found to perform relatively well. We should also emphasize that a theoretical optimal solution for the UG has not yet been developed [8].

5 Experimental evaluation

As mentioned in Sect. 1.3, human competitors do not obey the theoretical equilibrium, and thus their behavior cannot be a priori simulated. Additionally, human behavior cannot be accurately statistically modeled, especially in small populations as in our case.

In the model studied in this paper, an agent participating in the one-shot CE interaction can choose one of a large number of alternatives, and in this paper we set the number of alternatives at $N = 100$. The agent faces several interactions of each type, each time against a new opponent, and it wishes to maximize its total profits from the set of all interactions.

In this section, we describe comparisons of the performance of the proposed reward selecting algorithms given simulations which are based on real data obtained from humans in different CE environments, and we provide a theoretical analysis of the simulation results.

5.1 Experimental description

In order to evaluate the performance of the proposed reward selecting algorithms, we experimentally examined agents' interaction with human opponents in five different CE environments, as follows:

1. UG, where the players have to divide 100 New Israeli Shekels (NIS, where U.S. \$ 1 $\approx$ NIS 3.9).
2. UG variant: a variant of UG in which the responder always receives its own allocation and determines only whether the proposer will receive its share. [27]. In the UG variant, there is no outcome different from the responder's point of view. Thus, while the responder's decision is based mainly on its own outcome in the other games considered, , in the UG variant this parameter plays no role (since it is equal for all options), and the responder's decision is influenced only by fair or vindictive considerations.
3. Auction: a sealed-bid auction with one opponent.
4. All-pay auction: an all-pay sealed-bid auction for NIS 100, where all of the bidders must pay their bids [33],[10] In this version, risk plays an important role since the agent's utility becomes negative if it loses.
5. Dynamic pricing: price setting, where the agent has to decide how much to charge for products which originally cost NIS 100.

We chose these environments, since they are typical environments that artificial agents and humans have to deal with in real world situations, and all of them can also be viewed as "CE environments", as we explained in the introduction. Note that in all our experiments, if a tie exists between the computational agent and any other participant, the system chooses the agent as the winner. The human's thresholds in the first four environments are plotted in Fig. 2.

Dynamic pricing was not experimentally examined since no specific item can be unquestionably desired by all of our participants, and thus it cannot really reflect a market of potential

[10] For simplicity, we considered one opponent in the auction game.'

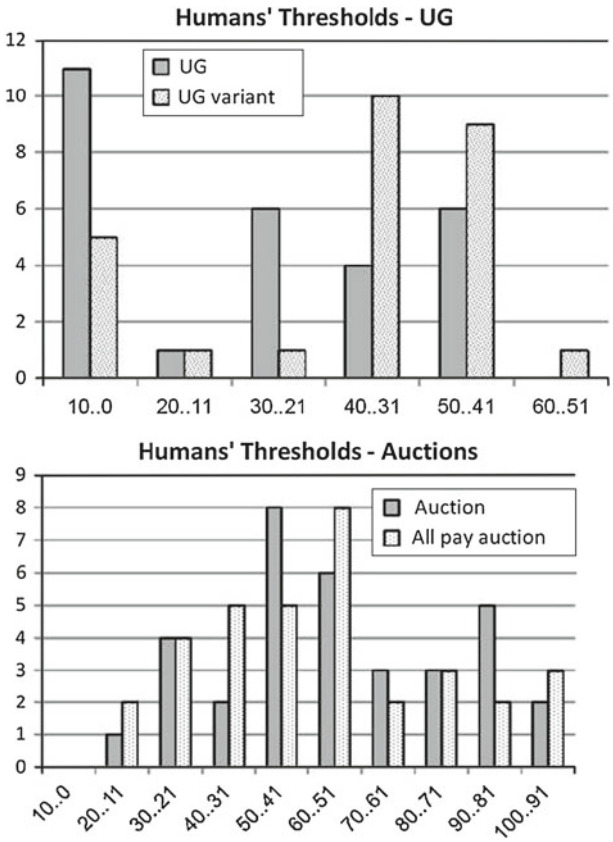

Fig. 2 Humans' threshold distributions

customers.[11] However, we artificially generated a normal distributed population of buyers, as explained in the last paragraph of Sect. 5.2.

The following algorithms were implemented and examined:

1. Bayesian approach, as described in Sect. 3.2.
2. R&E, as described in Sect. 3.5.
3. ZWK, as described in Sect. 3.5.
4. Virtual RL (VRL), as described in Sect. 3.7.
5. Basic Gittins' index strategy (GS), as described in Sect. 4.3.
6. Virtual Gittins' index strategy (VG), as described in Sect. 4.3.
7. DVRL, as described in Sect. 4.2.
8. Deviated virtual extension of GS (DVG), as described in Sect. 4.3.

The first six algorithms were suggested by others and adapted to our environment as detailed in Sect. 3, while the last two algorithms are based on the deviation meta-algorithms that we developed.[12]

[11] This was also the reason for running auctions on a total amount of money rather than on a particular item, i.e., to be able to ignore the influence of different valuations of the item itself.

[12] Another new approach, the segmentation principle, which is described in Sect. 7.3, attained results which improved the basic algorithms but were lower than the results of the deviation principle in all of the non-

5.2 Experimental details

All of the participants in this study were students at Bar-Ilan University, aged 20–28 (average 23.75), who were not experts in either negotiation strategies or in economic theories directly relevant to the experiment (e.g. game theory, decision theory). Each of the participants, who sat in an isolated room at a computer work-station, participated in all four domains of CE interactions, one after the other. The instructions that were given to the participants are provided in Sect. 11.

In the first experiment, 34 human participants (17 males and 17 females) participated as responders in the UG games and as bidders in the auctions. Each person participated once in each of the five environments (UG, UG variant, sealed-bid auction, all-paid sealed-bid auction, dynamic pricing). The automated agents, on the other hand, interacted serially against different human opponents.

In the auctions the participants were required to propose a bid, which was any integer from NIS zero to 100. The winner gained a virtual NIS 100. In the UG, where NIS 100 was the full amount to be shared, the participants were asked for the minimal amount they would accept.

After extracting the bids from people in the auctions, as well as their acceptance thresholds in the UG, we constructed sets of opponents' reactions for each environment. The sizes of the sets were 34 for the auctions, 28 for the UG and 27 for the UG variant.

At this stage we examined the performance of each of the 11 algorithms detailed in Sect. 3 and 4, which were run serially against the sets of opponents' thresholds. 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 the order is known to have a crucial effect on the learning process results, we used the following experimental method: We constructed a representative society of real experimental results, collected from human thresholds, as described above. The society was found to behave similar to known results from the literature, considering the distribution of the participants' thresholds [45]. Then, we constructed 200 random permutations of the human decision series for each environment. We proceeded by running the different algorithms on each of the permutations, and collecting the average payoffs for each algorithm and for each permutation. This method of syntactical simulations based on a real experimental baseline is well documented in the literature [45], and it enabled us to check the algorithms behavior for different ordering of observations from the representative society.

In cases of randomized algorithms, each of the permutations was run several times for different possible behaviors of algorithms on the particular ordering of thresholds, and the average behavior of the algorithm was considered the result for that particular ordering.

More explicitly, for each environment (UG, auction, etc.):

- We collected data based on real human thresholds, for the different environments studied (UG and UG variant, auctions and all-pay auctions, and dynamic pricing).
- We created 200 random permutations of the collected data.
- For each permutation, we ran the different algorithms, and collected the reward obtained by the learning agent for each permutation.
- We ran each randomized algorithm 50 times for each permutation, and for each algorithm we collected the average reward it gained (namely, for each permutation, one number

simultaneous CE domains [29]. However, the segmentation principle reached competitive results for some of the simultaneous CE domains. Thus, it is presented as a solution in the simultaneous CE domains.

was collected, which was evaluated as the average reward of all 50 runs of the algorithm for that permutation).
- The non-parametric Friedman test was chosen in order to compare the results.[13]

Finally, we also created simulated syntactic data in order to compare our results in the auction domain and in the pricing domain. The different algorithms were run against an artificial series of 50 auction opponents, constructed randomly according to a normal distribution N(71, 10).

Similarly, we generated a series of 50 potential buyers for the pricing environment according to a normal distribution N(140, 18).[14] These values represented the maximal price which the buyers agreed to pay for a product that originally cost 100. In this manner, we aspired to examine the performance of the algorithms with a theoretical population that was distributed normally.

5.3 Payoff analysis

Table 2 presents the average payoffs and standard deviation for each reward-selecting algorithm when run 200 times, each time using one permutation of human opponents (given their threshold). Due to the fact that the algorithms' random factors caused a variation in the results, we ran each algorithm repeatedly: 50 times for each permutation.

Given the results of the 50 runs over the set of 200 permutations, we calculated the average payoff and the standard deviation of the performance of each algorithm.

In the calculation, each 50 runs over one permutation were considered to be one point. We use $x_{i,j}$ to denote the results of run number j of the algorithm, given permutation i. Then, X_i denotes the average results of the algorithm over all 50 runs given permutation i, which is calculated as follows: $X_i = \Sigma_{j=1}^{50}(x_{i,j})/50$.

The average $\overline{X}$ of the algorithm results over all permutations is calculated as follows: $\overline{X} = \Sigma_{i=1}^{200} X_i/200$.

Finally, $StdX$ is the standard deviation of the algorithm results over all permutations.

The results show that the DVRL algorithm was almost always the most profitable algorithm. A Friedman test[15] revealed significant differences in the ranking of the algorithms ($p < 0.001$). Further pairwise Wilcoxon tests showed that the most efficient algorithm (with the highest payoff) was clearly the DVRL algorithm ($p < 0.001$), except in the auctions, where DVG was slightly but not significantly better. Moreover, the extension of the DVL meta-algorithms always improved the payoff. Pairwise Wilcoxon tests revealed that the DVRL and DVG algorithms always achieved higher payoffs than the first five algorithms ($p < 0.001$).

Virtual learning was revealed as an efficient method, as can be concluded by the general superiority of VRL over ZWK (except for the UG variant) and the superiority of VG over GS

[13] Since the normal assumption did not hold in our tests, we used a non-parametric Friedman test ([20], pp. 410–412), which was appropriate for cases where the assumptions underlying the analysis of variance were not satisfied.

[14] The distribution of the pricing environment is supposed to have a higher mean than the price suggested in the bidding environment. This is due to the fact that in the pricing environment the buyer has no incentive to decrease its true valuation in the bid, while in the first-price bid, as known from the literature, the bidder is motivated to bid a lower value than its real valuation.

[15] The Friedman test is a non-parametric test for $K \geq 3$ correlated samples.

