1-1-2013

Exact And Representative Algorithms For Multi Objective Optimization

Ozgu Turgut
Wayne State University,

EXACT AND REPRESENTATIVE ALGORITHMS FOR MULTI-OBJECTIVE OPTIMIZATION

by

OZGU TURGUT

DISSERTATION

Submitted to the Graduate School

of Wayne State University,

Detroit, Michigan

in partial fulfillment of the requirements

for degree of

DOCTOR OF PHILOSOPHY

2014

MAJOR: INDUSTRIAL ENGINEERING

Approved by:

_____

Advisor Date

_____

_____

_____

_____

DEDICATION

to my Mom and Sisters...

ACKNOWLEDGEMENTS

Foremost, I would like to thank to my advisor, Dr. Alper E. Murat, for his support and motivation during the time I spent at WSU and for his contribution to my character as a professional through his hardworking, smart and persevering perspective. Next, I would like to thank all of my committee members, Dr. Kenneth Chelst, Dr. Ratna Babu Chinnam, Dr. Leslie Monplaisir and Dr. Boris S. Mordukhovich, each of whom were valuable researchers and professors with outstanding achievements. They have been perfect examples of faculty members that it will always be an honor for me to have been a graduate of their school. I want to express my gratitude to Dr. Ratna Babu Chinnam for introducing me to "multi-objective optimization" with Dr. Alper E. Murat and for creating this motivating and productive research environment in our department with his impressive vision and discipline.

Special thanks go to Dr. Monplaisir for his determined support of the research through sincere encouragement and immense knowledge, despite his heavy schedule as a department chair. He and Dr. Celestine C. Aguwa led me to work on interesting projects, each of which helped me improve in terms of creativity, team work and solution-based thinking. One last special thanks to Dr. Evrim Dalkiran for the contributions she made to my dissertation and life through her deep and inspiring knowledge, creativity, teaching ability and flawless character. Last but not least, important thanks go to Shanshan Qui for her sincere, supportive and joyful companionship since we met. I also want to thank to my company, LLamasoft, Inc. in the name of Dr. Nejat Karabakal, along with my advisor, Dr. Alper E. Murat, for maintaining this flexible research environment that allowed me to continue working on exciting OR projects at LLamasoft while finishing my dissertation.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I: INTRODUCTION

In real life most of problems have to be evaluated by considering more than one objective (criterion). Optimizing an objective has a long history. Multi-objective optimization (MOP) has been studied in the literature as a result of the emerging necessity to consider conflicting objectives created by complex systems. Since objectives are generally conflicting by nature, a single point that optimizes all of the objectives cannot be found in these cases. So the aim of the MOP problem is to reach a set of solutions that cannot be replaced by better points in the feasible space considering all of the objectives. This situation gave rise to the specific definition of optimality for MOP problems, called Pareto optimality. A problem that has different aspects to be considered can be handled by a utility function. However, the shape of this utility function may not always have a linear structure; besides, deriving a utility function is neither simple nor even possible in every case. So trying to reach to all or part of the Pareto optimal set is necessary in most cases.

A MOP problem can be formulated as follows:

$$\min (\max) \{ (z1(x), \dots , zp(x)) = Cx : x \in X \} \qquad (1)$$

where $p \geq 2$ and ($C \in R^{p \times n}$; X denotes the set of feasible set of solutions and is defined by

$$X = \{x \in Z^n : Ax \leq b, x \geq 0, x \in R\} \qquad (2)$$

where $A \in R^{m \times n}$ and $b \in R^m$. The corresponding objective space is defined by $Y := \{Cx : x \in X\}$. It is assumed that objectives are conflicting and cannot be optimized simultaneously within the feasible solution space. When $x \in R$ are all continuous, the valued problem is called a multi-objective linear programming (MOLP) problem; on the other hand, when $x \in R$ is replaced with $x \in Z$, it becomes the well-known multi-objective integer programming (MOIP) problem. Multi-

criteria optimization problems with integer decision variables differ from their continuous counterparts and thus require different solution approaches. In particular, problems with integer decision variables have non-convex and finite feasible space in comparison with continuous problems.

There are some basic methods that can be used for all types of MOPs in practice. The weighted sum method is the most popular Pareto generation method. The well-known epsilon-constraint method establishes a series of hyper cubes in the objective space by constraining all objectives except one. Non-dominated solutions are then obtained by solving a single-objective problem in each hypercube based on the excluded objective. Another methodology is called scalarization techniques, in which all objectives are combined into a single function. A discussion about scalarization functions can be found in Ehrgott ((2006)). Reference point-related approaches exist which can also be covered under scalarization functions. The distance to a reference point for all objectives is potentially minimized in this approach rather than distance to special points as defined in the MOP context (i.e., anchor or nadir points) (A.P. Wierzbicki, 1980). Several studies summarize the contributions and open problems in solving MOP problems (Chinchuluun & Pardalos, 2007); (Marler & Arora, 2004); (Klamroth & Tind, 2007).

MOP is essential in many complex systems and product design decisions. In practice, decision makers (DM) prefer to select from a diverse set of non-dominated solution alternatives before finalizing their decision. In most practical applications, the process of obtaining the full Pareto front is impossible with reasonable computational effort. Besides, as indicated by Karasakal and Koksalan (2009) and Steuer (1986), even if a DM can generate the whole Pareto surface, selecting the most preferred solution remains difficult and may cause information overload if the entire Pareto set is presented to the DM. Hence, there is a need for efficient

methods for solving multi-objective programs, which provide well dispersed non-dominated solution sets that are also representative of the Pareto front.

Approximate methods for MOLP aim to generate an approximation set for the whole Pareto front rather than for the exact Pareto front. The performance of approximate methods is measured by how representative the final solution set is (Hansen & Jaszkiewicz, 1998). Although there are several measures of representativeness, the three most common are coverage, uniformity and cardinality (Faulkenberg & Wiecek, 2010); (Sayin, 2000). Approximate methods can be classified based on how candidate Pareto points are generated. As one of the most popular tools used in many problems, metaheuristic-based methods produce approximation sets that may include dominated solutions (Hanne, 2000), even if solutions are filtered at the end of the algorithm. Gunawan et al. (2003) proposed a method based on a multi-objective genetic algorithm. The nonlinear multi-objective optimization algorithm of Fu and Diwekar (2004) is based on the principles of probabilistic uncertainty analysis and the traditional constraint method in an effort to generate a representation of the nondominated frontier. A good literature survey for metaheuristic-based approximation methods can be found in Ehrgott and Gandibleux (2008) and Konak et al. (2006).

Representativeness of a solution set is an important part of approximation algorithms as it ensures that certain regions of Pareto surface that contain interesting solutions to the DM are not omitted and that the solutions are evenly distributed on the criteria space. In this study, we only focus on methodologies that guarantee producing Pareto optimal points using exact algorithms. We refer the reader to the survey paper of Ruzika and Wiecek (2005), which reviews all of these exact approximation methods. Considering the exact nature of these algorithms, as

opposed to metaheuristics, these methods can be also named as “representative set” generation algorithms.

For representative set generation algorithms, one key concept is evaluating the quality of the solution set. Sayin (2000) defined the coverage error as the distance between the worst represented point on the nondominated frontier and the corresponding representative point. Any point in the nondominated set is considered represented by its closest representative point in the criterion space. The distance between the two points gives the error in representing the nondominated point. The coverage error is defined as the maximum of such errors over all nondominated points. Uniformity is defined as the minimum distance between representative points. Smaller coverage errors and larger uniformity levels are desirable for better representation. Cardinality refers to the cardinality of the solution set, and it represents the number of solutions that are apart from each other for a predetermined distance. Other approaches purportedly evaluate the quality of an approximation of the nondominated frontier, which does not require generation of an actual nondominated frontier (Zitzler, Thiele, Laumanns, Fonseca, & Fonseca, 2003); (Laumanns, Thiele, Deb, & Zitzler, 2002); (Wu & Azarm, 2001); (Fleischer, 2003).

The second contribution of this thesis is another algorithm that aims to generate the Pareto front of MOIP problems. MOIP problems are unique in the sense that the structure of their Pareto front is non-convex. The MOIP methodologies can be broadly categorized into two main groups: exact and approximate methods. Exact methods aim to generate the whole Pareto front and have been extensively studied over the past decade. Przybylski et al. (2010b) compare four different exact methods for solving MOIP problems that have more than two objectives. The authors conclude that their proposed “two phased method" outperforms the algorithms of Sylva

and Crema (2004), Tenfelde-Podehl (2003) and Laumann et al. (2005). Laumann et al. (2005) was used as benchmark in this study as it is an adaptive epsilon constraint-based method for problems with more than two objectives. Lemesre et al. (2007) then put forward the 2-Parallel Partioning Method (2-PPM) to solve biobjective problems. This method works by partitioning the objective space and finding one nondominated point and an associated solution in each part. The remaining solutions are then found by exploring the feasible solution set, reduced by the previously identified solutions. This method was later extended to any number of objectives by Dhaenens et al. (2010), and it was given the name K-PPM. In another recent study, Przybylski et al. (2010a) propose a recursive algorithm for finding all nondominated extreme points for MOIP problems based on weight space decomposition. However, the algorithm resented by Ozlen and Azizoglu (2009) is more efficient in terms of computational requirements. Lokman and Koksalan (2012) presented another exact algorithm for MOIP problems, which seems to outperform all previously mentioned algorithms from the aspect of computational time, which will be explained in detail in the third chapter of this dissertation. However, it should be noted that the application of exact methods in the most practical MOIP problem instances is not practical since the computational effort required to generate whole Pareto increases rapidly with the number of variables and objectives. Despite this, running exact methods for practical MOIP problems is still important since they can be used as benchmarks to evaluate approximate methods.

1.1) Motivation

An analyst can contribute to the decision process, if the preferences of the decision maker (DM) have an appropriate mathematical structure. This structure is either a relation (preference relation) or a function (value function) (Keeney & Raiffa, 1976). A value function, denoted by v,

is a real-valued function defined on the criterion space with the property that the DM prefers a feasible solution y to another one, z if and only if $v(y) > v(z)$. In general, the function v is not known to either the DM or the analyst; one can suppose, however, that the function v is non-decreasing in each of the criteria. In the absence of any other information about this value function, one can say that the DM wishes to optimize each of the criteria (Marcotte & Soland, 1986). However, not all solutions sets necessarily constitute a good representation of the efficient set, as it usually contains too many points or is not uniformly spread across the actual Pareto front. This has motivated the search for discrete representations that consist of efficient points that are different from the extreme points (Sayin, 2003). At this point, one can consider using metaheuristics or exact algorithms to get an approximation set of the actual Pareto front of MOP. As mentioned earlier, metaheuristics do not offer solutions that are guaranteed to be Pareto optimal. Hence, an exact algorithm that produces a solution set that is also representative of the Pareto set is a necessary tool for practical purposes. In this study, our first goal is to propose an algorithm that is exact and generates a solution set that is representative of the actual Pareto front, which can be used for real-life MOP problems.

MOIP problems are of special kind of problem among MOP problems in the sense that their Pareto front is also discrete and cannot be expressed with efficient faces. As it is explained later in the preliminaries section, the Pareto front's unique structure has led to MOIP-specific definitions, such as supported or nonsupported solutions. Again, due to their non-convex structure, weighted sum, for example, one of the more well-known methods, becomes obsolete, as it cannot generate all of the Pareto front but only extreme points of it. There are two phased methods in the literature that use weighted sum in order to generate the extreme points, that then resorts to other techniques to generate the rest of the Pareto front. Likewise, there are many exact

methodologies that generate the Pareto front by relying on the previous solutions and generating the whole Pareto starting from one of its corners. Latter methodologies are more efficient than two phased approaches in terms of running time. However, they cannot generate a representative set at an intermediate stage of the algorithm. Hence, one has to wait until the termination of the algorithm to have a complete understanding of the Pareto front. Besides, to the best of our knowledge, none of the existing exact algorithms is capable of being parallelized, which makes them computationally hard to tackle when faced with substantial MOIP problems. Hence, there is a need for an exact algorithm that can generate the whole Pareto set for the MOIP problem, and which can be used as an approximation method under time restrictions. Based on this, the second goal of this thesis can be expressed as proposing such an algorithm.

In summary, the research objectives of this thesis can be summed up as follows:

1. Design algorithms that
    a. can be used in order to generate representative solutions;
    b. are exact in nature; and
    c. can be used to generate the whole set of Pareto solutions in cases where the Pareto front is finite and cannot be expressed in closed form.
2. Incorporate the branch and bound (B&B) idea to the MOP area.

1.2) Dissertation Organization

Preliminary concepts are presented in the remainder of this chapter regarding MOP literature. The second chapter presents an exact representative set generation algorithm with all the benchmark studies. Chapter Two starts with related literature and continues with the details of the proposed algorithm. Before the final section of first chapter, the benchmark algorithms are

explained, and the chapter concludes with computational experiments and a summary of conclusions based on the tests and analysis. Chapter Three presents another new exact algorithm proposed for MOIP problems that can be used to generate the whole Pareto set or as an approximation algorithm. After a brief introduction, the benchmark algorithms are explained. Then, the details of the proposed approach are presented. This chapter concludes with computational analysis and results related to the algorithm. The final chapter contains a summary of the results based on the studies in the dissertation and recommendations for further research related to the proposed algorithms.

1.3) Preliminaries

A linear multi-objective optimization problem with continuous variables (MOLP) is defined as follows:

$$\min\{\left(z_1(x), \dots, z_p(x)\right) = Cx: \ \ x \in X\},$$

where $p \geq 2$ and $C \in R^{p \times n}$; X denotes the set of feasible set of solutions and is defined by

$$X = \{x \in Z^n : Ax \leq b, x \geq 0\}$$

where, $A \in R^{m \times n}$ and $b \in R^m$. The corresponding objective space is defined by $Y = \{Cx: \quad x \in X\}$. By the nature of the objectives, they cannot be optimized simultaneously within the feasible space. If this could be done, the Pareto set would consist of a single point.

A feasible solution $x^* \in X$ is *efficient* if there does not exist any other feasible solution $x \in X$, such that $z(x) \leq z(x^*)$. If x^* is efficient, $z(x^*)$ is *nondominated*. If $x, x' \in X$ are such that $z_i(x) \leq z_i(x')\ \forall i = \{1, \dots, p\}$ ($z_i(x) \geq z_i(x')$ (for maximization type of objectives), we say that $x\ dominates\ x'$; and $(x)\ dominates\ z(x')$. Feasible solutions $x, x' \in X$ are $equivalent$ if $z_i(x) = z_i(x')$. Y is the set that contains all non-dominated solutions.

$Supported\ nondominated$ points are the points located on the boundary of the convex hull of Y; $nonsupported\ nondominated$ points are located on the interior of the convex hull of Y. From the solution space point of view, supported efficient points are the solutions, and those can be found by the equivalent weighted sum single objective problem

$$\min\{\lambda_1 z_1(x) + \cdots + \lambda_p z_p(x) : x \in X\}$$

for some $\lambda \in R^p_{\geq}$. Accordingly, $nonsupported\ efficient\ points$ are the efficient solutions that cannot be found as optimal solutions of P_λ for any $\lambda \in R^p_{\geq}$. Each supported efficient solution is an optimal solution of some weighted sum problem (Geoffrion, 1968). It is well-known that all efficient solutions of MOLP are supported, but unsupported efficient solutions may exist for MOIP (Vincent et al, 2013). Based on these definitions, the following observations can be made: The weighted sum approach cannot guarantee the generation of the whole Pareto set for nonconvex Pareto surfaces. Aside from this, the greater the degree to which there is conflict among the objectives, the greater the degree to which the gradients of the objective functions are radially dispersed, the smaller the dominated set; the smaller the domination set, the greater the likelihood of unsupportedness.

A feasible solution $\hat{x} \in X$ is $weakly\ efficient$ if there is no $x \in X$ such that $z_i(x) < z_i(\hat{x})\ \forall i = 1, \dots, p$. The point $z(\hat{x})$ is then called $weakly\ nondominated$.

A feasible solution $\hat{x} \in X$ is $strictly\ efficient$ if there is no $x \in X, z_i(x) \neq z_i(\hat{x})$ such that $z_i(x) \leq z_i(\hat{x})\ \forall i = 1, \dots, p$. The point $z(\hat{x})$ is then called $strictly\ nondominated$.

There are also several $proper\ efficiency$ definitions in the literature (e.g., Geoffrion, 1968). A feasible solution $x \in X$ is called $properly\ efficient$ if it is efficient and if there is a real number $M > 0$ such that for all i and $x \in X$ satisfying $z_i(x) < z_i(\hat{x})$ there exists an index j such $z_i(x) < z_i(\hat{x})$.

$$\frac{z_i(\hat{x}) - z_i(x)}{z_i(x) - z_i(\hat{x})} > M$$

The corresponding point $z_i(\hat{x})$ is called a properly nondominated point. However, this definition of proper efficiency becomes obsolete if MOIP, i.e.,efficient and proper efficient sets, becomes the same set when the decision space is integer valued. For a more detailed discussion of efficiencies and their comparisons, we refer readers to Ehrgott (2005).

Several points in the outcome space serve as auxiliary points when constructing approximation sets. These are based on the following definitions

$$y_i^I = \min\{z_i(x): x \in X\},\ \ i = 1, \dots, p$$

$$y_i^{AI} = \max\{z_i(x): x \in X\},\ \ i = 1, \dots, p$$

y_i^I is called "$anchor\ point$" if the objectives are all minimization type. Then, the ideal point, y^I, the utopia point, y^U and anti-ideal, $y^{AI}$and anti-utopia, y^{AU} points are defined as follows:

$$y^I = (y_1^I, \dots, y_p^I)^T$$

$$y^U = y^I - \epsilon$$

$$y^{AI} = (y_1^{AI}, \dots, y_p^{AI})^T$$

$$y^U = y^{AI} + \epsilon$$

where $\epsilon \in R^p$ is a vector with small positive components. The range of attainable set is given by $Y^i = [y_i^I, y_i^{AI}] \quad i \in \{1, \dots, p\}$.

The set of anchor points is denoted by IM. There is another definition that is key to most of the algorithms, and this is called the "$nadir\ point$." The nadir point is a point in the design space in which all objectives are simultaneously at their worst values. Since it is not simple to find the nadir point, one can estimate it by constructing the payoff table. This table is constructed by entering all anchor points into a table. Then, one can derive the worst values for each objective in this table, $z_i^N\ i \in \{1, \dots, p\}$. When all of these z_i^N values are combined, an estimate of the nadir point can be calculated.

1.3.1) Normalization of objectives

If the ranges of objectives are significantly different, the methods (explained later in this study) cannot produce well-dispersed solution sets. So, in order to carry all of the objectives to a common scale, we perform the following normalization:

$$\text{L}=[\boldsymbol{l_1}, \dots, \boldsymbol{l_p}\ ]= [\boldsymbol{z^N - z^I}]$$

where $\boldsymbol{L \in R^p}$. Then, we calculate the normalized value of an objective as follows:

$$\boldsymbol{z_i' = \frac{z_i - z_i^I}{l_i}} \qquad \forall \boldsymbol{i} \in \{\mathbf{1}, \dots, \boldsymbol{p}\}$$

This way, all the objectives are measured on a 0-1 scale, which indicates the relative position of an objective with respect to its ideal and the nadir point. Figueira et al. (2010) used a similar scheme in their study, which converted the 0-1 range to percentages.

The pure normalization schemes are designed to normalize just the objectives. Psychologically-oriented schemes also exist, and these have been designed to obtain some effect, regarded as psychologically desired, when optimizing the achievement scalarizing function. Other schemes use the information from previous iterations to build a normalizing or preferential set of weights. Finally, the user-controlled preferential schemes can be specifically used by the DM in order to introduce preferential information to the process. It is empirically proven that, in general, these schemes produce different solutions for the same reference point (Ruiz, Luque, & Cabell, 2009).

1.3.2) Scalarization Techniques: Weighted Sum, ε-Constraint Method and others

Scalarization is a single objective related to a MOP problem with additional variables and/or parameters. It is usually solved repeatedly in order to find some subset of efficient solutions of the MOP problem (Ehrgott, (2006)). Wierzbicki (1980) discussed all relevant aspects of the main scalarization techniques specifically for MOLP problems. First, we start by describing two main approaches commonly used in MOP area: weighted sum" and the ε-constraint method.

1.3.2.1) Weighted sum

Weighted sum is a convex combination of the p objectives of MOP problem, in which the feasible set stays unchanged:

$$\min_{x \in X} \sum_{k=1}^{p} \lambda_k c_k x$$

Solutions of this technique are only supported efficient solutions with $\lambda_k > 0$.

1.3.2.1) ε-Constraint Method

In this method, one of the p objectives ($j \in \{1, \dots, p\}$) is retained for minimization and other, $p - 1$ are returned into constraints:

$$\min_{x \in X} c_j x$$

$$s.t. \;\; c_k x \leq \varepsilon_k \qquad k \neq j$$

The optimal solution for this method is weakly efficient.

The differences between the two approaches can be summarized in a succinct list as follows:

1. For linear problems, the weighting method is applied to the original feasible region and results in a corner solution, thereby generating only efficient extreme solutions. On the contrary, the ε -constraint method alters the original feasible region and can produce non-extreme efficient solutions. As a result, with the weighting method, many runs may be redundant in the sense various combinations of weights result in the same efficient extreme solution. On the other hand, with the ε -constraint method, we can exploit almost every run to produce a different efficient solution, thereby obtaining a more rich representation of the efficient set.

2. The weighting method cannot produce unsupported efficient solutions in multi-objective integer and mixed integer programming problems, while the ε -constraint method does not suffer from this disadvantage.

3. In the weighting method, the scaling of the objective functions has a strong influence on the obtained results. Therefore, the normalization of the objective functions is necessary before forming the weighted sum; in the ε-constrained method, this is not necessary.

4. An additional advantage of the ε -constraint method is that we can control the number of the generated efficient solutions by properly adjusting the number of grid points in each of the objective function ranges. This is not as easy with the weighting method (Mavrotas, 2009).

1.3.2.3) Tchebycheff Method and Achievement Scalarization

The structure of scalarization that will be explained strictly relies on the concept of "order preserving functions," which can be defined as follows:

A function $s(\bar{q}, q_1)$ is order preserving if and only if $q_2 < q_1 \rightarrow s(\bar{q}, q_2) < s(\bar{q}, q_1)$, where $\bar{q}$ is a constant. Wierzbicki (1980) describes many scalarization functions of this type that use a reference point (aspiration level) to generate non-dominated points. Among those, the following functions happen to be the most popular ones throughout the MOP literature due to their linear or linearizable structure.

Augmented weighted Tchebycheff method:

This method is especially popular for interactive methods (Steuer & Choo, 1983); (Steuer, Silverman, & Whisman, 1993). This method considers the distance between a feasible point Cx in criterion space and the ideal point y^I:

$$\min_{x \in X} \max_{k=1 \ldots p} v_k(c_k x - y_k^I) + \gamma \sum_{k=1}^{p} (c_k x - y_k^I),$$

where $v > 0$ is a vector of weights. If $\gamma > 0$ an optimal solution, x^* is efficient; on the other hand, if this augmentation is avoided, it can generate weakly efficient points. The non-linear "max" term can be linearized by adding a variable and p number of constraints to the model, which will be explained in detail in the next section where proposed algorithm is presented.

Achievement scalarization and reference point functions:

The most general form of the achievement scalarization function is as follows:

$$\min_{x \in X} \max_{k=1 \ldots p} v_k(c_k x - \rho_k) + \gamma \sum_{k=1}^{p} (c_k x - \rho_k),$$

where ρ_k is the reference point for objective k, $v > 0$ is a vector of weights, and $\gamma > 0$. In Wierzbicki (2000), this function has been called a "prototype" achievement scalarizing function. This function is order preserving and is preferred mostly for MOIP problems. For general cases, other versions of this function can be obtained by replacing $(c_k x - \rho_k)$ with just $c_k x$, as it is applied by Karasakal and Koksalan (2009). Scaling objective values play an important role in terms of preserving the properties of these scalarization functions.

CHAPTER II: REPRESENTATIVE SET GENERATION FOR MULTI-OBJECTIVE LINEAR PROGRAMMING PROBLEMS

Vector maximization approaches aim to find all efficient solutions to present to the DM (Sayin, 1996). Since the exact solution set is often not attainable, an approximate description of the solution set becomes an appealing alternative (Ruzika & Wiecek, 2005). Hence, the reasons for developing "approximating approaches" in lieu of exact methods can be summarized as follows:

- To represent the solution set when this set is numerically computable (linear or convex MOPs) in order to have a general idea of the Pareto front space rather than all Pareto solutions;
- To approximate the solution set when some, but not all, of the efficient or Pareto points are numerically computable (nonlinear MOPs);
- To approximate the solution set when the efficient or Pareto points are computationally prohibitive to solve (discrete MOPs).

These approximations can be obtained in the form of set points or surfaces. Discrete (point-wise) approximations are among the simplest forms of approximations and are called approximations of the 0^{th} order. In this approach, the Pareto efficient solutions generated by a particular solution method serve as the approximating points and no further structure is computed. Other approximation forms in the literature are piecewise linear (1st order), quadratic (2nd order) and cubic (3rd order) approximations of the Pareto front.

While finding a discrete set of points that are well dispersed over the nondominated frontier and that represent all parts of the frontier is desirable, it is also important to achieve this with a reasonable amount of computational effort. Further, a solution set is usually not

considered a good approximation of the efficient set in terms of "representativeness" if it contains too many points that do not necessarily cover the Pareto front while spreading uniformly. These observations motivate the search for discrete representations of the Pareto front that consist of efficient points that are different from the extreme points (Sayin, 2003).

The approximation algorithms can be classified based on the underlying methodology. Some approximation approaches are exact in the sense that they only find efficient (locally or globally) solutions, while other approaches are heuristic (i.e., they do not provide efficiency guarantee, like NSGA II, and are commonly used due to good time performance). Our focus in this work is on the approximation methods that guarantee efficiency of the solutions generated.

All observations and classifications have led to the goal of this work, which is to develop a practical approximation method that generates representative approximations of the Pareto front such that the points generated are guaranteed to be efficient. This chapter starts with a brief review of literature. It is followed by details of the proposed algorithm. The next section summarizes two methods from the literature that are used as benchmark algorithms. We first describe the quality measures used to assess the performance of the approximation techniques and finally present the experimental results. The chapter concludes with the discussion of results and future research directions.

2.1) Related Literature

The subject of the approximation of the Pareto set of MOPs has been of interest to scientists for almost forty years. To the best of our knowledge, pioneering studies in this field were presented in 1970s (Polak & Payne, 1976).

In this section, aside from the proposed algorithm, we want to cover three studies that are important for their resemblance to the proposed algorithm, either in terms of the general idea

behind the algorithm or their purposes. Sayin (2003) proposed an approach for generating a representative subset from the nondominated frontier while guaranteeing a specified level of quality. The procedure starts with the generation of efficient faces. Then, the points are added to the solution set in a way that a desired level of the coverage error is reached. When the nondominated set is nonconvex, a procedure called REPR-D is used that generates representations of faces sequentially considering all of the points included in the face representations so far (Sayin, 2003). The approach performs well in terms of the quality of the representation. However, it requires the generation of all nondominated faces before creating the representative points. Hence, there are computational difficulties in the generation of the nondominated faces and in obtaining representative points (which requires solving 0–1 integer programs), especially for substantial problems. Since Sayin's (2003) approach is also used to assess the coverage measure, the procedure of generating efficient faces is covered in detail in section 2.5.1) Coverage Measure

In 2009, Karasakal and Koksalan presented another algorithm for representative set generation purposes. The general idea behind the algorithm is that the Pareto surface is approximated with a fitted surface, and reference points are generated systematically on this surface. The authors used an edited version of the algorithm proposed by Sayin (2003) for benchmarking purposes, called REPR. Details of this algorithm, along with its complexity analysis, will be presented in more detail in 2.6) Experimental Results.

The final study we want to mention is called the "Normal Constraints" method (Mattson, Mullur, & Messac, 2004) and is important in terms of supplying the idea of dividing the criterion space into equal grids on our algorithm. After presenting some rigorous information about this approach, we will continue with the details of the proposed approach.

2.1.1) Normal Constraints (NC) Method

The NC method was originally designed for MOP problems, and it guarantees an even distribution of the solution set (Mattson et al., 2004). The idea for the algorithm is to convert the MOP problem into a single objective optimization problem that is solved repeatedly, subject to a judiciously constructed set of constraints. These constraints are placed evenly on a surface, which is called an "anchor plane." Accordingly, the search is performed within this restricted area. Hence, the algorithm produces evenly distributed solution points.

The graphical representation of the algorithm in two and three objective problems are shown in the figures below.

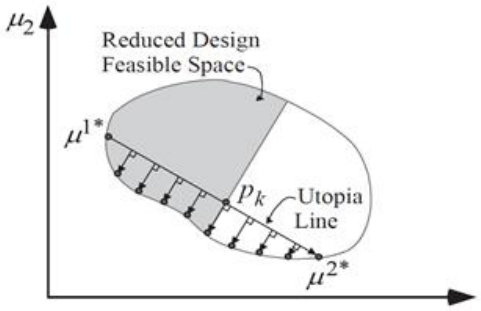


Figure 1: Segmentation of utopia line in a two objective MOP by NC (Mattson et al., 2004)

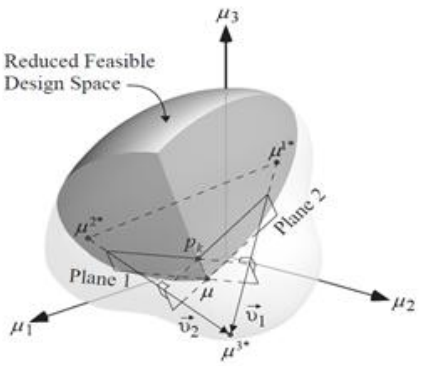


Figure 2: Segmentation of the search space in a three objective MOP by NC (Mattson et al., 2004)

The steps of the algorithm are summarized in Table 1:

Table 1: Steps of Normal Constraints method

STEP 1	*Obtain anchor points*: z^{i*} for all for $i\epsilon\{1,\ldots,k\}$
STEP 2	Normalize objective vectors
STEP 3	Set $j=1$ where $j\epsilon\{1,\ldots,k\}$
STEP 4	Define the Utopia Plane Vectors for objective j
STEP 5	Compute normalized increments
STEP 6	Generate hyperplane points, N
STEP 7	Generate Pareto points
STEP 8	If the model is solved N times and $j \neq k$ return to STEP 4; otherwise continue with STEP 9
STEP 9	Return to the non-normalized space

This algorithm is repeated p times, or the number of objectives. Anchor points and the nadir point are obtained using the method that was described in Preliminaries.

