# Multiple Offer Strategy for Automated Negotiation Agents

Kivanc Ozonat
HP Labs
1501 Page Mill Road
Palo Alto, CA
kivanc.ozonat@hp.com

## ABSTRACT

Automated negotiation agents negoatite issues of an e-commerce transaction with human consumers on behalf of e-commerce vendors, and they can increase the financial benefits of the consumer and the vendor jointly. Both the artificial intelligence community and the economics (game-theory) community have proposed methods for negotiation agent design. The focus has been on agents that are restricted to make a single counteroffer to the consumer at every round of the negotiation. Recent studies from the psyhchology community, however, indicate that making multiple counteroffers per negotiation round can be beneficial to both the consumer and the vendor. In light of this, we design an automated software agent that makes multiple offers at every round. We devise a probabilistic strategy that guides the agent to find the optimal set of counteroffers.

## Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

## Keywords

automated, negotiation, multiple offers, statistics

## 1. INTRODUCTION

Negotiation is the process of reaching an agreement between two (or more) parties on the issues underlying a transaction. Negotiations are an integral part of business life, and are instrumental in reaching agreements among the parties.

Over the last two decades, electronic commerce has become widely adopted, providing consumers with the ability to purchase products and services from businesses online. Electronic commerce offers many advantages, including increased operational efficiency for the businesses, reduction of the inventory costs, and availability of the product and service 24 hours a day. Yet, after two decades, electronic commerce systems still lack the ability to enable the businesses to negotiate with consumers during the purchases. There is a need for automated, online software agents that can negotiate with consumers on behalf of businesses.

Designing automated agents for negotiation has been explored across multiple disciplines, in artificial intelligence [5, 6, 7], human psychology [17, 18, 19, 22] and statistical learning [3, 9]. One form of negotiation that has received much attention in these disciplines is the *bilateral sequential negotiation.* A bilateral sequential negotiation typically starts with one party (e.g., the *buyer*) making an offer on each of the negotiated issues. The issues can include, for instance, the price of a product or service, its delivery time and its quantity. The opposing party (e.g., the *seller*) has three options: (i) it accepts the offer, (ii) it rejects the offer and makes a counter offer on each issue, or (iii) it rejects the offer and terminates the negotiation. The process continues in rounds, where, the buyer and the seller make and respond to counter offers.

The artificial intelligence community has focused on finding game-theoretic solutions to the bilateral sequential negotiation problem. Each of the two parties is assumed to have a *utility function* of the negotiated issues. The utility function measures how preferable an offer is to the party. In the game-theoretic model, each party is assumed to have the goal of maximizing its utility function, and the aim is to find an agreement that is *Pareto-optimal*, i.e., an agreement where the utility of one party cannot be increased any further without decreasing the utility of the other [6].

Many game-theoretic models of negotiation assume that each party knows the opposing party's utility function [6], while more realistic models employ statistical algorithms that *learn* the opposing party's utility function [3, 5, 9]. The learning can be based, for instance, on a training set of issue offers from the parties.

One of the primary goals of an automated agent during the negotiation is to prevent the buyer from terminating the negotiation without reaching an agreement. If the agent's offers are consistently far away from what is acceptable to the buyer, the buyer is likely to walk away from the negotiation. Recent studies from the psychology community indicate that making multiple counteroffers per negotiation round can be beneficial to both the consumer and the vendor [21]. The focus of the game theory and artifical intelligence literature, however, have been on agents that are restricted to make a single counteroffer to the consumer at every round of the negotiation.

We address the question of designing an automated software agent that makes multiple offers at every round. We devise a probabilistic strategy that guides the agent to find the optimal set of counteroffers. Our approach to positioning the multiple offers strategically would be to cluster the issue vectors based on their proximity under some distance $d$ such that the intra-cluster distances are small and the inter-cluster distances are large [10]. Then, from each cluster, we select the cluster member that is closest to the buyer's last offer.

## 2. MULTIPLE OFFER MODEL

### 2.1 Assumptions

We assume the following about the buyer's behavior during the negotiation:

1. The agent has the knowledge of (or can estimate from the buyer's history of offers during the negotiation) how close any two offers are from the perspective of the buyer.

2. For any buyer, there exist an "indifference volume" $V(u)$ around each vector $u$ in the issue space, defined as the set of issue vectors where the buyer's gains from preferring any $v \in V(u)$ over $u$ is very small. In particular, the buyer is unlikely to risk termination of the negotiation by insisting on $u$ instead of some other $v \in V(u)$. The agent does not know the size or shape of the volume $V(u)$.

