

Algorithms for Automated Negotiations and Their Applications in Information Privacy

Haifei Li, David Ahn

Department of Computer Science, Nyack College, USA

{lih, ahnd}@nyack.edu

Patrick C. K. Hung

Department of Computer Science, Hong Kong University of Science and Technology, Hong Kong

cschk@cs.ust.hk

Abstract

Automated negotiations have been an active research topic for many years. Most of the research work on this area focuses either on the abstract and theoretical models or on the system architectures for standalone negotiation applications. There is little work on identifying and studying practical algorithms for automated negotiations. In this paper, two algorithms have been proposed and their innovative applications have been discussed. The first algorithm guarantees that negotiation results are Pareto optimal solutions. The second algorithm guarantees that an agreement can be agreed upon after a certain number of proposal/counterproposal exchanges. These two algorithms can be used in the two-phase model for the automated negotiation process. In addition to these algorithms, their applications in information privacy negotiation have been described. In the information privacy management domain for service industries, it is critical for service requestors to only reveal the absolute necessary private information to the service providers. Traditionally, service requestors usually give whatever private information service providers have asked for. The grave consequence is that service providers may misuse the private information provided by service requestors, even though the service providers may have promised not to reveal it. Service requestor should have an efficient way to negotiate with service providers about the appropriate private information to be revealed. Algorithms described in this paper can facilitate the privacy negotiation process. In order to show the concept of negotiation in information privacy, credit card information has been used to illustrate the application of algorithms.

1. Introduction

Negotiation has been widely studied in various disciplines such as diplomacy, psychology, sociology, law, business administration, and computer science [8,

10, 12, 14, 18]. In a stereotype of negotiation, two or more negotiators, usually wearing formal business suits, bargain face-to-face for topics of high stakes. The topics vary from corporate purchases, corporate mergers and acquisitions, and labor/management disputes to international relations. Emotional outbursts, bluffs, and ultimatums are considered norms. The negotiation outcomes largely depend on negotiators' negotiation skills. Negotiation skills usually include "soft" skills, such as communication, collaboration, persuasion, personality, aspiration, and ambition. At a first glance, it seems that information technology does not have a place in the area of negotiation.

In recent years, there are many researches on e-business negotiation, but there is little work on negotiation for information privacy [16]. Part of the reason is the fixed mindset for participants in privacy issues. When participants in handling private information are get involved, they are usually think in the way of "black/white" or "0/1". For example, if a company asks for the Visa credit card information and the customer is not willing to give the Visa credit card information, the deal between the company and the customer might fall through. However, there are spaces for manipulation. For example, if the customer is not willing to give the information about the Visa credit card information (i.e., credit card number, expiration date, customer address, home phone number, amount limitation), the company can ask for the Mastercard information. In some cases, the customer may be willing to give the Mastercard information. In this particular example, a deal can be made because both participants are able to "fine-tune" their privacy preferences.

The intended contribution of the work is as follows. First, we divide the whole negotiation process into two phases: pre-negotiation phase and bargaining phase. The division makes the whole negotiation process more manageable. The whole negotiation process can be fully automated, as long as certain assumptions have been met.

Second, we have proposed an algorithm to eliminate the non-Pareto optimal solution for the pre-negotiation phase. Third, we have proposed an algorithm to conduct automated negotiation on behalf of human negotiators. The algorithm guarantees that the negotiation process converges in a finite amount of time. However, the actual time spent on negotiation depends on the parameters set by these negotiators and the dynamics of the negotiations. Fourth, we have presented an innovative application of these algorithms in information privacy. To our best knowledge, there is no prior work on applying automated negotiation algorithms to information privacy.

2. Related Work

A Negotiation Support System (NSS) [6, 7, 15] is a computer system that assists human negotiators in making decisions in a negotiation process. Lim proposed a theoretical model of NSSs in [9]. The paper divides an NSS into two major components: decision aid component (i.e., DSS) and an electronic communications component. Due to the information processing capability of the decision aid component, solutions with NSS are better than those without NSS. Compared with a verbal communication channel, an electronic communications channel is more “task-oriented” than “social-oriented.” Therefore, the time to settlement is reduced and the satisfaction with the system is higher. Like decision support systems (DSSs), the focus of NSSs is still on “support,” not on “making decisions” by computers.

