Mat-2.108 Independent Research Project in Applied Mathematics

A Simulation Study on the Computation of Potentially Optimal Multicriteria Portfolios

Lauri Salminen 50070F
Systems Analysis Laboratory
Helsinki University of Technology

26th January 2005
Contents

1 Introduction ......................................................... 3

2 Multi-Criteria Portfolio Optimization ................................. 4
   2.1 Problem Description .............................................. 4
   2.2 Incomplete Preference Information .............................. 6
   2.3 Solution of a MCPO-problem .................................... 7
   2.4 Overview of the Algorithm ..................................... 8

3 Simulation Setup .................................................... 10
   3.1 Experimental Design ........................................... 10
   3.2 Generating Random MCPO-problems ......................... 12

4 Results .................................................................... 14
   4.1 General ............................................................... 14
   4.2 Full Weight Region ............................................... 14
   4.3 Complete Rank Ordering for Criteria ......................... 17
   4.4 ROC Weights with 10% Relative Interval ....................... 19

5 Conclusions ............................................................ 21
1 Introduction

With the term multi-criteria portfolio optimization (MCPO) problem we refer to the problem of choosing a combination - a portfolio - of alternatives from a large set of alternatives taking many objectives simultaneously into account. In our approach, we assume the outcomes and resource consumption of the alternatives to be known. This kind of problems arise, for example, when dealing with R&D portfolios and many relevant aspects are taken into account simultaneously (Stummer and Heidenberger, 2003). Similar kind of complex resource allocation tasks arise in many non-profit and public organizations (see, for example Kleinmuntz and Kleinmuntz, 2001).

This paper presents a simulation study on the performance of an algorithm that finds potentially optimal portfolios and the corresponding criterion weight regions in a portfolio optimization problem with incomplete preference information. The algorithm is developed at the Systems Analysis Laboratory of Helsinki University of Technology, and its detailed description is presented in a recent Master’s Thesis (Mild, 2004). The results of this study give some implications on the applicability of the algorithm in solving real life MCPO-problems.

The paper is structured as follows. Chapter 2 briefly describes the MCPO-problem structure, incomplete preference information and the associated solution concepts. Chapter 3 presents the experimental design of the simulation and describes how the random MCPO-problems were generated. Chapter 4 presents the results and chapter 5 concludes.
2 Multi-Criteria Portfolio Optimization

2.1 Problem Description

Let there be a set of \( m \) projects \( X = \{x^1, \ldots, x^m\} \) that are each evaluated with regard to \( n \) criteria. A score vector \( v^j = [v^j_1, \ldots, v^j_n] \) where \( j = 1, \ldots, m \) describes the performance of each project with regard to the \( n \) criteria. The score vectors of all \( m \) projects together form a score matrix denoted as \( v \), so that \( [v]_{ji} = v^j_i \). The projects also consume resources; let \( q \) be the number of different resources, and let \( C(x^j) = [c^j_1, \ldots, c^j_q]^T \) where \( j = 1, \ldots, m \) represent the resource consumption of each project. In addition, we assume that the projects are mutually independent and all the scores are non-negative.

Comparing projects with multiple criteria calls for a way to find an aggregate value to describe overall performance of the project. A widely used aggregate value representation in multi-criteria decision making (MCDM) literature is the additive value function (see, for example, Keeney and Raiffa, 1976). Additive value function is basically a weighted sum of criterion specific scores. The criterion weights \( w = (w_1, \ldots, w_n)^T \) are coefficients that represent the relative importance of the different objectives. Typically, the weight coefficients are normalized so that they are non-negative and sum up to one. The overall value of a single project \( x^j \) can now be obtained as

\[
V(x^j) = \sum_{i=1}^{n} w_i v^j_i
\]

Let us define a portfolio of projects as a combination of the available projects. Portfolio is presented as \( p \subset X \). The overall value \( V(p, w, v) \) of a portfolio \( p \) with weights \( w \) is computed as
\[ V(p, w, v) = \sum_{x^j \in p} V(x^j) = \sum_{x^j \in p} \sum_{i=1}^{n} w_i v^j_i = \sum_{i=1}^{n} w_i \sum_{x^j \in p} v^j_i \] (2)

