Master's Degree in Informatics Engineering

Dissertation

Implicit Enumeration for Representation Systems in Multi-objective Optimisation

September, 2015

Alexandre Daniel Borges de Jesus
ajesus@student.dei.uc.pt

Advisors

Prof. Dr. Luís Paquete
Prof. Dr. José Rui Figueira

FCTUC DEPARTMENT
OF INFORMATICS ENGINEERING
FACULTY OF SCIENCES AND TECHNOLOGY
UNIVERSITY OF COIMBRA

Master's Degree in Informatics Engineering

Dissertation

Implicit Enumeration for Representation Systems in Multi-objective Optimisation

September, 2015

Alexandre Daniel Borges de Jesus
ajesus@student.dei.uc.pt

Advisors

Prof. Dr. Luís Paquete
Prof. Dr. José Rui Figueira

Jury

Prof. Dr. Carlos Fonseca
Prof. Dr. Jorge Cardoso

FCTUC DEPARTMENT
OF INFORMATICS ENGINEERING
FACULTY OF SCIENCES AND TECHNOLOGY
UNIVERSITY OF COIMBRA

Abstract

The main focus of this thesis is the design and analysis of algorithms to find a representative subset, with a given cardinality, of the Pareto-optimal set for the unconstrained bi-objective knapsack problem, according to some notions of representation quality. The representative subset should be obtained without prior knowledge of the Pareto-optimal set. Two main algorithms are discussed in this thesis. The first reformulates the recurrence of the existing Nemhauser-Ullman algorithm for the unconstrained bi-objective knapsack problem by selecting a representative subset at each recursive step. The second by pruning solutions that may not contribute to find the optimal representation based on the sum of the weights or the set of supported solutions. Analysis on the time and error regarding the uniformity, coverage and ϵ-indicator is performed.

Keywords: Unconstrained bi-objective knapsack problem, Nemhauser-Ullman algorithm, Representative subset, Representation quality

Resumo

O principal foco desta tese assenta em desenhar e analisar algoritmos que permitam encontrar um subconjunto representativo, com uma dada cardinalidade, do conjunto óptimo de Pareto para o problema de knapsack bi-objectivo sem restrição, de acordo com alguma noção de qualidade da representatividade. Este subconjunto deve ser encontrado sem conhecimento prévio do conjunto óptimo de Pareto. Dois algoritmos são discutidos neste relatório. O primeiro reformula a recorrência do algoritmo de Nemhauser-Ullman para o problema de knapsack bi-objectivo sem restrição, adicionado a cada etapa recursiva uma restrição de cardinalidade. O segundo algoritmo corta soluções que podem não contribuir para a representação óptima através da soma dos pesos ou do conjunto de soluções suportadas. É ainda feita uma análise de tempo e qualidade de acordo com os indicadores de uniformidade, cobertura e ϵ.

Palavras chave: Problema de knapsack bi-objectivo sem restrição, Subconjunto representativo, Algoritmo de Nemhauser-Ullman, Qualidade da representação

Acknowledgements

First of all, I would like to thank my family, particularly my parents and brother, for supporting me, providing a great familiar environment, and helping me get where I am today.

Moreover, I would like to thank my advisers, Prof. Luís Paquete and Prof. José Rui Figueira, for their guidance, readiness and effort to make my thesis better as well as trusting my abilities.

I would also like to thank the ECOS lab and all its current and former members for providing a great work environment.

To my friends, thank you for all the support and the leisurely moments throughout the years.

Lastly, I would like to thank the Instituto Superior Técnico de Lisboa for supporting my research under grant BL80/2014 CEG-IST.

Contents

List of Figures

Chapter 1

Introduction

Optimisation problems arise in everyday situations. Consider, for example, a person that wants to find the fastest route between two cities. More formally, the problem can be defined as, given a set of cities and distances between every pair of cities, find a path that minimises the travelling time. However, in many situations, real-life problems tend to have more than one objective, which makes them more complex to solve. For the problem above, the minimisation of monetary cost could be an additional objective, due to tolled highways. Problems with two or more objective are called *multi-objective optimisation problems*.

One difficulty with multi-objective optimisation problems is the conflicting nature of the objectives. For example, the faster routes tend to use highways, which usually incur tolls. Therefore, in this scenario, finding the fastest route would conflict with finding the least expensive one. As a result, it may be impossible to find a single solution that satisfies all the objectives. One way to overcome this difficulty is to consider the *Pareto set*, a set of *efficient solutions*, for which does not exist other solution that improves all the objectives simultaneously; a solution for the problem above is said to be efficient if no other solution exists that is both faster and cheaper. Figure 1.1 shows an hypothetical set of solutions, in the objective space, for the problem described above, where blue points correspond to efficient solutions, whereas red points correspond to non-efficient.

Usually, solving a multi-objective optimisation problems consists of two steps. First, the Pareto set is found. Algorithms that find this set (or an approximations of it) has been an important area of study. See Ehrgott and Gandibleux [5] for a general overview. Afterwards, the decision maker is presented this set and chooses a solution according to its preferences. In the problem above, the decision maker could choose a solution that has low monetary cost but takes more time, if money is an essential concern, or a

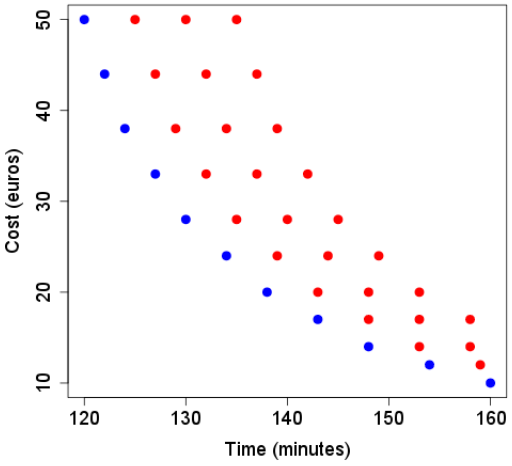

FIGURE 1.1: Illustration of efficient solutions, in blue, and remaining feasible solutions, in red, in the objective space

solution which is faster but more expensive, if time is more important, or even something in between to balance both time and money.

However, multi-objective optimisation problems may generate a large amount of efficient solutions, thus, overwhelming the decision maker. One way to overcome this problem is to find a smaller number of efficient solutions. Furthermore, it is also important to find a meaningful representative subset since if a poor representation is given to the decision maker, the quality of the solutions to choose from gives few information about the trade-off, as is depicted in Figure 1.2. However, if the quality of the representation is good, the variety of the solutions is much more interesting, as is shown in Figure 1.3.

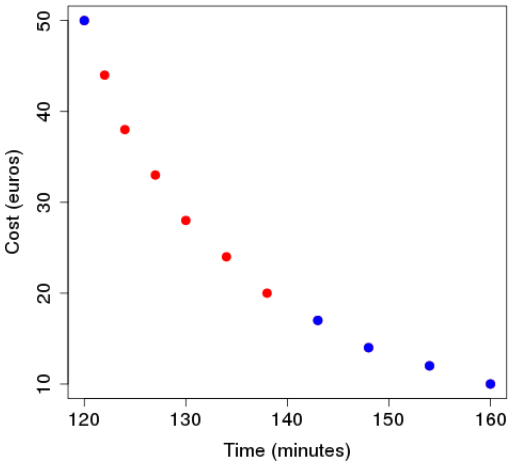

FIGURE 1.2: Illustration of a Pareto set of solutions (in blue and red), and a subset with poor representation (in blue), in the objective space

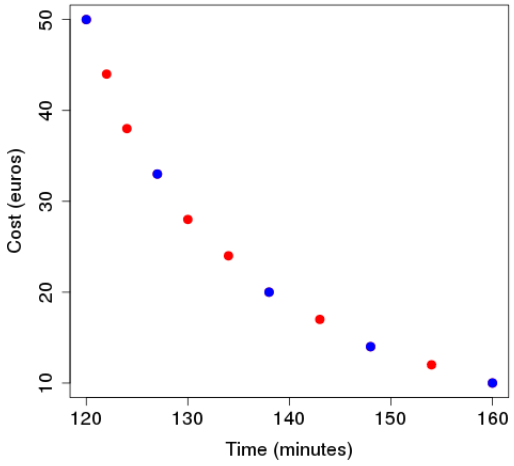

FIGURE 1.3: Illustration of a Pareto set of solutions (in blue and red), and a subset with good representation (in blue), in the objective space

It is possible to define the quality of a representation, in the objective space, using some existing measures. Two such measures are *uniformity* and *coverage*, which were introduced by Sayın [14]. Uniformity, measures the minimum distance between any two chosen points; a higher value of uniformity indicates that the points are more evenly spread. Coverage compares the distance between the uchosen and closest chosen points. A lower value means a better representation. A third measure which is also relevant to this thesis is the ϵ-indicator [20], which is used for assessing the performance of heuristic approaches. In a recent study by Vaz et al. [17], this measure was used as a way to evaluate representations.

Finding a representation of the efficient set for the three measures above can be solved in polynomial time in the bi-objective case by the approaches discussed in Vaz et al. [17]. However, for some problems, it is not possible to find the efficient set in a reasonable amount of time, for instance, because the set may be exponentially large. For those cases, obtaining a representative set without pre-computing the set of efficient solutions is a more interesting alternative.

Some procedures exist, albeit relatively few, to find a representative set with quality guarantees, e.g. the works by Sayin [15], Hamacher et al. [8], Eusébio et al. [6], and Sayın and Kouvelis [16].

The main contribution of this thesis is the development of methods capable of finding a representative subset with a target cardinality. Furthermore, studies are performed on the quality of the uniformity, coverage and the ϵ-indicator measures. While the theoretical results are not particularly appealing in certain scenarios, the experimental results show that the methods presented can be quite effective. This work differs from

[8], [6], and [16] since these do not allow the choice of cardinality. It also differs from [15], which allows choice of cardinality, but only performs an analysis regarding the coverage measure.

The algorithms proposed in this thesis are applied to a specific problem which has not been studied in this context, the unconstrained bi-objective knapsack problem, which has both minimising and maximising objectives.

The thesis is structured as follows. In Chapter 2 the necessary definitions and notation are presented. In Chapter 3 the literature on finding representations of the Pareto set with quality guarantees is reviewed. In Chapter 4 both existing and new algorithms relevant to the thesis are presented. In Chapter 5 the methodology for the experimental analysis is presented. In Chapters 6, 7, and 8 an analysis on uniformity, coverage and ϵ-indicator respectively is made for the algorithms proposed in Chapter 4. Furthermore, improvements to the algorithms are proposed for each measure. In Chapter 9 a final discussion and some ideas for future work are presented.

Chapter 2

Preliminaries

In this chapter, some necessary definitions are introduced. In Section 2.1 the unconstrained bi-objective knapsack problem is presented. In Section 2.2 the notions of optimality are defined. Lastly in Section 2.3 three quality measures for representations are presented.

2.1 Unconstrained bi-objective knapsack

In a general multi-objective optimisation problem there are $t \geq 2$ objective functions, which can be either minimising or maximising. For the purposes of this thesis a specific problem with $t = 2$ objectives is considered, the unconstrained bi-objective knapsack problem. The problem is related to the knapsack problem, where there is maximising objective on the profit and the constraint on the weight is transformed into a minimising objective.

More formally, consider a set of items I, where each item i has an associated profit p_i and weight w_i, $i = 1, \ldots, n$ where $n = |I|$. The two objective functions are defined as:

$$\max \; P(x) = \sum_{i=1}^{n} p_i x_i \tag{2.1}$$

$$\min \; W(x) = \sum_{i=1}^{n} w_i x_i \tag{2.2}$$

where x represents a feasible solution to the problem, as a subset of items where $x_i = 1$ denotes that item i is in the solution, and $x_i = 0$ otherwise. The set of all feasible solutions is denoted by X.

2.2 Notions of Pareto Optimality

Associated to each solution $x \in X$, an objective vector $f(x) = (P(x), W(x))$ is defined. In the objective space the dominance relation is defined as follows:

Definition 1. A solution $x \in X$ is said to dominate a solution $x' \in X$ if $P(x) \geq P(x')$, $W(x) \leq W(x')$ and $f(x) \neq f(x')$. If x dominates x' then $f(x) > f(x')$.

Using the dominance relation previously defined, efficient solutions and non-dominated points are defined.

Definition 2. A feasible solution $x \in X$ is called efficient if there is no other feasible solution $x' \in X$ such that $f(x') > f(x)$. As a consequence, if x is efficient, then $f(x)$ is a non-dominated point.

The set of all efficient solutions is called the *Pareto set*, and is denoted as X_{eff}. The set of all non-dominated points is called the *Pareto front* and denoted as Y.

Another important definition is *supported solutions*, which are those that belong to the convex hull of the Pareto front. Otherwise, those that do not belong to the convex hull are called *unsupported solutions*. The set of all supported solutions is denoted as X_s, and the set of points associated to those solutions is denoted as Y_s. Figure 2.1 shows a Pareto front where points in blue represent supported solutions and points in red unsupported solutions. Note that an optimal solution to a weighted sum formulation of the multi-objective problem is also a supported solution.

