Computing and Informatics, Vol. 31, 2012, 1235–1253

GENERATING PARETO-OPTIMAL OFFERS IN BILATERAL AUTOMATED NEGOTIATION WITH ONE-SIDE UNCERTAIN IMPORTANCE WEIGHTS

Hamid Jazayeriy

Faculty of Electrical and Computer Engineering
Babol Noshirvani University of Technology, Babol, Mazandaran, Iran
e-mail: jhamid@nit.ac.ir

Masrah Azmi-Murad, Nasir Sulaiman, Nur Izura Udzir

Faculty of Computer Science and Information Technology
University Putra Malaysia, 43400 Serdang, Malaysia
e-mail: {masrah, Nasir, Izura}@fstm.upm.edu.my

Communicated by Prabhat Kumar Mahanti

Abstract. Pareto efficiency is a seminal condition in the bargaining problem which leads autonomous agents to a *Nash-equilibrium.* This paper investigates the problem of the generating Pareto-optimal offers in bilateral multi-issues negotiation where an agent has incomplete information and the other one has perfect information. To this end, at first, the bilateral negotiation is modeled by *split the pie* game and *alternating-offer protocol.* Then, the properties of the Pareto-optimal offers are investigated. Finally, based on properties of the Pareto-optimal offers, an algorithmic solution for generating near-optimal offers with incomplete information is presented. The agent with incomplete information generates near-optimal offers in $O(n \log n)$. The results indicate that, in the early rounds of the negotiation, the agent with incomplete information can generate near-optimal offers, but as time passes the agent can learn its opponents preferences and generate Pareto-optimal offers. The empirical analysis also indicates that the proposed algorithm outperform the *smart random trade-offs (SRT)* algorithm.

Keywords: Bilateral negotiation, pareto-optimal offer, uncertain information, algorithm

1 INTRODUCTION

Pareto-efficiency is a seminal condition to form a *Nash-equilibrium* where self interested agents try to satisfy each other [22, 17]. In the non-cooperative multi-issues bilateral negotiation, generating Pareto-optimal offers with incomplete information is a computationally complex problem. With Pareto-optimal offer it is impossible to make one agent better off without necessarily making the other agent worse off.

There are different sources of uncertainty in bilateral negotiation such as: information about the opponent's deadline, importance weights over negotiation issues and outside options. This information is rarely available which makes automated negotiation complicated. To generate Pareto-efficient offers, an agent needs information about the opponent's importance weights over negotiation issues. These weights are used to form a greedy order (agenda) to generate offers by using sequential maximum trade-offs [10, 19, 14]. In fact, given the n negotiation issues, there are $n!$ sequences that can be used to generate offers, but only one of them makes the Pareto-optimal offer (in a special case there may be multiple sequences). In other words, if an agent has uncertain information about the opponent's importance weights then the problem of finding a Pareto-optimal offer will be computationally intractable.

The following assumptions are considered to conduct this study. It is assumed that agents bundle all the negotiation issues to generate offers. Moreover, it is assumed that agents are computationally bounded rational, meaning that they have limited time (and resources) to reach agreement.

In this study, the bilateral negotiation is modeled by *alternating offer protocol* [18] and negotiation over each single issue is like a *split the pie* game [1, 18, 2, 14]. In other words, the bilateral multi-issues negotiation is modeled by multiple *split the pie* games.

This study investigates the properties of the Pareto-optimal offers. To this end, the problem of generating Pareto-optimal offers with perfect information and the *maximum greedy trade-offs (MGT)* algorithm is considered [14]. Then, these properties are used to conduct a learning method to reveal the greedy order (agenda) and generate the Pareto-optimal offer where one agent has uncertain information and the other has perfect information.

The rest of the paper is organized as follows. Next section details related work in bilateral automated negotiation. Section 3 describes the negotiation model used in this study by introducing the negotiation protocol and some basic concepts. Then, Section 4 describes the *MGT* algorithm and the properties of the Pareto-optimal offers. It also presents an extension to *MGT* algorithm that generates near Pareto-optimal offers with one-side incomplete information. Section 5 provides an experimental analysis to evaluate the efficiency of the proposed method. Finally, Section 6 draws the conclusions and our plans for future studies.

2 RELATED WORK

Automated negotiation has received wide attention in the fields of game theory and artificial intelligence. In game theory, researchers study on negotiation models, axioms and equilibrium solutions through some rigorous assumptions. These assumptions are not necessarily realistic. On the other hand, researchers in AI community try to develop software agents that negotiate on behalf of their owners in realistic environments.

The amalgamation of game theory and AI can empower autonomous agents to make deals in e-marketplaces by finding approximate solutions for the problems that are computationally intractable. In bilateral negotiation, agents can find a computationally tractable solution if they have some information about their opponents, such as: deadline, preferences and outside options. During the negotiation process, an agent can make decisions about its aspiration-level based on the information about outside options and the opponent's deadline, while information about the opponent's importance weights is needed to find Pareto-optimal offers. Usually, agents have incomplete information about their opponent which arises uncertainty and makes automated negotiations an interesting area of research in AI field.

Bilateral negotiation is analogous to the well-known *bargaining* problem [2, 17]. This problem can be modeled by *split the pie* game [18]. Fatima et al. [10] have modeled the multi-issues bargaining problem with the *split the pie* game. They assumed that not only negotiation over each single issue is like *split the pie* game, but also the total outcome of the negotiation is like a *pie* of size 1 and each agent takes a share of the *pie*. In other words, an agent gains an amount of the *pie* if the other agent loses the same amount. That is, in their study, bilateral negotiation is a kind of *zero-sum* game. A real world *bargaining* is not necessarily a *zero-sum* game, i.e., an agent may make a trade-off, while keeping its aspiration-level unchanged, to increase the opponent's payoff. However, in our study, multi-issues bilateral negotiation is modeled by multiple *split the pie* games, and the whole negotiation is not a *zero-sum* game.