The first assumption is very common in the artificial intelligence view of the negotiation literature. While the game-theoretic approaches assume that the agent makes its counteroffers based on a buyer's utility function known to both parties, the artificial intelligence-based approaches often focus on designing agents that make counteroffers by estimating distances between issue points in the vector space. In [5], for instance, the agent offers the point closest to the buyer's last offer based on some estimated distance.

The second assumption follows from that the buyer is unlikely to form mathematical functions (e.g., utility functions, iso-curves, etc.) upon which its decisions are based. Instead, the buyer is more likely to make its decisions based on a mixture of quantitative and qualitative criteria. We note that we only assume the existence of an indifference volume around each issue vector. We do not make any assumptions about the size or the shape of the volume.

### 2.2 Model

In designing negotiation agents in the artificial intelligence community, a very common offer strategy is to offer the issue, which is closest to the buyer's last offer among a set of issue vectors. Often, the agent forms the set such that all issue vectors in the set has the same utility for the agent, i.e., the issue vectors are on the same agent utility *isocurve*. Then, the agent offers the member that is closest to the buyer's last offer.

In this strategy, the distance between an issue vector $u$ and the buyer's last offer $y$ is often measured as a weighted sum of their elementwise distances, i.e.,

$$D(u, y) = \sum_i w_i D_i(u_i, y_i), \tag{1}$$

where the $u_i$ and $y_i$ are the $i^{th}$ elements of the vectors $u$ and $y$, and the weights $w_i$ are computed based on the estimated preferences of the buyer. The agent can estimate these preferences from the buyer's offer history. The distance measure $D_i$ can be, for instance, the elementwise absolute difference ($|u_i - y_i|$) between the two vectors.

What if the agent is to make two offers at each round? Extending the approach in [5], the agent can select the closest two issue vectors to $y$ on the agent's isocurve as offers. However, this is not necessarily a good strategy especially if these two offers are very close to each other. If the two closest vectors are within a close neighborhood of each other, for instance, this strategy would not add much beyond offering just one of these two vectors.

At the other extreme is to select the first offer as the closest one to $y$ on the isocurve, and the second offer on the isocurve as far from the first offer as possible. This strategy is far from optimal since the second offer takes into account neither the buyer's offer $y$ nor his/her preferences.

To quantify the discussion, assume the buyer offers $y$ in the last round. Let $p_y(u_1)$ denote the probability that the buyer terminates the negotiation in response to agent's offer $u_1$, and let $p_y(u_2|u_1)$ denote the conditional probability that the buyer terminates the negotiation in response to the offer $u_2$ given the buyer would terminate it in response to $u_1$. Then, by the definition of conditional probability, the probability of the buyer continuing with the negotiation is

$$1 - p_y(u_1, u_2) = 1 - p_y(u_1)p_y(u_2|u_1). \tag{2}$$

How to model $p_y(u_1)$ and $p_y(u_2|u_1)$? From the discussion in [2002] and [2006], one viable option is to let $p_y(u_1)$ to be proportional to the distance between $u_1$ and $y$, i.e.,

$$p_y(u_1) = k_1|y - u_1|, \tag{3}$$

for some constant $k_1$.

What about $p_y(u_2|u_1)$, i.e., the probability that $u_2$ gets rejected given that $u_1$ would be rejected? Intuitively, this conditional probability should depend on both the distance between $y$ and $u_2$, and the distance between $u_1$ and $u_2$. In particular, if the distance between $u_1$ and $u_2$ is small, the conditional probability should depend more on the distance between $u_1$ and $u_2$. On the other hand, if the distance between $u_1$ and $u_2$ is large, the conditional probability should be more dependent on the distance between $y$ and $u_2$. Accordingly, we let

$$p_y(u_2|u_1) = \begin{cases} k_2|y - u_2| & \text{if } |u_1 - u_2| \geq \epsilon, \\ 1 & \text{if } |u_1 - u_2| < \epsilon, \end{cases} \tag{4}$$

for some constants $k_2$ and $\epsilon$.

Then, the probability that the buyer will continue with the negotiation becomes

$$1 - p_y(u_1, u_2) = \begin{cases} 1 - k_1k_2|y - u_1||y - u_2| & \text{if } |u_1 - u_2| \geq \epsilon, \\ 1 - k_1|y - u_1| & \text{if } |u_1 - u_2| < \epsilon. \end{cases} \tag{5}$$

We note that the probability $k|y - u| < 1$ for any $k_1$,$k_2$ and $u$ since $p_y$ is a probability mass function.

If $u_2$ is placed very close to $u_1$ (i.e., within a distance of $\epsilon$), then the acceptance probability is guaranteed to be lower than the case with $u_2$ placed far away from $u_1$ (i.e., outside a distance of $\epsilon$). This follows from

$$1 - k_1k_2|y - u_1||y - u_2| > 1 - k_1|y - u_1|, \tag{6}$$

since $k|y - u| < 1$ for any $k_1$,$k_2$ and $u$.

