Pareto-optimal Algorithm in Bilateral Automated Negotiation

Hamid Jazayeriy, Masrah Azmi-Murad, Nasir Sulaiman, Nur Izura Udizir
Faculty of Electrical and Computer Engineering, Noshivani University of Technology, 47148-71167 Babol, Iran, jhamid@nit.ac.ir
Faculty of Computer Science and Information Technology, University Putra Malaysia, 43400 Serdang, Malaysia, {masrah, nasir, izura}@fsktm.upm.edu.my
doi:10.4156/jdcta.vol5. issue3.1

Abstract

In this paper we present a Pareto-optimal algorithm in bilateral automated negotiation where the negotiation is modeled by “split the pie” game and alternating-offer protocol. Pareto-optimality is the seminal condition in the bargaining problem which leads autonomous agents to the Nash-equilibrium. Generating Pareto-optimal offer in multi-issue bargaining is a computationally complex problem, specially, when autonomous agents have incomplete information about deadline, outside options and the opponent's preferences. Unfortunately, yet to date, there is no articulation that clearly describes an algorithm to generate offer in multi-issue negotiation with perfect information. To this end, we present the maximum greedy trade-offs (MGT) algorithm that generate offers at any aspiration-level in \(O(n)\) with assuming that the order of greedy choices is given, otherwise the complexity will be \(O(n \log n)\). We also provide analytical proof for the correctness of the maximum greedy trade-offs algorithm.

Keywords: Bilateral Negotiation, Pareto-optimal, Multi Issues, Algorithm

1. Introduction

Multi-agents systems (MAS) can provide a powerful tool to deal with some emergent problems in complex systems[1, 19, 29]. Automated negotiation is a multi-disciplinary area of research that enables autonomous agents to track their intentions by providing communication among them [26]. Usually, agents have conflicting intentions that should be resolved by proposing a series of offers and counter-offers. Unfortunately, generating offers is computationally complex, especially, when agents have incomplete information. To find an equilibrium solution agents need to know about their opponent preferences, deadline and outside options. There are an extensive body of work that investigates the effect of the incomplete information about deadlines and outside options on the negotiation outcomes.

It is important to note that having information about outside options and the opponent's importance deadline help agent to maximize its own utility, while having information about the opponent's importance weights over negotiation issues helps agent to generate Pareto-optimal offers and, consequently, satisfy the opponent.

Although extensive academic research has studied the effect of the outside options and deadlines information on the quality of the outcomes in automated negotiations [22, 14, 24, 4, 11, 8, 3], much less research has investigated the effect of the opponent's importance weights that can increase the negotiation success rate [6, 27, 28, 17, 5].

Studies on opponent's preferences still need more attentions, because, without knowing the opponent's preferences, agents cannot generate Pareto-optimal offers. Although, Fatima et al. [14] suggest that multi-issue bargaining is like a fractional knapsack problem, yet to date, there is no clear articulation for bargaining problem with perfect information. Lack of algorithmic solution for generating Pareto-optimal offers with perfect information, motivate us to propose an algorithm that generate offers by maximum greedy trade-offs (MGT).

In this respect, a greedy choice that helps agent to select an issue to make a maximum trade-offs is introduced. Moreover, a recursive version of MGT algorithm is presented to simplify the analysis of the algorithm. The complexity of MGT algorithm, like other greedy algorithm, is \(O(n \log n)\). In fact, the agent needs \(O(n \log n)\) to sort the greedy choices, while the greedy algorithm just needs \(O(n)\) to generate a Pareto-optimal offer.
The rest of the paper is organized as follows. Section 2 details related work in bilateral automated negotiation. Section 3 describes the negotiation model used in this study by detailing the negotiation protocol and some basic concepts. Section 4 describes our greedy algorithms to generate Pareto-optimal offers. In section 5, we analyze the time complexity of the proposed algorithms. Finally, Section 6 outlines the conclusions and our plans for future study.

2. Related Work

Automated negotiation has received wide attention in the fields of artificial intelligence and game theory. 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: opponent's deadline, opponent's preferences and outside option. Usually, agents have incomplete information about their opponent which arises uncertainty and makes automated negotiations as an interesting area of research in AI field.

