Multi-Objective Linear Programming Revisited: Exact and Approximate Approaches

Paschal Bisong Nyiam

A thesis submitted for the degree of

Doctor of Philosophy (PhD)

Department of Mathematical Sciences

University of Essex

August, 2018.

Dedication

To my wife Blessing, and Children: Paschaline, Favour and Paschal Jnr.

Abstract

Most real world decision making problems involve more than one objective function and can be formulated as multiple objective linear programming (MOLP) problems. Some exact methods have proven to be effective on small and medium scale MOLP instances. The thesis considers prominent exact methods, implements and modifies some of them and compares them on existing test problems. Heuristics or approximate methods on the other hand, have been commonly applied to nonlinear and discrete multi-objective optimisation problems, and not so much to MOLP. Given the complexity of MOLP, it is worth investigating heuristics as a solution approach. This has also been considered here.

The thesis presents an extensive state-of-the-art survey of MOLP algorithms developed over the past five decades and modifies/extends some of them to generate the set of all nondominated points of the problem. It then compares these extended variants with others such as Benson's algorithm, the affine scaling interior-point MOLP algorithm and the recently introduced parametric simplex algorithm. Furthermore, the thesis investigates heuristic approaches namely non-

dominated sorting genetic algorithm II and the plant propagation algorithm as alternative approximate methodologies for MOLP. It also presents a procedure to compute the most preferred nondominated point of the problem. All algorithms have been tested and compared on existing test instances.

Declaration

The work in this thesis is based on research carried out at the Department of Mathematical Sciences, University of Essex, United Kingdom. No part of this thesis has been submitted elsewhere for any other degree or qualification and it is all my own work, unless referenced to the contrary in the text.

Acknowledgements

I would like to express my deep gratitude to my Supervisor, Professor Abdel Salhi for his insights, guidance, help and continuous encouragement during this work. This work would not have been possible without his help and guidance. I consider myself fortunate to have worked with him. May God Almighty bless and reward him.

I would also like to thank the Head of Department of Mathematical Sciences, University of Essex, Professor Berthold Lausen and all the members of academic and administrative staff (most especially Shauna), for all their support and encouragement through out my stay at this University.

Most importantly, I would also wish to say a big thank you to my beloved wife, Blessing and Children Paschaline, Favour and Paschal Jr. whom I left for a period of four years, my parents and all members of the family for their continuous support and prayers.

Finally, in terms of financial support, I would like to thank the University of Calabar, Calabar, Cross River State, Nigeria whose platform I used to obtain funding from the Tertiary Education Trust

Fund (TETFUND) to undertake this programme of research. Thank you for giving me the opportunity.

Abbreviations

ASF	Achievement Scalarizing Function
ASPA	Affine Scaling Primal Algorithm
ASIMOLP	Affine Scaling Interior Multiple Objective Linear Programming
AHP	Analytic Hierarchy Process
BOA	Benson's Outer-approximation Algorithm
BNP	Best Nondominated Point
BD	Boundary Dictionaries
CP	Compromise Programming
DM	Decision Maker
EEP	Efficiency Equivalent Polyhedron
EMSA	Extended Multiobjective Simplex Algorithm
GA	Genetic Algorithm
iMOLPe	Interactive Multiple Objective Linear Programming explorer
IPM	Interior Point Method

LP	Linear Program
MEF	Maximal Efficient Faces
MPNP	Most Preferred Nondominated Point
MADM	Multi-Attribute Decision Making
MCDM	Multiple Criteria Decision Making
MOEA	Multi-Objective Evolutionary Algorithm
MOGA	Multi-Objective Genetic Algorithm
MOLP	Multiple Objective Linear Programming
MOO	Multi-Objective Optimisation
MOPPA	Multi-Objective Plant Propagation Algorithm
MOPLIB	Multi-Objective Problem Library
MSA	Multi-objective Simplex Algorithm
MSM	Multi-objective Simplex Method
NES	Number of Efficient Solutions
NIA	Nature Inspired Algorithm
NNP	Number of Nondominated Points
NSGA	Nondominated Sorting Genetic Algorithm
OSM	Objective Space Method

PSA	Parametric Simplex Algorithm
PPA	Plant Propagation Algorithm
PDIMOLP	Primal-Dual Interior Multiple Objective Linear Programming
STEM	Step Method
SIMOLP	Simplified Interactive Multiple Objective Linear Programming
VEGA	Vector Evaluated Genetic Algorithm
VMA	Vector Maximization Approach
VD	Visited Dictionaries

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

INTRODUCTION

Most problems faced by Decision Makers (DMs) in the real-world are of the multiple objective optimisation type. That is, they have two or more objective functions by which the success of a particular solution can be measured. Frequently, these objectives which are meant to be achieved, are in conflict with each other and as a result, there does not exist a unique solution that satisfies the DM across all objectives at the same time. Therefore, a most preferred solution must be sought in accordance with the subjective preferences of the DM. The mathematical process of seeking such a solution is known as multiple objective programming, [85]. In the past few decades, there has been an increase in the awareness of multiple objective or

multiple criteria optimisation and the design of multiple objective programming techniques. Most of the earlier techniques were based on the simplex and interior-point methods of Linear Programming (LP). These are the so called decision space approaches. However, objective space approaches are becoming more and more prominent.

Before discussing these further, it is useful to first define the general concept of Multiple Criteria Decision Making (MCDM). Many definitions exist, but most researchers in the field accept the following general one: MCDM refers to the solving of planning and decision problems involving multiple and generally conflicting objectives. "Solving" means that a DM will choose one "reasonable" alternative or most preferred solution from a set of available ones, [95].

MOLP can be define as a branch of MCDM that seeks to optimize two or more linear objective functions subject to a set of linear constraints. It has been an active area of research since the 1960s because of its relevance in practice, [57]. Indeed, many decision making problems that arise in the real world involve more than one objective and can be formulated as MOLP problems. Consequently, it has been widely applied in many fields and has become a useful tool in

decision making.

1.1 Statement of the Problem and Basic Concepts

1.1.1 The MOLP problem

Let us consider a manufacturing firm that uses two raw materials to produce three products A, B, and C. The firm has 300 units of raw material I and 200 units of raw material II in stocks. Each unit of A uses 4 units of raw material I and 5 units of raw material II. B uses 6 units of material I and 3 units of material II and C uses 8 units of material I and 2 units of material II. Let x_1, x_2, x_3 denote the amounts of A, B and C to be produced. The firm is faced with the following constraints

$$4x_1 + 6x_2 + 8x_3 \leq 300$$

$$5x_1 + 3x_2 + 2x_3 \leq 200$$

$$x_j \geq 0, j = 1, 2, 3.$$

The cost prices of producing each unit of A, B and C is 1.5, 1.2 and 2.0 respectively. Product A yields a profit of 2.0 per unit, B yields 1.5 and C yields 2.2. The firm would like to simultaneously minimize total production cost

$$1.5x_1 + 1.2x_2 + 2x_3$$

and maximize total profit

$$2x_1 + 1.5x_2 + 2.2x_3$$

as its objectives, subject to the above constraints.

One can clearly see that these objectives are not compatible; they are conflicting. Ideally, the firm would wish to have a feasible solution that minimizes as well as maximizes the objective functions. Unfortunately, such a solution does not exit. Practitioners usually find a number of solutions before deciding on which is most preferred. Indeed, this is what makes MOLP challenging [106].

More formally, in MOLP, one considers q linear objective functions

defined on x as

$$\begin{aligned} f_1(x) &= c_{11}x_1 + c_{12}x_2 + \ldots + c_{1n}x_n \\ &\vdots \\ f_q(x) &= c_{q1}x_1 + c_{q2}x_2 + \ldots + c_{qn}x_n, \end{aligned}$$

or

$$f_i(x) = \sum_{j=1}^{n} c_{ij}x_j, \, i = 1, ..., q,$$

to be optimized subject to a set of m linear constraints:

$$\begin{aligned} g_1(x) &= a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n \leq b_1 \\ &\vdots \\ g_m(x) &= a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n \leq b_m, \end{aligned}$$

or

$$g_r(x) = \sum_{j=1}^{n} a_{rj}x_j \leq b_r, \quad r = 1, ..., m.$$

These linear inequalities together with the nonnegativity conditions $x_j \geq 0$ for all j form the feasible set of solutions X. To "optimize"

means either "minimize" or "maximize" the functions $f_i(x)$ over a nonempty convex polyhedron X which amounts to finding a vector $\hat{x} \in X$ such that $\sum_{j=1}^{n} c_{ij}x_j \leq \sum_{j=1}^{n} c_{ij}\hat{x}_j$, or $\sum_{j=1}^{n} c_{ij}x_j \geq \sum_{j=1}^{n} c_{ij}\hat{x}_j$ with $\sum_{j=1}^{n} c_{ij}x_j \neq \sum_{j=1}^{n} c_{ij}\hat{x}_j$.

Coefficients c_{ij} can be positive, negative, or zero; they indicate the amount of gain (or loss) to be realized with respect to the ith objective per each unit of increase in the jth variable. Coefficients a_{rj} indicate how much of the rth resource must be expanded per each unit of increase in x_j, [163].

The general MOLP (minimization) problem can be expressed as

$$\begin{aligned} \min \quad & c_1 x = f_1 \\ & \vdots \\ & c_q x = f_q \\ \text{subject to} \quad & x \in X = \{x \in \mathbb{R}^n : Ax \leq b, b \in \mathbb{R}^m, x \geq 0\} \end{aligned} \tag{1.1}$$

or alternately as the linear vector optimization problem in matrix

form

$$\begin{aligned} \min \quad & Cx \\ \text{subject to} \quad & Ax = b \\ & x \geq 0, \end{aligned} \tag{1.2}$$

where C is a $q \times n$ criterion matrix consisting of the rows c_k, $k = 1, ..., q$, A is an $m \times n$ constraint matrix, $b \in \mathbb{R}^m$ is the right hand side vector and "min" amounts to finding an element $\hat{x} \in X$ such that no other point $Cx, x \in X$ is smaller than $C\hat{x}$. It is worthy to note that the solution $\hat{x}$ is not worst than any other solution, but in no way the best, that is, the point $C\hat{x}$ cannot be smaller than or equal to all points $Cx, x \in X$ in general [106]. We say that (1.2) is stated in Standard form when the constraint are written as equalities. It is given in Canonical form when the constraints are inequalities $Ax \leq b$. The feasible set in the decision space is $X = \{x \in \mathbb{R}^n : Ax = b, x \geq 0\}$ and in the objective space it is $Y = \{Cx : x \in X\}$. The set Y is also referred to as the image of X. The upper image is defined as $Y + \mathbb{R}^q_+$ [100].

In practice, problem (1.2) is typically solved by the DM with the

support of the analyst looking for a most preferred (best) solution in the feasible region X. This is because optimizing all the objective functions simultaneously is not possible because of their conflicting nature. Consequently, the concept of optimality as was used in single-objective optimization is replaced with that of Pareto-optimality first introduced by the Italian economist Vilfredo Pareto [115]. It states that, a feasible solution is Pareto-optimal or efficient or nondominated or noninferior if there is no other feasible solution that is equal or better with respect to all objectives in the model, [61]. Therefore solving MOLP is understood as finding either all the efficient or nondominated points or a subset of them, or a most preferred point depending on the purpose for which it is needed.

1.1.2 Efficient Solutions and Nondominated Points

Definition 1.1.2: An efficient solution of the problem is a solution that cannot improve any of the objective functions without deteriorating at least one of the other objectives.

Definition 1.1.3: A weakly efficient solution is one that cannot improve all the objective functions simultaneously.

Definition 1.1.4: A nondominated point in the objective space is the image of an efficient solution in the decision space.

The set of all nondominated points forms the nondominated set. Let $\hat{x} \in X$ be a feasible solution of (1.2) and let $\hat{y} = C\hat{x}$:

- $\hat{x}$ is called efficient if there is no other $x \in X$ such that $Cx \leq C\hat{x}$ and $Cx \neq C\hat{x}$; correspondingly, $\hat{y} = C\hat{x}$ is called nondominated point.

- $\hat{x}$ is called weakly efficient if there is no other $x \in X$ such that $Cx < C\hat{x}$; and $\hat{y} = C\hat{x}$ is called weakly nondominated point, [56].

The set of all efficient solutions and the set of all weakly efficient solutions of (1.2) are denoted by X_E and X_{WE} respectively [29]. $Y_N = \{Cx : x \in X_E\}$ and $Y_{WN} = \{Cx : x \in X_{WE}\}$ are the nondominated and weakly nondominated sets in the objective space of (1.2), respectively.

Definition 1.1.5: The ideal objective point y^* is the minimum criterion values over the efficient set X_E. The ideal objective values are easy to obtain by simply minimizing each objective function individually over the feasible region X, [4].

Definition 1.1.6: Robustness of method can be defined in different ways; in terms of computing efficiency or the ability of a method to solve problems depending on the researcher. In this thesis, we consider it as the ability of a method to solve all problems (both simple and difficult).

As an illustration of the concept, let us consider the following example with two objectives adapted from Hartley [78]

$$x_1 - x_2 + x_3$$

$$2x_1 + x_2 - 3x_3.$$

The extreme points A: (2, 1, 1), B: (6, 3, 2), C: (6, 6, 3), D: (4, 1, 0) and E: (6, 0, 2) have objective values (nondominated points) A: (2, 2), B: (5, 9), C: (3, 9), D: (3, 9) and E: (8, 6). A is dominated by B, B dominates C, but C dominates neither D or E. In other words, a feasible solution is efficient (or nondominated) if it is dominated by no other feasible solution, [78]. Consequently, a rational DM will never choose point A:(2, 2) which can be deleted from consideration. There is no clear dominance among the other four points, they are all nondominated points.

The nondominated faces in the objective space of the problem consti-

tutes the nondominated frontier and the efficient faces in the decision space of the problem constitutes the efficient frontier.

1.2 Solution Approaches to MOLP

1.2.1 Exact Methods

A number of exact approaches has been suggested for the problem as would be seen in the next chapter. Some of the prominent methods include the parametric programming or weighting method, multi-objective simplex method, interior-point methods and the objective space approaches. Apart from the interior-point methods that find a most preferred efficient and nondominated point, the other exact methods are used to generate the set of efficient solutions and non-dominated points of the problem. The drawback of some of these methods is, as the size of the instance increases, the total computation time which increases exponentially. However, instances of small size can be solved efficiently by these methods. Some of them will be highlighted in the following.

Parametric programming or weighting method [160]: Weight-

ing the objectives to obtain efficient solutions appears to be the oldest MOLP technique [41]. It is the most commonly used approach for computing efficient solutions and works in the decision space of the problem. Its basic idea is to transform the MOLP problem into a single objective by using non-negative weighting coefficients $\lambda_1, \lambda_2, ..., \lambda_q$ which are multiplied by the corresponding objective and are then added up to form a weighted sum objective. Formally, it can be expressed as

$$\begin{aligned} &\min\ f_\lambda = \lambda_1 c_1^T x + \lambda_2 c_2^T x + ... + \lambda_q c_q^T x = \sum_{i=1}^{q} \lambda_i c_i^T x \\ &\text{subject to} \quad x \in X. \end{aligned} \tag{1.3}$$

Problem (1.3) is then repeatedly optimized over the original feasible region X using different combinations of weights usually specified by the DM. If $\lambda_i > 0$ for all q, an optimal solution to (1.3) is an efficient solution to MOLP (1.2). But, if $\lambda_i = 0$ for some i, and (1.3) has alternative optimal solutions, then the solutions obtained may only be weakly efficient solutions to (1.2). Usually, normalized weights are considered so that $\lambda \in \mathbb{R}^q : \lambda_i > 0, i = 1, 2, ..., q$, and $\sum_{i=1}^{q} \lambda_i = 1$.