In the last decade, an extensive body of research in bilateral negotiation has demonstrated that uncertainty in opponent deadline [21, 7, 11, 10] and outside options [12, 6] can affect the quality of the negotiation outcome. In addition to these studies, there are some prominent works that try to generate near-optimal offers with uncertain importance weights [3, 4, 9, 13, 23].

Although the idea of generating Pareto-optimal offer with perfect information has been originally proposed by Raiffa in [19], the algorithmic solution is presented in [14]. Jazayeriy et al. [14] presented the MGT (Maximum Greedy Trade-offs) algorithm to generate Pareto-optimal offers with perfect information. In this paper, an extension to MGT algorithm is presented to generate Pareto-optimal offers with incomplete information.

Finding a near Pareto-optimal can also be ideal, if agents have incomplete information about the opponent's importance weights. Faratin et al. [9] present a fuzzy similarity approach to select the most similar offer to the last received offer in a pool of generated offers by *random trade-offs*. They showed that the quality of generated offer is highly related to the accuracy of the importance weights and the number of random offers. Ros and Sierra [20] presented an improvement on *random trade-offs* algorithm. They proposed *smart random trade-offs (SRT)* algorithm to consider priority over negotiation issues. Their random approach has high complexity. However, in this study, an agent can learn the greedy order very fast and generate near optimal offers.

There are also some research works that try to learn the opponent's importance weights [20, 23, 13, 3, 4]. Learning the order of issues' importance weights is studied by Ros and Sierra [20]. They argued that issues with fewer changes considered as more important than those with more changes during the negotiation process. They used the order of importance weights to improve the random trade-offs algorithm. However, our work differs in that it learns the greedy order of issues which is needed to generate Pareto-optimal offers.

Although Bayesian learning is a popular approach to explore the opponent preferences [23, 13, 3], it needs *a priori* information about the probability distribution of the negotiation likely outcome and updating the probability of all hypotheses in each round of the negotiation. Kernel density estimation (KDE) is a statistical method that can be used to find issues' priorities [4]. This method needs an offline process of previous negotiation encounters to estimate an initial probability density function over the opponent's importance weights. Then, new information can be augmented by online learning from the ongoing negotiation. The main problem related to Bayesian learning and KDE is that they work in supervised way, while negotiation with incomplete information is unsupervised. Therefore, in these studies, some assumptions about agents' concession strategy are needed to update agents' beliefs. In this respect, agents usually assumed to have a decreasing aspiration level. However, our work differs in that it can generate (near) Pareto-optimal offers without any assumption about agents' concession strategy.

In the following sections we present a bilateral negotiation model and algorithms to generate Pareto-optimal offers.

3 MULTI-ISSUE BILATERAL NEGOTIATION MODEL

This model is an extension to the *"split the pie"* game [1, 18, 2] and the *alternating offer protocol* [18] where two autonomous agents, a and b, negotiate over n issues (such as $x_1 = \textit{price}$, $x_2 = \textit{delivery}$, $x_3 = \textit{warranty}$, $\dots$) by sending and receiving offers $x = (x_1, x_2, \dots, x_n)$. Each issue, i, is like a *pie of size 1* that should be divided between a and b by:

$$f_i^a(x_i) + f_i^b(x_i) = 1 \tag{1}$$

where f_i^a and f_i^b are the shares of agents a and b, respectively. $f_i : D_i \to [0,\ 1]$ is also called scoring function that evaluates the desirability of x_i, where D_i is the domain that presents all possible values for x_i.

Having $f^a + f^b = 1$ implies that a single issue is like a *zero-sum* game where taking a portion of benefit (pie) by the agent causes a loss (with same amount) for the opponent, but multi-issues negotiation is not a *zero-sum* game because issues have different worth for agents. For example, issue i may be very important for agent a while it has low importance for agent b. In fact, agent a assigns importance weight w_i^a to issue i which may differ from the opponent importance weight, w_i^b. We assume that negotiation issues are independent and agents have normalized importance weights:

$$\sum_{i=1}^{n} w_i = 1.$$

Given an offer x, agent's utility is additive function over weighted issues [19]. The utility function $u : S \to [0,\ 1]$ can be formulated as:

$$u(x) = \sum_{i=1}^{n} w_i . f_i(x_i) \tag{2}$$

where S is the set of the all possible offers ($||S|| = \Pi_{i=1}^{n} ||D_i||$).

Without loss of generality, assume that negotiation begins by sending an offer from agent a at time $t = 1$ (starter can be selected randomly to remove the advantage/disadvantage of the first mover). Then the opponent, agent b, accepts the received offer if $u^b(x) \geq u^b_{min}$ (where u^b_{min} is the utility threshold for agent b) or rejects the received offer and continues the negotiation by sending a counter-offer. This process continues until one of the agents reaches its deadline (t_{max}).

$$Action(x,t) = \begin{cases} Agree & u(x) \geq u_{min} \\ Withdraw & t > t_{max} \\ Continue & t \leq t_{max} \end{cases} \tag{3}$$

To continue the negotiation, agent should generate an offer. To this end, agent should make a decision about its aspiration-level θ (target utility). Usually, at the beginning of the negotiation, agent's aspiration-level is close to 1; however, when time passes to t_{max} it becomes close to u_{min} (t_{max} and u_{min} are private information). Aspiration-level depends on:

- current time, t
- agents' deadline, t^a_{max} and t^b_{max}
- negotiation history H (set of the sent and received offers)
- outside options (possible agreements if agent withdraws from the current negotiation and communicates with other agents, or concurrent negotiations).

