Towards Faster Performance of PROMETHEE II in a Lower Class of Complexity

Arash Niknafs  Nasrollah Moghaddam Charkari  Ali Akbar Niknafs
Department of Information Technology Engineering  Faculty of Electrical and Computer Engineering  Department of Computer Engineering
Tarbiat Modares University  Shahid Bahonar University of Kerman
Tehran, Iran  Tehran, Iran  Kerman, Iran
arash.niknafs@gmail.com  charkari@modares.ac.ir  niknafs@uk.ac.ir

Received: August 1, 2010 - Accepted: September 21, 2010

Abstract—PROMETHEE II is one of the most popular members of the well-known family of multi-criteria decision-making methods. One of the main concerns in developing PROMETHEE-based systems is the rapid growth of the response time as the number of alternatives (n) and criteria (k) grow. PROMETHEE II belongs to the computational complexity class of O(n^2). In this paper, a simplified version of PROMETHEE II is proposed and a novel estimation of the simplified PROMETHEE II is introduced. This simplified version reproduces the results of the original method and requires fewer operations. The estimation belongs to the complexity class of O(n log n) and consequently has a shorter response time than that of the simplified version.

The proposed simplification and estimation are tested and evaluated with real-world data. When compared to the original PROMETHEE II and even other similar MCDM methods, such as AHP, ELECTRA, and TOPSIS, the experiments reveal the satisfactory results with a considerably reduced computational complexity and response time.

Keywords—multi criteria decision making; PROMETHEE II; decision support systems; recommender systems

I. INTRODUCTION

Multi-criteria decision making (MCDM) finds an optimal solution from all available alternatives evaluated on multiple (both qualitative and quantitative) and usually conflicting criteria [1]. A well-known family of MCDM methods is called PROMETHEE (Preference Ranking Organization Method for Enrichment Evaluations) [2]. PROMETHEE II, which has been applied to different fields, is the most popular and widely used method among the members of this family.

One of the main concerns of developing information systems (e.g. Decision Support Systems (DSSs), Recommender Systems (RSs)) based on PROMETHEE II is its high computational complexity and consequently its long response time. The response time becomes a more critical issue in systems for which the number of alternatives in the MCDM problem is high. In other words, we face with a computational challenge when the problems are not of trivial dimension. In such circumstances, the high response time of PROMETHEE II strongly reduces the advantage of this method for information systems.

Dias et al. proposed a parallel processing architecture to deal with this issue [3]. Recommender systems, known as a type of decision-support system [4, 5], deal with a large number of alternatives. Consequently, we chose recommender systems to
examine the performance of the two proposed modifications of PROMETHEE II. The application of MCDM methods within recommender systems has attracted researchers’ attention. For example, PROMETHEE-based and AHP-based recommender systems were developed in [5] and [6], respectively.

Today, customers encounter an increasing amount of information in selecting products. In e-commerce, this is also referred to as the information overload problem [7]. One solution is the recommender system. Recommender systems support users by identifying interesting products and services in situations where the number and the complexity of offered items and services outstrip the user’s capability to survey them and make a decision [8]. There are also a lot of applications of MCDM methods in the field of information technology and communication (ICT) about which there are some examples that are presented in section 5.

There are different approaches to categorizing recommender systems. The most common approach suggests that recommender systems can be divided into three categories [7, 9, 10]: content-based, collaborative filtering (CF), and a hybrid.

Content-based recommender systems store the content of information about each item to be recommended. This information will be used to recommend items similar to those the user has preferred in the past [11]. CF systems make recommendations based on their past experiences. They focus on the similarity among users or the similarity among items using users’ ratings [12]. Hybrid recommender systems combine two or more techniques, and in this way they overcome the weaknesses of separately applying each technique [13, 14]. The recommender system developed in our research is a content-based system.

The following three MCDM methods were used for evaluation and comparison purposes in this paper: 1- AHP [15] 2- ELECTRE III [16] (In order to get a complete ranking in ELECTRE III, we used the method proposed in [17] that combines the concordance and discordance indexes.) 3- TOPSIS [18].

Any increase in the number of alternatives causes a big increase in the response time, which discourages researchers from using this method. Yet, some interesting advantages of PROMETHEE II in a recommender system motivated us to investigate reducing the response time and the computational complexity class of this method.

The purpose of this paper is to reduce the complexity class of PROMETHEE II (O(n^2)). First, a simplified version of PROMETHEE with a shorter response time is introduced. Then an estimation (O(n log n)) of the simplified version is proposed. A prototype system is developed to evaluate and analyze the performance of the proposed modifications of PROMETHEE II. The results show a considerable reduction in the computational complexity and response time while outputs maintain a high level of correlation with that of (simplified) PROMETHEE II.