Holsapple, Lai and Whinton proposed a negotiation model in [6, 7]. Kersten and Szpakowicz proposed another negotiation model in [13]. These models are comprehensive models from a theoretical perspective. However, it is not clear how these models can be used for the implementation of an automated negotiation system. For the automation purpose we have to restrict the number of elements and well define their relationships.

Kasbah [2] is a Web-based multi-agent marketplace where users create buying and selling agents to help exchange goods. A user wanting to buy or sell goods can create an agent, initialize it with a particular negotiation strategy, and send it off into the marketplace. Kasbah’s agents proactively seek out potential buyers or sellers and negotiate with them on behalf of their owners. The solution is not practical. As pointed out by the developers, agents sometime make apparently “stupid” decisions.

Sierra, Faratin, and Jennings present a formal model of negotiation between autonomous agents in [17]. The model defines a range of strategies and tactics that agents can employ to generate initial offers, to evaluate proposals, and to generate counterproposals. The intelligence of a negotiation agent is manifested by the use of an evaluation function for each negotiation attribute (a simple aggregation function), and a simple hard-coded monotonic concession negotiation strategy.

The field of machine learning attempts to let software agents learn how to negotiate by themselves, rather than exhaustively implementing human negotiation knowledge into software agents. In [19], Zeng and Sycara present Bazaar, an experimental system for updating negotiation offers between two intelligent agents during a bilateral negotiation. The paper explicitly models negotiations as a sequential decision-making task and uses Bayesian probability as the underlying learning mechanism. The authors present an example by using price as the issue of negotiation.

Another machine learning approach is to use genetic algorithms, based on the principle of Darwinian evolution. In the context of negotiations, it works as follows: each of the software agents begins with a population of various, randomly generated (and not necessarily good) negotiation strategies. It then employs its strategies against the strategies of the other agents in a round of bargaining, which takes place under specific predetermined rules and payoffs. The size of the initial population, the number of generations, the crossover rates, and the mutation rate are parameters of the algorithm. In principle, after many trials the negotiation agent is able to derive a good strategy; however, the actual result depends on many elements: e.g., the number of training trials, the algorithm parameter configuration, and the quality of the negotiation strategy of the training opponent. For instance, to achieve a good strategy, the number of trials varies from about 20 generations [11] to 4000 generations [5] and all runs must be made against opponent(s) whose strategies are as realistic as possible.

The machine learning approach to negotiation is promising in theory since it is able to derive negotiation strategies based on learning algorithms. However, it is not clear how effective the approach will be when it is used to implement a multi-issue, bargaining type negotiation system. The learning approach used in Bazaar only deals with a single issue: price.

In summary, current research on negotiations has yielded fruitful results. However, most of the work has suffered from one important shortcoming: lack of real

world scenarios. In this work, we present two algorithms and have demonstrated the usefulness of these algorithms in an active research topic: information privacy.

3. Two-phase Model

We assume that there are two negotiators N_1 and N_2 . They want to negotiate on issues $I_1, I_2, \dots, I_k, \dots, I_m$, where m is the number of issues to be negotiated. It is assumed that both N_1 and N_2 publish their issues as web services, most likely in a UDDI repository. After these issues have been published, they can be downloaded and digested by the potential collaborators (negotiators in our case) all around the world. We assume that N_1 and N_2 use the same ontology for representing these issues. The problem of how to make N_1 and N_2 understand each other is beyond the scope of this paper. Interested readers are encouraged to consult related literature [4]. The nature of issues depends on the problems to be negotiated. They can vary from pricing to dispute methods to delivery destination to payment methods. It is assumed that these issues are independent so they can be negotiated separately. In the context of this paper, the negotiation process consists of repeated applications of algorithms on different issues. Once negotiated alternatives for all the issues have been identified, an agreement has been reached between two negotiators.

The algorithm is divided into two phases: pre-negotiation phase and bargaining phase. During the pre-negotiation phase, two negotiators work together to identify non-Pareto-optimal alternatives and delete them from the bargaining phase. During the bargaining phase, one negotiator continuously offers the alternative (may the same offer for many times) to the other party until a match has been found. Because of the work done in the pre-negotiation phase, the algorithm is guaranteed to terminate in a finite amount of time.