Available resources limit the number of projects that can be chosen. Let \( B_k \) be amount of resource \( k = 1, \ldots, q \). All the available resources are then presented by a budget vector \( B = [B_1, \ldots, B_q]^T \). The resource consumption \( C(p) \) of a portfolio \( p \) is obtained as a sum of its constituent projects’ resource consumption, namely

\[ C(p) = \sum_{x^j \in p} C(x^j) \] (3)

In our notation \( P_F \) denotes the set of feasible portfolios, i.e those that meet the resource constraints. These feasible portfolios can be expressed as a set

\[ P_F = \{ p \in P \mid C(p) \leq B \} \] (4)

The portfolio optimization problem is to find a portfolio of projects that gives the highest overall value (2) subject to the resource constraints;

\[ \max \ V(p, w, v) \quad \text{s.t} \quad p \in P_F \] (5)

With a fixed weight vector \( w = (w_1, \ldots, w_n)^T \) the problem can be solved by integer linear programming (ILP), for example.
2.2 Incomplete Preference Information

The elicitation of precise criterion weights can be impossible or difficult and time-consuming (e.g. Weber, 1987). Some have even argued, that for a decision support system to be simple and easily applicable, no a priori preference information should be required (Stummer and Heidenberger, 2003). In response to this shortcoming, the use of incomplete preference information has been widely studied (e.g. Weber, 1987; Arbel, 1989; Salo & Hämäläinen, 1992; 1995; 2001). Incomplete preference information refers to a setting where the constraints implied by the decision maker’s preference statements are satisfied by several parameter values (Salo & Hämäläinen, 2004).

In our case, the use of incomplete preference information expands the notion of weights from a single point estimate to a set of feasible weights. The feasible weight region, denoted by $S_w$, covers all the weight vectors $w$ that are consistent with the decision maker’s statements. By normalization, all the feasible weight vectors belong to the set

$$S_w^0 = \{ w \in \mathbb{R}^n \mid w \geq 0, \sum_{i=1}^{n} w_i = 1 \}$$

(6)

The set $S_w^0$ corresponds to a situation where no weight information is given. Incomplete preference information can be given through absolute or relative intervals, rankings or differences, and the feasible weight region $S_w$ is reduced by adding the resulting linear inequalities. Since the set $S_w^0$ is a convex polytope (e.g., Grünbaum, 2003) and the decision maker’s preference statements are added as linear constraints, the resulting subset $S_w$ is also a convex polytope.
2.3 Solution of a MCPO-problem

When the criterion weights are not precisely known, the solution of a MCPO-problem - maximize (2) subject to resource constraints - can not be generally expressed as a single optimal portfolio. Instead, dominance structures imposed by the decision maker’s preference statements become a matter of interest. A portfolio \( p \in P_F \) dominates \( p' \in P_F \) if

\[
V(p, w, v) \geq V(p', w, v) \quad \forall w \in S_w \quad \text{and} \quad \exists w \in S_w \quad \text{s.t.} \quad V(p, w, v) > V(p', w, v)
\]  

(7)

A portfolio is dominated if there exists one or more portfolios that dominate it. Otherwise the portfolio is non-dominated. The set of non-dominated portfolios is here denoted by \( P_{ND} \). On the other hand, a portfolio is potentially optimal if and only if

\[
\exists w \in S_w \quad \text{s.t.} \quad V(p, w, v) \geq V(p', w, v) \quad \forall p' \in P_F
\]  

(8)

Thus, each potentially optimal portfolio is optimal in some subset of the feasible weight region. This subset is called an optimal region. We use notation \( P_{PO} \) for the set of all potentially optimal portfolios. It should be noted that a potentially optimal portfolio is not necessarily non-dominated, nor is a non-dominated portfolio necessarily potentially optimal (Rios Insua and French, 1991). For the set of portfolios that are both non-dominated and potentially optimal, we use notation \( P_{POND} \) \( (P_{POND} = P_{PO} \cap P_{ND}) \).

