Non-dominated Portfolios in Capital Budgeting with Interval-valued Project Outcomes
Contents

1 Introduction 3

2 Multi-Criteria Capital Budgeting 4

3 Framework 5

4 Determination Non-dominated Portfolios 8
   4.1 Linear Programming Approach ................................. 8
   4.2 Tree Approach .................................................. 10

5 Computational Efficiency 15
   5.1 Experimental Design .......................................... 15
   5.2 Results .......................................................... 16

6 Analysis of Non-dominated Portfolios 18
   6.1 Decision Rules ............................................... 18
   6.2 Screening ..................................................... 19

7 Illustrative Example 20
8 Discussion and Conclusions

9 References
1 Introduction

In Multi-Criteria Decision Analysis (MCDA) we consider a situation where the decision maker (DM) has to choose one of the given alternatives that are evaluated against multiple criteria (see e.g. French, 1988; Clemen, 1996). Several methods have been developed for aggregating the criterion-specific scores into an overall value measure (see e.g. Hazen, 1986; Weber, 1987; Salo and Hämäläinen, 1992; Eum, Park and Kim, 2001; Salo and Hämäläinen, 2001; Salo and Punkka, 2003). These methods also allow the DM to use incomplete information concerning the importance of criteria and the outcome of criterion-specific scores.

Some work has been done to extend the use of MCDA-methodology into cases where the DM is not faced with the problem of choosing one of the alternatives, but is interested in choosing the best combination (or portfolio) of alternatives that can be afforded with given resources. With a single criterion, this is the classical Capital budgeting problem (see. e.g. Luenberger 1998). Multi-criteria methods in a portfolio selection setting have been studied, for example, by Mild (2003) and Stummer and Heidenberger (2003). Both models use incomplete information about the importance of criteria, but assume complete information on project outcomes.

In this paper, we introduce value intervals as a mean of describing incomplete information in criterion-specific outcomes of the projects. Such value intervals have been widely used in the field of MCDA (Salo and Hämäläinen, 1992, 2001; Salo and Punkka, 2003). We see that it is important to model incomplete information about the criterion-specific scores for two reasons. First, the use of point-estimates for the scores may weaken the DM’s willingness to use the results of the analysis, if the DM feels that the incompleteness of information is not accounted for. Second, estimation of exact values for the criterion-specific outcomes of every alternative may be time-consuming and the benefit gained from thorough analysis may be too modest.

The use of value intervals to model the incomplete information is appealing, since many criteria are not readily measurable with a probability distribution or even a point estimate. For example, if the portfolio selection problem includes such criteria as environmental or
societal impacts, the DM may be more willing to give interval valued outcomes that precise values. From a numerical point of view, the large number of portfolios limits the number of methods that are usable, especially if we want to implement an interactive decision support software. For example, with 40 alternatives we have $2^{40} \approx 10^{12}$ portfolios. Therefore, it is not possible to enumerate all feasible portfolios and analyze them as individual alternatives within the usual MCDM-framework or with Monte Carlo simulation.

This paper is structured as follows. Section 2 discusses earlier work in the field of Multi-Criteria capital budgeting and Section 3 introduces our framework. Section 4 develops algorithms for the determination of non-dominated portfolios. The efficiency of the algorithms is assessed with a simulation study in Section 5. Section 6 extends the use of decision rules to a portfolio selection setting. Section 7 presents an illustrative example of Multi-Criteria Capital Budgeting and Section 8 discusses future research directions.

2 Multi-Criteria Capital Budgeting

The classical capital budgeting problem is characterized by a set of alternatives (or projects) and a limited budget (see e.g. Luenberger 1998). Each project is defined by its cost and its outcome, which contributes to the objective function to be maximized. Mathematically a capital budgeting problem is a binary linear programming (LP) problem and several algorithms have been developed to solve such problems (see e.g. Ignizio and Cavalier 1994). Also, the model can readily extended to account for project synergies, multiple resources and so on.

In many cases, the DM may not have exact a priori preference information about the importance of criteria. Stummer and Heidenberger (2003) present a three-phase process for determining an optimal portfolio in a multi-criteria capital budgeting setting. They do not present an algorithm to find ‘good’ portfolios, but enumerate all possible portfolios and determine Pareto-optimal (non-dominated) ones from that set. This brute force approach limits the number of projects to about 30 and therefore they propose a screening procedure to cut down the size of the project set that is examined in the last two phases.
of the analysis. Since all Pareto-efficient portfolios are calculated in the second phase of the analysis, no a priori information of the DM’s preferences is needed. In the third phase the DM interactively sets threshold levels for the criterion-specific performances of the Pareto-efficient portfolios and iterates toward the optimal portfolio. The brute force approach allows the modelling of various project interdependencies, but the limitation of 30 projects can be seen as a major shortcoming.

Mild (2003) presents an algorithm for determining potentially optimal portfolios in a multi-criteria capital budgeting setting. The set of potentially optimal portfolios is the same as the set of Pareto-optimal portfolios when no a priori preference information is given. Mild’s approach also allows the use of various additional constraints (such as project interdependencies) and does not limit the number of projects to 30. Mild’s framework enables the use of various preference elicitation methods commonly used in the field of MCDA (e.g. Salo and Punkka, 2003), which can be seen as a major advantage.