All explanations for the other important steps of the algorithm can be found in Mattson et al. (2004). The advantages of this algorithm can be summarized as follows:

- Overall CPU time requirement of the algorithm is reasonable, and not much is affected by the size of the problem after tuning the partition parameters;
- It can be run in parallel; and
- It guarantees the representativeness of the solution set.

On the other hand, an important drawback of this approach is the objective function used during optimization. In other words, the solution set is constructed by optimizing only one of the objectives within the grids, which are constructed based on all the remaining objectives (i.e., without relying on a scalarization technique). This increases the chance of generating points that can be dominated.

2.2) Proposed Algorithm

The general idea of the proposed algorithm is to solve the achievement scalarization function in a way that guarantees dispersion of the solution points that are Pareto optimal. The scalarization function is constructed such that the maximum distance among all objectives to a reference point is minimized. In order to achieve a well dispersed solution set, the reference points generated in the objective space should have two properties. First, the reference points should be well dispersed on the whole Pareto surface. In order to achieve this, equal grids are constructed on the objective space using all but one of the objective axes. The second property is to ensure that the reference points remain as close as possible to the Pareto front. This property also improves the computational efficiency by strengthening the scalarization function formulation. In the following sections, we explain the details of the algorithm by providing the terminology and definitions used in the algorithm.

The first step of the algorithm is to identify the most desirable points of the objectives in the feasible space in terms of each objective (anchor points,z_i^*). Then, the nadir point z^W is estimated by constructing the payoff table, as explained in preliminaries section.

2.2.1) Partition parameter ($\boldsymbol{T}$)

This parameter controls the number of locally Pareto optimal points in the generated solution set. One extreme value for this parameter is "1," which means no grid partitioning. In

this case, the algorithm produces at most p unique points, all of which are anchor points. As T approaches infinity, the grid size decreases to zero and the CPU time requirement of the algorithm approaches infinity. Hence, there is a direct relationship between the algorithm's total runtime and the partition parameter T.

2.2.2) Determining grid boundaries

As mentioned previously, the algorithm relies on the idea of dividing the objective space into equal grids. In order to do this, the range of all objectives is divided into T equal parts, i.e., $I_i = (z^N - z_i^*)/T$, for all objectives except for the "main" objective, j, selected for each loop, $i, j \in \{1, .., p\}$ and $i \neq j$. A column vector, C_i, is used to store the boundaries of each grid for the objective i. The first element in a row of all vectors are the values of the anchor points, i.e., $C_i = z_i^*$. By adding I_i to the previous row in C_i, all of the T rows are filled.

There are two different mathematical models that need to be solved during the algorithm, the first of which is generating the reference point at each grid. This model is called the "intermediate model" and is expressed as follows:

$$Min\ z_j$$

$$s.t.\ \ x \in X$$

$$lb_i \leq z_i \leq ub_i \quad \forall i \in \{1, \dots, p\}; i \neq j$$

In this formulation LB_i and UB_i are determined by the boundaries of the current grid. After solving for this model, the best point in terms of main objective,j,z_j^*, is obtained. Then, a reference point, z^0, is calculated using this value. Finally, a model with an achievement scalarizing function is solved with the reference point in order to identify the closest Pareto point, which will be presented later. By repeating the same process on each grid, we complete the exploration for objective j. After completing all of the grids for objective j, we change the

index of main objective from j to $j+1$. Hence, we explore all parts of the objective space from all aspects by setting each objective as the main objective one time.

2.2.3) Achievement scalarization of reference points

The reference points are projected onto the nondominated frontier by an achievement scalarizing problem, which is formulated as follows:

$$\min \alpha + \gamma \sum_{k=1}^{p} w_i(c_k x - z^0),$$

$$s.t. \qquad \alpha \geq w_i(c_k x - z^0) \qquad i = 1, \dots, p$$

$$x \in X$$

In this formulation, X is the feasible decision space and γ is a small positive constant that avoids dominated solutions. The achievement scalarizing program always finds a feasible nondominated point if any exists (Karasakal & Koksalan, 2009).

This scalarization function minimizes the maximum of the Tchebycheff distances between the reference points and the solution point. The emphasis is on one objective, which has the maximum distance to the desired point from the reference point. In addition, the distance to the other objectives is also taken into account through the augmented part of the equation with a weight multiplier, γ.

There are three important parameters in this formulation: the reference point z^0, the weighting coefficients, w, and the coefficient of the augmentation part, γ . The only parameter that is selected randomly is γ. All other parameters are set by some procedures as part of the proposed algorithm, and this is explained in detail in the following sections. To the best of our knowledge, there is no theoretical or empirical study in the literature about setting γ . However, it is required to be $0 < \gamma < 1$. Steuer and Choo (1983) proved that a γ always exists that is

small enough to obtain all the non-dominated sets for cases with finite-discrete and polyhedral feasible regions. In this study, we choose γ as 0.001, which is sufficiently large enough not to be ignored by the first part of the objective function and small enough not to dominate the same part.

2.2.4) Finding the reference points

The algorithm is executed for each coordinate in the objective space; that is, optimization is performed by focusing on each of the objectives one by one. The remainder of the objectives are divided into equal parts, and the grids are formed. Let the main objective be denoted by j again, where $j \in \{1, \dots, p\}$. The grids are then obtained by dividing each objective i axis into "partition parameter," T, many parts, where $\forall i \in \{1, \dots, p\}$ and $i \neq j$.

In order to maintain both approximation and representativeness in the reference points, this study proposes calculating reference points that combine the information obtained from intermediate model and grid corners. The first source that supplies one of the coordinates of the reference point is the point obtained by optimizing the j^{th} objective (i.e., the current main objective) on the current grid, and this serves to obtain a point that approximates to the Pareto front. The objective value of this intermediate model for the j^{th} objective, z_j^*, is then assigned as the j^{th} coordinate value of reference point. The second source of information is the boundaries of current grid. The $p - 1$ coordinates of the reference point are determined based on the current grid, and incorporating this information helps to spread the reference points diversely. This procedure is illustrated for the three objective cases in Figure 3 with an example. It is then formalized for the general case.

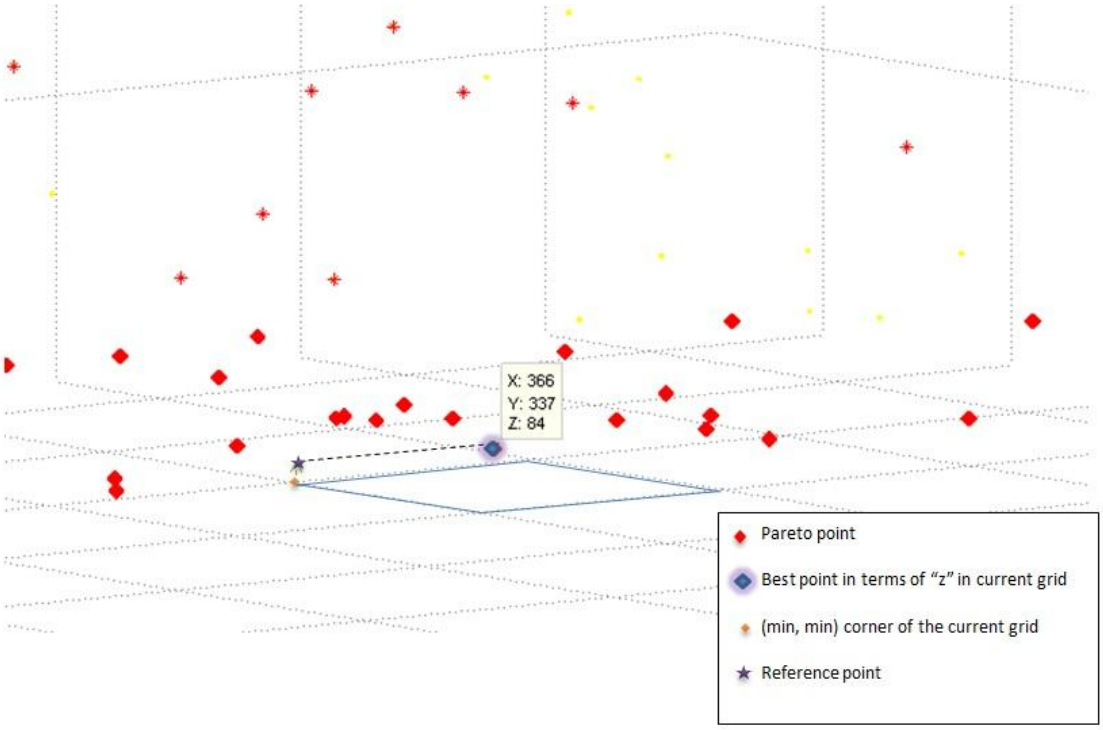


Figure 3: Three dimensional representation of how reference points are generated

In Figure 3, the current grid is denoted by a blue square lying on the x-y plane; in the intermediate model the third objective, the z dimension, is the main objective and is optimized to find the best point in terms of the third dimension. In this example, the objective value of the intermediate model is 84. The information from the current grid is then incorporated. Since all the objectives are assumed to be the minimization type in this example, the corner of the grid, whose coordinates are determined by the minimum of x, 366, and the minimum of y, 337, are incorporated in the reference point. Hence, the final reference point is (366, 337, 84).

In general, the first step is to identify the best point in terms of the main objective, j, z_j^*, by optimizing the intermediate model on the grid. Next, the coordinates of the minimum (maximum), $LB_i(UB_i)$ corner points of the grid are determined if the existing objectives of the MOP are of the minimization (maximization) type. Finally, a reference point, z^0, is calculated by equating $z_j^0 = z_j^*$; $z_i^0 = LB_i(UB_i)\ \forall i \in \{1, \dots, p\}, i \neq j$.

Figure 4 and Figure **5** represent a sample for a bi-objective MOIP case in which both objectives are of the minimization type. The yellow points below the red line are the reference points generated (e.g., 32 represents the reference point generated on the '3rd' grid while optimizing objective '2'). The blue dots on in Figure 4 are the Pareto points that can be obtained after solving the achievement scalarization model on the corresponding grids (i.e., the Pareto solutions in the feasible region, above the red line).

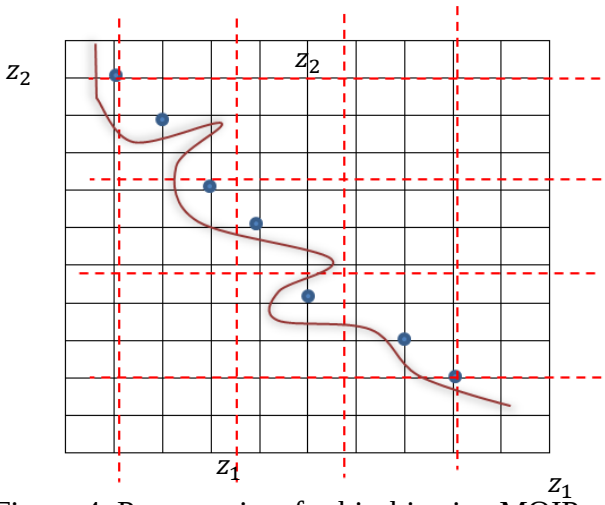


Figure 4: Pareto points for bi-objective MOIP

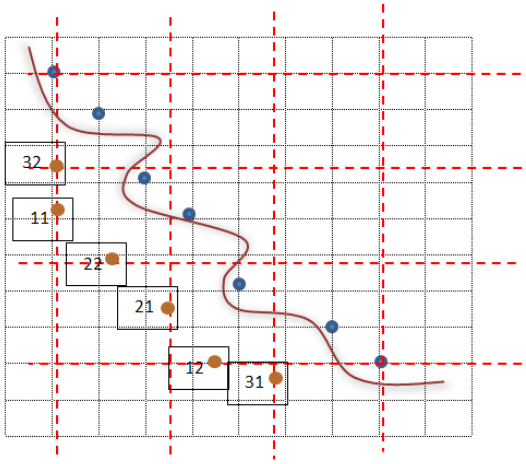


Figure 5: Reference points for bi-objective MOIP

2.2.5) Finding the weighting coefficients

Ruiz et al. (2009) studied the weighting schemes in reference point procedures. Illustratively, if the reference point is outside the feasible region, or are not achievable, the non-positive orthant is projected from the reference point following the direction given by the weight vector, $w = [w_1, \dots, w_p]$, until it touches the efficient frontier. If the reference point is achievable, the solution is the last nondominated point that the nonpositive orthant touches, as shown in Figure 6 and Figure **7**. In this algorithm, "***GUESS***" and "***STOM***" weighting schemes are implemented, which are expressed as follows, respectively:

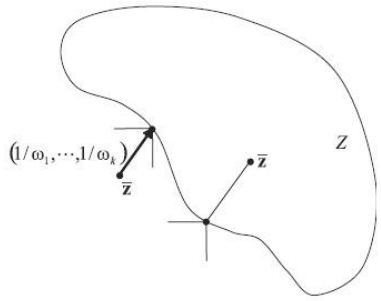


Figure 6: "GUESS" type of weighting scheme, Ruiz et al. (2009)

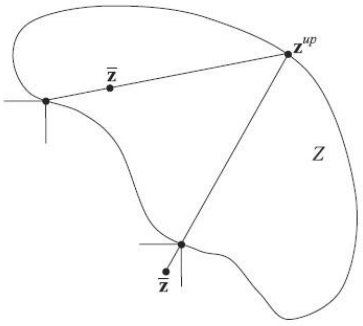


Figure 7: "STOM" type of weighting scheme, Ruiz et al. (2009)

$$w_i = \frac{1}{\bar{z}_i^U - z_i^0} \quad \text{(GUESS)}$$

$$w_i = \frac{1}{z_i^0 - \bar{z}_i^I} \qquad \text{(STOM)}$$

where $\bar{z}_i^U = z_i^N + \varepsilon$. The STOM scheme is preferred when the problem is of the minimization type; the GUESS is suggested for maximization problems. Throughout the experimental runs of the thesis, ε will be 0.001.

However, this calculation is not enough since a true weighting scheme should have the property of $0 < w_i < 0$, $\sum_i^p w_i = 1$. In order to map the results of the above calculation on a [0,1] scale, we apply the following scaling:

$$w_i = \frac{1}{\sum_{i=1}^{p} \frac{1}{\bar{z}_i^U - z_i^0}} \left(\frac{1}{\bar{z}_i^U - z_i^0}\right)$$

After presenting all of the details for the proposed approach, we will summarize the algorithm. The following steps represent the outline of the proposed algorithm:

- **Step 0.1**: Obtain anchor points, $z^{i*}, \forall i = \{1, \dots, p\}$, and estimate the nadir point, z^N, from the payoff table.
- **Step 1.0 (Generating reference points):** Set the number of partitions, T; initialize solution set, $E = \emptyset$, and the reference point set, $R = \emptyset$.
    - **Step 1.1:** Select the j^{th} objective, $j \in \{1, \dots, p\}$ as the main objective;
    - **Step 1.2:** Divide $\forall i \in \{1, \dots, p\}$, $j \neq i$ into T parts, set the grid boundaries $[lb_i, ub_i]$; in the total construct, $K = T^{p-1}$ number of grids for objective j.

- **Step 1.3:** Solve the *intermediate model*, get z_j^*, and construct the reference point, z^0, using z_j^* and $lb_i (or\ ub_i)$; populate the reference point set with z^0 (i.e., $R = \{z^0\} \cup R$)
    - Repeat steps 1.1-1.3 for ∀j ∈{1,…,p} and construct K grids for each j.

- **Step 2.1(Generating representative points):** Solve the *achievement scalarizing model* for $\forall\ z^0 \in R$, get the representative Pareto solution $z,^*$ and update solution set $E = \{z^0\} \cup E$.

2.3.1) Overview of the Benchmark Methods: Outer Surface Approximation-Based Approach

The algorithm proposed by Karasakal and Koksalan (2009) starts by solving the series of augmented Tchebycheff programs (Steuer 1986) systematically and introducing different lower bounds to each of the objectives. As a result, an initial set of nondominated solutions is obtained. Afterwards, a surface is fitted that approximates the shape of the nondominated frontier by minimizing the sum of square distances from the selected nondominated points. A set of approximately evenly spaced reference points on the fitted surface are then selected, and each reference point is projected onto the nondominated frontier in the gradient direction of the fitted surface. When the fitted surface approximates the nondominated frontier well, the projections of these reference points are expected to be approximately uniformly distributed over the nondominated frontier and to form a good representation, i.e. coverage. An illustration of this concept is presented in Figure 8 for two objectives.

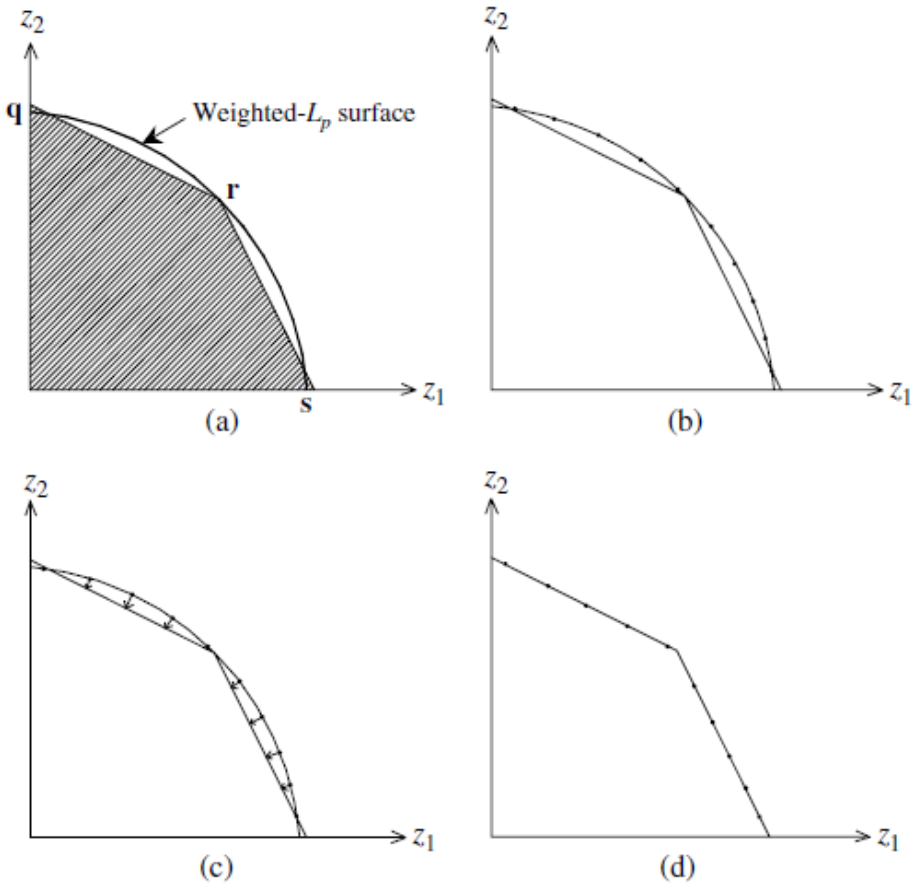


Figure 8: An illustration of the approach by Karasakal and Koksalan (2009)

The following is a summary of the steps of the algorithm, presented in a similar way as the proposed approach in order to show similarities and differences more clearly:

- Step 0.1: Obtain anchor points and estimate the nadir point from the payoff table, $z^{i*}\ \forall i = \{1, \dots, p\}$ and z^N.
- Step 1.0: Generate the initial points to fit the space solving of augmented equal-weighted Tchebycheff programs (i.e., if the size of initial set is *k*.)
- Step 1.0 (Fitting the surface)
  - Step 1.1: Find a nondominated point b that is the minimum Tchebycheff distance from the ideal criterion vector in the diagonal direction.

- Step 1.2: In the weighted $-Lm$ function, set $\lambda_i = 1$ and $z_i = b_i$ $(i = 1, \dots, p)$ and solve $\left[\sum_{i=1}^{p}(b_i - q_i^*)^m\right]^{1/m} = 1$ using a bi-section to determine m.
- Step 1.3: After p is determined, new λ values are calculated using $\lambda = (\lambda_1, \dots, \lambda_p) = R^{-1}S$, where R and S are two matrices, presented within the text previously, that contain the initial set and nadir point information.

- Step 2.0 (Generating reference points):
  - Divide the curve lying on the $z_1 z_m$-plane into approximately equal-length arcs using a step size of Δz for $i = \{1, \dots, p-1\}$.
  - Project the endpoints of the arcs on the $z_1 z_m$-plane onto the z_i-axis for all $\{i = 1, \dots, p-1\}$. (Let n_i be the number of projections on the z_i-axis.)
  - Find $n_1 n_2 \cdots n_{p-1}$ points on the $z_1 z_2 \dots z_{p-1}$ hyperplane as the intersection of points projected to each of the axes.
  - Project each of these $n_1 n_2 \dots n_{p-1}$ points onto the weighted-Lp surface by fixing objectives $1,2, \dots, p$ at their corresponding values and maximizing objective m over the weighted-Lm surface to find the value of z_p that corresponds with the combination under consideration.

- Step 2.0 (Generating representative points):
  - Solve the model with the achievement scalarization objective for each reference point.

2.3.2) Overview of the Benchmark Methods: Modified REPR Algorithm

This algorithm relies on generating efficient faces of convex Pareto fronts and is based on Sayin (2003). Karasakal and Koksalan (2009) modified the algorithm to generate representative points, as follows:

- Step 0.1: Generate efficient faces of the Pareto front and initialize coverage error as 1 for all faces.
- Step 1.0: Generate a representative point by considering the worst representative point in a face.
    - Step 1.1: If the last representative point is already in the set, discard it; if not, add it to the final set.
    - Step 1.2: Calculate the new coverage error for the current face.
    - Step 1.3: Check if the desired number of points is generated. If true, stop the algorithm; if not, continue with the face that has worst coverage error.

Details regarding the efficient face generation and coverage error calculation steps are presented in the 2.5) Quality Measures for Approximation Techniques since these models have been also used to calculate the coverage error for the proposed approach.

2.4) Complexity of the Algorithm and Advantage over Benchmark algorithms

The number of optimizations within each grid is restricted to two linear optimization procedures; one optimization is required for finding the local optimum, and the other to identify the closest Pareto optimal solution. Calculation of reference points and weights of the scalarization model are not affected by any other parameters of the algorithm or the problem. Thus, there is a constant number of iterations for each grid. This means that the complexity of the algorithm is only affected by the number of partition parameter, *T*, and the number of objectives, p.

As summarized previously, Karasakal and Koksalan's (2009) algorithm requires an initial effort for surface fitting in order to determine the approximation surface and generate reference points on this surface. The proposed algorithm follows a relatively straightforward approach to

generate good reference points that are both well approximated and diverse. Algorithm complexity is designed to compare two algorithms at the idea level, but this ignores low-level details such as the implementation programming language, the hardware the algorithm runs on, and the instruction set of the given CPU (Leiserson, Rivest, Stein, & Cormen, 2001). Hence, a detailed complexity analysis of both algorithms is presented below in order to show the differences between the two approaches in terms of time requirement.

Common notation:

p: number of objectives

R: number of reference points

Complexity Analysis of Proposed Approach

T: number of partitions ($R = (p * T^{p-1})$)

Step 0.1*:* p many LP (MIP) solves to find anchor points; p comparisons to construct payoff table.

Step 1.0 (Generating reference points):

Step 1.2: $(\mathrm{p}-1)$R many summations to determine grid boundaries

Step 1.3: R many LP solves (intermediate model)

Step 2.1(Generating representative points): R many LP (MIP) solves with scalarization objective.

In Total:

$(R + p)$ LP (MIP if the problem is MOIP) solves+

R LP solves+ $(p-1)R$ summations

Complexity Analysis of Karasakal and Koksalan (2009)

k: Size of initial set used to fit the surface

Ti: Number of partition on $\frac{p*(p-1)}{2}$ many planes determined by step size, Δz

Step 0.1: p many LP(MIP) solves to find anchor points; p comparisons to construct payoff table

Step 1.0 (Fitting the surface)

Step 1.1: k many LP(MIP) solves

Step 1.2: $2p$ summations (to calculate the special direction for point b)

Step 1.3: At least "1" bi-section solve ($\left[\sum_{i=1}^{p}(b_i - q_i^*)^m\right]^{1/m} = 1$) to find m.

Step 1.4: $2m * p^2$ multiplications+ $[p^2 * (pk + p + 1)]$ summations to construct R matrix; matrix inversion of R; $m * p$ multiplications+ $p * (pk + p + 1)$ summations to construct S matrix; p^2 summation+ p^2multiplication (for matrix multiplication) to find weights.

Step 2.0 (Generating reference points)

Step 2.1: $4R$ summations to partition line segments

Step 2.4: R many **order of m** model solves (i.e., non-linear models when m>1)

Step 3.0 (Generating representative points): R many LP (MIP) solves

In Total:

$[((k+1)p^3 + (k+3)p^2 + 3p)] + 4R]$ summations +

$((2m+1) * p^2 + p * m)$ multiplications+

1 bi-section solve + matrix inversion+

$R + k$ many LP (MIP) solves+

R many order of m model solves (nonlinear if $m > 1$)

When the totals of all computations are compared, including the basic but minor computations such as summations or multiplications, it can easily be observed that the proposed approach requires significantly less computational effort. It is worth emphasizing that the total calculation effort calculated for Karasakal and Koksalan's (2009) algorithm considers only one pass to determine an "m" parameter to approximate the surface. However, the most significant difference between the two algorithms is the number of optimizations that are needed to generate the reference points. Unless the fitted surface function is an order of 1, Karasakal and Koksalan (2009) propose to solve non-linear models to determine reference point, whereas our algorithm proposes to solve same number of linear programs for any problem.

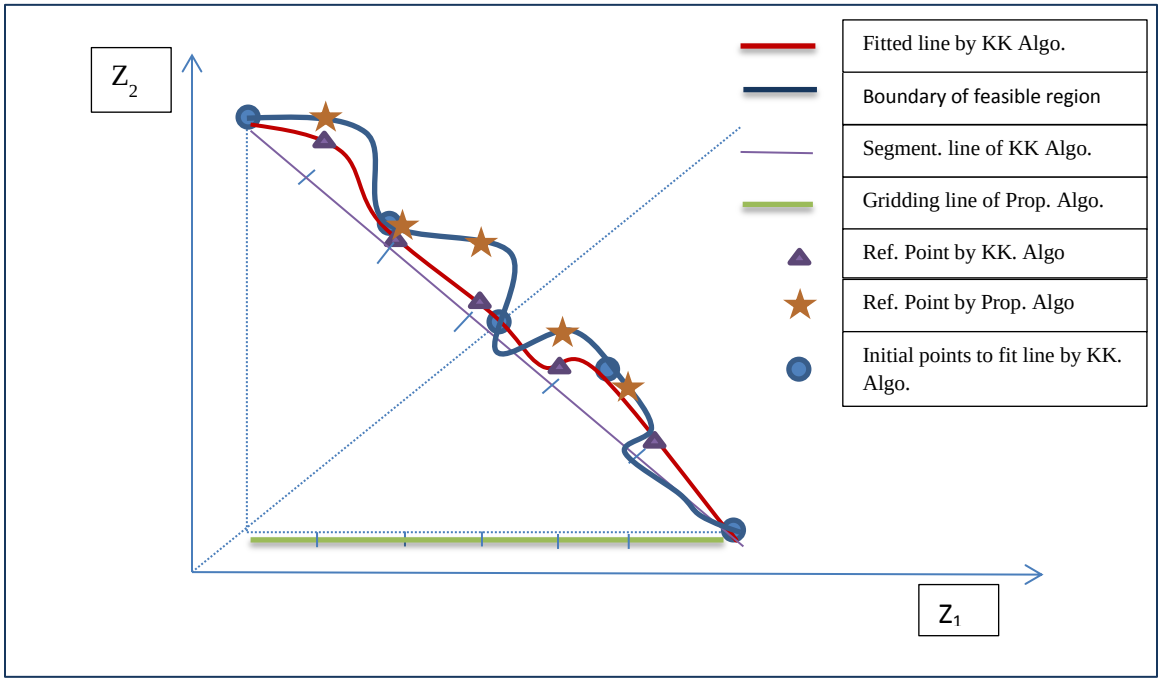


Figure 9: An illustration of how reference points are generated by the proposed algorithm and Karasakal and Koksalan's (2009) algorithm on a two objective MOP with the maximization type of objectives and a non-convex Pareto front

The functions of Karasakal and Koksalan's (2009) algorithm and the proposed algorithm are illustrated in Figure 8 above on a MOLP with two objectives. The proposed algorithm starts with initial points to approximate the surface, which are represented by blue dots on the figure . Indeed, the blue dot on the diagonal line drawn in the direction of the nadir to ideal point is a special point that is generated on purpose. This is called point b, since it is considered one of the most important points that gives insight about the shape of actual Pareto front. A high order line (it is a line rather than a surface since it is an example with two objectives) is represented by red is fitted, and this passes through the initial points. The border of the feasible region on the objective space is represented by a blue line; points above this blue line are feasible solutions. Karasakal and Koksalan propose to draw a segmentation line first and determine equally distant points on that line; the fitted surface line then needs to be optimized by fixing one of the objective values to the corresponding objective value of previously generated, equally distant points, one by one. Through this procedure, they obtain the points represented by purple triangles. Although the resultant points lie more uniformly on the fitted line, they are generally staying in a distance to the feasible region. On the other hand, reference points generated by the proposed approach lie in the feasible region since we propose to optimize one of the objectives on each grid. This might become an advantage, especially with problems of the non-convex Pareto front, because a fitted surface might miss some non-convexities; therefore, the algorithm proposed by Karasakal and Koksalan (2009) might generate reference points that are distant from the actual feasible space; despite this, these reference points will result in some Pareto points since the algorithm requires solving an achievement scalarization objective that guarantees Pareto optimality, some regions still might remain not well represented at these type of cases.

2.5) Quality Measures for Approximation Techniques

2.5.1) Coverage Measure

In Sayin (2000), the coverage error is defined as follows:

> Let $\epsilon > 0$ be a real number. Let D $\subseteq$Z be a discrete set. D is called a d_ϵ representation of Z, if for any z $\in$ Z there exists y $\in$ D such that d(z, y)$\leq \epsilon$.

Coverage has other definitions in the objective space, one of which is proposed by Wu and Azarm (2001), denoted by "OS":

$$OS(D)=\prod_i | \max_{y\in D} y_i - \min_{y\in D} y_i |$$

where $i = 1 \dots p$; this measure gives an idea about the spread of the points in the representative set.

In order to calculate the measure proposed by Sayin (2000) on the decision variables space for MOLP problems, there are a couple of steps that require solving two different mathematical programs. The first program is used to determine the efficient faces of faces MOLP problems, and it is a linear program. The second program is a mixed integer program that is used to calculate the coverage measure. Since these steps are also the main steps of the algorithm used for benchmarking in Karasakal and Koksalan's (2009) study, this is also an important procedure that needs to be mentioned.