These potentially optimal non-dominated portfolios are generally regarded as (supported) efficient solutions to the MCPO-problem. When also the optimal regions are known, many different kinds of measures can be calculated for describing the robustness and performance of the portfolios. For example, the volume of the optimal region in proportion to the volume of the feasible weight region could be regarded as a simple performance measure. This measure has been proposed, under the name of acceptability index, by Lahdelma et al. (1998) as well.
On the other hand, also the robustness of individual projects can be assessed on the basis of the results. Project level measures can be helpful in a situation when the final solution is constructed gradually project by project. Many performance and robustness measures for both portfolios and projects have been introduced at Systems Analysis Lab at HUT (e.g., Mild, 2004; Liesiö 2004; Liesiö et al, 2005).

2.4 Overview of the Algorithm

The algorithm that is assessed in this paper determines the potentially optimal non-dominated portfolios and the respective optimal regions in a multi-criteria portfolio optimization problem. Next, we present the basic structure of the algorithm; detailed description and theoretical formulation of the algorithm are omitted as they are thoroughly presented in Mild (2004).

As noted earlier, the feasible weight region is a polytope; a convex hull of its extreme points. The linear properties of the MCPO-problem make it possible to determine whether a portfolio is optimal inside a weight polytope by solving the MCPO-problem (5) in each extreme point of the polytope. If all of the extreme points imply the same optimal portfolio, we can conclude that the portfolio is optimal within the whole weight polytope.

If the optimal portfolios are different in the extreme points of the weight polytope, the original polytope is cut into two descendant polytopes with a hyperplane. The extreme points of these descendant polytopes can then be recursively computed from the extreme points of the parent polytope. After this, the MCPO-problem is solved in all new extreme points. The resulting two polytopes are then stored and cut again later if necessary.

In this matter, the algorithm repeats a cycle of cutting polytopes smaller and then solving the MCPO-problems in the new extreme points. As there is a finite number of potentially optimal non-dominated portfolios and corresponding optimal regions, this kind of recur-
sive operation finally produces a number of (small) optimal regions, i.e. weight polytopes that each correspond to a unique potentially optimal non-dominated portfolio.

For this study, the algorithm was implemented in Java programming language. This allowed fast development, platform independence and rapid prototyping of new features. The algorithm was implemented to use an external solver engine. This was important, as the algorithm can be used with many different solving methods for finding the optimal portfolio with a fixed weight vector.

A natural choice for a solver would be an integer linear programming engine. However, as this study concentrates on a simple form of MCPO-problem with only one resource and no interdependencies between different projects, it is possible to use a solver based on a knapsack algorithm. The current knapsack implementation is based on Horowitz-Sahni algorithm (Horowitz and Sahni, 1974). See for example (Martello and Toth, 1990) for description on different knapsack algorithms, and (Liesö, 2004) on using knapsack algorithms in multi-criteria capital budgeting problems.
3 Simulation Setup

3.1 Experimental Design

In order to assess the performance of the algorithm, an experimental design with different size MCPO-problems was developed. Although there are many different factors contributing to the complexity of a MCPO-problem, in the scope of this study only the most obvious factors were varied.

The feasible weight region $S^0_w$ is an $n-1$ dimensional simplex (e.g., Grünbaum, 2003), which means that it has $n$ extreme points and $n(n-1)/2$ edges connecting the extreme points. As the algorithm basically cuts the feasible weight region into small subsets using the coordinates and the adjacency relations of the extreme points, an increase in the dimension substantially increases the computational effort needed. To see how the number of criteria, i.e. the dimension, effects the computation, the simulations included MCPO-problems with 3, 5 and 7 criteria.

The number of project proposals is also an important factor as each new project doubles the number of possible portfolios. To study the effects of the number of projects, four different size project proposal sets were used: 10, 20, 30 and 40 projects.

Altogether, the experimental design consists of $3 \times 4 = 12$ different size MCPO-problems. For each of these 12 cases, a set of 50 random MCPO-problems was generated. Thus, each case had 50 instances, each computed with different random project data. To study how the use of incomplete preference information effects the computation of the potentially optimal portfolios, all the cases were solved with regard to three different specifications of preference information.