All the methods above assume point estimates for the criterion-specific project outcomes. In this paper, we assume point estimates for the weights and consider the use of value intervals in the criterion-specific scores.

3 Framework

Let us assume that \( m \) projects \( x_1 \ldots x_m \) are evaluated against \( n \) criteria. The performance (score) of project \( j \) against criteria \( i \) is \( v_{ij} \). We use an additive value function (see e.g. Keeney and Raiffa, 1976) to aggregate these criterion specific scores into an overall value measure. The relative importance of \( i \)-th criteria is measured by the weight \( w_i \) and the overall value of project \( j \) with given criteria weights \( w = (w_1, \ldots, w_n) \) is

\[
V(x_j, w) = \sum_{i=1}^{n} w_i v_{ij}.
\]

A portfolio \( p \) is a set of projects, so that each project is either included in the portfolio of excluded from it, that is, \( p \subseteq \{x_1, \ldots, x_m\} \). The cost of project \( x_i \) is \( C(x_i) \) and the cost
of a portfolio is \( C(p) = \sum_{x_i \in p} C(x_i) \). The set of feasible portfolios with a given budget \( b \) is therefore \( F = \{ p \mid C(p) \leq b \} \). We assume that the value of a portfolio is the sum of values of its projects, that is

\[
V(p, w) = \sum_{x_j \in p} V(x_j) = \sum_{x_j \in p} \sum_{i=1}^{n} w_i v_{ij} .
\]

We model incomplete information about the criterion-specific scores through value intervals \( v_{ij} \in [\underline{v}_{ij}, \overline{v}_{ij}] \). With a given weight \( w \), the value of a portfolio belongs to the interval \( V(p, w) \in [\underline{V}(p, w), \overline{V}(p, w)] \), where the upper and lower bounds of the interval are defined as

\[
\underline{V}(p, w) = \sum_{x_j \in p} \underline{V}(x_j) = \sum_{x_j \in p} \sum_{i=1}^{n} w_i \underline{v}_{ij} ,
\]

\[
\overline{V}(p, w) = \sum_{x_j \in p} \overline{V}(x_j) = \sum_{x_j \in p} \sum_{i=1}^{n} w_i \overline{v}_{ij} .
\]

In this paper we assume that exact (point estimate) weights are known. For notational convenience we assume given \( w = w_1, \ldots, w_n \) such that \( \sum_{i=1}^{n} w_i = 1 \), \( w_i \geq \forall i \) and denote \( \overline{V}(p) = \overline{V}(p, w) \) and \( \underline{V}(p) = \underline{V}(p, w) \).

If the least possible value of portfolio \( p \) is greater than the highest possible value of \( p' \), then portfolio \( p \) dominates \( p' \) and we denote \( p \succeq p' \). Projects contained in both portfolios have to be discarded when calculating the lower bound of value interval for \( p \) and the upper bound of value interval for \( p' \), since these projects contribute the same amount of value to both portfolios.

**Definition 1** Assume two different non-empty portfolios \( p \) and \( p' \), then

\[
p \succeq p' \iff \underline{V}(p \setminus p') \geq \overline{V}(p' \setminus p) .
\]

If \( p \) does not dominate \( p' \) we use the notation \( p \nmid p' \). Note that this does not imply that \( p' \) dominates \( p \). With two portfolios \( p \) and \( p' \) one of the following holds:
i) $p > p' \land p' \not> p$

ii) $p' > p \land p \not> p'$

iii) $p' \not> p \land p \not> p'$

The following properties of portfolio value intervals imply dominance or the lack of it:

**Lemma 1** If $p$ and $p'$ are two different non-empty portfolios, then

$$
\bar{V}(p) > \bar{V}(p') \Rightarrow p' \not> p
$$

**Proof.** The result follows from $\bar{V}(p) > \bar{V}(p') \iff \bar{V}(p) - \bar{V}(p \cap p') > \bar{V}(p') - \bar{V}(p' \cap p) \iff \bar{V}(p \setminus p') > \bar{V}(p' \setminus p) \Rightarrow \bar{V}(p \setminus p') > \bar{V}(p' \setminus p) \iff p' \not> p \square$

**Lemma 2** If $p$ and $p'$ are two different non-empty portfolios, then

$$
\underline{V}(p) > \underline{V}(p') \Rightarrow p' \not> p
$$

**Proof.** The result follows from $\underline{V}(p) > \underline{V}(p') \iff \underline{V}(p) - \underline{V}(p \cap p') > \underline{V}(p') - \underline{V}(p' \cap p) \iff \underline{V}(p \setminus p') > \underline{V}(p' \setminus p) \Rightarrow \underline{V}(p \setminus p') > \underline{V}(p' \setminus p) \iff p' \not> p \square$

Notice, that Lemmas 1 and 2 imply that the feasible portfolio with the maximum upper bound is non-dominated as well as the feasible portfolio with the maximum lower bound.