2.5.2.1) Calculating Efficient Faces

The following definitions need to be outlined in order to explain the algorithm that aims to generate all efficient faces. Let F be a subset of X, where X is the feasible decision space of the MOLP problem. F is a face of X if every line segment in X with a relative interior point in F has both end points in F. A face F is an efficient face if all the elements of F are efficient.

Based on these definitions, Sayin (1996) proposed the following model to find the efficient faces of a MOLP problem:

$$\boldsymbol{SP1}: Max\ v_1 = e^T Cx - e^T Cy$$

$$s.t.\ \bar{A}x \leq \bar{b},$$

$$\bar{A}y \leq \bar{b},$$

$$A^{-I}y = b^{-I},$$

$$-Cx + Cy \leq 0,$$

where $x \in R^n$ and $y \in R^n$; and

$$\bar{A} = \begin{bmatrix} A \\ -I_n \end{bmatrix} \text{ and } \bar{b} = \begin{bmatrix} b \\ 0 \end{bmatrix},$$

where I_n is the $n \times n$ identity matrix and $0 \in R^n$. It should be noted that X can be rewritten as $X = \{x \in R^n | \bar{A}x \leq \bar{b}\}$. Let $M = \{1, \dots, m+n\}$ and $\mu = \{I | I \subseteq M\}$. Define A^{-I} as the matrix derived from $\bar{A}$ by deleting rows of $\bar{A}$ not in I; b^{-I} is defined as the vector derived from $\bar{b}$ by deleting elements of $\bar{b}$ not in I. For $I \in \mu$, define $F(I) = \{x \in X | A^{-I}x = b^{-I}\}$. $I \in \mu$, $F(I)$ represents a face of X (Yu & Zeleny, 1975). Note that $F(\emptyset) = X$, and for $I \in \mu$, $F(I) = \emptyset$ is possible. We will refer to $F(I)$ as a proper face of X if $F(I) \neq \emptyset$.

The proposed algorithm for finding X_E is based on checking elements of μ^k starting with k=0. For k=0, the only element of $\mu^k\ is\ \emptyset$ and $F(\emptyset) = X$. Thus, by solving problem ($SP_\emptyset$), the algorithm first checks whether the problem (MOLP) is completely efficient (Benson & Sayin, 1994). The algorithm terminates with the conclusion $X_E = X$. If not, then for each element I of μ^I, problem (SP_I) is solved. The following three rules are followed based on the solutions of each problem:

- If (SP_I) is infeasible: I is dropped from further consideration since $F(I) = \emptyset$; it is then placed on a list that keeps index sets yielding infeasible combinations, IFS.
- If (SP_I) has an optimal value of "0": I is dropped from further consideration since $F(I)$ is efficient. I is placed in another list that keeps index sets yielding efficient faces, FS.
- If (SP_I) is unbounded or has positive optimized value: It is concluded that $F(I)$ has at least one element that is not efficient. Therefore, it is possible that $F(J) \subset F(I)$ efficient; thus, supersets of I,i.e., $J \supset I$, should be checked via solving (SP_J). This means that immediate supersets of I , e.g., index sets that contain I and belong to $\mu^{|I|+1}$, are placed in a list for later consideration, CFS. After all the elements of μ^I are considered, the index sets that were placed in the CFS. Before solving (SP_J), the following checks are performed:
    - If J$\supset IFS_m$ $m \in \{1 \dots |IFS|\}$, then J also generates an infeasible solution and is discarded.
    - If J$\supset FS_m$ $m \in \{1 \dots |FS|\}$, then J also generates a feasible solution and is discarded.
    - If none of these is true, (SP_J) is solved and placed in appropriate set based on its solution.

The procedure stops when there are no furthere faces to consider.

2.5.2.2) Models to Solve in Order to Obtain Lower and Upper Bound of Coverage Measure

For the representation of the Pareto front of a MOLP problem, the coverage measure is calculated based on the efficient faces that are calculated following the method previously described. After coming up with representative solution set, D, and the efficient faces set, FS, the

last step is to calculate the coverage measure. Sayin (200) proposed several mathematical models in order to calculate the lower and upper bound bounds for coverage error. Based on these definitions, consider a d_ϵ- representation D of set $Z \in R^n$. Take an element z of Z. The point x in D that represents z is the one that is closest to z. If the Tchebycheff distance between x and z is $l^\infty(x,z)$, it is known that all distances $(|x_j - z_j|, j = 1, \dots, n)$ between x and z are less than or equal to $l^\infty(x,z)$ in all coordinates. Following this, the mixed integer problem is solved to determine $l^\infty(x,z)$, based on following definitions: Let M denote a sufficiently large positive number. Define $e_n \in R^n$ and $e_N \in R^N$ as vectors whose entries are all 1. Let $d = (d_1, \dots, d_N) \in R^N, z, u^i, o^i, t^i, s^i \in R^n, i = 1, \dots, N$ denote the variables.

$$\boldsymbol{MP}(\boldsymbol{l}^\infty): \qquad \max \epsilon$$

$$s.t. \qquad e_N\epsilon - d \leq 0, \qquad (1)$$

$$-e_n d_i + z + u^i = x^i \qquad i = 1, \dots, N \qquad (2)$$

$$e_n d_i + z - o^i = x^i \qquad i = 1, \dots, N \qquad (3)$$

$$u^i - Mt^i \leq 0 \qquad i = 1, \dots, N \qquad (4)$$

$$o^i - Ms^i \leq 0 \qquad i = 1, \dots, N \qquad (5)$$

$$e_n^T t^i + e_n^T s^i \leq 0 \qquad i = 1, \dots, N \qquad (6)$$

$$z \in Z,$$

$$u^i, t^i \geq 0 \qquad i = 1, \dots, N$$

$$t^i, s^i \in \{0,1\} \qquad i = 1, \dots, N$$

In addition to the Tchebycheff distance version of the model, $MP(l^\infty)$, which gives the upper bound on the coverage error, the rectilinear distance also needs to be calculated by solving $MP(l^1)$. Building on the model defined for $MP(l^\infty)$, additional variables are defined $a^i \in$

$R^n, i = 1, \dots, N$. Then, the 2nd and 3rd constraints in the above model are replaced with the following equations:

$$-a^i + z + u^i = x^i \quad i = 1, \dots, N$$

$$a^i + z + u^i = x^i \quad i = 1, \dots, N$$

The 6th equation is replaced with following equation:

$$t_j^i + s_j^i \leq 1 \; i = 1, \dots, N \;\; j = 1, \dots, n$$

The following equation is added to the formulation:

$$d_i - e_n^T a^i \leq 0 \quad i = 1, \dots, N$$

Keeping the set of efficient faces, FS, $MP(l^\infty)$ and $MP(l^1)$ are used to measure the coverage error over individual faces, where $MP(l^\infty)$ is lower and $MP(l^1)$ is upper bound on the coverage error. In particular, the following procedure can be applied to compute overall coverage error, ϵ of D:

For elements of FS, X_i, denoting an efficient face $i \in \{1, \dots, |FS|\}$, $|FS|$ number of $MP(l^\infty)$ and $MP(l^1)$ are solved with $Z = X_i$ and D in each to compute ϵ_i. The overall ϵ of D is then calculated by $\epsilon = \max_{i=1} {\epsilon_i}^F$. In order to insert efficient face information to each formulation, all indices in the efficient face are set as equality in the formulation.

2.5.2) Uniformity Measure

The uniformity measure concept is a measure of spacing between Pareto points. Ideally, we desire a discrete representation of the Pareto set with equally spaced Pareto points; however, it not possible with the unsymmetrical nature of the Pareto front, even if it is convex and polyhedral. Here, we use the measure proposed by Sayin (2000), which is defined as the minimum distance, δ, between any two distinct points in the discrete representation of the efficient set:

$$\delta_D = \min_{y_1, y_2 \in D}\{ y_1 \neq y_2 | d(y_1, y_2)\}$$

where $\overline{X}_E$ denotes the set of efficient solutions found by the algorithm.

This measure may be misleading when considered on its own; that is, the number of points in the set should also be taken into account when evaluating a uniformity quality result. For example, a set of solutions that has more points may produce a worse (lower) uniformity result than a set that has only two points that are placed far away from each other. Hence, it should be considered with cardinality measure.

2.5.3) Cardinality Measure

The cardinality measure concept refers to the number of points in the representation. This number should be high enough to fully represent the solution set; at the same time, it should be low enough not to overwhelm the DM with choices.

The measure proposed by Wu and Azarm(2001), called the “number of distinct choices" is used. This methodology is based on the idea that Pareto solutions within a certain distance of each other are counted as a single point. In this study, the number of points in the solution set is used as measure of cardinality.

2.6) Experimental Results

2.6.1) Method of Sample Problem Generation

During the experiments, sample instances are generated following the same method as Karasakal and Koksalan (2009). In other words, randomly generated polytopes are used as sample problems, which are expressed in the form $\{z \in R^m, z = Cx, Ax \leq b, x \geq 0\}$, where A is a $p \times r$ matrix and $b \in R^p$. In the first set of experiments, a structure originally proposed by Steuer (1994) for random problem generation is used. Criterion space is used to generate these

instances. An identity matrix of C is used to simplify the search procedure for finding the nondominated faces. The elements of the constraint matrix A were randomly generated from a uniform distribution; the interval is denoted by $[LA, UA]$. To generate the right-hand side vector b first, $Me_p \in R^p$ is constructed as an interior point on the feasible region, where the parameter M is a nonnegative scalar and $e_p \in R^p$ is a vector of ones. Then, a nonnegative value $\alpha_i \in [L\alpha, U\alpha]$ to form point $(Me_p + \alpha_i a_i) \in R^p$ on the boundary level of the i^{th} constraint is selected randomly (where a_i is the i^{th} constraint's gradient). Finally, the i^{th} constraint is formulated as follows:

$$(a_i)^T x \leq (a_i)^T (Me_p + \alpha_i a_i)$$

We used the same values as Karasakal and Koksalan (2009) as parameters: $[L\alpha, U\alpha] = (0,100]$ (i.e., zero density is 0%); $[L\alpha, U\alpha] = [0,0.01]$; and $M = 5$. Ten randomly generated problem instances are generated for each parameter combination. The levels of the parameters m are $(2,3,4,5)$ and of parameter p are $(5, 10, 30)$.

The second part of the study consists of two problem size parameters. These are the number of objectives (m) and the number of constraints (p). In this set of experiments, the elements of matrix A, vector b, and matrix C are randomly generated from the discrete uniform distribution in the intervals $(1,20), (1000,2500)$ and $(-10,10)$, respectively (see Sayin, 2003). A 25% zero density was provided in matrix A. The problems are created in two different $p \times r$ combinations, 40×50 and 80×100. For each combination, three different sets of problems with different levels of parameter m $(2,4,6)$ were generated. For each problem set, 10 instances are generated in the MATLAB R2007b environment.

2.6.2) Computational Results

The proposed algorithm has been coded in ILOG OPL 6.3 with CPLEX 12.1, and runs have been performed on a laptop with a RAM of 4GB and a dual-core processor of 2.1GHz. The first set of instances are used to test the coverage and uniformity performance of the proposed algorithm with the benchmark study of Karasakal and Koksalan (2009). The following table shows the average number of representative points generated for all sizes:

Table 2: Average Number of representative points (reported results from Karasakal and Koksalan, 2009).

No. of criteria	Average Number of representative points (reported results from Karasakal and Koksalan, 2009)		
	No. of constraints		
	5	10	30
2	**20(21.1)**	**19.7(20.5)**	**18.4 (26.1)**
3	26.4(24.9)	**34.2(34.5)**	**41.4 (44.8)**
4	31.8(24.8)	49.2(40.6)	**58.4 (60.2)**
5	47(25.5)	49.5(44.5)	**64.0 (64.1)**

The values in parentheses are the average number of representative points generated by the benchmark study. Based on this table, it can be concluded that the average number of representative solutions generated for the proposed algorithm do not exceed the number of solution sets generated by the benchmark for the group of instances written in bold. Similar to the gap between the targeted number and the actual number of solution points observed in the benchmark algorithm, obtaining the exact number of aimed representative solutions is not always possible for the proposed algorithm. This is due to the equal partitioning of grids. Keeping the number of partitions the same for each axis is important in order to get reference points with well-covered and uniform distribution. The total number of reference points is presented as $R = (p * T^{p-1})$ in the above calculations. Hence, when setting the partition parameter, T,

deviations from the actual number of representative points we want to obtain might become inevitable in many cases (e.g., when $(p * T^{p-1})$ is equated to a certain value, it needs to be solved to find T which is supposed to be a positive integer).

As mentioned previously, coverage and uniformity levels are the two quality measures that require further calculation. All algorithms that are used to calculate these are coded in Microsoft Visual Studio 2010, and the linear program required to find the efficient faces is solved with a CPLEX 12.5 C++ library. Table 3 and 4 ummarize the quality measure results for all instances in which the corresponding size of the representative sets are presented in Table 2. From this point forward, Karasakal and Koksalan's (2009) algorithm will be referred to as Algorithm-1; Sayin's as Algorithm-2; and our proposed algorithm as Algorithm-3. The results regarding Algorithm-1and 2 are obtained from Karasakal and Koksalan (2009); the can refer to this article for more details.

Table 3: Average coverage errors obtained by approaches 1–3 in the first set of experiments.

No. of criteria	Average coverage errors								
	No. of constraints								
	5			10			30		
	Algo. 1	**Algo. 2**	**Algo.3**	**Algo. 1**	**Algo. 2**	**Algo.3**	**Algo. 1**	**Algo. 2**	**Algo.3**
2	0.0460	0.0490	***0.0529***	0.0500	0.0530	***0.0511***	0.0440	0.0530	***0.0501***
3	0.2130	0.2170	***0.2515***	0.2280	0.2160	***0.2180***	0.2230	0.1970	***0.1011***
4	0.3790	0.3740	***0.4516***	0.4190	0.3760	***0.3040***	0.4170	0.3490	***0.2140***
5	0.4790	0.4420	***0.7174***	0.4500	0.4010	***0.5536***	0.4800	0.4130	***0.4043***
Ave.	**0.2793**	**0.2705**	**0.3683**	**0.2868**	**0.2615**	**0.2817**	**0.2910**	**0.2530**	**0.1924**

Table 4: Average uniformity levels obtained by approaches 1–3 in the first set of experiments

No. of criteria	Average uniformity levels								
	No. of constraints								
	5			10			30		
	Algo. 1	**Algo. 2**	**Algo.3**	**Algo. 1**	**Algo. 2**	**Algo.3**	**Algo. 1**	**Algo. 2**	**Algo.3**
2	0.0560	0.0360	***0.0064***	0.0550	0.0380	***0.0119***	0.0470	0.0130	***0.0110***
3	0.1110	0.0150	***0.0170***	0.1020	0.0130	***0.0166***	0.0930	0.0070	***0.0110***
4	0.1460	0.0200	***0.0173***	0.1320	0.0060	***0.0156***	0.1250	0.0060	***0.0098***
5	0.3160	0.0140	***0.0130***	0.1670	0.0180	***0.0188***	0.1360	0.0060	***0.0081***
Ave.	**0.1573**	**0.0213**	**0.0134**	**0.1140**	**0.0188**	**0.0157**	**0.1003**	**0.0080**	**0.0100**

All the objectives in this experiment are scaled between zero and 1; hence a coverage error of 0.47 is interpreted as having a representative point for each nondominated point at most 47% of the range of the objectives. Uniformity is the smallest distance between any representative points in the representative set. Hence, the smaller coverage measure and larger uniformity measure are desirable in terms of quality perfective. Based on this statement, one can conclude that the proposed approach is compatible with previous approaches. Indeed, as the problem size grows, the proposed approach generates better results; i.e., the number of constraints and variables increase, hence the model coverage error becomes relatively smaller. However, the proposed approach does not perform well in terms of the uniformity measure since in most cases uniformity is under the average of benchmark Algorithm-1 and is compatible with Algorithm-2.

Table 5: Average CPU times required by approaches 1–3 in the first set of experiments

No. of criteria	Average CPU time								
	No. of constraints								
	5			10			30		
	Algo. 1	**Algo. 2**	**Algo. 3**	**Algo. 1**	**Algo. 2**	**Algo. 3**	**Algo. 1**	**Algo. 2**	**Algo. 3**
2	13.20	95.60	0.5057	15.40	97.70	***0.4615***	20.90	176.60	***1.5813***
3	15.10	135.20	0.1927	19.70	188.70	***0.6514***	36.30	282.10	***0.7351***
4	21.80	222.20	1.3916	29.10	267.00	***2.3882***	47.30	551.70	***2.4290***
5	25.40	221.00	1.1422	33.90	372.20	***1.0549***	52.40	960.80	***3.0114***
Ave.	**18.875**	**168.500**	**0.8081**	**24.5250**	**231.400**	**1.1390**	**39.2250**	**492.800**	**1.9392**

As the last important performance measure of the proposed approach, we have compared the time performance of the proposed approach with the benchmark algorithms. In order to do this, we used two different sets of data, the first of which is one that has been used to assess quality performance and that has a polyhedral structure. The results are presented in Table 5. The second set of data includes instances that are generated in a different way, as explained at the begging of the section. These do not necessarily have a polyhedral Pareto front structure and are larger. These results are presented in Table 6.

Table 6: CPU Time of obtaining one solution point for proposed approach and the Karasakal and Koksalan (2009) algorithm

No. of criteria	No. of constraints	No. of decision variables	Average No. of representative points		CPU time of proposed approach in sec. /solution point		CPU time of Algorithm-1 in sec. /solution point	
2	40	50	100	150	0.0165	0.0103	0.25	0.22
4	40	50	84	148.5	0.0413	0.0480	0.27	0.24
6	40	50	197.4	1274.8	0.0190	0.1135	0.30	0.27
2	80	100	100	150	0.0189	0.0200	0.31	0.27
4	80	100	80	145	0.0648	0.0749	0.33	0.29
6	80	100	198	1320.3	0.0229	0.1638	0.40	0.33

For the second set of experiments, only the results of Algorithm-1 are presented, since Algorithm-2 has already been deemed inefficient compared to Algorithm-1 (Karasakal and Koksalan, 2009). The time performance of the proposed approach can be observed in both of the tables 5 and 6 above. This is especially true in larger sizes that have a non-polyhedral structure. As such, the proposed approach is becoming significantly better than the benchmark algorithms, as predicted by the complexity analysis presented earlier.

We have done further analysis in order to assess how performance of quality measures changes for different partition values.

Table 7: Change of performance parameters with different values of partition parameters on the first set of experiments

	2 objective								
	T1(36)			T2(70)			T3(136)		
	uniformity	coverage	CPU Time	uniformity	coverage	CPU Time	uniformity	coverage	CPU Time
max	335594	0.2911	0.358	173368	0.1688	0.587	95333	0.0899	1.083
min	15557	0.0047	0.972	4280	0.0032	1.444	900	0.0047	1.493
St.dev.	108349	0.0884	0.218	64127	0.0494	0.272	39353	0.0297	0.154
average	129841	0.0406	0.751	58146	0.0382	0.935	20686	0.0369	1.334
	3 objective								
	T1(230)			T2(499)			T3(846)		
	uniformity	coverage2	CPU Time	uniformity	coverage2	CPU Time	uniformity	coverage2	CPU Time
max	4440000	0.4455	2.019	1820000	0.4387	4.181	536660	0.4586	7.324
min	44205	0.0172	2.791	15313	0.0000	6.072	5414	0.0220	13.136
St.dev.	1592416	0.1296	0.262	632195	0.1445	0.677	177474	0.1210	1.793
average	1040043	0.2119	2.405	395516	0.1876	5.272	90148	0.1669	9.255
	4 objective								
	T1(80)			T2(477)			T3(1032)		
	uniformity	coverage2	CPU Time	uniformity	coverage2	CPU Time	uniformity	coverage2	CPU Time
max	1278540	0.5655	2.227	358862	0.4848	12.940	8465	0.4805	24.698
min	33477	0.1304	3.128	10812	0.1801	19.822	800	0.1268	46.381
St.dev.	382532	0.1338	0.340	108499	0.1181	2.187	2998	0.1218	6.560
average	209034	0.2965	2.691	56924	0.2855	15.506	2851	0.2704	31.088
	5 objective								
	T1(50)			T2(402)			T3(941)		
	uniformity	coverage2	CPU Time	uniformity	coverage2	CPU Time	uniformity	coverage2	CPU Time
max	1205750	1.0000	1.015	399010	0.5627	8.719	4240	0.4113	21.569
min	13024	0.3366	2.098	4038	0.1606	22.977	371	0.3174	33.789
St.dev.	563040	0.2153	0.341	148111	0.1476	5.147	1330	0.0440	3.944
average	428746	0.5388	1.556	83202	0.4091	12.389	961	0.3512	25.100

Although uniformity measure seems not better than the Karasakal and Koksalan(2009)'s algorithm, and coverage error does not seem to beat Sayin(2003)'s algorithm in the first set of experiments, there is an important factor that need to be paid attention, that is the cardinality of the sample sets which can be observed in Table 2. At this point, it should be noted that both uniformity and the coverage error are both sensitive to the number of points in the set. That is, perfect spacing between points changes as the number of represented points increases; similarly the maximum distance that a representative point can be found for any Pareto point is also affected by the number of representative points generated. In order to show the sensitivity of uniformity with a true measure which considers both the spacing and the number of points in the representative set, we have resorted to another measure. Wavelength analysis based uniformity measure has been proposed by Meng et al. (2005) which can be summarized as follows:

$$SP = \sqrt{\frac{\sum_{h=1}^{N}(1 - F(d_h, \bar{d}))^2}{N - 1}}$$

where $F(a,b) = \begin{cases} \frac{a}{b} \text{ if } a > b \\ \frac{b}{a} \text{ else} \end{cases}$, and d_h is the minimum distance between point h and its closest neighbor, and $\bar{d}$ id the mean of these distances. According to this calculation, smaller SP values indicate a better uniformity.

Table 7 shows the change of these two quality measures with changing number of representative points, i.e. increasing partition parameter. As number of partitions increases, CPU time requirement increases accordingly, however the improvement in coverage and uniformity measures seems much faster than this change. This means the advantage in run time can be used in order to close the gap between our algorithm and Karasakal and Koksalan's algorithm in

terms of uniformity ; and the gap between our algorithm and Sayin's algorithm in terms of coverage.

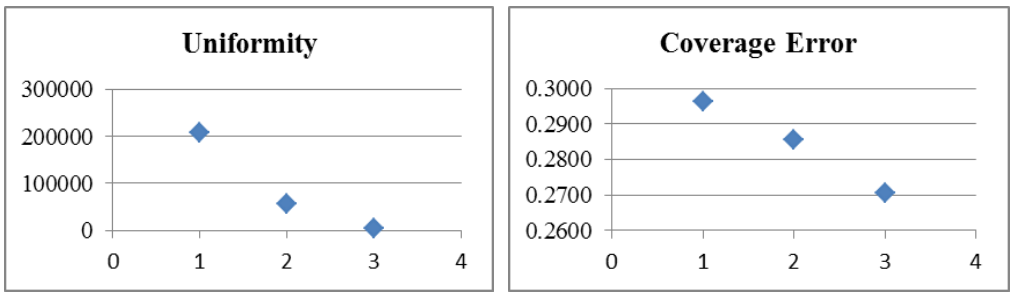


Figure 10: Change of average of uniformity and coverage measure per three different partition values in increasing order for the set of instances with 4 objectives and 10 constraints

2.7) Conclusion and Future Work

In this chapter, an algorithm for MOLP was proposed, which can be used to generate a representative set of Pareto solutions in a relatively fast way. The proposed approach is an exact method that guarantees the generation of Pareto optimal solutions by relying on an achievement scalarization function. From this perspective, the proposed algorithm is distinguishable from metaheuristics and algorithms like the normal constraints method. The proposed approach is comparable to the algorithms proposed by Karasakal and Koksalan (2009), which contains two other exact approaches that serve the same purpose. In this study, the authors use a modified version of an algorithm that was initially proposed by Sayin (2003) to benchmark their algorithm. All of the computational results of the proposed approach are compared with the averages of both algorithms on instances generated in the same way that authors of this study followed. Based on these results, the proposed approach falls behind Karasakal and Koksalan's

(2009) algorithm in terms of uniformity, but is compatible with Sayin's (2000) approach. On the other hand, the proposed algorithm outperforms both approaches in terms of time performance. Considering the fact that tests are performed in different environments, computers and not using exactly the same instances, this observation is supported by the previously described complexity analysis. Since the latter algorithm is already proven to be better than Sayin's approach in terms of time performance, the complexity analysis for this algorithm has not been presented. In addition, the proposed approach generates compatible results for the coverage error, and it is starting to perform better than benchmark algorithms for problems with bigger sizes.

Future studies may consider trying the algorithm on MOIP instances since it has significant potential to perform well on non-convex Pareto fronts. Furthermore, when determining the reference point, improvements should be made that take advantage of integer feasible space, such as rounding the reference points systematically up and down in a way to get an integer approximation of the Pareto front with 0^{th} order of approximation (i.e., in a way that does not depend on the achievement scalarization function to get Pareto optimal points).

CHAPTER III: MULTI-OBJECTIVE BRANCH AND BOUND APPROACH FOR MOIP

3.1) Introduction

A MOIP problem can be formulated as follows:

$$\text{P1: (max)} \left\{ \left(z1(x), \dots, zp(x)\right) = Cx : x \in X \right\} (1)$$

where $p \geq 2$ and ($C \in R^{p \times n}$; X denotes the set of feasible set of solutions and is defined by

$$X = \{x \in Z^n : Ax \leq b, x \geq 0, x \in Z\} (2)$$

where $A \in R^{m \times n}$, $b \in R^m$ and all decision variables are required to be integers. If some of the x's are continuous and some are integer, then the problem becomes a multi-objective mixed integer programming (MOMIP) problem.

The difference between MOLP and MOMIP problems are stated in terms of the "topologically connectedness" concept. A set is called topologically connected if there are no non-empty open sets, S_1 and S_2, such that $S \subset S_1 \cup S_2$ and $S_1 \cap S_2 = \emptyset$. For MOLP, the efficient set X_E and the nondominated set Y_N are topologically connected and Y_N is composed of nondominated faces of dimensions 0 to $p-1$. However, neither X_E nor Y_N are topologically connected for MO(M)IP in general.

Multi-objective combinatorial optimization (MOCO) problems are a special class within MOIP problems such that the feasible set of a combinatorial problem is defined as a subset $X \subseteq 2^A$ of the power set of a finite set $A = \{a_1, \dots, a_n\}$. In terms of the feasible set, this definition comprises multi-objective versions of the shortest path, a minimum spanning tree, an assignment, a knapsack, a travelling salesperson, or set covering problems, to name a few.

All MOIP problems have a discrete solution space that is non-convex by nature. As it is mentioned in the definition of "supported solutions," which can be identified as a solution of a

weighted sum of objectives, there are unsupported solutions in MOIP problems that cannot be generated by weighted sum approaches. Therefore, identifying a closed form of efficient faces is not possible in this context. Even though approaches that aim to generate a representative set for MOIP problems are used in real-life problems in the decision-making process, obtaining the whole Pareto set also becomes necessary for verification purposes. Based on this observation, both representative and exhaustive algorithms might be of interest for the DM.

Exact algorithms for MOIP problems have become a popular topic of optimization that is in line with the increasing practical usage of this type of algorithm in many real-life problems and decision support systems. In the next sections, the main studies that are proposed to solve MOIP problems are explained in detail which are important in terms of the proposed algorithm. However, to the best of our knowledge, most of these efficient algorithms explore the Pareto surface starting from one corner of it and approaching the opposite end, even if they intend to take some smart actions that speed up this traversal. Hence, when the DM stops the algorithm at a certain point before termination, he or she ends up with a partial Pareto optimal set that does not give much information about a certain part of the Pareto surface. In addition, there are also meta-heuristic approaches that are superior in terms of time performance but that do not guarantee any type of Pareto optimality while acting as an approximation tool. Hence, there is a need for an exact method that proceeds more diversely while maintaining the Pareto optimality notion. The algorithm proposed in this chapter aims to fill this gap.

The branch and bound (B&B) method is one of the main approaches that is often resorted to when optimizing problems with integer variables. Accordingly, it has received significant attention in the multi- objective optimization field. All of this research falls under the umbrella of multi-objective branch and bound (MOB&B). Since the concept of B&B mostly relies on the

existence of integer valued variables, the explicit word “integer” has been dropped from the general naming convention of the approach. In addition to its natural convenience for integer variables in single objective, mixed integer linear programming, it has some additional advantages in multi-objective optimization context. The B&B approach is preferable because it finds Pareto efficient solutions with a finite number of iterations.

This thesis emphasizes the distinction between the two B&B perfectives that are used in the literature: Branching on decision variables or branching on multiple objectives. The second approach seems to be more meaningful, especially when the number of objectives is greater than the number of variables. This decision about which perspective to choose can be made by evaluating the tradeoff between solving linear (relaxed) problems while handling all decision variables, and solving integer problems while performing a reasonable amount of branching on the multiple objectives. Potentially, the first approach requires more iterations than the second one, since the number of decision variables is generally much greater than the number of objectives, which means more branching is needed to fully characterize the Pareto front. Furthermore, there might be multiple solutions that lead to the same Pareto point, and thus some of the computational effort spent does not lead to new Pareto solutions. None of the existing approaches can identify these cases, which in turn might lead to numerous branching in order to repeatedly identify the same solution.

Another classification of the multi-objective optimization methods is based on the difference between the usage and design of algorithms and closely relates to the role of the DM. Whenever the DM guides the search process, the algorithm falls into the “interactive” category. In contrast, non-interactive methods allow the DM to be involved at the end of the solution process through a “posterior selection” procedure. In this category, the DM can evaluate the

entire Pareto front, which improves the DM's confidence in the results since he or she has a broader characterization of the tradeoff space. From this perspective, the proposed algorithm has an exact nature if it is allowed to run until termination. On the other hand, if it is used as an approximation algorithm, the DM's search can be interactively restricted to certain regions, and hence can be run as an interactive algorithm.

In the next section, we present a general overview of the literature regarding exact methods for MOIP. Next we explain, in more detail, some of the earlier work that led to the proposed algorithm. We then present the detailed steps of the proposed approach. Finally, we discuss the results of computational experiments performed to assess the main attributes of the algorithm.