4. Algorithm to Find the Pareto-optimal Solutions

Bilateral negotiation involving two negotiators is considered in the paper. In order to have a clear presentation of the idea, it is assumed that the negotiation proposal has a practical and simple model. Although negotiation issues vary from one negotiation to another, it is assumed that during a specific negotiation process, the number and nature of the issues are fixed. The messages exchanged between two negotiators include a set of predetermined issues with their values. Therefore, a proposal is a set of attribute / value pairs. For example, in the negotiation of digital camera purchase, we have

the following issues to be negotiated: price, quantity, delivery_day, resolution. One proposal may look like: (price = 700.00USD, quantity = 500, delivery_day = 3 weeks, resolution = 1.3megapixel).

A proposal becomes a solution if a proposal proposed by one negotiator is accepted by the other negotiator. There are many possible solutions, and some are Pareto optimal and some are not. Simply speaking, Pareto optimal solutions are solutions that are not “dominated” by other solutions. It is necessary to explain what exactly “domination” means. Our discussion limits to two-part situation where bilateral negotiation fits in. Pareto optimal solutions deal with two negotiators that have opposite interests. A (possible) solution is an assignment of values to issues involved. Apparently, there are many possible solutions. If a 2D diagram is used to show the utilities to both negotiators, the solutions correspond to points in the 2D space. The term “utility” is used in a very broad sense. It could be preferences, benefits or gains to a negotiator. Figure 1 shows the two dimensions. The points in the upper right space correspond to the possible solutions. The Pareto frontier is made up of points that cannot expand up right any further. Solution S1 “dominates” solution S2 if S1 would improve the utilities of both parties or improve the utility of one party and does not decrease the utility of the other party. It is clear from figure 1 that C “dominates” A and D “dominates” B. E “dominates” both A and B. However, A does not “dominate” B and vice versa.

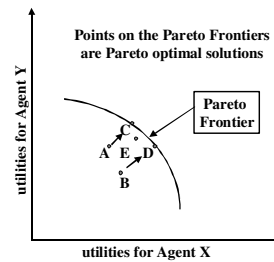


Figure 1. Pareto optimal solutions

Pareto optimal solutions are useful to bilateral negotiation. It is clear that the final agreement should be Pareto optimal. Otherwise, a Pareto optimal solution that dominates the final agreement should be preferable to both negotiators. However, it is important to point out the Pareto optimal solutions are usually a set that contains multiple solutions. It is up to the negotiators to choose which one to be the final agreement. Both C and D are Pareto optimal solutions in Figure 1.

Messages exchanged between negotiators are usually related to a complete proposal. That is to say, when proposal / counter-proposal are exchanged between two negotiators, all issues must have values. Delivery schedule is a good example that can be used to illustrate the algorithm. Modern enterprises make extensive use of ERP and other enterprise information systems for production planning and execution. It is quite possible that the enterprise's preference over delivery is not as early as possible (for buyer), or as late as possible (for supplier). For example, In JIT (Just-In-Time) manufacturing, the raw materials must arrive at the specified time, therefore, both the early arrival time and late arrival time have preferences less than that for the optimal arrival time. The consequence is that the preference over the delivery schedule is NOT a monotonically increasing or decreasing function over the time. Figure 2 shows the relationship between buyer's and seller's delivery schedule. We assume that buyer and seller are two negotiators for this example.

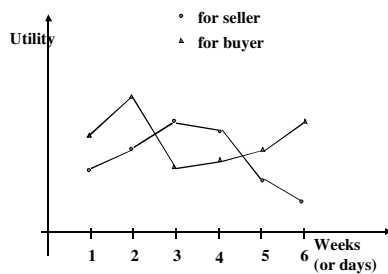


Figure 2. Preferences of Delivery Schedule for Both Seller and Buyer

The preference from the buyer's side is 2, 6, 1, 5, 4, and 3. The preference from the seller's side is 3, 4, 2, 1, 5, and 6. There is an algorithm to find the Pareto optimal solutions to the problem. The following is the description of the algorithm:

Input: inputListN1 and inputListN2 for the negotiator N1 and the negotiator N2's preferences. It does not matter whether the negotiator N1 is for the buyer or the seller. The same is true for the negotiator N2. List elements are arranged from the most preferable to the least preferable.