**Lemma 3** If $p$ and $p'$ are two different non-empty portfolios, then

$$
\underline{V}(p) \geq \bar{V}(p') \Rightarrow p \succ p'
$$

**Proof.** The result follows from $\underline{V}(p) \geq \bar{V}(p') \Rightarrow \underline{V}(p) + \bar{V}(p \cap p') - \underline{V}(p \cap p') \geq \bar{V}(p') \iff \underline{V}(p \setminus p') \geq \bar{V}(p' \setminus p) \iff p \succ p' \square$

Lemma 3 is referred as *absolute dominance* in the MCDA literature, whereas the dominance relation of Definition 1 could be interpreted as *pairwise dominance* (Salo and
Hämäläinen, 1995). The dominance relation is transitive, which ensures that a partial preference order for the portfolios can be established.

**Theorem 1** If \( p, p' \) and \( p'' \) are three different non-empty portfolios, then

\[
p \succ p' \land p' \succ p'' \Rightarrow p \succ p''.
\]

**Proof.** Since \( p \) dominates \( p' \) we get \( V(p \setminus p') \geq V(p' \setminus p) \iff V(p \setminus (p' \cup p'')) + V((p \cap p'') \setminus p') \geq V(p' \setminus (p \cup p'')) + V((p' \cap p'') \setminus p') \) (1) and since \( p' \) dominates \( p'' \) we get \( V(p' \setminus p'') \geq V(p'' \setminus p') \) \( \iff V(p' \setminus (p'' \cup p)) + V((p' \cap p'') \setminus p'') \geq V(p'' \setminus (p' \cup p)) + V((p'' \cap p') \setminus p') \) (2). By adding the inequalities (1) and (2) and rearranging we get \( V(p \setminus (p' \cup p'')) + V((p' \cap p) \setminus p'') \geq [V((p' \cap p) \setminus p'') + V((p' \setminus (p' \cup p'')) \setminus p')] + [V((p' \setminus (p' \cup p'')) \setminus p') + V((p'' \setminus (p' \cup p)))]. \)

Since \( [V((p' \cap p) \setminus p'') - V((p' \setminus (p' \cup p'')) \setminus p')] \geq 0 \) and \( [V((p' \setminus (p' \cup p'')) - V(p' \setminus (p'' \cup p)))] \geq 0 \) we get \( V(p \setminus (p' \cup p'')) + V((p' \cap p) \setminus p'') \geq V((p' \cap p) \setminus p'') + V((p'' \setminus (p' \cup p)) \iff V(p \setminus p'') \geq V(p'' \setminus p) \iff p \succ p'' \). 

The set of non-dominated portfolios is an important concept, since a rational decision maker (i.e. a DM maximizing the overall value of the portfolio) would choose among the non-dominated portfolios. This is intuitive, since if the DM chooses a dominated portfolio, a portfolio with a certainly better outcome can always be found. The set of non-dominated portfolios is \( N = \{ p \mid \not\exists p' \text{ s.t. } p' > p \}. \)

## 4 Determination Non-dominated Portfolios

### 4.1 Linear Programming Approach

The search for non-dominated portfolios can be formulated as a series of binary LP problems. We start from the feasible portfolio with the maximum upper bound and recursively find a new portfolio with the next largest upper bound. A stopping rule for this search can be implemented by solving the portfolio with the maximum lower bound: when we find the first portfolio that has an upper bound lower than the maximum lower bound, we
know that all the remaining portfolios are absolutely dominated. While the resulting set of portfolios includes all non-dominated portfolios, it may also include dominated portfolios. Since a necessary condition for the existence of dominance (Lemma 1) is that the dominated portfolio has a smaller upper bound, it is sufficient to check if any of the previously generated non-dominated portfolios dominates the new portfolio. The algorithm for the determination of all non-dominated portfolios can be formulated as follows.

1. Let $p^{\text{max}} = \arg \max_{p \in P} \tilde{V}(p)$ and $p^{\text{min}} = \arg \max_{p \in P} \underline{V}(p)$

2. $p^{\text{new}} \leftarrow \arg \max_{p \in P} \tilde{V}(p)$ s.t. $\tilde{V}(p) \leq \tilde{V}(p^{\text{max}}) \land p \neq p^{\text{max}}$

3. If $\exists p \in N$ s.t. $p > p^{\text{new}}$, then $N \leftarrow N \cup p^{\text{new}}$

4. Let $p^{\text{new}} \leftarrow p^{\text{new}}$ and if $\tilde{V}(p^{\text{max}}) > \underline{V}(p^{\text{min}})$ GOTO 2 else STOP.

All optimization tasks in the algorithm are binary LP problems. Let $P$ be an $m$ dimensional binary vector, where $P_i = 1$ if $x_i \in p$ and $P_i = 0$ otherwise. Furthermore, let $\tilde{V}$ be a vector with the upper bounds of the overall value of individual projects and $\underline{V}$ the lower bounds respectively. By letting $C$ be a vector of project costs the LP-problem is $\max \tilde{V}^T P$ (or $\underline{V}^T P$) subject to $C^T P \leq b$. We can easily add further linear constraints to account for multiple limited resources, project synergies and so on. The constraint $p \neq p^{\text{max}}$ in Step 2, which guarantees that a new portfolio is generated, can be formulated as a linear constraint $P_i P^{\text{max}} \geq 1$, where $P^{\text{max}}$ is created from $P^{\text{max}}$ by replacing ones with zeros and vice versa. This ensures that $P$ contains at least one project that is not included in $P^{\text{max}}$.