Table 2 Average payoff and standard deviation of various algorithms against human opponents

Environment		Bayes	R&E	ZWK	VRL	GS	VG	DVRL	DVG
UG	Payoff	32.09	35.55	39	41.3	37.43	37.44	**46.13**	44.36
	SD	1.73	5.56	4.2	4.03	4.02	3.90	1.53	1.88
UG variant	Payoff	29.14	28.93	33.99	32.41	31.88	30.44	**43.44**	40.72
	SD	1.63	5.86	5.01	5.6	5.31	5.96	1.66	1.93
Auction	Payoff	16.54	13.35	15.95	16.18	15.89	16.52	18.08	**19.73**
	SD	0.74	3.21	3.81	3.64	4.12	3.57	1.47	1.39
All-pay auction	Payoff	2.61	−3.27	−1.88	−1.09	−2.14	−1.11	**3.26**	1.96
	SD	1.77	6.07	5.2	5.14	5.22	4.43	2.13	2.9
Dynamic pricing	Payoff	19.01	13.61	16.84	17.6	16.48	17.68	**20.25**	18.9
	SD	0.49	2.68	3.46	2.96	3.9	2.94	0.88	1.27
Normal distribution auction	Payoff	14.74	7.83	11.66	12.39	11.08	12.63	**15.59**	14.45
	SD	0.28	2.12	3.18	2.94	4.16	2.87	0.54	0.86

The bold numbers indicate the highest average payoff achieved for each environment

(except for the UG and for the UG variant). The Bayesian approach under the exponential distribution assumption was almost always the worst, both for the UG and the UG variant.[16]

Following Conitzer and Garera [12], the Bayesian algorithm we used in Table 2 assumed an exponential distribution of the opponents' thresholds. The poor results of this algorithm are due to the fact that the assumption of exponential distribution was not true in any of the CE interactions studied in this paper. A more realistic assumption is that the threshold distribution is normal or close to the normal distribution. Thus, we implemented Algorithm 11 as defined in Appendix A.4 and compared the performance of the DVRL with the performance of the Bayesian approach under the normal distribution assumption for the different CE interactions. Our results are presented in Table 3.

As a result, we can see that these two approaches reach relatively close results. The Bayesian approach is slightly better in most of the environments, while the DVRL is the best for the UG variant for 50 rounds. The Pairwise Wilcoxon test revealed that the advantage of the Bayesian approach is significant in the UG and in the dynamic pricing environment for all the number of rounds considered, and it is also significant for all the environments when taking into account a permutation of all the thresholds (all rounds). The difference was not found to be significant for the auction game, the UG variant, and the all-pay auction for ten rounds, and for the UG variant and the all-pay auction for 20 rounds.

When comparing the DVRL approach with the Bayesian approach under the normal distribution assumption, we should emphasize that the Bayesian inference approach requires much more effort with regards to DVRL. The Bayesian inference, under the normal distribution, needs to handle and update a probability table for each possible set of parameters of the normal distribution, while the DVRL only needs a Q-vector which handles one quality value for each possible decision from a total of N possible decisions. Moreover, in DVRL, we are not obligated to assume any particular distribution of the thresholds, while the Bayesian algorithm requires beliefs about the opponents' thresholds. Thus, DVRL is beneficial when the agent does not know the thresholds' distribution.

[16] In environments where all players have the same threshold, the Bayesian approach attained the best solution. However, such an assumption is not valid in real world CE interactions.

Table 3 Comparison of DVRL and Bayesian approach based on normal distribution

Environment	DVRL 10 rounds	Bayesian 10 rounds	DVRL 20 rounds	Bayesian 20 rounds	DVRL All rounds	Bayesian All rounds
Auction	16.958	17.279	18.023	19.937	18.577	20.454
STD	4.348	4.914	3.443	3.656	2.188	1.332
UG	46.953	51.063	47.691	52.078	48.281	53.076
STD	3.487	6.363	2.405	4.369	1.651	1.963
UG variant	38.922	39.492	40.107	40.223	41.458	41.232
STD	4.026	2.137	3.006	2.982	3.063	2.137
All-pay auction	−0.903	0.927	−0.119	−0.454	1.025	2.315
STD	10.828	3.503	6.317	1.766	1.635	0.897
Dynamic pricing	20.293	30.163	20.284	31.835	20.456*	32.794*
STD	3.487	6.857	3.577	4.789	2.880	3.045
Normal auction	16.961	18.01	17.891	19.119	18.01*	16.961*
STD	5.168	5.612	3.976	3.621	5.612	5.168

* Results of 50 rounds

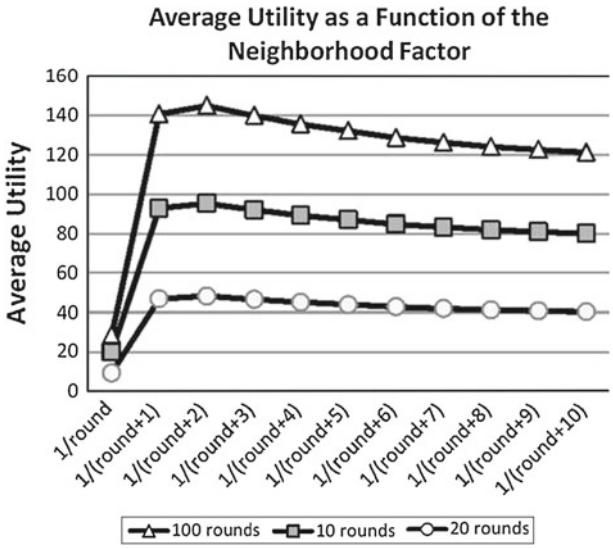

Fig. 3 Average utility as a function of the neighborhood factor

Note that the DVRL could be improved if a tuning phase was added in order to choose the best parameters for it. For example, we can see that different factors of neighbourhood could be used in the UPDATE procedure instead of $\frac{1}{round+1}$.

Figure 3 demonstrates that, for the UG environment, the optimal neighborhood factor was $\frac{1}{round+2}$ for a length of 10, 20 and 100 rounds of interactions (where each opponent was randomly chosen from the set of human thresholds). Similar results were also obtained for the other environments studied in our paper. However, even without choosing the optimal parameter setting, DVRL still performed better than alternative algorithms, as demonstrated in the rest of this paper.

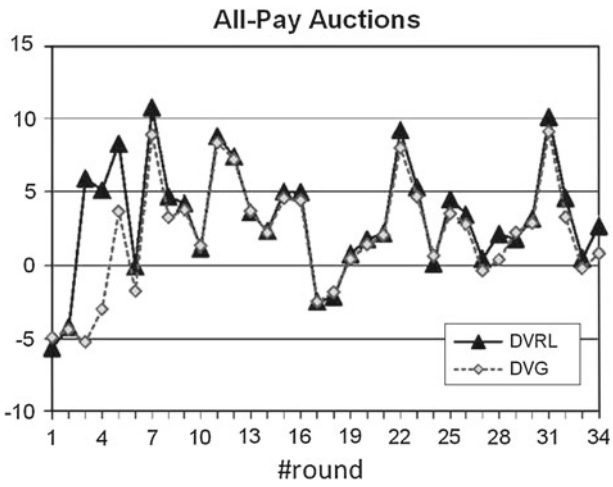

Fig. 4 Average payoff during the all-pay auction

5.4 Discussion of the results

In order to understand the reason for the advantage of DVRL over DVG, we should observe Fig. 4. This plot describes the average payoff of each algorithm—DVRL and DVG—(the y-axis) during the interactions (the x-axis) over all runs. The plot, which enables easier tracking of the learning process, presents the results of the all-pay auction.[17] The left side of the plot shows a noticeable preliminary advantage of the DVRL method. Particularly, the payoff of DVRL was higher than the others until the ninth round, where it became approximately equal to DVG.

This preliminary advantage of DVG can be attributed to the low extent of reinforcement in the RL's Q-values update procedure, compared to the extensive reinforcement conducted by Gittins' indices. For example, the Q-value of an offer which succeeded in two out of three interactions was reinforced in the RL method by 66 % of its reward, while using Gittins' method it was reinforced by 91 %. An offer which was successful only once out of five interactions was reinforced in the RL approach by 20 % of its reward, whereas while using Gittins' method it was reinforced by 47 %. Thus, offers which were extremely far from optimal were filtered more quickly with the RL methods.

A comparison of the two algorithms is shown in Fig. 5 for different sets of environments. We can see in general that in most of the cases, DVRL outperformed the other algorithms considered. However, in the auction domain, the deviated virtual extension of GS (DVG) was superior.

The possible explanation of this behavior is that the auction game has the lowest relative standard deviation of human thresholds (36 %, in comparison to 42, 78 and 50 % in the all-pay auction, the UG and the UG-variant, respectively). The reason for this lowest relative standard deviation in the auction game was the fact that an auction was held on an amount of money, thus all bidders had the same valuation and they knew that their opponents had this valuation as well. In addition, DVRL was found to be superior in situations with large variation, probably due to the structure of the UPDATE process.

[17] The data of the other environments were quite similar as depicted in Fig. 5.

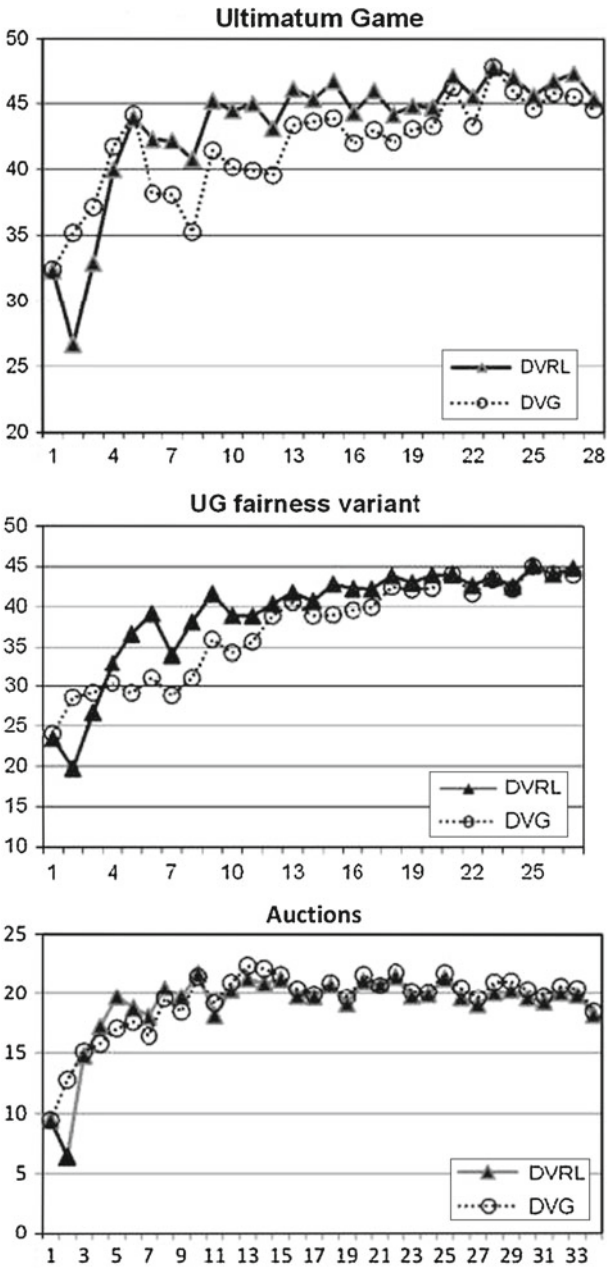

Fig. 5 Average payoff during different types of interactions

Another finding in our experiment was the general improvement caused by virtual learning. This can be explained by the fast updating of a large set of offers after every interaction, with the VL method. Thus offers which were clearly irrelevant were filtered within a few interactions, and it was easy, for example, to find a non-virtual learner that proposes 90 %

in the UG after the 20th interaction. This important advantage offset the disadvantage in accuracy caused by the information asymmetry in the virtual updating process mentioned in Sect. 3.6.

In order to measure the stability and self-consistency of each algorithm, we compared the standard deviations of the payoff results of 50 runs with each permutation of the twelve algorithms.