3.2) Literature Review

Early work on exact solution methods for multi-objective optimization mostly focuses on finding supported nondominated points. An excellent review of the exact and approximation methods developed specifically for the MOCO problems can be found in Ehrgott and Gandibleux (2000). Some authors in these early studies separate the generation of the nondominated points into two phases. In the first phase, all supported nondominated points are generated using the weighted sum scalarization. In the second phase, all unsupported nondominated points are obtained by employing problem-specific techniques. This approach has been applied to several biobjective combinatorial problems. Visée et al. (1998) proposed a two-phase method and B&B procedure for the biobjective knapsack problem. Ramos et al. (1998) and Steiner and Radzik (2008) developed a two-phase method to generate all nondominated trees for the biobjective spanning tree problem. Przybylski et al. (2010) worked on the two-phase method for MOIP problems and experimented with three-objective assignment problems. Ozlen

and Azizoglu (2009) developed an algorithm to generate all nondominated points for MOIP problems based on the epsilon constraint method. They do not conduct computational experiments but they demonstrate their algorithm on a three-objective assignment problem. Laumanns et al. (2006) also developed an algorithm to generate nondominated points based on the epsilon constraint method. Sylva and Crema (2004) developed an exact algorithm to generate all nondominated points for MIPs. Lastly, Lokman and Koksalan (2012) developed an algorithm that is superior to those proposed in both Sylva and Crema (2004) and Ozlen and Azizoglu (2009). The study of Lokman and Koksalan will be explained in detail since it is the inspiration for the objective function used in the proposed algorithm.

Prior work using B&B for MOIP problems dates back as early as 1983. Kiziltan and Yucaoglu (1983) proposed an algorithm for multi-objective zero-one linear programming problems. Fifteen years later, one of the major studies that aimed to design a B&B algorithm for multi-objective mixed integer problems was published, which branched on decision variables by Mavrotas and Diakolulaki (1998). This is claimed to be the first attempt to develop a general purpose vector maximization algorithm applicable to all kinds of Mixed 0-1 MOLP problems of small or medium size (i.e., a few hundreds of variables). Mavrotas and Diakoulaki revised this work in (2005) with some improvements (2005). However, this study was corrected by Vincent et al. (2013) who claimed that Mavrotas and Diakolulaki (2005) do not allow for a complete description of the nondominated set YN of an MOMIP. Moreover, a solution set may still contain dominated points. In the same study, Vincent et al. (2013) suggested some corrections to the filtering rule by pointing to the fact that one might need some interior points (not extreme nondominated) in order to come up with correct domination results; edges are generated as a result of MOLP solves performed at each node. The algorithm proposed by the authors is for bi-

objective mixed integer optimization problems. Furthermore, the bound sets discussed in detail in this study constitute a fundamental concept introduced to B&B for MOP. Ehrgott and Gandibleux (2007) presented a study that aimed to identify good upper and lower bound sets to be used in a MOB&B context, where several lower bound candidates are presented. These bounds differ from eachother in terms of performance and computation effort spent to identify the set, while the ideal point of a node at a B&B tree is the most common lower bound used in the literature. On the other hand, the upper bound is defined by any set of feasible solutions such that no two points dominates the other; similarly, several candidates for an upper bound set are presented. Abbas and Chergui (2012) proposed adding cuts during a B&B traversal for MOMIP problems while branching on decision variables. So far, no algorithm that branches on objective values has been mentioned that can be used for any type of MOIP problem (i.e., not just for MOCO). This is because, to the best of our knowledge, Marcotte and Soland (1986) presented the only study in this field. This algorithm will be covered in detail in the next section following a detailed explanation of the main exact methods in the literature.

3.2.1) Exact Algorithms for MOIP

A method for finding the set of nondominated vectors for multi objective integer linear programs by Sylva and Crema (2004)

Sylva and Crema (2004) developed a general algorithm that enumerates a full set of solutions, requiring a solution of the following model:

(P_λ^n)

$$Max \sum_{j=1}^{p} \lambda_j z_j (x)$$

$$s.t.$$

$$z_j(x) \geq (z_{tj} + 1)y_{tj} - M(1 - y_{tj}) \quad \forall j \; \forall t$$

$$\sum_{j=1}^{p} y_{tj} \geq 1 \quad \forall t$$

$$y_{tj} \in \{0,1\} \; j = 1, \dots, p \quad t = 1, \dots, n$$

$$x \in X$$

where $z^t = (z_{t1}, z_{t2}, \dots, z_{tp})$ denotes the t[th] nondominated point in a set of solutions; M is a sufficiently large number to unconstraint $z_j(x)$. The qwquation $\sum_{j=1}^{p} y_{tj} \geq 1$ ensures at least one of the objective values to be improved. Hence, a new nondominated solution is generated. Since the algorithm keeps adding new binaries and constraints every time a new solution is generated, it grows as it evolves until infeasibility, or the stopping condition.

General Approach Generating All Non-dominated solutions by Ozlen and Azizoglu (2009)

This algorithm is an exact algorithm proposed by Ozlen and Azizoglu (2009). It is a modified version of the classical ε-constraint method, which searches within narrower efficiency ranges and jumps between non-dominated solutions rather than taking incremental steps. One important difference from the original ε -constraint method is the structure of the objective function used throughout the algorithm. An important requirement of this algorithm is the integrality of the objective function coefficients. If this is not the case, then the coefficients can always be converted to integers by proper scaling. The general formulation of the model as follows:

$$Min\ z_1(x) + \frac{1}{(z_2^{GUB} - z_2^{GLB} + 1)} z_2(x) + \cdots + \frac{1}{(z_2^{GUB} - z_2^{GLB} + 1)(z_3^{GUB} - z_3^{GLB} + 1) \ldots (z_p^{GUB} - z_p^{GLB} + 1)} z_p(x)$$

$$s.t. \quad z_2(x) \le I_2$$

$$z_3(x) \le I_3$$

$$:$$

$$z_p(x) \le I_p$$

$$x \in X$$

where z_{ij}^{GUB} and z_j^{GLB} are the upper bound and lower bound of the j^{th} objective, respectively; and X is the feasible set of the original problem.

After generating a solution point by solving the model above, the right-hand sides of the constraints set are updated according to the algorithm presented in the study. The main function of the algorithm is to find the best solutions hierarchically for each objective while ensuring all levels of objectives that are fixed by some additional constraints are searched. So, the right-hand side values of the constraint, which stands for the objective that is being optimized in the first place, is updated and the model is changed, so that a new solution has a better value of this objective under the same requirements on all other objectives. The main advantages of using this algorithm are as follows:

- This algorithm generates the whole Pareto surface without performing full enumeration.
- This algorithm is a good method for determining the true Pareto and assessing the global Pareto proportion for approximation algorithms.

The following are the algorithm's disadvantages:

- This algorithm generates the whole Pareto surface by starting from one corner of the Pareto region and then iteratively finding a neighboring non-dominated solution. Hence, this algorithm

produces the Pareto front by growing it locally. Therefore, it is not implementable in parallel computing and requires considerable CPU time as the problem instance grows.

- An excessive amount of duplicate solutions are produced by the algorithm, which also increases the CPU time. This problem is attempted to be overcome in a later version of the study by documenting the same regions that should not be searched again.

The complexity of the algorithm grows exponentially based on the size of the Pareto space. Furthermore, as the number of objectives increases, the number of iterations increases exponentially. While it is difficult to make an average case analysis; the worst-case complexity analysis suggests that the complexity can be expressed as $O(N^N)$, where N is the number of points in a Pareto set.

Finding all non-dominated points of multi-objective integer programs by Lokman and Koksalan (2012)

Lokman and Koksalan (2012) propose two algorithms that aim to generate all Pareto sets of MOIP problems.

Algorithm I:

This first algorithm improves upon Sylva and Crema (2004) by reducing the number of constraints and binaries through the change of objective function. One of the objectives is selected arbitrarily; let us denote that *m*, and it stays as the main objective that is being maximized by the new special structure of the objective. The basic form of the new objective function is as follows:

$$Max\ z_m(x) + \varepsilon \sum_{j \neq m} z_j(x)$$

where ε is small positive constant that ensures that resultant solution is nondominated. This objective function generates the best non dominated solution in terms of the m^{th} objective. Considering this, the improvement constraints used in Sylva and Crema (2004) has been reduced by one. Hence, rest of the resultant model needs to be solved until infeasibility is as follows:

$$Max\ z_m(x) + \varepsilon \sum_{j \neq m} z_j(x)$$

$$s.t.$$

$$z_j(x) \geq (z_{tj} + 1) y_{tj} - M(1 - y_{tj}) \qquad \forall j \neq m, \forall t$$

$$\sum_{y \neq m} y_{tj} = 1 \qquad \forall t$$

$$y_{tj} \in \{0,1\} \qquad t = 1, \dots, n \quad j = 1, \dots, p \quad j \neq m$$

$$x \in X$$

where t stands for the indices of solution points in the solution set, denoted by Sn, collected so far; m is the active objective. It has been proved that if this model ends up with infeasibility, the resultant Sn will contain all of the nondominated points of the Pareto surface. This algorithm requires np additional constraints and $n(p-1)$ binary variables in total.

Algorithm 2

The authors presented a second algorithm that improves upon the previous algorithm by separating the main model into submodels. It has been observed that, at most, one constraint is sufficient to define the region that contains the nondominated points relative to the available points. Based on this, it is concluded that the right-hand side values of each p-1 objective for each submodel can be determined more effectively.

This algorithm basically solves a different model for each nondominated point collected so far in set Sn. After setting the first point's 1st objective value as the right-hand side value of z_1 in the below model, i.e., $b_1^{k,n}$, the points in Sn are searched for the point that has the value of 1st objective is greater then $b_1^{k,n}$;and maximum of values of 2nd objective is set for the bound value of 2nd objective, i.e., $b_2^{k,n}$.This continues until the pth objective, in a manner in which each time a

point is selected to set the next objective's boundary by only considering the points in Sn, which are not dominated by the previously set boundary values:

$$\text{Max } z_p(x) + \varepsilon \sum_{j=1}^{p-1} z_j(x)$$

$$\text{s.t.} \quad z_1 > b_1^{k,n}$$

$$z_2 > b_2^{k,n}$$

$$:$$

$$z_{p-1} > z_{p-1}^*$$

$$x \in X$$

Two important improvements have been offered by the authors to prevent extra model solving. The first is about setting the bounds. It basically attempts to improve the boundary values. If an optimal solution $z^{(P^{b_1})}$ is generated by using a certain set of boundary values, $b_1 = \{b_1^1, \dots, b_{p-1}^1\}$, then any boundaries that will be used between this point and b_1 will generate the same solution.

The second suggestion is to use the information obtained from the submodel solves that result in infeasibility. The authors proposed to store boundary values in a list that generates infeasibility rather than using some boundary values vector that dominates one of the vectors in this list as boundary value; since outcome will be nothing but infeasibility.

Although the worst-case complexity of the algorithm is presented as $O(N^{p-1})$, with the rules mentioned above applied, the number of models solved is observed to be significantly less than the worst case bound on a small sample case with three objectives.

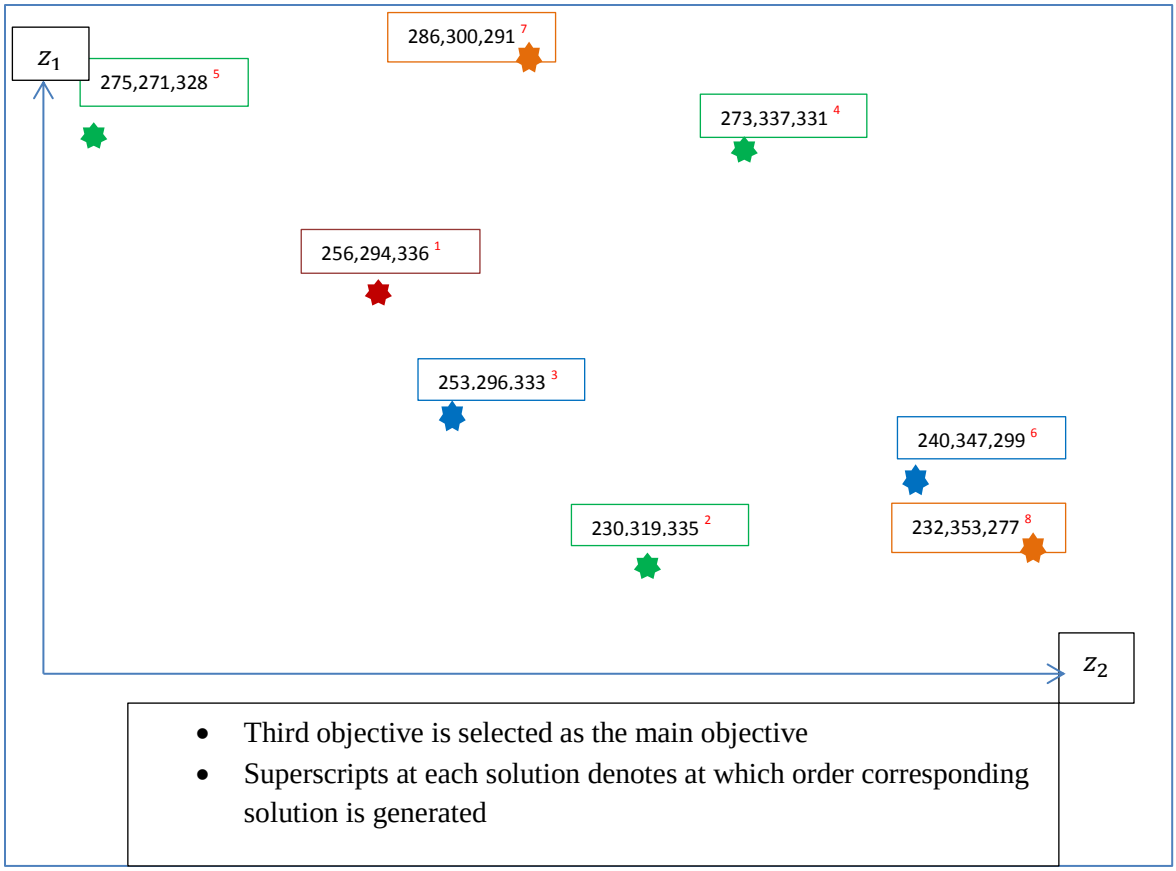


Figure 11: Projection of solution set onto z1-z2 plane for a three-objective MOIP problem

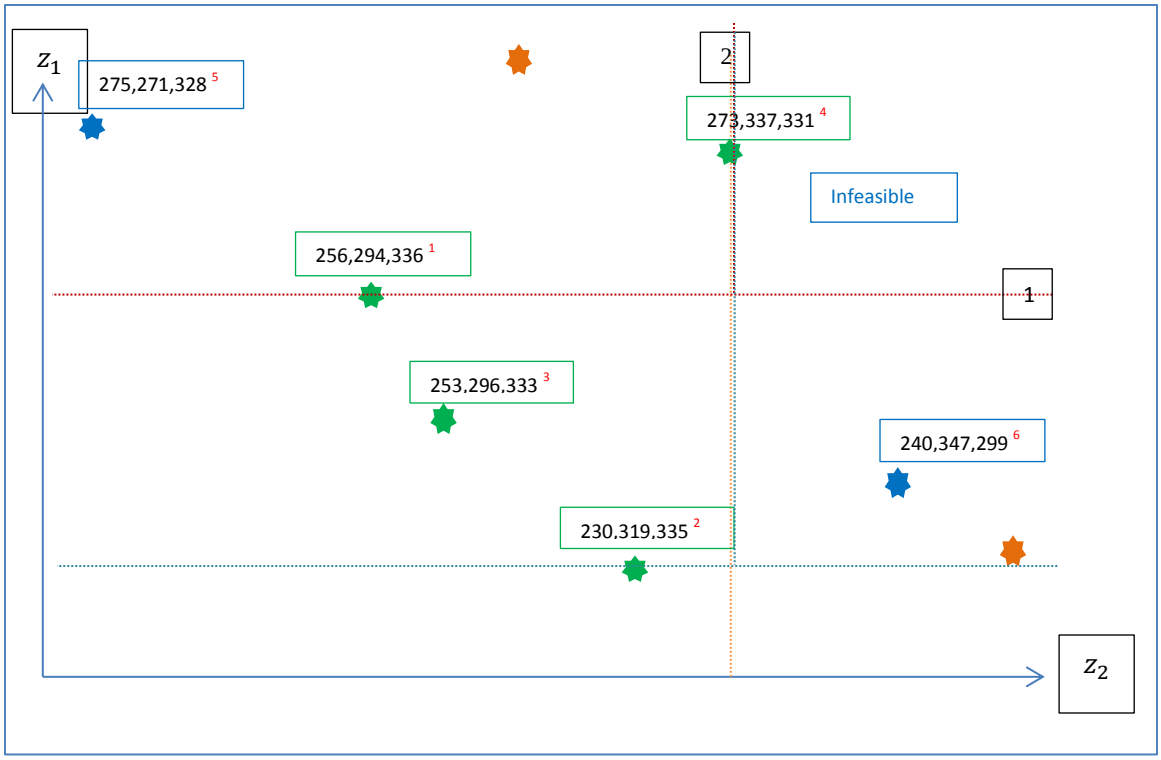


Figure 12: Search space after obtaining first four solution points, according to Algorithm-2

3.2.2) B&B Tree Branching on objective values

3.2.2.1) MOB&B on Objective Space in Literature

Marcotte and Soland (1986) presented the first B&B algorithm, which is designed for the objective space. They designed this algorithm as an interactive approach, which evolves based on a DM's preferences and does not rely on the Pareto optimality concept. The following generalization stated by Marcotte and Soland (1986) is worth mentioning here:

> *A branch-and-bound algorithm is defined as the collection of rules which specify following: (a) how to determine whether a given subset Yi can or cannot contain an optimal solution, and how to recognize an optimal solution; (b) how to carry out branching at each intermediate node; (c) how to calculate the upper bounds and (d) how to choose an intermediate node for branching. Such an algorithm terminates because either an optimal solution has been identified or no intermediate node can be chosen for the next branching.*

Their algorithm applies both to the case in which the feasible set is convex (e.g., MOLP) and to the case in which the feasible set is discrete and non-convex. Furthermore, the algorithm is designed as an interactive algorithm. However, it only requires the DM to make comparisons between two or several points in the objective space (points that are not always feasible). It is claimed that the hypotheses required regarding the DM's preferences are minimal. In other classical B&B approaches, a solution found at a node is rejected only if the ideal value of current node is dominated by an efficient solution. In the algorithm presented in this paper, a node may be rejected because its ideal (i.e., best values that each objective can attain at a node) is not preferred by the DM to an efficient solution already found by the algorithm. The algorithm proceeds with the calculation of ideal values of the newly created nodes. The incumbent solution is defined to be that efficient point, among all those found thus far, that is preferred by the DM. The node whose ideal is preferred (among the ideals of the newly created nodes) is inserted into the *master list*, whereas the other nodes (which have been arranged in order) form the *partial list*, Plist(k), corresponding to the separation of node N_k. They will only be inserted into the master list later, when the preferred node gets to the "top" of the master list. At that time, the DM will be asked to insert into the master list the successor, in its partial list, of the node that is about to be separated. This insertion is only made, however, after verification that the ideal of the node in question is preferred to the incumbent solution. Although the master list does not contain all the intermediate nodes, it is totally ordered by preference and its first element is always the intermediate node preferred (i.e., whose ideal is preferred) by the DM**.**

The algorithm can terminate in any of the three following ways: (a) the ideal of the first node of the master list is feasible, (b) the ideal of the first node of the master list is not preferred to the incumbent solution, or (c) the master list is empty. In case (a), it is the ideal that is the

optimal solution since it is no less preferable than the ideals of all the intermediate nodes. In cases (b) and (c), the incumbent solution is optimal because it is, by transitivity, no less preferable than the ideals of all the intermediate nodes. Cases (a) and (b) correspond to the two ways to fathom a node in a usual B&B algorithm—by feasible solution and by bound.

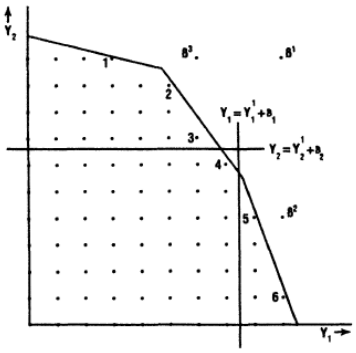
Let Eff(Y) denote the Pareto set of problem defined as "Maximize y subject to $y \in F$," where y is a vector in R^p. N_k the node examined, by Y_k the proper subset of Y corresponding to N_k, and by $\beta^k = (\beta_1^k, \beta_2^k, \dots, \beta_p^k)$ the ideal of Y_k. The first step of branching consists of finding a point belonging to Eff (Yk) that is also guaranteed to be an efficient point for the original problem.

To find an efficient point at node N, the following problem P is solved:

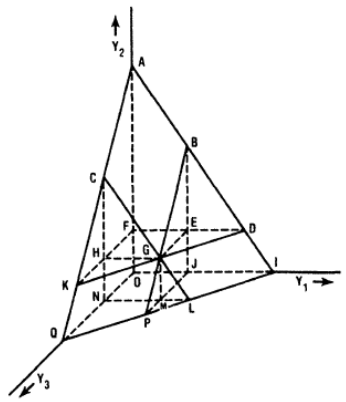
$$Maximize \sum_{j=1}^{p} c_j^l y_j \qquad subject\ to\ \ y \in Y$$

where c_j^l are all positive. It is claimed that, in the B&B tree, each node represents a nonempty subset containing at least one efficient point.

Let F be the collection consisting of the set $\{y^i\}$ and the sets $Y_k \cap \{y | y_k > y_j^k\}$ for $k \in I_j$, where $I_j = \{k | 1 \le k \le p, \beta_k^j - y_k^j > 0\}$; F, the separation at node N_k, is generally neither a partition of Y_k nor a cover of Y_k. First, the set $\bigcup_{s \in F} S$ is, in general, a proper subset of Y_k. Second, if p > 2, the intersection of the sets belonging to Y is not empty in general (see Figure 12 for pictures of the sets involved when p = 2 and p = 3, respectively). However, it is claimed that the first of these two characteristics does not affect the validity of the separation since Y is a cover of Eff(Y_k) and the solution sought necessarily belongs to Eff(Y_k) by the authors.



(a) $Y = Y_1$ consists of the 86 discrete points shown.
(b) Eff(Y) consists of the six points labeled 1, 2, . . . , 6.
(c) The efficient point y^1 is point 4.
(d) $Y_2 = Y_1 \cap \{y \mid y_1 > y_1^1\}$ and consists of 7 points. It contains efficient points 5 and 6. $Y_3 = Y_1 \cap \{y \mid y_2 > y_2^1\}$ and consists of 23 points. It contains efficient points 1, 2 and 3.
(e) β^1, β^2 and β^3 are shown.



(a) $Y = Y_1$ is the tetrahedron $AIOQ$.
(b) Eff(Y_1) is the triangle AIQ.
(c) y^1 is the point G.
(d) $\bar{Y}_2$ is the tetrahedron $BIJP$. $\bar{Y}_3$ is the tetrahedron $ADFK$. $\bar{Y}_4$ is the tetrahedron $CLNQ$.
(e) $\bar{Y}_2 \cap \bar{Y}_3$ is the tetrahedron $BDEG$. $\bar{Y}_2 \cap \bar{Y}_4$ is the tetrahedron $GLMP$. $\bar{Y}_3 \cap \bar{Y}_4$ is the tetrahedron $CGHK$.
(f) $\bar{Y}_2 \cap \bar{Y}_3 \cap \bar{Y}_4$ is the singleton $\{G\}$.

Figure 13: Two- and three-dimensional examples for the interactive B&B algorithm of Marcotte and Soland (1986)

The proposed algorithm is supposed to be used for MOLP problems as well. Thus, they have defined a small value, δ_k, which can help create discrete regions in which DM is indifferent between the solutions that have less difference than δ_k.

There are two important observations about the proposed algorithm that may prevent exploration of the entire Pareto front of a MOIP problem: The structure of the objective function presented above is the weighted sum of objectives, which is simply the convex combination of the points on the surface. Unfortunately, it is known that MOIP problems have a non-convex Pareto structure and weighted sum can generate only supported points on the Pareto front, as previously indicated. This means the proposed approach can only generate supported points to the DM. If it is allowed to run until the end without any intervention by the DM, there is no guarantee that the whole Pareto front will be obtained. In addition, the authors indicate that when

a number of objectives is greater than or equal to three, sibling nodes have intersecting efficient frontiers. Hence, it is possible to determine the same solutions that in turn cause inefficiencies in terms of running time.

3.3) Proposed Approach

The following algorithm is proposed for multi-objective problems with integer variables (MOIP). This is the necessary condition in order to obtain a discrete Pareto surface that consists of a finite number of solution points, as opposed to the case of MOIP's mixed integer (MOMIP) or linear (MOLP) counterparts.

The general idea behind the proposed approach is similar to the algorithm of Marcotte and Soland (1986) in the sense that branching is performed on solution points in the objective space. However, Marcotte and Soland (1986) uses a weighted sum approach to generate a new solution at each node, and this can exclude the non-supported parts of the Pareto surface. In addition, their algorithm includes searches on overlapping regions for sibling nodes, which might increase the running time of the algorithm. Our approach aims to apply B&B in objective space while avoiding these drawbacks. To achieve that, branching is done in a manner in which sibling nodes create partitions of the efficient sets of the parent node without overlapping regions. Also, in order for the algorithm to find both supported and non-supported points, weighted sum structure is avoided. Instead, an objective function that focuses on optimizing one of the objectives is chosen as the main structure, which is similar to that used in Lokman and Koksalan (2012). As a result, the proposed algorithm can identify dominated and non-dominated solutions during the solution process. In order to eliminate the identification of dominated solutions, we developed fathoming rules with the help of some dominance relationships developed among node types. The proof for the convergence of the algorithm, or finding all non-dominated

solutions, is also supplied. Suggested fathoming rules differ based on the memory requirements and are discussed in more detail in the fathoming schemes section.

Assume we have the following general form of MOIP problem, *P*:

$$Maximize\ z_k\ (x) \quad for\ \forall k \in \{1, \dots, p\}$$

$$subject\ to \quad x \in X$$

$$x\ integer$$

where $z_k(x) = \sum_{l=1}^{n} c_l^k x_l$ and c_l^k are the positive coefficients of the decision variable l for each objective k. As previously stated, the Pareto surface of the MOIP problems is a closed non-convex set of a finite number of solution points.

The initial decision that needs to be made at the start of the algorithm is to choose one of the objectives as the main objective, which will be optimized throughout the search tree. This decision might affect the order of solutions obtained throughout the traversal, and accordingly, this decision might impact total running time. However, none of these effects can be foreseen, so this selection remains somewhat arbitrary at this point. After making this decision, the main objective and all other objectives are combined into one single expression that is used as the objective function throughout the algorithm. This structure is the same as the one used by the second algorithm of Lokman and Koksalan (2012). This structure is primarily used to generate the best solution in terms of the main objective while eliminating weakly dominated solutions on the valid feasible region. With the usage of this objective, a single objective MIP is solved at each node, denoted by a.

Let p denote the number of objectives, again and assume the p^{th} objective is the effective objective. Then, the following MIP problem, $P_{1,}$ is solved:

$$\text{P1: Max } z_p(x) + \varepsilon \sum_{j=1}^{p-1} z_j(x)$$

$$\text{s.t. } x \in X$$

where ε is a sufficiently small positive constant that prevents obtaining weakly nondominated but dominated solutions. Let the optimal solution obtained as the result of above model at node *i*, N_i, be represented by $z^{i*} = z_1^{i*}, z_2^{i*}, \dots, z_p^{i*}$; and let Y denote the set of nondominated solutions on $x \in X$. Based on the B&B idea, N_i is the predecessor of some child nodes. We note that, unlike the single objective B&B, the number of children of a node is more than two nodes.

A sample Pareto space with three objectives is represented in Figure 13. In this example, the optimal solution of the current node, z^{i*}, can be used to divide the whole area of N_i into eight sub-regions, which can be expressed as 2^p in general. In this expression, "2" comes from the number of partitions obtained by considering the '<' and '>' sides of each axis, with z^{i*} being the origin of whole partitions. Since there are "p" many axes to consider, the total combination adds up to 2^p. Hence, each sub-region becomes a branch of a node on the B&B tree.

