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 others [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. These information are rarely available that 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 an especial 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 the 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.
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 as 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 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 agent 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 among
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 com-
plexity. However, in this study, an agent can learn the greedy order very fast and
generate near optimal offers.

There are also some research work 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 high important than those with having 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 pref-
erences [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 an 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 den-
sity 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 ne-
gotiation with incomplete information is unsupervised. Therefore, in these studies,
some assumptions about agents’ concession strategy are needed to update agents’
believes. 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 $x_1 = price$, $x_2 = delivery$, $x_3 = warranty$, ...) by sending and receiving offers
$x = (x_1, x_2, \ldots, x_n)$. Each issue, i, is like a pie of size 1 that should be divided
between a and b by:

$$f_a^i(x_i) + f_b^i(x_i) = 1$$

(1)

Where $f_a^i$ and $f_b^i$ are the share of agent a and b, respectively. $f_i : D_i \rightarrow [0, 1]$ is also
called scoring function that evaluates the desirability of $x_i$, where $D_i$ is the domain
that presents all possible values for $x_i$. 

H. Jazayeriy, M. Azmi-Murad, N. Sulaiman, N.I. Udizir
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 an importance weight \( w^a_i \) to issue \( i \) which may differ from the opponent importance weight, \( w^b_i \). 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 \( u : S \rightarrow [0, 1] \) 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 the 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 \( u^b(x) \geq u^b_{\text{min}} \) (where \( u^b_{\text{min}} \) is the utility threshold for agent \( b \)) or it rejects the received offer and continues the negotiation by sending a counter-offer. This process continues until one of the agents reaches its deadline \( (t_{\text{max}}) \).

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

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

- current time, \( t \)
- agents’ deadline, \( t^a_{\text{max}} \) and \( t^b_{\text{max}} \)
- 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 \( (u_{\text{min}}) \). 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 \( (t_{\text{max}}) \) and utility threshold \( (u_{\text{min}}) \). 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 guide 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 \( (\theta) \) is a challenging problem in automated negotiation. To generate the optimal offer, an agent should fill its aspiration-level, \( \theta \), with the most valuable issues which maximize the opponent’s utility. This problem can be mathematically stated as:

\[
\begin{align*}
\text{maximize} & \quad u' = \sum_{i=1}^{n} w'_i f'_i(x_i) \\
\text{subject to} & \quad u = \theta = \sum_{i=1}^{n} w_i f_i(x_i)
\end{align*}
\]

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 \( \theta \) 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 \( i \).
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. $i \leftarrow \text{Select the greedy choice from } (A - D)$
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. $x_i \leftarrow \frac{1}{w_i} f^{-1}(u_i/w_i)$ /\* 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 the agent assigns values to all issues and generates the output offer $x = (x_1, x_2, \ldots, x_n)$ with utility $\theta$.

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

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

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

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

**Lemma 1.** Agents, in bilateral negotiation, should have reverse greedy orders to generate Pareto-optimal offers.

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

where $\lambda^*$, $\lambda^*$ 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 revers orders and the best greedy choice for the agent is the worst greedy choice for the opponent and vice versa, $\lambda_{i\rightarrow j}^* \iff \lambda_{j\rightarrow i}^*$.

Usually, the greedy order, $\lambda^*$, is unique and, therefore, the Pareto-optimal offer at any aspiration level is also unique. But there is an especial 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:

$$r_i = r_j \quad \text{or} \quad \frac{w_i}{w'_i} = \frac{w_j}{w'_j}$$

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.

$$\lambda_{i\rightarrow j}^* \iff f_i(x_i) \leq f_j(x_j)$$

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:

$$\lambda_{i\rightarrow j}^* \implies f_i(x_i) \leq f_j(x_j)$$

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 mean 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 mean 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 it 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.

\[ f_i(x_i) \leq f_j(x_j) \implies \lambda^*_{i\to j} \]

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

\[ \square \]

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 to solve this problem comes from the Theorem 1 where agent can partially/fully learn the order of the greedy ranks.