Observing the standard deviations in Table 2, we can conclude that the standard deviation of the Bayesian approach was always the lowest ($p < 0.001$), apart from the UG where the DVRL was lower, and for the UG variant, where there was no significant difference between these two algorithms. This finding is not surprising, considering the fact that the Bayesian algorithm did not include a random factor. Among all of the other eleven algorithms that were not based on a distribution assumption, the DVRL was always the lowest ($p < 0.001$), excluding the auction, where DVG was slightly lower than DVRL ($p < 0.05$). Since low values of standard deviation guarantee more certain and stable results, using the proposed DVRL algorithm is preferable from this aspect as well. This property may be attributed to the fact that the DVL meta-algorithm does not include a random decision, excluding the first interaction. In addition, the fast screening in the DVRL as well as the lower sensitivity to over-filtering of the optimal solution discussed earlier provide a stable and consistent pattern of searching for the solution. For similar reasons, the relative ranking of the other algorithms in reference to the deviation measures is compatible with the relative ranking in reference to the payoff measures.

5.5 Convergence accuracy

Another factor that we examined was the accuracy of the various algorithms in converging to the optimal solution. Given a series of opponents' reactions, the optimal offer can easily be calculated in retrospect, i.e. which amount, when constantly offered, would yield the best reward.

In our experiments, we examined the average difference between the static optimal offer for the set of the first ten opponents, and the amount that each algorithm offered for this set of opponents. The lowest average difference was found for the DVRL algorithm in all the CE environments, apart from the auction where the DVG achieved a higher average accuracy. This finding is consistent with the payoffs of the algorithms presented in Table 2. We can therefore conclude that the DVRL converged very fast (ten interactions) towards the optimal offer. A detailed calculation revealed that the learning of only ten opponents yielded overall payoffs of about 83 % of the optimal payoffs. We found this amount to be very impressive, especially in light of the fact that the optimal offer could be calculated only after observing the *explicit* series of *all* of the opponents' thresholds, compared to the implicit accept/reject information which was supplied on-line to our algorithms.

6 The simultaneous CE model

The second set of problems addressed in this paper is the problem faced by an agent that participates in K SCE interactions, in which the agent has to decide about more than one offer or bid simultaneously and where the distributions of the opponents are independent across the interactions.

Such situations often occur in the auction domain, where substitutability or complementarity exists between two or more items for sale [6,17,22,38]. This situation is especially

common in web-based auctions, where one can find many simultaneous auctions of similar popular items on different sites [7,51].

Obviously, many bidders have substitute preferences, which means they bid on multiple items in order to obtain only one of them. Other bidders have complementary preferences, which means that they would like to win all of the simultaneous auctions, as in the case of purchasing flight tickets and hotel reservations for a group vacation (where the whole group is supposed to be on the same flight and stay in the same hotel). In addition, 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, or in other cases, from all of them [15,32].

6.1 Formal SCE model

The general pattern of one-shot SCE interactions considers an agent participating in K independent simultaneous interactions, each with a different unknown opponent, where it has to give an offer in each interaction. Each offer for each interaction can be accepted or rejected, and the reward of the agent is a function of the results of all of the K interactions. Formally, the agent has to choose K offers $i_1 \cdots i_K$ which are all integers, $0 \leq i_j \leq N$, $1 \leq j \leq K$, where N is the maximal choice. Then, a positive reward $R(i)$ corresponding to the offers $i_1 \cdots i_K$ is determined, depending on whether the offers passed the acceptance thresholds set by the current opponents.

6.2 Types of SCE interactions

Since the SCE set includes various environments, we detail the models of six environments upon which this paper focuses, whereby other SCE environments can be similarly modeled as well.

- Substitutional auctions (SA): K similar goods, each with a common value of N, are simultaneously auctioned. The agent offers K bids, $i_1 \cdots i_K$ which are all integers, $0 \leq i_j \leq N$, $1 \leq j \leq K$. Given the bids, a reward $R(i)$ is determined according to the highest bids $b_1 \cdots b_K$ of all of the other bidders (one or more) in each of the simultaneous auctions. If $1 \leq j \leq K$ exists such that $b_j \leq i_j$, then $R(i) = N - \sum_{j|b_j \leq i_j} i_j$ (the value of attaining the item, subtracted by the winning bids). Otherwise $R(i)=0$.
- Retractable substitutional auctions (RSA): K similar goods, each with a common value of N, are simultaneously auctioned. Thus the agent can offer K bids, $i_1 \cdots i_K$, which are all integers, $0 \leq i_j \leq N$, $1 \leq j \leq K$, and can retract any of its offers later.[18] Given the bids, a reward $R(i)$ is determined according to the highest bids $b_1 \cdots b_K$ in each of the simultaneous auctions. If $1 \leq j \leq K$ exists such that $b_j \leq i_j$, then $R(i) = N - \min_{j|b_j \leq i_j} i_j$, which is the value of attaining the item subtracted by the minimal bid that won the auction. We only consider the price of the minimal bid since, in a retractable auction, the agent will retract all of the other successful bids in order to minimize its costs. Otherwise $R(i)=0$.
- Perfect complementary auctions (PCA): K similar goods, each with a common value of N are simultaneously auctioned[19]. The agent offers K bids, $i_1 \cdots i_K$ which are all integers, $0 \leq i_j \leq N, 1 \leq j \leq K$. Given the bids, a reward $R(i)$ is determined according to the

[18] 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.

[19] We handle a case where all of the perfect complementary items are equal, which is consistent with the other SCE environments. We can justify the reality of such a scenario using the case of someone who tries to

Table 4 Attributes of different SCE interactions

Environment	Meaning of N	Meaning of K	Winning criteria	Reward
SA	Common value	K auctions	At least one bid wins	$N - \sum_{j\|b_j \le i_j} i_j$
RSA	Common value	K auctions	At least one bid wins	$N - \min_{j\|b_j \le i_j} i_j$
PCA	Common value	K auctions	All bids win	$K \cdot N - \sum_{j=1}^{K} i_j$
OMUG	Amount to divide	$K+1$ players	At least one offer is accepted	$N - \sum_{j\|t_j \le i_j} i_j$
RMUG	Amount to divide	$K+1$ players	At least one offer is accepted	$N - \min_{j\|t_j \le i_j} i_j$
AMUG	Amount to divide	$K+1$ players	All offers are accepted	$N - \sum_{j=1}^{K} i_j$

highest bids $b_1 \cdots b_K$ of all of the other bidders (one or more) in each of the simultaneous auctions. If for **all** $1 \le j \le K$ $b_j \le i_j$, then $R(i) = K \cdot N - \sum_{j=1}^{K} i_j$ (the value of attaining the items subtracted by all of the bids). Otherwise, the agent has to pay for all its winning bids, and in this case, $R(i) = -\sum_{j|b_j \le i_j} i_j$. The attained items are worthless since not all of the K complementary items were obtained and none of the goods obtained in the winning auctions can be consumed without the rest.

- One-accept multi-player UG (OMUG): the agent needs to divide an amount, N, among itself and the other K players. It needs to choose integers $i_1 \cdots i_K$, $\sum_{j=1}^{K} i_j \le N$, which are the amounts proposed to the other players. The reward *R(i)* is determined by the opponents' acceptance thresholds $t_1 \cdots t_K$. If $1 \le j \le K$ exists such that $t_j \le i_j$, then $R(i) = N - \sum_{j|t_j \le i_j} i_j$, otherwise *R(i)=0*. Thus, the fewer the number of accepted offers (but at least 1) and the lower the offers, the higher the reward.
- Retractable OMUG (RMUG): the agent needs to divide the amount N among itself and the other K players. It is required to choose integers $i_1 \cdots i_K$ ($0 \le i_j \le N$, $1 \le j \le K$), which are the amounts proposed to the other players, and later it can retract any offer, even if it was accepted. The reward *R(i)* is determined by the opponents' acceptance thresholds $t_1 \cdots t_K$. If $1 \le j \le K$ exists such that $t_j \le i_j$, $R(i) = N - \min_{j|t_j \le i_j} i_j$, otherwise *R(i)=0*.
- All-accept Multi-player UG (AMUG). the agent needs to divide the amount N among itself and the other K players. It needs to choose integers $i_1 \cdots i_K$, $\sum_{j=1}^{K} i_j \le N$, which are the amounts proposed to the other players. The reward *R(i)* is determined by the opponents' acceptance thresholds $t_1 \cdots t_K$. If for **all** $1 \le j \le K$ $t_j \le i_j$, then $R(i) = N - \sum_{j=1}^{K} i_j$, otherwise *R(i)=0*. Thus, the proposer has to satisfy all of the responders in the game. The lower the offers, the higher the reward.

The three latter environments are multi-player UG versions which reflect situations similar to the former three, respectively. Table 4 summarizes the attributes of the six types of games described above.

6.3 The goal of an agent in the SCE interaction

In the simultaneous CE model, the agent has to decide about K offers simultaneously. Obviously, each SCE interaction contains a trade-off between the reward in the case of success and the probability of success: as the agent chooses an offer with a higher expected reward

simultaneously order several vacation packages for all of his family members. Of course, the whole family wants to go together on the vacation.

Table 5 Utility functions for different SCE interactions

Environment	Utility function $U(i_1, i_2)$	Description
SA, OMUG	$P(i_1)P(i_2)(N - i_1 - i_2) + P(i_1)(1 - P(i_2))(N - i_1) + (1 - P(i_1))P(i_2)(N - i_2)$	The agent wins if at least one offer wins and the agent pays all of its winning offers
RSA, RMUG	$P(i_1)P(i_2)(N - min(i_1, i_2)) + P(i_1)(1 - P(i_2))(N - i_1) + (1 - P(i_1))P(i_2)(N - i_2)$	The agent wins if at least one offer wins and the agent pays only the lowest offer
PCA	$P(i_1)P(i_2)(2 * N - i_1 - i_2 + P(i_1)(1 - P(i_2))(-i_1) + (1 - P(i_1))P(i_2)(-i_2)U(i_1, i_2) = P(i_1)P(i_2)(2 * N - i_1 - i_2)$	The agent wins if all of its offers win and it pays all of its winning offers
AMUG	$P(i_1)P(i_2)(N - i_1 - i_2)$	The agent wins if all of its offers win and it pays them. Otherwise, it does not pay anything

in one of the K simultaneous interactions, the probability that it will win decreases. Thus, the reward-chance trade-off exists in each of the K decisions to be made.

To demonstrate the challenge of a competitor in SCE interactions, let $P(i)$ be the probability that the amount offered, i, will succeed, i.e., if the offer is higher than the other bids in the case of an auction, or is accepted by all/at least one of the responders in the (O/R/A)MUG. In addition, let $U(i_1, i_2 \ldots, i_K)$ be the value of the agent's utility function, given its offers $i_1 \cdots i_K$ for each of the simultaneous interactions $1 \cdots K$. Table 5 presents the agent's utility functions for the basic case of $K = 2$, given the interaction type, and the winning probability for each interaction. Similar utility functions can be built for the general case in which $K > 2$.

In the general case of K simultaneous interactions, the agent has to make K decisions about K offers for each simultaneous interaction, and as a result it is informed whether it has won and what its reward will be. Given the type of interaction, the winning probability of each offer and the utility function structure, an efficient agent will be motivated to choose its offers $i_1, i_2, \ldots i_K$ in order to receive the highest possible value of the utility function U. Note that as in basic CE interactions, a trade-off exists between the agent's reward in case of success and its probability to succeed. However, the decision process of the agent is more complex in the simultaneous case, since its reward and probability of winning depend on all the K offers it gives.

7 Proposed approaches for the simultaneous CE interactions

In this section we present a detailed description of the proposed algorithms for competing in SCE environments given $K = 2$. Algorithm 4 outlines the general approach, given the UPDATE procedure which determines how to update the success probabilities vector, P, after observing the success of the latest action, and the SELECT procedure, which determines how to select the next action (apparently according to both the current expected utility evaluation, and considerations of optimal exploration).