Output: outputListN1 for the negotiator N1 and outputListN2 for the negotiator N2.

Steps:

1. Initialize both outputListN1 and outputListN2 to empty list,
2. Set pointerN1 to the first element of inputListN1,

3. Select one element from inputListN1, call the element eleN1, let pointerN1 point to the next element in inputListN1,
4. Find the element in inputListN2 that has the same value as eleN1 by scanning inputListN2 starting from the first element, call the element found in inputListN2 eleN2,
5. Get the ancestor set ancestorSetN1 of eleN1 in inputListN1. This can be done by traversing from the first element of inputListN1 down to the element before eleN1,
6. Get the ancestor set ancestorSetN2 of eleN2 in inputListN2. Use the same approach as in step(5),
7. Check whether the intersection of ancestorSetN1 and ancestorSetN2 is empty. If the intersection is empty, insert eleN1 to the end of outputListN1. Otherwise eleN1 (also eleN2) is dominated by other elements, so it will not be added to outputListN1,
8. Check whether pointerN1 points to null. If it is, go to step 9, otherwise go back to step 3,
9. Copy outputListN1 to outputListN2, reverse elements in outputListN2,
10. Output outputListN1 and outputListN2.

It does not matter whether the lists are represented as either array-based lists or linked lists. However, data structure "set" is not good for the representation since the relative order among elements, which is very important to the algorithm, is not captured. OutputListN1 and outputListN2 give the orders that both the negotiator N1 and the negotiator N2 should follow when it is time for one side to make a concession. However, Pareto-optimal solution list does not tell a negotiator whether it is good to make the concession or not. The decision to make such a concession is solely up to the negotiator.

In the example, the Pareto optimal solution is (2, 4, 3) for the buyer. That means if buyer is negotiating on this issue, he / she will gradually relax on the order of 2, 4, and 3. On the other hand, the seller will gradually relax on the order of 3, 4 and 2, exactly the opposite order for the buyer. Other solutions are "dominated" by others. For example, 1 is "dominated" by 2. If we move the solution from 1 to 2, both seller and buyer are better off.

Time complexity analysis is not difficult to figure out. Steps from 3 to 8 forms a loop. Step 7 takes $O(n * n)$ in the worst scenario. Therefore the algorithm takes $O(n * n * n)$ at most, where n is the number of possible values for the issue.

5. Algorithm to Conduct Automated Negotiations

We have assumed that the following set of alternatives for the issue I_k has been identified: $A_1, A_2, A_3, \dots, A_j$. It is natural for a negotiator to have preferences over these alternatives. For example, there may be many alternatives for delivery destinations for an international trade. We assume that these alternatives are Portland, San Francisco, and Los Angeles. If the manufacturing site for company C_1 is located in Portland, it is understandable that the company prefers the delivery to Portland to the delivery to San Francisco or Los Angeles. On the other hand, if the trading partner (Company C_2) for the company is located in Mexico, Company C_2 may prefer to deliver goods to Los Angeles since it is much closer than Portland.

During the negotiation process, the negotiator wants to get the most preferable alternative. Unfortunately, the negotiator usually cannot get it because the negotiation partner may prefer other alternatives. In the above example, company C_1 prefers Portland while Company C_2 prefers Los Angeles. That is to say, two negotiators have conflicts and need to resolve them. These two negotiators can use the algorithm illustrated in the previous section to get rid of non-Pareto optimal solutions. The problem is: even though non-Pareto optimal solutions have gone, the Pareto optimal solutions are not unique.

Fortunately, after the pre-negotiation phase, only Pareto-optimal alternatives (i.e., solutions) are left. Negotiator N_1 and N_2 have completely opposite positions on these alternatives. For example, if N_1 has the preference list of $A_1, A_2, A_3, \dots, A_j$, N_2 must have the preference list of $A_j, \dots, A_3, A_2, A_1$.

N_1 and N_2 independently assign counting numbers for all alternatives as C_{nm} . The counting number is defined as the number of times that a negotiator wants to offer this particular alternative persistently to his/her opponents before moving to the next less preferable alternative.