Negotiation history, H, shows the opponent behavior. In case an agent does not have perfect information about its opponent, negotiation history can be used to learn the opponent's preferences.

Outside options can affect the agent's utility threshold (u_{min}). In case there are many opponents in e-marketplace to negotiate with, the agent may increase its utility threshold because it has more chance to find another opponent and reach a better agreement.

Agents may use time dependent or behavior dependent (or both) decision functions to determine the aspiration level [8]. Choosing the best aspiration level is important in automated negotiation because it can lead agents to a *Nash-equilibrium* solution [17]. But, unfortunately, agents have incomplete information about their opponent's deadline (t_{max}) and utility threshold (u_{min}). Therefore, agents should find a *sequential equilibrium* [15] by updating their beliefs during the negotiation process.

It is important to note that generating Pareto-optimal offer needs information about the opponent's importance weights. On the other hand, having information about deadlines and outside options guides agents to find an aspiration level that can be used to find equilibrium solution.

4 GENERATING PARETO-OPTIMAL OFFER

Generating a near Pareto-optimal offer (x) at the given aspiration level (θ) is a challenging problem in automated negotiation. To generate the optimal offer, an agent should fill its aspiration level, θ, with the most valuable issues which maximize the opponent's utility. This problem can be mathematically stated as:

- maximize
$$u' = \sum_{i=1}^{n} w'_i.f'_i(x_i)$$
- subject to
$$u = \theta = \sum_{i=1}^{n} w_i.f_i(x_i)$$

where u', w' and f' are the opponent's utility, importance weight and scoring function, respectively.

The MGT algorithm generates Pareto-optimal offer with perfect information. Agents with incomplete information can generate near-optimal offers by learning their opponent's preferences. Here, an extension to MGT algorithm is used to generate near-optimal offers.

4.1 Maximum Greedy Trade-Offs (MGT) Algorithm

The problem of the generation Pareto-optimal offer at given aspiration level θ is somehow similar to fractional knapsack problem [16, 5, 14].

To maximize the opponent's utility, an agent must fill its aspiration level based on the greedy order (agenda). An issue i is the best greedy choice if it has maximum worth to the opponent while it has minimum occupation of the aspiration level.

Definition 1. For any issue i call $r_i = w_i/w'_i$ its greedy ratio. A greedy choice k is an issue which minimizes the greedy ratio, i.e. $r_k \leq r_i$ for all i's.

Algorithm 1 shows the MGT algorithm which generates offer at given aspiration level θ. Let us say A is the set of negotiation issues and D is the set of fixed issues (that the agent has already decided about their utilities). In each iteration, the agent selects the greedy choice (issue i) from $(A - D)$ (line 6) and tries to maximize the opponent's utility (line 9).

Algorithm 1 Maximum Greedy Trade-off (MGT) [14]

```
1:  $A \leftarrow \{1, 2, \ldots, n\}$      /* set of issues */
2:  $D \leftarrow \emptyset$                /* set of decided issues is initially empty */
3:  $D_w \leftarrow 1$                     /* summation of undecided issues' weight */
4:  $S_u \leftarrow 0$                     /* summation of decided issues' utility */
5:  while $(\theta - S_u) > 0$ do
6:      $i \leftarrow$ Select the greedy choice from (A-D)
7:      $D \leftarrow D \cup \{i\}$
8:      $D_w \leftarrow D_w - w_i$
9:      $u_i \leftarrow max(0, \;\; \theta - S_u - D_w)$   /* the lowest value for $u_i$ */
10:     $S_u \leftarrow S_u + u_i$
11:     $x_i \leftarrow f_i^{-1}(u_i/w_i)$   /* using the reverse function to generate issue value */
12: return $(x_1, x_2, \ldots, x_n)$ as the generated offer
```

The loop continues until agent assigns values to all issues and generates the output offer $x = (x_1, x_2, \ldots, x_n)$ with utility θ.

The agent with perfect information needs $O(n)$ to generate optimal offers by MGT algorithm. The correctness of the MGT algorithm is shown in [14]. An agent can generate Pareto-optimal offers if it selects the greedy choice in each iteration (line 6). Given the number of the negotiation issues, n, there are $n!$ orders that can be used to generate offers.

Definition 2. The order, λ, is the sequence of the negotiation issues that can be used to generate offer in MGT algorithm. The greedy order, λ^*, is the sequence that an agent uses to generate a Pareto-optimal offer.

Two operators ($\rightarrow$ and $\rightsquigarrow$) are used to show the order. The expression $\lambda_{i \rightarrow j}$ means that j is exactly after i in the order λ. The expression $\lambda_{i \rightsquigarrow j}$ means that j is one of the issues that should be selected after i in the order λ.

As already discussed, in *MGT* algorithm, agent selects issues based on their greedy ratio. In other words:

$$\frac{w_i}{w'_i} \leq \frac{w_j}{w'_j} \implies \lambda^*_{i \rightsquigarrow j}.$$

Lemma 1. In bilateral negotiation, agents should have reverse greedy orders to generate Pareto-optimal offers.

$$\lambda^* = -\lambda'^*$$

where λ^*, λ'^* are the greedy orders that the agent and its opponent use to generate offers, respectively.