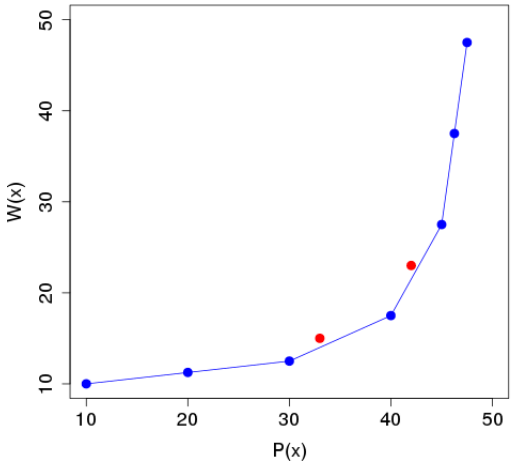

FIGURE 2.1: Illustration of supported (in blue) and unsupported (in red) solutions

2.3 Quality Measures for Pareto front Representations

One of the difficulties when dealing with multi-objective optimisation problems is the dimension of the Pareto front. Since having a large number of points can overwhelm a decision maker, it is important to be able to find good representations of the Pareto front according to some measures. In the following sub-sections three such measures are defined with respect to the Pareto front of an unconstrained bi-objective knapsack problem.

2.3.1 Uniformity

Uniformity was proposed by Sayın [14] and can be used to measure how far apart the elements of a given set $R \subseteq Y$ are from each other. It represents the minimum distance between any given distinct pair of points, and is formulated as

$$U(R) = \min_{\substack{u,v \in R \\ u \neq v}} \|u - v\| \tag{2.3}$$

where $\|\cdot\|$ is a p-norm with $1 \leq p \leq \infty$. The problem of finding a representation in terms of uniformity can be formalized as finding the subset $R^* \subseteq Y$, with cardinality k, that maximises $U(R)$, that is,

$$U(R^*) = \max_{\substack{R \subseteq Y \\ |R|=k}} U(R) \tag{2.4}$$

2.3.2 Coverage

Coverage was also proposed by Sayın [14] and it relates the distance of the unchosen elements to their closest chosen elements. Coverage can be defined in terms of a set R with respect to Y as

$$C(R, Y) = \max_{y \in Y} \min_{r \in R} \|r - y\| \tag{2.5}$$

The problem of finding a representation in terms of coverage can be formalized as finding the subset $R^* \subseteq Y$, with cardinality k, that minimises $C(R, Y)$, that is,

$$C(R^*, Y) = \min_{\substack{R \subseteq Y \\ |R|=k}} C(R, Y) \tag{2.6}$$

2.3.3 ϵ-indicator

The ϵ-indicator corresponds to the smallest factor that when multiplied to each element of a set $R \subseteq Y$, every point in the set Y becomes dominated. It can be defined as

$$E(R, Y) = \max_{y \in Y} \min_{r \in R} \epsilon(r, y) \tag{2.7}$$

where

$$\epsilon(r, y) = \max((r_p + \alpha)/(y_p + \alpha), (\widehat{W} - r_w + \alpha)/(\widehat{W} - y_w + \alpha)) \tag{2.8}$$

for a very small α (since $(0, 0)$ is always a non-dominated point), where $r = (r_p, r_w)$, $y = (y_p, y_w)$ and $\widehat{W} = \sum_{i=1}^{n} w_i$.

The problem of finding a representation in terms of the ϵ-indicator can be formalized as finding the subset $R^* \subseteq Y$, with cardinality k, that minimises $E(R, Y)$, that is,

$$E(R^*, Y) = \min_{\substack{R \subseteq Y \\ |R|=k}} E(R, Y) \tag{2.9}$$

Chapter 3

State-of-the-art

Most approaches to solve multi-objective optimisation problems are based on evolutionary algorithms, since the notion of population and set of solutions fits well together. However, these algorithms can in general give no guarantee on the quality of the solutions. There are, however, methods capable of returning the Pareto set of a multi-objective problem, but most, either due to the nature of the problem or the method itself, can not solve the problem in polynomial time in general. Moreover, the number of solutions in the Pareto set can be exponentially large which can overwhelm the decision maker.

In this chapter, the literature regarding the solution to the two problems referred above, the dimension of the Pareto set, and the intractability of the problem, will be reviewed. In Section 3.1 some methods used to find a representation of the Pareto set, when the set is known are analysed. These methods are useful when the problem is tractable or the size of the Pareto front is small. In Section 3.2 four methods, which are capable of finding a representation without knowledge of the Pareto set, are presented. These are useful when the problem is not tractable.

3.1 Representation with knowledge of the Pareto front

Multi-objective optimisation problems may generate a great number of efficient points, which can overwhelm the decision maker. One way to overcome this is to use subset selection methods. These are also relevant for the selection process of heuristic methods used to solve multi-objective optimisation problems.

Subset selection can be defined as finding the best subset with a given cardinality with respect to some quality measure. In the following, a summary of currently relevant

methods for solving the subset selection problem for various quality indicators is presented.

For the uniformity indicator [14] a dynamic programming algorithm was proposed by Wang and Kuo [18], and later improved by Vaz et al. [17]. At each step the best uniformity value for a given cardinality and containing certain elements, can be calculated by looking at the uniformity values from the previous step and distance calculation between the considered elements.

For the coverage indicator [14], a similar dynamic programming approach to that for the uniformity indicator, is proposed by Vaz et al. [17]. This approach follows the same principle for uniformity, but a correction is needed at the end of the procedure.

The ϵ-indicator [20] is also a common and important metric. A method was proposed by Ponte et al. [13] initially, and improved by Vaz et al. [17]. This method is similar to the one used for the coverage indicator. Koltun and Papadimitriou [10] also propose a greedy algorithm for a variant of the ϵ-indicator where the cardinality is not considered but rather the quality of the indicator. More recently, Bringmann et al. [3] present an algorithm which performs a binary search over a sequence of possible sets with different ϵ-indicator values.

One extra indicator which has been gaining some attention, due to its relevance in the performance assessment of heuristic methods, is the hypervolume indicator [19] which measures the volume of the dominated objective space for a set of solutions. Although the following approaches have been used in the context of subset selection of solutions, within multi-objective evolutionary algorithms, these approaches can be used to find a representative subset of efficient solutions. Auger et al. [1] propose a dynamic programming method based on the idea that a point only needs its two neighbours to calculate its contribution to the hypervolume. Similarly, Bader [2] proposes a dynamic programming algorithm, based on the idea that the left most point depends only on its immediate neighbour. Bringmann et al. [3] present a method for the problem where at each step i all maximum hypervolume values containing i elements and certain points are calculated. Kuhn et al. [11] show that the hypervolume subset selection problem can be modelled as a k-link shortest path problem in a directed graph. This particular shortest path problem with a cardinality constraint can then be solved with a dynamic programming approach. Lastly, it is also worth mentioning Guerreiro et al. [7] who propose a greedy algorithm that returns an approximation of the optimal subset. At each iteration the algorithm adds the point which contributes the most hypervolume to the set of already selected points. The results are especially interesting for three dimensions where optimal algorithms are very computationally expensive.

3.2 Representation without knowledge of the Pareto front

Finding the Pareto set can be a computationally expensive task. For those cases, finding a representation of it may be preferable. Not many approaches exist, which are capable of doing this with quality guarantees. In the following, approaches capable of computing a representation with quality guarantees and containing only elements of the Pareto set are reviewed.

Sayin [15] proposes a method to find the representation for multi-objective optimisation problems with coverage guarantees or a target cardinality. Up to our knowledge, it is the only method capable of returning a representation with a given cardinality. The method consists of an iterative process that at every step finds the point which results in the largest coverage value. It stops when either a given cardinality or a given coverage value is reached.

Sayın and Kouvelis [16] present a method using a parametric search algorithm to calculate the set of efficient solutions, and propose an alteration to the method to be able to find a representation by stopping the parametric search pre-emptively based on a distance criterion.

Hamacher et al. [8] introduce a method to find a representative subset for the bi-objective optimisation problem, using box algorithms. The idea is to start with a box given by two points which contains all non-dominated points. Whenever a new non-dominated point is found the box is split into smaller boxes. The algorithm stops when a given criterion based on the area of the box is reached. In the method proposed, there is a guarantee on the maximum cardinality with respect to the criterion.

Eusébio et al. [6] propose a method to find a representation for the Pareto set on the bi-objective network flow problem. At each iteration a constraint problem is solved to find a new non-dominated point until the representation has the guaranteed quality.

Chapter 4

Representation Algorithms

In this chapter the main contributions of this thesis are introduced. First, a state-of-the-art dynamic programming algorithm, to solve the unconstrained bi-objective knapsack problem, proposed by Nemhauser and Ullmann [12] is presented in Section 4.1. Moreover, a dichotomic search algorithm to calculate the set of supported solutions is presented in Section 4.2. Then, in Section 4.3 several modifications to the Nemhauser-Ullman algorithm are presented to find a representation to each of the three measures.

4.1 Nemhauser-Ullman Algorithm

The Nemhauser-Ullman [12] algorithm is a dynamic programming approach that keeps, at each iteration, only the efficient solutions for each sub-problem. It was originally proposed to solve the knapsack problem but it can be easily modified to solve the unconstrained bi-objective variation.

The sequential process consists of n stages. At any stage i, the algorithm generates a set of states S_i that corresponds to a subset of the feasible solutions made up of items belonging exclusively to the i first items, $i = 1, \ldots, n$. A state $s = (s^1, s^2) \in S_i$ represents a solution of profit s^1 and weight s^2. At each stage $i = 1, \ldots, n$, the states are generated according to the following recursion:

$$S_i = \Delta(T_i = S_{i-1} \cup \{(s^1 + p_i, s^2 + w_i) : s \in S_{i-1}\}) \tag{4.1}$$

with the basis case $S_0 = \{(0, 0)\}$. Operator $\Delta(\cdot)$ corresponds to the use of the dominance relation to remove dominated states. Note that $S_n = Y$.

4.2 Dichotomic search

The dichotomic search [12] algorithm is a method to find the set of supported points. At each step, two previously calculated supported points, $a = (a_w, a_p)$ and $b = (b_w, b_p)$ are used to find a point v, where $v_w = |a_w - b_w|$, $v_p = |a_p - b_p|$. A new supported point $d = (d_w, d_p)$ can then be calculated as follows:

$$d_p = \sum_{i=1}^{n} \begin{cases} p_i & \text{if } v_w \cdot p_i - v_p \cdot w_i \geq 0 \\ 0 & \text{if } v_w \cdot p_i - v_p \cdot w_i < 0 \end{cases} \tag{4.2}$$

$$d_w = \sum_{i=1}^{n} \begin{cases} w_i & \text{if } v_w \cdot p_i - v_p \cdot w_i \geq 0 \\ 0 & \text{if } v_w \cdot p_i - v_p \cdot w_i < 0 \end{cases} \tag{4.3}$$

where n is the number of existing items, and w_i, p_i the weight and profit of each item. If point d matches either a or b, it means that no new point can be found between those points and there is no need to search further. Otherwise, point d is added to the set of supported solutions and two searches are performed between points a and d, and also b and d.

The basis case for the algorithm are the extreme points of the efficient set: $\{0, 0\}$ and $\{\sum_i^n p_i, \sum_i^n w_i\}$. A recursive approach is presented in Algorithm 1.

4.3 Nemhauser-Ullman modifications

In this subsection two modifications to the Nemhauser-Ullman are presented. First an algorithm which solves the subset selection problem at every stage of the Nemhauser-Ullman algorithm, is presented. Second, the problem of finding a representative set is redefined as a sequence of k knapsack problems with capacity constraints. A pruning technique based on the computation of lower and upper bounds for the Nemhauser-Ullman algorithm is then presented in order to solve the problem. Lastly, a modification to the previous approach based on the set of supported solutions is introduced.

4.3.1 Filter Algorithm

As mentioned in Section 4.1, the Nemhauser-Ullman algorithm keeps a set of non-dominated points S_i at each stage $i = 1, \ldots, n$. The idea behind this algorithm is to solve the representation problem at every i-th stage, obtaining a representative subset

Algorithm 1 Dichotomic search for a min-max bi-objective problem

input: Set of weights w and profits p. Number of items n
output: Set of supported points Y_s
$Y_s = \{\}$
$a_w, a_p = 0$
$b_w = \sum_i^n w_i$
$b_p = \sum_i^n p_i$
REC(a,b)
function REC(a,b)
    $v_w = |a_w - b_w|$
    $v_p = |a_p - b_p|$
    $d_w, d_p = 0$
    for $i = 1$ **to** n **do**
        if $v_w \cdot p_i - v_p \cdot w_i \geq 0$ **then**
            $d_w = d_w + w_i$
            $d_p = d_p + p_i$
    if $d = a$ **or** $d = b$ **then**
        return
    $Y_s = Y_s \cup \{d\}$
    REC(a,d)
    REC(b,d)

$S^*_{i-1} \subseteq S_{i-1}$ with $|S^*_{i-1}| = k$ to use in the recursion of the Nemhauser-Ullman algorithm when $|S_{i-1}| > k$. Therefore, the recurrence equations can be reformulated as

$$S_i = \begin{cases} \Delta(T_i = S_{i-1} \ \cup \{(s^1 + p_i, s^2 + w_i) : s \in S_{i-1}\}), & \text{if } |S_{i-1}| \leq k \\ \Delta(T_i = S^*_{i-1} \ \cup \{(s^1 + p_i, s^2 + w_i) : s \in S_{i-1}\}), & \text{if } |S_{i-1}| > k \end{cases} \tag{4.4}$$

Note that, this algorithm does not guarantee that the elements of set S_n belong to the Pareto front.

