The Ultimatum Conundrum

Alan H. Karp, Kay-yut Chen, Ren Wu
Mobile and Media Systems Laboratory
HP Laboratories Palo Alto
HPL-2004-77
April 21, 2004*

E-mail: {alan.karp,kay-yut.chen,ren.wu}@hp.com

automated negotiation, game theory, bargaining

We have developed software that adapts the techniques used by game playing computer programs to the problem of negotiating good deals. In every test case, this strategy recommends making an ultimatum. On examining the protocol, we found that it is equivalent to a standard form of the bargaining game, which has an ultimatum as the Nash equilibrium. In practice, this strategy is unlikely to lead to a desirable result.

We report several failed attempts to induce the strategy to recommend more reasonable counteroffers. Only when we include uncertainty in the other participant's constraints does the strategy recommend a counteroffer other than an ultimatum. We show how this uncertainty can be included in the strategy algorithm without introducing heuristics.

* Internal Accession Date Only
Approved for External Publication

The Ultimatum Conundrum

Alan H. Karp, Kay-yut Chen, Ren Wu
Hewlett-Packard Laboratories
{alan.karp,kay-yut.chen,ren.wu}@hp.com

April 12, 2004

Abstract

We have developed software that adapts the techniques used by game playing computer programs to the problem of negotiating good deals. In every test case, this strategy recommends making an ultimatum. On examining the protocol, we found that it is equivalent to a standard form of the bargaining game, which has an ultimatum as the Nash equilibrium. In practice, this strategy is unlikely to lead to a desirable result.

We report several failed attempts to induce the strategy to recommend more reasonable counteroffers. Only when we include uncertainty in the other participant's constraints does the strategy recommend a counteroffer other than an ultimatum. We show how this uncertainty can be included in the strategy algorithm without introducing heuristics.

1 Introduction

Negotiation is an important and constant theme of research in multiple fields [5]. In this paper, we propose a reasonable framework for negotiations and an associated algorithm that can automate negotiations in this framework with some intelligence. The framework is reasonable in the sense that it includes many factors seen in face-to-face negotiations. The automation exhibits intelligence in the sense that the offers it makes can be explained in terms of the goals of the negotiator. The aim of this research is to create a tool that negotiates deals comparable to those people reach.

The framework for the negotiation [4] is a specific version of the bilateral offer-counteroffer game [1] in which an offer is defined by a set of parameters in a multiple dimensional space. Each parameter can take on values in a finite discrete set or in a subset of the real line. The initial offer is a subset of the parameter space of the contract template. The framework requires each subsequent offer to be a subset of the previous one. Each player also has the option of declaring "no deal" instead of making a counteroffer.

We developed an algorithm similar to that used by game playing computer programs to conduct automated negotiation under this framework [3]. This algorithm examines the whole decision tree and the associated expected payoffs,

with a special allowance for continuous decision variables, and finds the optimal strategy by backward induction. In the case where there is a cost associated with successive rounds of offers, this algorithm successfully finds the subgame perfect equilibrium in which the first mover poses an ultimatum to the other player and the second player accepts [6].

We know that most real world negotiations, which resemble the offer-counter-offer game, do not end in ultimatums, even when there is a time penalty. One of the reasons is that the model game assumes that the utility functions of both players are common knowledge and it does not take into account any learning effects. If I only have partial information about your utility function, and if I can learn about it through negotiation, it may pay for me to spend some time (and thus money) to learn about your utility function.

One way to represent my uncertainty in your utility is for me to represent it in the form of a mean and standard deviation. The learning can consist of updating the estimates of these values as the negotiation proceeds. However, simply adding this uncertainty does not solve the problem. The tree strategy simply recommends an ultimatum that is *likely* to have positive utility for the other party. A number of other attempts at getting the tool to recommend "reasonable" offers also failed. This failure represents the conundrum of the title of this paper.

One factor was found to produce reasonable offers, uncertainty in the other party's constraints. If I don't know what offers might cause you to walk away from the bargaining table, my best offer is one that leaves open as many options as possible. This offer must balance the goal of driving the negotiation in a direction favorable to me while maximizing the likelihood of reaching a deal. One way to achieve this goal is by using some heuristic, such as limiting the amount by which the space is narrowed on each offer. Such heuristics are often specific to each negotiation, and quantifying them is difficult.