Proof. The agent uses the w_i/w'_i to rank the negotiation issues, while the opponent uses the w'_i/w_i. It means that agents have reverse orders and the best greedy choice for the agent is the worst greedy choice for the opponent and vice versa – $\lambda'^*_{j\rightsquigarrow i} \iff \lambda^*_{i\rightsquigarrow j}$. □

Usually, the greedy order, λ^*, is unique and, therefore, the Pareto-optimal offer at any aspiration level is also unique; but there is a special case that the agent can generate more than one Pareto-optimal offer.

Corollary 1. The generated Pareto-optimal offer is not unique if there are two (or more) issues, like i and j, with the same ratio:

$$r_i = r_j$$

or

$$\frac{w_i}{w'_i} = \frac{w_j}{w'_j}.$$

Proof. The number of generated offers at given aspiration-level depends on the number of greedy orders. In other words, each order will produce a different offer. If there exist two (or more) issues, like i, j, with the same ratio $r_i = r_j$ then there will be more than one greedy order that can be used to generate offers. Therefore, the maximum greedy trade-offs can generate more than one Pareto-optimal offers at given aspiration-level. □

The value for scoring functions depends on the aspiration-level that Pareto-optimal offer should be generated at. The following theorem states that scoring functions get the same ranks as the greedy order does.

Theorem 1. Given a Pareto-optimal offer, x, the following statement is always true.

$$\lambda^*_{i\rightsquigarrow j} \iff f_i(x_i) \leq f_j(x_j)$$

Proof. Part I: At first, we prove that if issue i is preferred to issue j for making trade-offs in *MGT* algorithm then issue i has lower scoring value:

$$\lambda^*_{i\rightsquigarrow j} \implies f_i(x_i) \leq f_j(x_j).$$

Without loss of generality, assume that i and j are the first and second greedy choices, respectively ($\lambda_1 = i$, $\lambda_2 = j$). Therefore, at first, the agent makes a maximum trade-off on issue i, and then it selects issue j. The following cases may possibly happen:

Case 1: $(1 - w_i \leq \theta)$.

In this case $0 \leq u_i \leq w_i$ and $u_j = w_j$ which means that $0 \leq f_i \leq 1$ and $f_j = 1$.

Case 2: $(1 - w_i - w_j \leq \theta \leq 1 - w_i)$.

In this case $u_i = 0$ and $0 \leq u_j \leq w_j$ which means that $f_i = 0$ and $0 \leq f_j \leq 1$.

Case 3: $(\theta \leq 1 - w_i - w_j)$.

In this case $u_i = 0$ and $u_j = 0$ which means that $f_i = f_j = 0$.

As can be seen, in all possible cases the implication $(f_i \leq f_j)$ is true. This deduction can be continued by considering $2nd$ and $3rd$ greedy choices, and so on.

Part II: Now, we prove that a Pareto-optimal offer can reflect the greedy order.

$$f_i(x_i) \leq f_j(x_j) \implies \lambda^*_{i \rightsquigarrow j}$$

Let issue j be selected before issue i, to generate a Pareto-optimal offer; then according to Part I, we have $f_j(x_j) \leq f_i(x_i)$ which contradicts the initial assumption. Therefore, the issue with smaller scoring function, i, should be selected before the issue with higher scoring function, j. □

4.2 One-Side Incomplete Information

In this section we propose an algorithmic solution for the bargaining problem where one agent has perfect information but the other one has incomplete information. Specifically, we assume that one agent has incomplete information about the opponent's importance weights.

The main idea that helps solve this problem comes from the Theorem 1 where agent can partially/fully learn the order of the greedy ranks.

Let us say that the agent has incomplete information about the opponent importance weights while the opponent has perfect information. The following theorem shows how the agent can learn the greedy order to generate a Pareto-optimal offer.

Theorem 2. If agent receives a Pareto-optimal offer, y, from the opponent with perfect information, then the greedy order can be learned based on the agent's scoring functions.

$$f_i(y_i) < f_j(y_j) \implies \lambda^*_{i \rightsquigarrow j}$$

Proof. Here, the opponent has generated a Pareto-optimal offer, y; therefore based on Theorem 1 we can write:

$$f'_i(y_i) > f'_j(y_j) \implies \lambda'^*_{j \rightsquigarrow i}$$

then according to the Lemma 1 we have $(\lambda'^* = -\lambda^*)$:

$$f'_i(y_i) > f'_j(y_j) \implies \lambda^*_{i \rightsquigarrow j}$$

from the *split the pie game* we have $f + f' = 1$, then:

$$f_i(y_i) < f_j(y_j) \implies \lambda^*_{i \rightsquigarrow j}.$$

In other words, the greedy order can be detected based on the ascending order of the scoring values. The best greedy choice is an issue which has the lowest scoring value. □

Although Theorem 2 gives some clues to find the greedy order, the quality of the learning depends on the opponent's aspiration-level. The following example illustrates that in the early rounds of the negotiation, the agent can just partially learn the greedy order.

4.2.1 Example: Learning the Greedy Order

Consider two agents that negotiate to buy/sell a product. Agents negotiate over three issues $A = \{\text{price}, \text{delivery}, \text{warranty}\}$ with the following domains:

$$\begin{aligned} D_{\text{price}} &= [100, 250] \ \$ \\ D_{\text{delivery}} &= [1, 14] \text{ days} \\ D_{\text{warranty}} &= [3, 24] \text{ months.} \end{aligned}$$