Step 3 checks if any of the already generated non-dominated portfolios dominates the new candidate portfolio. By Lemma 1 and Theorem 1, the search for dominance can be limited to the set $N$. Theorem 1 implies that if portfolio $p$ is dominated, then at least one of the dominating portfolios is itself non-dominated. Furthermore, Lemma 1 implies that this portfolio must have a greater upper bound for overall value. Since our algorithm finds the non-dominated portfolios in a descending order of upper bounds, the limited dominance check in step 3 is sufficient. The termination of the algorithm in step
4 is consequence of Lemma 3: Any portfolio with an upper value smaller than \( \hat{V}(p^{\min}) \) is absolutely dominated by \( p^{\min} \).

The algorithm could be made more efficient by adding constraints to eliminate portfolios that are dominated by the non-dominated portfolios found earlier. This approach would guarantee that all candidate portfolios are non-dominated and therefore the dominance check in step 3 would no longer be needed. The dominance relation in Definition 1 can be formulated \( p' \neq p \iff \hat{V}(p' \setminus p) < \hat{V}(p \setminus p') \iff \hat{V}(p) - \hat{V}(p \cap p') < \hat{V}(p') - \hat{V}(p' \cap p) \iff \hat{V}(p' \cap p) - \hat{V}(p \cap p') + \hat{V}(p) < \hat{V}(p') \). With the binary notation the intersection of portfolios \( p \cap p' \) is \( \mathcal{P} \times \mathcal{P}' \), where \( \times \) is componentwise multiplication of the respective vectors. We introduce a real valued variable \( z \) and constrain it to have the value of \( \hat{V}(p) \), that is \( z = \mathcal{V}^T \mathcal{P} \). The resulting linear constraint \( \hat{V}(\mathcal{P}^T \times \mathcal{P}') - \mathcal{V}(\mathcal{P}^T \times \mathcal{P}') + z \leq \hat{V}(p') \iff \mathcal{P}^T((\hat{V} - \mathcal{V}) \times \mathcal{P}') + z \leq \hat{V}(p') \) excludes all portfolios dominated by \( p' \). One constraint is needed for every non-dominated portfolio found previously. This algorithm is formulated as follows.

1. Let \( p^{\max} = \arg \max_{p \in \mathcal{F}} \)

2. \( p^{\text{new}} \leftarrow \arg \max_{p \in \mathcal{F}} \hat{V}(p) \text{ s.t. } \hat{V}(p) \leq \hat{V}(p^{\text{max}}) \land p' \neq p \forall p, p' \in N \)

3. If no feasible solution in step 2 then STOP

4. \( N \leftarrow N \cup p^{\text{new}}, p^{\max} \leftarrow p^{\text{new}} \) and GOTO 1

Although formally elegant, the LP-based approach is not efficient. The solution to each LP-problem is solved using an algorithm that actually generates a number of good candidates for non-dominated portfolios, but only the optimum is returned. A lot of effort is therefore in vain. On the other hand, in this approach we can readily include additional constraints.

### 4.2 Tree Approach

The classic Capital budgeting problem can be formulated as a binary linear programming problem (see e.g. Luenberger 1998). A common way to solve binary LP-problem is to
use a tree where each node represents a specific value of the decision variable (Branch and Bound). The fathomed nodes, e.g. nodes that can not produce a better objective function value than the optimum already found or contain only infeasible solutions, are not analyzed (see e.g. Ignizio and Cavalier 1994). In our approach, we develop a tree algorithm that excludes as many dominated portfolios as possible without slowing down because of too many constraints.

![Diagram](image)

**Figure 1:** All non-dominated portfolios lie in the gray area

Figure 1 illustrates the problem. Each portfolio corresponds to a point on the two dimensional plane defined by the cost of a portfolio and the upper bound value of a portfolio. We are interested in portfolios that are feasible (i.e., their cost does not exceed the budget). We may also constrain the region of interest simply by solving the maximum upper bound of overall value interval in the set of feasible portfolios as a solution of a single LP-problem. Since we are interested only in non-dominated portfolios, Lemma 3 gives a lower bound for the upper values of non-dominated portfolios: every portfolio with a upper bound value lower than $V^{\text{max}}$ is absolutely dominated. Also, a lower bound for the cost of a non-dominated portfolio can be derived, since a non-dominated portfolio has to be complete in the sense that the remaining budget cannot exceed the greatest cost of a
single project. If we could afford one more project, this would imply that the new full portfolio would dominate the first one.

We assume that the projects are arranged in a descending order according to their overall upper bound values, that is \( i < j \Rightarrow \bar{V}(x_i) \geq \bar{V}(x_j) \). The decision variable in our framework is an \( m \)-dimensional binary vector \( P \), where the \( i \)-th element indicates whether or not \( x_i \) is included in portfolio \( p \). All portfolios \( P \) can be described as nodes of a tree (see Figure 2). In level \( j \) of tree we can either include project \( x_j \) to the portfolio (1) or exclude it (0). Since several nodes of the tree correspond to the same portfolio, we use the variable \( k \) to identify which node the algorithm is analyzing. To summarize, the state of the algorithm is defined by the current portfolio \( p \) and by the value of \( k \). The main idea of the algorithm is the following.