We avoid heuristics in our model by allowing the algorithm to maintain a set of internally consistent estimates of the likelihood of rejection of different deals. For example, say that option B is worth slightly more to me than option C. In the Rubinstein bargaining model, I would make an ultimatum including option B. In our model, I take into account the fact that either of them may be unacceptable to you. For example, I may believe that each of them has a 50% chance of being rejected. If I pick option B, my expected payoff is half the utility of any deal containing B. Assuming independence, not a necessary condition but one that makes the example simpler, I get higher expected utility if I offer a set that includes both B and C and let you choose between them. Giving you the choice to choose B or C increases my probability of completing the deal substantially (75% *vs.* 50%), which can be interpreted as a kind of implicit learning.

Our model includes a full specification of utility functions with uncertainty and a set of internally consistent estimates of the probabilities of rejections of different deals. The estimate of the opponent's utility function are used to predict the opponent's counteroffers in order to estimate the direction the negotiation will take. The estimate of the opponent's constraints balance the

tendency toward the Nash equilibrium. Simulation has shown that this approach can result in "reasonable" negotiations in which players will try to use offers and counter-offers to gain knowledge of the other player's constraints.

The paper is organized in the following manner. Section 2 describes the negotiating environment. Section 3 shows an example of how the algorithm reaches an ultimatum. Section 4 illustrates how the additional assumptions about learning resulted in much more "reasonable" negotiations.

2 Negotiating Environment

The problem identified in the preceding section arises because of the particular rules of the game being played. Section 2.1 summarizes those rules [4]. The heuristics-free resolution of that problem depends on the negotiating strategy is summarized in Section 2.2.

2.1 Rules of Engagement

We assume that the goal of the negotiation is to fill in the blanks in a *contract template* provided by the *marketplace* in which we're negotiating. A contract template consist of a set of *sections*. Each section defines a specific aspect of the contract, such as terms of payment or product description. The description in a section is specified in a *vocabulary*, which consists of a collection of *attributes*. Each attribute is a name-value pair and has a number of other properties, such as multiplicity and matching rule [4].

The negotiating parties, two in the examples studied, take turns exchanging offers. An offer consists of values, numeric ranges or sets of values, for a subset of the attributes included in the previous offer. A legal counteroffer must narrow the range of potential deals by eliminating at least one attribute value and/or narrowing at least one numeric range. Once an attribute has appeared with a single value in both an offer and counteroffer, it is *settled* and may not be reintroduced into the negotiation. A binding contract is formed when all the attributes are settled. Either party can declare the negotiation *failed* at any time before an agreement is reached.

Figure 1 illustrates the process. The axes represent the payoffs, of various deals to the negotiators Sally Seller and Bob Buyer. Bob wants the deal highest in the graph; Sally, the one farthest to the right. Points on the segment of the boundary between these two points are Pareto optimal.

Every possible deal reachable from the initial offer is contained in the outer boundary, but not all points in the interior represent deals. The protocol specifies that a legal counteroffer must narrow the range. We see from the figure that the first counteroffer does not preclude reaching an efficient deal, but that the second one does. The final deal reached is quite far from the efficient frontier. Both players would benefit if either or both used a better strategy.

2.2 Game Tree Strategy

The protocol defined in Section 2.1 constitutes a game that ends in a finite number of moves. If this game were chess, we'd build a game tree to find the best move [2]. We use the same ideas for negotiation, but there are some complications. In chess, we know that the other player wants to win but will settle for a draw. In a negotiation, we have only an estimate of what the other party wants. In addition, chess pieces can only move an integer number of squares, but negotiations can be over continuous amounts, such as liquid measures, or effectively continuous quantities, such as price.

The input to the tree strategy code is the current offer, a measure of time since the start of the negotiation, and the set of players. Each player has a utility function, known for the player using the strategy and estimated for the other player. These utility functions [4] are sufficiently complex that we treat them as black boxes; put in a potential contract and get out a payoff. We make no assumptions of linearity or even continuity. In addition, each player is identified by a strategy and a set of parameters that denote how to handle the continuous attribute values.