Unfortunately, this method has some limitations in real and general

applications due to the linear form of the weighted sum objective and as the efficient set X_E is usually complicated and non-convex, [30]. The weighted sum method cannot find efficient solutions or nondominated points that are on a non-convex part of the efficient frontier. It can only capture efficient solutions that are located on the convex part of the efficient frontier and cannot be used to appropriately approximate the true efficient frontier because a uniform variation of the weights often leads to an uneven distribution of the efficient solutions, [164]. This is largely due to the fact that the method is often implemented as a convex combination of the objective functions.

Multi-objective simplex method (MSM) [63, 117, 161]: The MSM is a generalized version of the conventional simplex method of LP and works in the decision space of the problem. It is one of the earliest method used to find the set of all efficient solutions of the problem. This is done by moving from one efficient extreme point to adjacent efficient extreme point (using simplex pivots) until all the efficient extreme points have been found. The MSM is known for producing a huge number of efficient extreme points and the computational efforts required to compute them grows exponentially with the problem dimension.

Interior-point methods: Interior-point methods are also decision space based. They appeared in the early 90s following the appearance of the Ellipsoid algorithm of Khachiyan [91] and Karmarkar's algorithm [89]. The first to adapt a variant of Karmarkar's [89] interior-point method to MOLP seems to be Abhyankar [1]. Whereas the MSM makes use of the vertices of the feasible region, interior-point methods generate series of iterates inside the feasible region following a combined search direction that moves the current iterate to a new one. An assessment of the suitability of points used to derive the search direction is done using a utility or preference function [5]. The process continues to generate search directions and new feasible points at each iteration until the algorithm converges to a most preferred efficient point.

Objective space methods: Due to the various difficulties arising from solving the problem in the decision space, efforts were made to solve them in the objective space. One of the most popular objective space method is Benson's Outer-approximation Algorithm (BOA) [30]. It computes the set of all nondominated points of the problem. The algorithm starts by constructing an initial surrounding polyhedron containing the image and an interior point of the image is

determined. At each iteration, the algorithm determines appropriate cuts which are appended to the surrounding polyhedron to make it shrink. This process continues and the surrounding polyhedron keeps on shrinking at each iteration until some of its vertices coincide with the boundary of the image. The algorithm stops and returns the vertices on the boundary of the image as the set of nondominated points of the problem. Another Outer-approximation method is that presented by Csirmaz [43] which is an improved version of BOA. This version solves only one scalar linear program in each iteration during the search process, unlike what is obtainable in [30] where two scalar linear programs are solved in each iteration. The modification in [43] dramatically lead to a significant improvement in computation time compared to what is obtainable when using the method of Benson [30].

1.2.2 Heuristics

Real life MOLP problems are difficult to solve. In a worst-case situation all vertices might be efficient, meaning that the problem would be intractable as there might be exponentially many efficient vertices, [98]. It is, therefore, clear that MOLP is intractable in the

worst-case. Moreover, looking at [90] which shows that listing all vertices of a polyhedron is NP-hard, one can deduce that MOLP is also NP-hard since in the worst-case scenario all vertices must be found to determine the efficient ones. Thus, exact methods are sometimes inefficient and costly, especially when the problem size is large. Instead of finding efficient and nondominated points, heuristics generally find good approximations or near efficient solutions in acceptable computational times. For this reason, they are widely used in multi-objective optimisation (MOO). Given that finding all efficient solutions of MOLP is NP-hard since it is equivalent to enumerating all vertices of the feasible region, [34]. It is astonishing to note that only one approximate method, namely NSGA [140], has been applied to it [37]. Some well-known heuristic methods to MOO and the ones recently developed will be discussed in the following.

Genetic Algorithm: The Genetic Algorithm (GA) was developed by Holland in 1975, [81]. It is based on the idea of natural selection. The algorithm works with three operators; crossover, mutation and reproduction [134]. To implement GA the following are needed:

Initial Population A predetermined number of individuals is ran-

domly generated to form an initial population. The basic GA starts with this population.

Fitness Function This measure is essential for the implementation of GA. It allows to rank individual solutions in the population. It is often the objective function of the optimisation problem.

Selection of Parents The main idea of selection is choosing individuals from the population to be parents to new individuals. The latter are expected to be better than the parents. There are different selection methods such as the Roulette Wheel and Tournament Selection [121].

Genetic Operators: There are three such operators.

Crossover The crossover operator selects a random point which shows a position on the individual. Then, parts of two selected individuals are exchanged to generate two new individuals. This procedure is called a single-point crossover. Another type is called two-point crossover. In this variant, two random positions are selected and parts of parents are exchanged.

Mutation A predetermined number of individuals are mutated. This

is done by changing/flipping some of the entries of an individual. This operator helps exploration of the search space.

Reproduction This copies good individuals into the new population as they are.

Stopping Criteria The algorithm stops when the number of generations reaches a predetermined maximum number of generations. Another commonly used stopping criterion is the maximum number of generations without improvement in the current solutions, [81,134].

Vector Evaluated Genetic Algorithm (VEGA): VEGA is the first population-based evolutionary multi-objective genetic algorithm (MOGA) applied to MOO problems. It was introduced by Schaffer [130]. Here, the population is divided randomly into equal sub-populations at each iteration. Fitness values are assigned to all the solutions in a sub-population based on one of the objective functions and each objective is used to evaluate members in the population. In each sub-population, a fitness proportionate selection is done and the selected members are used for procreation. The process is repeated until convergence is achieved.

Multi-Objective Genetic Algorithm (MOGA): MOGA is the

first population-based evolutionary algorithm that uses the nondominated classification of the population. In MOGA, each solution is checked for its domination in the population and a rank i, equal to n_i the number of solutions that dominates solution i, is assigned to it. To ensure that diversity is achieved, the algorithm uses a sharing function model, [74].

Nondominated Sorting Genetic Algorithm (NSGA): NSGA is one of the multi-objective evolutionary algorithms (MOEA) which has the capacity to find nondominated points in a single run. It was introduced by Srinivas and Deb [140]. In NSGA the population is sorted according to nondomination and classified into a number of fronts $(F_1, F_2, ..., F_n)$. Using niching and nondominated sorting of solutions in every generation, the good solutions are selected for procreation. The algorithm also uses a sharing function model to ensure diversity. Its main issues are:

- It requires the potential user to specify the sharing parameter, which is difficult for the user to determine the ideal value for.
- The nondominated sorting technique is time consuming and computationally expensive.

- It lacks elitism, which may be important in preventing the loss of good solutions once they are found, [159].

Nondominated Sorting Genetic Algorithm II (NSGA-II): NSGA-II [49, 50] is an improved version of NSGA [140]. Though NSGA enjoyed patronage in the multi-objective evolutionary community, it was also widely criticized for the above three issues (lack of elitism, high computational cost of nondominated sorting and the requirement for specifying the sharing parameter). The NSGA-II succeeded in solving all the three issues at once by introducing a fast nondominated sorting and tournament selection using the concept of crowding distance, [116]. In NSGA-II, in addition to the genetic operators of crossover and mutation, two new specialized multi-objective operators or mechanisms have been proposed to solve the above three issues:

- ***Nondominated Sorting***: NSGA-II employs a fast nondominated sorting that is aimed at reducing the complexity of sorting as compared to that used in NSGA.

- ***The Crowding Distance***: It is a technique to replace the sharing parameter that was needed in the old version. This

approach involves ranking among members of a front those that are dominating or being dominated by each other.

These two procedures are used together with the genetic selection operators to create the population of the next generation. The pseudocode of NSGA-II adapted from [159] is given as Algorithm 1.

Algorithm 1 Nondominated Sorting Genetic Algorithm II

```
Initialization:
 ◇ Generate random population
 ◇ Evaluate objective values
 ◇ Assign rank (level) based on nondomination
 ◇ Generate child population
      - Tournament selection
      - Crossover and mutation
For i = 1 to number of generations
 ◇ Parent and child population are assigned rank based on nondomination
 ◇ Generate sets of nondominated fronts
 ◇ Determine the crowding distance between points on each front
 ◇ Select points based on crowding distance calculation
   and fill into the parent population until full
 ◇ Create next generation
 ◇ Tournament Selection
 ◇ Crossover and Mutation
 ◇ Evaluate Objective Values
 ◇ Increment generation index
End
```

Among all the above mentioned MOEAs, NSGA-II is the most popular and known for its capacity to promote the quality of solutions, [87].

There are new nature-inspired algorithms (NIA's) which have shown a lot of promise on nonlinear single and MOO problems. One such

algorithm is the so called Plant Propagation Algorithm or PPA. It emulates the way plants and in particular the strawberry plant propagate [128]. Before going further, we will briefly present the strawberry plant and PPA. The details of multi-objective plant propagation algorithm (MOPPA) will be provided in Chapter 6 where it is investigated and used to solve MOLP.

1.3 The Strawberry Plant

The strawberry plant (Fragaria x ananassa) belongs to the Rose family. The strawberry-growing industry started in Paris in the seventeenth century with the European variety. In 1714, Amedee-Francois Frezier, a mathematician and engineer, hired by Louis XIV [62] to draw maps of South America returned from Chile with some Chilean strawberry plants which give a larger fruit. Subsequent crossings with the European variety and selections led to the modern plant, [128].

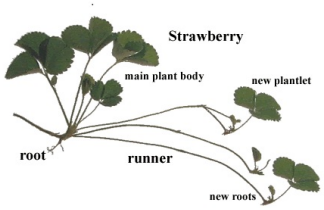

Figure 1.1: The Strawberry Plant ($FragariaXananassa$)

Looking at mature strawberry plants, one will observe after a period of time, a concentration of younger plants around strong and well-established ones; that is the plants send many short runners as they are in good spots. Plants that are not well-established and are not looking very strong, send few but longer runners to explore the environment in search of better spots with enough water, nutrients and sunlight. These basic principles are behind the design of PPA and subsequently MOPPA.

1.4 The Basic Plant Propagation Algorithm

The Plant Propagation Algorithm (PPA) introduced by Salhi and Fraga, [128], emulates the strategy that plants deploy to survive by colonising new places which have good conditions for growth. Plants, like animals, survive by overcoming adverse conditions using often basic but effective strategies. The strawberry plant, for instance, has a survival and expansion strategy which is to send short runners to exploit the local area if the latter has good conditions (with enough water, nutrients and light), and to send long runners to explore new and more remote areas, that is to run away from a not so favourable current area (with poor water supply, nutrients and light). The mechanism of the basic PPA is described below, [134, 147].

The algorithm starts with a population of plants each of which represents a solution in the search space. X_i denotes the solution represented by plant i in an n-dimensional space. $X_i \in R^n$, i.e. $X_i = [x_{ij}]$, for $j = 1, \ldots, n$ and $x_{ij} \in R$. NP is the population size. This iterative process stops when g the counter of generations reaches its given maximum value $g_{\max}$.

Algorithm 2 The Plant Propagation Algorithm (PPA)

```
Generate a population P = X_i, i = 1, ..., NP of plants;
 g ← 1
 for g = 1 : g_max
     Compute N_i = f(X_i), ∀ X_i ∈ P
     Sort P in ascending order of fitness values N (for minimization);
     Create new population φ
     for each X_i, i = 1, ..., NP
         r_i ← set of runners where both the size of the set and the distance for
         each runner (individually) are proportional to the fitness values
         N_i;
         φ ← φ ∪ r_i (append to population);
     endfor
     P ← φ (new population);
 endfor
Return P, (the population of solutions).
```

Individuals/plants/solutions are evaluated and then ranked (sorted in ascending or descending order) according to their objective (fitness) values and whether the problem is a min or a max problem. The number of runners of a plant is proportional to its objective value and conversely, the length of each runner is inversely proportional to the objective value, [128]. For each X_i, $N_i \in (0, 1)$ denotes the normalized objective function value space. The number of runners for each plant to generate is

$$n_r^i = \lceil (n_{\max} \; N_i \; \beta_i) \rceil \tag{1.4}$$

where n_r^i shows the number of runners and $\beta_i \in (0, 1)$ is a randomly

picked number. Thus, for each plant, the minimum number of runners is 1. The distance value found for each runner is denoted by dx_j^i. It is:

$$dx_j^i = 2(1 - N_i)(r - 0.5), \ for \ j = 1, \ldots, n. \tag{1.5}$$

where $r \in [0, 1]$ is a randomly chosen value.

Calculated distance values are used to position the new plants as follows:

$$y_{ij} = x_{ij} + (b_j - a_j) \ dx_j^i, \ for \ j = 1, \ldots, n. \tag{1.6}$$

where y_{ij} shows the position of the new plant and $[a_j, b_j]$ are the bounds of the search space. If the bounds of the search domain are violated, the point is adjusted to be within the domain $[a_j, b_j]$.

The new population that is created by appending the new solutions to the current population is sorted. In order to keep the number of population constant, the solutions that have lower objective value are dropped, [134].

The algorithm was originally designed for single-objective nonlinear optimisation problems. It has been successfully tested on single-objective and bicriteria continuous optimisation problems in [128].

It has also been applied successfully to a single-objective dynamic optimisation problem in the built environment [68].

1.5 The Algebra of MOLP

Let us re-write (1.3) as

$$\min\{\lambda^T Cx \,:\, Ax = b, x \geq 0\} \tag{1.7}$$

and use the notation $\bar{C} = C - C_B A_B^{-1} A$ for the reduced cost matrix with respect to basis B, $R = \bar{C}_N$ for the nonbasic part of the reduced cost matrix and also note that $\bar{C}_B = 0$ (the basic part of the reduced cost matrix). It has been shown in [56] that if $X_E \neq \emptyset$, then X has an efficient basic feasible solution.

Definition 1.5: A feasible basis B is called efficient if B is an optimal basis of LP(1.7) for some $\lambda \in \mathbb{R}^q_{>0}$. A pivot is said to be a feasible pivot if the solution obtained after the pivot step is feasible, even if the pivot element is negative.

Definition 1.6: Two bases B and $\hat{B}$ are called adjacent if one can be obtained from the other by a single pivot step.

Definition 1.7: (a) Let B be an efficient basis. Variable x_j, $j \in N$ is called efficient nonbasic variable at B if there exist a $\lambda \in \mathbb{R}^q_{>0}$ such that $\lambda^T R \geq 0$ and $\lambda^T r^j = 0$ where r^j is the column of R corresponding to variable x_j.

(b) Let B be an efficient basis and let x_j be an efficient nonbasic variable. Then a feasible pivot from B with x_j entering the basis is called an efficient pivot with respect to B and x_j.