4.3.2 Capacity Constraint Pruning Algorithm

For the second algorithm, the problem of finding a representative set is redefined as finding the set of k solutions with the lexicographically largest profit and smallest weight, each satisfying a given capacity constraint value $W_j, j = 1, \ldots, k$. W_j are assumed to be equally spaced in the range $(0, \sum_{i=1}^n w_i)$, that is:

$$W_j = \frac{j-1}{k-1} \sum_{i=1}^{n} w_i \tag{4.5}$$

Figure 4.1 illustrates the capacity constraints for a given cardinality of $k = 4$.

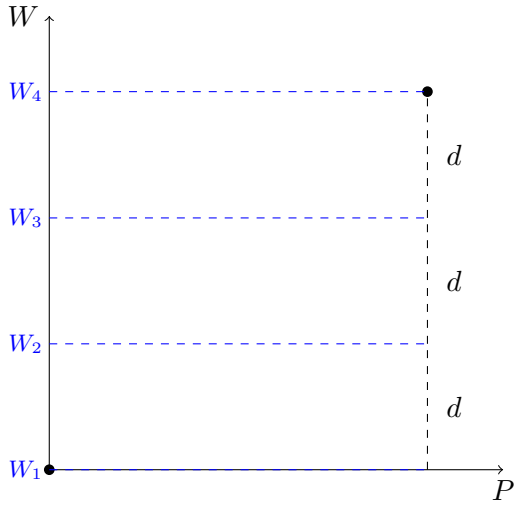

FIGURE 4.1: Illustration of capacity constraints for $k = 4$

More formally, the goal is to find set R_{lex}

$$R_{\text{lex}} = \{ \underset{x \in X}{\text{lex}} (P(x), W(x)) \mid W(x) \leq W_j, j = 1, \ldots, k \} \tag{4.6}$$

where lex denotes the lexicographic operator.

Definition 3. For $x \in X$, we define that x is lexicographically optimal if there is no $x' \in X$ such that $P(x') > P(x)$ or $(P(x') = P(x)$ and $(W(x') < W(x))$.

Note that due to the lexicographic operator, set R_{lex} only contains non-dominated solutions. However, it may only provide an approximation to the optimal representation value. Furthermore, Proposition 1 shows that finding a representation for a given k may not be possible.

Proposition 1. There exists an instance for which $|R_{\text{lex}}| < k \leq |Y|$ holds.

Proof. Consider an instance with two variables and the following profits and weights: $p_1 = w_1 = \theta$ and $p_2 = w_2 = 3 - \epsilon$, with $0 < \theta < 1$. Then, set $Y = \{(0,0), (\theta, \theta), (3 - \theta, 3 - \theta), (3,3)\}$. For $k = 4$, we define $W_2 = 1$ and $W_3 = 2$. Then, there exists no efficient solution, with total weight in the range $(1,2)$, therefore $|R_{\text{lex}}| < k \leq |Y|$. □

Figure 4.2 illustrates the proof of Proposition 1.

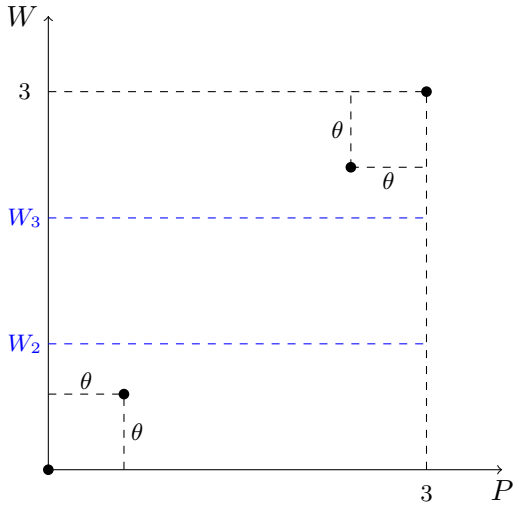

FIGURE 4.2: Illustration of the proof for Proposition 1

Note that, an optimal solution for the related knapsack problem for each constraint value W_j, may not be an efficient solution for the unconstrained bi-objective knapsack problem [9]. A suitable approach to find set R_{lex} is to use the Nemhauser-Ullman algorithm since it keeps a set of non-dominated states at each iteration. A technique for the Nemhauser-Ullman algorithm that allows to prune additional states that will not contribute to set R_{lex} is presented in the following.

4.3.2.1 Pruning technique

The pruning technique is based on the computation of lower and upper bounds for a sequence of related k knapsack problems with capacity constraint values $W_j, j = 1, \ldots, k$. For each state $s \in S_i, i = 1, \ldots, n$, a lower bound on the maximum profit value that can be achieved for that state for each constraint W_j is calculated; note that if this lower bound is sufficiently tight, then it may be a candidate element for the representation. In addition, the upper bound on the profit value of state s for each constraint W_j is also computed; if the upper bound is inferior or equal to the profit of the candidate element for the representation of every constraint W_j, then state s can be discarded.

Let W be the value of a capacity constraint. At stage $i = 1, \ldots, n$, let $e_W(s) = (e^1_W(s), e^2_W(s))$ denote the extension of a state $s \in S_i$ as follows

$$e_W(s) = \left(s^1 + \sum_{j \in J} p_j, s^2 + \sum_{j \in J} w_j \right) \tag{4.7}$$

where $J \subseteq \{i+1, \ldots, n\}$ such that $e^2_W(s) \leq W$. This extension can be obtained by Dantzig's greedy algorithm for the knapsack problem [4]. Without loss of generality, assume that $i < j$ implies that $p_i/w_i > p_j/w_j$. Equation 4.7 can be reformulated as follows

$$e_W(s) = \left(s^1 + \sum_{j=i+1}^{c-1} p_j, s^2 + \sum_{j=i+1}^{c-1} w_j \right) \tag{4.8}$$

where $c \in \{i+1, \ldots, n\}$ denotes the index of the critical variable whose weight cannot be added to $e^2_W(s)$ without breaking the capacity constraint W.

An upper bound $u_W(s)$ on the profit value of a given state $s \in S_i, i = i, \ldots, n$ is also given by Dantzig [4]. Let $\overline{W}(s) = W - s^2$ be the residual capacity associated to state s. Let

$$\overline{c} = \overline{W}(s) - \sum_{j=i+1}^{c-1} w_j \tag{4.9}$$

Thus given the same ordering of the items, the upper bound is computed as

$$u_W(s) = s^1 + \sum_{j=i+1}^{c-1} p_j + \left\lceil \overline{c} \frac{p_c}{w_c} \right\rceil \tag{4.10}$$

Let ℓ_{W_j} denote the profit of an extension or state that lexicographically maximises the profit and minimises the weight for each constraint $W_j, j = 1, \ldots, n$; then set $\{\ell_{W_1}, \ldots, \ell_{W_n}\}$ is a potential approximation to the representation. A state s can be pruned from $S_i, i = 1, \ldots, n$ if and only if $u_{W_j} \leq \ell_{W_j}$ for every $j = 1, \ldots, k$.

The calculation of the extension and upper bounds for a given state, take $O(n)$-time for all capacity constraints $W_j, j = 1, \ldots, k$.

4.3.2.2 Using the set of supported solutions

In the approach described in the previous section, the distance between each constraint takes only into account the weight. While this is an easy way to formulate the problem it might not be ideal for most problems. Figure 4.3 shows a Pareto front, and four capacity constraints. The circled points represent the points that would be chosen by solving the problem considering these capacity constraints. While the chosen solutions

are somehow separated they are not optimal for any measure. Figure 4.4 shows the optimal solution for the uniformity indicator.

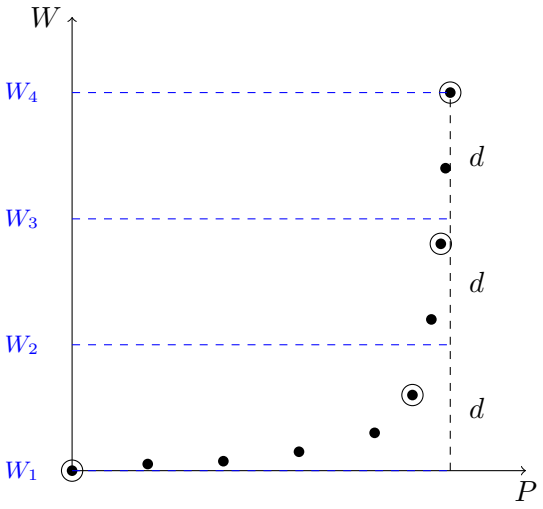

FIGURE 4.3: Illustration of solution for a problem given four capacity constraints

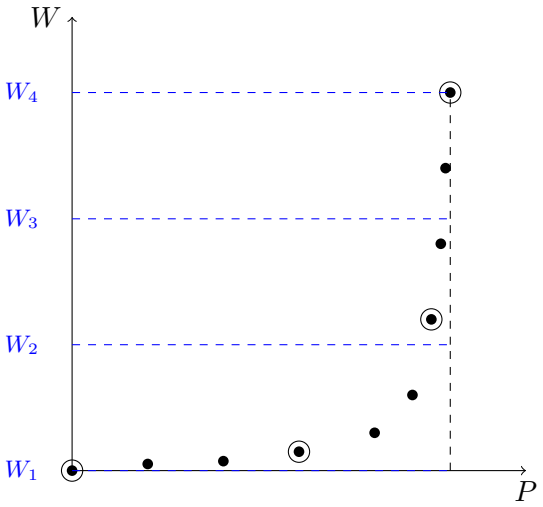

FIGURE 4.4: Illustration of optimal solution for the uniformity representation problem

Instead of using only the weight, the idea of this modification is to take into account some knowledge of the Pareto front, specifically the set of supported solutions. This set can be calculated with the dichotomic search algorithm described in Section 4.2. To find the capacity constraints, first the total distance between the points must be calculated as follows:

$$D = \sum_{i=1}^{|Y_s|} \|Y_s^i - Y_s^{i+1}\| \tag{4.11}$$

where Y_s is the set of supported points and Y_s^i represents the i-th in this set. It is assumed that Y_s is ordered according to one of the coordinates. The distance between each constraint is then calculated by:

$$d = \frac{D}{k-1} \tag{4.12}$$

Afterwards, the constraints are calculated through a series of steps:

1. Set $j = 1$, $W_j = 0$, $i = 1$ and $z = 0$.
2. Set $a = Y_s^i$ and $b = Y_s^{i+1}$.
3. Set $t = |a - b|$.
4. If $z + t < d$, then set $z = z + t$, increment i by one and go back to Step 2. Else if $z + t = d$, then set $z = z + t$, and go to Step 5. Else if $z + t > d$ go to step 6.
5. Set $W_{j+1} = b_w$, $z = 0$ and increment j by one. If $j = k$ the algorithm terminates. Else increment i by one and go back to Step 2.
6. Set $a_w = a_w + ((d - z)/t) \cdot a_w$, $a_p = a_p + ((d - z)/t) \cdot a_p$ and $W_{j+1} = a_w$. Increment j by one, if $j = k$ the algorithm terminates, else go to Step 3.

Chapter 5

Methodology

In the following chapters, several algorithms are presented to solve the representation problems presented above. Two measurements are used to compare the algorithms, the CPU-time, and the ratio between the representation value obtained by the algorithms above and the optimal according to each representation measure. In Chapter 6, an analysis is made on the uniformity indicator and because the representation problem is a maximising one, the ratio is given by the fraction of the uniformity value of the representation divided by the optimal uniformity. In Chapters 7 and 8, an analysis is made on the coverage and ϵ-indicator respectively. Since the representation problems for these measures are minimising problems, the ratio is given by the fraction of the optimal value divided by the indicator value of the representation obtained. These ratios were chosen so that the values are always between 0 and 1, where 1 represents a representation equal to the optimal.