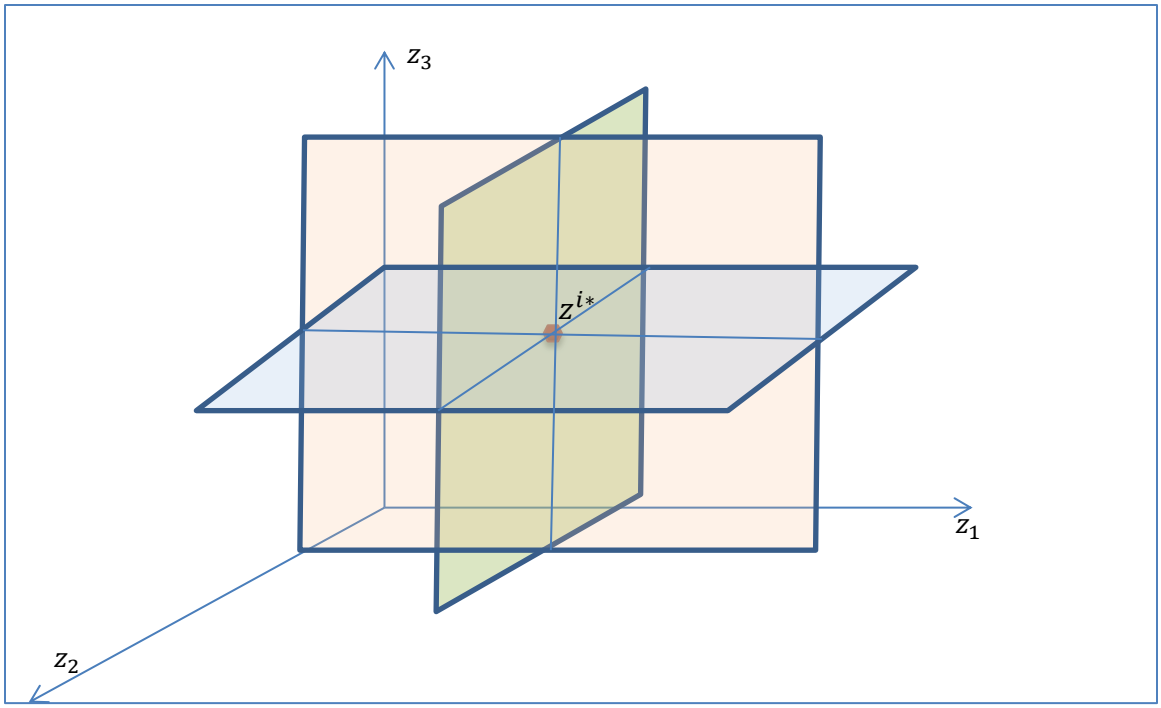


Figure 14: Partition of objective space based on a single solution point in three objectives

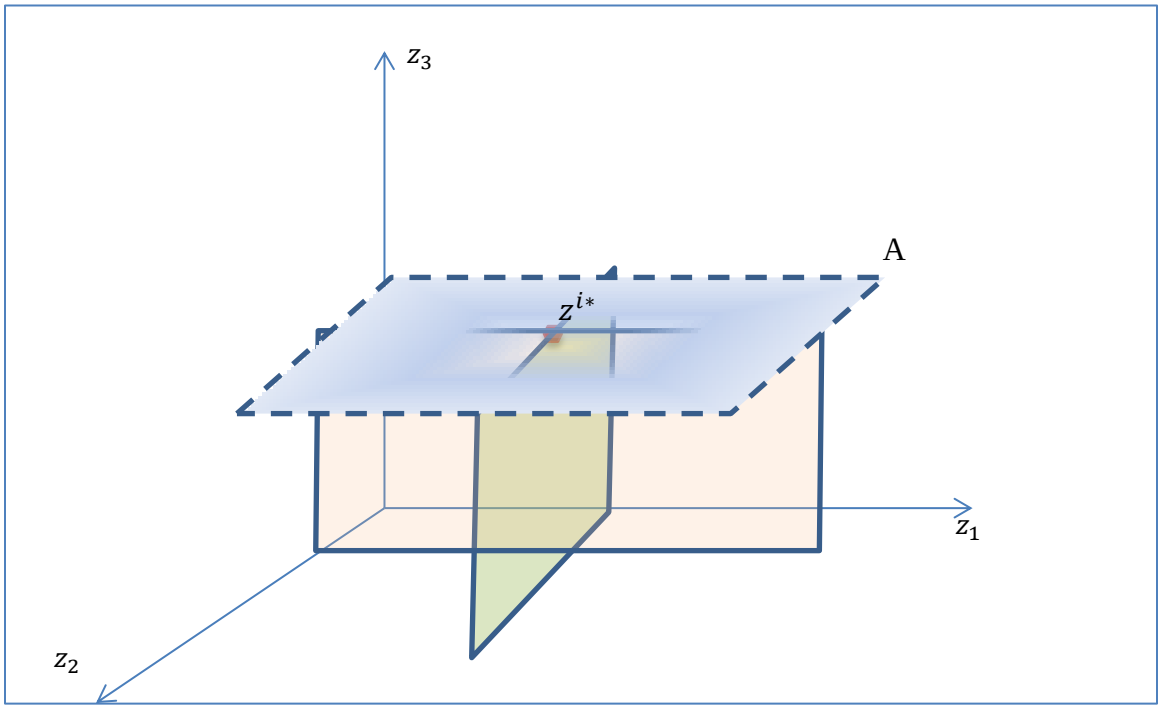


Figure 15: Partitions that contain the Pareto solutions based on their 3rd objective value

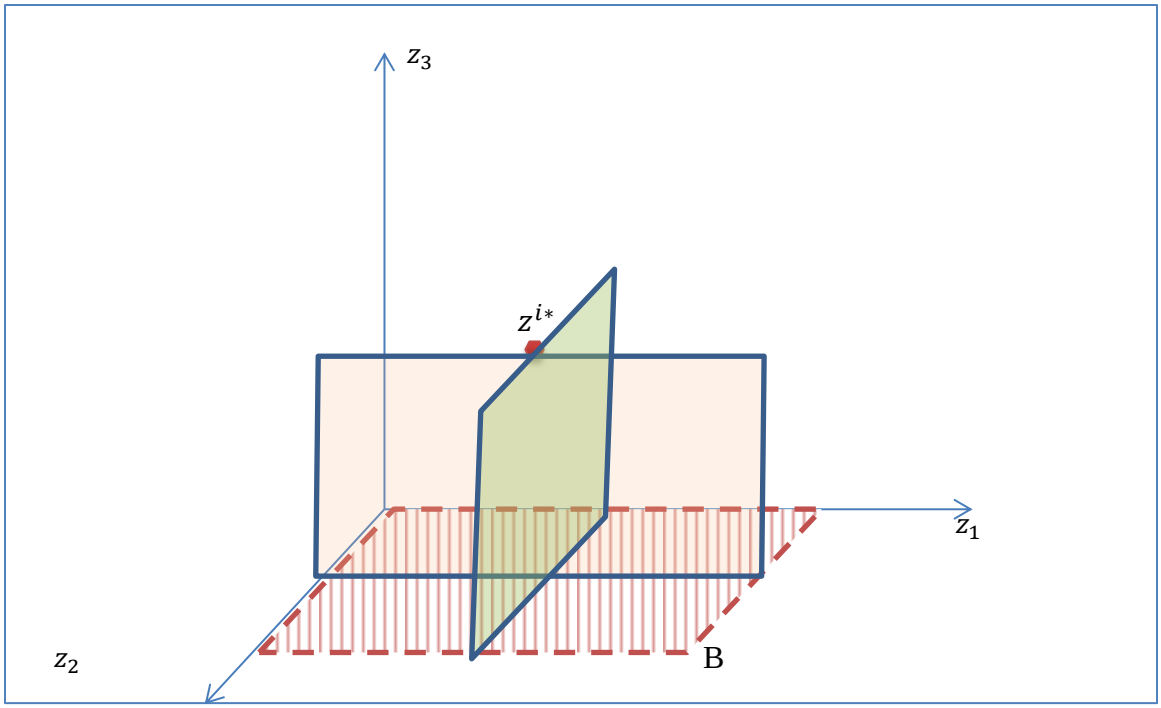


Figure 16: Partition space for a three-objective MOIP

The sub-problem solved for each sub-region corresponding to node i can be defined as:

$$\text{P2:} \quad \text{Max } z_p(x) + \varepsilon \sum_{j=1}^{p-1} z_j(x)$$

$$\text{s.t.} \quad ub_1^i > z_1 > lb_1^i$$

$$ub_2^i > z_2 > lb_2^i$$

$$\vdots$$

$$ub_{p-1}^i > z_{p-1} > lb_{p-1}^i$$

$$ub_p^i > z_p$$

$$x \in X$$

where lb_j and ub_j , $j = 1..p$ are the bounds on objectives, which are determined using the parent node for each node. In order to construct the sub-regions, $2p-1$ number of constraints are added at most to the existing X; p of them are upper bounds; $p-1$ of them are for lower bounding.

A nondominated solution that has the best value of objective p on $x \in X$ is generated by the objective function of P_1 at the root node. This means there is no need to search the region for better values of objective p (i.e., z_p), which is the pth objective value of last generated solution. Moreover, the objective function form prioritizing pth criteria with an augmentation term ensures that there can be no weakly nondominated solutions in the sub-region of N_i with a better pth objective value than z_p (Steuer 1986). Accordingly, we can exclude the solutions with z_p^{i*} value in the lower levels of the B&B search tree. Hence, the algorithm evolves in a way that generates smaller values in terms of the pth objective at every level, adding only an upper bound constraint for the objective p.

At the root node, all upper bounds are initially set at infinity, and all lower bounds initially start with negative infinity, which means there are no boundary constraints at the root node.

For each branch, i.e. child node, a different combination is created in order to maintain the general branching idea. For all nodes but the root node, boundaries are determined initially based on the Pareto solution produced by parent node. $\forall j \in 1 \dots p$, z_j^{i*} acts either as upper bound or lower bound for a child node of node i. So, with the exception of the root node, the jth objective lower and upper bounds for each sub-problem are determined by two values: (1) the jth objective of the solution obtained at the parent node, z^{i*} and (2) bounds of the parent node for the jth objective. At this point, an important remark should be made about difference of applying the upper and lower bound. For the problem P2, the upper bound is set as inequality, while the lower bound is set as strict inequality while using z^{i*}, i.e., $z_j \leq z_j^{i*}$ and $z_j > z_j^{i*}$ in order to ensure sibling nodes are optimized with mutually exclusive regions without identifying the same solutions. These boundaries need to be combined with the boundaries inherited from the parent

node in order to determine the sub-region of the node that will be optimized to find or fathom a new solution.

Based on the refinements mentioned so far, the model that needs to be solved at each node can be formulated as follows:

P3: Max $z_p(x) + \varepsilon \sum_{j=1}^{p-1} z_j(x)$

s.t. $ub_1^i \geq z_1 > lb_1^i$ (1)

$ub_2^i \geq z_2 > lb_2^i$ (2)

:

$ub_{p-1}^i \geq z_{p-1} > lb_{p-1}^i$ (p-1)

$ub_p^i \geq z_p$ (p)

$x \in X$

This can be seen in Figure 15 and Figure **16** clearly. In Figure 14, it can be observed that the area that has greater values in terms of objective p (in a three-objective context p=3) has been eliminated due to the fact that the best point of p^{th} objective has been obtained at the root node. That is, plane A in this figure represents the constraint p of model P3.

By eliminating the area that contains the values better than z_p^{i*} of node i, the number of sub-regions to be searched is reduced from 2^p. Since the number of dimensions is reduced by one, the resultant number of sub-regions, namely the branches on each node, becomes $2^{(p-1)}$.In

Figure 15, plane B represents the whole two dimensional search region, which has bounds on it for all but the 3rd objective.

3.3.1) Branching Strategy

Lower and upper bounds are attributed to each child node, k, $k = 1..2^{(p-1)}$. To determine the boundaries of the sub-region, the algorithm uses both the solution of the parent node and the bounds inherited from the parent node. Let us refer to the bounds determined by the new solution at node i, z^{i*} , "new solution bounds," and denote them as $lb_j^{k,SB}$ and $ub_j^{k,SB}$, $\forall j \in 1 \dots (p-1)$. These bounds can be determined by generating all combinations of strictly greater than,$>$, and less than inequalities,$\leq$, for each objective $j, j = 1 \dots (p-1)$. The following nested loop structure can be used to assign the bounds that need to be set due to z^{i*}:

for j_1=1..2

if j_1=1, $lb_1^{k,SB} = z_1^{i*}$ else $ub_1^{k,SB} = z_1^{i*}$

for j_2=1..2

if j_2=1, $lb_2^{k,SB} = z_2^{i*}$ else $ub_2^{k,SB} = z_2^{i*}$

:

for j_{p-1}=1..2

if j_{p-1}=1, $lb_{p-1}^{k,SB} = z_{p-1}^{i*}$ else $ub_{p-1}^{k,SB} = z_{p-1}^{i*}$

end

end

end

This procedure can be named the "child creation: step 1" since the child creation requires one more important step.

Similarly, let us name the bounds inherited from the parent node as "parent node bounds" and denote them as $lb_j^{k,PB}$ and $ub_j^{k,PB}$, $\forall j \in 1 \dots (p-1)$. The necessary methodology to determine the final bounds for each child node k, lb_j^k / ub_j^k, which is used in the model of the node, can be summarized as follows:

For $\forall j \in \{1 \dots (p-1)\}$ at node k

If $lb_j^{k,SB} < lb_j^{k,PB}$ then $lb_j^k = lb_j^{k,PB}$ else $lb_j^{k,SB}$

If $ub_j^{k,SB} > ub_j^{k,PB}$ then $ub_j^k = ub_j^{k,PB}$ else $ub_j^{k,SB}$

It can be summarized as choosing the most restricting value among $lb_j^{k,SB}$ and $lb_j^{k,PB}$/ $ub_j^{k,SB}$ and $ub_j^{k,PB}$ for each objective; there is a smaller value for the upper bounds, i.e., $lb_j^k = \max\{lb_j^{k,SB}, lb_j^{k,PB}\}$; a bigger value for the lower bounds, i.e., $ub_j^k = \min\{ub_j^{k,SB}, ub_j^{k,PB}\}$. This procedure can be named as "child creation: step 2." At the end of these two steps, we perform a minor verification, as follows:

If $\exists j \in \{1 \dots (p-1)\}\ s.t.\ lb_j^k = ub_j^k$ then $fathom\ node\ k$

Indeed, the search region of this node will already be contained in other nodes. Proposition 2 in the next section shows this result.

As mentioned previously, lower bounds are applied as strict inequalities, while upper bounds are applied as inequalities. In the pure integer context, strict inequalities can be converted to

inequalities by subtracting δ from ub_j^k, where δ is a positive real number with $\delta \ll 1$. Recalling that this algorithm is proposed for MOIP models, each solution with different objective values differs from the other by at least "1"; thus we can use $\delta = 1$, or $z < ub \equiv z \leq ub - 1$. This convention is also useful in creating mutually exclusive search spaces for each sibling node, which is explained within the proof of Proposition 1 in the next section.

It is indicated previously, that there are $2^{(p-1)}$ child nodes of a node. However, special structure of the objective function and the way branches are constructed can further be exploited to reduce this number. That is, searching in the sub-region of one of the child nodes is actually redundant since it is guaranteed to generate a dominated solution unless infeasible. Hence, the number child of nodes can be reduced by one for each node; i.e, $2^{(p-1)} - 1$ nodes are created per node. The reason this particular region contains dominated solutions is clarified with the following proposition.

Proposition 1 ***(Redundant subregion elimination):*** Let $N_{k(2^{(p-1)})}$ denote child node of N_k, which has $Y_{k(2^{(p-1)})}$ as set of non-dominated points; the sub-region is determined with the following bounds: $z_1 \leq ub_1^{k(2^{(p-1)})}, z_2 \leq ub_2^{k(2^{(p-1)})}, \dots, z_{p-1} \leq ub_{p-1}^{k(2^{(p-1)})}$ for P1. It can then be concluded that $Y_{k(2^{(p-1)})} = \{z^{k*}\}$; the same solution as parent node or a dominated solution is obtained as the result of this solve.

Proof: Let $z^{k(2^{(p-1)})}$denote the solution that is generated in this particular branch. We know that the points generated by child nodes generate results with equal or worse values than the parent node (i.e., $z_p^{k(2^{(p-1)})} \leq z_p^k$). For the other objective values of this child node, $j = 1..p-1$, since $ub_j^k = \min\{ub_j^{k,SB}, ub_j^{k,PB}\}$.

Let's assume for all dimensions that $ub_j^{k,SB} < ub_j^{k,PB}$; this means all the $z_j^{k(2^{(p-1)})}$ are forced to be less than or equal to z_j^{k*} for all but the main objective. The objective function has to find the solution that has greatest z_p on this region, which means $z_p^{k(2^{(p-1)})} = z_p^{k*}$. We also know that all other objectives are required to be less than or equal to the z_j^{k*} . Hence, the objective function that identifies the greatest value for the main objective and a total sum of other objectives is supposed to generate the same solution $z^{k(2^{(p-1)})} = z^{k*}$.

Let's assume for objective i, $ub_i^{k,SB} > ub_i^{k,PB}$ and $ub_j^{k,SB} < ub_j^{k,PB}$ for all others, which means all the $z_j^{k(2^{(p-1)})}$ are forced to be less than or equal to z_j^{k*} for all but objective i. Since $ub_j^k = \min\{ub_j^{k,SB}, ub_j^{k,PB}\}$ for node k, it will be $ub_i^{k(2^{(p-1)}} = ub_i^{k,PB}$ for child $2^{(p-1)}$. Hence, $ub_i^{k(2^{(p-1)}} < z_i^{k*}$. The objective function has to find the solution that has greatest z_p on this region, which means $z_p^{k(2^{(p-1)})} = z_p^{k*}$. We also know that all other objectives are required to be less than or equal to the z_j^{k*} .On the other hand, the solution will be strictly less than z^{k*} for objective I. Hence, the solution that will be generated under these circumstances will be strictly worse than z^{k*} for one objective, even though it is equal to z^{k*} in all other objectives. So $z^{k(2^{(p-1)})} \ll z^{k*}$.

This conclusion completes the proof ■

3.3.2) Node selection and Stopping Condition

In the classical B&B method for integer programs, "breadth first" and "depth first" are the two main strategies followed while traversing the tree. Depending on the use cases of the model, both have some advantages and disadvantages. In the proposed approach, the algorithm starts with

"breadth first" traversal of the tree. Figure 16 illustrates how breadth-first evolves with child creation procedure.

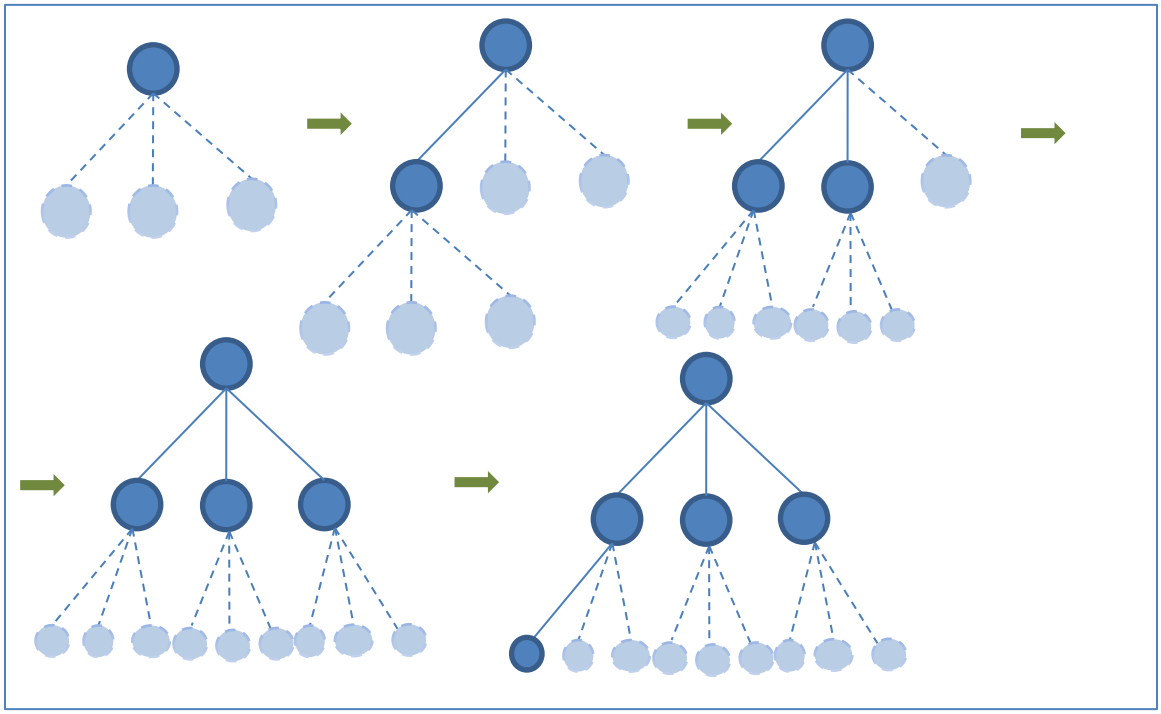


Figure 17: Proceeding of breadth first on a B&B for a three-objective MOIP problem

When this algorithm is allowed to terminate without any intervention, it generates all Pareto solutions of the original problem. The following two propositions prove the validity of this claim. The first proposition establishes the mutual exclusivity of the child nodes in the objective space.

Proposition 2 (***mutually exclusive solutions***): Let $N_{k1}, N_{k2}, .., N_{2^{(p-1)}-1}$ denote the child nodes of N_k. Let $Y_{k1}, Y_{k2}, .., Y_{2^{(p-1)}-1}$ denote the set of non-dominated solutions that can be generated by the subtree of each child node, with Y_k being the set of non-dominated points for N_k itself. Then we have mutual exclusivity, $Y_{km} \cap Y_{ki} = \emptyset,\ \forall m, i \in \{1 \dots (2^{(p-1)} - 1)\}$,

Proof: Let all the feasible solutions in search space of k be denoted by S_k. It is then observed that $Y_k \subset S_k$. At this point, it is assumed all alternative solutions, i.e., those with the same objective values but that differ in terms of decision variables, can be obtained once one of the alternatives is identified.

Recalling the steps of child creation, namely steps 1 and 2, the bounds for the solution of this node can be shown as bounded as $ub^k \geq z^k > lb^k$. Using this result, the following scenarios emerge in terms of the boundaries of child nodes:

The solution for parent node, z^{k*}, might fall in somewhere between the upper and lower bound, or $ub^k > z^{k*} > lb^k$. Then the boundaries for each child node of k are determined by using z^{k*} and ub^k& lb^k by following the procedure defined in previous section. As *child creation: step 1* of the procedure suggests, z^{k*} is used as lower and bounds for each objective $j, j \in \{1, .., p-1\}$ in such a way that each child will have a different set of bounds. For those child nodes that have z_j^{k*} as the lower bound, the ub_j^k acts as the upper bound; the other child nodes that take on the upper bound values as z_j^{k*} ; lb_j^k act as the lower bound. In this setting, even if two nodes use z_j^{k*} as the lower or upper bound at the same time, and both nodes generate solutions on this border, these solutions have to differ in terms of at least one of the objectives because of the different combination of boundaries created; hence, the solutions are different from each other.

The second case is when $ub_j^k = z_j^{k*}$ for the child nodes $lb_j = z_j^{k*}$ and accordingly $ub_j = ub_j^k$; hence, the search region on the j^{th} dimension reduces to a single point, which is z_j^{k*}. Assume this child node has the set of A as the upper and lower bounds for the rest of the

objectives, i.e., A=$\{ub_{jj}, lb_{jj}\}$ where $jj = 1..p-1 \; and \; jj \neq j$. However, there is another node with the same set of bounds A, but $ub_j = z_j^{k*}$; since upper bounds are set as inequality, this allows for the generation of solutions with $z_j = z_j^{k*}$. Hence, nodes with $lb_j = z_j^{k*}$are redundant and fathomed with a check in the procedure; remaining nodes do not have overlapping regions, as shown in the first part of the proof. ■

Hence, each child node is a mutually exclusive partition of the solution space defined by the parent node. Another important characteristic is whether these mutually exclusive sub-regions, defined by all child nodes, cover the entire solution space of the parent node. Next, we show that these child nodes cover for the search region of parent node minus its solution point; i.e., the set of non-dominated points of a parent node can be obtained by finding all the non-dominated points of the child nodes except for the solution point obtained at the parent node.

Proposition 3*:* $Y_{k1} \cup Y_{k2} \cup \ldots \cup Y_{k[(2^{(p-1)})-1]} \equiv Y_k - \{z^{k*}\}$

Proof: At the branching strategy part, it has been shown that S_i, or the set of all feasible solutions that lies on $ub^k \geq z^k > lb^k$, is divided into two around z^{k*}on all dimensions except for p. At this point, it can be observed that all dimensions of z_j^{k*} $j \in \{1,..,p\}$ are used as bound and are included in the search space of the child node m as upper bound, $ub_j^m = z_j^{k*}$, when $z_j^{k*} < ub_j^m$. It has already been shown in a previous proof that a child node is fathomed when $z_j^{k*} = ub_j^m$ and includes some other nodes as the upper bound. The only region that is left that is also covered by child nodes is the upper border of the parent node, i.e., ub^k. By resorting to the same case partitioning again, if the solution of parent node is between the upper and lower bounds $ub^k > z^{k*} > lb^k$, there will be some child nodes that use ub^k as the upper bound and z^{k*} as the lower

bound. And ub_j^k is divided into two for the rest of $p-2$ objectives; hence, it will be covered as being the upper bound by node I, which has also one of "$z_m^i > z_m^{k*}$ "or "$z_m^i \leq z_m^{k*}$" combinations of $m = \{1, \dots, p-2\}$. If the solution is on the upper bound of some or all of the coordinates, i.e., z^{k*}=ub^k, the upper bound can be thought as the z^{k*} in the previous argument, and it can be concluded that ub^k is covered for this case, too.

Furthermore, only one of these regions is omitted due to Proposition 1, which is proven not to contain any new non-dominated solution.■

Hence, it is proven that the proposed algorithm does not generate duplicate solutions and explores the entire solution space in which nondominated solutions might exist.

Lemma 1: All the nodes at the same level have mutually exclusive solution sets, and a combination of their search regions is equal to the search area of original problem.

Proof: From the observations in Propositions 2 and 3, we arrive to this result by considering that all child nodes on a level are the children or grandchildren of some sibling nodes. We know that siblings have no intersection and cover all the search space of their own parent.

A numerical example is shown in Figure 18, and Figure 18 illustrates the steps of the proposed algorithm for the following knapsack problem.

$$Max\ \{Px\}$$

$$s.t. Wx \leq\ q$$

$$x \in \{0,1\}^{10}$$

Where

$$P = \begin{bmatrix} 54\ 64\ 46\ 37\ 31\ 62\ 52\ 33\ 87\ 35 \\ 52\ 65\ 58\ 63\ 46\ 66\ 72\ 95\ 42\ 29 \\ 56\ 90\ 34\ 13\ 71\ 33\ 66\ 74\ 88\ 71 \end{bmatrix}$$

$$W = \begin{bmatrix} 52 & 52 & 28 & 23 & 95 & 69 & 13 & 61 & 32 & 68 \\ 88 & 98 & 49 & 28 & 43 & 98 & 53 & 52 & 84 & 66 \\ 57 & 30 & 86 & 50 & 97 & 96 & 59 & 94 & 67 & 14 \end{bmatrix}$$

$$q = [246 \quad 329 \quad 325]^T$$

$$x = [x_1 \quad x_2 \quad x_3\, x_4 \quad x_5 \quad x_6\, x_7 \quad x_8 \quad x_9 \quad x_{10}]^T$$

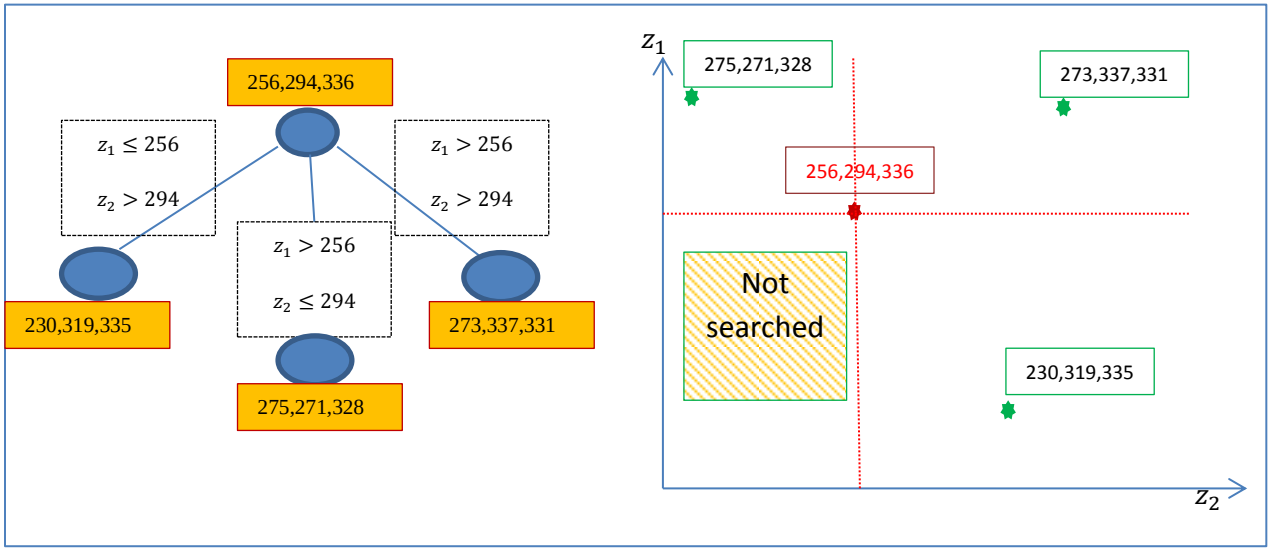


Figure 18: Root node and its branches and their corresponding region projected on two dimensions for the sample problem

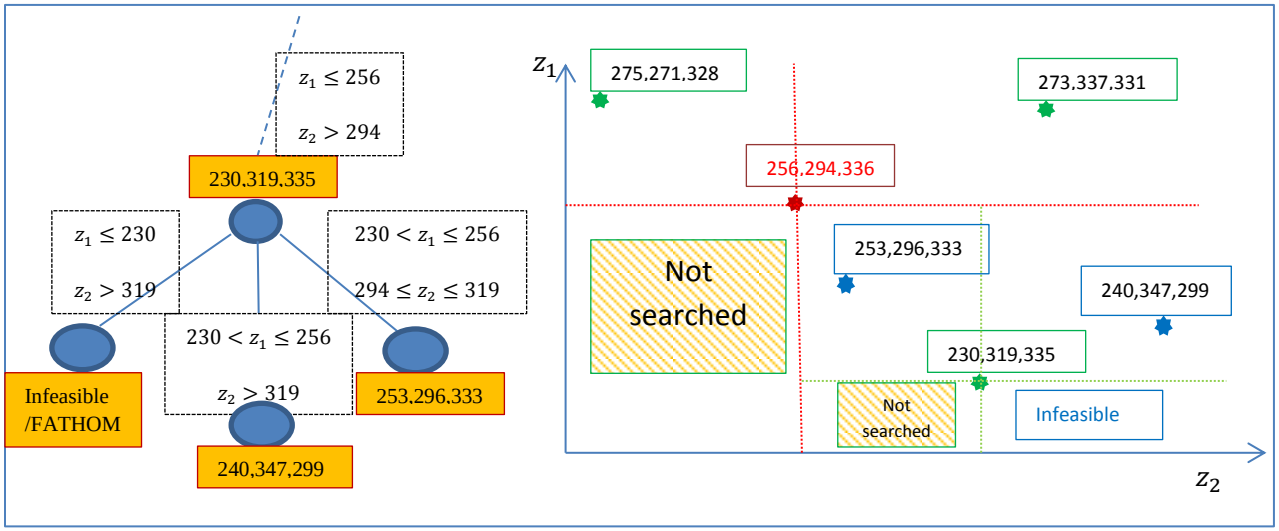


Figure 19 : One of the child nodes of the root node and its branches and their corresponding region projected on two dimensions for the sample problem

It should be added that even though a node is automatically fathomed per the infeasibility (see Proposition 1), the problem size will grow exponentially due to the addition of an exponential number of nodes. Hence, it is always worth investigating what further actions can be taken in order to reduce the number of nodes to be explored. At this point, it is important to delve further into domination relations between nodes of the B&B tree. Propositions in the next section attempt to reveal these relations more rigorously. For our purposes, we make the following definition:

A "***dominating corner (DC) node***" of a parent node is the one with all branching constraints that guarantees that the solution will be better than the solution of parent node for all but the main objective, or if this node is represented by i, $z_m^i > z_m^{k*} \ \forall m = \{1, \dots, p-1\}$. All other child nodes are called "***regular nodes***."

3.3.3) Domination Relationship between Nodes

The following two propositions establishe the dominance relationships between parent and child nodes.

Proposition 4: None of the child nodes can dominate the parent node.

Proof: Recall that node that is worse than all but the main objective is excluded from the search.
In the remaining nodes, there are regular nodes and the dominating corner node.
For the regular nodes, it is required that they are better than their parent node for at least one objective with the branching constraints, but worse at least for one objective.