In the last decade, an extensive body of research in bilateral negotiation has demonstrated that uncertainty in opponent deadline can affect the quality of the negotiation outcome. For example, Fatima et al. [13] have studied the effect of deadline in an agenda-base negotiation. They also extend their study [12, 14] by modeling the bilateral multi-issue negotiation in the form of split the pie game [2, 23, 4]. Although they introduced the greedy solution for the bargaining problem, they did not present an algorithm that generate Pareto-optimal offer. Moreover, their guess for the time complexity of the greedy solution is $O(n)$ which is not, rigorously, realistic (later in section 5 we show that the time complexity at the first round is $O(n \log n)$ while at the consecutive rounds will be $O(n)$).

Finding equilibrium strategy with uncertain deadline is an interesting area of study [24]. Giunta and Gatti [9] present an equilibrium solution for bargaining in finite horizon. They also showed that there is an equilibrium solution in bargaining with one-side uncertain deadlines [15].

Another source of uncertainty in bilateral negotiation is outside option where agents have several opponents to negotiate with. There are some valuable research works that studied the effect of the outside options on the negotiation outcomes. For example, Gerding and Poutre [16] modeled a competitive market in the form of ultimatum game [25] where a buyer have several options (sellers) before making a purchase decision. They show that the negotiation outcome is largely depend on the information available to the agents. Another valuable study is conducted be Li et al. [8] that investigates the effect of the dynamic outside options on bilateral negotiation. They showed that considering outside options can increase the negotiation joint utility.

Having information about opponent's preferences helps agent to generate Pareto-optimal offers. Finding a near Pareto-optimal can also be ideal, if agent have incomplete information about the opponent's importance weights. Faratin et al. [10] present a fuzzy similarity approach to generate offers. They showed that the quality of generated offer is highly related to the accuracy of the importance weights. There are also some research work that try to learn the opponent's importance weights [28, 17, 5, 6].

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 [2, 23, 4] and the alternating offer protocol [23] where two autonomous agents, $a$ and $b$, negotiate over $n$ issues (such as
Pareto-optimal Algorithm in Bilateral Automated Negotiation
Hamid Jazayeriy, Masrah Azmi-Murad, Nasir Sulaiman, Nur Izura Udizir

\[ x_1 = \text{price}, \quad x_2 = \text{delivery}, \quad x_3 = \text{warranty}, \ldots \) 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. \( f_i \) is also called scoring function to evaluate the desirability of \( x_i \). Let’s say, \( D_i \) is the domain that presents all possible values for \( x_i \) then the scoring function \( f_i: D_i \rightarrow [0, 1] \) can be simply formulated by a linear function. The scoring function can be increasing (S shape) or decreasing (Z shape) and presented by two parameters: \( \alpha \) and \( \beta \) \( (\alpha, \beta \in D_i) \).

\[ S(x, \alpha, \beta) = \frac{x - \alpha}{\beta - \alpha} \]  \( (2) \)

\[ Z(x, \alpha, \beta) = \frac{\beta - x}{\beta - \alpha} \]  \( (3) \)

Although split of a single issue is like a zero-sum game where taking a portion of benefit (pie) by an agent causes a loss (with same amount) for the opponent, the multi-issue 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 [23]. The utility function \( u: S \rightarrow [0, 1] \) can be formulated as:

\[ u(x) = \sum_{i=1}^{n} w_i f_i(x_i) \]  \( (4) \)

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

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 \( u^b(x) \geq u^b_m \) (where \( u^b_m \) 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 \( t_{max} \).

\[ \text{Admin}(x, t) = \begin{cases} \text{Accept} & u(x) \geq u^b_m \ \\ \text{Reject} & t > t_{max} \ \\ \text{Continue} & t \leq t_{max} \end{cases} \]  \( (5) \)

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 \( t_{max} \) it becomes close to \( u^b_m \). 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 agreement if agent withdraws from the current negotiation and communicate with other agents, or concurrent negotiations).