1. Let portfolio \( p \) be empty and set \( k \) to zero. Furthermore, mark the subtree of each node not fathomed.
2. If the subtree of the current node (defined by \( p \) and \( k \)) is not fathomed then goto 3 else goto 4.
3. Move downwards in the tree (\( k \) increases) and include the first project (take the 1-branch) that is affordable with the unspent budget. Then goto 2.
4. If the portfolio \( p \) is empty then stop, since all candidate portfolios have been found.
5. If the portfolio \( p \) is full (i.e. there exists no project that is not included in \( p \) that can be afford with the unspent budget), add \( p \) to the set of candidate portfolios.
6. Set \( k \) to indicate the largest index of projects included in the portfolio (i.e. if you are in a 0-node move upwards until you reach a 1-node). Remove project \( x_k \) from the portfolio (i.e. move one branch up and then take the zero branch down).

The algorithm generates a set of candidate projects, which includes all non-dominated portfolios. These can be found by checking all candidate portfolio pairs for dominance. In order to determined which subtrees are fathomed we define the \( \Upsilon_i \)-function.
Definition 2 Assume projects $x_1, \ldots, x_m$ and the available budget is $c \geq 0$. The maximum upper value that can be achieved using projects $x_i \ldots x_m$ and with $c$ resources is

$$\Upsilon_i(c) = \max_p \hat{V}(p) \text{ s.t. } C(p) \leq c \land \forall x_j \in p \ j \geq i.$$ 

For notational convenience we define $\Upsilon_{m+1}(x) = 0 \ \forall x$. In order to use this function to rule out fathomed subtrees and to speed up the calculation of non-dominated portfolios, the construction of the function cannot be too complex. It turns out that we can recursively construct the functions by starting with $\Upsilon_m$.

Theorem 2 Assume projects $x_1, \ldots, x_m$, $1 < i \leq m$ and $c \geq 0$, then

$$\Upsilon_m(c) = \begin{cases} 0, & \text{if } c < C(x_m) \\ \hat{V}(x_m), & \text{if } c \geq C(x_m) \end{cases}$$

$$\Upsilon_{i-1}(c) = \max \{ \Upsilon_i(c), \Upsilon_i(c - C(x_{i-1})) + \hat{V}(x_{i-1}) \}$$

Proof. For $\Upsilon_m$ trivial, since we we either have enough resources for $x_m$ or not $\square$. For $\Upsilon_i$, $i < m$ we have $\Upsilon_i(c) = \max_p \hat{V}(p) \text{ s.t. } C(p) \leq c \land \forall x_j \in p \ j \geq i$ (1) and $\Upsilon_{i-1}(c) = \max_p \hat{V}(p) \text{ s.t. } C(p) \leq c \land \forall x_j \in p \ j \geq i - 1$ (2). Assume that with a given $c^*$ we have
that maximizes (2). Trivially if \( x_{i-1} \notin p^* \) then \( \Upsilon_i(c^*) = \Upsilon_{i-1}(c^*) \). On the other hand, if \( x_{i-1} \in p^* \) then \( \Upsilon_{i-1}(c^*) = \tilde{V}(x_{i-1}) + \max_p \tilde{V}(p) \) s.t. \( C(p) \leq c - C(x_{i-1}) \land \forall x_j \in p \) \( j \geq i \iff \Upsilon_{i-1}(c^*) = \tilde{V}(x_{i-1}) + \Upsilon_i(c^* - C(x_{i-1})) \).

Functions \( \Upsilon_1, \ldots, \Upsilon_m \) are created before the start of the algorithm. The algorithm uses these functions in two ways. First, consider that we have a partly constructed portfolio \( p \) such that \( i < k \) \( \forall x_i \in p \) and that we have \( x \) resources left, that is \( c = b - C(p) \). Now we can determine using Lemma 3 that there is no need to analyze the subtree of \( p \) if \( \Upsilon_k(c) < \Upsilon_{\text{max}} - \tilde{V}(p) \). This means that all portfolios that include \( p \) are (absolutely) dominated by the portfolio with the greatest lower bound of overall value. Second, if we have a feasible portfolio \( p = p_1 \cup p_2 \) so that \( \forall x_i \in p_1 \) \( i < k \) and \( \forall x_i \in p_2 \) \( i \geq k \land x_k \in p_2 \). Furthermore, let \( \Delta = \Upsilon(p_2) \). The subtree of node defined by \( p_1 \) and \( k \) is fathomed if \( \Delta > \Upsilon_{j+1}(b - C(p_1)) \). This means that all portfolios that include \( p_1 \) and do not include \( x_k \) are dominated by \( p \).

Now we may mathematically formulate the algorithm.

1. Set \( p = \emptyset \), \( N = \emptyset \), \( k = 0 \), \( \Delta = 0 \) and \( \Upsilon_{\text{max}} = \max_p \tilde{V}(p) \) s.t. \( C(p) \leq b \).

2. If \( \max(\Delta, \Upsilon_{\text{max}} - \tilde{V}(p)) < \Upsilon_{k+1}(b - C(p)) \) GOTO 3 else GOTO 4.

3. \( k \leftarrow \min i \ s.t. C(x_i) + C(p) \leq b \land i \geq k + 1 \), \( p = p \cup x_k \) and \( \Delta \leftarrow \max(\Delta - \tilde{V}(x_k), 0) \), GOTO 2.