Each C_{nm} must satisfy the following constraints:

- (1) $1 \leq n \leq 2$, n represents the identification for negotiators. In our example, it is either N_1 or N_2 .
- (2) $1 \leq m \leq j$, m represents the m th alternative for the issue I_k , and j is the total number of Pareto optimal alternatives for a particular issue.
- (3) C_{nm} is an integer number and $C_{nm} \geq 1$, meaning that negotiator n must offer the m th alternative at least once.

In summary, we have got two lists for negotiators N_1 and N_2 respectively:

List 1: $C_{11}, C_{12}, \dots, C_{1(j-1)}, C_{1j}$, is the list of counting numbers assigned by the negotiator N_1 , ordered from the most preferable to the least preferable.

List 2: $C_{2j}, C_{2(j-1)}, \dots, C_{22}, C_{21}$, is the list of counting numbers assigned by the negotiator N_2 , ordered from the most preferable to the least preferable.

Please note that the second index y for C_{xy} is completely opposite for N_1 and N_2 . While y goes from 1 to 2, up to $(j-1)$ and j for N_1 , it goes from j to $(j-1)$, down to 2 and to 1 for N_2 .

The bargaining phase works as follows:

Each negotiator (N_1 and N_2) maintains a pointer to the alternative to be offered and a running counter for that particular alternative. The initial value for the counter is the counting number C_{xy} assigned by negotiator x . The value for the running counter decreases by one after a proposal has been sent to the other side. When the value of the running counter reaches zero, the pointer points to the next alternative in the preference list, and the value of the running counter is set to the next C_{xy} value.

At first, N_1 proposes the first alternative (the most preferable one for N_1 and the least preferable one for N_2) and sends it to N_2 . N_1 would decrease its running counter by one after sending the proposal to N_2 . When N_2 receives the proposal, it checks whether the value (i.e., the name of the alternative) in the counterproposal matches the alternative that its pointer points to. If so, the negotiator accepts the counterproposal and finishes the bargaining phase. If not, N_2 proposes his/her first alternative (the most preferable for N_2 and the least preferable for N_1) and sends it to N_1 . N_2 's running counter decrease by one. When N_1 receives the counterproposal from N_2 , N_1 goes through the same checking process. This bargaining phase guarantees to terminate (with acceptance) after a finite amount of exchange of proposal/counterproposal.

The algorithm has the following properties:

- (1) Guaranteed Termination

We use a diagram to mathematically prove this important property.

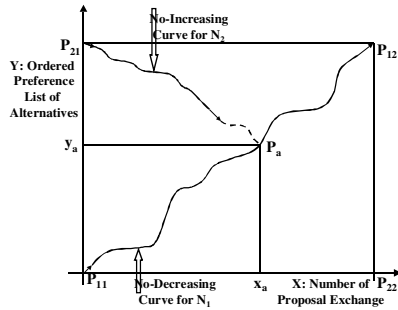


Figure 3. Diagram to Show Guaranteed Termination

Figure 3 is used to mathematically prove that the algorithm is guaranteed to terminate. X-Y coordinate system has been used to show two related dimensions. The X coordinates represent the number of proposal exchange. The Y coordinates represent the ordered preference list of alternatives. As we have shown before, Negotiator N_1 and N_2 have the exact opposite position on the preferences of alternatives. We reasonably assume that the number of alternatives is finite. Therefore, it is appropriate for us to draw a dimension Y and these two negotiators offer alternatives from opposite directions. In the diagram, Negotiator N_1 goes up from the point P_{11} , but Negotiator N_2 goes down from the point P_{21} .

Since each negotiator assigns a finite Counting Number to all the alternatives, the number of proposal exchange is also finite. In the worst-case scenario, negotiators would continuously make concessions until they have offered the counting number of times for the least preferable alternatives. Therefore, Negotiator N_1 would follow the direction from P_{11} to P_{12} , and Negotiator N_2 would follow the direction from P_{21} to P_{22} .

Without losing of generality, it is assumed that the Negotiator N_1 would go from P_{11} to P_{12} . The proof is similar if the Negotiator N_2 is used. Since Negotiator N_1 must follow his/her order to offer the alternatives, the curve from P_{11} to P_{12} is non-decreasing. When the curve is not decreasing, the curve would be bounded by the rectangle formed by the four points P_{11} , P_{22} , P_{12} and P_{21} . The entity formed by the three points P_{11} , P_{12} and P_{21} is closed. P_{22} is outside of the entity. This can be easily understood by the drawing of the curve in Figure 3.