Negotiation history, \( H \), shows the opponent behavior. In case 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 \((u_{m,n})\). In case there are many opponents in e-marketplace to negotiate with, agent can 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 \([11]\). Choosing the best aspiration-level is important in automated negotiation because it can lead agents to a Nash-equilibrium solution \([21]\). But, unfortunately, agents have incomplete information about their opponent's deadline \((t_{m\omega})\) and utility threshold \((u_{m,n})\). Therefore, agents should find a sequential equilibrium \([18]\) 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 Algorithm

Generating a near Pareto-optimal offer \((x)\) at the given aspiration level \((\theta)\) is a challenging problem in automated negotiation. This section presents the “Maximum Greedy Trade-off” (or simply MGT) algorithm to generate offers where agents have perfect information about its opponent.

4.1. Maximum Greedy Trade-offs (MGT) Algorithm

The problem of the generation Pareto-optimal offer at given aspiration level \(\theta\) is similar to fractional knapsack problem \([20, 7]\). In fractional knapsack problem, one should fill a fixed-size knapsack with the most valuable items. Analogously, in generating Pareto-optimal offer, an agent should generate an offer at given aspiration-level \(\theta\) (fixed-size knapsack) that has the maximum utility for its opponent. 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. Now the question is that, how should an agent assign values to negotiation issues that generate optimal utility for the opponent? Assigning a value \(x_i\) to issue \(i\), will occupy the knapsack with \(u_i = w_i f_i(x_i)\) while it worth \(u'_i = w'_i f'_i(x_i)\) for its opponent. Note that, agents negotiate over conflicting issues, meaning that increasing the scoring function \(f_i(x_i)\) for one agent causes a decreasing scoring function \(f'_i(x_i)\) for the opponent and vice versa (note that \(f + f' = 1\)). In other words, the opponent can gain more if agent select the lowest possible utility for a given issue \(i\).

To maximize the opponent's utility, an agent must fill the knapsack based on a greedy choice. An issue \(i\) is the best greedy choice if it has maximum worth to the opponent while it has minimum occupation of the knapsack (aspiration-level \(\theta\)). The ratio \(r_i = (w_i/w'_i)\) can be used to express the greedy choice. Issue \(i\) with the minimum ratio is the greedy choice.

A greedy algorithm can find an optimum solution for fractional knapsack by selecting the greedy choice and then taking as much as possible of the selected item \([7]\). If there is still room, it finds the next greedy choice and takes as much as possible and continues until the knapsack become full. The only difference in generating Pareto-optimal offer is that, here, agent should take as less as possible from the issue with greedy choice to maximize the opponent's utility.

Now the question is that how can an agent find the maximum trade-off to generate the Pareto-optimal offer at \(\theta\). In other words, given \(i\)-th issue as a greedy choice, what is the lowest boundary for \(u_i\).
Given the target utility $\theta$ and $i$-th issue as the best greedy choice, issue $i$ has its lowest utility ($L_i$) if other issues have their maximum possible scoring value ($f_j(x_j) = 1$). Therefore,

$$
\theta = u_i + \sum_{j=1\atop j \neq i}^n w_j \cdot f_j(x_j)
$$

$$
L_i = \theta - \sum_{j=1\atop j \neq i}^n w_j
$$

It is obvious that $L_i$ is not a negative value. Moreover, agent may have a reserved minimum value ($\rho_i$) over $i$-th issue that limit the $u_i$ (in case, agent has no reserved minimum utility then $\rho_i = 0$).

$$
L_i = m \alpha(\rho_i, \theta - \sum_{j=1\atop j \neq i}^n w_j)
$$  \hspace{1cm} (6)

Equation 6 states the lowest possible utility for the first greedy choice ($i$-th issue). After finding the lowest value ($L_i$) for $u_i$ then the agent repeats the process by selecting the next greedy choice (the issue with next lowest ratio $r_i$) until fixing utilities for all issues where $\sum u_i = \theta$.

Let’s say $A$ is the set of negotiation issues and $D$ is the set of fixed issues (that agent have decided about their utilities). The set of the decided issues ($D$) is initially empty and agent insert the greedy choice (selected issue for making trade-off) into it. Agent can sort the issues based on the greedy ratio $r_i = w_i/w'_i$ to simplify the finding of the best greedy choice from the set of undecided issues ($A - D$).

$$
\{i \in (A - D) \mid \frac{w_i}{w'_i} \leq \frac{w_j}{w'_j} \forall j \in (A - D)\}
$$

Finally, the lowest possible utility for undecided issue $i$ can be elaborated based on the aspiration level $\theta$ and the set of the fixed issues ($D$) and undecided issues ($A - D$).

$$
L_i = m \alpha(\rho_i, \theta - \sum_{j \in D} u_j - \sum_{j \in (A - D)} w_j)
$$  \hspace{1cm} (7)

Note that Equation 6 can only be used to find $L_i$ for the first issue (the best greedy choice), but Equation 7 can be used to find the lowest possible utility ($L_i$) for all issues because the effect of fixed issues is considered.