The implementations were written in C and compiled with GCC 5.2.0 using the optimization flag *-O2*. The tests were run in a computer with the following specifications:

- *OS*: ArchLinux 4.1.6-1
- *CPU*: Intel i5 750 @ 2.67 GHz
- *RAM*: 4 Gb (2x2Gb) 1600MHz

The instances were generated for three different parameters: $n = \{20, 40, \ldots, 240\}$, $k = \{0.1n, 0.2n, \ldots, 1n\}$, and correlation $\rho = \{-0.8, -0.4, 0.0, 0.4, 0.8\}$. For each parameter combination, 10 instances were generated. Therefore, in total there are $n \cdot k \cdot \rho \cdot 10 = 6000$ instances. Each algorithms was run once for each instance. For each run, we recorded

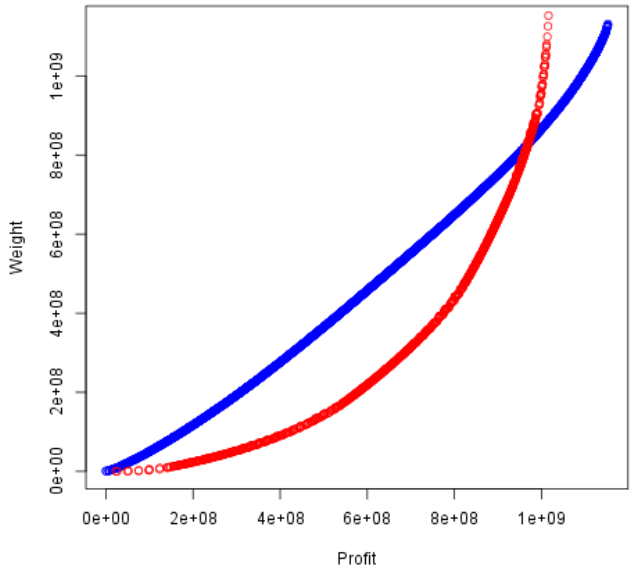

FIGURE 5.1: Pareto front for $\rho = -0.8$ (in red) and $\rho = 0.8$ (in blue) for an instance of size $n = 80$

the CPU-time and the value for each of the three measures (uniformity ,coverage , ϵ-indicator). The experimental analysis is performed in terms of the average CPU-time and the (harmonic) average error ratio.

While the importance of values n, and k for the problem is easy to understand, the importance for the correlation might not be. The correlation parameter is important because it allows the study of instances of the same size which result in Pareto fronts of different size and shape. In general a smaller negative correlation will result in a wider curve and a smaller number of points in the Pareto front, and a higher positive correlation in the opposite. Figure 5.1 illustrates this.

Chapter 6

Uniformity

In this chapter, an analysis on the algorithms previously described is performed. In Section 6.1 the filter algorithm is analysed, which calculates the optimal subset of size k at each stage considering the uniformity indicator. In order to retrieve the optimal subset, two algorithms are presented which take into account the optimal uniformity value calculated by the dynamic programming algorithm of Vaz et al. [17]. In Section 6.2 the capacity constraint pruning algorithm is analysed.

6.1 Filter Algorithm for Uniformity

The filter algorithm described in Section 4.3.1 works by solving the representation problem at each stage S_i of the Nemhauser-Ullman algorithm. In the following analysis, the dynamic programming algorithm to solve the uniformity representation problem by Vaz et al. [17] is used. This algorithm calculates the optimal uniformity value for a given set of non-dominated points and cardinality k. However, there might be more than one optimal subset S_i^* for a given stage S_i as shown through Proposition 2.

Proposition 2. There exists an instance for which there are $\binom{|S_i|}{k}$ combinations for possible optimal subsets with uniformity value $\sqrt{\epsilon}$ for the euclidean distance.

Proof. Consider an instance with three variables and the following profits and weights: $p_1 = w_1 = \epsilon$ and $p_2 = w_2 = p_3 = w_3 = 2\epsilon$ with $\epsilon > 0$. Then, at stage $S_3 = \{(0,0), (\epsilon,\epsilon), (2\epsilon,2\epsilon), (3\epsilon,3\epsilon), (4\epsilon,4\epsilon), (5\epsilon,5\epsilon), \}$. For $k = 4$, we have $15 = \binom{6}{4} = \binom{|S_3|}{k}$ possible optimal combinations. □

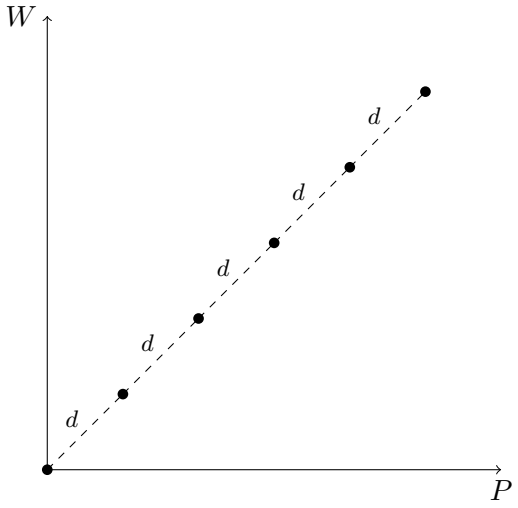

FIGURE 6.1: Illustration of the proof for Proposition 2

Since testing all possible combinations would be infeasible, two methods to generate an optimal subset are presented below. For the following methods, consider that S_i is ordered according to one of the objective functions.

Subset Generation Method 1 Given the optimal uniformity value $V = U(S_i)$, and the points $s_j \in S_i$ with $j = \{1, \ldots, m\}$ where $m = |S_i|$, the first point s_1 is added to the subset S_i^*, and the next point s_2 is selected. Afterwards, the distance between the last chosen point and the currently selected point is calculated. If the distance is equal or greater than V the currently selected point is added to S_i^*. Otherwise, the next point is selected. The procedure is repeated until k points have been added to the set. Algorithm 2 depicts the procedure.

Algorithm 2 Subset Generation Method 1

input: A set of solutions $S_i = \{s_1, \ldots, s_m\}$, uniformity value V, and cardinality value k.
output: The subset S_i^*.
$S_i^* = \{s_1\}$
$j = 1$
for $\ell = 2$ **to** m **do**
  if $\|s_j - s_\ell\| \geq V$ **then**
    $S_i^* = S_i^* \cup \{s_\ell\}$
    $j = j + 1$
    if $j = k$ **then**
      break

The choices made by this method are biased towards the first elements of S_i. This can be seen in Figure 6.2 where the circled dots are the chosen points for the subset for $V = d$

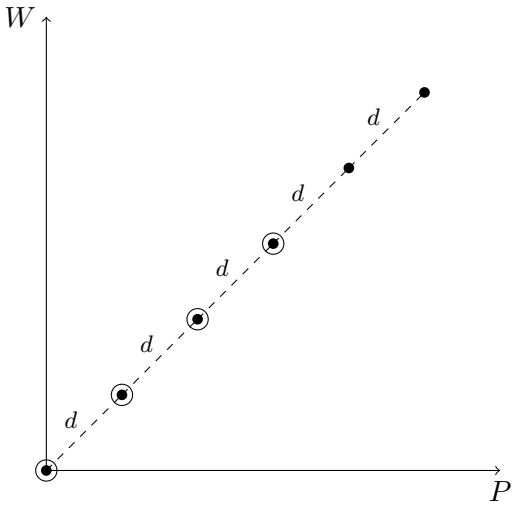

FIGURE 6.2: Example of subset selection method #1 for $k = 4$

Subset Generation Method 2 One way to overcome the problem mentioned above, is to consider $k/2$ points starting from the first point s_1 and checking the following points, $s_2, s_3, \ldots, s_m$, and the remaining $k/2 + (k \mod 2)$ points starting from the last point s_m, and checking the points in the opposite direction, that is, $s_{m-1}, s_{m-2}, \ldots, s_1$. The pseudo-code for this method is described in Algorithm 3. Figure 6.3 shows the points selected by the second method.

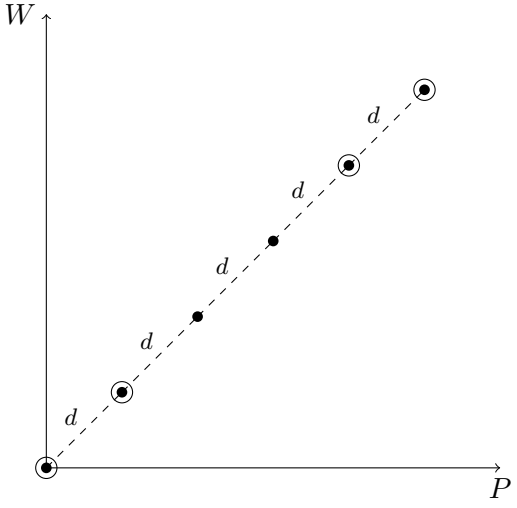

FIGURE 6.3: Example of subset selection method #2 for $k = 4$

Algorithm 3 Subset Generation Method 2

input: A set of solutions $S_i = \{s_1, \ldots, s_m\}$, uniformity value V, and cardinality value k.
output: The subset S_i^*.
$S_i^* = \{s_1\}$
$j = 1$
for $\ell = 2$ **to** m **do**
  if $\|s_j - s_\ell\| \geq V$ **then**
    $S_i^* = S_i^* \cup \{s_\ell\}$
    $j = j + 1$
    if $j = k/2$ **then**
      break
$S_i^* = S_i^* \cup s_m$
$j = j + 1$
for $\ell = m - 1$ **to** 1 **do**
  if $\|s_j - s_\ell\| \geq V$ **then**
    $S_i^* = S_i^* \cup \{s_\ell\}$
    $j = j + 1$
    if $j = k$ **then**
      break

6.1.1 Results

In this subsection the results for the filter algorithm with the the subset selections presented above are presented. The results are compared with the exact algorithm which consists of solving the unconstrained bi-objective knapsack problem, and running the dynamic programming algorithm by [17] to calculate the optimal uniformity.

Figures 6.4, 6.5, 6.6 show the comparison between the CPU-time. The algorithms are quite fast, rarely taking more than 0.1 seconds to complete. They are especially interesting for lower values of k and increasing values of n and correlation. Figures 6.7, 6.8 and 6.9 show the ratio between the uniformity value of the filter algorithm and the optimal uniformity value. As predicted in the algorithm analysis the second algorithm for subset selection performs slightly better. In conclusion both approaches are quite fast, although the quality of the representation is not very good.