First, all 12 cases were solved without any preference information, i.e. with full weight region, $S_w = S^0_w$. Second, the cases were solved by setting a complete rank ordering for
the criteria. This means that the criteria were given an order of preference, resulting in
\[ S_w = \{ w \in S^0_w \mid w_1 \geq w_2 \geq ... \geq w_n \} \]. The third specification of preference information
was a fixed weight vector with relative intervals. The fixed weight vector was the *rank order centroid weight* \( w^{ROC} \), the center of the weight region after complete rank ordering
(see Barron & Barrett, 1996 for more information about ROC weights). Up to +/-10% relative interval was assigned to each component of this weight vector, resulting in
\[ S_w = \{ w \in S^0_w \mid 0.9w_i \leq w_i^{ROC} \leq 1.1w_i \quad \forall i \} \]

The main goal of this study was to assess the possibilities of using the algorithm as a single, independent solution method for multi-criteria portfolio optimization tasks. Hence, the most important performance measure was determined to be the running time of the algorithm. Many other measures were also collected, such as the number of potentially optimal portfolios. To some extent, these other measures can be used to suggest what caused the individual differences in the running time.

<table>
<thead>
<tr>
<th>Number of Criteria</th>
<th>3, 5, 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Projects</td>
<td>10, 20, 30, 40</td>
</tr>
<tr>
<td>Weight Region</td>
<td>Complete Weight Region</td>
</tr>
<tr>
<td></td>
<td>Complete Rank Ordering</td>
</tr>
<tr>
<td></td>
<td>Rank Order Centroid Weights +/-10%</td>
</tr>
</tbody>
</table>

Table 1: Experimental setup of the simulation study. All combinations were solved 50 times with random project data. Hence, the total number of simulation runs was 3*4*3*50=1800
3.2 Generating Random MCPO-problems

For the purpose of this study, a large collection of random MCPO-problems was needed. First step in meeting this need was to generate a large set of random project data. As the biggest cases had 40 projects, a set of 2000 random project proposals was needed in order to compute the 50 instances. For each project, the components of the score vector \( v^j_i = [v^j_1, \ldots, v^j_n] \) were generated using uniform distribution

\[
v^j_i \sim \text{Uniform}(0, 1) \quad i = 1, \ldots, 7 \quad j = 1, \ldots, 2000 \quad (9)
\]

To generate suitable cost for the projects, the individual score vector of each project had to be taken into account. A large and more beneficial project would - on average - also have to be more expensive. This was achieved by generating an individual ratio for the costs and benefits of each project. The ratio of the costs to the sum of scores - \( k \) - was randomly generated for each project using a lognormal distribution (10). By applying the ratio to the score vector, the cost \( C_j \) of project \( j \) evaluated with respect to \( n \) criteria is calculated from (11).

\[
k_j = e^{y_j} \quad y_j \sim \text{Norm}(0, \sigma) \quad (10)
\]

\[
C_j = k_j \sum_{i=1}^{n} v_{ji} \quad (11)
\]

A standard deviation (\( \sigma \)) of 0.95 was used in (10) to obtain a reasonable variation for the project costs. Figure 1 illustrates how the generated projects varied with respect to the ratio between the sum of scores and the project cost.

To form the MCPO-problems, project sets with the required properties \( (m = 10, 20, 30, 40 \) and \( n = 3, 5, 7) \) were taken from this random project data. For each of these sets, the resource constraint (budget) was set to cover 20% of the costs of all project proposals in the set. This means that the resource constraint was set to one fifth of the sum of resource consumption of all project proposals.
Figure 1: The figure illustrates how the ratio between cost of the project and the project benefits vary. The percentages between the different slope values present the proportional amount of projects. For example, 15% of the projects had a cost to sum of scores ratio between 0.25 and 0.5.
4 Results

4.1 General

The simulation study was conducted on a normal desktop workstation. The largest case (40 projects, 7 criteria) proved to be too difficult to be solved 50 times with the full weight region in the available time. All the other results of the experimental design were obtained as text files and analyzed using statistical software. In the analysis phase it became apparent that when dealing with very short time periods (less than 50ms) the operating system of the workstation has 10ms granularity on measuring system time. This reduces the precision of the measurements in the simplest (fastest) cases.