Algorithm 4.1 shows the steps of the maximum greedy trade-off ($MGT$) algorithm which generates offer at given aspiration level $\theta$ in bilateral automated negotiation. It works as follows. At first, it initializes the set of the decided issues $D$ as an empty set $\emptyset$ and considers the set $A$ as the set of the negotiation issues. Next, variable $Dw$ is initialized by 1 to show the summation of the importance weights for undecided issues and variable $Su$ is initialized by zero to show the utility summation of the decided issues. Next, in each repetition of the loop (lines 5-11), agent selects the best greedy choice $i$ from the set of the undecided issues ($A - D$) (line 6) and then insert $i$ into the $D$ (line 7). Then, agent update the summation of the importance weights for undecided issues $Dw$ (line 8). To make the maximum greedy trade-off, agent calculate the lowest possible utility for issue $i$ (by using Equation 7) and assign it to $u_i$ (line 9). Then the fixed utility ($u_i$) will be added to the $Su$ (line 10).
Algorithm 1: Maximum Greedy Trade-off (MGT)

\[
\begin{align*}
A & \leftarrow \{1, 2, \ldots, n\} & /* \text{Set of issues} */ \\
D & \leftarrow \emptyset & /* \text{Set of decided issues is initially empty} */ \\
D_w & \leftarrow 1 & /* \text{Summation of undecided issues' weight} */ \\
S_u & \leftarrow 0 & /* \text{Summation of decided issues' utility} */ \\
\text{for } k = 1 \text{ to } n \\
& i \leftarrow \text{Select the greedy choice from } (A - D) \\
& D \leftarrow D \cup \{i\} \\
& D_w \leftarrow D_w - w_i \\
& u_i \leftarrow m \alpha(p_i, \theta - S_u - D_w) & /* \text{the lowest value for } u_i */ \\
& S_u \leftarrow S_u + u_i \\
& x_i \leftarrow f_i^{-1}(u_i/w_i) & /* \text{using the reverse function to generate issue value} */ \\
& \text{return } (x_1, x_2, \ldots, x_n) & /* \text{as the generated offer} */ 
\end{align*}
\]

After calculating the \(u_i\) by using the maximum greedy trade-off, agent can find the value for selected issue \(x_i\) (line 11) by using reverse scoring function \(x_i = f_i^{-1}(u_i/w_i)\). 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 \(\theta\). Note that, at the end of the algorithm \(S_u\) will be \(\theta\) and \(D_w\) will be zero.

4.2. A Scenario to Trace the MGT Algorithm

To illustrate the MGT algorithm the example of two agents that negotiate to buy/sell a product can be considered. Agents negotiate over three issues \(A = \{\text{price}, \text{delivery}, \text{warranty}\}\) with following domains:

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

The importance weights of issues (price, delivery time, warranty duration) for the seller agent is \(\mathbf{w} = [0.6, 0.15, 0.25]\) and for the opponent (the buyer) is \(\mathbf{w}' = [0.4, 0.3, 0.3]\). Moreover, agent and the opponent have the following scoring functions:

- \(f_{\text{price}} = S(100, 250); \ f'_{\text{price}} = Z(100, 250)\)
- \(f_{\text{delivery}} = S(1, 14) ; \ f'_{\text{delivery}} = Z(1, 14)\)
- \(f_{\text{warranty}} = S(3, 24) ; \ f'_{\text{warranty}} = Z(3, 24)\)

Now, MGT algorithm can be traced to generate an offer \(x\) at utility \(\theta = 0.75\). At first agent should sort issues based on the ratio \(r = w_i/w_i'\) to find the best greedy choice.