It has also been shown in [56] that if B is an efficient basis and x_j an efficient nonbasic variable, then any efficient pivot from B will lead to an adjacent efficient basis $\hat{B}$. To determine if a nonbasic variable x_j at an efficient basis B is efficient, one needs to perform Evans-Steuer [63] nondominance test which involves solving the LP

$$\begin{aligned} \max \quad & e^t v \\ \text{subject to} \quad & Rz - r^j\delta + Iv = 0 \\ & z, \delta, v \geq 0 \end{aligned} \tag{1.8}$$

It has also been shown in [56] that (1.8) is always feasible and can only have an optimal solution with $v = 0$ or be unbounded. Therefore:

- x_j is an efficient nonbasic variable if and only if (1.8) is bounded

and has optimal value 0,

- x_j is an inefficient nonbasic variable if and only if (1.8) is unbounded.

In MSM, one must consider negative pivot elements because if nonbasic variable x_j is efficient and column j of $\bar{A}$ contains no positive element, then the increase of x_j is unbounded, a fact that indicates an unbounded LP in the single objective case. However, since $\lambda^T r^j = 0$ this is not the case in MOLP, rather unboundedness of X_E is detected in the direction given by the vector with components $\frac{-\bar{b}_i}{\bar{A}_{ij}}, i \in B$, $x_j = 1$. Though, this is not a feasible pivot as it does not lead to another basis, but does not constitute unboundedness of the objective function. The results so far allow one to move from efficient basis to efficient basis until all efficient solutions are found. [56].

In MOLP, one and only one of the following cases can occur:

- The MOLP is infeasible, that is $X = \emptyset$,
- it is feasible ($X \neq \emptyset$) but has no efficient solutions ($X_E = \emptyset$), or
- it is feasible and has efficient solutions, that is $X_E \neq \emptyset$.

The MSM handles these situations in three phases as follows:

Phase I: Determine an initial basic feasible solution or stop with the conclusion that $X = \emptyset$.

Phase II: Determine an initial efficient basis or stop with the conclusion that $X_E = \emptyset$.

Phase III: Pivot among efficient bases to determine all efficient extreme points and unbounded efficient edges of X_E.

In the first phase, the algorithm is initialized with a basic feasible solution by solving the following auxiliary LP:

$$\begin{aligned} \min \quad & e^T z \\ \text{subject to} \quad & Ax + Iz = b \\ & x, z \geq 0 \end{aligned} \tag{1.9}$$

where $e^T = (1, ..., 1) \in \mathbb{R}^q$ and I is the identity matrix of proper order.

The MOLP is feasible, that is $X \neq \emptyset$ if and only if (1.9) has an optimal solution zero.

In phase II, the solution of (1.7) with $\lambda > 0$ will yield an efficient

basis, provided (1.7) is bounded. First, the following auxiliary LP is solved to check if $X_E = \emptyset$

$$\begin{aligned} \min \quad & u^T b + w^T C x^0 \\ \text{subject to} \quad & u^T A + w^T C \geq 0 \\ & w \geq e, \end{aligned} \tag{1.10}$$

$X_E \neq \emptyset$ if and only if (1.10) has an optimal solution $(\hat{u}, \hat{w})$ with $\hat{u}^T b + \hat{w}^T C x^0 = 0$. This optimal solution returns an appropriate weighting vector $\hat{w}$. Then $\hat{u}$ is also an optimal solution of the LP(1.11)

$$\min\{u^T b \,:\, u^T A \geq -\hat{w}^T C\}. \tag{1.11}$$

The dual of (1.11) has an optimal basic feasible solution which is efficient and it is equivalent to the weighted sum LP

$$\min\{\hat{w}^T C x \,:\, Ax = b, x \geq 0\}. \tag{1.12}$$

The LPs (1.10) and (1.12) are the necessary tools in phase II of the algorithm. If (1.10) is infeasible, $X_E = \emptyset$. Otherwise an optimal solution of (1.10) yields an appropriate weighting vector $\hat{w} = \lambda$ for which (1.7) has an optimal basic feasible solution which is an initial

efficient solution of the MOLP, [56].

1.6 Research Objectives

This research study is intended to look at the state-of-the-art technology for MOLP. It considers prominent exact algorithms, understanding them, implementing and modifying some of them, as well as comparing them comprehensively on a series of existing test problems. We would also apply two population based algorithms to MOLP. The research objectives of this thesis can be categorized as follows:

- to present a state-of-the-art survey of MOLP algorithms developed over the past few decades;
- to extend the Multi-objective Simplex Algorithm (MSA) of Evans and Steuer [63] to generate the set of all nondominated points and compare it with Benson's [30] Outer-approximation Algorithm (BOA) in order to determine its effectiveness;
- to further compare the extended multi-objective simplex algorithm (EMSA) with Arbel's Affine Scaling Interior MOLP Algorithm (ASIMOLP) [6] and BOA [30];

- to comprehensively compare the Parametric Simplex Algorithm (PSA) of Rudloff *et al.* [124] with BOA [30];

- to apply nature-inspired population based stochastic algorithms such as the multi-objective strawberry plant propagation algorithm (MOPPA) [67] and NSGA-II [49, 50] which were originally designed to solve multi-objective nonlinear programming problems to MOLP using the penalty function method to handle general constraints.

1.7 Main Contributions

We have outlined some of the thesis objectives above, now we outline the main contributions of the thesis as follows:

- An extensive survey of the topic is presented. No source of any note in the literature going back at least five decades has been ignored.

- We have extended MSA to make it a competitive alternative to what is on offer today in terms of exact methods.

- We have introduced a procedure to compute the Most Preferred

Nondominated Point (MPNP).

- We have, we believe, applied heuristic approaches for the first time to MOLP. These are the well established NSGA-II [49, 50] and the recently introduced Plant Propagation Algorithm or PPA, [128].

- We have carried out an extensive empirical evaluation of these approximate approaches against the most prominent exact methods investigated here, EMSA, ASIMOLP, BOA and PSA.

1.8 Organization

This thesis is divided into seven chapters. Chapter 2 presents a state-of-the-art survey of MOLP algorithms developed over the past five decades. The survey classifies the algorithms into two broad categories: Non-interactive and Interactive algorithms. Section 2.2 provides a review of Non-interactive algorithms while Section 2.3 reviews the Interactive approaches. Section 2.4 discusses heuristic approaches to MOO. Finally, Section 2.5 summarizes Chapter 2.

Chapter 3 presents an extension of the MSA of Evans and Steuer [63]

to generate the set of all nondominated points of the problem. It also compares the extended and the original version with BOA. Section 3.2 presents the MSA of Evans and Steuer [63] while Section 3.3 presents its extended version. Section 3.4 discusses two scalarization techniques. We present BOA in Section 3.5. Section 3.6 provides the experimental results obtained with these algorithms. We present the summary of experimental results in Section 3.7 and a summary of Chapter 3 is presented in Section 3.8.

Chapter 4 provides a computational investigation on EMSA, ASIMOLP [6] and BOA [30]. Section 4.2 presents ASIMOLP. The determination of a priority vector used in ASIMOLP is presented in Section 4.2.2. We presents Interactive ASIMOLP in Section 4.3. Section 4.4 discusses the selection of a most preferred nondominated point. We provide experimental results in Section 4.5. The summary of experimental results is presented in Section 4.6 and a summary of the chapter in Section 4.7.

Chapter 5 provide a computational investigation of two MOLP algorithms namely PSA of Rudloff *et al.* [124] and BOA. PSA is presented in Section 5.2. We provides additional illustration of BOA in Section

5.3. Section 5.4. presents the experimental results obtained with the two algorithms. We present a summary of experimental results in Section 5.5. Finally, Section 5.6 summarizes Chapter 5.

Chapter 6 applies nature-inspired population based stochastic algorithms, MOPPA [67] and NSGA-II [49,50] to MOLP. It also compares them with exact methods. Section 6.2 presents the multi-objective plant propagation algorithm. The solution procedure is presented in Section 6.3. Section 6.4 provides the experimental results obtained with MOPPA and NSGA-II and exact methods EMSA, ASIMOLP, BOA and PSA. We present a summary of experimental results in Section 6.5. A summary of Chapter 6 is presented in Section 6.6.

Chapter 7 concludes the whole thesis, provides its findings and suggests some future research directions.

Chapter 2

REVIEW OF LITERATURE

2.1 Introduction

This chapter presents a state-of-the-art survey of MOLP algorithms developed over the last five decades. The algorithms are classified into two broad categories: Non-Interactive algorithms and Interactive ones. The algorithms in the first category are of the Simplex, Interior-point and Objective space based algorithms. While those in the second category are of the Simplex and Interior-point based ones. A tabulated list of all algorithms is included in the appendices section.

This chapter is organized as follows. Section 2.2 is a review of lit-

erature on the Non-interactive algorithms. These algorithms include the simplex, interior-point and objective space based methods. Section 2.3 is a review on the Interactive algorithms which include only the simplex and interior point based ones. In Section 2.4, we discuss heuristic approaches to MOO problems in general. A summary of the chapter is presented in Section 2.5.

Before going any further, let us reiterate that the algorithms considered are in the following categories:

1. **Non-Interactive Algorithms**
    - Simplex based algorithm and its variants, (MSA)
    - Interior-point based methods, (IPM)
    - Objective space based methods, (OSM)
2. **Interactive Algorithms**
    - Simplex based algorithm and its variants, (MSA)
    - Interior-point based methods, (IPM).

We note that in the objective space methods, the simplex algorithm or the dual simplex algorithm are being invoked at some point or

the other during the search process. This may suggest that they should be put in the simplex based class. However, for more clarity and given that there is a strong trend to refer to them as objective space methods, we prefer to put them on their own. Their underlying philosophy is different from that of the simplex methods.

2.2 Non-Interactive Algorithms

2.2.1 Simplex based algorithms

In the last five decades a number of approaches has been suggested for generating either the entire efficient decision set X_E or the non-dominated set Y_N or a subset thereof, or a most preferred solution to the problem.

The first parametric programming approach to MOLP appears to be due to Schonfeld, [132]. The author presented an algorithm for the enumeration of efficient solutions of the problem using parametric programming. Few years later, Geoffrion [73] presented a bicriterion parametric linear programming algorithm to solve the problem. It was noted that solutions are not extreme points of the feasible region,

pointing to the fact that one should not rely only on algorithms that consider extreme point solutions as in ordinary LP.

Eiselt and Sandblom [61] noted that, Evans and Steuer [63], Philip [117] and Zeleny [162] all derived generalized versions of the simplex method known as MSA. That of Philip, [117] first determines if an extreme point is efficient and subsequently checks if it is the only one that exists. If not, the algorithm finds them all. This MSA approach, however, may fail at a degenerate vertex. In [118], Philip modified it to overcome this difficulty.

The MSA of Evans and Steuer [63] also generates all the efficient extreme points and unbounded efficient edges of MOLPs; see also Algorithm 7.1, page 178 of [56]. The algorithm first establishes that the problem is feasible and has efficient solutions. Thereafter, it generates them all. An LP test problem is solved to determine the pivots that lead to efficient vertices. The algorithm is implemented as a software called ADBASE, [144].

The MSA variant of Zeleny [162] also uses an LP test problem to determine the efficiency of extreme points. But, here, vertices are tested for efficiency after they have been obtained unlike in [63] where

the test problem determines pivots leading to efficient vertices.

Yu and Zeleny [156,157] used the approach in [162] to generate the set of all efficient solutions and presented a formal procedure for testing the efficiency of extreme points. The efficient solutions are derived from the efficient faces, in a top-to-bottom search strategy. Numerical illustrations with three objectives were used to demonstrate the effectiveness of the method. In a similar paper, Yu and Zeleny [158] applied their approach expanded in [157] to parametric linear programming. Two basic forms of the problem and two computational procedures for computing the efficient set were presented: the direct decomposition of the parametric space into subspaces associated with extreme points and the indirect algebraic method. From a numerical experience point of view, the indirect algebraic method outperforms the direct decomposition.

In [24], Belenson and Kapur developed a technique which applies LP approach to solve two person zero-sum games with mixed strategies. The problem was formulated as a weighted sum MOLP problem and solved with the simplex method. Relatively small MOLP instances with two objectives were used to demonstrate the applicability of the

method. It was noted however, that the number of individual LPs to be solved during the search process could be too large which makes it difficult to solve larger problems.

Isermann [83] proposed a variant of the MSA of Evans and Steuer [63] that solves fewer LPs when determining the entering variables. The algorithm first establishes whether an efficient solution for the problem exists, and solves a test problem to determine pivots leading to efficient vertices. It was implemented as a software called EFFACET in Isermann and Naujoks [84].

In [27], Benson validated Isermann's method in [83] for finding an initial efficient extreme point for the problem and also proposed a method to do the same.

Ecker and Kouada [55] proposed a method for finding an initial efficient solution of the problem. It was however noted in [83] that the efficient solution obtained with the method in [55] may not be an efficient basic feasible solution of the problem.

The MSA of Gal [69] generates the set of all efficient vertices and higher-dimensional faces. This approach is meant to address the problem of determining efficient faces and higher dimensional faces

not resolved in [63] and [117]. Here, efficient extreme points are generated using a test problem. The algorithm also determines higher-dimensional efficient faces for degenerate problems which were only discussed in [83] and [162] but not solved. The efficient faces are generated in a bottom-to-top search strategy unlike what was suggested in [156, 157].

Steuer [143] applied the MSA of Evans and Steuer [63] to parametric and non-parametric MOLP. Different methods for obtaining an initial efficient extreme point as well as different LP test problems were also presented. Efficient extreme points are generated through the decomposition of the weight space into finite subsets that provide optimal weights corresponding to extreme point solutions.

In [56], Ehrgott applied the MSA of Evans and Steuer [63] to solve MOLP problem instances with two and three objective functions. Ecker and Kouada [54] also proposed a variation on the MSA of Evans and Steuer [63]. They noted that algorithms usually started from an initial efficient extreme point and moved to an adjacent one following the solution of an LP problem. The proposed method does not require the solution of any LP problem to test for the efficiency

of extreme points and the feasible region needs not be bounded. The algorithm enumerates all efficient extreme points and appears to have computational advantage over other methods.

In a different paper, Ecker *et al.* [53] presented yet another variant of MSA. The algorithm first determines the maximal efficient faces incident to a given efficient vertex (i.e. containing the efficient vertex) and ensures that previously generated efficient faces are not regenerated. This is done following a bottom-to-top search strategy as in [69], which dramatically improves computation time. The proposed approach was illustrated with a degenerate example given in [157], to demonstrate its applicability. It was computationally more efficient than the method in [157].

The MSA of Armand and Malivert [20] determines the set of efficient extreme points even for degenerate MOLPs. The approach follows a bottom-to-up search strategy and utilizes a lexicographic selection rule to choose the leaving variables which proves effective when solving degenerate problems. It was tested successfully on a number of degenerate problems. A numerical example with five objectives and eight constraints which was solved in [157] was also used

to demonstrate its effectiveness. The proposed MSA was superior to that in [157]. In a similar paper, Armand [19], proposed another algorithm for finding all maximal efficient edges of the problem. The lexicographic pivoting rule is also used here to determine all the maximal efficient faces. It was reported that this algorithm is computationally superior to that in [20] upon comparison.

A modification of PSA for single objective LP to solve bounded bicriterion LP problems was presented by Ruszczynski and Vanderbei [125]. The approach was applied to a large mean-risk portfolio optimization problem for which the nondominated portfolios were generated.