The importance weights of issues (*price*, *delivery time*, *warranty duration*) for the seller agent $w = \langle 0.6, 0.15, 0.25 \rangle$ and for the opponent (the buyer) is $w' = \langle 0.4, 0.3, 0.3 \rangle$. Moreover, the agent and its opponent have the following scoring functions:

$$\begin{aligned} f_{\text{price}}(x) &= \frac{x - 100}{150} \\ f_{\text{delivery}}(x) &= \frac{x - 1}{13} \\ f_{\text{warranty}}(x) &= \frac{24 - x}{21} \end{aligned}$$

$$\begin{aligned} f'_{\text{price}}(x) &= \frac{250 - x}{150} \\ f'_{\text{delivery}}(x) &= \frac{14 - x}{13} \\ f'_{\text{warranty}}(x) &= \frac{x - 3}{21}. \end{aligned}$$

The agent does not know the opponent's importance weights and receives some offers. Here, we want to see how Theorem 2 helps learn the greedy order. Let us say the opponent has used its greedy order $\lambda'^*_{\text{price} \to \text{warranty} \to \text{delivery}}$ to generate the following offers:

$$\begin{array}{lll}
\text{round } 1: & \theta' = 0.95; & y_1 = (118\,\$, 1_{\text{day}}, 24_{\text{months}}) \\
\text{round } 2: & \theta' = 0.85; & y_2 = (156\,\$, 1_{\text{day}}, 24_{\text{months}}) \\
\text{round } 3: & \theta' = 0.75; & y_3 = (193\,\$, 1_{\text{day}}, 24_{\text{months}}) \\
\text{round } 4: & \theta' = 0.65; & y_4 = (231\,\$, 1_{\text{day}}, 24_{\text{months}}) \\
\text{round } 5: & \theta' = 0.55; & y_5 = (250\,\$, 1_{\text{day}}, 21_{\text{months}}).
\end{array}$$

According to Theorem 2, the agent can partially learn the greedy order based on the scoring values of the received offers. Here, f_p, f_d and f_w are used to show the scoring functions for the *price, delivery* and *warranty*, respectively.

$$\begin{array}{ll}
\text{round } 1: f_p(118) = 0.125; f_d(1) = 0; f_w(24) = 0 & \Longrightarrow \lambda^*_{?\to?\to\text{price}} \\
\text{round } 2: f_p(156) = 0.375; f_d(1) = 0; f_w(24) = 0 & \Longrightarrow \lambda^*_{?\to?\to\text{price}} \\
\text{round } 3: f_p(193) = 0.625; f_d(1) = 0; f_w(24) = 0 & \Longrightarrow \lambda^*_{?\to?\to\text{price}} \\
\text{round } 4: f_p(231) = 0.875; f_d(1) = 0; f_w(24) = 0 & \Longrightarrow \lambda^*_{?\to?\to\text{price}} \\
\text{round } 5: f_p(250) = 1.0 \quad ; f_d(1) = 0; f_w(21) = 0.167 & \Longrightarrow \lambda^*_{\text{delivery}\to\text{warranty}\to\text{price}}
\end{array}$$

As can be seen, in rounds 1-4, the agent can just detect *price* as the last/worst greedy choice. However, it cannot recognize the rank of *delivery* and *warranty* because they have equal scoring value ($f_d(1) = f_w(24) = 0$).

In fifth round, finally, the agent can determine the greedy order from the non-equal scoring values. Thus, from fifth round on, the agent can generate Pareto-optimal offers according the learned greedy order. In this example, as long as the opponent keeps its aspiration-level higher than 0.60 and generates Pareto-optimal offers, the agent cannot reveal the greedy order.

This example shows that in the early rounds of the negotiation, agent with incomplete information can just partially learn the greedy order.

Now, let the opponent replace the first offer with a near-optimal offer like (113, $1, 23$). Although this offer is based on the greedy order, the opponent has not applied the maximum trade-offs to generate this offer. Then, the agent can detect the following scoring values:

$$\text{round } 1: f_p(113) = 0.09; f_d(1) = 0; f_w(23) = 0.047 \implies \lambda^*_{\text{delivery}\to\text{warranty}\to\text{price}}.$$

Actually, $(118, 1, 24)$ and $(113, 1, 23)$ have same utility ($\theta' = 0.95$) for the sender but they have different advantages for the receiver. The former one has higher utility for the receiver (Pareto-optimal) and the latter one has non-equal scoring values that can be used by receiver to reveal the exact greedy order. This example addresses the problem of having uncertain greedy order.

4.2.2 Learning the Greedy Order

According to Theorem 2, the greedy order can be learned by sorting the scoring values. Let F be a vector that contains issues' cumulative scoring values which is initialized by zeros at the beginning of the negotiation:

$$F_i = \sum_{k=1}^{m} f_i(y_{k,i})$$

where m is the number of the received offers and $y_{k,i}$ is the i^{th} issue of the k^{th} received offer.

If vector F has elements with unique values, then it can reflect the exact greedy order. In this case, the agent with incomplete information can generate a Pareto-optimal offer. Otherwise, the agent should find a near Pareto-optimal offer.

Algorithm 2 presents a solution to the problem of generating offer with incomplete information (uncertain greedy order). The first part of the algorithm (lines 1–8) is related to learning the greedy order. Then, in the second part (lines 9–17), it generates a near optimal offer. The second part is almost similar to algorithm 1, but some changes in algorithm 1 are made to adapt it with incomplete information.

At first, the agent updates the cumulative scoring values based on the last received offer, y. Then, issues' position in the greedy order can be revealed if they have non-zero cumulative scoring values. Therefore, the agent can make a maximum trade-off to assign a value to the issue; but it may happen that some issues (like *delivery* and *warranty* in Section 4.2.1) have zero cumulative scoring value ($F_i = 0$). In this case, issues have uncertain positions in the greedy order and will be collected in the set A_e (line 7).