For the DC node, all but the main objectives are required to be better than parent node. Assume there does exist a solution which is strictly better than all but main objective is generated by this child, which has equal value of main objective to the parent node for the main objective, p, then

the objective value would be higher than the value obtained at parent node. However, this contradicts the fact that, parent node is maximized over the superset of the same search space with the same objective function.■

Lemma 2: A parent node cannot dominate a child node.

Proof: The proof follows from the fact that branching constraints are set up in a way that the solution of the parent node is improved for at least one of the objectives for each child node. According to Proposition 1, only the child needs to be less than the solution of parent node is never created.

The following propositions are presented to show the relationship between sibling nodes and nodes in different generations (levels).

Lemma 3: A regular child node of a parent has a solution whose main objective equals or less than solution of parent node; the solution of the DC node's main objective is strictly less than the main objective value of parent node.

Proof: The proof follows from the last part of Proposition 4 for a DC node. On the other hand, a solution with the same value as main objective can be generated by a regular node, with the requirement that total of all but the main objective values is less than the parent node's solution value. That is, let i denote this child node, and z^{i*} and z^{k*} solutions of the child node and parent node, respectively; then, $z_p^{i*} = z_p^{k*}$can happen if $\sum_m^{p-1} z_m^{i*} < \sum_m^{p-1} z_m^{k*}$. Otherwise, z^{i*} would have been identified as the optimal solution of the parent node.■

Proposition 5: Solution of a DC node can dominate the solutions obtained from regular siblings of the DC node.

Proof: Let us denote the solution obtained from DC node as z^{DC*},the solution of a regular sibling i as z^{i*}, and the solution of their parent k as z^{k*}. Both of the child nodes will have z_p^{k*} as upper bound on their pthobjective. Let node i be required to be better than z^{k*} for a set of objectives $A \subset \{1, \dots, p-1\}$ and worse than or equal to z_m^{k*} for objective(s) m, s.t. $m \in \{1, \dots, p-1\}\backslash$A and $m \neq \emptyset$ based on the definition of regular node. Then, if $z_p^{i*} < z_p^{k*}$ and $z_p^{DC*} \leq z_p^{k*}$, which is a possible case, and $z_n^{i*} \leq z_n^{DC*}\ \forall n \in A$; besides, it is already guaranteed that $z_m^{i*} < z_m^{DC*}\ \forall\, m \in \{1, \dots, p-1\}\backslash$A , by the nature of branching constraints; i.e., DC node is forced to be better than z^{k*} for all but objective p , whereas $z_m^{i*} \leq z_m^{k*}$. Hence, $z^{i*} \ll z^{DC*}$■

Proposition 6: The child of a DC node can dominate a solution obtained from sibling nodes of corresponding DC node.

Proof: In addition to the notation used in the previous proof, let DC_j denote the child node of the DC node and its solution by z^{DC_j*}. Then, it is claimed that there can be cases where $z^{DC_j*} \gg z^{i*}$ even if z^{i*} is not dominated by z^{DC*}. We first need to assume that $ub^{DC_j} - lb^{DC_j} \geq 2$ and $ub^{DC} - lb^{DC*} \geq 3$ in all objectives, which is a possible case.

Let $z_p^{i*} = z_p^{k*} - 1$ (since it is allowed that $z_p^{i*} \leq z_p^{k*}$, this outcome is possible) and $z_p^{DC*} = z_p^{k*}$(since it is allowed that $z_p^{DC*} \leq z_p^{k*}$, this outcome is also possible). In addition to this, set A is defined in the same way as the previous proof: $z_n^{i*} > z_n^{k*}, n \in A, A \subset \{1, \dots, p-1\}$; hence, $z_m^{i*} \leq z_m^{k*}$; hence, $m \in \{1, \dots, p-1\}\backslash$A and $m \neq \emptyset$. Then, assume $z_p^{DC_j*} = z_p^{k*}$, (since it is allowed that $z_p^{DC_j*} \leq z_p^{DC*}$, this outcome is possible); and for the rest of the objective dimensions and $z_l^{DC_j*} \geq z_l^{DC*} > z_l^{k*}$ where $l \in \{1, \dots, p-1\}$. On the other hand, assume for the subset of A, $z_n^{DC*} = z_n^{k*} + 2$ and $z_n^{i*} = z_n^{k*} + 1, \forall n \in A$ that z^{DC_j*} dominates z^{i*} for the

objectives in set A. For the rest of the objectives, it is already guaranteed that $z_m^{i*} \leq z_m^{k*}\ m \in \{1, \dots, p-1\} \backslash A$ with the branching constraints of node i. This observation completes the proof■

Based on this proposition following lemma can be added.

Lemma 4: The child of a DC node can dominate a solution obtained from the children of the regular siblings of the DC node.

Proof: For this case, assume that $ub^{DC_j} - lb^{DC_j} \geq 3$ and $ub^{DC} - lb^{DC*} \geq 4$ in all objectives, which a possible case. In addition to the notation used in previous proof, let i_j denote the child node of regular node i, which is also a sibling of node DC. Following the proof of Proposition 6, if it is assumed that $z_n^{i_j*} = z_n^{i*} + 1 = z_n^{k*} + 2$ and $z_n^{DC*} = z_n^{k*} + 3$ for $\forall n \in A$, it can be concluded that z_n^{DC*}dominates $z_n^{i_j*}$ ■

All of these conclusions are represented in Figures (a) and (b), where (a) shows the case between DC and regular siblings of a generation, and (b) shows the case between two generations. The second generation is classified only based on the nature of their parents.

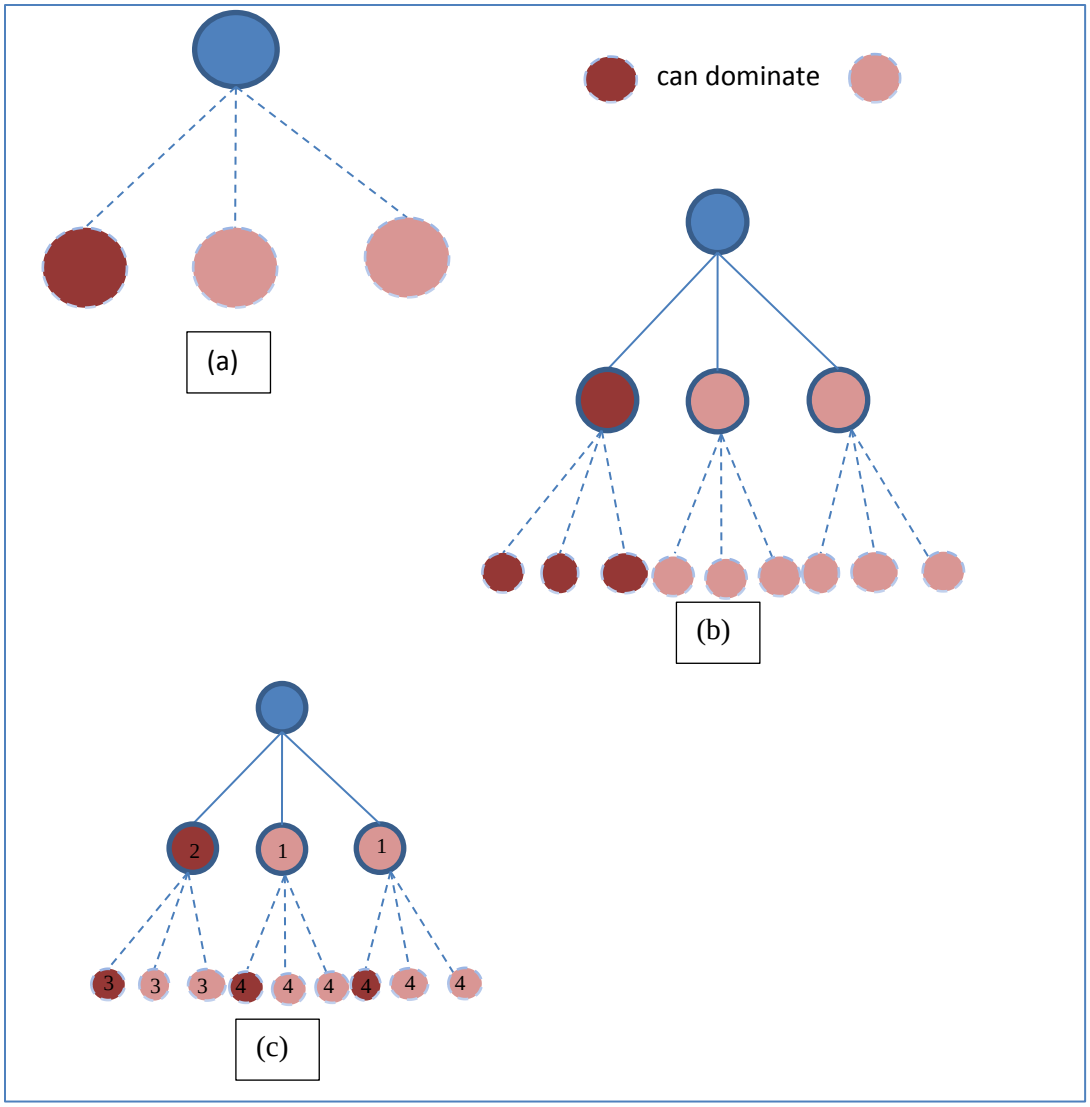


Figure 20: Domination relations (a) for sibling nodes based on their own types, (b)for nodes in different generations based on their parent types, and (c) for the nodes in different generations based on both their parent types and their own types

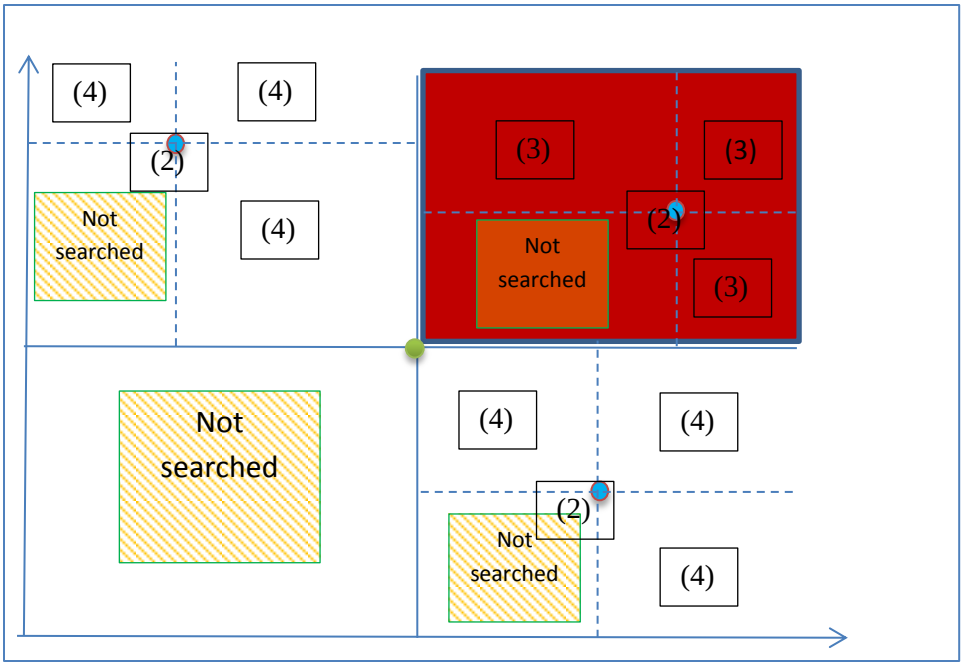


Figure 21: Representation of each node type for a problem with three objectives

Table 7 summarizes the domination relationships among nodes more accurately. It can also be observed in Figure 19, Part (c). In addition, Figure 20 shows the representation of the each node type for a problem with three objectives.

1. Regular node
2. DC nodes
3. Child of the DC node
4. Child of the regular node

Table 8: Domination relations: (if row element dominates column element, 'yes'; otherwise 'no')

	(1st type)	**(2nd type)**	**(3rd type)**	**(4th type)**
(1st type)	**no**	**no**	**no**	**no**
(2nd type)	**yes**	**-**	**no**	**yes**
(3rd type)	**yes**	**no**	**yes**[1]	**yes**
(4th type)	**no**	**no**	**no**	**yes**[2]

3.3.4) Fathoming of Nodes

Based on the domination relations presented for the proposed algorithm, some nodes can be removed from further consideration; in other words, there is no further branching on these nodes. This is the "fathoming" operation, as in the case of single-objective B&B. Fathoming in the proposed approach is performed for two reasons: infeasibility of a node or derivation of some upper bounds through the use of domination relations between nodes, as described in the previous section.

1. **Fathoming due to infeasibility:** Following lemma shows that a node with infeasible solution can be eliminated from further consideration.

   Lemma 5: If the solution of a node is infeasible, all of its child nodes will result with infeasibility.

   Proof: According to Proposition 3, a parent node has the search region that contains all the Pareto solutions that can be generated by its child nodes. So if the Pareto set existing in the search region of node k is empty $Y_k = \emptyset$, then $Y_m = \emptyset \ \forall m \in \{1 \dots \ (2^{(p-1)} - 1)\}$.

[1] This is possible if this child node is also DC node
[2] This is possible if this child node is also DC node

2. **Fathoming due to upper bounding (domination):** A DC node solution can dominate its regular sibling based on Proposition 5. Besides, descendants of a regular node can be dominated by its DC sibling and all descendant nodes this DC node, based on Proposition 6 and Lemma 4. These two simple derivations cause nodes to generate a dominating result of solutions that can be used as upper bounds for fathoming the next nodes to be created. Due to Lemma 2, this node cannot be fathomed totally. It has the potential to create non-dominated solutions. On the other hand, if a node itself is dominated either by its DC sibling or descendant of a DC (grand) uncle, the child nodes that will be created from this dominated node can also be dominated by the same node. This result can sometimes be reached directly, just by comparing the upper bounds attributed to the new child node with the dominating node. If the upper bound of the newly created node is dominated by the same node that dominates the parent node, this child is fathomed totally.

   Figure 21 shows a sample case with three objectives. In this figure result of node (3) is dominated by its DC sibling, node (1). After observing this domination, the bounds of the child nodes of node (3) are determined. However, before solving these child nodes, if the upper bounds of each child node (3.1, 3.2 and 3.3) are compared with dominating node(1), it is observed that none of the solutions obtained from node (3.2) can outperform the result obtained from node (1); hence, this child node can be fathomed.

As it can be observed from this example, with increasing fathoming, the size of the tree to be traversed shrinks accordingly. However, keeping track of ancestors of a node requires significant memory. The task of keeping the record and comparisons of the DC node versus regular node might be cumbersome. Besides, creating an upper bound pre-requires finding a node that

dominates the parent node. This means that before coming up with an upper bound, some domination checks needs to be made. As the number of this comparison increases, the runtime of whole algorithm also increases. Hence, it is possible to design the algorithm in a way that keeps the record of ancestor relations for a reasonable number of generations. This way, the size of set of nodes that can dominate the newly generated solutions is kept in a reasonable size. Accordingly the number of domination checks and the memory requirements for storing the record of relations between nodes is kept manageable.

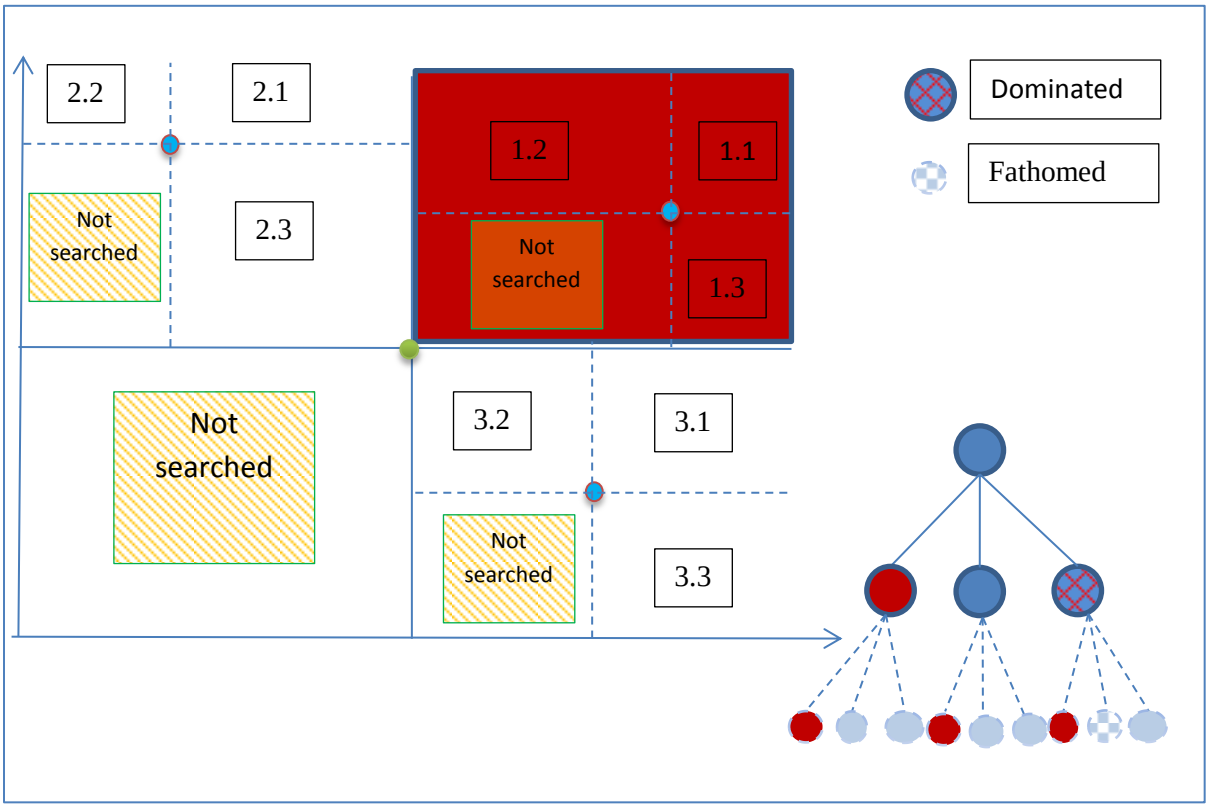


Figure 22: Fathoming sample on a problem with three objectives.

The following two propositions show that the proposed algorithm converges and that the converged set includes all the Pareto efficient solutions.

Proposition 7: Proposed algorithm terminates.

Proof: Assume one node is created and solved for all the solutions on the feasible space. Since it is proven that none of the nodes generate duplicate solutions, and solution set of the problem is finite; child nodes that are created from the last identified solutions will result with infeasibility; hence, it will be fathomed■

Proposition 8: Proposed algorithm can generate all Pareto points.

Proof: All of the solution space is covered by search space of root node since upper and lower bounds of root node is plus and minus infinity. The objective function optimizes main objective, p, over this region, avoiding the weakly dominated solutions, i.e., solutions with the same main objective value, but having a smaller value of sum of other objectives. By Lemma 3, there is no way of skipping a value between generations; i.e., there cannot be a solution z^* between the optimal solution of node k, z^{k*} , and solution of one of its child node i, z^{i*}, such that $z_p^{i*} < z_p^* < z_p^{k*}$. Besides, by Proposition 3, child nodes cover the same space without skipping any region. Only region that is excluded is the part of solution space where it is proven that no Pareto solution can exist by Proposition 1. That is, no solution point which can be a Pareto candidate is skipped in both vertical and horizontal directions of p. Then, it is obvious that algorithm will terminate with a solution set which contains Pareto set of the problem■

Effect of fathoming can be seen in the following tables which belong to the small knapsack problem used to show branching structure and presented above where first objective is the main objective.

Table 9: Results without fathoming in 4.813 seconds

node #	Obj 1	Obj 2	Obj 3
1	**286**	**300**	**291**
2	**273**	**337**	**331**
3	261	324	236
4	**275**	**271**	**328**
6	**240**	**347**	**299**
7	**230**	**319**	**335**
8	234	333	267
10	*253*	*310*	*259*
11	**256**	**294**	**336**
12	271	288	328
13	251	247	330
20	**232**	**353**	**277**
23	*211*	*311*	*280*
24	*203*	*317*	*258*
25	230	304	274
27	**253**	**296**	**333**
30	*238*	*298*	*296*
44	*215*	*310*	*282*

Table 10: Results after fathoming applied by using the nodes at the same level in 3.109 seconds

node #	Solution point 1	Solution point 2	Solution point 3
1	**286**	**300**	**291**
2	**273**	**337**	**331**
3	261	324	236
4	**275**	**271**	**328**
6	**240**	**347**	**299**
7	**230**	**319**	**335**
8	234	333	267
11	**256**	**294**	**336**
12	271	288	328
13	251	247	330
20	**232**	**353**	**277**
24	**253**	**296**	**333**

In this small sample, rows which are presented in bold represent the true Pareto solutions. Solutions shown in italics in Table 8 are the solutions of fathomed nodes which do not appear in Table 9. Total number of nodes traversed is 44 in the first tree, whereas this number reduced to 24 in the tree with fathoming. More exhaustive test results are presented later in the computational experiments section of the chapter which aims at assessing the effect of fathoming.

3.3.5) Pareto Filtering Strategies

Based on the domination relations presented previously, the proposed approach finds solutions that can be dominated by another solution that is obtained in the later stages of the solution process. In order to have a 100% non-dominated set, it should be waited till the algorithm terminates. At this point, it is obvious that some type of filtering is necessary in order to eliminate dominated points from the solution set, generally called as "Pareto Filtering." This step creates another variation point for the proposed algorithm. The most common approach is to perform filtering in the end. Alternatively, the filtering could be performed after finding of a new solution or intermittently, e.g., filtering every other 100 solutions or at the end of each level. The obvious tradeoff is the computational time spent during the filtering and the improved efficiency through the increased and timely node fathoming.

Simultaneous Filtering: At the end of a MIP solve at each node, unless it ends up with infeasibility, a new solution is obtained and this solution can be dominated by or dominates one of solutions added previously to the solution set. If a solution named based on the node it is obtained, and is represented by c, the set of nodes that have the potential to generate the solutions which can dominate the solution c, can be represented by $S(c)$; while set of nodes which can generate solutions that can be dominated by the a can be denoted by $I(c)$. So filtering

means, after the new solution is obtained at c , checking if it dominates any of the solutions in $I(c)$; or being dominated by any of the solutions obtained from $S(c)$. This later step is similar to the actions that can be taken during the derivation of upper bound to be used for fathoming. However, covering all $S(c)$ is not a must for fathoming; on the other hand if one wants true information about Pareto optimality of a solution, it is necessary to make sure all elements of $S(c)$ is covered.

Elements of $S(a)$ and $I(a)$ are determined according to the propositions presented in the section about "domination relationships between nodes." According to these relations, node c, can be in S set for node a, but can be in set I for another node b, i.e. $c \in S(a), c \in I(b)$. Figure 23 below shows this type of a node on a three-objective-case. Since b is child of the DC uncle of a and c, it can dominate both of them; However, since c is DC node of its own small family, it can dominate its regular sibling a. Based on this observation it can easily be deduced that, S and I sets of each node needs to be dynamically updated during a tree traversal, which require keeping significant amount of information in memory or some type of recording process which requires and writing in order to construct the each S and I set of a solution correctly. By following this filtering technique it can be guaranteed that none of the solutions obtained so far can be dominated by the solutions in the candidate solution set, although it does not guarantee that one of these points cannot be dominated by a solution that can be obtained by one of the succeeding nodes. Besides, the result of domination check for set S can also be used for fathoming of nodes. As the cardinality of set S increases, fathoming becomes more powerful in reducing the size of the tree; hence, using the information regarding set S of each node fathoming due to upper bounding is used at with highest performance.

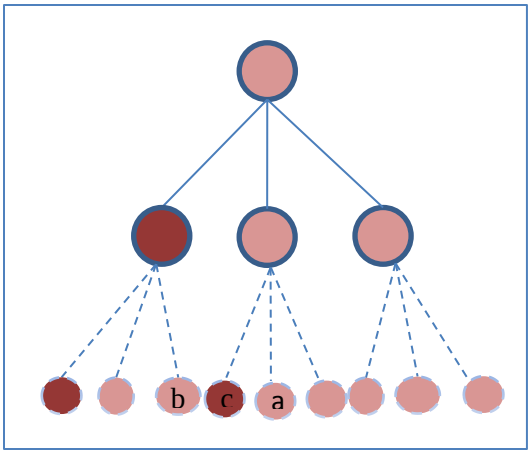


Figure 23: Changing S and I memberships for a node in a MB&B tree with three objectives

Final filtering: As opposed to filtering the results simultaneously, one can choose to filter all the solutions obtained during the traversal of tree at the end of termination of tree search as a last step to the algorithm. The filter that is used for this type requires checking if a solution is dominated by any of other the solutions in the solution set, which is done by each solution with all others and removing it from the set if it turns out to be dominated by one of the solutions. The advantage of this method is that it needs less memory space since it does not necessarily require to keep the information among nodes. On the other hand, its disadvantage is that is does not have a clear idea about the domination chances of a solution point, which is a concept that will be clarified in 3.3.6*) Probability of being non-dominated for solution set*.

Intermittent filtering: The third option is to have intermittent filtering. To summarize, simultaneous filtering goes hand in hand with the fathoming procedure and can supply all the information that is necessary for fathoming as well, which helps keep the tree size under control. Furthermore, by keeping the domination relationships among all nodes, we only need to compare a solution with the ones that have the potential to dominate. On the other hand, retrieving all the information regarding the previous nodes requires either memory or spending time on read/write processes, and final filtering does not suffer from this advantage. Hence, a filtering approach

based on intermittent filtering keeps the complete information in memory for fathoming (i.e., S(c), for each node), but performs the final filtering at the end (i.e., not keep information of I(c) for any node c). Since this approach provides all the available information necessary to take advantage of fathoming while reducing the storage requirements, this method is used during the computational tests conducted for the assessment of the general performance of the algorithm.

3.3.6) Probability of being non-dominated for solution set

As stated above, not all solutions generated up until an intermediate stage of the algorithm are guaranteed to be Pareto optimal. Hence, we developed a probability measure of being dominated that can be helpful when results are evaluated by the user if the proposed approach is terminated before the convergence. Similarly, these results, at intermediate steps, could be used to provide real-time information to the user on the non-dominance probability of solutions found thus far. To the best of our knowledge no other existing approximation methods, either exact or meta-heuristic, supplies this type of information.

A solution can be dominated if another point is generated that has equal or better values for all the objectives, one of which is strictly better. Indeed, the best solutions for each objective, namely the anchor points, supply the broadest estimate how much a solution can be improved upon in one dimension on the solution space. The probabilities are calculated for each objective in the objective space. Hence, this value can always be used in order to calculate the sample space for this probability. However, we claim that it is possible to come up with more accurate estimates about this probability in the context of proposed algorithm. Actually, the search space is restricted by the area of active nodes, which means the search space can be much smaller than the region determined by the anchor points. Hence, the total area defined by all active nodes

which have the potential to generate points that can dominate the current node give the sample space for probability. So, the calculation of this measure can be summarized as follows:

$M(c)$: Set of active nodes when node c is about to be solved

$$\text{Total Search Area}_i = \sum_{d \in \text{M(c)}} ub_i^d - lb_i^d \qquad \forall i \in p,$$

$$\text{Total Domination Area}_i = \sum_{d \in \text{M(c)}} ub_i^d - ub_i^c \qquad \forall i \in p$$

$$Probability\ Measure_c = \prod_{i=1}^{p} \left(1 - \left(\frac{\text{Total Domination Area}_i}{\text{Total Search Area}_i}\right)\right.$$

The above probability measure is defined for node c's solution. An aggregated probability for the entire set of solutions could be calculated by averaging of all the individual solution's probability measures. This type of a measure is calculated for the samples in the computational results part.

Given the critical role of the active nodes in accurately estimating the probability of domination, the node selection strategy of the proposed MOB&B for branching is breadth first. This way we ensure that we always have well-dispersed active nodes with similar sized node specific unexplored search spaces. Hence, at any given time, the set of active nodes can be either from the same level or at the next level of the current node. However, this is not a necessary condition to calculate this probability. If one wants to apply the algorithm with the depth-first approach, the calculation presented previously is still valid.