For simplicity of the pseudocode, we present the solution for K=2, which can be easily adjusted to higher K values. The main idea is to maintain a table P for the success probability of each combination of offers i_1, i_2 , and a Table Q, containing the expected utility of each

Algorithm 4 THE GENERAL APPROACH FOR SIMULTANEOUS CE INTERACTIONS

Notation: N is the upper bound of possible offers, P is the probability vector, Q is the quality matrix, which indicates the quality knowledge for each possible pair of offers.

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 **Do**
6: **For** $l_2 = i_1$ to N, **Do** $i_1 \leq i_2 \leq N$
7: Update $Q(l_1, l_2)$ according to the appropriate utility function and the current $P(l_1)$, $P(l_2)$ values.

combination of offers, based on the utility function described in Table 5.[20] Note that the P and Q matrices are symmetric, since no prior information is known about each auction; thus the probability of bidding (i_1, i_2) is equal to the probability of bidding (i_2, i_1). As a result it is sufficient to fill in the matrix only until the diameter.

7.1 Simultaneous DVRL (SDVRL)

We proceed by providing an extension to the deviated virtual reinforcement learning algorithm (DVRL), as previously defined and explained in Sect. 4.2. The simultaneous version of DVRL is called **SDVRL**. The details of the SDVRL are presented in Algorithm 14, Appendix A.7. During the learning process, the DVRL evaluated model comes closer to the real distribution of the opponent's population, and therefore the deviation extents (the size of neighborhood of each proposal) are gradually reduced. This fast exploration process was found to be very efficient in basic CE interactions and in the SCE interactions as well. However, the "inaccurate" updating of the P values according to the deviation principle may cause a wrong valuation of the Q values, which crucially rely on P and $(1 - P)$ values as can be shown in Table 5 above. Therefore, we examined the efficiency of the deviation principle in simultaneous problems by comparing it with more guarded algorithms, as described in the next section.

Another disadvantage of the SDVRL is its high complexity. As can be shown in Appendix A.7, the SDVRL method chooses the optimal offers $i_1, i_2\,(i_1 \leq i_2)$ with regard 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.[21]

Consequently, the SDVRL algorithm is applicable to only a few simultaneous alternatives and a few interactions, which is usually sufficient in common real-world applications. For environments with many interactions we may use heuristics or search algorithms, such as the segmentation-based algorithm described in Sect. 7.3 and the FSP heuristic described in Sect. 7.4.

[20] In the OMUG and AMUG environments line 6 is changed to: For $l_2 = l_1$ to N-l_1, since the sum of l_1, l_2 cannot exceed N.

[21] In the retractable environments, it can be easily proven that the upper offer i_K always needs to maximize $P(i_K)(N - i_K)$, independent of other offers (for $K = 2$, e.g., if $i_1 \leq i_2$, $Q(i_1, i_2) = P(i_1)(N - i_1) + (1 - P(i_1))P(i_2)(N - i_2)$). Thus, for these environments the complexity actually equals $O(I \cdot N^{k-1})$.

7.2 Gittins index based algorithm (SDVG)

In Sect. 4.3 we explained how Gittins' indices strategy [24] can be used for the UG, by considering each optional offer as an arm. We explained that the CE interactions are different from the classical multi-armed bandit problem, since the arms are independent in the classic bandit problems, whereas in CE interactions the success probability of an offer depends on the size of the offer (i.e. the higher the offer, the higher the probability). Thus we suggested a way to extend the basic Gittins method using the deviation principle.

In this section, we suggest how Gittins' strategy can be applied to decide about offers in the simultaneous case. The simultaneous Gittins index based algorithm is presented in Algorithm 15, Appendix A.8. Generally, its structure is similar to that of the DVG algorithm presented in Algorithm 3. Nonetheless, the values learned by Gittins' indices are the P (probability) vector, and the decision about which offer to choose is done according to the Q matrix, which is calculated by the utility function and given the P vector values, as explained in Sect. 7.1. The complexity problem which exists for large values of K exists also in the SDVG algorithm, for reasons similar to those provided for the SDVRL algorithm. Thus for large values of K, again we recommend the heuristics or search algorithms described in Sects. 7.3 and 7.4.

7.3 The segment-based approach

The segment-based approach is a second meta-algorithm, based on the idea of divide-and-conquer. We present this approach here, because though it attained limited results in the case of CE interactions [29], it achieved successful results in the simultaneous interactions model.[22] The idea of the segment-based approach is to divide all of the options into segments and to activate the learning method in the initial interactions on these segments, rather than on specific discrete options.

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. The number of hierarchies, H, is manually configured, and its value extremely affects the algorithm's behavior. Of course, when $H = 1$, no hierarchies exist and the meta-algorithm has no effect. As H increases, an additional level of abstraction is obtained. For all of the hierarchies over the last hierarchy, the agent will choose an alternative from a small set of alternatives, where each option represents a segment. After a segment is chosen, one offer within this segment is chosen, with equal probability for each offer. The segmentation process may help to speed up the decision process, since in the initial steps it has to consider a reduced set of options.

Clearly, as the number of possible options, N, increases, we would like the number of hierarchies to also increase, in order to simplify the agent's decision process. Thus, deciding about H needs to be done with regard to the size of the options set. On the other hand, in setting this parameter, the order of magnitude of the expected number of interactions must be considered, since we need to have enough interactions for all of the hierarchies and also for the H's hierarchy, in order to obtain a good approximation for the result of each individual offer. The segmentation meta-algorithm is described in Algorithm 5.

[22] The segment-based approach enables the agent to focus on groups of offers rather than on the offers themselves. This may be the reason why this approach was outstanding in the simultaneous problems, where the set of alternatives of the agent is extremely large (N^K where K is the number of simultaneous interactions).

Algorithm 5 THE SEGMENT- BASED META- ALGORITHM

Notation: H denotes the number of hierarchies of segmentations, configured manually. S_h is the size of each segment in the h^{th} segmentation hierarchy, Q_h are the Q-value vectors corresponding to the h^{th} segmentation hierarchy and T_h is the serial number of the last interaction of the h^{th} segmentation hierarchy, where we define T_0 as 0. This means that for each hierarchy x, $T_x - T_{x-1}$ interactions are performed.

1: **For** h=1 to H, **Do** **For** j=0 to $\lceil \frac{N}{S_h} \rceil$, **Do** Q_h(j)=1,
2: **For** x=1 to H, **Do**
3: **For** each interaction $(T_{x-1}+1)$ .. T_x **Do**
4: Select segment j from the x^{th} segmentation hierarchy according to a SELECT procedure, using vector Q_x
5: Select offer i uniformly from segment j
6: Observing opponent's move, calculate result
7: **For** h=x to H, **Do** update Q_h according to an UPDATE procedure

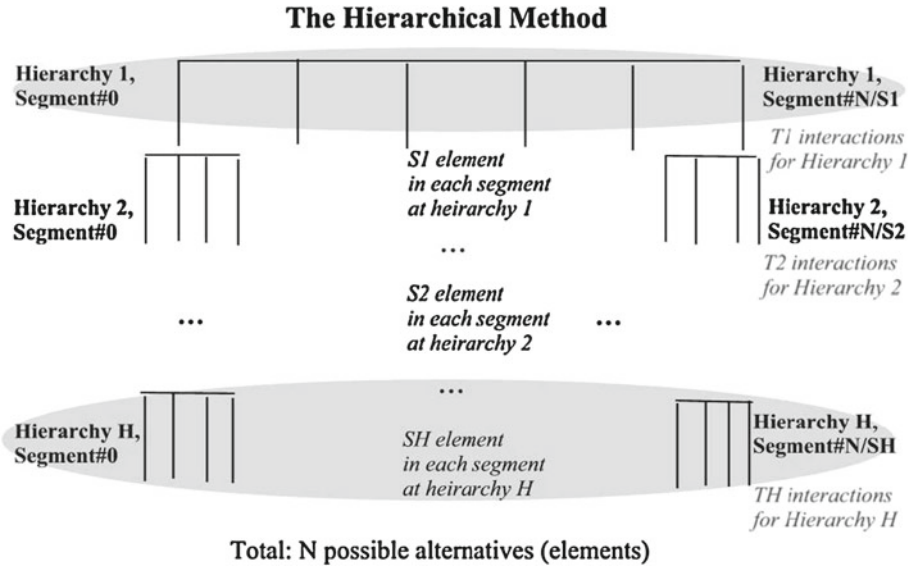

Fig. 6 A diagram of the hierarchies used in the hierarchical method

Figure 6 depicts the segmentation process and the hierarchies used in this method. We can see how the solution space is divided in each hierarchy, where each segment of hierarchy x is divided again into segments in hierarchy $x+1$, until the last segmentation's hierarchy, where in each segment there are S_H elements (for the most part $S_H = 1$). Note that for each hierarchy x the algorithm runs $T_x - T_{x-1}$ interactions, and for each interaction the algorithm updates the Q values of the current hierarchy, as well as the Q values of the hierarchies following it.

This meta-algorithm is mainly useful in environments where the locality attribute holds, i.e. adjacent options yield similar average profits. Clearly, in an environment where neighboring options may have a completely different expected utility, the hierarchical method is not beneficial, since we cannot learn about the behavior of its neighbors from an arbitrary offer. But, in the SCE environments, the probability, as well as the gains, from similar offers for the most part is very similar. Thus, it may be useful to apply the hierarchical method for algorithms when dealing with CE interactions. We can decide to start learning about some

groups of alternatives and then we can continue with higher resolutions and consider smaller offer groups, etc., until we reach the individual decision in the last hierarchy.

Also, in an environment with a small set of possible alternatives, the hierarchical method is redundant, since we can simply start by considering all possible offers. However, in the environments with a large set of possible offers, as well as in continuous spaces of possible offers and thresholds, we would like to have some level of abstraction without losing the accuracy of the solution. In this case, the hierarchical method may be useful, and its relative efficiency depends on the size of the set of possible alternatives. The advantage of this approach is the gradual filtering of the optimal solution. A common problem in the conventional RL, for instance, is that an option might have a high Q-value in a progressive stage of the learning process, although it is far from the optimal option. With the proposed meta-algorithm, this situation is already prevented in the preliminary stages of the learning by weakening the Q-value of the entire range surrounding this non-beneficial solution.

Note that when using the segmentation method, a situation may exist, where the probability of success is usually not 100 %, where optimal segments are sometimes rejected after a few interactions, and are hardly focused on in later stages.[23] This caused the segmentation approach to reach lower results w.r.t. the deviation-based algorithms (DVRL/DVG) in CE interactions. But in the simultaneous case, where the agent has to decide about K offers, and the number of alternatives it has is extremely large, the segmentation method, which enables the agent to focus on groups of offers, becomes very helpful. As a result, the segmentation approach reaches comparative solutions and sometimes even better results than those of the deviation approach. In Algorithm 6 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) [52]. Recall from Sect. 3.5 that, in each interaction in this algorithm we choose the offer which supposedly yields the highest profits, with a probability of $(1-\epsilon)$.[24] In order to explore new options, with a probability of ϵ, we select any offer uniformly. In addition, according to the *virtual* RL principle (without deviation), we increase the success probability evaluation, P-values, of each offer higher than the actual offer, provided it was successful (line 10). Similarly, we reduce the P-values of all offers lower than the actual offer (line 8), provided it was unsuccessful (while with the deviation principle we also increase (reduce) several offers below (above) the actual offer).