The relative standard deviation of the running time is high, in the larger cases often more than 200%. Hence, predicting the running time of a given problem based on the number of projects and criteria is nearly impossible. For example, in the case of five criteria and 40 projects half of the cases were solved in less than a minute, but one of the 50 instances took an hour to finish. This suggests that the complexity of a MCPO-problem is strongly dependent on the project data (cf. Gustafsson and Salo, 2005).

4.2 Full Weight Region

The average running time of the algorithm solving different size MCPO-problems with full weight region is presented in figure 2. From the graph it can be seen, that both \( n \) and \( m \) have a major effect on the running time of the algorithm. The largest case that could be solved 50 times during the simulation study had seven criteria and 30 projects. The average running time for this problem size was 8 minutes 30 seconds. If the required computation time would continue to grow accordingly, the case with 40 projects would have had an average running time of some 3...11 hours. A problem with seven criteria and
Figure 2: Average running time of the algorithm with complete weight region. Points represent averages and the intervals mark 95% confidence levels. Notice the logarithmic scale.

50 projects would then require weeks to finish. With similar assumptions the number of projects in a problem with three criteria could easily reach more than one hundred before the running time of the algorithm becomes an issue.

The number of potentially optimal portfolios found in different size cases are presented in figure 3. Both factors, $m$ and $n$, have strong influence on the number of potentially optimal portfolios. But it is only the combined effect of these two that causes the number of potentially optimal portfolios to rise quickly. This suggests that a problem with many projects is solvable as long as the number of criteria is only small, and vice versa.

The effect of increasing the number of projects seems to come mainly from the increasing number of potentially optimal portfolios. This can be observed by comparing cases which have same number of criteria and same number of potentially optimal portfolios but
Figure 3: Average number of potentially optimal non-dominated portfolios with complete weight region. The case with 40 projects and 7 criteria is missing, as it proved to be too time consuming to be solved 50 times.

different number of projects. Significant differences were not found between these cases. However, at some point the knapsack-based optimization routine will inevitably slow down when the number of projects is large enough.

On the contrary, the effect of weight dimension - number of criteria - becomes visible when comparing cases with same amounts of potentially optimal portfolios. Figure 4 demonstrates how the number of criteria changes the required computing time as the number of potentially optimal portfolios increases. With moderate size problems (150-200 potentially optimal portfolios) the effect of weight dimension becomes visible. Beyond this, the problems with 7 criteria consume significantly more time than problems with 5 criteria.
Figure 4: The running time of the algorithm with different number of criteria as a function of number of potentially optimal portfolios. The effect of the weight dimension can clearly be seen in the cases with more than 150 potentially optimal non-dominated portfolios.

4.3 Complete Rank Ordering for Criteria

The average running time of the algorithm solving different size MCPO-problems with complete rank ordering for criteria is presented in figure 5. Figure 6 presents the average number of potentially optimal portfolios. The preference information decreased both the running time and the number of potentially optimal portfolios strongly. Also, the differences between different size cases became much more smaller. For example, in cases with 10 projects there was no significant difference in the running time between three, five or seven criteria. For larger cases the differences in the running time were still visible. The average running time for the biggest problem size was less than six seconds.

Complete rank ordering for criteria reduces the volume of the weight region by factor of $1/n!$ where $n$ is the number of criteria. This could lead to believe that the number of potentially optimal portfolios would reduce approximately at the same rate. However, in our study we found that ratio of portfolios found with complete rank ordering to portfolios
Figure 5: Average running time of the algorithm with complete rank ordering for criteria. Points represent averages and the intervals mark 95% confidence levels. Notice the logarithmic scale and its reduction compared to figure 2.

The phenomenon is caused by the optimality regions that are located only partly inside the complete rank ordering constrained polytope. This makes the amount of potentially optimal portfolios in the subsets of a weight region non-additive in nature.