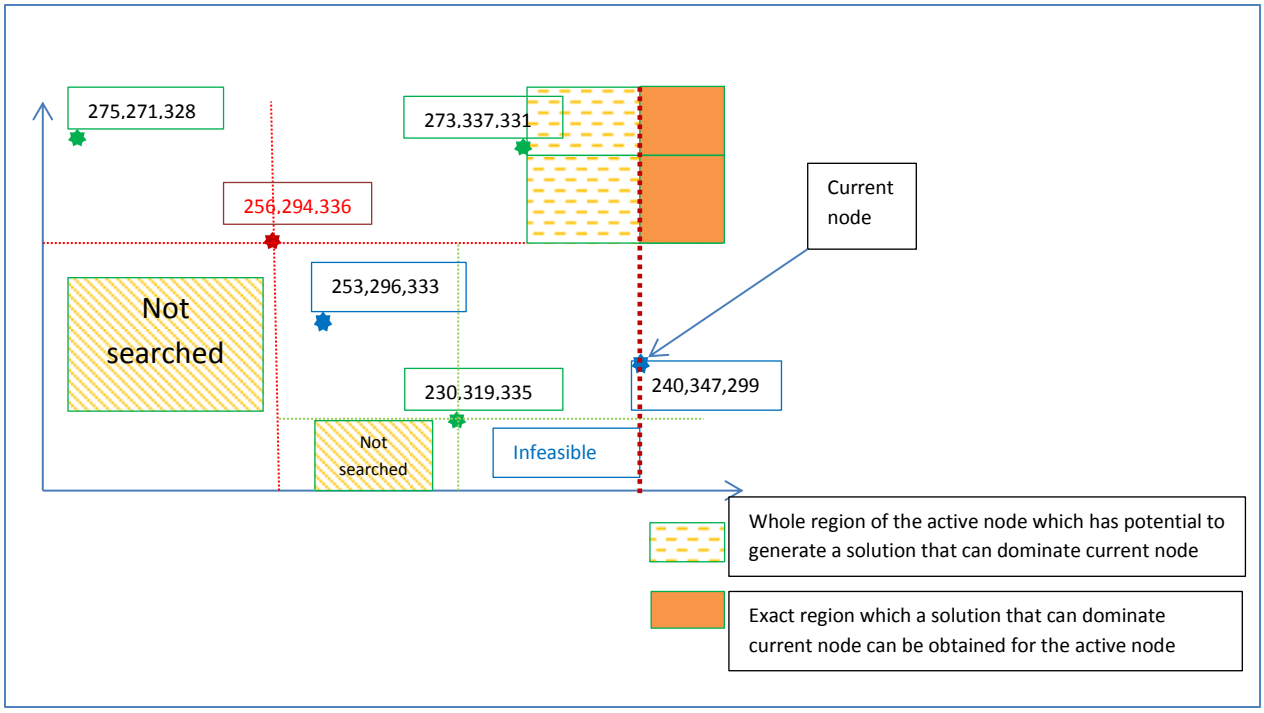


Figure 24: Current node and relevant regions of active node represented on the three objective MOIP example

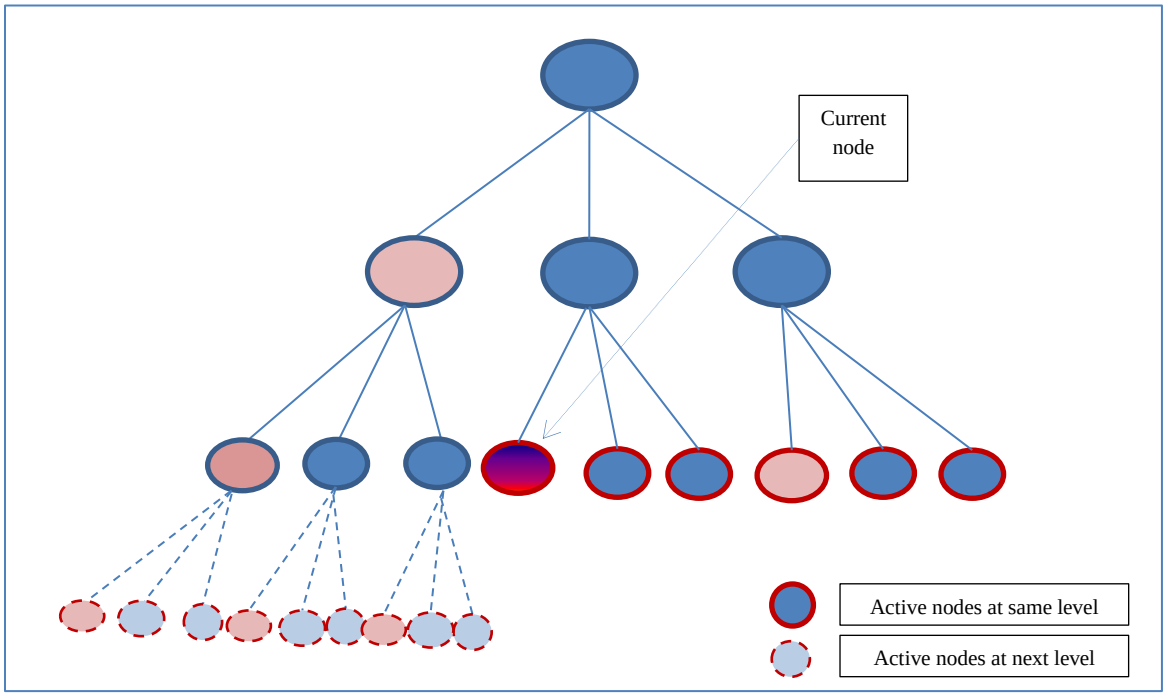


Figure 25: Snapshot of a sample tree showing current node and all active nodes with three objectives

3.3.7) Parallelizing the algorithm

As mentioned earlier, the proposed algorithm can be run in parallel, which is extremely useful from the computational perspective. There are two ways to run the algorithm in parallel, depending on whether the solution is aimed for the whole Pareto set or a well-dispersed approximation.

Different main objectives in parallel: The proposed algorithm, irrespective of which objective is selected as the main objective, is guaranteed to find the whole Pareto set as long as it is run until convergence, or when all the active nodes are fathomed. However, an early termination the proposed algorithm (e.g., prior to fathoming all nodes) would lead to different solution sets for different choices of the main objective. This parallelization approach is preferable for generating well-dispersed solution set that is an approximation of the whole Pareto set. This parallelization approach selects different objectives as the main objective and assign each main objective's run to a separate thread. When the solution process is terminated, the solutions from each thread are combined and filtered to obtain the final approximating set.

- **Siblings in parallel**: This is a method that can be resorted to when the proposed algorithm is used to obtain the whole Pareto set. After the root node, nodes at a level can be partitioned into groups and can be restarted on different machines, threads, etc., by carrying over the boundary information from root node. This is due to two reasons:
- The search space of each sibling is mutually exclusive (Proposition 3).
- Since the search spaces of the nodes do not change, domination relations among the nodes also do not change. That is, intermediate filtering is not a must for the algorithm;

thus, all solutions gathered from different parallel runs can be combined and filtered at the end or even some intermediate moments before termination.

This method of parallelization can be thought as switching from breadth-first to depth-first search, starting from an early stage of the algorithm (after the root node) and proceeding with breadth-first again on the separate reduced trees. For any type of parallelization, as long as all the solutions are combined to be filtered, all solutions can still be supplied with precise probability calculations for being non-dominated.

3.3.8) Handling alternative solutions

There might be more than one solution in decision space, which results in same objective values, namely alternative solutions. Since the algorithm is designed to run in the objective space, it cannot differentiate between two solutions that are identical in terms of objective vectors but distinct in terms of decision vectors. However, if this algorithm were used as a decision support tool, there are a couple of ways to reach alternative solutions, none of which are hard. If one is interested in alternative solutions for a certain objective value outcome, the same problem can be resolved by adding as many constraints as the number of objectives, each of which would fix the values of corresponding objective to the value of preferred solution and generate different results by inserting the algorithms that generates alternative solutions. In many optimization software packages, this property is implemented as a side tool that can be coded within the algorithm with a single command, based on a user inputted option.

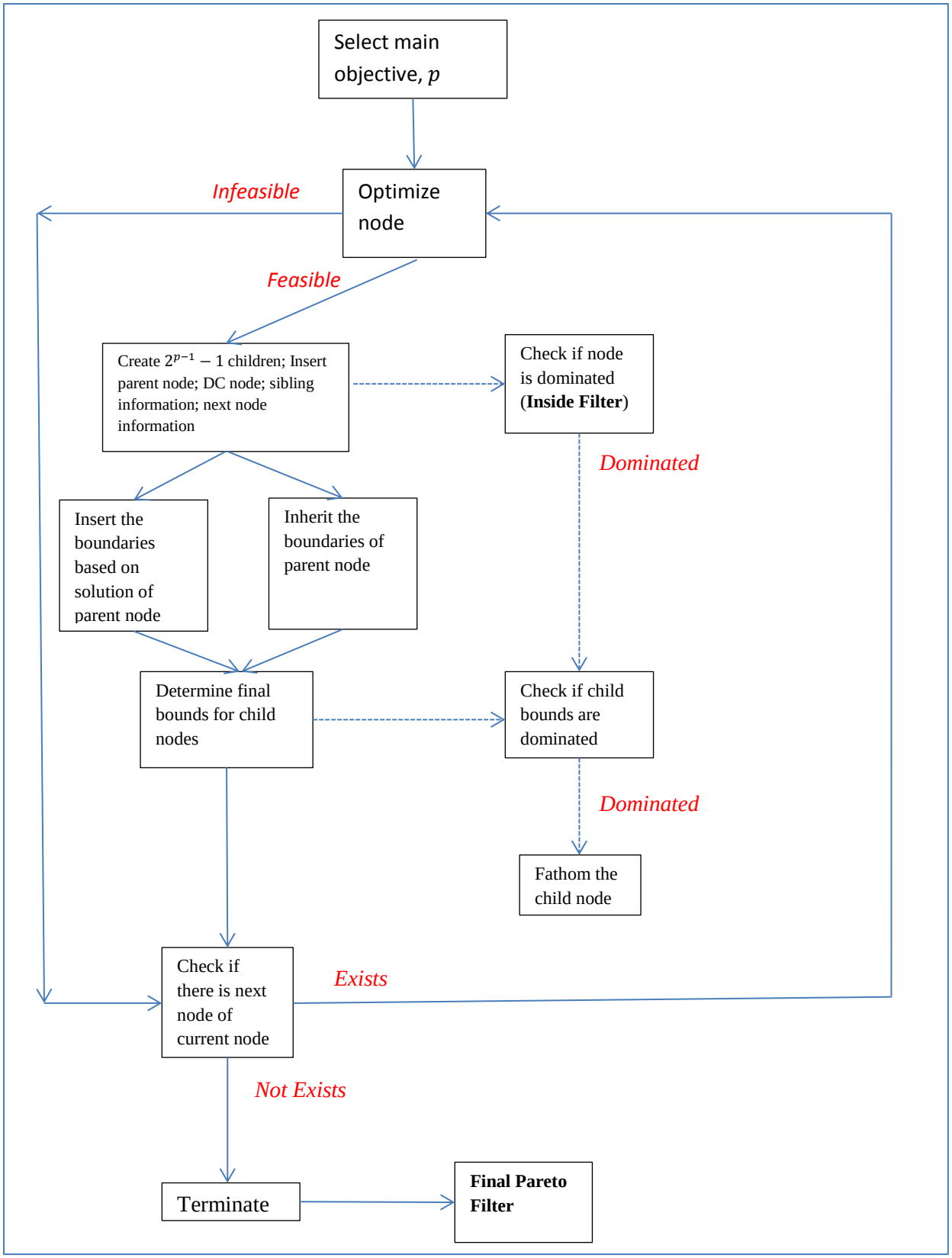


Figure 26: General flow chart of the breadth-first algorithm

STEP 0 (*Root node*): Set main objective, i. Solve model P3 with no bounds on objectives; i.e., set the $lb_j = -Infinity; ub_j = Infinity\ \forall j = \{1 \dots p\}$. Initialize Pareto Candidate Set, i.e., $E = \emptyset$.

- If solution is feasible, obtain the objective function values for corresponding solution, z^*; $E = E \cup z^*$; create child nodes of root node; *current node=first child node of root node.* If solution is infeasible STOP.

STEP 1(*Bound setting*): Compare the bounds of current node with the bounds of parent node and set the final bounds, lb_j and ub_j at P3.

STEP 2(*Optimization*): Solve P3. If feasible, obtain the objective function values for corresponding solution $z_j^*, \forall \mathrm{j} = 1 \dots \mathrm{p}$; $E = E \cup z^*$; continue with Step 3. If P3 is infeasible go to step 6.

STEP 3(*Child creation*): Create child nodes, make connections between them, enter level and relative information. Enter the bounds based on z^*; i.e., solution of current node.

STEP 4(*Domination check*): Check if z^* is dominated by its dominating list, S($current$); If dominated go to Step 5. If not dominated go to Step 6.

STEP 5(*Fathoming*): Compare the objective values of dominating solution with upper bounds of child nodes; if any of them dominated fathom the child node.

STEP 6(*Stopping condition*): Check if there exists next node. If exists *current node=next node;* return to Step 1. If next node does not exist STOP (or Final Pareto filtering)

Figure 27: Pseudo-code of the algorithm

3.4) Experimental Results

3.4.1) Sample Problems

For testing purposes of the proposed algorithm, three MOCO problem types have been used, the multi-objective knapsack problem (MOKP), the multi-objective shortest path problem (MOSP), and the spanning tree problem multi-objective (MOST). These problems are solved with a

different number of variables and objectives. Instances used for each case are exactly the same as the instances used by Lokman and Koksalan (2012) for the MOKP problem. These instances are used to verify the results as well. However, for the other two problems, since formulation of the problems differs slightly, despite the fact that inputs of instances are the same, the sample models are not exactly the same. The following formulations are used in order to model the relevant problems.

MOKP

Weights and profits of the items are generated as integers uniformly distributed between 10 and 100. The capacity of the knapsacks is taken as half of total weight:

$$\begin{gathered}\text{Max}\{z_1(x), z_2(x), \dots, z_p(x)\} \\ s.t. \\ \sum_{j=1}^{m} w_{ij}x_j \leq C_i \quad i = 1 \dots p \\ x_j \in \{1,0\}\end{gathered}$$

where

$z_1(x) = \sum_{j=1}^{m} p_{ij}x_j$

$p_{ij} = profit\ of\ item\ j\ for\ knapsack\ i$

$w_{ij} = weight\ of\ item\ j\ for\ knapsack\ i$

$C_i = Capacity\ of\ knapsack\ i$

$x_i = \begin{cases} 1 & if\ item\ j\ is\ selected\ to\ put\ in\ knapsacks \\ 0 & otherwise \end{cases}$

$C_i = \frac{\sum_{j=1}^{m} w_{ij}x_j}{2}$

$p = the\ number\ of\ knapsacks$

$m = number\ of\ items$

MOSP

Preliminary experiments for the MOSP problem showed that the number of non-dominated solutions is small when complete graph is used. Typically, there are several paths from source to sink with a relatively small number of arcs in a complete graph, and these dominate many other paths. In order to overcome this difficulty, special random graphs are generated, where source and sink nodes are defined as 1 and n, respectively. The details about how these graphs are generated can be found in Lokman and Koksalan (2012).

$$\text{Max}\{z_1(x_{ij}), z_2(x_{ij}), \dots, z_p(x_{ij})\}$$
$$s.t.$$

$$\sum_{j=1}^{n} x_{ij} - \sum_{j=1}^{n} x_{ji} = \begin{cases} 1 & i = 1 \\ -1 & i = k \\ 0 & otherwise \end{cases} \qquad \forall i, k$$

$$x_{ij} \in \{0,1\}$$

where

$$z_p(x_{ij}) = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{vij}x_{ij} \qquad v = 1, \dots, p$$

$$c_{vij} = Cost\ of\ arc\ between\ i\ anj\ for\ objective\ v$$

$$x_{ij} = \begin{cases} 1 & if\ arcbetween\ i\ and\ j\ is\ used \\ 0 & otherwise \end{cases}$$

MOST

Multi-objective spanning tree (MOST) is the last type of problem used for testing purposes. In order to have a mathematical program, the minimum spanning tree problem is formulated as a multi-commodity flow problem. Then it can be written as follows:

$$\text{Max}\{z_1(x_{ij}), z_2(x_{ij}), \dots, z_p(x_{ij})\}$$
$$s.t.$$
$$\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij} = 1$$

$$\sum_{j=1}^{n} f_{ij}^{k} - \sum_{j=1}^{n} f_{ji}^{k} = \begin{cases} 1 & i = 1 \\ -1 & i = k \\ 0 & otherwise \end{cases} \qquad \forall i = 1,2,\dots n \qquad k = 2,3,\dots,n$$

$$f_{ij}^{k} \le w_{ij} \quad i,j = 1,2,\dots,n \quad k = 2,3,\dots,n$$
$$w_{ij} + w_{ji} = x_{ij} \quad \forall i,j$$

$$x_{ij} \in \{0,1\}$$

where

$$z_p(x_{ij}) = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{vij} w_{ij} \qquad v = 1,\dots,p$$

$c_{vij} = Cost\ of\ arc\ between\ i\ anj\ for\ objective\ v$

$$x_{ij} = \begin{cases} 1 & if\ any\ flow\ exists\ from\ node\ i\ to\ node\ j \\ 0 & otherwise \end{cases}$$

w_{ij}: $total\ flow\ from\ node\ i\ to\ node\ j\ in\ units$

f_{ij}^{k}: $total\ flow\ of\ commodity\ k\ from\ node\ i\ to\ node\ j$

When there is a complete graph with n nodes, node 1 is defined as the supply node of n commodities and the remaining nodes as demand nodes, where each demand node has a demand for a different commodity of exactly one unit. Therefore, the difference of outflow and the inflow of commodity k will be equal to 1 for the demand node k whereas it will be equal to -1 for the supply node 1. All other nodes will be transshipment nodes for this commodity k. This model results with a spanning tree since using only one supplier will guarantee a connected graph. In addition, no cycles occur in this connected graph to minimize the cost.

3.4.2) Computational Results

The sample MOCO problems mentioned above are used to assess the performance of variations and attributes of the algorithm as well as its performance. For this purpose, the first set of experiments were conducted to observe the effect of fathoming. The second set of experiments were used compare the effect of simultaneous filtering versus final filtering. The third set of experiments were conducted to evaluate the non-dominance probability estimation. The last set of experiments showed the time performance of the algorithm. All versions of algorithm are coded in Microsoft Visual Studio 2010 using C++, and MIP solver of CPLEX 12.5.1 is used through concert technology. Two different machines are used to collect the results—the first PC has Intel® Core™i3 M390 with 2.67 GHz speed and 4(3.80 GB usable) RAM. The other PC has Intel® Core™ i5 2400 with 3.10 GHz speed and 16(15.9 usable) RAM.

3.4.2.1) Effect of fathoming

Fathoming is an important specialty of the proposed MOB&B algorithm as it is a single optimization case and has been explained in detail in 3.3.4*) Fathoming of Nodes*. In order to assess the effect of fathoming, two levels of fathoming are implemented into the algorithm. The first one uses a very basic fathoming structure with two types of information, the first piece of which is the DC sibling of each node. The second piece is the information inherited from the first level, e.g., relationships based on the children of root node. This information indicates whether a node is grandchild of the first DC node. Hence, it is checked if upper bounds of a child node are dominated by solution obtained from its DC sibling or the nodes at the same level those are inherited from the first DC child, if node itself is not a DC grandchild. The type modeled with this type of fathoming is called “Type 1 fathoming.” Then, in order to observe the full effect of fathoming, a structure that uses all the information on the tree to fathom the nodes is coded. That

is, this version requires carrying the information of all nodes that can dominate each node based on the domination relationships among nodes explained previously. This version of algorithm is called "Type 2 fathoming."

Three levels from each type of sample problems are chosen for this analysis. Each of the set contains five instances. The first four columns contain the averages for these instances. Since Type 2 fathoming requires considerable memory usage, the biggest problem sizes that allowed us to run all of their instances with Type 2 fathoming turned out to be relatively smaller sizes of problems within the sample problem set. Table 11 shows the results for each problem type:

Table 11: Impact of fathoming on three different problem sizes

		Average # of Pareto Points	Average # of models solved	Average CPU Time of run in sec	Average Filtering Time in sec	Type 2 reductions from Type 1		
						% of reduction for # of models solved	% of reduction for CPU time of run	% of reduction for filtering time
MOKP-25 nodes 3 objectives	Type1 fathoming	211.8	3235.2	933.25	0.2316	12%	36%	83%
	Type2 fathoming		2839.2	600.56	0.0394			
MOSP-100 nodes 3 objectives	Type1 fathoming	217.4	15952.2	42931.91	0.1973	7%	24%	52%
	Type2 fathoming		14905.4	32428.00	0.0941			
MOST-10 nodes 3 objectives	Type1 fathoming	761.6	12427.8	9831.13	0.4738	10%	21%	42%
	Type2 fathoming		11205	7735.26	0.2740			

Based on these results, the number of models and the total CPU time required is reduced significantly. Even in the least effective case time that is required to traverse the tree reduces by more than 20%. Fathoming affects both filtering time and total tree traversal time. Filtering time

is affected due to the decreasing number of solutions that result from the reduced size of the tree. Hence, it is worth expending the effort to keep information necessary information for fathoming. Based on this result, we used full fathoming for the rest of the analysis.

3.4.2.2) Simultaneous Filtering vs. Final Filtering

As it has been explained in 3.3.5*) Pareto Filtering Strategies*, the proposed algorithm requires Pareto filtering, and one might choose to perform this filtering at different points of the tree traversal. Simultaneous filtering was previously proposed as the extreme level of this procedure, and this requires comparing the value of a node with the solutions obtained at the nodes that can dominate it immediately after it is obtained. It has been noted that this is a memory dependent procedure, but it is advantageous in the sense that dominated nodes from earlier levels of the tree by newly generated nodes are eliminated as soon as the dominating solution is generated, which saves time making unnecessary comparisons that are performed at the later filtering stage. The other extreme of this filtering mechanism is final filtering, and this is performed when tree traversal is finished. In order to assess the effect of the filtering on the total CPU time, two versions of the algorithm are examined with two different filtering schemes. The first one is close to the simultaneous filtering, or filtering the solution set after the completion of each level. All solutions obtained at that level are compared with the solutions obtained previously in order to have a filtered set at the end of each level. The version of the algorithm with this type of filtering scheme is called “Type 2 filtering.” The version with final filtering is called “Type 1 filtering.” Table 11 shows the effect of filtering. The same instances that are used in the fathoming analysis are used for this analysis as well. Both of the versions contain full fathoming schemes as the fathoming level.

Table 12: Impact of filtering for three different problem sizes

		Average # of Pareto Points	**Average** # of models solved	Average CPU Time of run in sec	Average Filtering Time in sec	Filtering effect of Type 2	
						% of reduction for CPU time of run	% of reduction for filtering time
Knapsack Problem with 25 nodes 3 objectives	Type1 filtering	211.8	2839.2	600.56	0.0394	3.10%	28.93%
	Type2 filtering			581.97	0.0280		
Shortest Path Problem 100 nodes 3 objectives	Type1 filtering	217.4	14905.4	32428.00	0.9412	2.83%	41.88%
	Type2 filtering			31508.96	0.5470		
Spanning Tree Problem 10 nodes 3 objectives	Type1 filtering	761.6	11205	7735.26	0.2741	3.94%	90.11%
	Type2 filtering			7430.12	0.0271		

Simultaneous filtering seems to have a significant effect on filtering time, although it does not affect the total running time with that strength. Small reductions in the total runtime stems from the difference created on fathoming checks by eliminating the dominated nodes from the dominating node lists of each node which are used to for fathoming. Before deciding upon which type of filtering to implement, one should make a tradeoff analysis between the increased speed and additional memory required to carry over all the required information.

3.4.2.3) **Probability measure of being non-dominated**

The proposed algorithm can be used as an approximation algorithm together with the probability calculation presented in 3.3.6*) Probability of being non-dominated for solution set*. For the testing purposes of this property, we have calculated a single probability that represents the average staying non-dominated probability of all solutions obtained after a certain number of

models are solved. The change of this measure has been presented in the following figures for three different problem types.

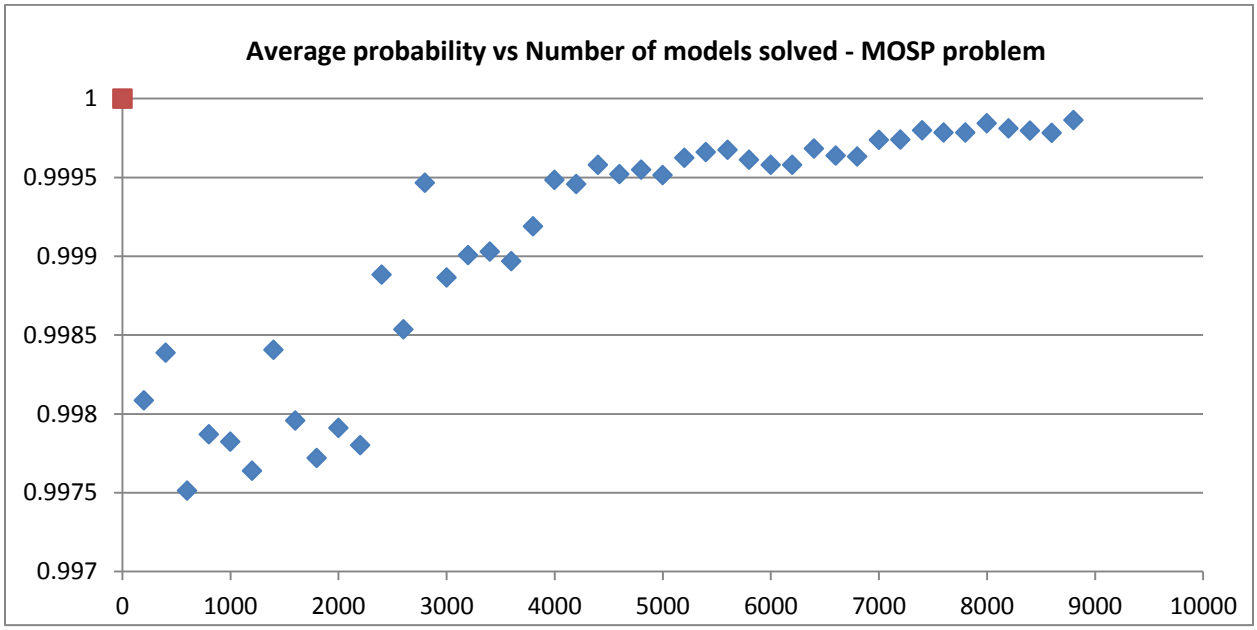


Figure 28: Change of average non-domination probability of solution set with respect to number of models solved for a MOSP instance with 100 nodes and 3 objectives

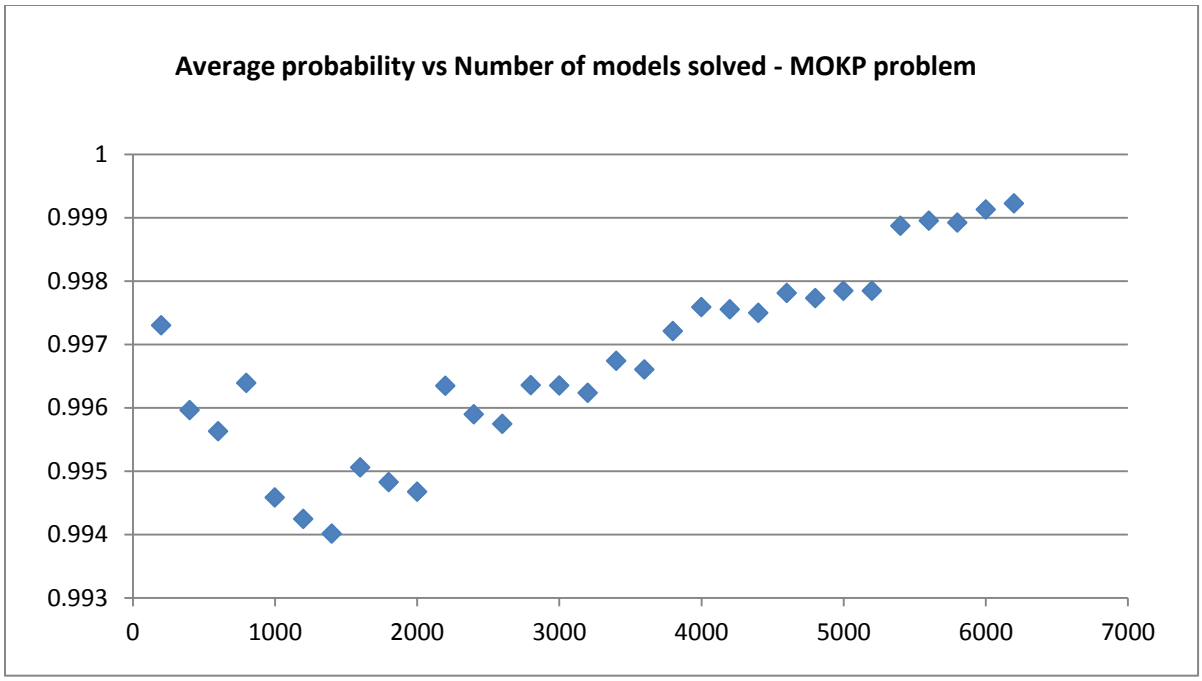


Figure 29: Change of average non-domination probability of solution set with respect to number of models solved for a MOKP instance with 50 items and 3 objectives

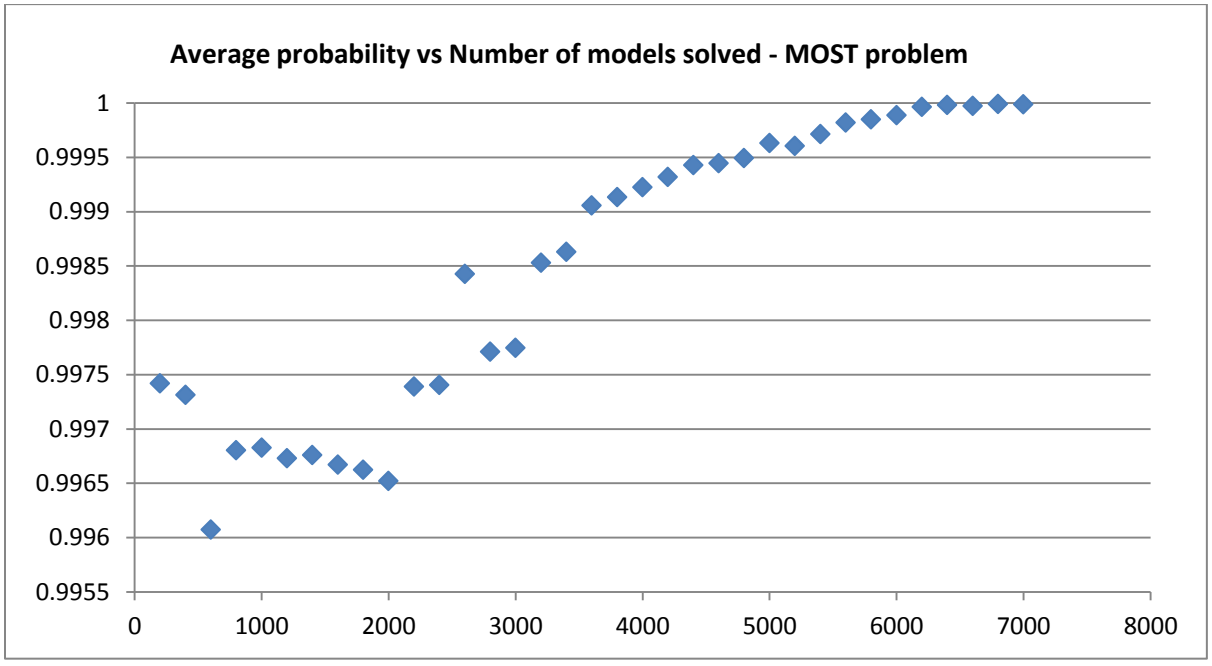


Figure 30: Change of average non-domination probability of solution set with respect to number of models solved for a MOST instance with 10 nodes and 3 objectives

As shown in the relevant calculations, one can also trace the non-domination probability calculated for the individual solutions. For instance, the following figure shows how the non-domination probability of a solution changes at the 187th node of the MOB&B tree belonging to the MOST problem instance used to generate the average non-domination probability above. Although the whole tree algorithm solved more than 8000 models, the non-domination probability became "1" after the 2000th model solution.