In line 8 of algorithm 2, agent ascendingly sorts the issues based on their cumulative scoring value to form the sequence λ. Since the first issues in the sequence λ may have uncertain position, the agent starts making trade-offs from the last issue in the sequence down to the first issue (line 11). In this vein, the agent assigns the highest possible utility to the selected issue in the sequence.

Having a set of issues with the same scoring values (A_e), the agent assigns equal scoring value to these issues (line 14) by using the following formula:

$$f = \frac{\textit{remained utility}}{\textit{total weights of issues in } A_e} = \frac{\theta - Su}{Sw_e} \tag{4}$$

where Sw_e is the summation of the importance weights for issues with uncertain position in the greedy order.

It is worth mentioning that Algorithm 2 can generate Pareto-optimal offers if the agent learns the greedy order. Otherwise, Algorithm 2 generates near Pareto-optimal offers. Moreover, the learning method presented in this algorithm is embedded in offer-generating algorithm. Therefore, this learning method cannot be applied in other offer-generating algorithms.

Algorithm 2 Generating offer with learning the order of greedy choices

Given:

A_e: *set of the issues with uncertain position in greedy order*

Sw_e: *summation of the importance weights for issues in* A_e

F: *a vector that contains the cumulative scoring values for the received offers*

```
1: $A_e \leftarrow \emptyset$
2: $Sw_e \leftarrow 0$
3: **for** $i = 1$ to $n$ **do**
4:     $F_i \leftarrow F_i + f_i(y_i)$
5:     **if** $F_i == 0$ **then**
6:         $Sw_e \leftarrow Sw_e + w_i$
7:         $A_e \leftarrow A_e + \{i\}$
8: $\lambda \leftarrow$ ***Sort*** *issues based on* $F_i$
9: $Su \leftarrow 0$
10: **for** $k = n$ **downto** 1 **do**
11:     $i \leftarrow \lambda_n$                          /* select the last/worst greedy choice */
12:     $u_i \leftarrow min(w_i,\ \theta - Su)$          /* the highest possible utility */
13:     **if** $i \in A_e$ **then**
14:         $u_i \leftarrow (\theta - Su).(w_i/Sw_e)$
15:     **else**
16:         $Su \leftarrow Su + u_i$
17:     $x_i \leftarrow f_i^{-1}(u_i/w_i)$
18: **return** $(x_1, x_2, \ldots, x_n)$
```

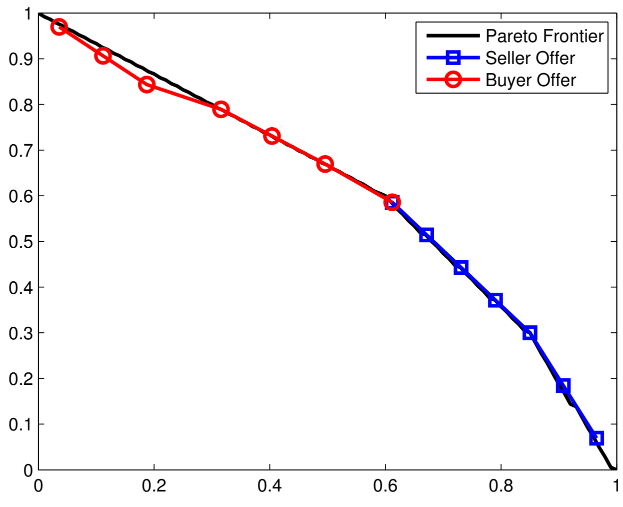
Algorithm 2 can generate a near Pareto-optimal offer with uncertain information in $O(n \log n)$. In fact, an agent should update the greedy order (line 8) in each round of the negotiation that needs $O(n \log n)$.

5 EXPERIMENT

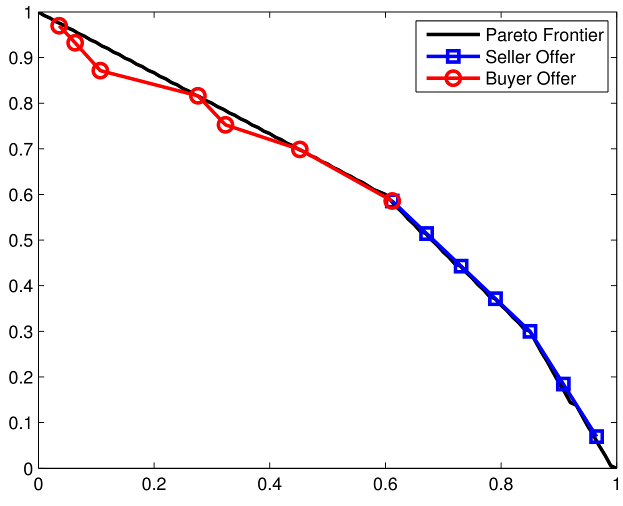
In the following experiments, two agents (a buyer and a seller) are considered, one with perfect information and the other with incomplete information. The agent with perfect information uses *MGT* algorithm to generate Pareto-optimal offers. The other agent uses algorithm 2 (*MGT* with learning), and *smart random trade-offs (SRT)* [20] to generate offers under uncertainty.

Distance form the Pareto-optimal curve can show the quality of the generated offers. Offers which are closer to Pareto-frontier curve are preferred. Experiments are based on the negotiation setting in Section 4.2.1 where the importance weights of issues *(price, delivery time, warranty duration)* for the seller agent $w^{\text{seller}} = \langle 0.6, 0.15, 0.25 \rangle$ and for the buyer is $w^{\text{buyer}} = \langle 0.4, 0.3, 0.3 \rangle$.

Figures 1 and 2 show the results from Algorithm 2 where the agent with incomplete information can learn the greedy sequence and generate near Pareto-optimal offers. The outcome can be compared to SRT algorithm.