Ehrgott *et al.* [58] introduced a primal-dual simplex algorithm for bounded MOLPs. This algorithm finds a subset of the efficient solutions that are used to generate the whole efficient frontier. The algorithm starts with a coarse partitioning of the weight space which continues in each iteration as well as solves a costly LP in each iteration. A vertex enumeration is then performed in the last step to obtain efficient solutions. Numerical illustrations show the applicability of the algorithm.

Recently, Rudloff *et al.* [124] suggested a PSA for the problem. The algorithm is a generalization of the algorithm in [125] and is similar to that in [58]. It works for any dimension, solves bounded and unbounded problems (unlike the algorithm in [58] and [125]), but does not find all the efficient solutions unlike the algorithm of Evans and Steuer [63]. Instead, it finds a subset of efficient solutions based on the idea of Löhne [100]. That is, a subset of efficient solutions that allows to generate the whole efficient frontier. The algorithm performs pivoting for only one leaving variable among the set of all possible leaving variables. It was compared with a version of BOA in [77] which is an objective space based algorithm, and that in [63] using small MOLP instances which were randomly generated with 3 and 4 objectives and up to 50 variables and constraints. Numerical experiments show that the proposed algorithm outperforms Benson's algorithm for non-degenerate problems. However, Benson's algorithm is better for highly degenerate ones. The parametric MSA was also found to be computationally more efficient than Evans and Steuer MSA, [63].

In [133], Seiford and Yu extended MSA to solve a multiple criteria and multiple constraint levels (right hand sides) problem (i.e. mul-

tiple discrete right hand sides). The approach was regarded as a symmetric extension of MSA that generates all the efficient solutions of the problem. It was noted however that the algorithm is limited, since it is only suitable for problems with multiple discrete right hand sides.

Strijbosch *et al.* [146] proposed a simplified MOLP algorithm (MOLP-S) based on the simplex method to trace the efficient solutions of the problem. It was reported that the proposed algorithm was computationally more efficient than the ADBASE software of Steuer [143] upon comparison.

A detailed discussion on the field of MCDM in general and its models is presented by Zeleny in [163]. The author also gave a complete account of MSA as well as presented different applied MOLP formulations. The problems were solved effectively using his approach in [162].

A method called the moving optimal method for finding the efficient solutions of the problem based on the simplex method was proposed in [114]. The algorithm proceeds by solving q single objective LPs of a given problem to obtain the optimal solution of each LP. Using the

optimal solution of the first LP as an initial solution of the next and repeating the process, efficient solution line segments are generated. An instance posed in [48] was used to demonstrate its effectiveness. The proposed method was found to be better than that in [48].

In [148], Suprajitno presented an interval arithmetic simplex-based algorithm for the problem. According to the author, the information needed to solve real world problems is uncertain, hence the need for using intervals to define the data. The problem is formulated as an interval MOLP problem to start with and solved using a modified simplex algorithm. The procedure was found to be effective when tested on interval MOLPs posed in [113] for which some efficient solutions were obtained.

Sayin [129] presented a method for finding the set of efficient solutions of the problem based on the approach in [157]. The method incorporates a simple LP test that identifies efficient faces and also employs a top-to-bottom search strategy to generate maximal efficient faces. Numerical experiments show that the computational effort increases with the problem dimension and the number of variables appears to have a significant effect on computation time.

In [154], Yan *et al.* investigated the structure of efficient and weakly efficient solutions as well as proposed a method for finding them. The algorithm determines a finite number of weights that corresponds to weighted sum problems. It was noted however, that heavy computational effort may be required as the procedure involved solving many LPs during the search process for efficient and weakly efficient solutions.

Foroughi and Jafari [65] presented a modification of the algorithm in [154]. It was shown that if in one stage of the algorithm, the weighted sum problem has no finite optimal solution, the method in [154] would be ineffective. The modified algorithm overcomes this difficulty without changing its complexity.

Pourkarimi *et al.* [119] also improved on the algorithm of Yan *et al.* [154] by presenting a method that solves fewer LPs during the search process. The proposed method used a bottom-to-up search strategy and gives a representation of the maximal efficient faces. A numerical illustration was used to demonstrate its computational efficiency over the method in [154].

Kim and Thien [93] presented an algorithm for generating the set of

all efficient extreme points and rays for the problem. A bottom-to-top search strategy was used to determine the efficiency of extreme points. An MOLP instance introduced in [157] which was also solved in [20] was used to demonstrate the applicability of the approach. It was noted however that the number of objectives has a significant effect on the computational time unlike what was reported in [129]. In a similar paper, Kim *et al.* [94] extended the approach in [93] to MOLPs associated with linear multiplicative (nonconvex global optimization) programming problems for generating the same.

An approach where the two dimensional simplex tableau is replaced with a three dimensional one was proposed in [36]. Contrary to MSA which allows displacement between zero dimensional faces (extreme points), this approach enables displacement between higher dimensional faces and leads to the determination of efficient faces of higher dimensions without necessarily determining its vertices.

More recently, Luc [106] presented his text on the fundamental concepts in MOLP which introduces readers to the subject. The author also presented different MOLP dual problems and discussed MSA in detail with illustrative examples. The author was motivated by the

fact that, apart from the work of Zeleny [162] and Steuer [143] nearly all other texts on the subject are devoted to nonconvex problems.

Of all these MSA variants discussed above, that of Evans and Steuer [63] is the most popular and successful for computing all efficient extreme points of the problem [131].

2.2.2 Interior-point based algorithms

Simplex-based approaches and its variants make explicit use of the vertices of the feasible region. Interior point approaches, however, generate iterates in the interior of the feasible region. Various such approaches have been suggested. The difference between them depends on the methodology employed to assess the suitability of points used to derive a combined search direction along which one heads towards the next iterate.

The first to adapt a variant of Karmarkar's [89] interior-point algorithm, to solve MOLP appears to be Abhyankar [1]. It relies on the method of centers. It uses a parameterization of ellipsoids in the n-dimensional space to approximate the efficient frontier of the problem in polynomial time.

Arbel [5] also modified and adapted a variant of Karmarkar's algorithm [89] resulting in the so called Affine Scaling Interior MOLP (ASIMOLP) algorithm. He used the convex combination of individual directions to derive a combined direction along which to step toward the next iterate. Specifically, the algorithm generates step direction vectors based on the objectives of the problem. The relative preference of these directions is then assessed using a utility (or preference) function to obtain the points used in combining them into a single direction vector that moves the current iterate to a new one. The process is repeated until the algorithm converges to a most preferred efficient solution after meeting some termination conditions.

In [6], Arbel proposed another ASIMOLP algorithm. This approach offers another means of assessing preference information to establish a combined search direction rather than using the DM's utility function. The Analytic Hierarchy Process (AHP) developed in [127] was applied to obtain the relative preference of points used to derive a combined direction along which the next step is taken. It is based on the assessed preferences to weigh the step direction vectors for each of the objectives in order to derive a combined step direction vector. This process continues to generate search directions and new

feasible points at each iteration, until the algorithm converges to a most preferred point on the efficient frontier.

A modification of the Affine Scaling Primal Algorithm (ASPA) is presented in [16]. Here, the procedure assumes that the DM has an implicitly known utility function and search directions are generated to approximate the gradient of this function. These directions are later combined to arrive at the next iterate. This continues in a sequence of steps until the algorithm converges to an efficient solution.

In a similar paper, Arbel [9] presented another modification of ASPA to generate search directions in the form of projected gradients which improves each objective. The projected gradients are then used to derive a combined direction that leads to a new iterate. By taking a full step along the combine direction enables one to reach a boundary or anchor point. The anchor points together with the current iterate defines a cone of opportunities at which to terminate the process upon convergence at the boundary of the feasible region.

In [7] yet another ASIMOLP algorithm based on the AHP has been suggested. The derived preference information is applied to the projected gradients in order to obtain anchoring points and cones used

in searching for a most preferred solution. The boundaries of the constraints polytope are constantly probed to make more directions available, which enables one to arrive at a most preferred solution.

The formal principles behind interior-point LP approaches is presented by Arbel in [8], the author provided the basic details of ASPA which are used in the implementation of ASIMOLP.

Wen and Weng [151] modified ASIMOLP in [5] to resolve zigzagging issues. Zigzagging is an alternating search path that is usually generated by ASIMOLP and it often leads to poor convergence. The modified algorithm, however, may not yield a most preferred efficient solution.

In [13], Arbel and Korhonen introduced a new starting mechanism which makes it possible to start an algorithm from a feasible or infeasible solution. The problem is first augmented so as to obtain an initial feasible solution. This augmentation is controlled by a nonnegative parameter which verifies the efficiency of the final solution. The parameter is then forced to zero at the end of the iterative process thereby achieving an efficient solution.

Zhong and Shi [165] applied ASIMOLP to find the solution of mul-

tiple criteria and multiple constraint level problems. The approach involves partitioning the columns of the right hand side matrix (with each column representing the right hand side vector) into different MOLP problems. The problems are then solved by ASIMOLP and their efficient solutions combined through a convex combination to obtain a most preferred efficient solution.

Lin *et al.* [99] also proposed a modification of ASIMOLP [5]. They adopted the utility function trade-off method to weigh the objective functions involved and compared the modified algorithm with that in [151] and the simplex method. Numerical experiments show that their algorithm is superior. On computing efficiency, the interior point based algorithms outperform the simplex-based ones on large scale problems.

In [64], Fliege presented a method for approximating the solution of the problem based on a warm-start strategy using the path-following primal interior-point algorithm. With the warm-start strategy, points from the iteration history of scalar problems already solved are used as starting points until an approximate solution to the problem is obtained.

Arbel and Sadka [18] presented a derivation of the Euclidean center (and its weighted version) which is defined as the point in the decision space which is the centre of the largest sphere that can be inscribed in the constraints polytope. By assigning weights to the different decision variables, one traverses the entire efficient frontier, thereby arriving at an efficient solution of the problem.

Weng and Wen [152] presented an ASIMOLP based algorithm. It computes a weighted sum of the different search directions involved using a utility function. These search directions are then normalized with the weights to obtain a combined direction that moves the current solution to an anchor point. Computational experiments show that the proposed algorithm is suitable for solving large scale instances.

A comprehensive account of the algorithm in [5] is given in Gal [70]. Also discussed were the modifications needed for adopting the single objective interior-point algorithm to MOLPs. Numerical illustrations with two and three objectives were used to demonstrate the effectiveness of the algorithm.

2.2.3 Objective space based algorithms

Due to the various difficulties arising from solving MOLP problems in the decision space (such as having different efficient solutions that map onto the same point in the objective space), efforts were made to look at the possibility of solving them in the objective space. The first attempt at exploring the possibility of solving them in the objective space appears to be due to Dauer [44]. He presented an analysis of the objective space and obtained a characterization between a face of the objective space and the corresponding face of the decision space. Numerical illustrations show the collapsing that occur when mapping the decision space onto the objective space, thereby forming the basis for a new procedure for MOLP.

In a similar paper, Dauer and Liu [46] presented another analysis of the objective space and developed a procedure for determining the nondominated extreme points and edges in this space. They observed that not all extreme points of the constraint space necessarily map to extreme points in the objective space. It was noted that their technique analyzes a simpler structure than those analyzed by algorithms based on the decision space.

In [47], Dauer and Saleh determined the nondominated extreme points of the problem by solving a resulting single-objective nonparametric LP. The optimal basic solution of this LP was used to obtain the corresponding nondominated extreme point in the objective space.

An algorithm for the construction of an Efficiency Equivalent Polyhedron (EEP) associated with the objective space was presented in [71]. The proposed algorithm generates a nonredundant inequality representation of the constructed EEP to obtain the Maximal Efficient Faces (MEF) of the problem. Similarly, Dauer and Gallagher [45] presented the MEF algorithm for determining higher dimensional MEF of the problem. The algorithm utilizes a nonredundant inequality representation of the EEP generated by the algorithm in [71] as input, and work in conjunction to obtain the MEF of higher dimension. It was noted however, that the proposed algorithm is limited to only problems with two and three objectives.

In [92], Kim suggested that it is better to work with an EEP that is smaller than the feasible set, because not all data in this set plays a role in determining the efficient set X_E. Based on his observation, he proposed an outer-approximation algorithm for constructing

a smaller EEP that is used to obtain all the nondominated points of the problem. This algorithm is quite similar to that in [71] as both worked with nonredundant inequalities representation of EEP, but differ in the definition of an EEP. Whereas in the former it was described as being associated with the objective space, here it is associated with the decision space.

Benson and Sun [33] proposed a weight set decomposition approach for generating the set of nondominated points using the idea of Zeleny [162]. The authors modified Zeleny's approach to resolve the issue of one-to-one correspondence with the efficient decision set. Numerical illustrations show that the approach was computationally less demanding than the decision set-based method in [162].

Benson [30], who presented a detailed account of decision space approaches, proposed an algorithm for generating the set of all nondominated points in the objective space. This is the so called BOA. According to the author, this algorithm is the first of its kind. Computational results suggests that the objective space based approach is better than the decision space based one. A further analysis of objective space based algorithm for the problem was presented in [28]. This

outer-approximation algorithm also generates the set of all weakly nondominated points, thereby enhancing the usefulness of the algorithm as a decision aid.

Another of Benson's [29] suggestions is a hybrid approach for solving the problem in the objective space. The approach partitions the objective space into simplices that lie in each face so as to generate the set of nondominated points. This idea was earlier presented in [23]. The algorithm is quite similar to that in [30]. The difference between them is in the manner in which the nondominated vertices are found. While a vertex enumeration procedure is employed in [30], a simplicial partitioning technique is used in the latter.

In [135], a modification of the algorithm of Benson [30] was presented. While in [30], a bisection method that requires the solution of many LPs in one step is required, here, solving one LP achieves the desired effect and in the process improves computation time. In [136] an approximate dual variant of the algorithm of Benson [30] for obtaining approximate nondominated points of the problem was proposed. The proposed algorithm was applied to the beam intensity optimization problem of radio therapy treatment planning for which approximate

nondominated points were obtained. Numerical testing shows that the approach is faster than solving the primal directly. Similarly, Ehrgott *et al.* [57] presented a dual variant of Benson's [30] algorithm using results from the geometric duality theory of Heyde and Lohne [79]. Numerical illustrations also suggests that the dual variant of the algorithm may have computational advantage.

The explicit form of BOA [30] as modified by Shao and Ehrgott [135] is presented in [100]. This version solves two LPs in each iteration during the process of obtaining the nondominated extreme points. Löhne [101], presented a Matlab implementation of this algorithm called BENSOLVE-1.2 for computing all the nondominated points and extreme directions (unbounded nondominated edges) of the problem.

Csirmaz [43] presented an improved version of BOA [30] that solves one LP and a vertex enumeration problem in each iteration. While in [30], solving two LPs to determine a unique boundary point and a supporting hyperplane of the image is required in two steps, here, the two steps are merged and solving only one LP does both tasks and improves computation time. The algorithm was used to generate all

the nondominated vertices of the polytope defined by a set of Shannon inequalities on four random variables so as to map their entropy region. Numerical testing shows the applicability of the approach to medium and large instances with 3 and 10 objectives and up to 5772 variables and 635 constraints.

Hamel *et al.* [77] introduced new versions and extensions of the algorithm in [30]. The primal and dual variants of the algorithm solve only one LP problem in each iteration. Tests reveal a reduction in computation time.