One can then extend the probability of continuing with the negotiation to $N > 2$ offers, and show that with $N$ simulatenous agent offers, the probability of the buyer walking away from the negotiation is minimized if no two offers are within a distance of $\epsilon$. In this case, the probability of the buyer walking away from the negotiation is

$$\prod_i k_i |y - u_i|, \tag{7}$$

provided no two offers are within a distance of $\epsilon$ of each other.

Minimizing (8) is equivalent to minimizing

$$\sum_i \log k_i + \log |y - u_i|. \tag{8}$$

The optimization problem in (8) is, in general, intractable, and thus we use a suboptimal strategy to minimize (8).

Our multiple offer strategy is to first partition the issue space using graph partitioning [10] into multiple clusters such that the partitioning satisfies the following conditions:

1. Any two issue vectors, $u$ and $v$, within the same clusters are (very likely to be) close to each other, i.e., within a distance $\epsilon$.

2. Any two issue vectors, $u$ and $v$, between two clusters are (very likely to be) far from each other, i.e., outside a distance of $\epsilon$.

Once the partitioning stage is done, we select one issue vector from each cluster as an offer. In each cluster, we select the offer that is closest to the last offer of the buyer. This strategy ensures that the offer closest to the buyer's last offer in the issue space is always selected. Thus, the first offer is the same as that one would get from the strategies in [5]. It further ensures that the remaining offers are not very close.

The motivation for the multiple offer strategy is that it is rare for the agent to have perfect estimates of the buyer's preferences $w_i$. The inaccuracies in estimating the preferences will impact the distance function. For most negotiation scenarios, it is reasonable to assume that $D_i$ would be the Euclidean distance or the city-block distance. In fact, some prior work just assume that $D_i$ is simply the absolute elementwise difference between vectors (i.e., the city-block distance).

## 3. CLUSTERING

### 3.1 Definitions

Let $V$ denote a finite set of elements, and $E$ be a set of edges $e$ such that each edge is connects of the elements in $V$. $G = (V, E)$ is called a graph with the vertex set $V$, and edge set $E$. A weighted graph is a graph that has a positive number $w(e)$ associated with each edge $e$, called the weight of edge $e$. Denote a weighted graph by $G = (V, E, w)$. An edge $e$ is said to be incident with a vertex $u$ when $e$ connects $u$ to another edge.

We say that there is a path between vertices $v_1$ and $v_k$ when there is an alternative sequence of distinct vertices and edges $v_1$, $e_1$, $v_2$, $e_2$,..., $e_{k-1}$, $v_k$ such that $v_i, v_{i+1} \in e_i$, for $1 \leq i \leq k - 1$. A graph is connected if there is a path for every pair of vertices. We assume that the graphs in this work are connected.

### 3.2 Graph clustering

The graph clustering problem with $k$ clusters is formalized as follows [10]: Given a graph $G = (V, E)$, find the clustering $V_1$, $V_2$,..., $V_k$ that minimizes the number of edges of $E$ whose incident vertices belong to different subsets, provided

- $V_i \cap V_j = \emptyset$ for $i \neq j$,
- $|V_i| = |V|/k$, and $\cup_i V_i = V$,

If the edges have weights associated with them, the graph clustering problem can be extended to account for the edge weights. In this case, the problem can be formulated as minimizing the sum of the edge weights belonging to different subsets provided the two conditions listed above. Given a clustering, the number of edges whose incident vertices belong to different subsets is called the edge-cut of the clustering.

The graph $G$ can be clustering using a multi-level algorithm. The graph $G$ is first coarsened down to a few hundred vertices, a partitioning of this smaller graph into $k$ clusters is computed, and then this partition is projected back towards the original graph (finer graph). At each step of the graph uncoarsening, the partition is further refined. The refinements reduce the edge-cut since Since the finer graph has more degrees of freedom.

Consider a weighted graph $G_0 = (V_0, E_0)$ with weights both on the edges. A multi-level graph clustering algorithm consists of the following three stages:

- Coarsening Phase: The graph $G_0$ is transformed into a sequence of smaller graphs $G_1$, $G_2$,..., $G_m$ such that $V_0 > V_1 > V_2 > .... > V_m$.
- Partitioning Phase A 2-way partition $P_m$ of the graph $G_m = (V_m, E_m)$ is computed that partitions $V_m$ into two parts, each containing half the vertices of $G_0$.
- Uncoarsening Phase The partition $P_m$ of $G_m$ is projected back to $G_0$ by going through intermediate partitions $P_{m-1}, P_{m-2}, ...., P_1, P_0$.