Even though the number of potentially optimal portfolios does not decrease in proportion to the decrease of the weight region we still found the complete rank ordering to be an effective way of decreasing the complexity of the portfolio selection problem. The harder the problem, the bigger the advantage of the preference information. For example, the average running time of problem with seven criteria and 30 projects was reduced by 99.8% (from 8.5 minutes to less than a second) after applying the complete rank ordering.
Figure 6: Average number of potentially optimal non-dominated portfolios with complete rank ordering for criteria. Notice the reduced scale compared with figure 3.

4.4 ROC Weights with 10% Relative Interval

The average running time of the algorithm solving different size MCPO-problems using the rank order centroid weights with 10% relative interval is presented in figure 7. The precision of the measurements was poor because the running times were too short to be measured with system time. This can especially be seen with three criteria where the average of the running time does not significantly differ from zero. Even the cases with five criteria do not significantly differ from each other. Cases with 7 criteria differ from each other, but even the largest case has an average running time of only 28 ms.

Figure 8 presents the average number of potentially optimal portfolios. The effect of constraining the weight region to include the rank order centroid weight with 10% relative intervals drops the number of potentially optimal portfolios radically. In most of the cases
Figure 7: Average running time of the algorithm using ROC-weights with 10% relative interval. Points represent averages and the intervals mark 95% confidence levels. Notice the reduced scale compared to figures 2 and 5.

only one or two potentially optimal portfolios were found, and even in the biggest cases the number of potentially optimal portfolios in this weight region averages 2.3. Still, in some individual cases up to 7 potentially optimal portfolios were found in this relatively small weight region.
Figure 8: Average number of potentially optimal non-dominated portfolios using ROC-weights with 10% relative interval. Notice the reduced scale compared to figures 3 and 6.

5 Conclusions

It has been reported, that in multi-criteria portfolio optimization the maximum number of projects that can be handled by an alternative method - complete enumeration - is no more than 30 projects (Stummer and Heidenberger, 2003). In our approach the number of projects does not have that dramatic effect. For example, when the number of potentially optimal portfolios remained the same, there was no significant difference between cases with 30 and 40 projects. Also, in the light of the results, problems with a relatively high number of projects (100+) could be solved as long as the number of criteria is limited.

On the other hand, an increase in the dimension of the weight space substantially increases the computational effort needed. Partly this is caused by the cutting procedure itself, but mainly because the number of needed optimization rounds grows. These two factors combined make the algorithm relatively sensitive to the number of criteria. This could
be seen as the main limiting factor for the use of the algorithm.

Weaknesses of the algorithm also include the unpredictability of the running time and the large number of optimization rounds needed. The first of these weaknesses prevents the use of the algorithm in a real time on-the-spot decision support systems as two similar looking problems may have running times that are of completely different scale. The latter issue limits the use of logical constraints, synergies and such as they would require the use of integer linear programming solver, which has proved to be much more slower than a knapsack based solver.
6 References