Similarly, Löhne *et al.* [102] extended the primal and dual variants of the algorithm in [30] to solve convex vector optimization problems approximately in the objective space.

Tantawy [149] presented a method for obtaining nondominated points. He also developed a condition for an efficient extreme point to have a corresponding nondominated point in the objective space. It was suggested that, DMs should rely on the nondominated points because they are fewer when compared with the efficient solutions.

Heyde and Löhne [79] proposed a geometric approach to duality in MOLP which was similar to the theory of convex polytopes [75]. It

was shown that there exists a one-to-one mapping between the set of all minimal faces of the primal problem and maximal faces of the dual problem, such that the mapping is inclusion-reversing. Using this mapping, the authors computed the minimum number of faces in the primal and the corresponding maximum number of vertices in the dual problem and vice versa. In a similar paper, Heyde *et al.* [80] presented a duality theory for the problem using a set-valued dual objective function. Numerical application using a bicriterion portfolio optimization problem reveals the practical application of the theory.

A framework for the problem based on oriented projective geometry (which allows unbounded polyhedra to be treated as bounded) was presented by Burton and Ozlen [35]. The framework when incorporated into Benson's outer approximation algorithm resulted in a new algorithm that works in an oriented projective space. Numerical experiments show that there is a reduction in the number of inefficient vertices and running time.

Löhne [100] presented a detailed and comprehensive account of set-valued and vector-valued approaches to the problem in the objective space. The exposition covers the general theory of MOLP, scalar-

ization techniques, duality and extended variants of Benson's [30] algorithm to mention but a few. Most importantly, a solution concept that takes into account the polyhedral structure of the problem was introduced. The author suggested that a solution of the problem should be a finite subset of the feasible set which consists of not only the set of efficient or nondominated points but also extreme directions.

Recently, Shao and Ehrgott [138] extended the primal and dual variants of the algorithm in [30] to solve linear multiplicative programming problems. The authors first improved their primal objective space algorithm in [137] which was also based on the algorithm of Benson [30] by making it solve only one LP in each iteration. Thereafter, the dual variant in [57] was sandwiched to obtain a modified dual objective space algorithm that also solve one LP in each iteration. Numerical results suggests that the proposed algorithm is superior to the global optimisation algorithm presented in [72].

2.3 Interactive Algorithms

2.3.1 Interactive simplex based algorithms

Having discussed the non-interactive solution approaches, we now turn our attention to the interactive algorithms that seek to find a most preferred solution to the problem. These methods consists of two main phases: a computational phase where efficient solutions are computed and a dialogue phase where the DM is required to express his or her preferences to guide the solution process. Interactivity in our view is an essential feature of usable tools. It enhances applicability and robustness of methods on one hand, but hinders them on the other by the mere fact that intervention is required.

The earliest to propose an interactive approach to the problem seems to be Benayoun *et al.* [25]. They called their procedure Step Method (STEM). Here, a payoff table is first constructed in order to obtain the ideal (optimum) solution for each of the objectives under consideration. In the computation phase, a weighted Tchebyshev distance to the ideal solution is solved to generate efficient solutions which are them presented to the DM to decide which is most preferred. Oth-

erwise, the DM specifies the maximum or minimum amount he/she is willing to sacrifice in one objective in order to improve the other objectives. The information supplied by the DM is transformed into additional constraints thereby altering the feasible region and objective values. This process continues until a most preferred solution is obtained.

Few years later, Zionts and Wallenius [166] presented an interactive method for solving the problem using a linear utility function instead of a payoff table. It was noted however that, the STEM algorithm in [25] may not work well in solving relatively small problems. Their method starts with arbitrary set of weights which are used to obtain a weighted sum objective function. This function is then optimized to obtain efficient solutions. The DM provides answers to certain questions in order to determine the trade off used to construct linear approximations of the utility function. Based on the answers provided by the DM, a new set of weights with associated efficient solutions are found. This continues repeatedly until a most preferred solution is found. In a similar paper, Zionts and Wallenius [167] extended their interactive approach in [166] to a class of nonlinear utility functions.

In [31], Benson *et al.* applied the STEM algorithm presented in [25] to solve the citrus rootstock selection problem in [26]. The algorithm returns unsatisfactory results when the payoff table approach was used as suggested in [25]. Benson-Sayin [32] global optimization heuristic was then applied to solve the problem. It was noted however, that the use of appropriate global optimization methods is crucial to the achievement of success in real-world problems and the use of unreliable payoff table approach in applied MOLP should be discouraged.

Arbel and Oren [14] also presented an interactive approach for the problem. Unlike the methods in [166] and [167] that utilize linear and nonlinear utility functions, here, AHP was used to assign priorities to vertices adjacent to the one representing the current basic solution to possibly improve the objective function values. These priorities were used to obtain an approximate gradient for weighing the objective functions in the next iteration. Here, the DM is not required to provide inputs for determining tradeoff, rather he or she evaluates adjacent vertices for possible improvements until no adjacent vertex is preferred to the current one. It was noted however, that the approach produces only approximate solutions.

Steuer [141] suggested an interactive algorithm which utilizes criterion weights specified by the DM in the dialogue phase to obtain a subset of efficient solutions. The procedure solved the resulting weighted sum problem to determine a subset of efficient solutions corresponding to the specified weights. These solutions are then presented to the DM to choose that which is most preferred. This approach, however, may produce a large subset of efficient solutions which may be difficult for the DM to choose the most preferred. In [142], Steuer improved on the approach in [141] to further reduce the subset of efficient solutions presented to the DM.

The interactive approach of Choi and Kim [38] provides the set of nondominated solutions for large instances. The procedure solves a Tchebycheff error problem to determine the closest point on a face of the feasible region from the reference point. It was reported that the method ensures a full coverage of the nondominated points and reduces the DM's burden in obtaining a most preferred solution.

In [145], Stewart presented an interactive method for solving the problem. It was noted that, the method of Zionts and Wallenius [166] does not specify how an interactive search over faces of the simplex

was to be carried out. However, his method avoid the unspecified search and use a single procedure that requires only pairwise preference statements from the DM.

A Simplified Interactive MOLP (SIMOLP) procedure for the problem was presented in [122]. This method does not require a payoff or utility function and starts by solving the single objective LP problems involved, to obtain a set of efficient points and the associated nondominated points. These points are then presented to the DM for a review. If the DM is satisfied with any of them, the process stops; otherwise, an augmented LP is solved to obtain an efficient solution and a corresponding nondominated point which is again presented to the DM for further review. The procedure continues until a most preferred solution is achieved. Numerical illustrations show that a most preferred solution can be achieved in a relatively few number of iterations.

Korhonen and Laakso [97] proposed a visual interactive algorithm to solve the problem using an unknown utility function to simulate the DM's behaviour. The approach utilizes aspiration levels or reference direction to determine the direction of improvement that is projected

on the efficient frontier in order to obtain a subset of efficient solutions. The corresponding objective values are then presented to the DM for evaluation. It was noted however, that the interface adopted here was based on a static graphic representation and could make the DM rely too much on the system.

In [96], Korhonen and Wallenius improved on the algorithm in [97] by presenting its dynamic version called Pareto race. Here, the DM provides preference information that guide the search process to the efficient frontier and thereby improving the objective values.

Haksever and Ringuest [76] conducted an investigation into the computing efficiency of four variants of SIMOLP [122]. Numerical results suggests that solving each successive LP using the optimal basis of the previous problem is an effective way of achieving a reduction in the number of iterations and CPU time.

In [107], Malakooti and Ravindran developed an interactive paired comparison simplex method for the problem. The approach assumes that the DM has an unknown utility function and performs a test for efficiency to reduce the number of efficient solutions and questions for the DM. This method was compared with that of Zionts and

Wallenius [166]. It was reported that the proposed method is superior in all criteria selected for evaluation.

Dell and Karwan [51] proposed an interactive procedure using a Tchebycheff utility function. This procedure was compared with the method of Zionts and Wallenius [166]. Computational results suggests that the procedure performs slightly better than that of Zionts and Wallenius in terms of the quality of solution it returns. However, Zionts and Wallenius method was computationally more efficient than the proposed method.

The interactive method of Lotfi *et al.* [105] utilizes the DM's aspiration levels and a Tchebycheff function. The approach facilitates the achievement of a most preferred solution at a non-extreme point. The approach was compared with SIMOLP procedure [122]. Experimental results show that the proposed method compared favourably in terms of the quality of solutions it returns. However, the SIMOLP procedure outperforms the proposed method in terms of computing efficiency.

Michalowski and Szapiro [111] presented a Bi-reference interactive procedure for the problem. Here, the DM is required to specify his or

her worst outcome, from where an ideal outcome is determined. The algorithm then construct an improvement direction from the worst to ideal outcome, leading to a trial solution. The DM is expected to partition the objectives into that which require to be improved, unchanged and relaxed. This action would lead to the displacement of outcomes and a new improvement direction is obtained that yields a most preferred solution. The method was compared with STEM algorithm of Benayoun *et al.* [25] and that of Zionts and Wallenius [167]. Numerical results show that the Bi-reference procedure is superior to both methods by returning preferred solutions in a few number of iterations.

The interactive method of Quaddus and Holzman [120] assumes that the DM has an implicitly known utility function. The approach is similar to that in [166]. The DM is also required to specify a set of weights used to construct a weighted sum objective which is then optimized to obtain a starting efficient solution. Here, each objective function is evaluated using the starting efficient point and the solutions presented to the DM for evaluation. The procedure is repeated until a most preferred solution is obtain. It was noted that the approach is more flexible and require minimum interaction than that

in [166].

In [39], a tricriteria algorithm (TRIMAP) for the problem was introduced. The algorithm was not aimed at finding a most preferred solution, but to assist the DM discard the subsets of efficient solutions which are of no use. The method employs a progressive and selective learning of the efficient set, until the DM has adequate knowledge of the set. The DM is required to provide lower bounds for the objectives which are graphically transformed into the weight space diagram to visually limit the search scope. The information provided by the DM also guides the search process to unexplored regions of the diagram until a good knowledge of the efficient set is achieved. This algorithm is implemented as a software in [40]. The program is implemented in Pascal and consists of three phases: the weight space decomposition phase, simplex and routine implementation phases. During the interactive process, a graphical representation of the weight space with regions corresponding to efficient vertices and their associated objective values are presented to the DM for evaluation. It was noted however, that the software may only solve problems with three objectives.

Recently, Alves *et al.* [3] introduced an interactive graphical-based computational tool for teaching, research and decision support purposes in MOLP. The software is equipped with interactive approaches such as: STEM, Pareto race, interval criterion weight and scalarization techniques: weighted sum, reference point and e-constraint techniques as well as a Vector Maximization Approach (VMA) that generates all efficient extreme points of the problem. Apart from the VMA which is noninteractive, a brief description of the interactive approaches is also included. Visualization of results as well as the graphical display of regions on the weight space diagram for problems with up to three objectives can also be done. It was noted however, that the solver is limited to solving problems with a maximum of six objectives, a hundred constraints and decision variables.

2.3.2 Interactive Interior-point based algorithms

Various interactive interior-point approaches has been suggested for the problem. These methods differ in the methodology adopted to elicit preference information from the DM that guide the search or solution process.

The first to present an interactive interior-point approach to the problem appears to be Arbel and Oren [15]. Their method was based on ASPA and assumes that the DM has an implicitly known utility function. The approach generates search directions used to approximate the gradient of the utility function. The DM is required to provide preference information that is used to update the approximated gradient, thereby resulting in a sequence of iterates. The iterative process continues until a most preferred solution is achieved.

Another of Arbel's [10] suggestions is an interactive path-following Primal-Dual Interior Multiple Objective Linear Programming (PDIMOLP) algorithm. The difference between ASIMOLP and PDIMOLP is in the way their interior step directions are generated. While ASIMOLP is concerned with cost reduction, PDIMOLP includes a centring scheme that keeps the current iterate centred in the constraints polytope.

In [12], Arbel and Korhonen introduced another interactive algorithm as above. This algorithm combines path-following primal-dual process with Achievement Scalarizing Function (ASF) first introduced in [153]. The DM is required to specify his or her aspiration levels of

the objectives and the ASF depends on these aspiration levels. The algorithm then generates a path from the current iterate to the optimum of the ASF. As the DM continue to change his or her aspiration levels, a new iterate is generated that is closer to the optimum of the ASF. This process continues until a most preferred solution is found.

Arbel [11] suggested yet another interactive PDIMOLP algorithm for the problem. This approach is quite similar to that in [15]. The difference between them is, while the former is based on ASPA, the latter is based on the path-following primal-dual process.

In a different paper, Arbel and Oren [17] proposed a modification of the path-following primal-dual algorithm. The authors applied AHP to derive priorities that are used to approximate the gradient of the DM's utility function. Projecting the approximated gradient onto the null space of the constraints matrix, provides the combined step direction along which one move from the current iterate to a new one. By taking a full step along the combined direction enables the algorithm to converge to a most preferred efficient solution on the boundary of the feasible region.

Aghezzaf and Ouaderhman [2] also presented an interactive interior-

point algorithm where an implicit utility function is known. Here, the DM is required to provide trade-offs used to derive an approximate gradient of the utility function as well as updates the lower bounds of the objectives. The algorithm then generates a sequence of smaller polytopes that tend to shrink towards a most preferred solution.

In [86], Junior and Lins suggested an interactive interior-point approach that is based on ASPA and makes explicit use of ASF. The ASF was formulated in such a way that the algorithm follows a path that avoid sudden changes in the search direction so as to exhibit a win-win property. The DM is required to specify the aspiration levels for each objective as well as the proportion among other objectives. In each iteration, intermediate solutions and objective values are presented to the DM for evaluation. It was noted however, that the method allows the DM to reach a most preferred solution without making trade-offs between objectives.

Trafalis and Alkahtani [150] proposed an interactive interior-point algorithm for the problem based on the method of analytic centers. Here, the DM is required to supply his or her trade-offs which are used to obtain a cut in the objective space and this induces a corresponding

cut in the variable space. The analytic center of the feasible region is then constructed and the DM's trade-offs are determined in the analytic center of the objective space, thereby obtaining a trajectory of analytic centers which converges to a most preferred solution.

2.4 Heuristic approaches to multi-objective optimisation

Having reviewed exact methods to the problem, we now turn our attention to heuristics or approximate approaches to MOO problems in general. As stated earlier, heuristics or approximate methods have been commonly applied to nonlinear and discrete multi-objective optimisation and not so much to MOLP.