### 3.3 Matching

A matching of a graph $G_i = (V_i, E_i)$ is a set of edges such that no two edges are incident on the same vertex. The coarser graph $G_{i+1}$ is constructed from $G_i$ by finding a matching of $G_i$ and collapsing the matched vertices into multinodes. The unmatched vertices are simply copied over to $G_{i+1}$. The matching of a graph is obtained through forming *maximal matchings.* A matching is maximal if any edge in the graph that is not in the matching has at least one of its endpoints matched.

A maximal matching can be generated efficiently using a randomized algorithm as follows:

- Vist the vertices in random order.
- If a vertex $u$ has not been matched yet, then randomly select one of its unmatched adjacent vertices.
- If such a vertex $v$ exists, include the edge $(u, v)$ in the matching and mark vertices $u$ and $v$ as being matched.
- If there is no unmatched adjacent vertex $v$, then vertex $u$ remains unmatched in the random matching.

The complexity of the above algorithm is $O(E)$.

While random matching is an efficent technique to obtain a maximal matching, it does not target minimizing the edge-cut. Consider a graph $G_i = (V_i, E_i)$, a matching $M_i$ that is used to coarsen $G_i$ , and its coarser graph $G_{i+1} = (V_{i+1}, E_{i+1})$ induced by $M_i$. If $A$ is a set of edges, define $W(A)$ to be the sum of the weights of the edges in $A$. It can be shown that

$$W(E_{i+1}) = W(E_i) - W(M_i). \tag{9}$$

Thus, the total edge-weight of the coarser graph is reduced by the weight of the matching. Hence, by selecting a maximal matching Mi whose edges have a large weight, we can decrease the edge-weight of the coarser graph by a greater amount. As the analysis in [27] shows, since the coarser graph has smaller edge-weight, it also has a smaller edge-cut. Finding a maximal matching that contains edges with large weight is the idea behind the heavy-edge matching. A heavy-edge matching is computed using a randomized algorithm similar to that for computing a random matching described earlier. The vertices are again visited in random order. However, instead of randomly matching a vertex $u$ with one of its adjacent unmatched vertices, we match $u$ with the vertex $v$ such that the weight of the edge $(u, v)$ is maximum over all valid incident edges (heavier edge).

## 3.4 Partitioning

The second phase of a multilevel algorithm computes a high-quality bisection (i.e., small edge-cut) $P_m$ of the coarse graph $G_m = (V_m, E_m)$ such that each part contains roughly half of the vertex weight of the original graph. Since during coarsening, the weights of the vertices and edges of the coarser graph were set to reflect the weights of the vertices and edges of the finer graph, $G_m$ contains sufficient information to intelligently enforce the balanced partition and the small edge-cut requirements.

The Kernighan-Lin algorithm [31] is iterative in nature. It consists of the following iterations:

1. Start with an initial partition of the graph.
2. Search for a subset of vertices from each part of the graph such that swapping them leads to a partition with a smaller edge-cut.
3. Stop if you cannot find such two subsets. This indicates that the partition is at a local minimum.

Each iteration of the KL algorithm described in [31] takes $O(ElogE)$ time.

The Kernighan-Lin algorithm finds locally optimal partitions when it starts with a good initial partition and when the average degree of the graph is large [4]. If no good initial partition is known, the KL algorithm is repeated with different randomly selected initial partitions, and the one that yields the smallest edge-cut is selected.

Suppose $P$ is the initial partition of the vertices of $G_m = (V_m, E_m)$. The gain $g_v$, of a vertex $v$ is defined as the reduction on the edge-cut if vertex v moves from one partition to the other. This gain is given by

$$g_v = \sum_{(v,u)\in E \cap P[v]\neq P[u]} w(u,v) - \sum_{(v,u)\in E \cap P[v]=P[u]} w(u,v) \tag{10}$$

where $w(v, u)$ is weight of edge $(v, u)$. If $g_v$ is positive, then by moving $v$ to the other partition the edge-cut decreases by $g_v$, whereas if $g_v$ is negative, the edge-cut increases by the same amount. If a vertex $v$ is moved from one partition to the other, then the gains of the vertices adjacent to $v$ may change. Thus, after moving a vertex, we need to update the gains of its adjacent vertices.

Given this definition of gain, the KL algorithm then proceeds by repeatedly selecting from the larger part a vertex $v$ with the largest gain and moves it to the other part. After moving $v$, $v$ is marked so it will not be considered again in the same iteration, and the gains of the vertices adjacent to $v$ are updated to reflect the change in the partition. The original KL algorithm [9], continues moving vertices between the partitions, until all the vertices have been moved.