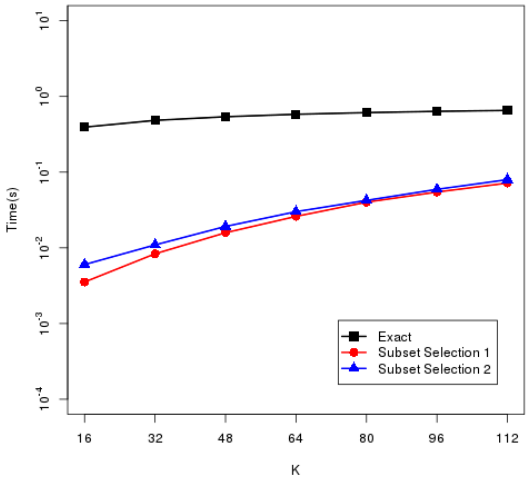

FIGURE 6.4: Time comparison for a varying k, $n = 160$, correlation $= 0$

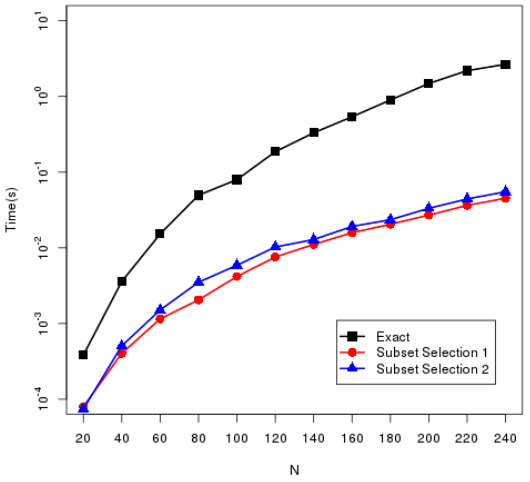

FIGURE 6.5: Time comparison for a varying n, $k = 0.3n$, correlation $= 0$

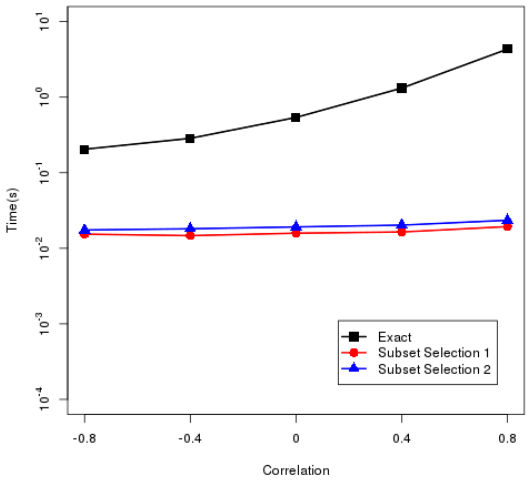

FIGURE 6.6: Time comparison for a varying correlation, $n = 160$, $k = 0.3n$

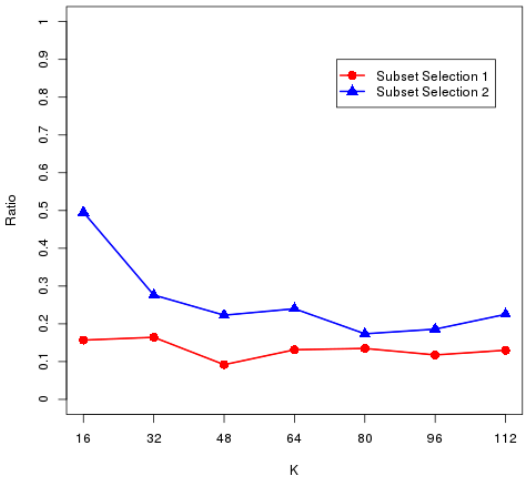

FIGURE 6.7: Error comparison for a varying k, $n = 160$, correlation $= 0$

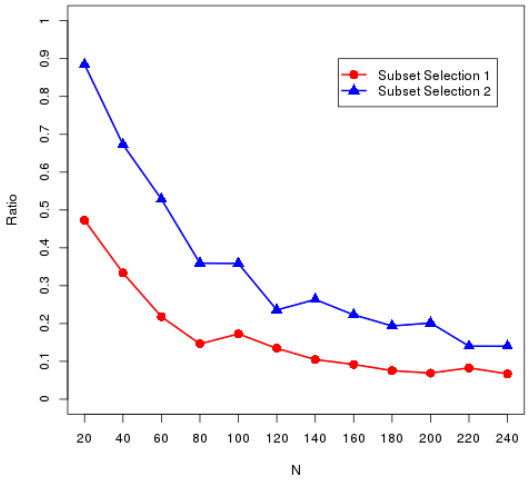

FIGURE 6.8: Error comparison for a varying n, $k = 0.3n$, correlation $= 0$

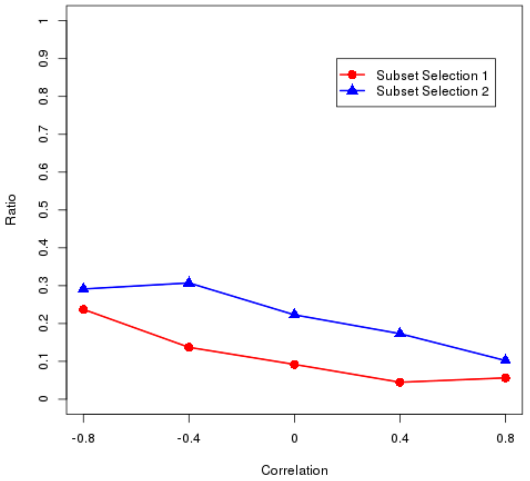

FIGURE 6.9: Error comparison for a varying correlation, $n = 160$, $k = 0.3n$

6.2 Capacity Constraint Pruning Algorithm Analysis

In this section, the uniformity guarantee for the capacity constraint pruning algorithm described in Chapter 4 is analysed with respect to the uniformity measure. Afterwards, the experimental results are presented.

6.2.1 Uniformity Guarantee

The following proposition shows that as the sum of weights and profits increases with a fixed cardinality, the relation between the optimal uniformity and the uniformity of the set R_{lex}, using the capacity constraints defined by the sum of the weights of the items, can be arbitrarily large.

Proposition 3. There exists an instance for an odd k, $\theta = \sum_{i=1}^{n} w_i = \sum_{i=1}^{n} p_i$ and $W_j = \frac{j-1}{k-1} \sum_{i=1}^{n} w_i$, for which, given the Euclidean norm, $U(R_{\text{lex}}) = \sqrt{2}$, and $U(Y^*) \approx \frac{2 \cdot \theta}{k-1}$ holds, where Y^* denotes an optimal solution for a representation with cardinality k.

Proof. Consider an instance with $\theta \geq 8$ and five variables where:

- $p_1 = \theta/2 - 3, w_1 = 1$
- $p_2 = \theta/2 - 2, w_2 = 1$
- $p_3 = 3, w_3 = \theta/2 - 2$
- $p_4 = 1, w_4 = \theta/2 - 1$
- $p_5 = 1, w_5 = 1$

Then, set $Y = \{(0,0), (\theta/2-2,1), (\theta-5,2), (\theta-4,3), (\theta-2,\theta/2), (\theta-1,\theta/2+1), (\theta,\theta)\}$. For $k = 5$ we define $W_2 = \theta/4, W_3 = \theta/2$ and $W_4 = 3\theta/4$, and obtain the sets

$$R_{\text{lex}} = \{(0,0), (\theta-4,3), (\theta-2,\theta/2), (\theta-1,\theta/2+1), (\theta,\theta)\} \tag{6.1}$$

$$Y^* = \{(0,0), (\theta/2-2,1), (\theta-5,2), (\theta-2,\theta/2), (\theta,\theta)\} \tag{6.2}$$

Then it holds that $U(Y^*) \approx 2\theta/4 = \theta/2$ and $U(R_{\text{lex}}) = \sqrt{2}$. □

Figures 6.10 and 6.11 illustrate the proof of Proposition 3, where it can be seen that there is a point on the bound $W3$ and another one above. These two points would

be chosen by the algorithm and the uniformity value would be quite small. A similar conclusion can be taken for an even k and for the capacity constraints defined with the set of supported solutions.

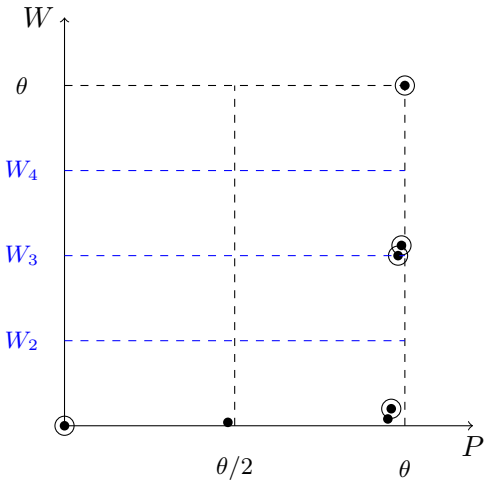

FIGURE 6.10: Illustration of the proof for Proposition 3 with set R_{lex} circled.

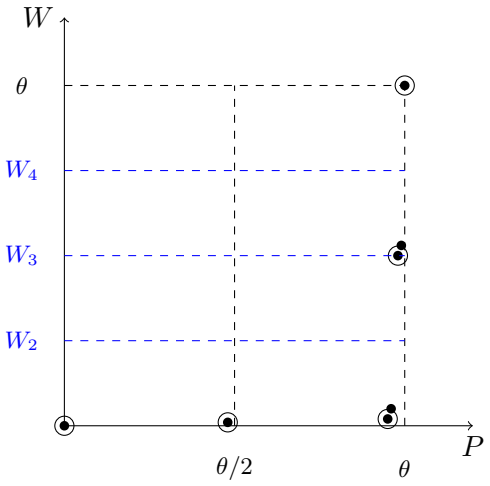

FIGURE 6.11: Illustration of the proof for Proposition 3 with set Y^* circled.

6.2.2 Results

In the following the capacity constraint method with weights, and the capacity constraint method with supported points, both presented in Chapter 4, are compared. Figures 6.12, 6.13 and 6.14 show that both capacity constraints methods take approximately the same time. It is also interesting to note that for higher values of n and correlation, and lower values of k, the algorithms are very fast obtaining values near the 0.01 second mark. For the ratio between the uniformity value obtained and the optimal uniformity value, where a higher value is better, there are some interesting results. Figure 6.15 shows that the algorithms perform worst for increasing values of k. However, it is interesting to note that the algorithm with capacity constraints defined by the set of supported solutions performs very well for low values of k obtaining uniformity values very near the optimal at $k = 16$. Figure 6.16 shows that the variation of n does not affect the results very much. Figure 6.17 shows that for large values of correlation, which imply more points in the Pareto front, the results are also very good. For correlation values of 0.8 the uniformity value obtained is very near the optimal for the algorithm with the set of supported solutions. In conclusion, the algorithms, especially the algorithm using the set of supported solutions perform very well for low values of k and high values of correlation, in terms of uniformity values, while providing a significant improvement in time.

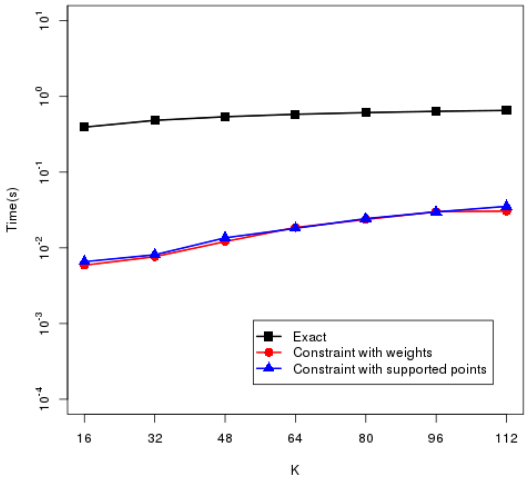

FIGURE 6.12: Time comparison for a varying k, $n = 160$, correlation $= 0$

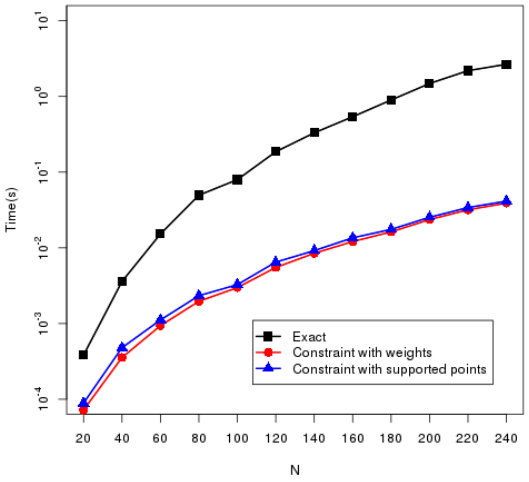

FIGURE 6.13: Time comparison for a varying n, $k = 0.3n$, correlation $= 0$

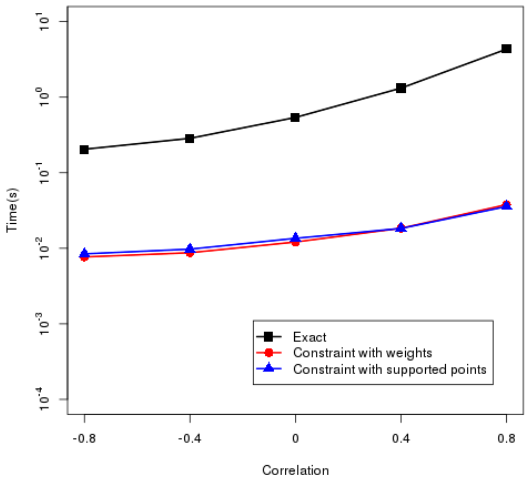

FIGURE 6.14: Time comparison for varying correlation, $n = 160$, $k = 0.3n$

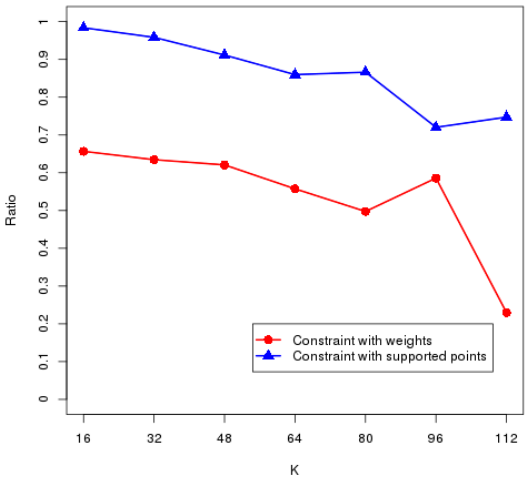

FIGURE 6.15: Error comparison for a varying k, $n = 160$, correlation $= 0$

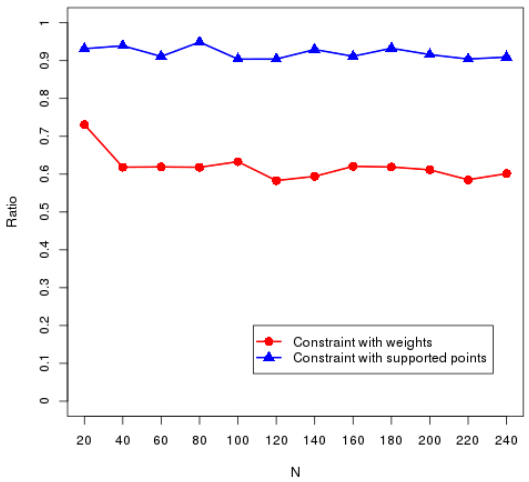

FIGURE 6.16: Error comparison for a varying n, $k = 0.3n$, correlation $= 0$

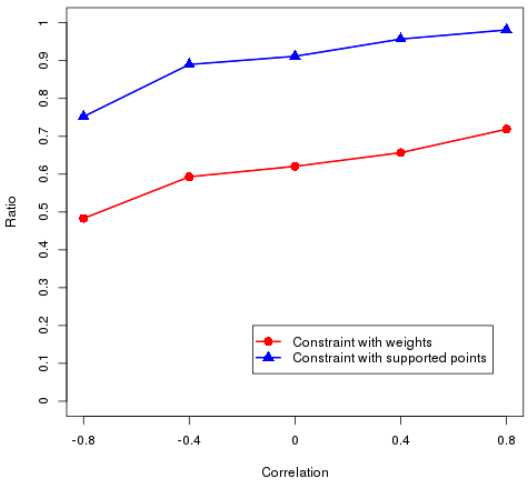

FIGURE 6.17: Error comparison for varying correlation, $n = 160$, $k = 0.3n$

Chapter 7

Coverage

In this chapter, an analysis on the algorithms previously described with respect to the coverage indicator is performed. In Section 7.1 the filter algorithm is analysed. In Section 7.2 the capacity constraint pruning algorithm is analysed and an improvement is suggested.