In [37], Chakraborty and Ray applied multi-objective parametric fuzzy programming and NSGA [140] to MOLP transportation problem. Here, the MOLP problem is transformed into a single objective parametric problem with interval parameters. Numerical illustration using a coal energy resource allocation problem show the applicability of the method. NSGA has been widely applied in different discrete and continuous multi-objective optimisation problems. In [21],

Bagchi applied NSGA to several scheduling problems for which approximate efficient solutions were found. For extensive applications of NSGA to chemical engineering problems, see [112]. NSGA-II [49, 50] which is an improved version of NSGA [140] and arguably the most popular in the context of nonlinear multi-objective problems has also had tremendous applications in different nonlinear multi-objective optimisation. It was successfully applied in the energy generation expansion planning problem in [88] for which the minimum investment and outage costs were approximated. Similarly, Hu *et al.* [82] also applied NSGA-II to a real-life combined gas and electricity network expansion planning problem in Hainan province (China) with an aim of minimizing investment, production and carbon emission costs. The problem was formulated as a bicriterion nonlinear MOO problem and solved using NSGA-II for which the nondominated front was approximated. In [109], Massobrio *et al.* applied NSGA-II to the taxi sharing problem in order to determine the minimum cost of journey and delay time by passengers from the same location to different destinations. The problem was formulated as a bicriteria multi-objective optimisation problem and solved with NSGA-II and greedy heuristics. Numerical results show that NSGA-II outperform the greedy

algorithms by achieving significant improvements in both objectives in acceptable computational time. Recently, NSGA-II was applied in the communication industry to solve the spectrum assignment problem in [108]. Here, the spectrum assignment problem was also formulated as a bicriterion MOO problem and solved using NSGA-II. Experimental results show that there is an improvement in throughput at the cost of spectral efficiency which offers useful guidelines to the service provider to maintain customer satisfaction in the spectrum sharing network. In [52], a modification of NSGA-II [49, 50] was presented and applied to the combined economic and emission dispatch problem. It was noted however, that NSGA-II ensures diversity along the nondominated front using the concept of crowding distance, but lateral diversity may be lost due to the lack of diversity in a particular decision variable which may push the search towards the nondominated front. The modified version resolved this issue by incorporating controlled elitism into NSGA-II and replaced the crowding distance operator with a dynamic version which proves effective when solving problems.

Salhi and Fraga [128] presented the plant propagation algorithm (PPA). This is the so called PPA. The algorithm emulates the way

plants and in particular, the strawberry plant propagate by sending many short runners when they are in good spots and fewer but longer runners to explore the environment when they are in a not so good spot. It was tested on a complex nonlinear process design problem and compared with the Nelder-Mead algorithm. Experimental results show the effectiveness of the proposed method, and its significantly outperforms the Nelder-Mead search method. In [67], Fraga and Amusat extended the PPA [128] to solve multi-objective nonlinear programming problems. A novel fitness function that emphasizes the end-points is introduced into the extended version and applied to the integrated energy systems design for off-grid mining operations problem for which good designs that achieve the desired objectives were approximated. Recently, Rodman *et al.* [123] applied the extended PPA (MOPPA) to the industrial beer fermentation process in order to minimize the production time and maximize ethanol production. The problem was modelled as a bicriteria nonlinear dynamic optimisation problem and solved with MOPPA. Numerical results show the effectiveness of MOPPA in solving complex multi-objective optimisation problems.

2.5 Summary

In this chapter, we have reviewed MOLP papers that have appeared since 1964. The survey classifies MOLP algorithms into two broad classes: Non-Interactive and Interactive algorithms. The Non-Interactive algorithms include simplex based, interior-point based and objective space based methods. The Interactive algorithms only include simplex and interior-point based methods. We have also presented a tabulated list of all algorithms reviewed during the period. This in the form of many tables is presented in Appendices A through E. Appendix A records the non-interactive simplex based methods; Appendix B records the interactive simplex based methods; the objective space based methods are recorded in Appendix C; Appendix D records the non-interactive interior-point based methods and Appendix E records the interactive interior-point based methods. Each of these appendices is organised chronologically. In the next chapter, we shall present an extension of MSA of Evans and Steuer [63] to compute the set of all nondominated points. The extended version will be compared with the original one and with BOA that also computes the set of all nondominated points of the problem.

Chapter 3

A COMPARISON OF BOA WITH AN EXTENDED VERSION OF MSA

3.1 Introduction

Many algorithms have been suggested for MOLP in the last few decades as can be seen from the previous chapter. Most of them are based on the simplex method for Linear Programming. Prominent among them is the Multiobjective Simplex Algorithm (MSA) and its variants. MSA and its variants work in the decision variable space and find the set of all efficient solutions of the problem. Current solution approaches finds either the entire set of all efficient solutions or a subset of them and also return the corresponding non-

dominated points. According to Schechter and Steuer [131], the MSA of Evans and Steuer [63] is the most popular and successful for computing all the efficient solutions of the problem. However, it is well known [28–30, 44, 46, 57, 59, 102, 135, 136] that in practice, the DM prefers to base his or her choice of a most preferred solution on the objective values (nondominated points) rather than the efficient set [30] and moreover, it was noted in [59] that finding the nondominated set instead of the efficient set is more important for the DM. This chapter presents an extension of the MSA of Evans and Steuer [63] to generate the set of all nondominated points of the problem.

We reiterate here that the chapter extends the MSA of Evans and Steuer [63] whose explicit form can be found in [56] to generate the whole set of nondominated points devoid of redundant nondominated points. We shall then compare the extended version with the original one as well as with the primal variant of BOA [30] which is an objective space based method that also computes the set of all nondominated points of the problem.

From the outset, MSA and BOA seem not comparable since one is decision space based while the other is objective space based; one com-

putes efficient solutions and the other nondominated points. However, if one can generate nondominated points from MSA then this can be performed.

This chapter is organized as follows. Sections 3.2 and 3.3 presents MSA and its extended version respectively. We discuss two scalarization techniques in Section 3.4. BOA is presented in Section 3.5. Section 3.6 presents experimental results obtained with the different algorithms. Section 3.7 present the summary of results. Finally, a summary of the chapter is presented in Section 3.8.

3.2 The Multi-objective Simplex Algorithm

A typical multiple objective simplex algorithm is that of Evans and Steuer [63]. The version described here can be found in [56], page 178. We consider this algorithm because of its popularity (see [131]), and because most of the MSA algorithms discussed earlier are either based on or are variants of it. It works in the decision space and finds the set of all efficient extreme points.

In an MOLP problem, only one of the following situations can occur:

the problem can be infeasible, meaning that the feasible set X is empty ($X = \emptyset$); the problem may be feasible, that is ($X \neq \emptyset$) but may not have efficient solutions, that is, ($X_E = \emptyset$); or it is feasible and has efficient solutions, that is $X_E \neq \emptyset$. This algorithm handles these situations in three phases: In the first phase, it finds an initial basic feasible solution or stop with the conclusion that $X = \emptyset$; in the second phase, it finds an initial efficient basis or stop with the conclusion that $X_E = \emptyset$; and in the final phase it pivots among efficient bases to determine all efficient extreme points of the problem, [56].

The algorithm is initialized by solving two auxiliary LPs to determine whether the problem is feasible and to verify that it has efficient solutions. If the feasible region X is not empty and the set of efficient extreme points X_E exists, a weighted sum LP is solved to determine an initial efficient basis B. Its implementation stores a list of efficient bases L_1 to be processed, a list L_2 of efficient bases for output, and a list of efficient nonbasic variables N_E. An LP test problem is solved to determine pivots that lead to efficient bases. The algorithm pivots from an initial efficient basis to an adjacent efficient basis until the list L_1 to be processed is empty. The algorithm stops and returns list L_2 from where all efficient extreme points are computed. Before we

describe MSA in pseudo-code form, we first explain the used notation.

Notation: A, b, C form the problem data; L_1 and L_2 as above; $e^T = (1, ..., 1) \in \mathbb{R}^q$; I is the identity matrix of proper order; X is the feasible set; X_E, the set of efficient solutions; B, the efficient basis; N_E, a list of efficient nonbasic variables; N, the set of nonbasic variables; $B^{'}$, the new basis; $\bar{A}$, and $\bar{b}$ are updated constraint matrix and RHS vector; R is the nonbasic part of the reduced cost matrix and r^j is a column of R corresponding to a nonbasic variable being tested for efficiency.

Algorithm 3 Multi-objective Simplex Algorithm

Input: A, b, C : *Problem data*

Initialize: *Set* $L_1 \leftarrow \emptyset,\ L_2 \leftarrow \emptyset$;

Phase I : *Solve the LP* $min\{e^T z : Ax + Iz = b, x, z \geq 0\}$. *If the optimal value of this LP is nonzero, STOP,* $X = \emptyset$;
Otherwise x^0 *is a basic feasible solution of MOLP*

Phase II : *Solve the LP* $min\{u^T b + w^T C x^0 : u^T A + w^T C \geq 0, w \geq e\}$. *If it is infeasible, STOP,* $X_E = \emptyset$;
Otherwise $(\hat{u}, \hat{w})$ *is an optimal solution*;
Find an optimal basis B *of the LP* $min\{\hat{w}^T C x : Ax = b, x \geq 0\}$;
Set $L_1 \leftarrow \{B\},\ L_2 \leftarrow \emptyset$.;

while $L_1 \neq \emptyset$
  Choose $B \in L_1$, $L_1 \leftarrow L_1 \backslash \{B\}, L_2 \leftarrow L_2 \bigcup \{B\}$;
  Compute $\tilde{A}$, $\tilde{b}$, *and* R *according to* B;
  $N_E \leftarrow N$;
  for *all* $j \in N$
    Solve the LP $max\{e^T v : Rz - r^j \sigma + Iv = 0; y, \sigma, v \geq 0\}$.
    If this LP is unbounded $N_E \leftarrow N_E \backslash \{j\}$;
    for *all* $j \in N_E$
      for *all* $i \in B$
        if $B' \leftarrow (B \backslash \{i\}) \bigcup \{j\}$ *is feasible*, $B' \notin L_1 \bigcup L_2$ **then;**
          $L_1 \leftarrow L_1 \bigcup B'$;
        endif;
      endfor;
    endfor;
  endfor;
endwhile.

Output: L_2 : *List of efficient bases.*

3.2.1 Illustration of MSA

Consider the following MOLP adapted from [86]. We solve this problem using a Matlab implementation of Algorithm 3 provided by the authors of [124].

$$
\begin{aligned}
&\min f_1 = -x_1 \\
&\min f_2 = \qquad -x_2 \\
&\text{Subject to} \\
&\qquad 6x_1 + 10x_2 \leq 60 \\
&\qquad x_1 \qquad\quad \leq 7 \\
&\qquad\qquad x_2 \leq 5 \\
&\qquad x_1, \quad x_2 \geq 0
\end{aligned}
\tag{3.1}
$$

The efficient solutions found are $x^1 = (7.0,\ 1.8)^T$, $x^2 = (1.6,\ 5.0)^T$, $x^3 = (1.6,\ 5.0)^T$, $x^4 = (7.0,\ 1.8)^T$. Where $x^1 = (x_1^1, x_2^1)^T$,..., $x^4 = (x_1^4, x_2^4)^T \in X_E$. The algorithm is prone to generating more efficient solutions due to the way it operates and due to the fact that it may find the same efficient solution in more than one iteration, as in this case; $x^1 = x^4$ and $x^2 = x^3$ are repetitive of what has already been found. Solutions x^3 and x^4 are redundant, and would be of little or no use to the DM. The feasible region in the decision space is shown in Figure 3.1.

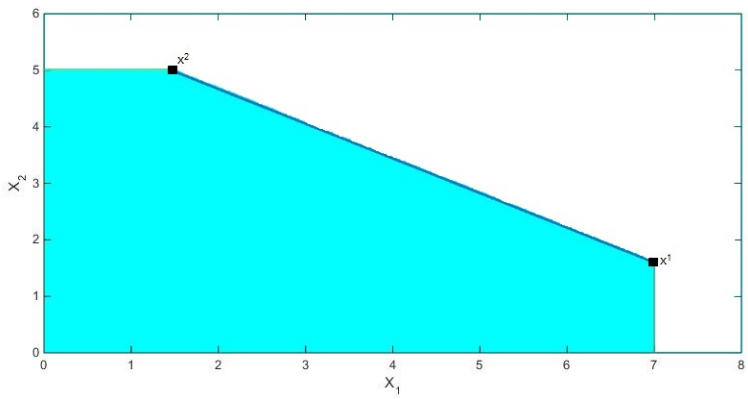

Figure 3.1: Efficient edge of the feasible region connecting two points in the decision space.

3.3 The Extended Multi-objective Simplex Algorithm

As part of the initialization step (line 1 of Algorithn 4), we have included the set of efficient extreme points X_E and that of nondominated points Y_N. In the second phase, as the algorithm finds an initial efficient basis B by solving a weighted sum LP, the algorithm also finds a corresponding efficient basic feasible solution and appends it to the set of efficient extreme points ($X_E \leftarrow \{\bar{x}\}$). The first nondominated point is also computed from $C^T\bar{x}$ and appended to the nondominated set ($Y_N \leftarrow \{C^T\bar{x}\}$).

As the algorithm iterates, a new efficient basis $B^{'}$ is obtained after each pivot and the corresponding efficient basic feasible solution $\bar{x}^{'}$ (line 11 of Algorithm 4) is found and added to the set of efficient extreme points X_E (line 13). Likewise, the corresponding nondominated points are also found at each iteration and added to the nondominated set Y_N (line 14). This continues until the set of efficient bases L_1 to be processed is empty. The algorithm returns the set of all efficient extreme points and the corresponding nondominated points (line 20).

Before we present Algorithm 4 as the extended MSA in pseudo-code form, we first state here that the structure of the algorithm and the used notation remain the same as that in Algorithm 2. The additional components are $\bar{x}$, $\bar{x}^{'}$, X_E and Y_N which stand for the efficient basic feasible solution, the new efficient basic feasible solution, the set of efficient extreme points and the corresponding set of nondominated points for output.

Algorithm 4 Extended Multi-objective Simplex Algorithm

Input: A, b, C $(data\ of\ MOLP\ problem)$

Initialize: $Set\ L_1 \leftarrow \emptyset,\ L_2 \leftarrow \emptyset,\ X_E \leftarrow \emptyset,\ Y_N \leftarrow \emptyset;$

$Phase\ I:\ Solve\ the\ LP\ min\{e^T z : Ax + Iz = b, x, z \geq 0\}.\ If\ the\ optimal\ value$
$of\ LP\ is\ not\ zero,\ STOP,\ X = \emptyset;$
$Otherwise\ let\ x^0\ be\ a\ basic\ feasible\ solution\ of\ MOLP$

$Phase\ II:\ Solve\ the\ LP\ min\{u^T b + w^T C x^0 : u^T A + w^T C \geq 0, w \geq e\}.\ If$
$it\ is\ infeasible,\ STOP,\ X_E = \emptyset.\ Otherwise\ let\ (\hat{u}, \hat{w})\ be\ an$
$optimal\ solution.\ Find\ optimal\ basis\ B\ and$
$basic\ feasible\ solution\ \bar{x}\ of\ LP\ min\{\hat{w}^T C x : Ax = b, x \geq 0\};$
$Set\ L_1 \leftarrow \{B\},\ L_2 \leftarrow \emptyset,\ X_E \leftarrow \{\bar{x}\},\ Y_N \leftarrow \{C^T \bar{x}\};$

while $L_1 \neq \emptyset$ **do**
  $Choose\ B \in L_1,\ L_1 \leftarrow L_1 \backslash \{B\}, L_2 \leftarrow L_2 \cup \{B\};$
  $Compute\ \tilde{A},\ \tilde{b},\ and\ R\ according\ to\ B;$
  $N_E \leftarrow N;$
  for $all\ j \in N$
    $Solve\ the\ LP\ max\{e^T v : Rz - r^j \sigma + Iv = 0; z, \sigma, v \geq 0\}.$
    $If\ this\ LP\ is\ unbounded\ N_E \leftarrow N_E \backslash \{j\};$
    for $all\ j \in N_E$
      for $all\ i \in B$
        if $B' \leftarrow (B \backslash \{i\}) \cup \{j\}\ is\ feasible\ , B' \notin L_1 \cup L_2,\ let\ \bar{x}'\ be\ its\ basic\ solution$ **then;**
          $L_1 \leftarrow L_1 \cup B';$
          $X_E \leftarrow X_E \cup \{\bar{x}'\};$
          $Y_N \leftarrow Y_N \cup \{C^T \bar{x}'\};$
        endif
      endfor
    endfor
  endfor
endwhile

Output: $X_E:\ The\ efficient\ set$
$Y_N:\ The\ nondominated\ set.$

3.3.1 Illustration of the Extended MSA

We modified and extended the Matlab implementation of Algorithm 3 provided by the authors of [124] and used it to solve problem 3.1 of Section 3.2.1. The efficient extreme points found are $x^1 =$

$(7.0,\ 1.8)^T$, $x^2 = (1.6,\ 5.0)^T$, and the corresponding nondominated points are $f^1 = (-7.0,\ -1.8)^T$ and $f^2 = (-1.6,\ -5.0)^T$ respectively. Where $x^1 = (x_1^1, x_2^1)^T$, $x^2 = (x_1^2, x_2^2)^T \in X_E$ and $f^1 = (f_1^1, f_2^1)^T$, $f^2 = (f_1^2, f_2^2)^T \in Y_N$. Notice here that, the efficient extreme points x^1, x^2 and the corresponding nondominated points f^1 and f^2 returned are devoid of redundant points. The algorithm is designed to avoid returning redundant efficient and nondominated points unlike the original version. The feasible region in the decision space is the same as in Figure 3.1.