Estimated PROMETHEE II can be used in complex decision-making problems where a great number of alternatives and criteria exist.

The structure of the remainder of this paper is as follows. Section 2 describes PROMETHEE II and the proposed simplified PROMETHEE II. The estimation of the simplified version is discussed in this section as well. Section 3 explains the architecture of the system and the implementation issues. Section 4 shows the experimental results. Finally, Section 5 draws some conclusions and discusses future work.

II. THE ORIGINAL PROMETHEE II

In this section a brief description of PROMETHEE II [19] is presented.

PROMETHEE uses preference functions to form the PCM matrix. Moreover, in contrast to AHP, the preference scores are not limited to the scale of 1 to 9 (integers) in this method. In PROMETHEE, \( P_j(a, b) \) denotes the preference of \( a \) over \( b \) with regards to criterion \( j \). It is computed by \( H_j(d_j) \), where \( H_j \) is one of the functions shown in Appendix A. \( d_j = f_j(a) - f_j(b) \) is the difference between the values of two alternatives at criterion \( j \).

Suppose \( a \) and \( b \) are two items that belong to \( A \) (a set of \( n \) items) and \( k \) is the number of criteria. Let \( f_j(a) \) denotes the value of item \( a \) at criterion \( j \), \((0 < j < k)\). The following steps derive the optimal solution using PROMETHEE II:

1. "Identify the preference function \( P_j \) and \( H_j \) for each criterion."

Depending on whether a criterion in our MCDM problem gets maximized or minimized, the preference function of \( a \) over \( b \) at criterion \( j \) is defined as follows (where \( d = f_j(a) - f_j(b) \) and \( H_j(d) \) is one of the six functions described in Appendix A). If criterion \( j \) is being maximized, then:

\[
P_j(b, a) = 0 \quad \text{and} \quad P_j(a, b) = H_j(d) \quad \text{if} \quad (d > 0)
\]

\[
P_j(b, a) = H_j(d) \quad \text{and} \quad P_j(a, b) = 0 \quad \text{if} \quad (d < 0)
\]

Similarly, if it is being minimized, then:

\[
P_j(b, a) = 0 \quad \text{and} \quad P_j(a, b) = H_j(d) \quad \text{if} \quad (d > 0)
\]

\[
P_j(b, a) = H_j(d) \quad \text{and} \quad P_j(a, b) = 0 \quad \text{if} \quad (d < 0)
\]

2. "Weight of each criterion (\( w_j \))."

Here, the relative importance of each criterion is used to weigh the criteria.

3. "Calculate the outranking degree of \( a \) relative to \( b \) ."

The higher the value of the outranking degree (\( \pi(a, b) \)), the more \( a \) is preferred.

\[
\pi(a, b) = \frac{1}{W} \sum_{j=1}^{k} w_j \frac{P_j(a, b)}{H_j(d_j)} , \quad W = \sum_{j=1}^{k} w_j
\]
(4) “Calculate the complete preorder (net flow), \( \varphi(a) = \varphi^+(a) - \varphi^-(a) \), for each item”.

\[
\varphi^+(a) = \frac{1}{n-1} \sum_{b \in A, b \neq a} \pi(a, b)
\]

\[
\varphi^-(a) = \frac{1}{n-1} \sum_{b \in A, b \neq a} \pi(b, a)
\]

The item with the highest net flow value will be the best item for recommendation. The items are sorted according to their net flow values in the final ranked list.

A. Simplified PROMETHEE II

Any increase in the number of alternatives or criteria will increase the number of operations and consequently response time in PROMETHEE II (which is \(3kn^2\)). Here a simplified version of PROMETHEE II, which requires a smaller number of operations to compute the net flow of an item, is introduced. First, we present our new definition of function \( H \) followed by Theorem which describes how the simplified version is formulated.