7.1 Filter Algorithm for Coverage

The filter algorithm described in Section 4.3.1 works by solving the representation problem at each step of the Nemhauser-Ullman algorithm. In the following analysis, the dynamic programming algorithm to solve the coverage representation problem by Vaz et al. [17] is used. This algorithm calculates the optimal coverage value for a given set of non-dominated points Y and a cardinality k. The subset of optimal points is retrieved from the matrix returned by the dynamic programming algorithm.

7.1.1 Results

Figures 7.1 tp 7.6 show the results for the filter algorithm regarding the coverage measure. Time-wise the results are similar to those of the uniformity, being between 10 to 100 times better than those for the exact algorithm. The ratio results, however, are not good, which is expected since the coverage indicator takes into the account the non-selected points of the Pareto front and the points returned by the filter algorithm are not necessarily part of the Pareto front and can be at a large distance. One positive change, however, is that the ratio for increasing values of k is also increasing because the algorithms filters less points at each stage.

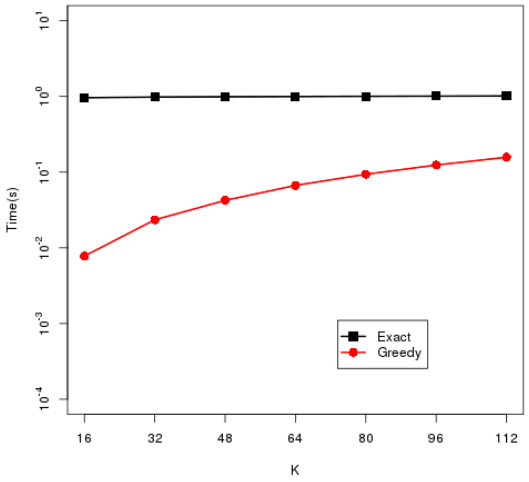

FIGURE 7.1: Time comparison for a varying k, $n = 160$, correlation $= 0$

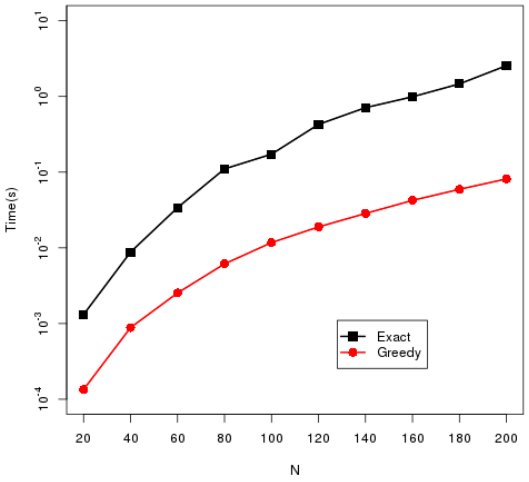

FIGURE 7.2: Time comparison for a varying n, $k = 0.3n$, correlation $= 0$

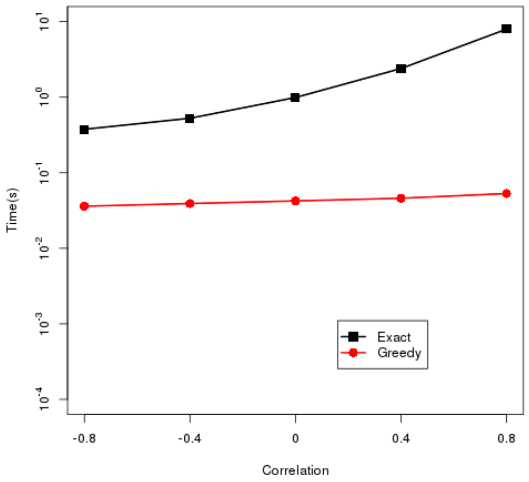

FIGURE 7.3: Time comparison for a varying correlation, $n = 160$, $k = 0.3n$

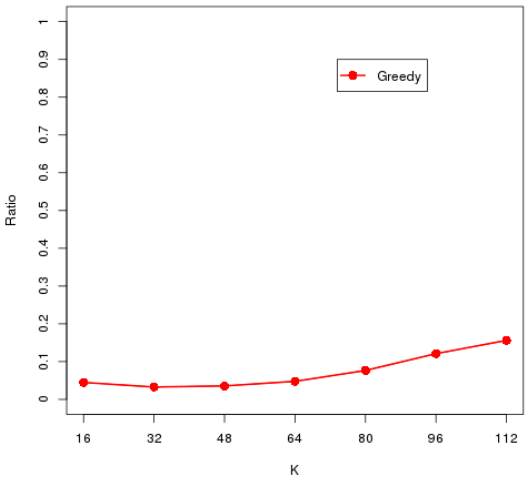

FIGURE 7.4: Error comparison for a varying k, $n = 160$, correlation $= 0$

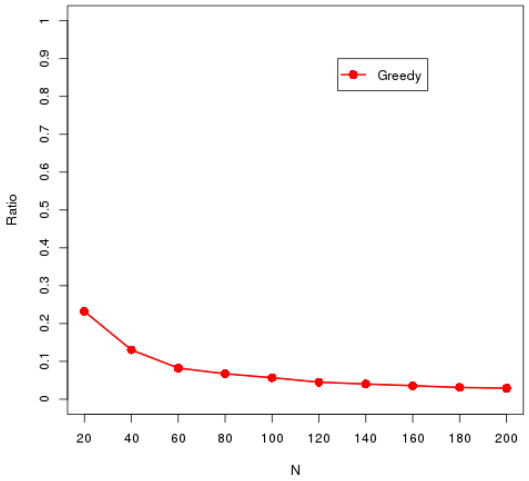

FIGURE 7.5: Error comparison for a varying n, $k = 0.3n$, correlation $= 0$

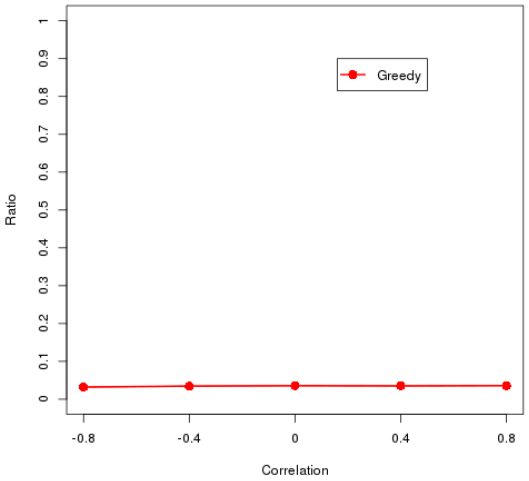

FIGURE 7.6: Error comparison for a varying correlation, $N = 160$, $k = 0.3n$

7.2 Capacity Constraint Pruning Algorithm Analysis

In this section, the coverage guarantee for the capacity constraint pruning algorithm described in Chapter 4 is analysed. Afterwards, an improvement is suggested and lastly the experimental results are presented.

7.2.1 Coverage Guarantee

The following Proposition 4 shows a similar result to that of Proposition 3.

Proposition 4. There exists an instance for an odd k, $\theta = \sum_{i=1}^{n} w_i = \sum_{i=1}^{n} p_i$ and $W_j = \frac{j-1}{k-1} \sum_{i=1}^{n} w_i$, for which, given the Euclidean norm, $C(R_{\text{lex}}) \approx \theta/2$, and $C(Y^*) = \sqrt{2}$ holds, where Y^* denotes an optimal solution for a representation with cardinality k.

Proof. Consider the same instance in the proof of Proposition 3. Then, it holds that $C(Y^*) = \sqrt{2}$ and $C(R_{\text{lex}}) \approx \theta/2$. □

A similar conclusion can be reached for an even k and using the capacity constraints based on the set supported solutions.

Figures 7.7 and 7.8 illustrate the proof of Proposition 4, the capacity constraint pruning algorithm would not choose point $(\theta/2 - 2, 1)$ which is part of the optimal solution, thus leading to a large coverage value.

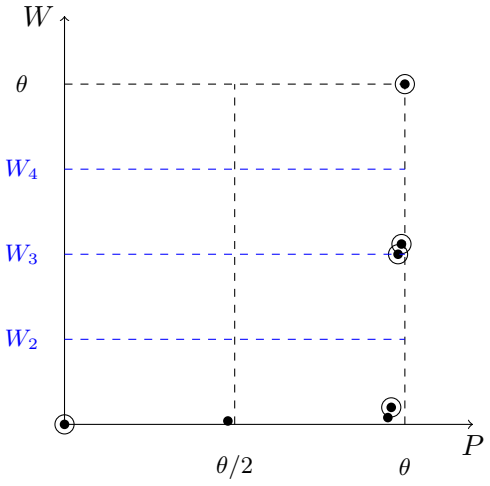

FIGURE 7.7: Illustration of the proof for Proposition 4 with set R_{lex} circled.

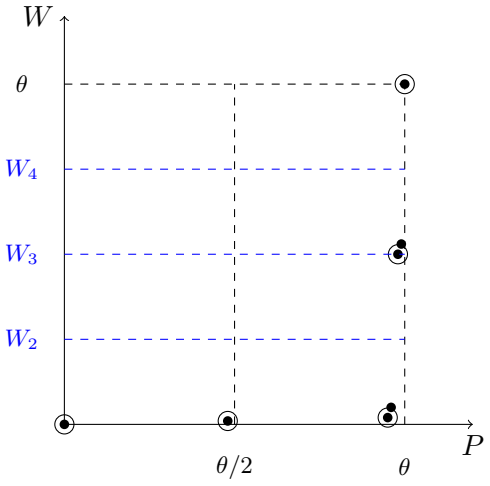

FIGURE 7.8: Illustration of the proof for Proposition 4 with set Y^* circled.

7.2.2 A Possible Improvement

Since the coverage representation searches for clusters of points a possible improvement might be to not consider the first and last constraints at the first and last points, as these will be at the edge of a cluster, but rather points a bit closer to the centre. Figure 7.9 illustrates this improvement, over the capacity constraints given by the weights.

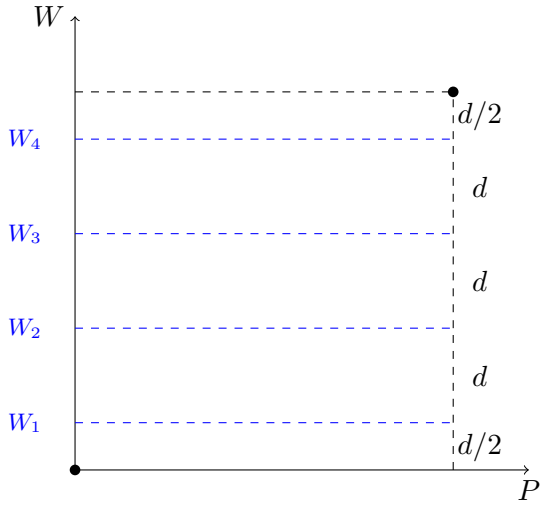

FIGURE 7.9: Illustration of possible improvement for constraints

7.2.3 Results

In the following, the capacity constraint method based on weights, and the capacity costraint method based on the set of supported points, both presented in Chapter 4, are compared. Also the improvement presented in the previous section is applied to the later algorithm.

Figures 7.10 to 7.12 show that the time improvements are similar to those of the uniformity case. However, the ratio results in Figures 7.13 to 7.15 show that the algorithm with the capacity constraint based on the set of supported solutions performs very well obtaining values near the optimal except for higher values of k. Furthermore, the improvement is not as good as expected, although, it provides slightly better results for smaller values of k and n and larger values of correlation.

In conclusion the experimental results for the coverage algorithms are quite good, and the capacity constraint pruning technique considering the set of supported solutions can be a good alternative to optimal methods.