4. If \( p = \emptyset \) then STOP.

5. If \( \exists x_i \notin p \) s.t. \( C(x_i) + C(p) \leq b \) then \( M \leftarrow M \cup p \).

6. \( k \leftarrow \max i \ s.t. x_i \in p \), \( p \leftarrow p \setminus x_k \) and \( \Delta \leftarrow \Delta + \Upsilon(x_k) \). GOTO 2.

The algorithm creates a set of candidate portfolios \( M \). By checking the set \( M \) for dominance relations and collecting all portfolios that were not dominated by any other portfolio, we get the set of non-dominated portfolios \( N \).

The variable \( \Delta \) measures how much the lower bound of portfolio’s overall value has decreased since the last candidate portfolio was created. In step 3 the algorithm adds
projects to the portfolio in a descending order of upper bounds. This is done only if the condition in step 2 holds, that is, if using projects \(x_{k+1} \ldots x_m\) an upper value that is greater than the maximum lower value and greater than \(\Delta\) can be achieved (i.e. subtree is not fathomed). If the subtree is fathomed we extract projects from the portfolio in step 6. Note that variable \(x_k\) is not included in the portfolio after step 6, which guarantees that we try to make up for the lower bound of overall value lost with the upper bounds of projects \(x_{k+1} \ldots x_m\) (i.e. \(x_k\) is not included). If this is not possible, the algorithm extract projects from the portfolio until it becomes possible. In step 5 the algorithm adds a new candidate portfolio to the set \(M\) only if the portfolio is full in the sense that the budget constraint does not allow inclusion of new projects.

5 Computational Efficiency

In this section we analyze how fast the different algorithms are in detecting all non-dominated portfolios. We compare four different algorithms, (1) total enumeration of all possible portfolios (brute force), (2) the LP-based algorithm in Section 4.1 without dominance constraints (LP 1), (3) with the dominance constraints (LP 2) and (4) the tree approach in Section 4.2 (tree). We also consider how the general nature of the problem depends on the size and structure of project data.

5.1 Experimental Design

As a hypothesis, we assume that the number of all projects \((m)\) and the average number of projects included in a full portfolio are the key factors that determine how time-consuming the problem is. More projects and a larger budget lead to a greater number of feasible and non-dominated portfolios. These two factors mean longer calculation times. The problem also becomes more time-consuming if the projects are quite similar. By this we mean that their costs are close to each other and their value intervals are overlapping.

If we set the expected cost of a project to one and define the budget \(b\) to be an integer, we get a rough estimate on the computational effort needed by calculating the number
of combinations $m!/(b!(m-b)!))$. Therefore, we draw the project costs from a log-normal
distribution, that is $C(x_i) = e^{X_i} \forall i = 1, \ldots, m, \ X_i = N(0,\sigma_c)$. Here $\sigma_c$ is an adjustable
parameter. The smaller the variance $\sigma_c$, the more computational effort is needed.

To maintain consistency between the cost of a project and its performance, that is, a more
expensive project can be expected to have a better performance, the average performance
is defined as a function of the cost $R(x_i) = C(x_i)e^{X^2} \forall i = 1, \ldots, m, \ X_2 = N(0,\sigma_r)$.
Again, $\sigma_r$ is an adjustable parameter with the same effect as $\sigma_c$.

Finally, we get the value interval for the overall value of a project from the following for-
mulas: $\bar{V}(x_i) = (1-d\cdot U(0,1))R(x_i) \forall i = 1, \ldots, m$ and $\hat{V}(x_i) = (1+d\cdot U(0,1))R(x_i) \forall i = 1, \ldots, m$, where $d$ is a parameter defining the maximum interval and $U(0,1)$ is a random
variable drawn from a uniform distribution between 0 and 1.

We used Matlab as a simulation platform. For each selected parameter combination, we
generated 50 problem instances and for each of them, we calculated the non-dominated
portfolios with all four algorithms. The parameter combinations where chosen so that
we could make some observations on the effect of each parameter on the computational
effort.

Comparing linear programming approaches against the binary tree approach, the time
consuming parts in LP and LP2 are the LP-problems. In our platform this part was
performed with a LPMEX-library written in C-programming language. On the other
hand, the tree-algorithm was written in Matlab, which is translated in real time and is
considerably more time consuming than any real programming language. A Java-written
version of the tree-algorithm is at least a hundred times faster than the one written with
Matlab, and therefore we also implemented the binary algorithm with Java.