**Definition 1.** Suppose \( H'_j(d_j) \) denotes the original definition of function \( H(d) \) for criterion \( j \). Let \( d_j = f_j(a) - f_j(b) \), then \( H(d) \) in the simplified version of PROMETHEE II is defined as follows (Ref. to Appendix A):

\[
H'_j(d_j) = \begin{cases} 
H'_j(d_j) & d_j \geq 0 \\
-H'_j(d_j) & d_j < 0 
\end{cases}
\]

**Theorem 1.** Let \( d_k = f_k(a) - f_k(b) \) and:

\( i, j \in \mathbb{N} \) where \( 0 < i < k \), \( k = \# \) of criteria, \( i \) to be minimized

\( j, k \in \mathbb{N} \) where \( 0 < j < k \), \( k = \# \) of criteria, \( j \) to be maximized

Using the new definitions of \( H(d) \), all the steps in PROMETHEE II, for net flow calculations, can be abstracted to:

\[
\varphi(a) = \frac{1}{W(n-1)} \sum_{s \in A} \sum_{s \in A} w_s H_s(d_s) - \sum_{s \in A} \sum_{s \in A} w_s H_s(d_s)
\]

**Proof.** From the description of the original PROMETHEE II in section 2, it can be inferred that:

\[
\varphi(a) = \frac{1}{n-1} \sum_{s \in A} \pi(a, b) - \frac{1}{n-1} \sum_{s \in A} \pi(b, a) = k
\]

\[
k = \frac{1}{n-1} \sum_{s \in A} \left[ \frac{1}{W} \sum_{s \in A} w_s P_s(a, b) - \frac{1}{n-1} \sum_{s \in A} \frac{1}{W} \sum_{s \in A} w_s P_s(b, a) \right]
\]

\[
= \frac{1}{n-1} [W \sum_{s \in A} (w_s P_s(a, b) - w_s P_s(b, a))]
\]

Whether \( a \) has to be maximized or minimized, one of the two functions \( P_s(a, b) \) and \( P_s(b, a) \) always becomes zero while the other one will be equal to \( H_s(d_s) \), where \( d_s = f_s(a) - f_s(b) \). This means:

\[
\forall h \in A, b \neq a: P_s(a,b) - P_s(b,a) = \begin{cases} 
H_s(d_s) & (d_s > 0 \text{ and } \maximized) \\
-H_s(d_s) & (d_s > 0 \text{ and } \minimized)
\end{cases}
\]

Changing the definition of \( H'_j(d_j) \), as indicated in Definition 1, the following is valid:

\[
\varphi(a) = \frac{1}{W(n-1)} [\sum_{s \in A} \sum_{s \in A} w_s H_s(d_s)] - \sum_{s \in A} \sum_{s \in A} w_s H_s(d_s)]
\]

As Theorem 1 suggests the net flow can be calculated by simply using the above formula, which requires a smaller number of operations to provide the same output as PROMETHEE II.

B. An Estimation of Simplified PROMETHEE II

Although the simplified PROMETHEE II requires fewer operations than PROMETHEE II, its computational complexity order is the same as PROMETHEE II. In this regard, we introduce the estimated PROMETHEE II, which not only has a shorter response time but also has lower class of complexity. In this section the estimation of the simplified PROMETHEE II is presented. The estimation works much faster than the simplified version and produces outputs that are highly correlated with (simplified) PROMETHEE II outputs.

**Theorem 2.**

For each increasing function like \( H \) the following is true:

\[
\forall a, x \in A, x \neq a:
\sum_{s \in A} H_s(f_s(a) - f_s(x)) > \sum_{s \in A} H_s(f_s(b) - f_s(x)) \Rightarrow f_s(a) > f_s(b)
\]

**Proof.**

\[
\forall d_i, d_{i+1}, t \in N \text{ and } t < n:
\sum_{s \in A} H_s(f_s(a) - f_s(x)) > \sum_{s \in A} H_s(f_s(b) - f_s(x))
\Rightarrow \sum_{s \in A} f_s(a) - f_s(x) > \sum_{s \in A} f_s(b) - f_s(x) \Rightarrow 
\sum_{s \in A} f_s(a) - f_s(x) - f_s(b) + f_s(x) > 0 \Rightarrow 
\sum_{s \in A} f_s(a) - f_s(b) > 0 \Rightarrow f_s(a) > f_s(b)
\]

If we could generalize Theorem 2 to:

\[
\sum_{s \in A} \sum_{s \in A} w_s H_s(f_s(a) - f_s(x)) > \sum_{s \in A} \sum_{s \in A} w_s H_s(f_s(b) - f_s(x))
\Rightarrow \sum_{s \in A} f_s(a) > \sum_{s \in A} f_s(b)
Then we could infer that:
\[ \varphi(a) > \varphi(b) \Rightarrow \sum_{i} w_{i} f_{i}(a) - \sum_{i} w_{i} f_{i}(b) > \sum_{i} w_{i} f_{i}(b) - \sum_{i} w_{i} f_{i}(b) \]

However the above mentioned generalization is not always true. That is because \( f_k(a) \) at each criterion has a different scale, meaning and concept. As a primary step towards introducing an estimation of net flow (\( \varphi_{es}(a) \)), the following introduces the relationship (function) \( T \), which maps \( f_k(a) \) values into a unit scale.