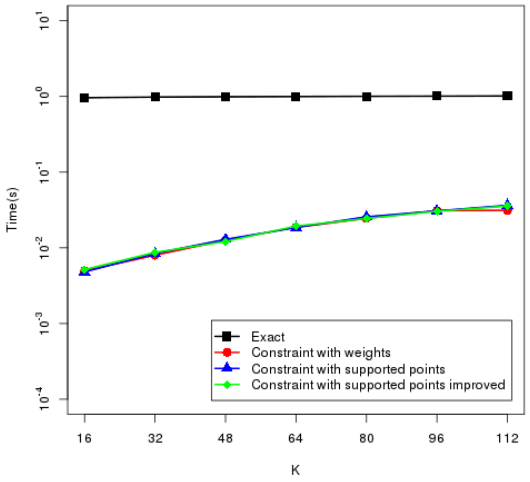

FIGURE 7.10: Time comparison for a varying k, $n = 160$, correlation $= 0$

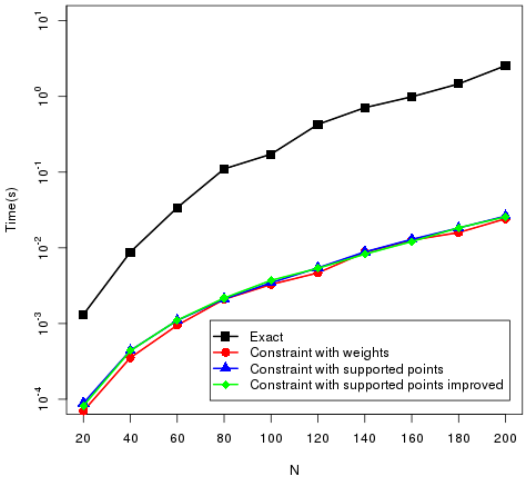

FIGURE 7.11: Time comparison for a varying n, $k = 0.3n$, correlation $= 0$

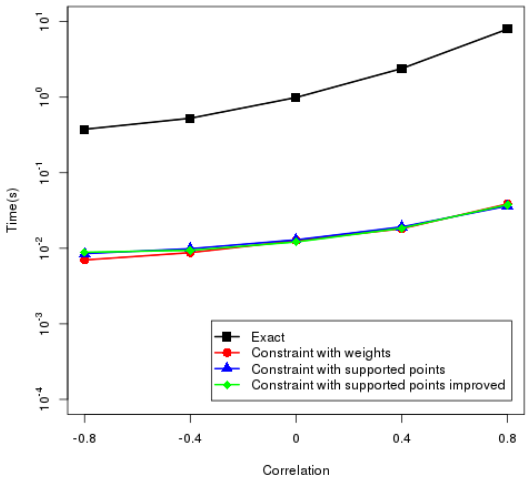

FIGURE 7.12: Time comparison for varying correlation, $n = 160$, $k = 0.3n$

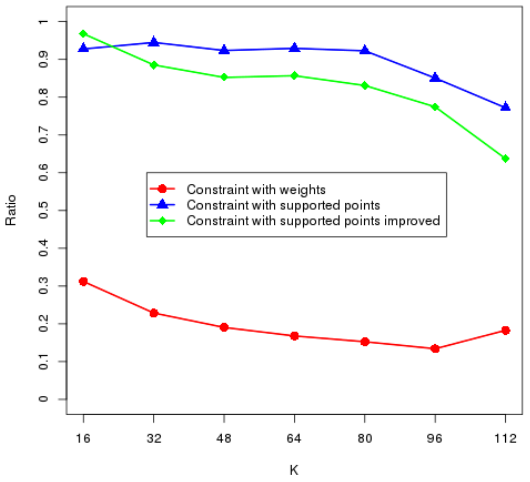

FIGURE 7.13: Error comparison for a varying k, $n = 160$, correlation $= 0$

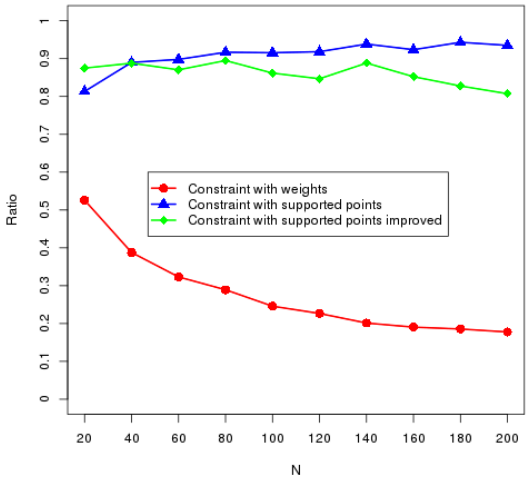

FIGURE 7.14: Error comparison for a varying n, $k = 0.3n$, correlation $= 0$

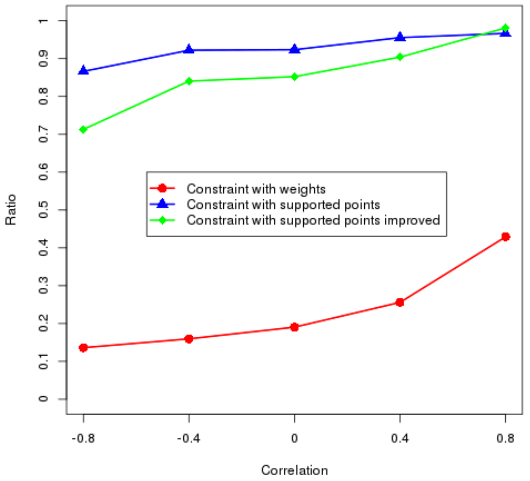

FIGURE 7.15: Error comparison for varying correlation, $n = 160$, $k = 0.3n$

Chapter 8

ϵ-indicator

In this chapter an analysis on the algorithms developed is presented with respect to the ϵ-indicator. In Section 8.1 the filter algorithm is analysed and the results are presented. In Section 8.2 the capacity constraint pruning technique is analysed and afterwards the experimental results are presented.

8.1 Filter Algorithm for ϵ-indicator

The filter algorithm described in Section 4.3.1 works by solving the representation problem at each step of the Nemhauser-Ullman algorithm. In the following analysis, the dynamic programming algorithm to solve the ϵ-indicator representation problem by Vaz et al. [17] is used. This algorithm calculates the optimal ϵ-indicator value for a given set of non-dominated points Y and a cardinality k. The subset of optimal points is retrieved from the matrix returned by the dynamic programming algorithm like in the coverage case.

8.1.1 Results

The results for the filter algorithm regarding the ϵ-indicator measure, as shown in Figures 8.1 to 8.6, are quite good. With respect to CPU-time the algorithm has similar performance to the coverage and uniformity cases, as expected. Furthermore, with respect to the ratio the results are very good, obtaining a ratio very close to 1 for larger values of k. This is expected because as k gets larger, each individual point of the representation will have to cover less points of the Pareto front thus lowering the ϵ needed. Similarly, the ratio decreases with a larger correlation, because the number of points in the Pareto front is increasing while maintaining a fixed k.

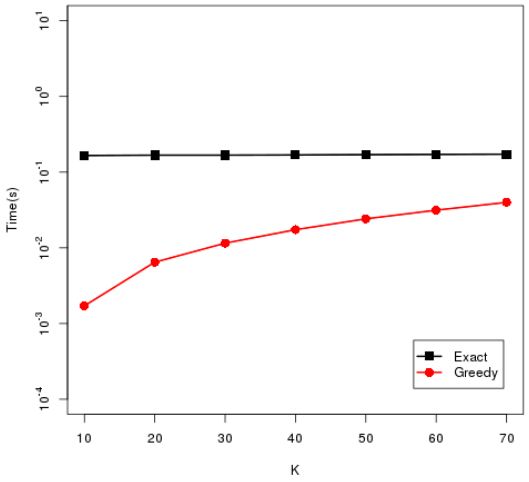

FIGURE 8.1: Time comparison for a varying k, $n = 100$, correlation $= 0$

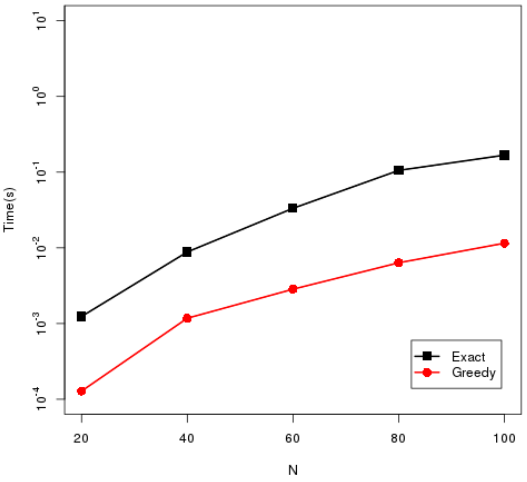

FIGURE 8.2: Time comparison for a varying n, $k = 0.3n$, correlation $= 0$

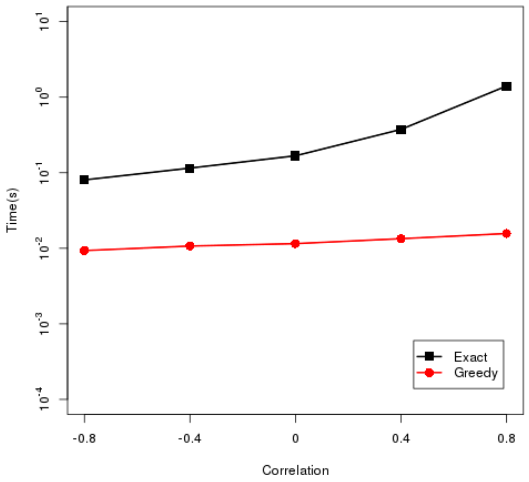

FIGURE 8.3: Time comparison for a varying correlation, $n = 100$, $k = 0.3n$

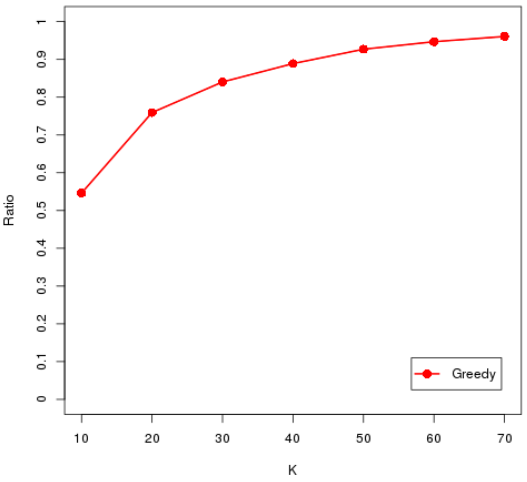

FIGURE 8.4: Error comparison for a varying k, $n = 100$, correlation $= 0$

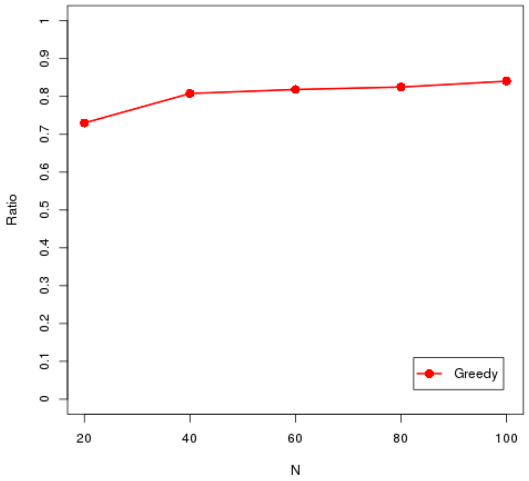

FIGURE 8.5: Error comparison for a varying n, $k = 0.3n$, correlation $= 0$

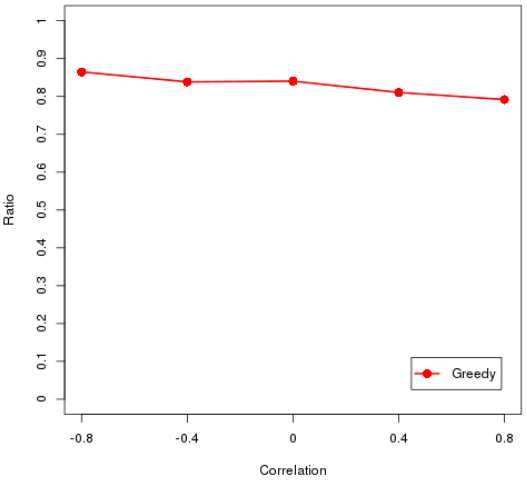

FIGURE 8.6: Error comparison for a varying correlation, $n = 100$, $k = 0.3n$

8.2 Capacity Constraint Pruning Algorithm Analysis

8.2.1 ϵ-indicator Guarantee

As in the previous indicators, the ϵ-indicator can take arbitrarily large values as shown in Proposition 5.

Proposition 5. There exists an instance for an even k, $\theta = \sum_{i=1}^{n} w_i = \sum_{i=1}^{n} p_i$ and $W_j = \frac{j-1}{k-1} \sum_{i=1}^{n} w_i$, for which, $E(R_{\text{lex}}) = \theta$, and $E(Y^*) = \theta/(\theta - 1) \approx 1$ holds, where Y^* denotes an optimal solution for a representation with cardinality k.

Proof. Consider the following items:

- $p_1 = \theta - 1, w_1 = 1$
- $p_2 = 1, w_2 = \theta - 1$

Then, set $Y = \{(0,0), (\theta-1,1), (\theta,\theta)\}$. For $k = 3$ define $W_1 = 0$ and $W_2 = \theta$ and obtain the sets

$$R_{\text{lex}} = \{(0,0), (\theta,\theta)\} \tag{8.1}$$

$$Y^* = \{(0,0), (\theta-1,1)\} \tag{8.2}$$

Then it holds that $E(Y^*) = \theta/(\theta-1) \approx 1$ and $E(R_{\text{lex}}) = \theta$. □

Figures 8.7 and 8.8 illustrate the proof of Proposition 5. A similar conclusion can be reached for an odd k and considering the set of supported solutions for the capacity constraints.

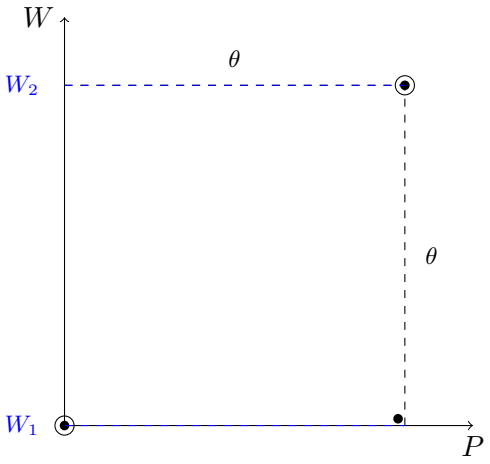

FIGURE 8.7: Illustration of the proof for Proposition 5 with set R_{lex} circled.