3.4 Scalarization Techniques

Before presenting BOA, we first present two basic scalarization methods that play an important role in its implementation. These methods are weighted sum scalarization and translative or scalarization by a reference variable. As noted in [100], scalarization is one of the most important techniques used in MOLP.

In the weighted sum method, a new objective function based on the q-linear objectives is obtained by assigning non-negative weights $w_i \in \mathbb{R}^q$ to each of the objectives. The weighted sum of the objectives is

$\sum_{i=1}^{q} w_i c_i x = w^T C x$. For each vector $w \in \mathbb{R}^q$, $w \geq 0$, we obtain a scalar linear program

$$\min w^T C x \quad \text{subject to } Ax \geq b \qquad P_1(w)$$

The weights are usually normalized so that $e^T w = 1$, with $e^T = (1, ..., 1)$. The dual of $P_1(w)$ is

$$\max b^T u \text{ subject to } \begin{cases} A^T u = C^T w \\ u \geq 0 \end{cases} \qquad D_1(w)$$

In the method of scalarization by a reference variable, the q objectives are associated to a common reference variable z and the i-th objective is restrained from being larger than the reference variable and a fixed real number y_i, that is $c_1 x \leq y_1 + z$, $c_2 x \leq y_2 + z$, $\ldots$, $c_q x \leq y_q + z$.

The reference variable z is the objective function that has to be minimized. By setting $e = (1, ..., 1)^T$, we obtain for each vector $y \in \mathbb{R}^q$

the scalar linear program

$$\min z \qquad \text{subject to} \begin{cases} Ax \geq b \\ Cx - ze \leq y. \end{cases} \qquad P_2(y)$$

The dual program is

$$\max b^T u - y^T w \ \text{ subject to} \begin{cases} A^T u - C^T w = 0 \\ e^T w = 1 \\ (u, w) \geq 0. \end{cases} \qquad D_2(y)$$

[100]. The above two scalarization techniques are fundamental for the implementation of BOA which is discussed in the next section.

3.5 Benson's Outer-Approximation Algorithm

This version of BOA is due to Shao & Ehrgott [135]. It can be found in [100]. It works in the objective space of the problem and returns the set of all nondominated points and extreme directions. The algorithm can be regarded as a primal-dual method because it also solves the dual problem. But here, we are only concerned with the solution of the primal. The algorithm first constructs an initial polyhedron

Y_0 (outer-approximation) containing the upper image Y in the objective space and an interior point $\hat{p}$ of the image is determined by solving $P_1(w)$. The inequality representation of the outer approximation is also determined by solving $D_1(w)$. The algorithm construct a sequence of decreasing polytopes $Y_0 \supseteq Y_1 \supseteq \ldots \supseteq Y_k = Y$. The vertices of each polytope Y_k as well as inequality representation (facets) are stored in each iteration. Then for each vertex v of the polytope, the algorithm checks if the vertex is on the boundary of Y. If the vertices are on it, the problem is solved. The external vertices of Y are among the vertices of Y_k. Otherwise, for any vertex v of Y_k that is not on the boundary of Y, the algorithm connects this vertex to the interior point $\hat{p}$ and finds the intersection y of this line with the boundary of Y by solving $P_2(v)$. Then a supporting hyperplane adjacent to y is constructed by solving $D_2(y)$. This hyperplane is added to Y_k to provide a smaller approximation. The algorithm is repeated in the same way until the vertices of Y_k coincide with the boundary of Y. The algorithm returns the set of vertices on the boundary of Y as the nondominated set $\bar{Y}$ and directions $\bar{Y^h}$ of the problem. The notation used in the pseudo-code of BOA is as follows.

Notation: A, b, C are the problem data; P^h is the homogeneous

problem; D^{*h} is the homogeneous dual problem; $\bar{T}^h$ is the solution of the homogeneous dual problem; $\hat{p}$ is an interior point; $\bar{T}$ is a set of solutions of the dual problem; Y_k^d is the inequality representation of the current polytope; k is the iteration counter; Y_k^p is the representation by vertices; $(\hat{y}, z)$ is an optimal solution to $P_2(v)$; $\delta(0 < \delta < 1)$ is a unique value that determines the intersection or boundary point y; $R(v)$ is the LP that finds the unique value δ; the command $solve()$ solves an LP; $vert()$ returns the vertices of a polytope Y_k; $\bar{Y}$ is the set of nondominated vertices; $(\bar{Y}^h)$ is the set of extreme directions.

Algorithm 5 Benson's Outer-Approximation Algorithm

0: Input: A, b, C : *Problem data*
a solution $(\{0\}, \bar{Y}^h)$ *to* P^h;
a solution $\bar{T}^h$ *to* D^{*h};
1: Initialize: $\hat{p} \leftarrow \mathcal{P}(solve(P_1(0))) + e$;
2: $\bar{T} \leftarrow \{(solve(D_1(w)), w) | (u, w) \in \bar{T}^h\}$;
3: while $z = 0$ **do**
4: $Y_k^d \leftarrow \{D^*(u, w) | (u, w) \in \bar{T}\}$;
5: $Y_k^p \leftarrow vert(Y^d)$;
6: $\bar{Y} \leftarrow \emptyset$;
7: **for** $i = 1$ *to* $|Y^p|$ **do**
8: $v \leftarrow Y_k^p[i]$;
9: $(\hat{y}, z) \leftarrow solve(P_2(v))$;
10: $\bar{Y} \leftarrow \bar{Y} \cup \{\hat{y}\}$;
11: **if** $z \neq 0$ **then**
12: $(x, \delta) \leftarrow solve(R(v))$, $(0 < \delta < 1)$;
13: $y \leftarrow \delta v + (1 - \delta)\hat{p}$;
14: $(u, w) \leftarrow solve(D_2(y))$;
15: $\bar{T} \leftarrow \bar{T} \cup \{(u, w)\}$;
16: **endif;**
17: **endfor;**
18: endwhile
19 Output: $(\bar{Y}, \bar{Y^h})$: *Nondominated set and directions*;
$\bar{T}$: *a solution to dual.*

3.5.1 Illustration of Benson's Outer-Approximation Algorithm

For continuity, we consider again problem 3.1 of Section 3.2.1 The nondominated points found using an existing Matlab implementation of Algorithm 5, namely Bensolve-1.2, [101], are $f^1 = (-1.6, -5.0)^T$ and $f^2 = (-7.0, -1.8)^T$ where f^1 and $f^2 \in Y_N$. These nondominated points are shown in Figure 3.2.

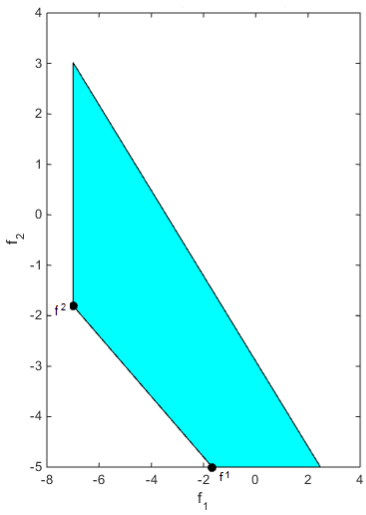

Figure 3.2: The edge joining the two nondominated points in the objective space.

3.6 Discussion of Experimental Results

In this section, we provide numerical results to study the computational efficiency of Algorithms 3, 4 and 5, and the number of non-dominated points returned by Algorithms 4 and 5.

Table 3.2 shows the numerical results for a collection of 55 problems from existing literature, ranging from small to moderate size MOLP instances. Problem 1 is taken from Ehrgott [56], and Problems 2 to 10 were taken from Zeleny [163]. Problems 11 to 21 are test problems from the interactive MOLP explorer (iMOLPe) of Alves *et al.* [3]. Problems 22 to 47 are taken from Steuer [143]. Problem 48 is a test problem in Bensolve-1.2 [101], while problems 49 and 53 are test problems in Bensolve-2.0 [104]. Problems 50 to 52 are obtained using a script in Bensolve-2.0 [104] that is used to generate problem 53 with the same number of variables and constraints. Finally, problems 54 and 55 are test problems in MOPLIB [103] which stands for Multi-Objective Problem Library.

Note that all the problems in Table 3.2 are non-degenerate. Problem 48 is such that the constraint matrix is sparse while the criterion

matrix is dense. The RHS vector is such that all the components are ones except for 200 at the end as the largest entry. Problem 49 has a dense constraint matrix with an identity matrix of order n as its criterion matrix where n is the number of variables in the problem. The RHS vector is such that all the components are zeros except for a one (1) at the begining as the only none zero element. Problems 50 to 53 have dense criterion matrices with identity matrices of order n as their constraint matrices where n is also the number of variables in the respective problem. All the elements in the RHS vectors are ones. Finally, Problems 54 and 55 have sparse constraints and criterion matrices with dense RHS vectors.

Results for Algorithm 3 were obtained using a Matlab implementation of this algorithm provided by the authors of [124]. We modified and extended Algorithm 3 of Evans and Steuer [63] into Algorithm 4 or EMSA the Extended Multi-objective Simplex Algorithm. We have implemented it in Matlab in the same way as in [124] and experimented with it on the test problems mentioned above. We also used an existing Matlab implementation of Algorithm 5 known as Bensolve-1.2 [101] to obtain results for this algorithm. The current version, Bensolve-2.0, [104] is implemented in the C programming

language. We employed Bensolve-1.2 [101] which is implemented in Matlab so as to test the algorithms with the same tools and for a meaningful comparison. All algorithms were executed on an Intel Core $i5$-2500 CPU at 3.30GHz with 16.0GB RAM. In all tests, n is the number of variables, m the number of constraints and q the number of objectives. Algorithm 3 is MSA of Evans and Steuer [63], Algorithm 4 its extended version and Algorithm 5 is BOA as presented in [135]. We recorded the CPU times (in seconds) returned by the algorithms for each problem. We also recorded the Number of Efficient Solutions (NES) returned by MSA, the Number of Nondominated Points (NNP) returned by EMSA and the NNP returned by BOA for each problem.

As can be seen in Table 3.2, the CPU times for all algorithms increase as the problem dimensions increase. We can also infer from Table 3.2 that the CPU times depend to some extent on the total number of efficient or nondominated points returned by the algorithms for a given problem. That is to say, the larger the number of efficient or nondominated points in a given problem, the more computational efforts would be required to compute them, see problems 25, 30, 36, 39, 45, 52, 53 and 55. With this observation, one can rightly say that

there is a relative influence of the size of the efficient or nondominated set on the CPU times. In all the problems considered, BOA was found to be computationally more efficient than the simplex-type algorithms.

As can also be seen in Table 3.2, the total number of nondominated points returned by EMSA are the same to that returned by BOA for most of the problems considered. This is so because EMSA has been designed to avoid returning redundant nondominated points that would be of no use to the DM after they have been computed. This feature is also reported in [30] that BOA avoids redundant calculations of points that would be of little or no use to the DM. This make EMSA compare favourably in terms of the total number of nondominated points it returns. However, we noticed a slight difference in the number of nondominated points returned by both algorithms in some of the problems considered. For these problems where this occurs, a few nondominated points have been repeated. It was also observed that the simplex-type algorithms could not produce results for problems 39, 40, 45, 46, 48, 49 and 55 after running for 3 days; it was aborted. The fact that some problems were aborted after 3 days of running time does not necessarily mean that MSA and EMSA

cannot solve these problems; if allowed to run further it would potentially return a huge number of efficient points or run out of memory which would indicate that the total number of efficient solutions has exceeded the Matlab solution capacity of the machine used. We also notice for the above problems that there is a slight difference in CPU times between MSA and EMSA. This was expected as a consequence of the extension, more computational efforts would be required to compute the corresponding nondominated points, after which they are sorted in each case. As can also be seen in Table 3.2, the CPU times returned by MSA are less than that returned by EMSA for all the problems considered. The difference in CPU times can be clearly seen in problems with a huge number of efficient solutions, see also problems 25, 30, 36 and 52.

3.7 Summary of Results

In this section, we present the summary of experimental results discussed in the previous section in Table 3.1. We have also presented the CPU time of MSA, EMSA and BOA for 47 out of the 55 instances (which represent 85.45 %) of the total problems solved by all

the methods in Figure 3.3.

Algorithms	Criteria for Evaluation	
	Computing Efficiency	**NNP returned**
MSA	Efficient on small to medium size instances and slightly outperform EMSA	Does not return nondominated points
EMSA	Efficient on small to medium size instances	Return the same NNP as BOA for over 85 percent of the problems considered
BOA	Computationally more efficient than MSA and EMSA	Return the same NNP as EMSA for most of the problems considered

Table 3.1: Summary of experimental results

3.8 Summary

In this chapter, we have presented MSA of Evans and Steuer [63] and extended the algorithm to compute the set of all nondominated points. We have also presented BOA [30] and illustrated the algorithms on a small MOLP instance. We then proceeded to investigate

the computational efficiency of EMSA and its original version, as well as compare the total number of nondominated points returned by EMSA with that returned by BOA on a collection of 55 problems ranging from small to moderate size. In the next chapter, we shall further compare the computing efficiency and the quality of nondominated points returned by EMSA with that returned by BOA and ASIMOLP.