## 3.5 Refinement

During the uncoarsening phase, the partition $P_m$ of the coarser graph $G_m$ is projected back to the original graph, by going through the graphs $G_{m-1}$, $G_{m-2}$,...., $G_1$. Since each vertex of $G_{i+1}$ contains a distinct subset of vertices of $G_i$, obtaining $P_i$ from $P_{i+1}$ is done by simply assigning the set of vertices $V_i^v$ collapsed to $v \in G_{i+1}$ to the partition $P_{i+1}[v]$.

Even though $P_{i+1}$ is a local minimum partition of $G_{i+1}$, the projected partition $P_i$ may not be at a local minimum with respect to $G_i$. Since $G_i$ is finer, it has more degrees of freedom that can be used to improve $P_i$, and decrease the edge-cut. Hence, it may still be possible to improve the projected partition of $G_{i-1}$ by local refinement heuristics. For this reason, after projecting a partition, a partition refinement algorithm is used. The basic purpose of a partition refinement algorithm is to select two subsets of vertices, one from each part such that when swapped the resulting partition has a smaller edge-cut.

The refinement algorithm consists of the following steps:

1. Use the projected partition of $G_{i+1}$ onto $G_i$ as the initial partition. (Note that the projected partition is already a good partition).
2. Apply vertex swaps to decrease the edge-cut.

$$g_v = \sum_{(v,u)\in E \cap P[v]\neq P[u]} w(u,v) - \sum_{(v,u)\in E \cap P[v]=P[u]} w(u,v) \tag{11}$$

3. Terminate the algorithm when no further decrease in the edge-cut can be made through edge swaps.

## 4. SIMULATIONS

Automated software agents for bilateral negotiations can be designed through either game-theoretic methods or statistical learning approaches. In either case, a robust design, evaluation and testing of the agents requires the existence of a (training) set of issue offers representing typical behaviors of the opposing party. When a set of issue offers recorded during an actual negotiation process is available, it can be used to design, evaluate and test the agents. However, it is very rare to have recordings of actual negotiations, in which case, the issue offers need to be obtained through simulations.

In [5], the buyers' negotiation behaviors have been modeled through a sequence of functions; each function imitates the buyer's offers in the issue space. The functions are monotonic in the issue values and belong to families of parametrized functions. The models in [5] assume a single-issue negotiation scenario, which we discuss in section 4.1. We then extend it to the scenario with multiple negotiated issues in section 4.2.

## 4.1 Single-issue buyer profiles

According to [5], negotiation tactics can be classified into two main categories: (i) time-dependent (or resource-dependent) tactics, and (ii) behavior-dependent tactics. Time-dependent tactics determine how the issue offer $y_{n,d}$ changes as a function of time $n$. Resource-dependent tactics are a generalization of the time-dependent tactics, where resources other than time (e.g., inventories) are considered. The behavior-dependent tactics, on the other hand, determine how the issue offer $y_{n,d}$ changes as a function of the counter offers of the opposing party.

We focus on the time-dependent tactics. In [5], the evolution of the offer $y_{n,d}$ through time is given as

$$y_{n,d} = y_{\min,d} + \alpha_d(n)(y_{\max,d} - y_{\min,d}) \qquad (12)$$

where $[y_{\min,d}\ y_{\max,d}]$ is the range of values for issue $d$, and $\alpha_d$ is a time-varying function.

There are many choices for the function $\alpha_d$ as discussed in [5]. Each function is monotonic in the issue value, and has five parameters: $\beta_d$, $t_{\max}$, $y_{\min,d}$, $y_{\max,d}$ and $\kappa$. The parameter $\beta_d$ controls the convexity of the function. When $\beta_d < 1$, the function is convex, and when $\beta_d > 1$ the function is concave. The following is one such example function:

$$\alpha_d(n) = \kappa + (1-\kappa)(\min(n, t_{\max})/t_{\max})^{1/\beta_d}. \qquad (13)$$

Of the five parameters, $\kappa$ is a constant (usually set to 0), $\beta_d$ controls the concession behavior, and $t_{\max}$ is the maximum number of negotiation rounds. The notation $\min(n, t_{\max})$ implies the smaller of $n$ and $t_{\max}$. As $\beta_d$ is decreased, the party makes more "boulware" offers, i.e., the party stays closer to the $y_{\min,d}$ value until the time is almost exhausted, whereupon it concedes up to $y_{\max,d}$. As $\beta_d$ is increased, the party makes more "conceder" offers, i.e., the party quickly moves up to $y_{\max,d}$.

## 4.2 Multi-issue buyer profiles

The treatment in [5] of the negotiation tactics focuses on single-issue negotiation settings. When the negotiation involves multiple issues, however, one needs to take into account the dependencies among the issues.