Recall also that the idea of the segment-based approach, is to hierarchically divide all options into segments, and in the initial interactions to activate the learning method on these segments, rather than on specific discrete options (lines 12, 13). 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 5).[25] The segment-based approach can also extend Gittins' strategy by modifying the bottom expressions in lines 12 and 15

[23] Extending the duration of the first stages, where a few large segments are observed, would have reduced this problem. However, since we considered environments with a total of 50 interactions at most, the average total payoff would not have risen by extending the duration of the first stages since it would often harm the later interactions, where the fine tuning is managed. The final stage of fine tuning is critically important, especially in the context of CE environments.

[24] The value of ϵ is described in Algorithm 12, Appendix A.5.

[25] In our simulations we used $H = 3$ hierarchies. In the first $T_1 = 5$ interactions we focused on five segments of $S_1 = 20$ integers each. Then, until the tenth 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 ($S_3 = 1$).

Algorithm 6 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, T_h is the serial number of the last interaction of the h^{th} segmentation hierarchy, and we define $T_0 = 0$. P_h and Q_h are the P-vector and the Utility table corresponding to the h^{th} segmentation hierarchy, respectively. ϵ denotes the probability of random selection.

round=1, **For** h=1 to H, **Do** **For** j=0 to $\lceil \frac{N}{S_h} \rceil$, **Do** P_h(j)=1
For x=1 to H, **Do**
  For each interaction $(T_{x-1}+1)\cdots T_x$, **Do**
    If *round=1* **then** select i_1, i_2 uniformly, $0 \le i_1 \le i_2 \le N$
    If $T_{x-1} < round \le T_x$:
      With a probability of (1-ϵ) select segments $j_1, j_2 = \arg\max_{k_1,k_2} Q_x(k_1,k_2)$
      With a probability of ϵ, select segments j_1, j_2 (from the x^{st} hierarchy) uniformly
      Select offers i_1, i_2 uniformly from segments j_1, j_2
    For each offer i_v, $1 \le v \le 2$ **Do**
      If offer i_v has failed **then**
        For h=x to H, **For** $w = 0$ to $\lfloor \frac{j}{S_h} \rfloor$ **Do**
          $P_h(w) = \frac{P_h(w)n(w)}{n(w)+1}$, $n(w) = n(w)+1$
      If offer i_v has succeeded **then**
        For h=x to H, **For** $w = \lfloor \frac{j}{S_h} \rfloor$ to $\lceil \frac{N}{S_h} \rceil$ **Do**
          $P_h(w) = \frac{P_h(w)n(w)+1}{n(w)+1}$,
    $n(w) = n(w)+1$
    round= *round*+1
    For h=x to H, **For** l_1=0 to N,
      For l_2=l_1 to N , **Do**
        Update $Q_h(l_1, l_2)$ according to the appropriate utility function
        and the current values of $P_h(l_1)$, $P_h(l_2)$

according to Gittins' strategy [24]. We call this version the simultaneous segment-based virtual Gittins (SSVG) algorithm.

7.4 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 of the interactions. Note that the chosen offers themselves might be noticeably modified from one interaction to another, due to updating the P-vector. However, the corresponding P-values of the chosen offers are quite stable.

A good demonstration of this phenomenon can be viewed in Fig. 7, where the average amounts of the offers during 34 interactions in the SA environment (left figure) and their corresponding average success probabilities, the P-values (on the right), are plotted. Each point is based on the average of 100 runs on different permutations. We consider the model with $K = 2$ simultaneous auctions, and we observe the suggested bid and the probability to win for the two simultaneous auctions (recall that in the SA, at least one bid must win). We can see that the agent's optimal strategy, is to give one offer which is competitive and has a high probability of winning and another very low offer with a probability near zero, for rare cases where the high bid fails and there are no serious competitors in the other auction.

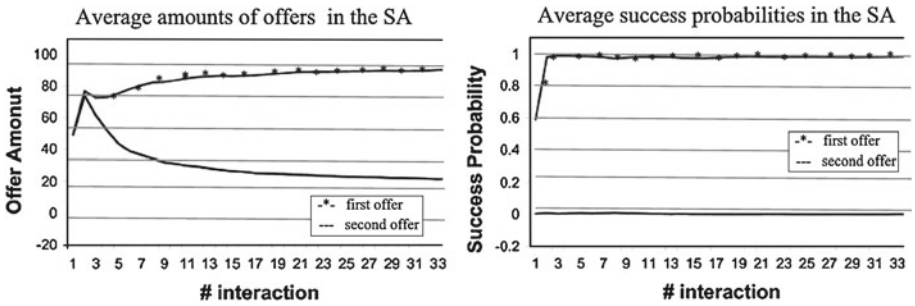

Fig. 7 Amounts of offers (on the *left*) and their corresponding success probabilities (on the *right*) during 34 interactions in the SA environment with K = 2 (average of 100 runs)

An interesting phenomenon that can be seen in the right figure is that 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 of the possible combinations of offers and calculate their utility at each interaction, we can simply choose two offers with current P-values of 0.005 and 0.98. Therefore the FSP algorithm (Algorithm 7) obeys the SDVRL method only in the first interactions (in all of the SCE environments examined we waited five interactions—lines 4, 20, 21— since, according to our preliminary tests, after five interactions there are almost no changes in the P-values). We then (line 22) calculate the P-values c_1, c_2 of the two offers chosen in the fifth interaction (when $\frac{round}{2} = 5$, since *round* is incremented by two at each interaction—Algorithm 2, line 15 in a loop). From then on, the FSP chooses two offers with the same P-values, according to the current P-vector at each interaction (lines 6–9), with a complexity of O(N).

Algorithm 7 THE FSP ALGORITHM

1-3: As in **Algorithm 4**
4: **If** $\frac{round}{2} \leq 5$ **then** offers i_1, i_2=arg max$_{l_1,l_2}$ $Q(l_1, l_2)$
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 $i_2 = j$
10-19: As in **Algorithm 2** lines 5-14
20: **If** $\frac{round}{2}$ <5 **then** **For** l_1=0 to N, **For** l_2=l_1 to N , **Do**
21: Update $Q(l_1, l_2)$ according to the appropriate utility function
· and the current $P(l_1)$, $P(l_2)$ values
22: **If** $\frac{round}{2}$=5 **then** c_1=$P(i_1)$, c_2=$P(i_2)$

It is worth noting that the high complexity of 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 [34,41] and Brent's method [9], as well as discrete methods such as the genetic search [40] and Hill-Climbing [39] methods. However, in this limited framework we mainly discuss methods which concern the uniqueness of SCE interactions. 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.

8 Experimental results for SCE

In this section we examine the performance of different reward selection algorithms competing in various SCE environments with human opponents. A description of the experiment's design is presented, followed by a description of the experimental results.

8.1 Experiment description

In order to evaluate the performance of the different algorithms described in Sect. 7, we experimentally examined agents interacting with human opponents in the six SCE environments mentioned in Sect. 6.2: SA, RSA, OMUG, RMUG, PCA and AMUG. It is important to note that the segmentation-based algorithms, SSVRL and SSVG, were run with a configuration of fixed parameters for all the different environments. These parameters were: the number of hierarchies of segmentations denoted H, the size of each segment h denoted S_h, and the probability of a random selection denoted ϵ. In the first experiment, human participants were used as responders in the OMUG, RMUG and AMUG games and as bidders in the auctions. The characteristics of the human participants were discussed in Sect. 5. Each person participated only once in each environment, while the automated agents interacted serially against different human opponents. In Sect. 5 we explained why agents designed for human-involved environments need to be examined based on real human data.

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 OMUG, RMUG and AMUG (in random order) where $K = 3$. In the two-responder UGs the participants had to determine their acceptance threshold, i.e. the minimal offer they were willing to accept as responders to the total NIS 100 that had to be divided among the three players (without knowing the other responders' offers). In the auctions the participants had to propose a bid, which was any integer from NIS 0 to 100. The winner gained a virtual NIS 100. At the end of the experiment, each participant was paid between 15 and 30 NIS, proportional 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 and the 34 acceptance thresholds in OMUG, in RMUG and in AMUG, we constructed sets of opponents' reactions for each environment. For $k = 2$ we constructed two sets of 34 integers, for $k = 3$, three 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 of the auctions' environments. With this, we simulated a realistic auction environment, where some of the bidders bid simultaneously in several auctions and some of them participated in only one auction.

It is important to note that the opponent in the simultaneous auction was able to participate in only one auction. For example, consider an auction for a vacation package: although our agent competes in several auctions in order to receive solutions for several needs, each of its opponents may need to participate in different auctions, or perhaps also in only one auction each. Thus the data collected about the thresholds of humans in the non-simultaneous auction mode can also be used here as legitimate opponents in cases where the agent participates in multiple auctions simultaneously. Consequently, we were able to use the same set of results of the auction thresholds for the SA, RSA and for the PCA simulations.

At this stage we examined the performance of each of the algorithms detailed in Sect. 7, which were run serially against the sets of opponents' data. Thus, each algorithm had one

Table 6 Average payoff and standard deviation of the various algorithms against human opponents in several SCE environments where $K = 2$

Environment	SSVG	SSVRL	SDVG	SDVRL	FSP
SA	81.08	88.43	90.31	94.59	**95.17**
SD	7.51	6.66	5.34	4.17	4.74
RSA	22.19	24.5	27.06	**27.8**	27.24
SD	4.34	3.71	1.4	1.41	2.27
PCA	20.13	15.36	26.38	**26.58**	26.06
SD	7.51	6.66	5.34	4.17	4.74
OMUG	47.49	**50.99**	46.79	49.69	49.6
SD	3.15	3.08	4.5	3.8	3.94
RMUG	58.14	62.35	63.53	**65.15**	63.23
SD	4.53	3.44	3.13	2.13	2.26
AMUG	14.33	16.01	**20.18**	18.47	18.47
SD	3.15	3.08	4.5	3.8	3.94
Substitutional normal	86.8	89.96	97.24	100.96	**101.19**
STD	5.8	5.83	3.87	2.71	2.69
Complementary normal	22.80	28.47	33.29	**36.07**	34.21
STD	9.44	6.78	4.01	3.26	4.277

The bold numbers indicate the highest average payoff achieved for each environment

interaction with each of the human opponents in each of the six environments, without knowing in advance the number of interactions.

Since the order of the opponents is important, we constructed 100 random permutations of the human decision series for each environment and compared the average payoffs of the different algorithms for each permutation. In addition, these algorithms were run (with substitute, retractable and complementary preferences) 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 with normal distribution.

8.2 Algorithm comparison

Table 6 presents the average payoffs for each algorithm, competing in the environments mentioned in Sect. 6.2, with $K = 2$. The average payoffs were calculated based on the data of all 100 permutations. Due to the fact that the algorithms' random factors cause a variation in the results, we ran each algorithm repeatedly, 30 times for each permutation.