5.2 Results

The results of the simulation study are in table 1. In the column referring to the tree
algorithm written in Java, the block ‘generated portfolios’ is in fact the number of non-
dominated portfolios to give an idea of the size of the problem. LP2 generates only
non-dominated portfolios.
Table 1: Simulation results

<table>
<thead>
<tr>
<th>m</th>
<th>time in seconds</th>
<th>Mean</th>
<th>LP</th>
<th>LP2</th>
<th>TAO</th>
<th>TAO (Java)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>5</td>
<td>0.02</td>
<td>1.26</td>
<td>0.17</td>
<td>0.32</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>0.2</td>
<td>0.06</td>
<td>1.13</td>
<td>0.05</td>
<td>0.11</td>
<td>-</td>
</tr>
<tr>
<td>0.2</td>
<td></td>
<td>0.00</td>
<td>2.70</td>
<td>8.60</td>
<td>28.00</td>
<td>-</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>m</th>
<th>time in seconds</th>
<th>Mean</th>
<th>LP</th>
<th>LP2</th>
<th>TAO</th>
<th>TAO (Java)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>5</td>
<td>1.37</td>
<td>4.29</td>
<td>0.37</td>
<td>0.48</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>0.2</td>
<td>0.30</td>
<td>2.89</td>
<td>0.33</td>
<td>0.16</td>
<td>-</td>
</tr>
<tr>
<td>0.2</td>
<td></td>
<td>0.00</td>
<td>40.40</td>
<td>6.75</td>
<td>42.40</td>
<td>-</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>m</th>
<th>time in seconds</th>
<th>Mean</th>
<th>LP</th>
<th>LP2</th>
<th>TAO</th>
<th>TAO (Java)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>5</td>
<td>36.30</td>
<td>26.50</td>
<td>5.60</td>
<td>1.77</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>0.2</td>
<td>16.10</td>
<td>30.00</td>
<td>0.58</td>
<td>1.37</td>
<td>-</td>
</tr>
<tr>
<td>0.2</td>
<td></td>
<td>0.00</td>
<td>81.00</td>
<td>5.53</td>
<td>73.20</td>
<td>-</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>m</th>
<th>time in seconds</th>
<th>Mean</th>
<th>LP</th>
<th>LP2</th>
<th>TAO</th>
<th>TAO (Java)</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>7</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>7</td>
<td>0.3</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>30.00</td>
</tr>
<tr>
<td>0.3</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>42.05</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>m</th>
<th>time in seconds</th>
<th>Mean</th>
<th>LP</th>
<th>LP2</th>
<th>TAO</th>
<th>TAO (Java)</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>7</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>7</td>
<td>0.2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>0.2</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>m</th>
<th>time in seconds</th>
<th>Mean</th>
<th>LP</th>
<th>LP2</th>
<th>TAO</th>
<th>TAO (Java)</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>10</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>10</td>
<td>0.3</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>0.3</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>m</th>
<th>time in seconds</th>
<th>Mean</th>
<th>LP</th>
<th>LP2</th>
<th>TAO</th>
<th>TAO (Java)</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>8</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>8</td>
<td>0.2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>0.2</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Clearly, the number of projects is the main factor that determines the computational effort needed. For example, the first and third simulation rounds have the same parameter values, but the number of projects increases from 10 to 15. This results in a considerable increase in calculation time. The variances (the similarity of the projects) affects the calculation time needed. For example, in the case of 15 projects with a budget of 5 the calculation time increases, when the variances are smaller.
Comparing the algorithms, we see that brute force and LP1-algorithm are ineffective. Also, based on the expected running time the binary algorithm performs better than LP2 in large problems and the standard deviation of the running time is remarkably larger for LP2. Notice that the Java-written binary tree algorithm is approximately one thousand times faster than the LP2, which is written mostly in C.

6 Analysis of Non-dominated Portfolios

As discussed above, a DM maximizing the expected value will choose one of the non-dominated portfolios. Since the number of non-dominated portfolios can be quite large, we now discuss some methods to analyze the set of non-dominated portfolios.

6.1 Decision Rules

In MCDA, the non-dominated alternatives are usually assessed with decision rules (Salo and Hämäläinen 2001). These rules do not make any assumptions about the probability distributions behind the value intervals, but only require the upper and lower bound of the value interval. The maximax-rule chooses the alternative with the maximum possible outcome, the maximin-rule the alternative that has the greatest lower bound for the outcome and the central values-rule chooses the alternative with the highest average of lower and upper bounds. These three rules can be directly applied in our framework. Mathematically the rules are formulated as follows:

maximax : \( \arg \max_{p \in N} \tilde{V}(p) \)

maximin : \( \arg \max_{p \in N} V(p) \)

central values : \( \arg \max_{p \in N} [\tilde{V}(p) + V(p)] \)

Fourth decision rule often used in MCDA is the minmaxregret-rule. This rule minimizes the maximum possible loss of value. In our framework we have to consider the fact that
the realization of a project's outcome is the same to all portfolios in which it is included. This leads to a similar formulation as the one with dominance,

\[
\text{minimax regret} : \quad \arg \min_{p \in N} \max_{p' \in N \setminus p} [\bar{V}(p') - \bar{V}(p \cup p')]
\]

Some simulation studies have been conducted to evaluate the effectiveness of decision rules in a classical MCDA setting (Punkka 2003; Salo and Punkka 2003). Since our portfolio setting differs from the selection of one alternative, further research should be conducted to assess the decision rules in a portfolio setting.

### 6.2 Screening

Preference elicitation methods are often able to use incomplete information about the DM’s preferences and the criteria specific scores. This has encouraged many authors (e.g., Punkka 2003) to propose to use these methods for screening of alternatives. When selecting one of the alternatives, screening means that incomplete preference statements may lead to a set of non-dominated alternatives that is considerably smaller than the set of all alternatives. A more thorough analysis can then be applied to the non-dominated alternatives to derive more dominance relations.