When the Negotiator N_2 offers his/her proposals, N_2 would go from P_{21} to P_{22} . Similar to the negotiator N_1 , the negotiator N_2 must follow his/her order to offer the alternatives. Therefore, the curve from P_{21} to P_{22} is non-increasing. Two arrows on the curve $P_{21}P_{22}$ have been used to illustrate direction and it goes to the lower right part of the coordinate system. As the curve moves that

way, it must initially move inside the closed entity formed by the three points P_{11} , P_{12} and P_{21} . Since the final destination of the curve $P_{21}P_{22}$ is P_{22} , which is the point outside of the closed entity, the curve $P_{11}P_{12}$ and $P_{21}P_{22}$ must intersect at a point inside the rectangle.

The point is labeled as P_a (a means accept). This point is the acceptance point and it can be reached in a finite number of proposal exchanges. According to the algorithm, once this point has been reached, it would terminate with acceptance. Therefore, our algorithm would terminate with acceptance within a finite amount of proposal exchange (i.e., in finite time).

As a special case, the Negotiator N_2 may have assigned a very large counting number to the most preferable alternatives. Therefore, the curve made by N_2 will be a straight line parallel to the X coordinate. As a result, the point P_a would be the same as P_{12} .

(2) Symmetry

In this algorithm, two negotiators N_1 and N_2 are symmetrical. We did not make any assumption about which negotiator is the primary party. This assembles the real world situation.

(3) Simplicity

It is obvious that our algorithm is simple and practical to use. It can be used in many different scenarios. Privacy negotiation is just one application of this algorithm. To our best knowledge, this is the first algorithm that has used negotiation in the information privacy management.

6. Applications in Information Privacy Negotiation

The world has been rapidly moving into a service-oriented economy. Service industries has exceeded manufacturing industries in terms of revenue for a long time. In order for an individual or a company to better serve their customers, it is necessary for customers to reveal private information to the service providers. Information privacy on the Internet is a top concern for business, government, media and the public. Opinion surveys consistently show that privacy concerns are a leading impediment to the further growth of e-commerce. Initial efforts by web sites to publicly disclose their privacy policies have had some impact. But these policies are often difficult for users to locate and understand, too lengthy for users to read, and changed frequently without notice.

Privacy is defined as a state or condition of limited access to a person. Information privacy relates to an individual's right to determine how, when, and to what extent information about the self will be released to another person or to an organization. In recent years, there are many industrial standards and research prototypes related to information privacy. P3P (Platform for Privacy Preferences Project) [3] initiated by W3C has been becoming the de facto standard for web sites to publish their stated preferences over privacy issues. Immediately after P3P standards have been released, there are many research prototypes built around P3P. Both Microsoft and Netscape have incorporated P3P into their browsers. AT & T has released a prototype called AT & T Privacy Bird [1] based on P3P.

Privacy issue does not exist in the interactions between web server and web browser alone. It actually exists in many other scenarios. Usually, credit cards are private information and customers may prefer to reveal as little information (about the credit card) as possible. For example, if a company accepts credit cards for the services, the company can publish its preferred types of credit cards. On the other hand, the customer, whether it is an individual or a company, may also have its preferences over releasing the types of credit cards to other parties. In the traditional way, the service provider may provide a simple match making service that mechanically check whether there is an exact match between their preferences. If an exact match is found, everyone is happy and the deal is done. However, if the exact match is not found, the current system simply determines that there is not a match, and the deal needs to be cancelled. In this paper, we argue that service provider and service requestor can actually negotiate on the preferences on private information. The key idea is to reduce the possibility of failure of service matching.

In this hypothetical scenario, the issue to be negotiated is the types of credit cards. For this particular issue, it is assumed that there are the following alternatives: Visa, DC (Dinner's Club), MC (for Master Card), AMEX (American Express). The problem the negotiators are going to be negotiated is: which credit card will be used between the service provider and the service requestor. Figure 4 shows the scenario.