The model in [5] assumes that the negotiating party (e.g., the buyer) starts with an initial issue offer $y_{\min,d}$, and increases the issue value at each time step (or round) $n$, according to, for instance (12) and (13). The multiple issue negotiation literature, on the other hand, focuses on game-theoretic approaches with utility functions. The assumption is that a negotiating party starts with an initial issue vector with utility equal to $U_{initial}$ and aims to reach an agreement at a target vector with utility $U_{target}$. By the very nature of negotiation, $U_{target}$ is less than $U_{initial}$. The literature also takes the view that the party decreases the utility value monotonically at every round.

To extend the model in [5] for simulating buyer's behavior in a multi-issue setting, we randomly select one of the buyer's target vectors as $y_{\max,d}$. We then randomly select another vector as $y_{\min,d}$, such that each element of $y_{\max,d}$ is greater than or equal to the corresponding element of $y_{\min,d}$. This condition is needed to ensure that the monotonic functions (e.g., (13)) can be used to move from $y_{\min,d}$ to $y_{\max,d}$ in a finite number of rounds. At each time $n$, we either keep the $\beta_d$ value for each issue $d$ (no change in the state), or switch to a new $\beta_d$ value for each issue $d$ according to some prior transition probability as discussed in [5].

The parameter $\beta_d$ needs to be selected to ensure the convexity (or concavity) of $\alpha_d$. According to the Faratin's approach, the negotiator's tactic can be characterized as either "boulware" or "conceder". In particular, for $\beta_d < 1$, the tactic is boulware, while for $\beta_d > 1$, we observe the conceder tactics. The function (13) is convex in $n$, and the degree of convexity, determined by the value of $\beta_d$, identifies the tactic. As $\beta_d$ is increased, the offers get more conciliatory, and as $\beta_d$ is decreased, the offers become more boulware. Faratin provides functions alternative to that in (13) (e.g., an exponential function instead of a polynomial function), yet their common characteristic is that they are convex in $n$, and their degree of convexity is determined through the parameter $\beta_d$.

Since a function is (strictly) convex if and only if its second derivative is (strictly) non-negative, and that a weighted sum of second derivatives is equal to the second derivative of the weighted sums, it follows that the additive utility function, $U$, defined as

$$U = \sum_d w_d U_d \qquad (14)$$

for some weights $w_d > 0$ and the issue utilities $U_d$, is guaranteed to be convex as long as $\beta_d < 1$ for each $d$, and is guaranteed to be concave as long as $\beta_d > 1$ for each $d$. Thus, in transitioning from one set of $\beta_d$ values to another in the simulations, we make sure that, in each state, either $\beta_d < 1$ for each $d$ or $\beta_d > 1$ for each $d$. This ensures the convexity (or the concavity) in the utility space as well as in the issue space.

## 4.3 Results

We evaluate the performance of the seller's agent. The agent negotiates issues with buyers on behalf the seller, where the buyer's offers on the issues are generated through the simulations described in sections 4.1 and 4.2. The agent is available to it a rank-ordered list of the issue vectors with the higher-ranking vectors being more advantageous to the seller. The top 20 percent of the issue vectors in the list are acceptable to the seller. The buyer has a different rank-ordered list of the issue vectors, and the top 20 percent of the issue vectors in the list are acceptable to the buyer.

In Fig. 1, we assigned a *utility score* (in the 0-100 range) to each issue point with higher-ranking points (in the buyer's ranked-ordered list) with higher utility scores, and computed the mean of the utility values of the agent's counter offers. In Fig. 1, the concession behavior of the buyer is that of a conceder (i.e., $\beta > 1$). As the number of offers $N$ by the agent increases, the mean utility value for the buyer increases. The gains are greatest when $N$ is increased from 2 to 5, and the gains are small when $N$ is further increased to 10. This might indicate that generating 5 offers per negoti-