It can be shown that SDVRL was the best algorithm in more than 50 percent of the cases, but in some of the environments other algorithms performed better. This result is different than the result of the non-simultaneous case, shown in Table 1 in Sect. 5, where DVRL was the winner in all but one of the environments. However, the simultaneous DVRL algorithm was still the most profitable algorithm (among the four basic algorithms) in most of the cases. A non-parametric Friedman test revealed significant differences in the ranking of the algorithms' payoffs ($p < .001$). Pairwise Wilcoxon tests revealed that SDVRL was the most efficient algorithm (had the highest payoffs) in most of the domains. But, in the PCA, no significant differences were found between SDVRL and SDVG, and the OMUG and the AMUG. SSVRL and SDVG (respectively) were the most efficient algorithms (although the differences were not substantial, compared to the outstanding advantage of SDVRL in the other environments). The FSP heuristic was found to be very efficient as well, and

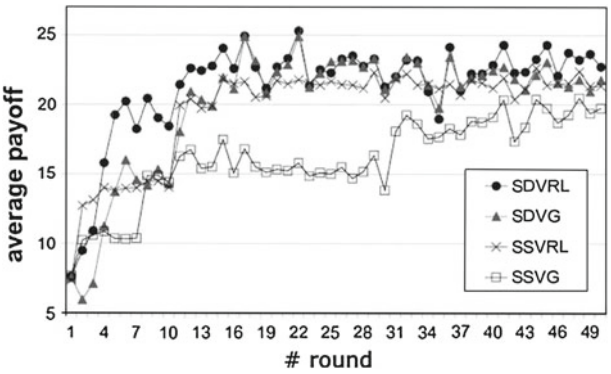

Fig. 8 Average payoff during retractable normal distribution auction interactions

its performance was not significantly different from the basic SDVRL's in the SA, PCA and in the substitutional normal distribution auction. Similar results were found also for environments with $K = 3$, i.e., three-responder MUG and RMUG, and three simultaneous auctions.[26]

8.3 Result explanations

Observing Fig. 8 may assist in understanding the origin of the general advantage of SDVRL over the other algorithms (except for FSP which is a heuristic method based on SDVRL). This figure depicts the average payoff (the y-axis) of each algorithm—SSVRL, SDVRL, SDVG and SSVG—during the interactions (the x-axis) in the retractable normal distribution auction environment (with $K = 2$). The dynamics in this environment were quite similar to the dynamics in other environments. The plot shows a noticeable preliminary advantage of the RL-based methods as compared to the Gittins-based methods. Particularly, the payoff of SDVRL was higher than SDVG's until the 17th round and the payoff of SSVRL was higher than SSVG's until the end (excluding rounds 8–10). This advantage can be attributed to the low extent of reinforcement in the RL's Q-values update procedure, compared to the extensive reinforcement conducted by Gittins' indices, as explained in Sect. 5.4. This difference affects the deviation-principle methods (SDVRL, SDVG) mainly in the preliminary stages. In the segment-based (SSVRL, SSVG) methods it is much more crucial since, as explained below, according to the segment-based principle, in the later stages we focus mainly on segments that were focused on in the preliminary stages.

Figure 8 clearly demonstrates the preliminary advantage of the segment-based (SSVRL, SSVG) methods compared to the Deviation-principle methods (SDVRL, SDVG, respectively) during the first three interactions. This advantage can be attributed to the foolhardy exploration process inspired by the deviation principle in the first stages. The conservative segment-based methods, on the other hand, focus mainly on "safe" segments, and therefore achieve higher payoffs. However, in more progressive stages the performance of the deviation-based methods surpassed their respective segment-based methods, thanks to their rapid and daring exploration. In addition, at times segment-based methods may not converge towards the optimal solution. This is because the success probability of the optimal segment is usually less than one, thus it may be rejected after a few interactions and then hardly focused on in later stages.

[26] Detailed results for these environments can be found in [30].

In the $K = 3$ simulations, however, the complexity of all of the parameters was much higher, the agent's decision was much more complicated and therefore the estimation of the P-values had to be much more precise. However, the foolhardy exploration process inspired by the deviation principle in the first stages may cause inaccurate results. In such cases where accuracy is important the problem of inaccurate results may cause the deviation-based methods to lose some of their advantage.

As a result, we can observe that there were cases where SSVG was better than SDVG, and in one case (OMUG), the SSVRL was also better than the SDVRL. In fact, Gittins' indices strategy suffers more from this problem since, in general, the Gittins' indices were found to be less accurate than the RL in our environments.

The results of the FSP heuristic show that its specific success depends on the opponents' distribution in each environment. If, as in the SA, there are only a few exceptional users, the initial exploration during the first five interactions is sufficient and the P-values are then approximately optimal. Moreover, the appearance of exceptional users in later interactions may cause an undesirable effect when using the SDVRL, but will cause no damage when using the FSP after the first five interactions (since the P values are already determined). This is the reason that FSP performed better than the basic SDVRL in a few cases.

9 Conclusion and future work

In this paper, we consider Cliff Edge (CE) interactions: a general group of real world situations in which an agent needs to make a decision about a certain offer or bid, where the probability of its success decreases monotonically as the expected reward increases.

We consider environments in which an agent participates repeatedly in one-shot interactions, each time against a new opponent, but all of the opponents come from the same population. Thus, the agent may learn and improve its knowledge about the general pattern of the population's behavior every time it interacts with a new opponent. In particular, it can learn the probability that each offer or bid will succeed, according to the information collected from previous interactions.

We present a new meta-algorithm, namely DVL, which extends existing methods for efficient competition in CE interactions against different human opponents. The DVRL is an algorithm which is easy to implement, and it is designated for various domains and can be beneficial for both further academic research and real economic applications.

Our experimental findings show that the DVL extension of an RL basic algorithm (DVRL) has results which are competitive with the results of the Bayesian algorithm based on Gaussian assumption, and it performs significantly better than the other algorithms surveyed in this paper.

We consider the SCE problem, which includes simultaneous auctions for substitute or complementary goods. We present a new algorithm, namely SDVRL, which efficiently competes in various multi-interaction SCE environments against different human opponents. To handle the size of the problem, we present the segmentation meta-algorithm, where the options' space is divided into segments. The segmentation meta-algorithm was combined in different learning algorithms and reached successful results. Finally, an optimization heuristic called FSP was found to demonstrate high performance, as well.

This study principally concerns adaptable agents, which compete in no more than several dozen repeated interactions. However, the DVRL algorithm does not necessarily perform well in environments which last for hundreds of interactions. This is mainly due to the fact that the directed search for an optimal option partially neglects the exploration of other options.

Thus, in future work we intend to check and improve its behavior in the long run, in order to suggest efficient and competitive approaches for interacting in long duration CE interactions.

In addition, we intend to consider the repeated version of CE interactions, in which several negotiation rounds can be conducted against the same opponent [4]. In such settings, a specific modeling of the current opponent is needed in addition to the generic modeling of the population. Moreover, in contrast to the one-shot version, a decision made in the present may influence the opponent's future behavior, and this fact 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.

Another interesting extension is the domain of continuous action spaces, where the threshold of each participant, as well as the agent's offers, can be any real number. To date, our learning algorithms were designed and examined in domains with a large but discrete number of options. If we consider domains with continuous action spaces, the algorithms suggested in this paper can be used by partitioning the continuous action space into buckets, where each bucket will represent a small part of the total range and each bucket will be represented by its own index and Q values. If a high level of accuracy is required, the segment-based approach, described in Sect. 7.3, can be used both for the CE interactions model and for the SCE interactions model. In future work, we intend to compare the different algorithms for the case of continuous action spaces.

Acknowledgments We thank the anonymous reviewers for their helpful remarks which enabled us to significantly improve the quality of this paper. This work is supported in part by the following Grants: ERC Grant # 267523, MURI Grant Number W911NF-08-1-0144, ARO Grants W911NF0910206 and W911NF1110344, MOST # 3-6797 and the Google Inter-university Center for Electronic Markets and Auctions.

10 Appendix The details of the proposed algorithms

10.1 A.1 A generalized meta-algorithm for CE interactions

In this section we present a detailed description of the proposed meta-algorithm for competing in CE interactions. As a meta-algorithm, it extends basic algorithms and improves their performance. Before each interaction, the algorithm chooses the best offer to suggest and, after the reward is obtained, the results may change future decisions. The current *round* of the algorithm is the number of the current interaction, e.g., the first interaction is at $round = 1$, the second interaction is at $round = 2$, etc.

We assume that the basic algorithms select their actions according to an evaluation of the expected utility that each decision option would yield, provided it is chosen. The evaluation of the expected utilities is determined according to the results of previous interactions. Hereinafter, we will refer to the database which stores the evaluations of each decision option as the Q-vector. For each offer i, the Q-vectors specify the quality value of choosing this offer, as learned from previous interactions.[27] The general meta-algorithm framework is described in Algorithm 8.

Each algorithm includes its own UPDATE procedure, which determines how to update the Q-vector after observing the success of the latest action. In addition, each algorithm includes

[27] In the CE domain this quality value is one-dimensional, since we assume that each interaction with an opponent is unique and does not influence future interactions. Thus, only one state exists in the CE game, in which the agent needs to suggest its offer to the new unknown opponent, and the Q-vector indicates the quality knowledge for each possible offer in this one state.

Algorithm 8 THE GENERAL META- ALGORITHM

Notation: N is the upper bound of possible offers.

round=1
For j=0 to N, **Do** Q(j)=1
For each interaction, **Do**
  If *round=1* **then** select i uniformly, $0 \leq i \leq N$
  Else select offer i according to a SELECT procedure
  Observing opponent's move, calculate reward
  If offer i has succeeded **Then**
    For j=0 to N, **Do**
      Update Q(j) according to the appropriate algorithm and the UPDATE procedure
  round=round+1

its own SELECT procedure, which determines how to select the next action (apparently according to the current Q-vector). The SELECT procedure moves between exploiting the evaluations stored in the current Q-vector and exploring options with lower current Q-values in order to improve the accuracy of the Q-vector's evaluations. The first action is chosen randomly in all of the algorithms presented in this paper.

10.2 A.2 RL: Roth and Erev

The basic Roth and Erev algorithm is described in Algorithm 9.

Algorithm 9 THE ROTH AND EREV RL ALGORITHM

Notation: N is the upper bound of possible offers. ϵ is the adjacency experimental parameter.

round=1
For j=0 to N, **Do** Q(j)=1
For each interaction, **Do**
  If *round=1* **then**
    select i uniformly, $0 \leq i \leq N$
  Else Select offer i,
    where the probability to choose each offer k is equal to $Q(k)/\sum_{l=1}^{N} Q(l)$.
  Observing opponent's move, calculate reward r
  If offer i has succeeded **Then**
    $Q(i) = Q(i) + (1-\epsilon)S(i)$
    Add the quantity $\epsilon S(i)$ to the strategies adjacent to i
  round=round+1

10.3 A.3 The Bayesian approach

According to the Bayesian approach, suggested by Conitzer and Garera [12], in each round, the principal chooses the reward that maximizes her expected utility, given her beliefs about the agents' distributions; then the principal updates these beliefs based on the observed result, using Bayes' rule. Accordingly, if the principal assumes that the type of distribution is exponential with parameter λ, it learns the parameter's value during the interactions.

Following [12], the agent considers some discrete values of the underlying distribution. For example, when considering the exponential distribution with parameter λ, the agent assumes that λ must be any member of $\{0.00, 0.01, \ldots 10.00\}$. The agent has prior beliefs about the probability of each distribution parameter and it updates its beliefs according to the results it obtains.

Note also that in our work, the agent only knows if its offer was better than the threshold of the responder, but it does not know what the responder's threshold was. Thus, the probability of each offer is updated given a Boolean result of accept or reject the agent's offer, using Bayes' rule.

The general Bayesian algorithm is presented in Algorithm 10.