Suppose \( f(i,k) \) represents the value of item \( i \), \( (i = \{0,1,\ldots,n-1\}) \) at criterion \( k \), \( (k = \{1,\ldots,K\}) \) then through the following steps, the values of each item at different criteria are mapped into a new scale with the range of \( 1 \) to \( n \) (underlined values show the \# of operations in each step). Fig. 1 depicts the pseudo code for these steps.

**Step 1.** Sort all values at each criterion and assign the indices of 1 to \( n \) to the corresponding items (in each criterion separately).

If a criterion is meant to be maximized (minimized), the items are sorted ascending by their values in that criterion. (descending). \((kn \log n)\)

**Step 2.** Now we count the number of unique values at each criterion; this number is denoted by \( l \). Then the sorted array of values of items at a certain criterion is divided into \( l \) parts (\( 0 < k \leq n \)). Starting from the first (smallest) index in the sorted array, all parts are indexed with a unique number from 1 to \( l \). \((kn + kn)\)

**Step 3.** To each item (\( \forall a \in A \)) in each of the above-mentioned \( l \) parts, a rating value of \( (R(a) = \frac{n}{t+1}) \) is assigned; where \( t = \{1,2,\ldots,l\} \) denotes the index of the part (Ref. to step 2) to which an item belongs. \((k)\)

**Step 4.** Now that we have mapped \( f_k(a) \) values to \( R_k(a) \) through the strictly increasing function \( T_k(f_k(a)) \), \( (T_k : f_k(a) -> R(a)) \), an estimation of \( \varphi_{es}(a) \) can be denoted as follows: \((k)\)

\[ \sum_{i} w_{i} T_{i}(f_{i}(a)) - \sum_{i} w_{i} T_{i}(f_{i}(a)) \]

The following Theorem, which is based on the above-mentioned mapping relationship, introduces the proposed estimation of the simplified version of PROMETHEE II. While the simplified PROMETHEE II has the same accuracy (output) as the original one, the proposed estimation of PROMETHEE II does not provide exactly the same outputs (a compromise has been made between response time and accuracy in following sections).

**Theorem 3.** \( \forall a, b \in A, b \neq a: \)
\[ \varphi(a) > \varphi(b) \Rightarrow \sum_{i} w_{i} T_{i}(f_{i}(a)) - \sum_{i} w_{i} T_{i}(f_{i}(a)) > \sum_{i} w_{i} T_{i}(f_{i}(a)) - \sum_{i} w_{i} T_{i}(f_{i}(a)) \]

**Proof:**
\[ \varphi(a) = \frac{1}{W(n-1)} \left[ \sum_{i} \sum_{j=1}^{l} w_{i} H_{i}(d_{i}) - \sum_{i} \sum_{j=1}^{l} w_{i} H_{i}(d_{i}) \right] \]

From Theorem 1 it can be concluded that:

\[ \sum_{i} \sum_{j=1}^{l} w_{i} H_{i}(f_{i}(a)) - \sum_{i} \sum_{j=1}^{l} w_{i} H_{i}(f_{i}(a)) \]
\[ > \sum_{i} \sum_{j=1}^{l} w_{i} H_{i}(f_{i}(b)) - \sum_{i} \sum_{j=1}^{l} w_{i} H_{i}(f_{i}(b)) \]

Using the above mentioned mapping technique and Theorem 2:
\[ \sum_{i} \sum_{j=1}^{l} w_{i} T_{i}(f_{i}(a)) - \sum_{i} \sum_{j=1}^{l} w_{i} T_{i}(f_{i}(a)) \]
\[ > \sum_{i} \sum_{j=1}^{l} w_{i} T_{i}(f_{i}(b)) - \sum_{i} \sum_{j=1}^{l} w_{i} T_{i}(f_{i}(b)) \]
\[ \sum_{i} \sum_{j=1}^{l} w_{i} T_{i}(f_{i}(a)) - \sum_{i} \sum_{j=1}^{l} w_{i} T_{i}(f_{i}(a)) \]
\[ > \sum_{i} \sum_{j=1}^{l} w_{i} T_{i}(f_{i}(b)) - \sum_{i} \sum_{j=1}^{l} w_{i} T_{i}(f_{i}(b)) \]

Theorem 3 suggests \( \varphi_{es} = \sum_{i} w_{i} T_{i}(f_{i}(a)) - \sum_{i} w_{i} T_{i}(f_{i}(a)) \) as an estimation of \( \varphi(a) \) in PROMETHEE II. Based on this estimation, net flows for a set of alternatives can be calculated a smaller number of operations.