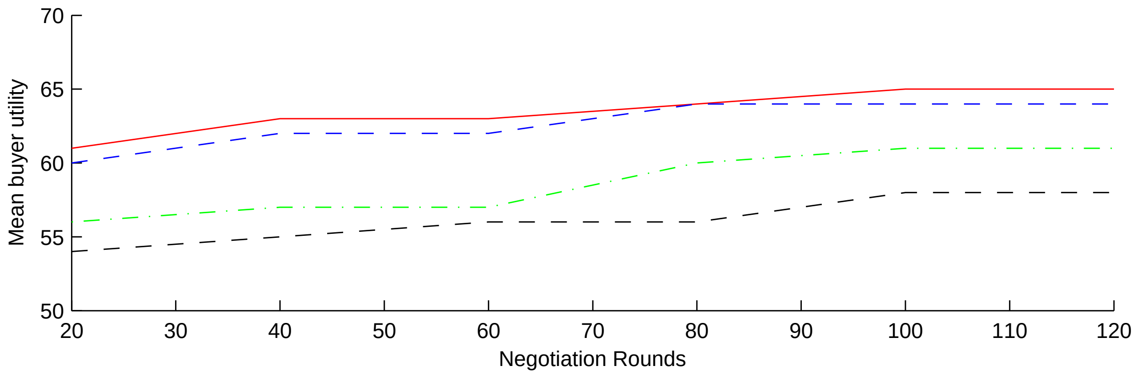

**Figure 1: The buyer's utility as a function of the number of negotiation rounds. Conceder behavior. Single offer (black dashed). Two offers (green dash dot). Five offers (blue dashed). Ten offers (red solid).**

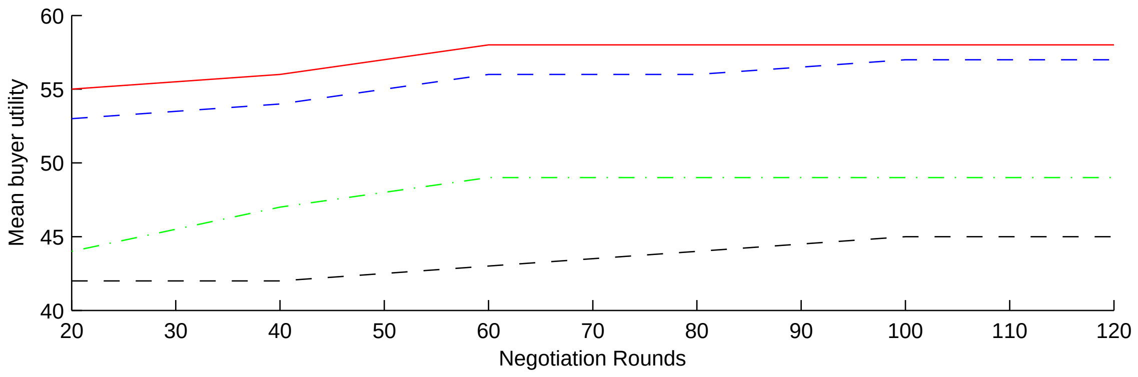

**Figure 2: The buyer's utility as a function of the number of negotiation rounds. Boulware behavior. Single offer (black dashed). Two offers (green dash dot). Five offers (blue dashed). Ten offers (red solid).**

ation round is a sufficiently good strategy.

Fig. 2 shows the mean of the utility values of the agent's counter offers for a boulware buyer model (i.e., $\beta < 1$). Similar to the number of offers by the agent increases, the mean utility value for the buyer increases. However, the gains from the multiple offers in this case are higher, indicating that the multiple offer strategy is more benefitial for boulware buyers than it is for conceder buyers. Finally, Fig. 3 shows the behavior for a linear buyer model. Similar to Fig. 1 and Fig. 2, the mean utility value of the buyer increases as the number of offers by the agent increases.

## 5. CONCLUSIONS

Automated negotiation agents negoatite issues of an e-commerce transaction with human customers on behalf of e-commerce vendors. The agents proposed by the artificial intelligence and game theory communities have focused on designing agents that make a single offer to the customer at every round of the negotiation. Recent studies from the psyhchology community, however, indicate that making multiple counteroffers per negotiation round can be beneficial to both the consumer and the vendor. In this work, we designed automated agents that make multiple offers to the customer at every negotiation round. We have utilized graph-based statistical clustering to partition the space of the offers and generate the multiple offers. Our results, based on popular buyer behavior models, indicate that this strategy leads to a significant increase in the customer's utility without decreasing the agent's utility.

## References

[1] A. Barron, J. Rissanen, and B. Yu. The minimum description length principle in coding and modeling. *IEEE Transactions in Information Theory*, 44(6):2743-2765, 1998.

[2] M. Beal, Z. Ghahramani, and C.E. Rasmussen. The infinite hidden Markov model. In *NIPS*, 2002.

[3] R.M. Coehoorn and N. Jennings. Learning an opponent's preferences to make effective multi-issue negotiation trade-offs. In *ICEC*, 2004.

[4] T. Cover and J. Thomas. *Elements of Information Theory*, New York, Wiley, 1991.

[5] P. Faratin, C. Sierra, and N. R. Jennings. Negotiation decision functions for autonomous agents. *Int. Journal of Robotics and Au- tonomous Systems*, 24(3-4):159Ű182, 1998.

[6] S. Fatima, M. Wooldridge, and N. R. Jennings. Multi-issue negotiation under time constraints. In *AAMAS*, 2002.