Algorithm 10 THE BAYESIAN ALGORITHM

Notation: N is the upper bound of possible offers.
U indicates the list of distribution parameters. For example, given the exponential distribution, $U = (\lambda)$, and given the normal distribution, $U = (\mu, \sigma)$.
$f[U]$ represents: for each list of parameters $u \in U$, the probability of u to be the correct distribution parameter according to the agent's beliefs.
$S(i)$ is the reward of offer i if it wins, $F(i)$ is the reward of offer i if it loses.
$fsucc(i|u)$ is the probability of offer i to succeed given distribution parameter $u \in U$.

1: Initialize $f[u]$ according to the prior distribution: For each value of $u \in U$, $f[u] = 1/|U|$ where $|U|$ is the size of the set U of all the values of parameters' set which are considered by the agent.
2: *round=1*
3: **For** each interaction, **Do**
4: **If** (round=1) **Then** select offer i uniformly at random
5: **Else** select offer i which maximizes the expected reward
given the distributions vector $f[U]$:
$i = argmax_{j=1\cdots N}\{\Sigma_{u\in U} f[u] * (fsucc(j|u) * S(j) + (1 - fsucc(j|u)) * F(j))\}$
6: Observing opponent's move, calculate reward
7: **If** offer i has succeeded **Then**
8: Update vector $f[u]$ by Bayes' rule, given the success of offer **i**:
$$f[u] = \frac{fsucc(i|u) * f[u]}{\Sigma_{w\in U} fsucc(i|w) * f[w]}.$$
9: **Else**
10: Update vector $f[u]$ by Bayes' rule, given the failure of offer **i**:
$$f[u] = \frac{(1 - fsucc(i|u)) * f[u]}{\Sigma_{w\in U}(1 - fsucc(i|w)) * f[w]}.$$
11: *round=round+1*

10.4 A.4 Normal distribution Bayesian algorithm

We present the details of the Bayesian approach algorithm when assuming normal distribution of the opponent's thresholds in Algorithm 11. Similarly to the method used by Conitzer and Garera [12] for other distributions (uniform and exponential), the algorithm considers several possible values of the normal distribution parameters; μ (the average) and σ (the standard deviation).

The decision about the offer is made in lines 6–10: In order to decide which offer to give, in any step except the first step, the algorithm calculates the expected reward of each offer, for each pair of distribution parameters.

Observing the result (acceptance or rejection) of the offer, the algorithm updates the probability of each pair of distribution parameters of the threshold. The prior probabilities

are defined in lines 2, 3, and are updated given each result (acceptance or rejection) of an offer, using Bayes' rule, in lines 12–21.

Note that since the distribution is characterized by a pair of parameters, the probability data is managed by matrix $f[\mu, \sigma]$. The possible values of the parameters considered are $\mu = 0, 1 \ldots 100$ and $\sigma = 1, 2 \ldots 100$. Each such pair represents a specific distribution of thresholds, and given the result of the agent's offer, we can calculate the posterior probability for each pair of distribution parameters μ and σ, using Bayes' rule. Note that we consider discrete values for μ, σ, but given each pair of μ, σ, the value of the threshold is taken from the continuous normal distribution.

Algorithm 11 THE NORMAL- BASED BAYESIAN ALGORITHM

!h **Notation:** N is the upper bound of possible offers,
$S(i)$ is the reward of offer i if it wins, $F(i)$ is the reward of offer i if it loses.
$NormF(threshold|\mu, \sigma)$ is the cumulative distribution function (CDF): the probability that $x \leq threshold$, when x is a normal variable with mean μ and standard deviation σ.
$fsucc(i|\mu, \sigma)$ is the probability of offer i to succeed if the opponent threshold is drawn from distribution $N(\mu, \sigma)$. For example, in the UG and in the auction game, $fsucc(i|\mu, \sigma) = NormF(i|\mu, \sigma)$.
$f[\mu, \sigma]$ is the probability that the distribution of the opponents' threshold is drawn from the normal distribution with parameters μ and σ.

1: Initialize distribution matrix $f[\mu, \sigma]$:
2: **For** each $\mu = 0, 1, \ldots N, \sigma = 1, 2 \ldots N$ do
3: $f[\mu, \sigma] = 1/((N + 1) * N)$
4: *round=1*
5: **For** each interaction, **Do**
6: **If** (round=1) **Then** select offer i uniformly at random
7: **Else** select offer i which maximizes the expected reward given $f[\mu, \sigma]$:
8: For each offer j, calculate the expected utility $Util_j$, as follows:
9: $Util_j = \sum_{\mu=0,1\ldots N,\sigma=1,2\ldots N} f[\mu, \sigma] * (fsucc(j|\mu, \sigma) * S(i) + (1 - fsucc(j|\mu, \sigma)) * F(i))$
10: Choose offer i which maximizes $Util_i$.
11: Observing opponent's move, calculate reward
12: **If** offer i has succeeded **Then**
13: Update vector $f[\mu, \sigma]$ by Bayes' rule, given the success of offer **i**:
14: $SProbSucc = \sum_{\mu=0,1\ldots N,\sigma=1,2\ldots N} fsucc(i|\mu, \sigma) * f[\mu, \sigma]$
15: **For each** $\mu = 0, 1 \ldots N, \sigma = 1, 2 \ldots N$
16: $f[\mu, \sigma] = (fsucc(i|\mu, \sigma)) * f[\mu, \sigma]/SProbSucc$
17: **Else**
18: Update vector $f[\mu, \sigma]$ by Bayes' rule, given the failure of offer **i**:
19: $SProbFail = \sum_{\mu=0,1\ldots N,\sigma=1,2\ldots N} (1 - fsucc(i|\mu, \sigma)) * f[\mu, \sigma]$
20: **For each** $\mu = 0, 1 \ldots N, \sigma = 1, 2 \ldots N$
21: $f[\mu, \sigma] = (1 - fsucc(i|\mu, \sigma)) * f[\mu, \sigma]/SProbFail$
22: *round=round+1*

The underlying assumption behind Algorithm 11 lies in the fact that the opponents' thresholds are normally distributed. Even if this assumption does not hold, the algorithm may be a heuristic for any case of threshold distribution which is close to the normal distribution.[28]

For each possible mean μ and standard deviation σ, the algorithm maintains the probability of the normal distribution to have this mean and standard deviation, respectively. We start by

[28] We assume that the mean of the thresholds is a discrete value between $0, 1 \ldots N$, and the standard deviation is a discrete value between one and N. However, the value of each given threshold is drawn from the continuous normal distribution, where the pair of parameters of the normal distribution is discrete.

giving equal probability to each such pair, and after each step of the algorithm we update the probability of each possible pair using Bayes' rule.

Whenever the agent should choose a new offer i, it calculates the expected utility for each i, based on the probability of each pair of μ and σ and based on the probability of offer i to win, given each particular pair. Then, it chooses the offer with the highest expected utility.

10.5 A.5 A modified version of Zhong, Wu and Kimbrough's (ZWK) RL algorithm

Algorithm 12 describes the ZWK algorithm, suggested by Zhong et al [52], which was modified as described in Sect. 3.5.

Algorithm 12 THE ZWK ALGORITHM

!b **Notation:** ϵ denotes the probability of random selection, γ is the probability of selecting an offer which is adjacent to the offer with the current maximal Q-value ($\epsilon + \gamma < 1$) and δ determines the adjacency range ($\delta \ll$ N).

1: **For** j=0 to N, **Do** Q(j)=1, n(j)=0
2: **For** each interaction, **Do**
3: m = arg max$_j$ $Q(j)$
4: With a probability of (1- ϵ- γ), offer i = m
With a probability of γ, select offer i uniformly, for i's such that
max(0, m - δ) $\leq$ i $\leq$ min(N, m + δ)
With a probability of ϵ, select offer i uniformly, for i's such that $0 \leq i \leq N$
5: Observing opponent's move, calculate reward r
6: n(i)=n(i)+1, Q(i) = $\frac{Q(i)(n(i)-1)+r}{n(i)}$

The values ϵ and γ can be gradually decreased during the learning process. In our experiments, we used the following parameters: $\epsilon = \frac{10}{round+25}$, $\gamma = \frac{15}{round+25}$ and $\delta = 1$. Clearly, as $round$ increases, the probability of choosing an offer different than m decreases. In other words, as the agent becomes more experienced, it tends to choose the best known offer and to reduce the probability of exploration.

10.6 A.6 Todd and Borger's virtual RL algorithm (VRL)

In Algorithm 13 we present an implementation of the virtual learning principle described in Sect. 3.7.

Algorithm 13 THE VRL ALGORITHM

Notation: $S(j)$ denotes the corresponding reward for a successful offer j. $F(j)$ is the corresponding reward for offer j when it fails.
1-5: As in **Algorithm 12**
6: **if** offer i has failed **then**
7: **For** j=0 to i, **Do** n(j)=n(j)+1, $Q(j) = \frac{Q(j)(n(j)-1)+F(j)}{n(j)}$
8: **else**
9: **For** j=i to N, **Do** n(j)=n(j)+1, $Q(j) = \frac{Q(j)(n(j)-1)+S(j)}{n(j)}$

Among the possible implementations, we consider the version that yields the best performance in our environment. This algorithm includes the same SELECT procedure as in ZWK.

10.7 A.7 The SDVRL algorithm

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 (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 9. 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 9, 10, 13, 14).

Algorithm 14 THE SDVRL ALGORITHM

Notation: N is the upper bound of possible offers.

1: *round*=1, **For** j=0 to N, **Do** P(j)=1
2: **For** each interaction, **Do**
3: **If** *round=1* **then** select i_1, i_2 uniformly, $0 \leq i_1 \leq i_2 \leq N$
4: **Else** offers i_1, i_2=arg max$_{l_1,l_2}$ $Q(l_1, l_2)$
5: Observe results of the 2 offers, calculate reward
6: **For** each $1 \leq v \leq 2$, **Do**
7: **If** offer i_v has succeeded **then**
8: **For** j=0 to N, **Do**
9: **If** j $\geq(i_v$ - $\lfloor \frac{i_v}{round+1} \rfloor$) P(j)= $\frac{P(j)(round-1)+1}{round}$
10: **Else** P(j)= $\frac{P(j)(round-1)}{round}$
11: **If** offer i_v has failed **then**
12: **For** j=0 to N, **Do**
13: **If** j $<(i_v$ + $\lfloor \frac{N-i_v}{round+1} \rfloor$) P(j)= $\frac{(P(j)(round-1)}{round}$
14: **Else** P(j)= $\frac{P(j)(round-1)+1}{round}$
15: *round=round*+1
16: **For** l_1=0 to N, **For** l_2=l_1 to N , **Do**
17: Update $Q(l_1, l_2)$ according to the appropriate utility function and the current $P(l_1)$, $P(l_2)$ values

10.8 A.8 The SDVG algorithm

In this section, we describe the simultaneous Gittins index based algorithm (SDVG), which was introduced in Sect. 7.2. In SDVG, updating of the P-values, is calculated according to Gittins' indices. Therefore, the P-values are changed to:

$$P(j) = G(S(j), round - S(j))$$

The details of the SDVG method are described in Algorithm 15.

11 B The instructions given to human subjects

Four different scenarios are presented below for which you should make your decisions. Please write down your own decisions in the appropriate locations, without consulting others.