We identify the root node of the game tree with the current position and generate child nodes representing every legal counteroffer. Any counteroffer involving any combination of attributes with continuous values is marked as such. If a child node and its parent are both marked in this way, we end the

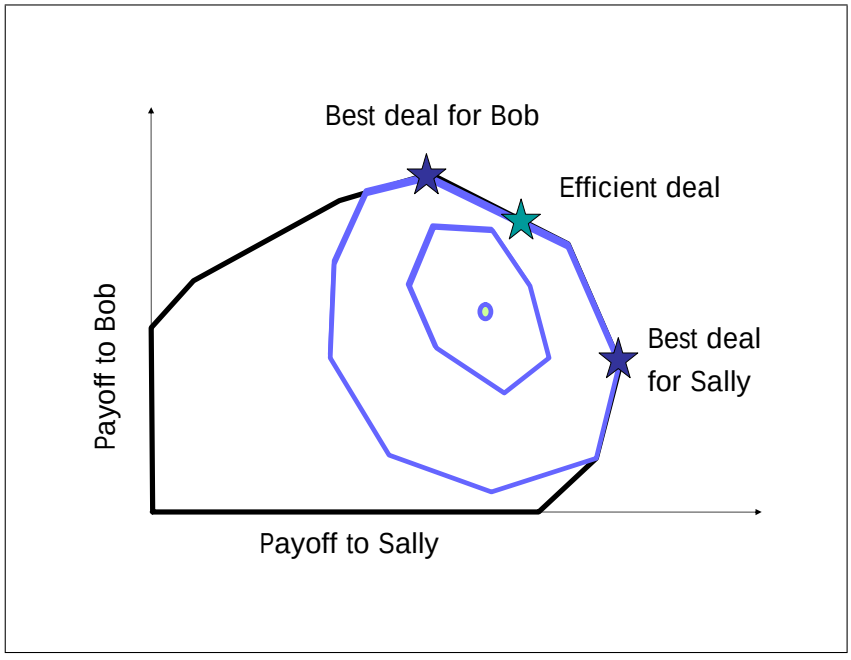

Figure 1: Narrowing the deal space.

expansion along that branch [3].

Once the tree is expanded, we ask the root node to evaluate itself. To do so, it asks each of its children to evaluate themselves. If one of the corresponding moves involves a change to a range-valued attribute, the node uses some optimization algorithm. It asks its children to evaluate themselves for one or more trial values of the attribute and uses the resulting payoffs to select an optimal value to represent the offer. On reaching a node representing a complete offer, a *leaf*, the node computes its payoff using the players' utility functions. The payoffs of a node's children are used to compute the node's payoff. These payoffs propagate to the children of the root node, which picks the one with the highest payoff as the recommended move.

3 Ultimatum Conundrum

We'll illustrate the problem with a simple example. We assume Bob wants to buy a pair of shoes from Sally. The contract template has three sections, *shoe* to describe the product, *delivery* to describe delivery options, and *payment* for payment terms. The shoe vocabulary has two attributes *style* and *color*; the delivery vocabulary, only the delivery delay, with a delay of 0 denoting that Bob will take the merchandise with him. The payment section has attributes for price and payment method. The tree strategy treats the two range-valued attributes, delay and price, as continuous.

Sally's utility function is

$$V_{Sally} = -t/5+b_w+b_l+b_{bl}+b_{br}+3(2b_{ca}+b_{cr})+2(b_{[0,0]}+b_{[1,\infty]})+H(P-100)/100, \tag{1}$$

where t is negotiation time , $b_x = 1$ if attribute value x is in the complete deal and 0 otherwise. Here w and l denote wingtips and loafers, respectively; bl and br, black and brown; ca and cr, cash and credit; $[a, b]$ delivery delays between a and b. The function $H(x) = x$ if $x \geq 0$ and is $-\infty$, otherwise. In this notation the Bob's utility is

$$\begin{aligned} V_{Bob} = \quad & -t/10 + 3[(3 - 2t/5)b_w b_{bl} + (2 - t/10)b_l b_{br}] + b_{ca} b_{[0,0]} + b_{cr} b_{[1,\infty]} + \\ & 2(b_w + b_l) + b_{bl} + b_{br} + H(300 - P)/25 \end{aligned} \tag{2}$$

In addition, each party has a set of parameters describing how range-valued attributes will be handled. These parameters specify the player's first and final offers, as well as the minimum and maximum acceptable concessions.

	Price		Delay	
Negotiator	Bob	Sally	Bob	Sally
First	50	100	3	2
Final	250	300	0	0
MinInc	53	-57	-2	-1
MaxInc	151	-133	-3	-10

Bob's strategy for range-valued attributes is to make the minimum concession. Sally's strategy is more complex. She tries the maximum increment from both ends. Although it seems strange for the seller to raise the buyer's price offer, nothing in the protocol prevents it.