Suppose that the service provider is able to publish its preferences (in the order from the most preferred to the least preferred): Visa, MC, AMEX and DC. The real order is not important here and we just want to show the concept. This is the public information and it is published in the web site. The service requestor also has its own preferences (in the order from the most preferred

to the least preferred): DC, MC AMEX, and Visa. Again, the real order is not important and we just want to show the concept. One possible explanation will be that he/she may have a large credit line on the DC card, therefore the service requestor is willing to give that credit card information. On the other hand, the service provider may prefer Visa card, possibly because compared with other card issuers, Visa card issuer charge less for its service.

For each alternative, both the service provider and the service requestor have counting number associated with it. The counting number is a positive integer that indicates the number of times that party (either service provider or service requestor) wants to offer the alternatives to the other party. For example, if the counting number is 3 for MC (on the service provider side), the service provider would like to persistently offer MC for three times before moving to the next alternatives, in this case, it is AMEX.

As we have discussed before, the negotiation process has two phases: pre-negotiation phase and bargaining phase. It is quite easy to figure out that in this example, AMEX is dominated by MC for both service provider and service requestor. Therefore, the AMEX would be gotten rid of by the first algorithm proposed in this paper. Underline has been used to indicate that AMEX has been out of consideration for the second phase.

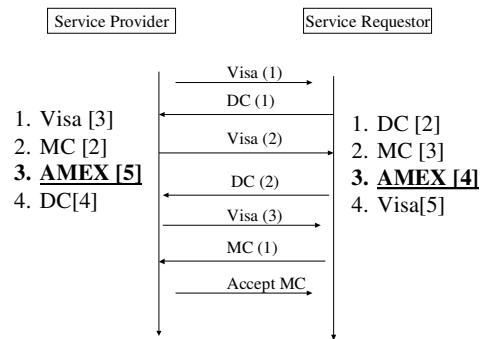


Figure 4. An Example Negotiation Process Based on Credit Card Type

At first, the service provider asks the service requestor whether he/she is willing to accept Visa card (arrow Visa(1)). In this case, Visa is the most preferable alternative for the service provider. When the service requestor checks his/her current value, he/she finds that the current value is DC (arrow DC(1)), meaning that the most preferable alternative for the service requestor is DC. Therefore, the service requestor tells the service provider that he/she accepts DC at this moment. When the service provider receives the information from the

service requestor, he/she finds that a counterproposal containing the Visa credit should be proposed (because the counting number for Visa is 3). Therefore the message Visa(2) is sent back to the service requestor. When the service requestor receives Visa(2) message, he/she checks to see what the next step will be. He/she figures out that another DC message is needed since the counting number for DC is 2. Therefore DC(2) is sent to the service requestor.

Once DC(2) has been received by the service requestor, the service requestor figure out that the value is still not what he/she wants. Since the counting number for Visa is 3, the service provider sends an Visa(3) to the service requestor. Once the service requestor receives the Visa(3) message, he/she needs to send MC(1) information to the service provider. When the service provider receives the information, he/she finds that MC is exact the next value he/she wants to offer. Therefore, the service requestor sends the acceptance message to the service requestor (arrow Accept MC). The negotiation process ends with final agreement on MC.

7. Conclusions, Limitation and Future Work

In this paper, we have presented two algorithms to conduct automated negotiation. When certain conditions hold, the algorithms are guaranteed to get the negotiated result. One advantage of algorithms is the full automation. It required neither the involvement of the negotiation parties nor the mediation of a third party. We have mathematically proved the guaranteed termination of the algorithm. We have shown two applications of the algorithm; one is in the area of delivery schedule, and the other is in the area of information privacy. Although these two areas are quite different, they could share the same conflict resolution mechanism. The mechanism requires both parties to resolve the conflict by concession. The concession by one negotiator would make the counterpart more willing to accept his/her offers. By making use of the algorithm to find Pareto-optimal solutions, it is possible to speed up a negotiation process by eliminating the time potentially spent on non-Pareto-optimal solutions.

There are strong assumptions on the computerized quantification of issues. The following potential problems have been identified: First, issues cannot be easily identified. Even if issues have been identified, there is a need to have a common understanding of the meaning of the issues. Second, alternatives cannot be easily identified. Third, there is ambiguity in identifying the preferences over the alternatives. Sometimes, negotiators

may have difficulty in justifying one alternative is definitely better than the others.