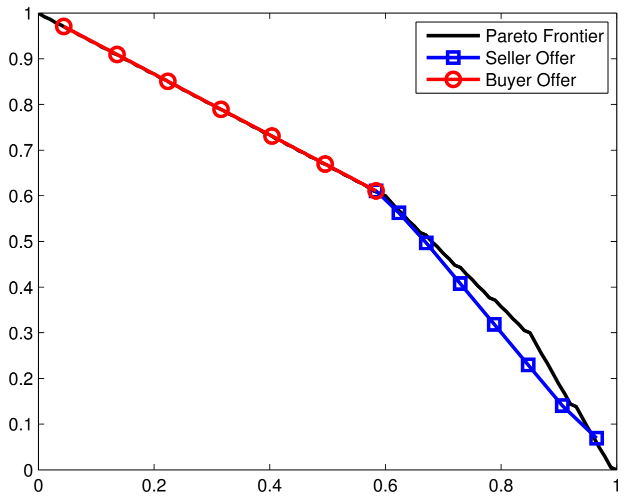


a)

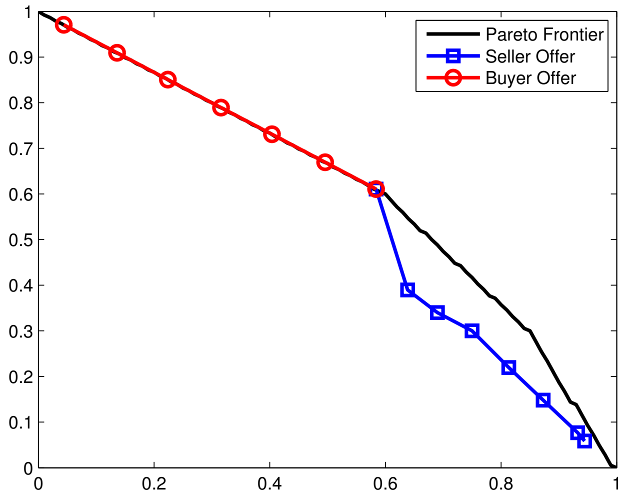


b)

Fig. 1. Generating offers with one-side incomplete information. The seller has perfect information and moves over Pareto-frontier curve while the buyer has incomplete information. a) the buyer uses algorithm 2 to generate offers, b) the buyer uses SRT algorithm to generate offers.



a)



b)

Fig. 2. Generating offers with one-side incomplete information. The buyer has perfect information and moves over Pareto-frontier curve while the seller has incomplete information. a) the seller uses algorithm 2 to generate offers, b) the seller uses SRT algorithm to generate offers

In the first experiment (Figure 1), the seller had perfect information and generated Pareto-optimal offers (by using MGT algorithm). On the other hand, the buyer had incomplete information and generated offers by using Algorithm 2 (graph a)) and SRT (graph b)). It can be seen that offers generated by algorithm 2 are closer to Pareto-frontier curve than those generated by SRT algorithm.

In the next experiment (Figure 2), it was assumed that the buyer has perfect information and the seller has incomplete information. In this experiment the buyer uses the MGT algorithm to generate Pareto-optimal offers. In graph a), the seller uses Algorithm 2 to learn the greedy sequence and generate near Pareto-optimal offers. In graph b), the seller uses SRT algorithm and generate near Pareto-optimal offers. Again, it can be seen that offers generated by algorithm 2 are closer to Pareto-frontier curve than those generated by SRT algorithm.

According to the results shown in Figures 1 and 2 learning the greedy sequence and generating offers by using Algorithm 2 is more effective than using SRT algorithm (learning the order of the opponent's importance weights).

6 CONCLUSIONS

This paper studies the problem of generating Pareto-optimal offer in bilateral multi-issue negotiation with one-side uncertain information about the opponent importance weights. The problem is modeled by multiple *split the pie* games and *alternating offer protocol* as a *non-zero-sum* game.

In this study, at first, the properties of Pareto-optimal offers are investigated. Then, an extension to MGT algorithm is presented to generate near Pareto-optimal offers. It has been proved that agents should have reverse greedy sequences (agendas) to generate Pareto-optimal offers. Moreover, the greedy sequence can be revealed from received offers by sorting cumulative scoring values. The agent with uncertain information can use these findings to learn the greedy sequence and generate the (near) Pareto-optimal offer. The empirical analysis indicates that the proposed learning method can effectively improve the quality of the generated offers.

Similar to game theoretic models, in this study, negotiation issues are considered to be continuous variables. Although continuous negotiation issues are widely used in game theory to model the negotiation, in real world marketplaces negotiation issues are mostly discrete. Therefore, generating offer with discrete issues can be studied in the future. Moreover, generating Pareto-optimal offers with both side incomplete information is still a challenging problem to be studied.

REFERENCES

[1] Binmore, K.—Osborne, M.—Rubinstein, A.: Noncooperative Models of Bargaining. Handbook of Game Theory with Economic Applications 1, 1992, pp. 179—225.

[2] Binmore, K.—Vulkan, N.: Applying Game Theory to Automated Negotiation. Netnomics 1, 1999, No. 1, pp. 1–9.

[3] Buffett, S.—Spencer, B.: A Bayesian Classifier for Learning Opponents's Preferences in Multi-Object Automated Negotiation. Electronic Commerce Research and Applications 6, 2007, No. 3, pp. 274–284.

[4] Coehoorn, R.—Jennings, N.: Learning on Opponent's Preferences to Make Effective Multi-Issue Negotiation Trade-Offs. In Proceedings of the 6th International conference on Electronic Commerce, 2004, ACM New York, NY, USA, pp. 59–68.