Bob makes the first offer, namely,

shoe	color	['black', 'brown']
shoe	style	['wingtip', 'loafer']
delivery	delay	[0, 1]
payment	price	[150.0, 250.0]
payment	method	['check', 'cash', 'credit']

The first test of the tree strategy is based on the assumption that the user of the strategy, Sally, knows Bob's utility function. The resulting recommendation

shoe	color	['black']
shoe	style	['wingtip']
delivery	delay	[0, 0]
payment	price	[250.0, 250.0]
payment	method	['cash']

is the ultimatum that has the highest utility for Sally, 13.1, that also as positive utility, 12.4, for Bob. This deal corresponds to the point marked "Efficient Deal" in Figure 1. At first, we found this recommendation surprising. After some thought, though, we recognized that the rules used are equivalent to the bargaining game [6], which has an ultimatum as its Nash equilibrium.

We know intuitively that making an ultimatum is not always the best strategy. Such an offer is particularly bad when there is substantial uncertainty about the preference of one's opponent or his assessment about you. In this case, leaving room for negotiation may allow one to obtain information about this uncertainty.

The original tree strategy indeed ignores the issue of uncertain and potential information gains from negotiation. If one does not update one's belief, an ultimatum is optimal because there is nothing to gain from waiting. The problem is how to modify the strategy so that we get more "reasonable" offers. We could play many times, a number of times unknown to the players, or a number of times determined at random during the play. Game theory shows that the Nash equilibrium for all these variants is still an ultimatum [1].

One unrealistic aspect of the first test is that we assume Sally knows Bob's utility function. Hence, we modified the test so that Sally only had an estimate, expressed in Equation 3,

$$\begin{aligned} V_{buyer} = \quad & -t(1 \pm 0.1)/10 + 3[(3 - 2t/5 \pm 0.1)b_w b_{bl} + (2 - t/10 \pm 0.1)b_l b_{br}] + \\ & (1 \pm 0.2)[b_{ca} b_{[0,0]} + b_{cr} b_{[1,\infty]}] + (2 \pm 0.2)(b_w + b_l) + \\ & (1 \pm 0.2)(b_{bl} + b_{br}) + (H(300 - P) \pm 10.0)/25, \end{aligned} \tag{3}$$

where we are assuming the estimate is represented as the mean and standard deviation of the contributions of the individual terms. The tree strategy recommended an ultimatum that Bob was *likely* to accept, the same deal as before in this case.

In retrospect it is clear what is happening. Both players' utility functions include a time penalty that represents the cost of the negotiation. This cost can be either the direct cost of the resources consumed or the opportunity cost of the delay in reaching a deal. A deal has a particular value to Sally, and she has a mean and standard deviation of her estimate of the deal to Bob. However, there is nothing in these utility functions that accounts for the path used to reach a deal. Hence, a deal reached early is always worth more than the same deal reached late. The earliest way to reach a particular deal is to make an ultimatum. Even without a time penalty, the players will be indifferent to a particular deal reached earlier rather than later, making an ultimatum offer the equilibrium strategy.

4 Resolution

Clearly, the tree strategy won't make a good business tool if it recommends an ultimatum. The problem, then, is to identify the factor in the strategy or the rules that makes an ultimatum the Nash equilibrium. One approach is to use a heuristic of some sort. We could penalize offers the more they narrow the deal space. Alternatively, we could increase the computed value of nodes deep in the tree. These solutions are unsatisfying because the heuristics will most likely need tuning for every situation.

There is a piece of information we've implicitly assumed is available, the other player's constraints. Sally may know quite accurately what Bob will pay for loafers. However, she may not know that Bob is shopping for formal attire and that only wingtips will meet his needs. When we include this uncertainty, the tree strategy makes what appear to be reasonable offers.