We have identified the following future work. First, we would like to actually implement the algorithms used a programming language, preferably Java. This should not be a very challenging work because of the groundwork has been finished. Second, we would like to find a way to conduct automated simulations. Third, we would like to find a way to conduct human-based simulations and compare their results with those from the automated simulations. Fourth, we would like to find more applications for these two algorithms.

8. References

- [1] AT & T, the AT & T Privacy Bird Software, available at www.privacybird.com.
- [2] A. Chavez and P. Maes, "Kasbah: An Agent Marketplace for Buying and Selling Goods," *Proceedings of the First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology*, London, UK, April 1996.
- [3] Lorrie Faith Cranor, *Web Privacy with P3P*, O'Reilly & Associates, Sebastopol, CA, 2002.
- [4] J. Davies, D. Fensel, F. van Harmelen (Editor), "Towards the Semantic Web: Ontology-Driven Knowledge Management," John Wiley & Sons, New York, NY, 2003.
- [5] G. Dworman, S. Kimbrough, and J. Laing, "On Automated Discovery of Models Using Genetic Programming in Game-Theoretic Contexts," *Journal of Management Information Systems*, vol. 12 (3), Winter 1996.
- [6] C. Holsapple, H. Lai and A. Whinston, "Analysis of Negotiation Support Systems: Roots, Progress, and Needs," *Journal of Computer Information Systems*, Vol. 35, No. 3, 1995, pp. 2-11.
- [7] C. Holsapple, H. Lai and A. Whinston, "Implications of Negotiation Theory for Research and Development of Negotiation Support Systems," *Group Decision and Negotiation*, Vol. 6, Issue 3, 1997, pp. 255-274.
- [8] D. A. Lax and J. K. Sebenius, *The Manager as Negotiator: Bargaining for Cooperation and Competitive Gain*, The Free Press, New York, NY, 1986.
- [9] L. Lim and I. Benbasat, "A Theoretical Perspective of Negotiation Support Systems," *Journal of Management Information Systems*, Vol. 9, No. 3, 1993, pp. 27-44.
- [10] G. Lo and G. E. Kersten, "Negotiation in Electronic Commerce: Integrating Negotiation Support and Software Agent Technologies," *5th Annual*

Canadian Operational Research Society Conference, Halifax, Nova Scotia, Canada, 1999.

[11] J. R. Oliver, "A Machine-Learning Approach to Automated Negotiation and Prospects for Electronic Commerce," *Journal of Management Information Systems*, Vol. 13, No. 3, 1997.

[12] C. L. Karrass, *Give and Take: The Complete Guide to Negotiating Strategies and Tactics*, HarperCollins Publishers, New York, NY, 1993.

[13] G.E. Kersten and S. Szpakowicz, "Decision Making and Decision Aiding. Defining the Process, Its Representations, and Support," *Group Decision and Negotiation*, Vol. 3, No. 2, 1994, pp. 237-261.

[14] H. Raiffa, J. Richardson, D. Metcalfe, *Negotiation Analysis: The Science and Art of Collaborative Decision Making*, The Belknap Press of Harvard University Press, Cambridge, MA, 2003.

[15] A. Rangaswamy and G. R. Shell, "Using Computers to Realize Joint Gains in Negotiations: Toward an "Electronic Bargaining Table," *Management Science*, Vol. 43, No. 8, 1997, pp. 1147-1163.

[16] F. D. Schoeman, *Philosophical Dimensions of Privacy: An Anthology*, Cambridge University Press, New York, NY, 1984.

[17] C. Sierra, P. Faratin, and N. Jennings, "A Service-Oriented Negotiation Model between Autonomous Agents," *Proceedings of 8th European Workshop on Modeling Autonomous Agents in a Multi-Agent World (MAAMAW-97)*, Ronneby, Sweden, 1997, pp. 17-35.

[18] I. Ståhl, *Bargaining Theory*, Stockholm, Sweden: The Economics Research Institute, 1972.

[19] D. Zeng and K. Sycara, "Bayesian Learning in Negotiation," *International Journal of Human Computer Systems*, Vol. 48, 1998, pp. 125-141.