- \(r_{\text{price}} = 1.5; \ r_{\text{delivery}} = 0.5; \ r_{\text{warranty}} = 0.83.\)

Then, agent will select the lowest ratio as the best greedy choice to make a maximum trade-off (the order of the greedy choice is: delivery \(\rightarrow\) warranty \(\rightarrow\) price ). Thus, at first, delivery time will be selected and then agent inserts it into \(D\) \((D = \{\text{delivery} \rightarrow \text{warranty} \rightarrow \text{price}\})\). Lines (6-11) of MGT algorithm are demonstrated by:

\[
\begin{align*}
A - D & = \{\text{price} , \text{delivery} , \text{warranty} \} \\
i & = \text{delivery} \\
D & = \{\text{delivery}\} \\
D_w & = 1 - 0.15 = 0.85 \\
u_{\text{delivery}} & = m \alpha(0, 0.75 - 0 - 0.85) = 0 \\
S_u & = 0 + 0 = 0 \\
x_{\text{delivery}} & = f_{\text{delivery}}^{-1}(0/0.85) = 1 \text{ day}\)
\end{align*}
\]

So far, the agent (seller) proposed the fastest delivery time hoping to satisfy the opponent (buyer) and the opponent gains \(u'_{\text{delivery}} = 0.3\). Then, agent selects the next greedy choice and repeats lines (6-11) of MGT.

\(A - D = \{\text{price} , \text{warranty} \}\)
i = warranty
D = \{delivery, warranty\}
Dw = 0.85 - 0.25 = 0.60
uw_{warranty} = m \alpha(0, 0.75 - 0 - 0.60) = 0.15
Su = 0 + 0.15 = 0.15
x_{warranty} = f_{warranty}^{-1} \left( \frac{0.15}{0.25} \right) = 11.4 \text{ months}

The opponent gains u'_{warranty} = 0.12 from x_{warranty} = 11.4. Finally, after fixing the delivery time and warranty, agent selects the last issue and repeats lines (6-11) of MGT.

A - D = \{price\}
i = price
D = \{price, delivery, warranty\}
Dw = 0.60 - 0.60 = 0
uprice = m \alpha(0, 0.75 - 0.15 - 0) = 0.6
Su = 0.15 + 0.60 = 0.75
x_{price} = f_{price}^{-1} \left( \frac{0.60}{0.60} \right) = 250 \$\

In this example, the opponent has no gain from the last issue (u'_{price} = 0). The generated offer at \( \theta = 0.75 \) is \( x = (250 \$, 1 \text{ day}, 11.4 \text{ months}) \). The opponent's utility for generated offer is \( u'(x) = 0.3 + 0.12 + 0 = 0.42 \). Later, we prove that generated offer by MGT algorithm is Pareto-optimal (meaning that the pair of (0.75, 0.42) is on Pareto-frontier curve).

Note that based in bilateral model (like split the pie game [2, 23, 4]) issues can have continuous values and 11.4 months as warranty duration is accepted. But, usually, in real world negotiation having continuous values for negotiation issues is meaningless. For example, agent cannot accept 6.37 days as delivery time. Thus, agent should fix delivery time to 6 days or 7 days. Similarly, having 11.4 months as a warranty duration is unacceptable and agent should round it to 11 or 12 months which may cause an error in total utility \( \theta \). In fact, generating offer with continuous issues is similar to fractional knapsack problem while generating offer with non-continuous issues is similar to 0-1 knapsack problem. Later, we prove that MGT algorithm can generate Pareto-optimal offer with continuous issues.

4.3. Recursive Maximum Greedy Trade-offs (RMGT) Algorithm

The maximum greedy trade-off MGT algorithm can be written in recursive form as well. Algorithm 4.3 shows the recursive version of the maximum greedy trade-off algorithm which is called RMGT.