In our portfolio framework, projects that are not included in any of the non-dominated portfolios can be left out and other projects may be further analyzed. We may also discard projects that are included in all non-dominated portfolios (core projects) out of the further analysis and concentrate on the analysis of the borderline projects. The use of methods of incomplete information for screening is a appealing, since the cost of the analysis versus the utility gained from has raised interest in the field of both MCDA (Salo and Hämäläinen, 2003) and portfolio analysis (Clemen and Kwit, 2001).
7 Illustrative Example

In this section we present a portfolio selection case to illustrate the use of our framework. The screen shots are from a portfolio selection software that implements our framework and the tree algorithm.

Consider for example a national research agency that launches a research programme that funds research in the field of innovative future energy production and has to decide which research projects to fund from a large number of projects proposals. The aim of the national technology agency is not simply to maximize the expected profit, but to select a project portfolio that performs well against multiple criteria. These criteria are economic impacts and societal impacts.

Twelve experts are asked to evaluate each of the 30 projects against the two criteria. Value intervals are set so that they cover all the expert judgements, that is, the lower bound of a value interval is the lowest score given and the upper bound is the greatest score given. The cost of each project is given in the project proposal. The data is given in Figure 3 and Figure 4.
Figure 3: Project data

Figure 4: Overall value intervals and costs (dots) of projects
A negotiation between the experts is carried out to find consensus about the criteria weights. It is finally agreed that both criteria should have equal weights \((w = (1/2, 1/2))\). The software then indicates that there exists 13 non-dominated portfolios. Analyzing these non-dominated portfolios, we find that 5 projects are not included in any of the non-dominated portfolios (projects 3, 6, 12, 13 and 16) and on the other hand 3 projects (2, 15 and 20) are included in all of them (see Figure 5). A more thorough analysis is carried out for the remaining projects (that are included in some but not all of the non-dominated portfolios) to get more exact estimates of criteria specific scores (see Figure 6). Only five non-dominated portfolios remain (see figure 7). Based on the decisions rules portfolio #1 is selected (see Figure 8).

![Figure 5: Core projects. Rate of inclusion in the non-dominated portfolios](image)

22
Figure 6: Project data after further analysis

Figure 7: Non-dominated portfolios
Figure 8: Value intervals of non-dominated portfolios
8 Discussion and Conclusions

Our algorithm can easily be extended to solve problems with additional constraints. Additional limited recourses can be accounted for by generating a Υ function for every resource. The algorithm considers a subtree fathomed if any of the Υ functions implies that with the given resources the demand for an overall upper value can not be met.

Project synergies can be modelled simply by generating the Υ functions with the assumption that the best synergies will always realize. This would increase the number of candidate portfolios generated as the constraints become less restrictive, but assuming that the synergies are not massive compared to the project outcomes, the increased computational effort should remain moderate. The dominance relation would still be valid, although the notation used would have to be changed.

Logical constraints, such as ‘only one of these projects can be selected’ or ‘at least two projects from these five projects have to be selected’, are more difficult to implement. One way to deal with these constraints is to discard the Δ variable and first generate all the portfolios that meet the requirement for a upper value greater than the maximum lower value. While performing the dominance check for the candidate portfolios we could discard the ones that do not meet logical constraints.

One weakness of our framework is that it requires specific point-estimates about the criteria weights. Stummer and Heidenberger (2003) emphasize that a multi-criteria portfolio selection model should not require too extensive a priori preference information. Mild modelled the DM’s preferences with a hierarchial value-tree using incomplete preference information concerning the criteria weights and developed an algorithm to find the potentially optimal portfolios. If no statements are given the set of potentially optimal portfolios is the same as Pareto-efficient in Summer and Heidenberger framework. Mild used point-estimates for criteria specific outcomes but allowed a region of feasible weights. Our framework on the other hand uses exact weights and intervals for scores. Combining these two would be fertile. Assume we have a set of feasible weights \( S \) and weight vector \( w \in S \). Using our framework we can calculate non-dominated portfolios using weight vector \( w \). These portfolios are potentially optimal in \( S \), since any of them may have the
greatest realization of overall value, if the DM’s ‘true’ preferences are weights corresponding to $w$. Further research should be conducted to extend our algorithm to find all the potentially optimal portfolios in $S$. A heuristic approach would be to randomly generate weight vectors from the set $S$ and to calculate non-dominated portfolios with each $w$.

Stummer and Heidenberger propose the use of various heuristic methods to detect the set of Pareto-efficient portfolios in complex multi-criteria capital budgeting problem. This may be problematic, since heuristic methods do not guarantee that all solutions of interest are found. If a stakeholder notices that a portfolio analysis model does not find one of the optimal portfolios, it will surely weaken the stakeholder’s willingness to trust the analysis.

In this paper we have presented a framework for the use of value-intervals in multi-criteria capital budgeting. We have also presented an exact algorithm that detects all the non-dominated portfolios and evaluated the algorithms efficiency. Although further research is needed, we believe that our work is an important basis for the development of a complete multi-criteria capital budgeting model, that uses incomplete information about both criteria weights and criteria specific scores and is able to handle several types of additional constraints.

9 References