[7] S. Fatima, M. Wooldridge, and N. R. Jennings. An agenda based framework for multi-issues negotiation. *Artificial Intelligence Journal*, 152(1):1Ű45, 2004.

[8] E. B. Fox, E. B. Sudderth, M. I. Jordan, and A. S. Willsky. An HDP-HMM for systems with state persistence. In *ICML*, 2008.

[9] T. Ito, H. Hattori, and M. Klein. Multi-issue negotiation protocol for agents: Exploring nonlinear utility spaces. In *IJCAI*,

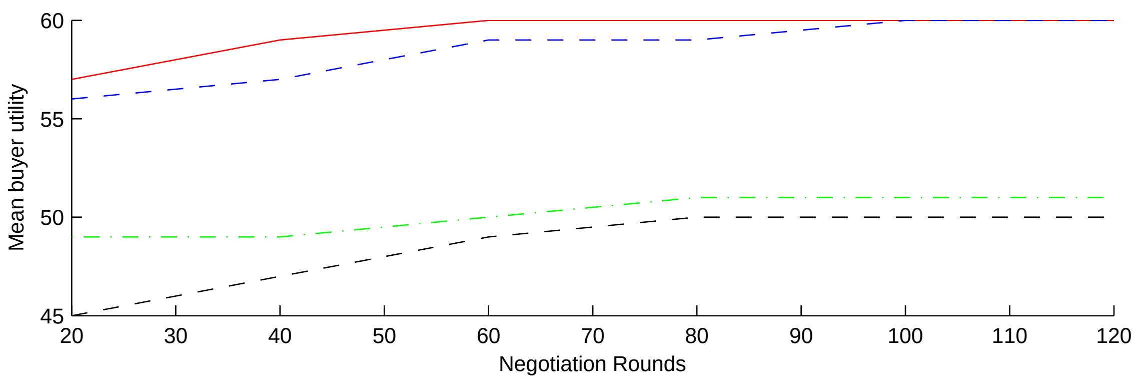

**Figure 3: The buyer's utility as a function of the number of negotiation rounds. Linear behavior. Single offer (black dashed). Two offers (green dash dot). Five offers (blue dashed). Ten offers (red solid).**

2007.

[10] G. Karypis and V. Kumar. A coarse-grain parallel formulation of multilevel k-way graph partitioning algorithm. *Siam Conference on Parallel Processing for Scientific Computing*, 2004.

[11] A. N. Kolmogorov. Three approaches to the quantitative definition of information. *Probl. Inform. Transm.* vol. 1. pp. 4-7, 1965

[12] V. Pavlovic, J. M. Rehg, T. J. Cham, and K. P. Murphy, A dynamic bayesian network approach to figure tracking using learned dynamic models. In *Intl. Conf. Computer Vision*, 1999.

[13] V. Pavlovic, J. Rehg, and J. MacCormick. Learning switching linear models of human motion. In *NIPS*, 2001.

[14] C. E. Shannon and W. Weaver. *The mathematical theory of communication*. University of Illinois Press, Urbana, IL, 1949.

[15] R. J. Solomonoff. A formal theory of inductive inference I,II, *Inform. and Control*, vol. 7 pp. 1-22, 224-254, 1964.

[16] Y. W. Teh, M. I. Jordan, M. J. Beal, and D. M. Blei. Hierarchical Dirichlet processes. *Journal of American Statistics Association*, 101(476):1566-1581, 2006.

[17] G. A. Van Kleef, C. K. W. De Dreu, and A. S. R. Manstead. The interpersonal effects of anger and happiness in negotiations. *Journal of Personality and Social Psychology*, 86, pp. 57Ű76, 2004.

[18] G. A. Van Kleef, C. K. W. De Dreu, and A. S. R. Manstead. The interpersonal effects of emotions in negotiations: A motivated information processing approach. *Journal of Personality and Social Psychology*, 87, pp. 510-528, 2004.

[19] R. Vetschera. Preference structures and negotiator behavior in electronic negotiations. *Decision Support Systems*, vol. 44, pp. 135-146, 2007.

[20] C. Wallace and D. L. Dowe. Minimum mesaage length and Kolmogorov complexity. *The Computer Journal*, vol. 42, no. 4, pp. 270-283, 1999.

[21] Y. Yang and S. Singhal.Designing an intelligent agent that negotiates tactfully with human counterparts: A conceptual analysis and modeling framework. *Proceedings of HICSS*, 2009.

[22] G. A. Yukl. Effects of situational variables and opponent concessions on a bargainerŠs perception, aspirations, and concessions. *Journal of Personality and Social Psychology*, vol. 29, pp. 227-236, 1974.