Algorithm 2: Recursive Maximum Greedy Trade-off (RMGT)

Given:
- \( A \) : set of the negotiation issues \( \{1, 2, \ldots, n\} \)
- \( \theta \) : aspiration-level
- \( Dw \) : summation of importance weights for undecided issues
- \( \rho = [\rho_1, \rho_2, \ldots, \rho_n] \) : minimum reserved utility

Procedure RMGT(A, \( \theta \), Dw)

i \leftarrow \text{Select the greedy choice from } (A)
ui \leftarrow m \alpha(\rho_i, \theta - (Dw - w_i))
xi \leftarrow f_i^{-1}(ui / wi)
if (\theta - ui) > 0
RMGT(A - {i}, \( \theta - u_i \), Dw - wi)

End Procedure

The recursive algorithm should initially be called by RMGT(A, \( \theta \), 1) to generate offer at \( \theta \) by considering all negotiation issues. It is obvious that initially all issues are undecided and the summation of the importance weights for undecided issues is 1.
Although both algorithms are equal and provide the same output, **RMGT** is more appropriate to show that the algorithm can generate Pareto-optimal offer. The recursive version of the algorithm can show the optimal subproblem property. Let’s say issue \( i \) is the first greedy choice and \( u_i \) is the lowest possible utility for \( i \)-th issue, then the empty room is \( \theta - u_i \) for the next greedy choice. Removing the \( i \)-th issue from the set of the undecided issues leads us to a new greedy problem where agent should generate an optimal offer at aspiration-level \( \theta - u_i \) by using the remaining issues (\( A - \{i\} \)). This subproblem has optimal solution if there is a greedy choice (like \( j \)). This optimal property will remain until all issues take their lowest possible utility and a Pareto-optimal offer is generated.

**Theorem 1** Given an aspiration-level \( \theta \) to generate offer at, the maximum greedy trade-off algorithm can generate Pareto-optimal offer.

**Proof.** Similar to greedy problems, the correctness of the RMGT (or MGT) can be proved by satisfying two properties:
(i) the greedy choice property (local optimum).
(ii) the optimal subproblem property (global optimum).

As already discussed, the issues with smallest ratio \( r_i = w_i / w'_i \) is the greedy choice. Because, it provides the maximum utility for the opponent with the smallest occupation of \( \theta \).

To prove the greedy choice property, we show that the opponent gains more benefit from greedy choice \( i \) than any other issues (like \( j \)). Without loss of generality, let \( i \in A \) be a greedy choice, then:

\[
\frac{w_i}{w'_i} \leq \frac{w_j}{w'_j} \quad \text{for all} \quad j \in A
\]

where \( A \) is the set of the yet undecided issues. If agent chooses \( i \) to make a maximum trade-off then the opponent gains \((u'_i)\) (according to Equation 7 and line-2 in RMGT algorithm):

\[
\begin{align*}
    u_i &= L_i = \theta - Dw + w_i \quad ; \quad f_i = \frac{u_i}{w_i} \\
    f'_i &= 1 - f_i = \frac{Dw - \theta}{w_i} \\
    u'_i &= w'_i, f'_i = \frac{w_i}{w'_i}(Dw - \theta)
\end{align*}
\]

Similarly, if agent chooses \( j \) to make a maximum trade-off then the opponent gains \((u'_j)\):

\[
\begin{align*}
    u'_j &= w'_j, f'_j = \frac{w_i}{w'_j}(Dw - \theta)
\end{align*}
\]

Now, we can compare the opponent benefits from the greedy choice \( i \) and an arbitrary choice \( j \).

\[
\begin{align*}
    u'_i - u'_j &= \frac{(Dw - \theta)}{20} \left( \frac{w_i}{w'_i} - \frac{w'_i}{w'_j} \right) \\
    &\geq 0 \\
    u'_i &\geq u'_j
\end{align*}
\]

Therefore, the opponent gains more benefit from the greedy choice than any other choices.

The next step is to prove the global optimality of the RMGT (MGT) algorithm. Let’s say the greedy solution \( x \) is not optimal, therefore, there should be an offer \( y \) which has maximum utility for the opponent (note that \( u(x) = u(y) = \theta \)). Now, we make some changes on \( y \) so that the resulting offer, \( z \), has higher utility than \( y \). Given the best greedy choice \( i \) and the worst greedy choice \( k \), we can write:
Then, we reduce the utility of the $i$-th issue by $\varepsilon$ $(\varepsilon > 0)$ and increase the utility of the $k$-th issue by $\varepsilon$ (without loss of generality, if $u_i = 0$ then we try the next best greedy choice until reaching an issue that we can reduce its utility. Similarly, if $u_k = w_k$ then we try the next worst greedy choice that we can increase its utility). Therefore, $u(y) = u(z) = \theta$ and,