**III. THE PROTOTYPE'S DESIGN ATTRIBUTES**

In PROMETHEE, the first step is to identify decision criteria and corresponding \( H(d) \) function types. The set of selected criteria should be complete and not redundant. This means that all major aspects should be taken into consideration, while keeping the number of criteria as small as possible; a double counting of impacts should also be avoided [20]. The criteria were selected from a set of studies in the literature of business and marketing [21-27].
(considering the fact that the system is going to recommend from among different, less frequently purchased items) and then discussed with experts. The following four criteria were selected: overall reviewer (previous users/customers) ranking, price, brand reputation, and customer’s interests.

A prototype of the system is developed in a single window with different tabs. In the first tab, users specify the weights of each factor (criterion) on a scale of 1 to 7 (integers). In the second tab, users specify their favorite categories by choosing from a list. Customer purchased items are then added to a cart in the third tab by their corresponding ID number in the database. The fourth tab, as shown in Fig. 2, is used to recommend items to the customer with details on the product features so that the customer can provide feedback based on the detailed information on each item.

The information about all items is stored in a table in SQL Server 2000 on the local host. The item’s ID number is set as primary key of the table. Part of the data table which stores items’ information in the database is shown in Fig. 3.

IV. EXPERIMENTS AND RESULTS

This section describes some experiments conducted on real-world data to measure the response time and the deviation of outputs of the two proposed modifications from that of the original PROMETHEE II. A prototype system was developed by Visual C# .NET 2005 and SQL Server 2000 Enterprise Edition.

A. Evaluation measures

Three metrics, the Spearman rank correlation, response time (in seconds) and mean absolute error (MAE), are commonly used in related works to measure correlation, speed and deviation respectively.

The first evaluation measure is the Spearman’s rank correlation coefficient (R) or the Spearman rho (ρ), a non-parametric measure of correlation, which was first introduced in [28] and since then it has been used in many studies, such as [7, 29]. It measures the relationship between two sets of ordinal (ranked) values. The Spearman rho has two advantages. First, a normal distribution is not necessary because it is a nonparametric measure, and, second, it is less affected by outliers [30]. R is computed by the following formula:

\[ R = 1 - \frac{6 \sum D_i^2}{n(n^2 - 1)} \]

\( D_i = U_i - V_i \), where \( U_i \) and \( V_i \) are ranks denoted by two different MCDM methods for an item like \( a_i \); \( n = 1, 2, ..., N \) is the number of alternatives. The greater the value of \( R \) is, the more the results of the two methods are correlated.

To measure how close the performance of the system is to the original PROMETHEE II, (Simplified PROMETHEE II) we use mean absolute error (MAE), which measures the average of absolute differences between ranks assigned by any pair of MCDM methods. MAE has been used in different works [11, 31, 32] to evaluate rank-based recommender systems.

\[ MAE = \frac{1}{N} \sum_{i=1}^{N} |p_i - r_i| \]

\( p_i \) and \( r_i \) are ranks of item \( i \) in two different MCDM methods and \( N \) is the number of all ranked items.

Response time (RT), in seconds, is one of the evaluation metrics of algorithms and systems [33]. RT shows the time that each method takes to produce a final result (a ranked list of items).

Herein, the precision measure is also adopted to measure accuracy and compare different methods’ outputs. The precision (P) is the ratio of erroneous estimates of rating to the correct ratings [34].

\[ P = 1 - \frac{1}{N} \sum_{i=1}^{N} \frac{\max(|R_{\text{max}} - r_i|, |R_{\text{min}} - r_i|)}{2} \]
Where $R_{\max}$ and $R_{\min}$ are the maximum and minimum possible ranks for an item, respectively.

**B. Experiment settings and data**

The main experimental goals were initially to examine the accuracy, and the degree to which the proposed modifications (simplification and estimation) contribute to response time reduction. For this purpose a prototype of the system was developed using C# .NET 2005 and SQL Server 2000 on a Pentium PC platform (1.7 G CPU, 256 M of RAM).

A set of real world data (from an electronics store database) was used in the experiments. A total of 500 records (items) from 15 different categories (such as PCs, scanners, printers) were stored in a table in SQL Server 2000 on the local host. Each item has a unique ID number. The column corresponding to IDNumber in the table design was set as the primary key. The other four important columns in the table contain information about price, brand reputation, category number and average of other customers’ ratings.