Let’s 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.

\[ f_i(y_i) < f_j(y_j) \implies \lambda^*_{i\to j} \]

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

\[ f'_i(y_i) > f'_j(y_j) \implies \lambda^*_{i\to j} \]

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

\[ f'_i(y_i) > f'_j(y_j) \implies \lambda^*_{i\to j} \]

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

\[ f_i(y_i) < f_j(y_j) \implies \lambda^*_{i\to j} \]

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

\[ \square \]

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 $A=\{\text{price, delivery, warranty}\}$ with the following domains:

- $D_{\text{price}} = [100, 250]$ 
- $D_{\text{delivery}} = [1, 14]$ days 
- $D_{\text{warranty}} = [3, 24]$ months

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

- $f_{\text{price}}(x) = x - 100$ 
- $f'_{\text{price}}(x) = 250 - x$ 
- $f_{\text{delivery}}(x) = x - 1$ 
- $f'_{\text{delivery}}(x) = 14 - x$ 
- $f_{\text{warranty}}(x) = 24 - x$ 
- $f'_{\text{warranty}}(x) = x - 3$

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

- **round 1**: $\theta' = 0.95$; $y_1 = (118$, 1 day, 24 months) 
- **round 2**: $\theta' = 0.85$; $y_2 = (156$, 1 day, 24 months) 
- **round 3**: $\theta' = 0.75$; $y_3 = (193$, 1 day, 24 months) 
- **round 4**: $\theta' = 0.65$; $y_4 = (231$, 1 day, 24 months) 
- **round 5**: $\theta' = 0.55$; $y_5 = (250$, 1 day, 21 months)

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.

- **round 1**: $f_p(118) = 0.125$; $f_d(1) = 0$; $f_w(24) = 0$  $\Rightarrow \lambda^*_{\text{price}\rightarrow\text{delivery}\rightarrow\text{price}}$ 
- **round 2**: $f_p(156) = 0.375$; $f_d(1) = 0$; $f_w(24) = 0$  $\Rightarrow \lambda^*_{\text{price}\rightarrow\text{warranty}\rightarrow\text{price}}$ 
- **round 3**: $f_p(193) = 0.625$; $f_d(1) = 0$; $f_w(24) = 0$  $\Rightarrow \lambda^*_{\text{price}\rightarrow\text{delivery}\rightarrow\text{price}}$ 
- **round 4**: $f_p(231) = 0.875$; $f_d(1) = 0$; $f_w(24) = 0$  $\Rightarrow \lambda^*_{\text{price}\rightarrow\text{warranty}\rightarrow\text{price}}$ 
- **round 5**: $f_p(250) = 1.0$; $f_d(1) = 0$; $f_w(21) = 0.167$  $\Rightarrow \lambda^*_{\text{delivery}\rightarrow\text{warranty}\rightarrow\text{price}}$

As it 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 nonequal 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 replaces the first offer with a near-optimal offer like (113, 1, 23). Although, this offer is based on the greedy order, the opponent have not applied the maximum trade-offs to generate this offer. Then, the agent can detect the following scoring values:

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

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 later one has none-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 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 none-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 the algorithm 2, agent ascendingly sorts the issues based on their cumulative scoring value to form the sequence \( \lambda \). Since, the first issues in the
sequence $\lambda$ may have uncertain position, the agent start 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:

$$f = \frac{\text{remained utility}}{\text{total weights of issues in } A_e} = \frac{\theta - Su}{Sw_e}$$  \hspace{1cm} (4)$$

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

It is worth to mention 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

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

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 is \( w_{\text{seller}} = \langle 0.6, 0.15, 0.25 \rangle \) 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.

In the first experiment (Figure 1), the seller had perfect information and generated Pareto-optimal offers (by using MGT algorithm). In 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, generated offers 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
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 form 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
Generating Pareto-optimal Offers

incomplete information is still a challenging problem to be studied.

REFERENCES