\[
\begin{align*}
    u_i(z) &= u_i(y) - \varepsilon \quad \rightarrow \quad f_i(y_i) - f_i(z_i) \geq 0 \quad \quad f_{i+1}(y_i) - f_{i+1}(y_i) \geq 0 \\
    u_k(z) &= u_k(y) + \varepsilon \quad \rightarrow \quad f_k(z_k) - f_k(y_k) \geq 0 \quad \quad f_{k+1}(y_k) - f_{k+1}(y_k) \geq 0
\end{align*}
\]  

(11)

Now, we can prove that the opponent can gain more benefit from $z$ than $y$ (note that $y$ and $z$ are the same in all issues but $i$ and $k$).

\[
\begin{align*}
    u'(z) - u'(y) &= w'(f_i'(z_i) - f_i'(y_i)) - w'(f_k'(z_k) - f_k'(y_k)) \\
    &= w'(f_i'(z_i) - f_i'(y_i)) + w'(f_k'(y_k) - f_k'(z_k)) \\
    &\geq 0
\end{align*}
\]

Contradiction, $u'(z) \geq u'(y)$ and $y$ cannot be the optimal offer. Therefore, $y$ cannot do better than the $x$ (generated offer from greedy algorithm). In other words, the opponent cannot gain more benefit than the offer generated by RMGT (or MGT) algorithm. It means $RMGT$ (or $MGT$) algorithm generates the Pareto-optimal offer.

Usually, the greedy order 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. The number of optimal offers at given aspiration-level depends on the number of greedy orders. In other words, each sequence 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 sequence that can be used to generate offers. Therefore, the maximum greedy trade-offs algorithm can generate more than one Pareto-optimal offers at given aspiration-level.

5. Time Complexity

Greedy algorithms need to choose an item (issue) in each iteration (or recursion) based on the criterion that provides greedy choice property. In $MGT$ (or $RMGT$), we assumed that issues are sorted in ascending order based on $r_i = w_i/w_i'$. This sorting will take $O(n \log n)$ time at the beginning of the negotiation. Fortunately, this rank is fix and the agent do not need to recalculate it during the negotiation process. Therefore, $MGT$ (or $RMGT$) can generated offer by assuming that the order of the greedy choices is available and selecting the greedy choice takes instance time $O(1)$.

$MGT$ algorithm has a loop with $n$ repetition to process all negotiation issues (similarly, $RMGT$ need $n$ recursion to process all issues). In each repetition, agent selects a greedy item (line 6 in $MGT$ or line 1 in $RMGT$) which take $O(1)$. Then, the algorithm follows by running lines (7-11) in $MGT$ (or lines (2-4) in $RMGT$) which takes $O(1)$. Therefore, the total complexity of the MGT (or RMGT) algorithm to generate a Pareto-optimal offer is $O(n)$. It means that the complexity of the generating offer with perfect information is less than the complexity of the sorting items which needs to be done before running the greedy algorithm.

6. Conclusions

This paper studies the problem of generating a Pareto-optimal offer in multi-issue negotiation. At first, the bargaining problem is modeled by *split the pie* game and alternating offer protocol.
as the negotiation model which emulate the real world market. Then, the maximum greedy trade-off algorithm is proposed based on the negotiation model to generate a Pareto-optimal offer with complexity of $O(n)$.

This work can be considered as a base optimal solution where agents have information about their opponent's preferences. For the future, the work can be improved in the following ways. In some marketplaces there are some a priori information about one type of agents (for example buyers). Therefore, we can assume that, in some marketplaces, sellers are aware of buyers’s importance weights and, consequently, they can generate Pareto-optimal offers. However, buyers should generate offers with incomplete information about sellers. Therefore, the problem of the generating Pareto-optimal offer with one-side uncertain importance weight can be studied in the future. In this study, negotiation issues are considered to be continuous variables. Although, this continuous 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.

7. Acknowledgment

Financial support from the School of Graduate studies (GSO) at University Putra Malaysia (UPM) is gratefully acknowledged. We thank Enrico H. Gerding for his time and useful comments.

8. References