In order to evaluate and compare the performance of the proposed modifications, the main part (core) of the recommender system, which is its MCDM algorithm, was developed in six different scenarios using four MCDM methods (AHP, ELECTRE III, PROMETHEE II and TOPSIS), the proposed simplification and estimation of PROMETHEE II. We have asked 30 users to use the system. In all iterations, the evaluation parameters (Precision, MAE, Response time and Spearman rank correlation) were calculated. The results are presented in the following subsection.

**C. Results**

As we discussed before, the simplified PROMETHEE II, contrary to the estimated version, produces exactly the same outputs as those of PROMETHEE within a relatively shorter time. There is also a compromise between speed and accuracy in estimated PROMETHEE. Table 1 shows the correlation between the results of the estimation and those of the other MCDM methods.

MAE shows the deviation of the results of each method from the results of the other methods. The MAE of the estimation outputs from those three MCDM methods’ outputs is shown in table 3. As it is shown in table 3, the range of the MAE values is below 17%. It is important to note that larger MAE values have been considered acceptable in other studies [11, 31, 32].

To show the level of confidence in the results in tables 1, 3 and 6, each of those tables is followed by a similar table (table 2, table 4 and table 7) that includes the confidence interval (CI) of the elements in its preceding table.

### Table 1. THE AVERAGES OF SPARAN RANK CORRELATION COEFFICIENTS (SRC) BETWEEN THE OUTPUTS OF DIFFERENT MCDM METHODS

<table>
<thead>
<tr>
<th>Method</th>
<th>AHP</th>
<th>TOPSIS</th>
<th>ELECTRE</th>
<th>Simplified</th>
<th>Estimation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Estimation</td>
<td>0.7620</td>
<td>0.7358</td>
<td>0.8027</td>
<td>0.7414</td>
<td>1</td>
</tr>
</tbody>
</table>

### Table 2. 95% CONFIDENCE INTERVAL (CI) OF SPARAN RANK CORRELATION COEFFICIENTS (SRC) BETWEEN THE OUTPUTS OF DIFFERENT MCDM METHODS

<table>
<thead>
<tr>
<th>Method</th>
<th>AHP</th>
<th>TOPSIS</th>
<th>ELECTRE</th>
<th>Simplified</th>
<th>Estimation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Estimation</td>
<td>± 0.03</td>
<td>± 0.04</td>
<td>± 0.02</td>
<td>± 0.03</td>
<td>0</td>
</tr>
</tbody>
</table>

### Table 3. AVERAGES OF MAE (%) BETWEEN THE OUTPUTS OF DIFFERENT MCDM METHODS

<table>
<thead>
<tr>
<th>Method</th>
<th>AHP</th>
<th>TOPSIS</th>
<th>ELECTRE</th>
<th>Simplified</th>
<th>Estimation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Estimation</td>
<td>15.89</td>
<td>15.85</td>
<td>15.83</td>
<td>15.83</td>
<td>13.31</td>
</tr>
</tbody>
</table>

### Table 4. 99% CONFIDENCE INTERVAL (CI) OF AVERAGES OF MAE BETWEEN THE OUTPUTS OF DIFFERENT MCDM METHODS

<table>
<thead>
<tr>
<th>Method</th>
<th>AHP</th>
<th>TOPSIS</th>
<th>ELECTRE</th>
<th>Simplified</th>
<th>Estimation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Estimation</td>
<td>0 ± 0.92</td>
<td>0 ± 0.98</td>
<td>0 ± 0.95</td>
<td>0 ± 0.95</td>
<td>0 ± 0.72</td>
</tr>
</tbody>
</table>

Beside the lower complexity class to which the estimation of (simplified) PROMETHEE II belongs, the results in table 5 indicate a shorter response time for both the estimated and simplified versions compared to PROMETHEE II. The estimation and TOPSIS rank first and second with regards to response time.

Finally, to show the competitive accuracy of the estimated and simplified versions, a pair wise comparison has been made between all methods with respect to the precision measure. The results are shown in table 6.

In general, there are two parameters affecting the complexity of these algorithms: the number of criteria $k$ and the number of items (alternatives) $n$. Fig. 4 illustrates the number of operations from two opposite points of view for PROMETHEE II, simplified PROMETHEE, and the estimated version, respectively. The uppermost surface represents the number of operations in PROMETHEE II ($3kn^2$). The surface in the middle and the one under that represent the number of operations in simplified ($kn^2$) and estimation ($kn \log n + 3kn + kl$) versions, respectively.