### 7 Appendices

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Projects</th>
<th>Mean</th>
<th>Std.Dev</th>
<th>Rel.Std.Dev</th>
<th>Min</th>
<th>Median</th>
<th>Percentile 75</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>10</td>
<td>8</td>
<td>8</td>
<td>1.0</td>
<td>0</td>
<td>10</td>
<td>10</td>
<td>40</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>17</td>
<td>14</td>
<td>0.8</td>
<td>0</td>
<td>10</td>
<td>20</td>
<td>90</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>86</td>
<td>61</td>
<td>0.7</td>
<td>10</td>
<td>65</td>
<td>130</td>
<td>270</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>404</td>
<td>385</td>
<td>1.0</td>
<td>50</td>
<td>265</td>
<td>441</td>
<td>1832</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>27</td>
<td>30</td>
<td>1.1</td>
<td>0</td>
<td>20</td>
<td>30</td>
<td>130</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>660</td>
<td>1021</td>
<td>1.5</td>
<td>21</td>
<td>310</td>
<td>781</td>
<td>6279</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>11307</td>
<td>14996</td>
<td>1.3</td>
<td>420</td>
<td>4887</td>
<td>13430</td>
<td>56792</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>233925</td>
<td>617312</td>
<td>2.6</td>
<td>4947</td>
<td>62480</td>
<td>180510</td>
<td>3607758</td>
</tr>
<tr>
<td>7</td>
<td>10</td>
<td>239</td>
<td>648</td>
<td>2.7</td>
<td>0</td>
<td>75</td>
<td>190</td>
<td>4406</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>13183</td>
<td>20502</td>
<td>1.6</td>
<td>721</td>
<td>4877</td>
<td>15042</td>
<td>89799</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>512066</td>
<td>1017966</td>
<td>2.0</td>
<td>8532</td>
<td>145805</td>
<td>540948</td>
<td>6306979</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 2: Running time of the algorithm with full weight region
<table>
<thead>
<tr>
<th>Criteria</th>
<th>Projects</th>
<th>Mean</th>
<th>Std.Dev.</th>
<th>Rel.Std.Dev</th>
<th>Min</th>
<th>Median</th>
<th>Percentile 75</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>10</td>
<td>5</td>
<td>6</td>
<td>1.1</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>5</td>
<td>7</td>
<td>1.4</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>10</td>
<td>9</td>
<td>0.9</td>
<td>0</td>
<td>10</td>
<td>20</td>
<td>40</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>24</td>
<td>21</td>
<td>0.8</td>
<td>0</td>
<td>20</td>
<td>31</td>
<td>100</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>5</td>
<td>7</td>
<td>1.4</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>31</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>24</td>
<td>26</td>
<td>1.1</td>
<td>0</td>
<td>20</td>
<td>30</td>
<td>121</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>95</td>
<td>108</td>
<td>1.1</td>
<td>0</td>
<td>70</td>
<td>130</td>
<td>611</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>383</td>
<td>536</td>
<td>1.4</td>
<td>20</td>
<td>225</td>
<td>451</td>
<td>3185</td>
</tr>
<tr>
<td>7</td>
<td>10</td>
<td>15</td>
<td>29</td>
<td>1.9</td>
<td>0</td>
<td>10</td>
<td>20</td>
<td>180</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>107</td>
<td>158</td>
<td>1.5</td>
<td>0</td>
<td>60</td>
<td>110</td>
<td>831</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>909</td>
<td>1891</td>
<td>2.1</td>
<td>40</td>
<td>256</td>
<td>992</td>
<td>12769</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>5877</td>
<td>11892</td>
<td>2.0</td>
<td>60</td>
<td>1097</td>
<td>4577</td>
<td>52476</td>
</tr>
</tbody>
</table>

Table 3: Running time of the algorithm with complete rank ordering for criteria
<table>
<thead>
<tr>
<th>Criteria</th>
<th>Projects</th>
<th>Mean</th>
<th>Std.Dev.</th>
<th>Rel.Std.Dev</th>
<th>Min</th>
<th>Median</th>
<th>Percentile 75</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>10</td>
<td>3</td>
<td>5</td>
<td>1.6</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>1</td>
<td>4</td>
<td>2.5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>2</td>
<td>4</td>
<td>2.2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>3</td>
<td>5</td>
<td>1.9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>20</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>7</td>
<td>7</td>
<td>1.0</td>
<td>0</td>
<td>10</td>
<td>10</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>7</td>
<td>6</td>
<td>0.8</td>
<td>0</td>
<td>10</td>
<td>10</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>9</td>
<td>7</td>
<td>0.7</td>
<td>0</td>
<td>10</td>
<td>10</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>16</td>
<td>22</td>
<td>1.4</td>
<td>0</td>
<td>10</td>
<td>20</td>
<td>151</td>
</tr>
<tr>
<td>7</td>
<td>10</td>
<td>20</td>
<td>12</td>
<td>0.6</td>
<td>10</td>
<td>20</td>
<td>20</td>
<td>70</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>34</td>
<td>24</td>
<td>0.7</td>
<td>10</td>
<td>30</td>
<td>40</td>
<td>150</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>51</td>
<td>44</td>
<td>0.9</td>
<td>20</td>
<td>30</td>
<td>60</td>
<td>200</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>88</td>
<td>91</td>
<td>1.0</td>
<td>20</td>
<td>55</td>
<td>110</td>
<td>370</td>
</tr>
</tbody>
</table>

Table 4: Running time of the algorithm with ROC weights +/-10% relative interval