1. You have NIS 100. You are participating as a bidder in an auction against another bidder. The amount of money on which you are bidding is $100. The one who places the highest

Algorithm 15 THE SDVG ALGORITHM

Notation: N is the upper bound of possible offers. $G(x, y)$ denotes the Gittins' index for x successful events and y unsuccessful events. $S(j)$ denotes the corresponding reward for a successful offer j.

```
1: round=1,
2: For i=0 to N, Do P(i)=1
3: For each interaction, Do
4:     If round=1 then
5:         select i_1, i_2 uniformly, 0 ≤ i_1 ≤ i_2 ≤ N
6:     Else
7:         offer i_1, i_2=arg max_{l_1,l_2} Q(l_1, l_2)
8:     Observe results of the 2 offers, calculate reward
9:     For each 1 ≤ v ≤ 2, Do
10:        If offer i_v has succeeded then
11:            For j=0 to N, Do
12:                If j ≥(i_v - ⌊i_v/(round+1)⌋ ) S(j)=S(j)+1, P(j) = G(S(j),round - S(j))
13:                Else P(j) = G(S(j),round - S(j))
14:        If offer i_v has failed then
15:            For j=0 to N, Do
16:                If j <(i_v + ⌊(N−i_v)/(round+1)⌋ ) S(j)=S(j)+1,P(j) = G(S(j),round - S(j))
17:                Else P(j) = G(S(j),round - S(j))
18:        round=round+1
19:    For l_1=0 to N,
20:        For l_2=l_1 to N , Do
21:            Update Q(l_1, l_2) according to the appropriate utility function
22:            and the current P(l_1), P(l_2) values
```

bid gains the $100 but has to pay back the amount of his bid. For example, if you offer amount X and the other bidder offers only X-5, you will win 100-X, while the other one will receive nothing.
How much will you offer?

2. This scenario is quite similar to the former, except that now the loser (the bidder with the lowest offer) also has to pay his offer, even though he does not win the money.
For example, if you offer amount X and the other bidder offers only X-5, you will win 100-X, while the other bidder will lose X-5. How much will you offer?
3. You are sitting in a room with an unfamiliar person. You will both be given the sum total of $100 if you can agree on how to divide the money. The other person should propose how to divide the money - how much goes to him and how much goes to you.
If you accept the proposal, each of you will receive the amounts specified in the proposal. If you reject the proposal, neither of you will receive anything. You will not meet this person ever again.
What is the minimal offer that you will accept?
4. This scenario is quite similar to the former, except that now you will always receive the amount proposed to you, whether you accept the offer or not. However, you can decide whether the other person will receive the amount he wanted for himself or not.
Therefore, if you accept the offer, each of you will receive the amount specified in the proposal. If you reject the offer, you will receive the amount specified in the proposal, while the proposer will not receive anything.
You will not meet this person ever again.
What is the minimal offer that you will accept?

References

1. Aggarwal, G., Goel, A., & Motwani, R. (2006). Truthful auctions for pricing search keywords. In *Proceedings of the 7th ACM Conference on Electronic Commerce 06*.
2. Auer, P., Cesa-Bianchi, N., & Fischer, P. (2002). Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, *47*(2–3), 235–256.
3. Azoulay-Schwartz, R., Kraus, S., & Wilkenfeld, J. (2004). Exploitation vs. exploration: Choosing a supplier in an environment of incomplete information. *Decision Support Systems and Electronic Commerce, 38*(1), 1–18.
4. Binmore, K., Shaked, A., & Sutton, J. (1985). Testing noncooperative bargaining theory: A preliminary study. *The American Economic Review*, *75*(5), 1178–1180.
5. Bourgine, P., & Leloup, B. (2000). May learning explain the ultimatum game paradox? In *Ecole Polytechnique*. GRID Working Paper No. 00–03.
6. Boutilier, C., Goldszmidt M., & Sabata, B. (1999). Sequential auctions for the allocation of resources with complementarities. In *Proceedings of the International Joint Conferences on Artificial Intelligence (IJCAI) 99*.
7. Boyan, J., Greenwald, A., Kirby, R., & Reiter, J. (2001). Bidding algorithms for simultaneous auctions. In *IJCAI Workshop on Economic Agents, Models, and Mechanisms* (pp. 1–11).
8. Brenner, T., & Vriend, N. (2006). On the behavior of proposers in ultimatum games. *Journal of Economic Behavior and Organization*, *61*(4), 617–631.
9. Brent, R. (1973). *Algorithms for Minimization without derivatives* (Chap. 4). Englewood Cliffs, NJ: Prentice-Hall.
10. Byde, A., Preist, C., & Jennings, N. R. (2002). Decision procedures for multiple auctions. *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, *02*, 613–620.
11. Camerer, C. F. (2003). *Behavioral Game Theory*. Princeton: Princeton University Press.
12. Conitzer, V., & Garera, N. (2006). Learning algorithms for online principal–agent problems (and selling goods online). *Proceedings of the International Conference on Machine Learning*, *06*, 209–216.
13. Davidson, A., Billings, D., Schaeffer, J., & Szafron, D. (2000). Improved opponent modeling in poker. In *Proceedings of the International Conference on Artificial Intelligence* (pp. 1467–1473).
14. Dempster, A. P., Laird, N. M., & Rubin, D. B. (2006). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society*, *61*(4), 617–631.
15. Diermeier, D., & Morton, R. (2005). Proportionality versus perfectness: Experiments in majoritarian bargaining. In D. Austen-Smith & J. Duggan (Eds.), *Social choice and strategic behavior* (pp. 157–196). Berlin: Springer.
16. DiMicco, J., Greenwald, A., & Maes, P. (2001). Dynamic pricing strategies under a finite time horizon. In *Proceedings of the ACM Conference on Electronic Commerce*.
17. Dumas, M., Aldred, L., Governatori, G., & ter Hofstede, A. (2005). Probabilistic automated bidding in multiple auctions. *Electronic Commerce Research*, *5*(1), 25–49.
18. Ebay Inc. (2012). Retrieved March 25, 2012, from www.ebay.com
19. Fatima, S., Wooldridge, M., & Jennings, N. R. (2005). Sequential auctions for objects with common and private values. *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, *05*, 635–642.
20. Ferguson, G. A. (1981). *Statistical analysis in psychology and education*. New York: McGraw-Hill.
21. Gal, Y., Pfeffer, A., Marzo, F., & Grosz, B. (2004). Learning social preferences in games. *Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI)*, *04*, 226–231.
22. Gerding, E. H., Dash, R. K., Yuen, D. C. K., & Jennings, N. R. (2007). Bidding optimally in concurrent second-price auctions of perfectly substitutable goods. *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, *07*, 267–274.
23. Ghose, T. K., & Tran, T. (2009). Dynamic pricing in electronic commerce using neural network. In *Proceedings of the 4th International MCETECH Conference on e-Technologies (MCETECH) 09* (pp. 227–232).
24. Gittins, J. (1989). *Multiarmed bandits allocation indices*. New York: Wiley.
25. Google Advertising Program. (2012). Retrieved March 25, 2012, from www.google.com/ads
26. Grosskopf, B. (2003). Reinforcement and directional learning in the ultimatum game with responder competition. *Experimental Economics*, *6*(2), 141–158.
27. Guth, W., & Huck, S. (1997). From ultimatum bargaining to dictatorship: An experimental study of four games varying in veto power. *Metroeconomica*, *48*(3), 262–299.
28. Guth, W., Schmittberger, R., & Schwarz, B. (1982). An experimental analysis of ultimatum bargaining. *Economic Behavior and Organization*, *3*, 367–388.

29. Katz, R., & Kraus, S. (2006). Efficient agents for Cliff-Edge environments with a large set of decision options. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS) 06*.
30. Katz, R., & Kraus, S. (2006). Efficient bidding strategies for simultaneous Cliff-Edge environments. In *Intelligent agent technology 06*.
31. Katz, R., & Kraus, S. (2007). Gender-sensitive automated negotiators. In *Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI) 07*.
32. Knez, M., & Camerer, C. (1995). Outside options and social comparison in a three-player ultimatum game experiments. *Games and Economic Behavior*, *10*, 65–94.
33. Krishna, V., & Morgan, J. (1997). An analysis of the war of attrition and the all-pay auction. *Journal of Economic Theory*, *72*, 343–362.
34. Lagarias, J. C., Lagarias, J. C., Reeds, J. A., Reeds, J. A., Wright, M. H., Wright, M. H., et al. (1996). Convergence properties of the nelder-mead simplex algorithm in low dimensions. *SIAM Journal of Optimization*, *9*, 112–147.
35. Leloup, B., & Deveaux, L. (2001). Dynamic pricing on the internet: Theory and simulations. *Journal of Economic Research*, *1*(3), 265–276.
36. Lin, R., Kraus, S., Wilkenfeld, J., & Barry, J. (2006). An automated agent for bilateral negotiation with bounded rational agents with incomplete information. In *Proceedings of the European Conference on Artificial Intelligence (ECAI) 06*.
37. Maes, P. (1995). Artificial life meets entertainment: Lifelike autonomous agents. *Communication of the ACM*, *38*(11), 108–114.
38. Milidiu, R. L., Melcop, T., Liporace, F. T. S., & Lucena, C. J. P. (2003). Simple: A multi-agent system for simultaneous and related auctions. In *Intelligent agent technology 03*.
39. Minton, S., Johnston, M. D., Philips, A. B., & Laird, P. (1992). Minimizing conflicts: A heuristic repair method for constraint satisfaction and scheduling problems. *Artificial Intelligence*, *58*, 161–205.
40. Mitchell, M. (1996). *An introduction to genetic algorithms*. New York: MIT press.
41. Nelder, J. A., & Mead, R. (1965). A simplex method for function minimization. *The Computer Journal*, *7*(4), 308–313.
42. Niklasson, L., Engstrom, H., & Johansson, U. (2001). An adaptive 'rock, scissors and paper player' based on a tapped delay neural network. In *Proceedings of the International Conference on Application and Development of Computer Games in the 21st Century (ADCOG)* (pp. 130–136).
43. Oliveira, E., Fonseca, J. M., & Jennings, N. R. (1999). Learning to be competitive in the market. In *AAAI Workshop on Negotiation: Settling Conflicts and Identifying Opportunities* (pp. 30–37).
44. Rosenfeld, A., & Kraus, S. (2009). Modeling agents through bounded rationality theories. In *Proceedings of the International Joint Conferences on Artificial Intelligence (IJCAI) 09*.
45. Roth, A., & Erev, I. (1995). Learning in extensive form games: Experimental data and simple dynamic models in the intermediate term. *Games and Economic Behavior*, *8*, 164–212.
46. Strens, M. (2000). A Bayesian framework for reinforcement learning. In *Proceedings of the International Conference on Machine Learning* (pp. 943–950).
47. Sutton, R., & Barto, A. (1998). *An introduction to reinforcement learning*. New York: MIT Press.
48. Todd, P., & Borges, B. (1997). Designing socially intelligent agents for the ultimatum game. In K. Dautenhahn (Ed.), *Socially Intelligent Agents-Papers from the 1997 Fall Symposium* (pp. 134–136). Menlo Park, CA: AAAI Press.
49. Vreind, N. (1997). Will reasoning improve learning? *Economics Letters*, *55*(1), 9–18.
50. Xianyu, B. (2010). Social preference, incomplete information, and the evolution of ultimatum game in the small world networks: An agent-based approach. *Artificial Societies and Social, Simulation*, *13*(2), 223.
51. Yuen, D., Byde, A., & Jennings, N. R. (2006). Heuristic bidding strategies for multiple heterogeneous auctions. In *Proceedings of the European Conference on Artificial Intelligence (ECAI) 06*.
52. Zhong, F., Wu, D., & Kimbrough, S. (2002). Cooperative agent systems: Artificial agents play the ultimatum game. *Group Decision and Negotiation*, *11*(6), 433–447.
53. Zhu, W., & Wurman, P. R. (2002). Structural leverage and fictitious play in sequential auctions. *Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI)*, *02*, 385–390.