### Table 5. AVERAGES OF RESPONSE TIMES (SECONDS)

<table>
<thead>
<tr>
<th>Method</th>
<th>Avg. response time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Estimation</td>
<td>1.2578</td>
</tr>
<tr>
<td>Simplified</td>
<td>694.467</td>
</tr>
<tr>
<td>ELECTRE</td>
<td>1393.0795</td>
</tr>
<tr>
<td>TOPSIS</td>
<td>1.8692</td>
</tr>
<tr>
<td>AHP</td>
<td>691.618</td>
</tr>
</tbody>
</table>


Table 6. PRECISION AVERAGES BETWEEN THE OUTPUTS OF DIFFERENT MCDM METHODS

<table>
<thead>
<tr>
<th>Method</th>
<th>Estimation</th>
<th>Simplified</th>
<th>TOPSIS</th>
<th>ELECTRE</th>
<th>AHP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>0.7790</td>
<td>0.7828</td>
<td>0.8177</td>
<td>0.7903</td>
</tr>
<tr>
<td>Simplified</td>
<td></td>
<td>0.8720</td>
<td>0.7546</td>
<td>0.8133</td>
<td></td>
</tr>
<tr>
<td>TOPSIS</td>
<td></td>
<td>1</td>
<td>0.7571</td>
<td>0.8072</td>
<td></td>
</tr>
<tr>
<td>ELECTRE</td>
<td></td>
<td>1</td>
<td></td>
<td>0.8064</td>
<td></td>
</tr>
<tr>
<td>AHP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 7. 90%CONFIDENCE INTERVAL (CI) OF PRECISION AVERAGES BETWEEN THE OUTPUTS OF DIFFERENT MCDM METHODS

<table>
<thead>
<tr>
<th>Method</th>
<th>Estimation</th>
<th>Simplified</th>
<th>TOPSIS</th>
<th>ELECTRE</th>
<th>AHP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>± 0.01</td>
<td>± 0.01</td>
<td>± 0.01</td>
<td>± 0.01</td>
</tr>
<tr>
<td>Simplified</td>
<td></td>
<td>0</td>
<td>± 0.01</td>
<td>± 0.01</td>
<td></td>
</tr>
<tr>
<td>TOPSIS</td>
<td></td>
<td>0</td>
<td>± 0.02</td>
<td>± 0.01</td>
<td></td>
</tr>
<tr>
<td>ELECTRE</td>
<td></td>
<td>0</td>
<td>± 0.01</td>
<td>± 0.01</td>
<td></td>
</tr>
<tr>
<td>AHP</td>
<td></td>
<td></td>
<td></td>
<td>± 0.01</td>
<td></td>
</tr>
</tbody>
</table>

V. CONCLUSIONS AND FUTURE WORK

Although PROMETHEE II is popular for solving MCDM problems, it is not recommended for problems with a large number of criteria and alternatives (e.g. a problem with more than a total number of 100 alternatives and criteria). PROMETHEE’s response time drastically increases with any increase in the number of criteria and (especially) alternatives. Its complexity class may discourage system designers from using this method in applications such as decision support and recommender systems despite its advantages.

Some application-specific advantages of PROMETHEE II in recommender systems, which motivated us to modify and use the method, are:

(a) From a seller’s point of view, recommending an item can be seen as a decision-making problem which requires a MCDA (Multi Criteria Decision Analysis) method with the following two criteria [35]: (1) the method does not allow for trade-offs between criteria, (2) it is simple enough to be understood by non-specialist users. PROMETHEE is identified as a suitable method for these purposes and it satisfies both of those criteria.

(b) As discussed in [7], most of the rank-based recommender systems require user item ratings to collect user preferences. Using the alternative approach, the user can express preference through multiple lists of ranked items. However, it is not feasible to ask a customer to rate all items in a store. Moreover, an underlying assumption of the previous rating scheme is that the user can compare the preference between any two rated items, i.e., the user either rates the two as equal or one better than the other. Imagine a situation in which the user knows that $r_a > r_b > r_c$ and $r_a > r_c > r_f$, but cannot differentiate the interest between items $r_a$ and $r_f$. PROMETHEE automatically makes the pairwise comparison helping the user out of this time consuming process while it can also compare any two items consistently—avoiding human errors—and without difficulty.

(c) As indicated in [33], one of the major concerns in recommender systems is the issue of trust in the recommender. This trust is provided by means of transparency. Transparency includes explaining why specific recommendations appear and helping user understand the recommendation process [36, 37]. As suggested in [35, 38, 39] the PROMETHEE method provides this transparency and increases the customer’s trust in the recommender by being simple and understandable.

The advantages of applying PROMETHEE II in decision support and recommender systems motivated us to work on reducing its complexity. In this paper, first a simplified version of PROMETHEE II was proposed, which worked faster than the original version. The simplified PROMETHEE II does not alter accuracy; it produces the same outputs as the original method does. Then, an estimation of the simplified method was presented which works much faster than the simplified version and belongs to a lower complexity class.