[5] Cormen, T. H.—Leiserson, C. E.—Rivest, R. L.—Stein, C.: Introduction to Algorithms. Third Edition. The MIT Press, September 2009.

[6] Cuihong, L.—Giampapa, J.—Sycara, K.: Bilateral Negotiation Decisions with Uncertainty Dynamic Outside Options. IEEE Transactions on Systems, Man and Cybernetics, Part C 36, 2006, No. 1, pp. 31-44.

[7] Di Giunta, F.—Gatti, N.: Bargaining Over Multiple Issues in Finite Horizon Alternating-Offers Protocol. Annals of Mathematics and Artificial Intelligence 47, 2006, No. 3, pp. 251–271.

[8] Faratin, P.—Sierra, C.—Jennings, N.: Negotiation Decision Functions for Autonomous Agents. Robotics and Autonomous Systems 24, 1998, No. 3, pp. 159–182.

[9] Faratin, P.—Sierra, C.—-Jennings, N. R.: Using Similarity Criteria to Make Issue Trade-Offs in Automated Negotiations. Artificial Intelligence 142, 2002, No. 2, pp. 205–237.

[10] Fatima, S.—Wooldridge, M.—Jennings, N.: Multi-Issue Negotiation with Deadlines. Journal of Artificial Intelligence Research 27, 2006, No. 1, pp. 381–417.

[11] Gatti, N.—Di Giunta, F.—Marino, S.: Alternating-Offers Bargaining with One-Sided Uncertain Deadlines: An Efficient Algorithm. Artificial Intelligence 172, 2008, Nos. 8–9, pp. 1119–1157.

[12] Gerding, E.—La Poutre, H.: Bilateral Bargaining with Multiple Opportunities: Knowing Your Opponent's Bargaining Position. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews 36, 2006, No. 1, p. 45.

[13] Hindriks, K.—Tykhonov, D.: Opponent Modelling in Automated Multi-Issue Negotiation Using Bayesian Learning. In Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems (IFAAMAS), Volume 1, 2008, pp. 331–338.

[14] Jazayeriy, H.—Azmi-Murad, M.—Sulaiman, M.—Udzir, N.: Pareto-Optimal Algorithm in Bilateral Automated Negotiation with Maximum Greedy Trade-Offs. JDCTA 5, 2011, No. 3, pp. 1–11.

[15] Kreps, D.—Wilson, R.: Sequential Equilibria. Econometrica: Journal of the Econometric Society 50, 1982, No. 4, pp. 863–894.

[16] Martello, S.—Toth, P.: Knapsack Problems: Algorithms and Computer Implementations. Wiley 1990.

[17] Nash Jr., J.: The Bargaining Problem. Econometrica: Journal of the Econometric Society 18, 1950, No. 2, pp. 155–162.

[18] Osborne, M.—Rubinstein, A.: A Course in Game Theory. The MIT Press 1994.

[19] Raiffa, H.: The Art and Science of Negotiation. Harvard University Pressn Cambridge, Mass. 1982.

[20] Ros, R.—Sierra, C.: A Negotiation Meta Strategy Combining Trade-Off and Concession Moves. Autonomous Agents and Multi-Agent Systems 12, 2006, No. 2, pp. 163–181.

[21] Sandholm, T.—Vulkan, N.: Bargaining with Deadlines. In Proceedings of the National Conference on Artificial Intelligence, 1999, John Wiley & Sons pp. 44–51.

[22] Wooldridge, M.: An Introduction to Multiagent Systems. Wiley 2009.

[23] Zeng, D.—Sycara, K.: Bayesian Learning in Negotiation. International Journal of Human-Computer Studies 48, 1998, No. 1, pp. 125–141.

Hamid Jazayeriy is a lecturer at the Faculty of Electrical and Computer Engineering, NIT, Iran. He received his Bachelor of Computer Engineering from university of Tehran in 1996 and his Master of software engineering from University of Isfahan in 2000. In 2011, he received his Ph. D. from University Putra Malaysia, UPM. He is a reader in multi-agent systems, swarm intelligence and graph theory.

Masrah Azmi-Murad received her Ph. D. in artificial intelligence from the University of Bristol, UK in 2005. She is currently an Associate Professor at the Department of Information Systems, Universiti Putra Malaysia. She has led two research projects funded by the Ministry of Science, Technology, and Innovation (MOSTI) and Research University Grant Scheme (RUGS); currently she is leading three ongoing research projects, and is a co-researcher in other eight projects. She has published nationally and internationally in cited journals and conference proceedings. She also serves as editorial member in some Scopus-indexed journals. Her current research interests include text mining, applied informatics, and semantic technology. She is also a member of IEEE Computer Society and of numerous international and national conference committees.

Nasir Sulaiman is a lecturer in computer science at Faculty of Computer Science and Information Technology, UPM, and an Associate Professor since 2002. He received his Ph. D. in neural network simulation from Loughborough University, UK in 1994. His research interests include intelligent computing, software agents and data mining. He is currently Head of Intelligent Computing Research Group, FSKTM, UPM.

Nur Izura Udzir has been an Associate Professor at the Faculty of Computer Science and Information Technology, Universiti Putra Malaysia (UPM) since 1998. She received her Bachelor of Computer Science (1996) and Master of Science (1998) from UPM, and her Ph. D. in computer science from the University of York, UK (2006). She is a member of IEEE Computer Society. Her areas of specialization include access control, secure operating systems, intrusion detection systems, coordination models and languages, and distributed systems. She is currently the Leader of the Information Security Group at the faculty.