We can include uncertainty in the other player's constraints without introducing any free parameters. Consider an offer by Sally representing an *ultimatum minus 1*, an offer with only two legal counteroffers, each representing an ultimatum. Clearly, this offer is at least as acceptable to Bob as either of the possible outcomes. Further, if Sally just made this offer, the probability that it is acceptable to her is 100%. Hence, the probability that the offer is acceptable to both Bob and Sally is

$$P = 1 - (1 - P_1)(1 - P_2) \tag{4}$$

where P_1 and P_2 are the probabilities that the first deal and second deals, respectively, are acceptable to Bob. Hence, the expected payoff of this offer to Bob is $u_B(P_j)P$, where j is 1 or 2 depending on which deal Bob prefers.[1]

[1] Other factors have been omitted for clarity.[3]

When we include the uncertainty in the other player's constraints in this way, the tree strategy makes reasonable counteroffers. Our test used as estimates

Term	Probability
$t - t_d$	0.3
$b_w b_{bl} \vee b_l b_{br}$	0.2
$b_{[0,0]} b_{ca} \vee b_{[1,\infty]} b_{cr}$	0.5
b_w	0.8
b_l	0.8
b_{bl}	0.8
b_{br}	0.8
$price < 300$	0.9

where the second column is the probability that the deal will be rejected if that term does not appear in the offer. In the case of the time, the probability represents the likelihood that the current time misses the deadline, t_d. In the case of price, the probability represents the likelihood that a price exceeding the threshold will be rejected.

In our example, Sally's first counteroffer is

shoe	color	['black']
shoe	style	['wingtip']
delivery	delay	[0, 0]
payment	price	[250.0, 250.0]
payment	method	['check', 'cash', 'credit']

which she estimates has a 91% probability of being acceptable to Bob. Other possible offers have the same or higher predicted payoff to Sally, but the probability that Bob will accept them is lower.

Bob uses a strategy that makes the smallest possible change, so he eliminates paying by check. Sally's last move is forced by the protocol; she selects cash as the payment method. The final deal is the same one reached previously, but in four steps instead of two. Hence the payoff to Sally is 12.7, instead of 13.1. Notice that we now have a quantification of the cost of Sally's lack of knowledge of Bob's constraints; she could get 0.4 additional utility with full knowledge of what Bob wants. Sally's estimate of the payoff to Bob is 9.8 ± 1.5, with a 75% probability that the deal is acceptable to him.

5 Summary

We have developed a software algorithm to play a specific bilateral offer-counteroffer bargaining game. This game has one special feature differing from the traditional version. In our game each offer is a subset of the offer space, not a point in that space. Furthermore, each successive offer is restricted to be a subset of the previous offer. Whenever a player offers a single point in the offer space, it is an ultimatum. The other side has only two choices: accept it or

reject it, which leaves the players with no deal. This process mirrors many real world situations [5].

The software algorithm was adapted from tree-search algorithms found in chess programs. Experience has shown that this approach performs better than other schemes, such as expert systems or neural networks [2]. Of course, the adaptation involves estimates of the opponent's utility and constraints as well as the need to handle continuous moves, which means that other approaches may reach better deals.

If we assume the utility functions of both players are common knowledge and that there is a time cost to negotiation, then this algorithm will correctly arrive at an ultimatum in its first move. However, if there is uncertainty about whether the other side will accept the ultimatum, then the algorithm will propose non-ultimatum offers to reduce the chances of rejection. This effect can also be interpreted as an implicit form of learning. With this addition the algorithm behaves in a much more "reasonable" manner.

Serendipity played an important role in determining that uncertainty in the other party's constraints was key to producing reasonable offers. Since the size of the tree grows exponentially with the number of terms in the negotiation, most tree strategies truncate the tree and use estimates of the values of non-leaf nodes. Indeed, the software we developed has code to compute these estimates. Fortunately, we chose to debug the program first on examples small enough to build the complete tree. Had we not made this choice, we would not have seen that the strategy always recommends an ultimatum.

References

[1] K. Binmore. *Fun and Games: A Text on Game Theory.* D. C. Heath and Company, Lexington, MA, 1992.

[2] F.-H. Hsu. *Beyond Deep Blue.* Princeton Univ. Press, Princeton, NJ, 2002.

[3] A. H. Karp. A game tree strategy for automated negotiation. Technical Report HPL-2003-154, Hewlett-Packard Laboratories, 2003. http://www.hpl.hp.com/techreports/2003/HPL-2003-154.html.

[4] A. H. Karp. Rules of engagement for web service negotiation. Technical Report HPL-2003-152, Hewlett-Packard Laboratories, 2003. http://www.hpl.hp.com/techreports/2003/HPL-2003-152.html.

[5] H. Raiffa. *The Art and Science of Negotiation.* Harvard University Press, Cambridge, Mass., 1982.

[6] A. Rubinstein. Perfect equilibrium in a bargaining model. *Econometrica*, 50(1):99–109, 1982.