It was not our goal to find the most robust methods
but to verify the reliability of the proposed methods. Therefore, to measure the deviation of the estimation outputs from that of the PROMETHEE II, the MAE between those two methods has been found, while the precision and the Spearman rank correlation are used to show the degree of relevance between their outputs. Calculating the same parameters for other MCDM methods was mainly done in order to prove the competency of two proposed modified versions (simplified and estimation) through comparison. However, a pair wise comparison of performance between all of the MCDM methods is also provided.

Fig. 4 shows that for a small number of alternatives \( n < 100 \), the number of required operations in all three versions is very close when compared to more complex situations \( n > 100 \). It can also be inferred from Fig. 4 that the distance between the upper surface (the number of operations in PROMETHEE II) and the middle one (the number of operations in the simplified PROMETHEE II) gets bigger for a small number of criteria and a large number of alternatives, which is an indication of an even faster performance for the latter in such situations.

Recommending new items is a challenging issue in recommender systems. Our future work includes proposing a solution to the new item problem through manipulation of preference functions and simplified versions of the original PROMETHEE II. It will be also interesting to take a fuzzy solution approach towards this sort of recommendation problems because of the fuzzy nature of customers’ interests and their uncertainties. Using Fuzzy MCDM is also another interesting topic that we are considering as a potential future work.

Finally, it is definitely worth mentioning that the application of multi-criteria decision making is not restricted to a specific field. Decision making is an important aspect of our daily life in different areas and information technology and communication is not exempt. The use of MCDM methods in ICT decision makings has received more attention in the recent years. For instance in [40] the application of MCDM methods in development of information technology industry has been studied. [41] uses these methods for evaluating mobile alternatives in the communication services while [42] uses them in information technology allocation problems. These are just a few examples but we hope that our research results can contribute to faster MCDM processes with less computational complexities in all fields of applications.

**APPENDIX A**

\[ H(d) \text{ returns a value within the range } [0, 1]. \]

The following table shows both the original six functions introduced in [19, 39] and the proposed function definitions in the simplified version. An index \( j \) is used to denote the criterion to which a specific type of function \( H(d) \) is assigned \( (H_j, j = \text{criterion index}). \]

<table>
<thead>
<tr>
<th>Simple version</th>
<th>Type I:</th>
<th>Type II:</th>
<th>Type III:</th>
<th>Type IV:</th>
<th>Type V:</th>
<th>Type VI:</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ H(d) ]</td>
<td>( \sum \frac{d_j}{q} )</td>
<td>( \sum \frac{d_j}{q} )</td>
<td>( \sum \frac{d_j}{q} )</td>
<td>( \sum \frac{d_j}{q} )</td>
<td>( \sum \frac{d_j}{q} )</td>
<td>( \sum \frac{d_j}{q} )</td>
</tr>
<tr>
<td>in the original version of PROMETHEE II</td>
<td>( 0 \leq d_j \leq q )</td>
<td>( 0 &lt; d_j \leq q )</td>
<td>( 0 &lt; d_j \leq q )</td>
<td>( 0 &lt; d_j \leq q )</td>
<td>( 0 &lt; d_j \leq q )</td>
<td>( 0 &lt; d_j \leq q )</td>
</tr>
</tbody>
</table>

REFERENCES


Arash Niknafs is currently a Ph.D. student at the Department of Computer Science in the University of Calgary, Canada. He got his M.Sc. from Tarbiat Modares University and his B.Sc. from Shahid Bahonar University of Kerman, respectively. His research interests include decision support, machine learning, and multi-agent systems.
Nasrollah Moghaddam Charkari received his B.Sc. in computer science from Shahid Beheshti University in Tehran, Iran in 1986 and his M.Sc. and Ph.D. degrees in computer engineering and information systems engineering from Yamanashi University, Japan in 1992 and 1995 respectively. He is currently an assistant professor in faculty of electrical & computer engineering, in Tarbiat Modares university, Tehran, Iran. His research interests include parallel algorithms, AI, and image understanding.

Ali Akbar Niknafs is an assistant professor at the Department of Computer Engineering in Shahid Bahonar University of Kerman (SBUK). He is also the head of the department in SBUK and head of Information Tech. Research Group of ICST (International Centre for Science, High Technology and Environmental Sciences). He got his Ph.D. in Applied Mathematics from SBUK. His research interests include: Data Mining, DSS, Business Intelligence, Quantum Logic circuits and computers, e-Commerce, MIS, Expert Systems.