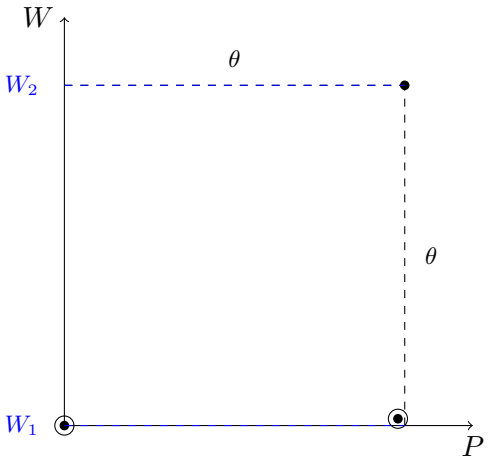

FIGURE 8.8: Illustration of the proof for Proposition 5 with set Y_* circled.

8.2.2 A Possible Improvement

The improvement is equal to that presented for the coverage case in Section 7.2.2.

8.2.3 Results

In the following, the capacity constraint pruning technique based on the weights for the constraints, and based on the set of supported points, are compared. Also the improvement presented in the previous section is applied to the capacity constraint with supported solutions.

Figures 8.9 to 8.11 show that the time improvements are similar to those of the uniformity and coverage case. However, the ratio results in Figures 8.12 to 8.14 show that all the algorithms perform very well obtaining values very close to the optimal. Similarly to the results in Section 8.1.1 the algorithms perform worse for smaller values of k and n.

In conclusion the experimental results for the algorithms in regard to the ϵ-indicator are very good, and make a very viable alternative.

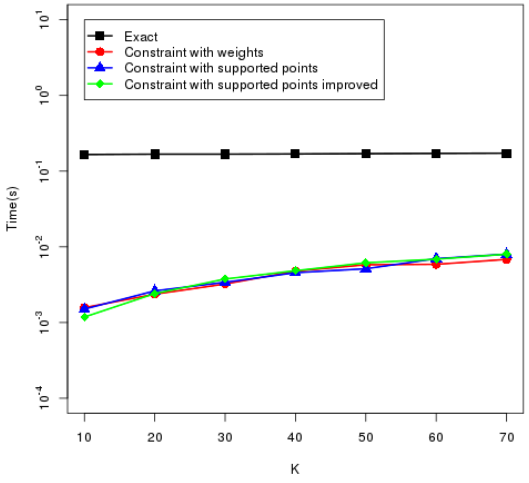

FIGURE 8.9: Time comparison for a varying k, $n = 100$, correlation $= 0$

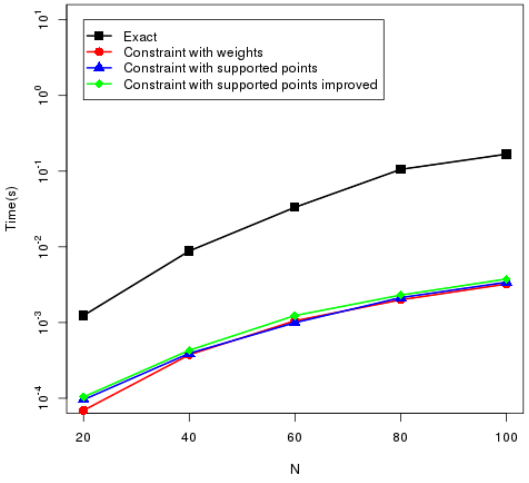

FIGURE 8.10: Time comparison for a varying n, $k = 0.3n$, correlation $= 0$

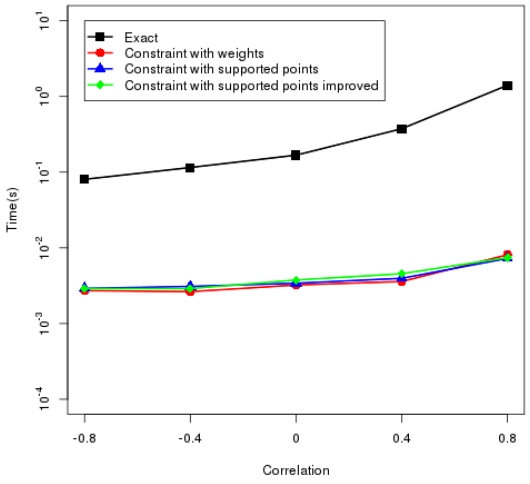

FIGURE 8.11: Time comparison for varying correlation, $n = 100$, $k = 0.3n$

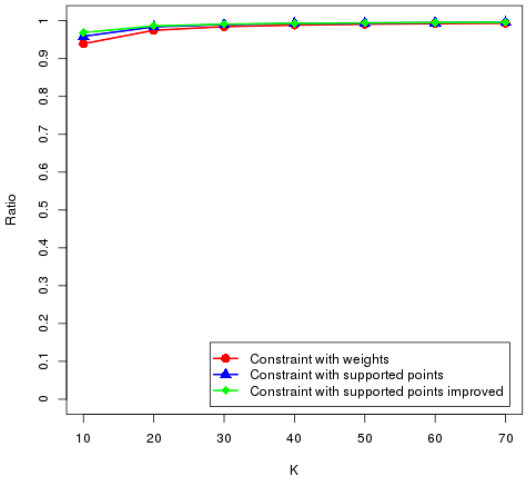

FIGURE 8.12: Error comparison for a varying k, $n = 100$, correlation $= 0$

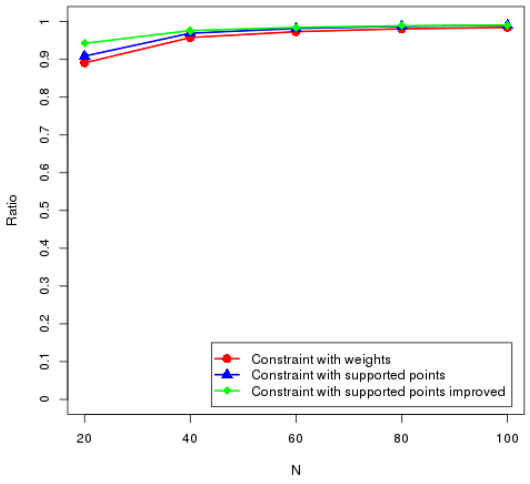

FIGURE 8.13: Error comparison for a varying n, $k = 0.3n$, correlation $= 0$

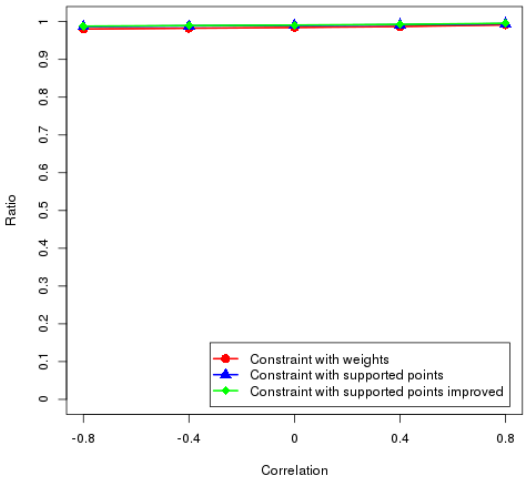

FIGURE 8.14: Error comparison for varying correlation, $n = 100$, $k = 0.3n$

Chapter 9

Conclusion

In this thesis, two modifications to the Nemhauser-Ullman algorithm were presented to find a representation of the Pareto front for the unconstrained bi-objective knapsack problem, considering three representation measures: uniformity, coverage, and the ϵ-indicator. These approaches take into account limited or even no knowledge of the Pareto front. The first approach, the filter algorithm, filters out unrepresentative points at each stage of the Nemhauser-Ullman algorithm, whereas the second, the capacity constraint pruning technique, uses information from upper and lower bounds to select potential representative points. Moreover, the second approach guarantees that the solutions found belong to the Pareto front.

An in-depth experimental analysis was carried out in order to investigate the performance of these approaches in several types of instances. The results for the filter algorithm were not very appealing in terms of quality, except for the ϵ-indicator. For the capacity constraint pruning technique, the results show that the error could be arbitrarily large for the three representation measures. The experimental results, however, showed that the algorithms can have a very good performance in terms of CPU-time and error.

9.1 Future Work

Even though the results were good, it may be possible to develop more efficient algorithms. The major bottleneck at the moment seems to be in the pruning technique, which has to calculate the bounds at each stage of the Nemhauser-Ullman algorithm. It could be interesting to investigate other notions of bounds that allow the reduction on the number of calculations needed. Another interesting approach could be to develop

methods based on a different algorithm paradigm or analyse this approach for more complex problems.

It would also be interesting to develop similar algorithms for the hypervolume-indicator, since it is gaining popularity due to its utility in the selection process for heuristic methods.

Lastly, the theoretical results presented in this thesis were negative in terms of representation quality, which is in contrast with the experimental results obtained. It would be more interesting to develop a better theoretical support that takes into account some property of the problems, e.g., assuming some distribution on profits and weights.

Bibliography

[1] A. Auger, J. Bader, D. Brockhoff, and E. Zitzler. Investigating and exploiting the bias of the weighted hypervolume to articulate user preferences. In *Proceedings of the 11th Annual conference on Genetic and Evolutionary Computation*, pages 563–570. ACM Press, 2009.

[2] J. M. Bader. *Hypervolume-based search for multiobjective optimization: theory and methods*. PhD thesis, Eidgenössische Technische Hochschule ETH Zürich, 2009.

[3] K. Bringmann, T. Friedrich, and P. Klitzke. Two-dimensional subset selection for hypervolume and epsilon-indicator. In *Proceedings of the 2014 conference on Genetic and evolutionary computation*, pages 589–596. ACM Press, 2014.

[4] G. B. Dantzig. Discrete-variable extremum problems. *Operations research*, 5(2): 266–288, 1957.

[5] M. Ehrgott and X. Gandibleux. A survey and annotated bibliography of multiobjective combinatorial optimization. *OR-Spektrum*, 22(4):425–460, 2000.

[6] A. Eusébio, J. R. Figueira, and M. Ehrgott. On finding representative non-dominated points for bi-objective integer network flow problems. *Computers & Operations Research*, 48:1–10, 2014.

[7] A. P. Guerreiro, C. M. Fonseca, and L. Paquete. Greedy hypervolume subset selection in the three-objective case. In *Proceedings of the 2015 on Genetic and Evolutionary Computation Conference*, pages 671–678. ACM Press, 2015.

[8] H. W. Hamacher, C. R. Pedersen, and S. Ruzika. Finding representative systems for discrete bicriterion optimization problems. *Operations Research Letters*, 35(3): 336–344, 2007.

[9] K. Klamroth and T. Jørgen. Constrained optimization using multiple objective programming. *Journal of Global Optimization*, 37(3):325–355, 2007.

[10] V. Koltun and C. H. Papadimitriou. Approximately dominating representatives. In *Proceedings of the 2005 International Conference on Database Theory*, pages 204–214. Springer, 2005.

[11] T. Kuhn, C. M. Fonseca, L. Paquete, S. Ruzika, M. M. Duarte, and J. R. Figueira. *Hypervolume subset selection in two dimensions: Formulations and algorithms.* 2015.

[12] G. L. Nemhauser and Z. Ullmann. Discrete dynamic programming and capital allocation. *Management Science*, 15(9):494–505, 1969.

[13] A. Ponte, L. Paquete, and J. R. Figueira. On beam search for multicriteria combinatorial optimization problems. In *Proceedings of the International Conference in Integration of AI and OR Techniques in Contraint Programming for Combinatorial Optimzation Problems*, pages 307–321. Springer, 2012.

[14] S. Sayın. Measuring the quality of discrete representations of efficient sets in multiple objective mathematical programming. *Mathematical Programming*, 87(3):543–560, 2000.

[15] S. Sayin. A procedure to find discrete representations of the efficient set with specified coverage errors. *Operations Research*, 51(3):427–436, 2003.

[16] S. Sayın and P. Kouvelis. The multiobjective discrete optimization problem: A weighted min-max two-stage optimization approach and a bicriteria algorithm. *Management Science*, 51(10):1572–1581, 2005.

[17] D. Vaz, L. Paquete, C. M. Fonseca, K. Klamroth, and M. Stiglmayr. Representation of the non-dominated set in biobjective discrete optimization. *Computers & Operations Research*, 63:172 – 186, 2015.

[18] D. Wang and Y.-S. Kuo. A study on two geometric location problems. *Information Processing Letters*, 28(6):281 – 286, 1988. ISSN 0020-0190.

[19] E. Zitzler and L. Thiele. Multiobjective optimization using evolutionary algorithms—a comparative case study. In *Proceedings of the International Conference on Parallel problem solving from nature—PPSN V*, pages 292–301. Springer, 1998.

[20] E. Zitzler, L. Thiele, M. Laumanns, C. Fonseca, and V. da Fonseca. Performance assessment of multiobjective optimizers: an analysis and review. *IEEE Transactions on Evolutionary Computation*, 7(2):117–132, April 2003.