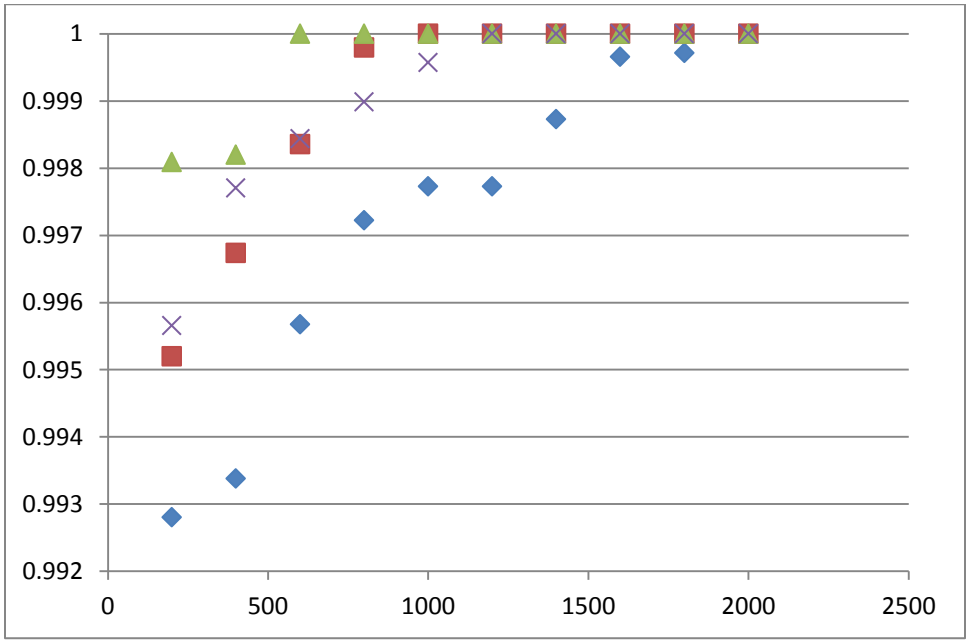


Figure 31: Change of non-domination probability for four different solutions with respect to number of models solved at the MOST instance with 100 nodes and 3 objectives

3.4.2.4) Time and Representativeness Performance of proposed algorithm

Further tests are performed on the whole test set used by Lokman and Koksalan (2012) in order to make a complete assessment regarding the time performance of the proposed algorithm. All versions used during these sets have full fathoming and final filtering property. The way the fathoming scheme is implemented requires memory usage, and this did not allow us to run all the models in the standard breadth-first approach. Hence, it created the necessity for two versions of

the algorithm. The first one is for the small sizes and with standard breadth-first approach, which will be referred as "standard traversal." The second version's aim is to decrease memory requirement, hence reducing the size of the tree that needs to be traversed is the way resorted and the whole tree is divided into smaller portions like it should be done when parallel running property needs to be used. In order to obtain the smaller portions, nodes at a certain level can be used to initiate the smaller portions of the tree. The results are summarized in Table 12 and indicate whether it is generated by the standard version of algorithm or by the partitioned version.

Table 13: Time performance of proposed approach in comparison with Lokman and Koksalan (2012)

	# of Pareto Points per instance	# of models solved per instance	Total runtime in sec. per instance	Average results per Pareto point for MOB&B		Average results per Pareto point for Lokman&Koksalan (2012)	
				models solved	CPU time	models solved	CPU time
Knapsack							
25 item with 3 objectives	211.8	2839.2	600.6	13.41	2.84	2.21	0.16
50 item with 3 objectives	570.2	11233	15072	19.70	26.43	2.17	0.41
100 item with 3 objectives	6786.2	97217	242172.3	14.33	35.69	1.86	2.91
25 item with 4 objectives[3]	425.2	42313	6289.34	99.51	14.79	8.46	0.8
Shortest Path							
25 nodes with 3 objectives	50.4	816	131.65	16.19	2.61	2.24	0.07
50 nodes with 3 objectives	109.2	3905.6	1893.46	35.77	17.34	2.26	0.19
100 nodes with 3 objectives	217.40	14905.4	32428.94	68.56	10.16	2.15	0.46
150 nodes with 3 objectives	649.5	22768.5	41245.3	32.99	62.8	2.09	0.75
25 nodes with 4 objectives	3726.4	12148.8	2682.67	3.26	0.72	7.65	0.25
50 nodes with 4 objectives[4]	2110	42341	29573.88	20.07	14.02	9.49	1.07
Spanning Tree							
25 nodes with 3 objectives	761.6	11205	7738.002	14.71	10.16	2.11	0.39

[3] Partitioned version of algorithm is used as opposed to standard breadth first
[4] Partitioned version of algorithm is used as opposed to standard breadth first

The CPU time performance of single runs cannot beat the CPU time performance of the benchmark algorithm. However, it should be noted that parallel running is an option for MOB&B, but not for the benchmark algorithm. As mentioned previously, MOB&B is implemented in a breadth-first manner in order to achieve a more representative set throughout the run. The following figures show how quality measures of both algorithms change on each Pareto point generated for a knapsack instance. Coverage measure used in this analysis is the measure of Wu and Azarm (2001), as presented in 2.5.1) Coverage Measure.

Recalling that larger coverage and uniformity values are preferable, it can be observed that MOB&B always evolves with better coverage; furthermore, the proposed algorithm is not worse than the benchmark algorithm in terms of uniformity quality.

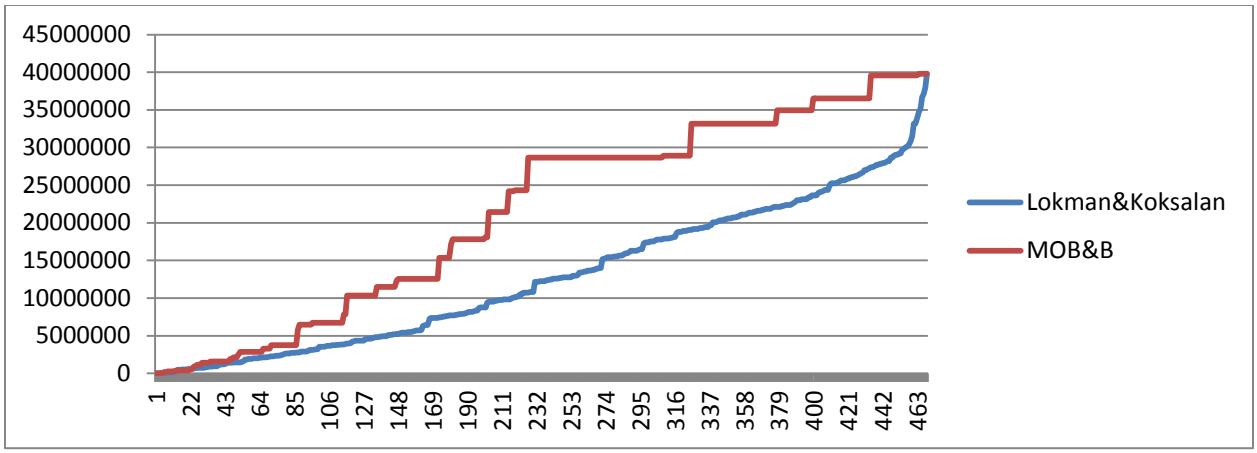


Figure 32: Change of coverage measure for Lokman and Koksalan (2012) and MOB&B at each Pareto point generated for a knapsack problem with 25 items.

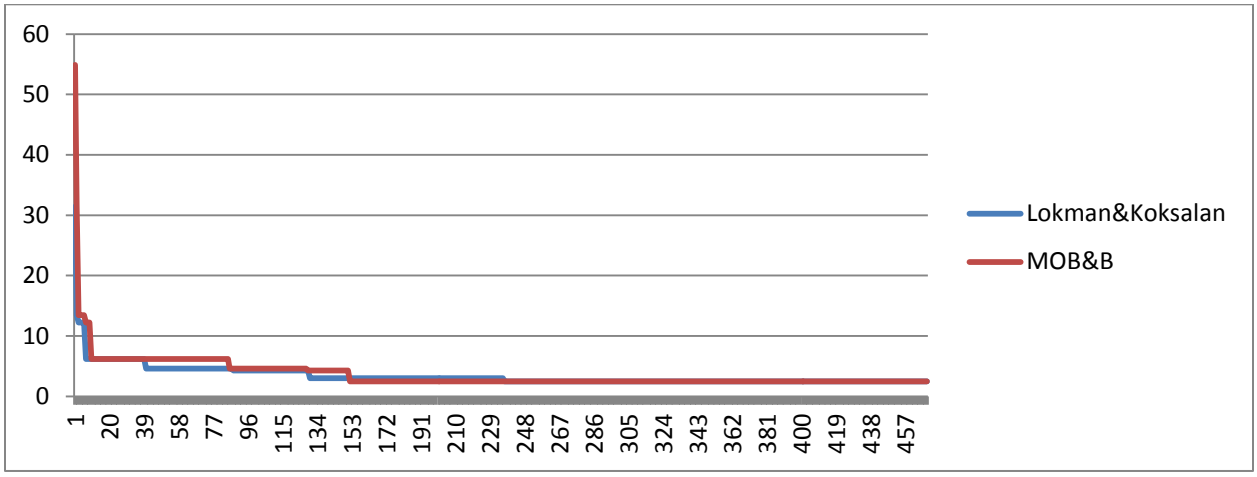


Figure 33: Change of uniformity measure for Lokman and Koksalan (2012) and MOB&B at each Pareto point generated for a knapsack problem with 25 items.

CHAPTER IV: CONCLUSION AND FURTHER STUDIES

4.1) Summary of Contributions

In this dissertation we proposed two different algorithms, each of which aims to generate representative points that can be used while approximating the actual Pareto front. Both of the algorithms are exact methods that rely on solutions of some mathematical models. The first algorithm is proposed for general MOP problems, which means it can be used both for MOLP and MOIP problems. Experimental results and other analysis support that the proposed algorithm is a practical and fast algorithm compared to the existing exact algorithms in the literature. Furthermore, in terms of quality measures, e.g., coverage and uniformity of representative points, the proposed algorithm is compatible with the best algorithm benchmark work.

The second algorithm that has been proposed is for MOIP problems, and it adapts the existing B&B idea in a systematic way to branch on Pareto candidates on the (objective) criteria space. Many properties of the algorithm have been shown with proves, figures and experimental studies, such as fathoming and filtering. Fathoming due to integer bounds has been improved in particular, with the relations explained between the nodes of the B&B tree. Aside from this, because of these existing domination relations, Pareto filtering has become the comparison of solutions that have the potential to dominate each other rather than making a simple pairwise comparison of all solutions in the candidate set. In addition to the standard features adapted from standard B&B, new features particular to the MOIP context have been introduced, such as precise probability of non-domination, convenience of running in parallel. Although time performance of algorithms does not seem to be better than the time performance of the benchmark algorithm from the point of generating whole Pareto front, the MOB&B approach generates a more representative candidate set than does the benchmark algorithm in case in

which they are both used to generate approximations of the Pareto front. This is due to the difference between how the feasible search space is traversed; that is, MOB&B considers whole feasible region as long as breadth-first type of traversing chosen, while benchmark approaches does this starting from a corner of the same region and proceeding by relying on the previously found points. This conclusion is supported by graphs that represent the relative change of quality measures on a sample problem. Finally, the proposed algorithm has the flexibility to change based on the conditional requirements. As an example, switching from breadth-first to depth-first at a certain level and continue with breath first at each smaller tree is tried on some test sets.

4.2) Further Studies

Both of the algorithms proposed in this thesis have the potential to be used as interactive approaches. In other words, they can be used to focus on certain parts of the feasible space rather than on the whole feasible space based on the preferences of the DM, if he or she does not intend to come up with the whole Pareto front. As mentioned in the last section of second chapter, the first proposed algorithm can be further adapted for MOIP problems in order to further speed up the algorithm.

The second algorithm is implemented to keep the required information in memory. However, with some sacrifice from running time, the algorithm can also be implemented with zero memory requirement thorough the usage of binary input/output files. Pareto filtering is a key component of this algorithm, and different variations for this component have been proposed. Final filtering has been used in most of the test cases, which compares whole solutions obtained throughout the all tree traversal. However, this procedure can be improved by keeping the domination relations between nodes until the end; this way, some time can be saved by avoiding some of the pairwise comparisons. The results indicate that fathoming has a significant

effect on the performance of the algorithm. Therefore, any effort that can enrich the fathoming rules will help enhance the performance of the algorithm. The MOB&B algorithm proposed in this study relies on integer solvers at this stage. It might worth investigating if this structure can be enriched by existing multi-objective simplex methods along with some rounding procedures in order to come up with the Pareto solutions, or at least to derive some upper bound sets to be used for fathoming.

APPENDIX

APPENDIX A: STATISTIC TABLES FOR THE RESULTS OF FIRST ALGORITHM

Table 14: Statistics for the uniformity analysis in Table 4

Uniformity				
		5 const	10 const	30 const
2 obj	min	0.000527	0.001425	0.001272
	max	0.019332	0.026639	0.025132
	stdev	0.006619	0.00975	0.008628
3 obj	min	0.001234	0.000148	0.000248
	max	0.038846	0.034031	0.022475
	stdev	0.010818	0.011909	0.008201
4 obj	min	0.000854	0.004926	0.000104
	max	0.037458	0.032464	0.023529
	stdev	0.015815	0.009305	0.007747
5 obj	min	0.000467	0.001259	0.0025
	max	0.035174	0.038968	0.0334
	stdev	0.013327	0.011529	0.010214

Table 15: Statistics for coverage error for the analysis in Table 3

Coverage				
		5 const	10 const	30 const
2 obj	min	2.74E-06	0.000412	6.38E-06
	max	0.123654	0.130514	0.14589
	stdev	0.042285	0.04182	0.056533
3 obj	min	0.165267	0.036445	0.02746
	max	0.466684	0.412547	0.324093
	stdev	0.081334	0.114158	0.098046
4 obj	min	0.270381	0.163809	0.081797
	max	0.595553	0.561461	0.306913
	stdev	0.108219	0.130771	0.08063
5 obj	min	0.407538	0.336629	0.0534
	max	0.99999	0.999969	0.479
	stdev	0.253937	0.212042	0.178496

Table 16: Statistics for the results presented in Table 5

Statistics of CPU time Set 1					
		2 obj	3 obj	4 obj	5 obj
5 const.	max	1.455	0.738	2.121	1.329
	min	0.218	0.115	0.799	0.735
	stdev	0.36752	0.045984	0.414595	0.194648
10 const.	max	0.895	1.504	3.082	1.36
	min	0.145	0	1.987	0.778
	stdev	0.254129	0.430887	0.381781	0.203522
30 const	max	3.384	1.508	5.381	3.278
	min	0.493	0.235	1.831	2.676
	stdev	0.908195	0.414539	1.035147	0.198275

Table 17: Statistics for the results in Table 6

		2 obj		4 obj		6 obj	
		SmallSize	Bigsize	SmallSize	Bigsize	SmallSize	Bigsize
	max	0.022	0.019467	0.048608	0.057088	0.020788	0.162369
	min	0.01279	0.008527	0.033148	0.038215	0.017904	0.09009
	stdev	0.002884	0.001346	0.005025	0.006823	0.000961	0.026089
		SmallSize	Bigsize	SmallSize	Bigsize	SmallSize	Bigsize
	max	0.02111	0.0232	0.078375	0.094538	0.025131	0.180755
	min	0.01724	0.016547	0.05445	0.064634	0.019197	0.138518
	stdev	0.001314	0.002037	0.007262	0.007661	0.001867	0.012296

APPENDIX B: STATISTIC TABLE FOR THE RESULTS OF SECOND ALGORITHM

Table 18: Statistics for the results presented in Table 13

	Total CPU Time			Total Models Solved		
	Min.	Max.	St.Dev.	Min.	Max.	St.Dev.
Knapsack						
25 item with 3 objectives	151.9	952.0	327.2	972.0	5162.0	1581.4
50 item with 3 objectives	3466.0	34021.0	13263.7	4562.0	23984.0	8774.1
100 item with 3 objectives	15836.0	412381.0	170426.3	63497.0	123541.0	30542.9
25 item with 4 objectives	2575.4	12208.0	4162.8	29473.0	67211.0	17293.9
Shortest Path						
25 nodes with 3 objectives	19.3	213.7	85.8	158.0	1255.0	486.9
50 nodes with 3 objectives	688.4	3606.2	1147.4	1750.0	6702.0	2109.7
100 nodes with 3 objectives	19573.0	43682.0	10939.6	10327.0	19792.0	3949.9
150 nodes with 3 objectives	30751.0	59826.0	13033.1	15298.0	31885.0	7074.9
25 nodes with 4 objectives	546.2	6244.2	2368.3	4038.0	21711.0	7035.0
50 nodes with 4 objectives	19753.0	39614.0	9745.2	24378.0	59326.0	16026.6
Spanning Tree						
25 nodes with 3 objectives	2137.5	17516.8	5912.9	5043.0	20123.0	6037.0

REFERENCES

A.P. Wierzbicki. (1980). *The use of reference objectives in multiobjective optimization*. Springer Berlin Heidelberg.

Abbas, M., Chergui, M. E., & Mehdi, M. A. (2012). Efficient cuts for generating the non–dominated vectors for Multiple Objective Integer Linear Programming. *International Journal of Mathematics in Operational Research, 4*(3), 302-316.

Benson, H. (1998). An Outer Approximation Algorithm for Generating All Efficient Extreme Points in the Outcome Set of a Multiple Objective Linear Programming Problem. *Journal of Global Optimization, 13*, 1–24.

Benson, H., & Sayin, S. (1994). Finding global representations of the efficient set in multiple objective mathematical programming (Discussion paper). Department of Decision and Information Sciences University of Florida.

Chinchuluun, A., & Pardalos, P. (2007). A survey of recent developments in multiobjective optimization. *Annals of Operations Research, 154*, 29-50.

Dhaenens, C., Lemsre, J., & Talbi, A. (2010). K-PPM: A new exact method to solve multi-objective to solve combinatorial optimization problems. *European Journal of Operational Research, 200*(1), 45-53.

Ecker, Z., & Kouada, I. (1978). Finding all efficient extreme points for multi-objective linear programs Math Programming. *14*, 249-261.

Ehrgott, M. ((2006)). A discussion of scalarization techniques for multiple objective integer programming. *Annals of Operations Research, 147*(1), 343-360.

Ehrgott, M. (2005). *Multicriteria optimization.* Springer.

Ehrgott, M., & Gandibleux, X. (2007). Bound sets for biobjective combinatorial optimization problems. *Computers & Operations Research, 34*(9), 2674-2694.

Ehrgott, M., & Gandibleux, X. (2008). Hybrid Metaheuristics for Multi-objective Combinatorial Optimization. *Studies in Computational Intelligence (SCI), 114*, 221-259.

Ehrgott, M., & Gandibleux., X. (2000). A survey and annotated bibliography of multiobjective combinatorial optimization. *OR Spectrum, 22*(4), 425-460.

Faulkenberg, S., & Wiecek, M. (2012). Generating equidistant representations in biobjective programming. *Computational Optimization and Applications, 51*(3), 1173-1210.

Faulkenberg, S., & Wiecek, M. M. (2010). On the quality of discrete representations in multiple objective programming. *Optimization and Engineering, 11*(3), 423-440.

Figueira, J. R., Liefooghe, A., Talbi, E. G., & Wierzbicki, A. P. (2010). A parallel multiple reference point approach for multi-objective optimization. *European Journal of Operational Research, 205*(2), 390-400.

Fleischer, M. (2003). The measure of Pareto optima applications to multi-objective metaheuristics. In *Evolutionary multi-criterion optimization* (pp. 519-533). Berlin Heidelberg: Springer .

Fu, Y., & Diwekar, U. M. (2004). An efficient sampling approach to multiobjective optimization. *Annals of Operations Research, 132*(1-4), 109-134.

Geoffrion, A. (1968). Proper efficiency and the theory of vector maximization. *Journal of Mathematical Analysis and Applications, 22*(3), 618–30.

Gülseren, K., & Yucaoglu, E. (1983). An algorithm for multiobjective zero-one linear programming. *Management Science, 29*(12), 1444-1453.

Gunawan, S., Farhang-Mehr, A., & Azarm, S. (2003). Multi-level multi-objective genetic algorithm using entropy to preserve diversity. In *Evolutionary Multi-Criterion Optimization* (pp. 148-161). Berlin Heidelberg: Springer.

Hanne, T. (2000). Global multiobjective optimization using evolutionary algorithms. *Journal of Heuristics, 6*(3), 347-360.

Hansen, M., & Jaszkiewicz, A. (1998). *Evaluating the quality of approximations to the non-dominated set.* Technical Universityof Denmark: IMM, Department of Mathematical Modelling.

Isermann, H. (1977). The enumeration of the set of all efficient solutions for a linear multiple objective program. *Operational Research Quarterly, 28*(3), 711–725.

Jahn, J., & Merkel, A. (1992). Reference-Point Approximation Method for the Solution of Bicriterial Nonlinear Optimization Problems. *Journal of Optimization Theory and Applications, 78*, 87-103.

Karasakal, E., & Koksalan, M. (2009). Generating a representative subset of the nondominated frontier in multiple criteria decision making. *Operations research, 57*(1), 187-199.

Keeney, R., & Raiffa, H. (1976). *Decision analysis with multiple conflicting objectives.* New York: Wiley& Sons.

Kiziltan, G., & Yucaoglu, E. (1983). An algorithm for multiobjective zero-one linear programming. *Management Science, 29*(12), 1444-1453.

Klamroth, K., & Tind, J. (2007). Constrained Optimization Using Multiple Objective Programming. *Journal of Global Optimization, 37*(3), 325-355.

Konak, A., Coit, D. W., & Smith., A. E. (2006). Multi-objective optimization using genetic algorithms: A tutorial. *Reliability Engineering & System Safety, 91*(9), 992-1007.

Laumanns, M., Thiele, L., & Zitzler, E. (2005). An adaptive scheme to generate the pareto front based on the epsilon-constraint method. *European Journal of Operations Research, 169*, 932-942.

Laumanns, M., Thiele, L., Deb, K., & Zitzler, E. (2002). Combining convergence and diversity in evolutionary multiobjective optimization. *Evolutionary computation, 10*(3), 263-282.

Leiserson, C., Rivest, R., Stein, C., & Cormen, T. (2001). *Introduction to algorithms.* The MIT press.

Lemesre, J., Dhaenens, C., & Talbi, E. (2007). An exact parallel method for a bi-objective permutation flowshop problem. *European Journal of Operational Research, 177*(3), 1641-1655.

Lemesre, J., Dhaenens, C., & Talbi, E. (2007). Parallel partitioning method (PPM): A new exact method to solve bi-objective problems. *Computers & Operations Research, 34*(8), 2450-2462.

Lokman, B., & Koksalan, M. (2012). Finding all nondominated points of multi-objective integer programs. *Journal of Global Optimization*, 1-19.

Marcotte, O., & Soland, R. M. (1986). An interactive branch-and-bound algorithm for multiple criteria optimization. *Management Science, 32*(1), 61-75.

Marler, R., & Arora, J. (2004). Survey of Multiobjective optimization methods for engineering. *Struct Multidisc Optim, 26*, 369-395.

Mattson, C., Mullur, A., & Messac, A. (2004). Smart Pareto filter: Obtaining a minimal representation of multiobjective design space. *Engineering Optimization, 36*(6), 721-740.

Mavrotas, G. (2009). Effective implementation of the e-constraint method in multi-objective mathematical programming problems. *Appl. Math. Comput., 213*, 455-465.

Mavrotas, G., & Diakoulaki, D. (2005). Multi-criteria branch and bound: A vector maximization algorithm for mixed 0-1 multiple objective linear programming. *Applied Math and Computing, 171*, 53–71.

Mavrotas, G., & Diakoulaki., D. (1998). A branch and bound algorithm for mixed zero-one multiple objective linear programming. *European Journal of Operational Research, 107*(3), 530-541.

Odile, M., & Soland, R. M. (1986). An interactive branch-and-bound algorithm for multiple criteria optimization. *Management Science, 32*(1), 61-75.

Ozlen, M., & Azizoglu, M. (2009). Multi-objective integer programming: A general approach for generating all non-dominated solutions. *European Journal of Operations Research, 199*, 25-35.

Polak, E., & Payne, A. N. (1976). *On multicriteria optimization. In Directions in Large-Scale Systems .* US: Springer.

Przybylski, A., Gandibleux, X., & Ehrgott, M. (2010a). A Recursive algorithm for finding all extreme points on the outcome set of a multiobjective integer program. *INFORMS Journal on computing, 22*(3), 371-386.

Przybylski, A., Gandibleux, X., & Ehrgott, M. (2010b). A two phase method for multi-objective integer programming and its application to the assignment problem with three objectives. *Discrete Optimization, 7*, 149-165.

R.E.Steuer. (1986). *Multiple criteria optimization: Theory, computation, and application.* New York, NJ: John Wiley & Sons.

Ramos, R. M., Alonso, S., Sicilia, J., & Gonzalez, C. (1998). The problem of the optimal biobjective spanning tree. *European Journal of Operational Research, 111*, 617-628.

Rockafellar, R. T. (1970). *Convex Analysis.* New York, NJ: Princeton University Press.

Ruiz, F., Luque, M., & Cabell, J. (2009). A Classification of the weighting schemes in reference point procedures for multiobjective programming. *Journal of Operational Research Society, 60*, 544{553.

Ruzika, S., & Wiecek, M. M. (2005). Approximation methods in multiobjective programming. *Journal of optimization theory and applications, 126*(3), 473-501.

Sarah, S., & Radzik, T. (2008). Computing all efficient solutions of the biobjective minimum spanning tree problem. *Computers & Operations Research, 35*(1), 198-211.

Sayin, S. (1996). An algorithm based on facial decomposition for finding the efficient set in multiple objective linear programming. *Operations Research Letters, 19*, 87-97.

Sayin, S. (2000). Measuring the quality of discrete representations of effcient sets in multi-objective mathematical programming: a weighted min-max two stage optimization approach and a bicriteria algorithm. *Management Science, 26*, 369-395.

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

Steiner, S., & Radzik, T. (2008). Computing all efficient solutions of the biobjective minimum spanning tree problem. *Computers & Operations Research, 35*(1), 198-211.

Steuer, R. E. (1994). Random problem generation and the computation of efficient extreme points in multiple objective linear programming. *Computational Optimization and Applications, 3*(4), 333-347.

Steuer, R., & Choo, E. (1983). An interactive weighted Tchebycheff procedure for multiple objective programming. *Mathematical Programming, 26*(3), 326-344.

Steuer, R., Silverman, J., & Whisman, A. (1993). A Combined Tchebycheff aspiration criterion vector interactive multiobjective programming procedure. *Management Science, 39*(10), 1255-1260.

Sylva, J., & Crema, A. (2004). A method for finding the set of nondominated vectors for multiobjective linear integer linear problems. *European Journal of Operations Research, 158*, 46-55.

Tenfelde-Podehl, D. (2003). *A recursive algorithm for multiobjective combinatorial optimizaztion problems with q criteria.* Graz, Austria: Institut für Mathematik, Technische Universität Graz.

Vincent, T., Seipp, F., Ruzika, S., Przybylski, A., & Gandibleux, X. (2013). Multiple objective branch and bound for mixed 0-1 linear programming: Corrections and improvements for the biobjective case. *Computers & Operations Research, 40*, 498-509.

Visee, M., Teghem, J., Pirlot, M., & Ulungu, E. L. (1998). Two phases method and branch and bound procedures to solve the bi-objective knapsack problem. *Journal of Global Optimization, 12*, 139-155.

Wierzbicki, A. (A-2361). The use of reference objectives in multiobjective optimization-Theoretical implications and practical experiences. 1980.

Wierzbicki, A., Makowski, M., & Wessels, J. (2000). *Model-based decision support methodology with environmental applications* (Vol. 475). Dordrecht: Kluwer Academic Publishers.

Wu, J., & Azarm, S. (2001). Metrics for quality assessment of multi-objective design optimization solution set. *Transactions of the ASME, 123*, 18-25.

Yu, P., & Zeleny, M. (1975). The Set of all nondominated solutions in linear cases and multicriteria cplex method. *J.Math Anal Appl., 49*, 430-468.

Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C., & Fonseca, V. (2003). Performance assessment of multiobjective optimizers: an analysis and review. *IEEE Transactions on evolutionary computation, 11*(3), 423-440.

ABSTRACT

EXACT AND REPRESENTATIVE ALGORITHMS FOR MULTI-OBJECTIVE OPTIMIZATION

by

OZGU TURGUT

May 2014

Advisor: Professor Alper E. Murat

Major: Industrial and Systems Engineering

Degree: Doctor of Philosophy

In most real-life problems, decision alternatives are evaluated with multiple conflicting criteria. The entire set of non-dominated solutions for practical problems is impossible to obtain with reasonable computational effort. The decision makers (DM) generally needs only a representative set of solutions from the actual Pareto front. The first algorithm we present aims to efficiently generate a well-dispersed, non-dominated solution set that is representative of the Pareto front and that can be used for general multi-objective optimization problem. The algorithm first partitions the criteria space into grids to generate reference points, and then searches for non-dominated solutions in each grid. This grid-based search utilizes an achievement scalarization function and guarantees Pareto optimality. The results of our experimental results demonstrate that the proposed method is very competitive with other algorithms in the literature when representativeness quality is considered. The algorithm is advantageous from the computational efficiency point of view.

Although generating the whole Pareto front does not seem practical for many real-life cases, it is sometimes required for verification purposes or in cases where the DM wants to run his or her decision-making structures on the full set of Pareto solutions. For this purpose, we present another novel algorithm. This algorithm attempts to adapt the standard branch and bound

approach to the multi-objective context by proposing to branch on solution points in objective space. This algorithm is proposed for multi-objective integer optimization problems. The various properties of branch and bound concept have been investigated and are explained within the multi-objective optimization context. These include fathoming, node selection, heuristics, and some multi-objective optimization specific concepts such as filtering, non-domination probability and parallel running. This approach has the potential to be used both for full Pareto generation or as an approximation approach, as has been shown with experimental studies.

AUTOBIOGRAHICAL STATEMENT

Ozgu Turgut was born in Kayseri, Turkey on November 13, 1981, the daughter of Billur Dincer and Ramazan Turgut. She received the Bachelor of Science in 2005 from Bogazici Univeristy, Istanbul/Turkey. When she was an undergraduate, she worked as an intern for the logistics and supply chain planning department of DSM Nutritional Products Co. After receiving a B.S. degree, she worked at Yeditepe Univeristy as a Research and Teaching Assistant. She also prepared projects for consulting companies and worked as an outsourced expert of various subjects such as simulation and scheduling. She received a Master of Science degree in Systems Engineering from Yeditepe University, Istanbul, Turkey, in 2007.

Following this, she continued her studies to earn a Ph.D. in Industrial Engineering at Bogazici University. After taking courses for three terms, she decided to finish her degree abroad and resumed her studies at Wayne State University. She has been working for LLamasoft Co. since May 2012 at an advanced research department as an OR scientist; she plans to continue working there upon graduation.

During her studies at WSU, she gave a number of technical presentations at INFORMS, CASC, the Midwest Optimization Seminar and several other conferences. Her articles have been published in *Expert Systems with Applications*, *Procedia Computer Science*, and *Wind Energy*. She presently has two other articles submitted and under review.