Table 3.2: Comparative results for individual problem

Algorithm					**MSA**		**EMSA**			**BOA**	
Prob.	**Origin**	**n**	**m**	**q**	**NES**	**CPU (s)**	**NES**	**NNP**	**CPU (s)**	**NNP**	**CPU (s)**
1	Ehrgott 2006	3	3	3	5	0.158	5	3	0.169	3	0.038
2	Zeleny 1982	2	2	2	5	0.018	5	3	0.027	3	0.021
3	”	2	4	2	16	0.063	16	2	0.072	2	0.026
4	”	2	4	3	36	0.161	36	3	0.182	3	0.161
5	”	2	6	2	64	0.373	64	3	0.399	3	0.212

Continued on next page

Table 3.2 – continued from previous page

Algorithm					MSA		EMSA			BOA	
Prob.	**Origin**	**n**	**m**	**q**	**NES**	**CPU (s)**	**NES**	**NNP**	**CPU (s)**	**NNP**	**CPU (s)**
6	”	3	3	3	15	0.069	15	5	0.075	5	0.046
7	”	5	3	3	6	0.039	6	4	0.041	4	0.043
8	”	5	2	2	1	0.045	1	1	0.055	1	0.016
9	”	6	4	2	36	0.201	36	1	0.299	1	0.017
10	”	7	4	3	36	0.273	36	5	0.317	4	0.163
11	iMOLPe	2	3	2	8	0.036	8	4	0.052	3	0.047
12	”	3	3	4	12	0.052	12	3	0.074	3	0.033
13	”	3	5	3	88	0.502	88	10	0.551	10	0.042
14	”	3	3	3	19	0.077	19	8	0.089	7	0.035
15	”	4	3	3	18	0.092	18	8	0.097	8	0.038
16	”	4	2	3	8	0.039	8	6	0.041	6	0.036
17	”	4	4	3	44	0.247	44	11	0.302	11	0.053
18	”	3	3	3	37	0.156	37	5	0.178	5	0.033
19	”	15	10	2	98	1.494	98	12	1.619	11	0.054
20	”	15	10	3	254	3.901	254	28	4.852	37	0.445

Continued on next page

Table 3.2 – continued from previous page

Algorithm					MSA		EMSA			BOA	
Prob.	**Origin**	**n**	**m**	**q**	**NES**	**CPU (s)**	**NES**	**NNP**	**CPU (s)**	**NNP**	**CPU (s)**
21	”	10	15	3	50	0.481	50	15	0.569	14	0.259
22	Steuer	5	5	2	28	0.191	28	5	0.197	5	0.036
23	”	4	4	3	10	0.061	10	3	0.065	3	0.015
24	”	5	5	4	154	1.406	154	14	1.432	14	0.098
25	”	10	8	4	2096	123.432	2096	51	125.406	63	1.973
26	”	5	4	3	26	0.161	26	9	0.175	9	0.089
27	”	6	8	4	560	10.963	560	13	11.182	20	0.236
28	”	7	6	4	48	0.416	48	12	0.451	36	0.286
29	”	7	6	4	152	1.568	152	9	1.601	9	0.192
30	”	8	8	6	1080	34.946	1080	56	36.125	286	73.963
31	”	8	8	3	208	2.689	208	5	2.726	5	0.168
32	”	8	8	3	64	0.694	64	1	0.706	1	0.135
33	”	5	5	4	74	0.541	74	12	0.618	12	0.277
34	”	6	6	3	304	3.829	304	17	3.907	17	0.212
35	”	5	5	4	202	1.861	202	10	2.061	9	0.183

Continued on next page

Table 3.2 – continued from previous page

Algorithm					MSA		EMSA			BOA	
Prob.	**Origin**	**n**	**m**	**q**	**NES**	**CPU (s)**	**NES**	**NNP**	**CPU (s)**	**NNP**	**CPU (s)**
36	”	10	10	4	3,072	318.13	3,072	6	325.555	6	0.333
37	”	8	8	3	608	12.972	608	13	13.251	13	0.217
38	”	6	7	4	440	6.051	440	25	7.001	21	0.386
39	”	12	16	4	*	-	-	-	-	601	31.034
40	”	10	14	5	*	-	-	-	-	132	102.952
41	”	7	6	3	40	0.353	40	3	0.447	3	0.165
42	”	7	7	3	56	0.507	56	8	0.583	7	0.153
43	”	6	6	4	128	1.248	128	5	1.311	5	0.158
44	”	6	6	4	168	1.781	168	10	1.856	10	0.211
45	”	10	14	5	*	-	-	-	-	471	307.611
46	”	10	14	5	*	-	-	-	-	128	114.653
47	”	7	7	3	60	0.545	60	6	0.601	6	0.159
48	Bensolve1.2	100	101	2	*	-	-	-	-	32	0.503
49	Bensolve2.0	5	31	5	*	-	-	-	-	22	2.877
50	”	36	36	2	82	1.899	82	31	1.921	8	0.211

Continued on next page

Table 3.2 – continued from previous page

Algorithm					MSA		EMSA			BOA	
Prob.	**Origin**	**n**	**m**	**q**	**NES**	**CPU (s)**	**NES**	**NNP**	**CPU (s)**	**NNP**	**CPU (s)**
51	”	64	64	2	292	13.968	292	57	14.455	14	0.403
52	”	100	100	2	1102	118.418	1102	99	125.495	20	0.621
53	”	343	343	3	x	-	-	-	-	1,368	55.302
54	MOPLIB	53	226	3	561	26.231	561	552	28.116	552	6.551
55	”	53	221	3	*	-	-	-	-	2552	1663.803

(*) Aborted after 3 days of running time

(x) Out of memory

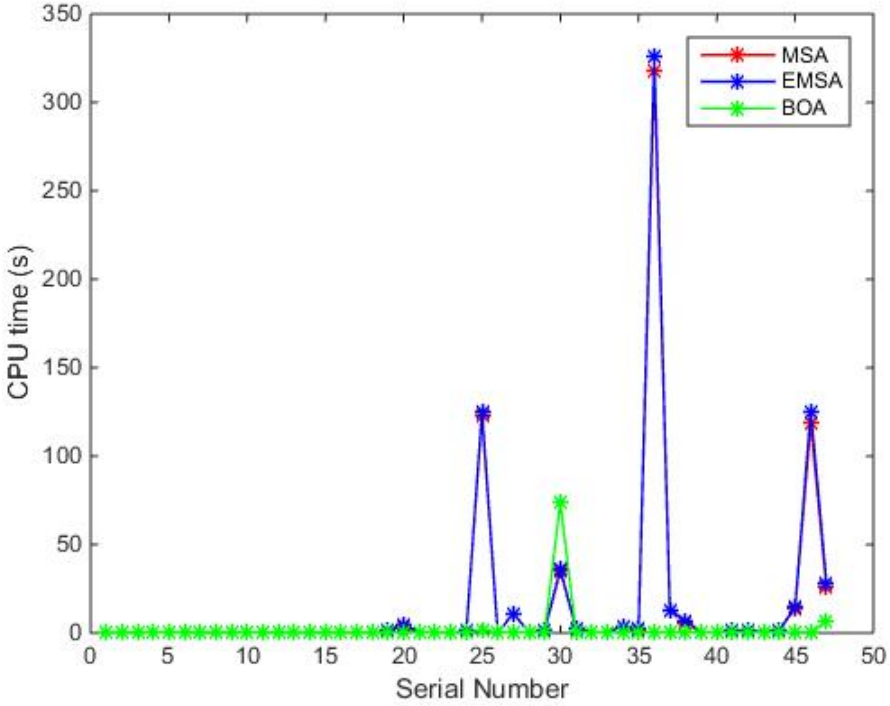

Figure 3.3: Running time of MSA, EMSA and BOA for the 47 instances solved.

Chapter 4

THE SIMPLEX, INTERIOR-POINT AND OBJECTIVE SPACE APPROACHES TO MOLP

4.1 Introduction

Most MOLP algorithms are based on the simplex algorithm and interior-point methods for Linear Programming. However, objective space based methods are becoming more and more prominent. Benson [30], argued that since the number of objectives in an MOLP is often much smaller than the number of decision variables and typically many efficient solutions in the decision space map to a single point in the objective space, generating the set of nondominated

points in the objective space would require less computation, [59]. He then suggested an outer-approximation algorithm for computing all the nondominated points in the objective space of the problem. This chapter compares the computational efficiency and the quality of a most preferred nondominated point returned by this algorithm with the most preferred nondominated point returned by EMSA as well as with that returned by Arbel's ASIMOLP [6].

This comparison may sound not possible given that the three algorithms are based on three different philosophies and compute different things: EMSA works in the decision space and finds the set of all efficient extreme points and also generates the set of all nondominated points; ASIMOLP also works in the decision space but finds a most preferred efficient point and also returns the corresponding most preferred nondominated point; BOA, on the other hand, works in the objective space to find the set of all nondominated points of the problem.

To achieve this comparison, we recall that it has been shown that in practice, the DM prefers basing his or her choice of a most preferred (best) solution in the nondominated points [30]. We shall then act as

the DM and choose a Most Preferred Nondominated Point (MPNP) whose components are as close as possible to an unattainable ideal objective point from the nondominated set returned by EMSA and BOA to compare with a MPNP returned by ASIMOLP.

One of the key issues in MOLP is computing the MPNP's. We give a detailed procedure for the purpose here which allows us to carry out the comparison.

To the best of our knowledge, no comparison of the computing efficiency and the quality of MPNP chosen from the nondominated set returned by BOA and EMSA with that returned by ASIMOLP has been carried out before. We intend to fill this gap here.

This chapter is organized as follows. Section 4.2 presents ASIMOLP. The determination of priority vector used in ASIMOLP is presented in Section 4.2.2. We presents Interactive ASIMOLP in Section 4.3. Section 4.4 discusses the selection of a most preferred nondominated point. We provide the experimental results in Section 4.5. A summary of results is presented in Section 4.6 and the summary of Chapter 4 is presented in Section 4.7.

4.2 The Affine Scaling Interior-Point Algorithm

ASIMOLP whose general form can be found in [6], works in the decision space and returns only one efficient extreme point of the problem, or at most, an efficient face of the feasible region. It also returns the corresponding nondominated point. The algorithm is initialized with a feasible and interior starting solution vector x^0 and generates q interior step direction vectors dx_i $(1 \leq i \leq q)$. AHP is then used to derive the relative priority or preference vector p for these directions by filling a pairwise comparison matrix, which is then normalized and the rows are averaged to obtain the priority vector. The components of the derived priority vector p are then used as coefficients of a convex combination of the q interior step directions that yields a combined step direction vector dx that moves toward a new feasible point. This process continues until the algorithm converges to a most preferred efficient extreme point after meeting some termination conditions. Before we present the pseudo-code form of ASIMOLP, the used notation is described.

Notation: A, b, C form the problem data; x^0 is the initial interior feasible solution vector; α_i is the step size $(1 \leq i \leq q)$; σ is a stopping

tolerance; ρ is the step size factor; D is the diagonal and scaling matrix; y_i is an estimate of the dual vector $(1 \leq i \leq q)$; dx_i is the ith interior step direction vector $(1 \leq i \leq q)$; dx is the combined step direction vector; p_i is the derived priority vector $(1 \leq i \leq q)$; x_{new} is the new feasible point; f_{new} is the new objective values; x_{end} is the most preferred efficient extreme point at the boundary of the feasible region at termination and f_{end} is the corresponding objective values at termination.

Algorithm 6 Affine Scaling Interior MOLP Algorithm

0: Input: A, b, C : *Problem data*
1: Initialize: *Choose* $x^0 > 0$, *Stopping tolerance* $\sigma\,(0 < \sigma < 1)$, *Step size factor* $\rho\,(0 < \rho < 1)$,
Converged $= 0$, $k \leftarrow 0$;
2: while *Converged* $\neq 1$ **do**
3: $k \leftarrow k + 1$
4: $D \leftarrow diag(x^0)$
5: $y_i(k) \leftarrow (AD^2A^T)^{-1}AD^2c_i, 1 \leq i \leq q$
6: $dx_i(k) \leftarrow D^2(c_i^T - A^Ty_i(k)), 1 \leq i \leq q$
7: **if** $dx_i(k) \geq 0, 1 \leq i \leq q$, **stop**
8: **else**
9: $\alpha_i \leftarrow min[\frac{-x_i}{dx_i(k)}, \forall dx_i(k) < 0], 1 \leq i \leq q$
10: $x_i(k+1) \leftarrow x(k) + \rho\alpha_i dx_i(k), 1 \leq i \leq q$
11: $dx \leftarrow \sum_{i=1}^{q} p_i\alpha_i dx_i$
12: $x(k+1) \leftarrow x(k) + \rho dx(k)$
13: $x_{end} \leftarrow x(k) + dx(k)$
14: $f_{end} \leftarrow C^Tx_{end}$
15: $dx_{end} \leftarrow x_{end} - x(k)$
16: **if** $k > 1$, **do**
17: $dx \leftarrow \sum_{i=1}^{q} p_i\alpha_i dx_i + p_{end}dx_{end}$
18: $x_{new} \leftarrow x(k) + \rho dx(k)$
19: $f_{new} \leftarrow C^Tx_{new}$
20: $x_{end} \leftarrow x(k) + dx(k)$
21: $f_{end} \leftarrow C^Tx_{end}$
22: $dx_{end} \leftarrow x_{end} - x_{new}$
23: **else**
24: **if** $\|dx_{end}\| \leq \sigma$, **stop**
25: **else**
26: $x^0 \leftarrow x_{new}$
27: *Go to step* 4
28: **endif**
29: **endif**
30: **endif**
31: endwhile
32 Output: x_{end} : *Most preferred efficient solution*
f_{end} : *Values of the objective functions*

4.2.1 Illustration of ASIMOLP

We developed the pseudo-code of this algorithm, implemented it in Matlab and used it to solve Problem 3.1 of Section 3.2.1. The most preferred efficient solution found using our Matlab implementation of Algorithm 5 is $x^1 = (3.6418,\ 3.7913)^T$, and the corresponding objective values are $f^1 = (-3.6418,\ -3.7913)^T$ where $f^1 \in Y_N$. The search path as generated by the Algorithm is shown in Figure 4.1.

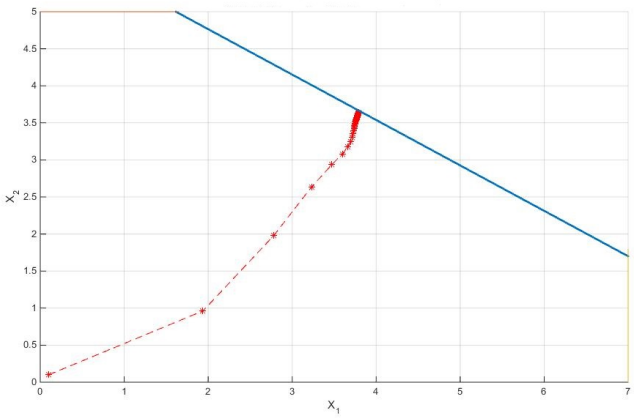

Figure 4.1: ASIMOLP search path showing convergence to the efficient frontier.

4.2.2 Determination of the priority vector used in ASI-MOLP

From [5], one can either use a utility function if it is available (as was done in [9]), to assess preference information needed to establish a combined step direction instead of interacting with the DM or use the AHP methodology, but in most cases, the utility function is not known [6]. In this chapter, we have used AHP as was done in [6, 7] to derive the relative preference or priority vector p whose components are used as coefficients of a convex combination of the q interior step directions that yields a combined step direction that moves the current iterate to a new one.

The procedure involves a pairwise comparison of the q interior step directions and construction of a $q \times q$ comparison matrix for comparing the interior step directions. A complete pairwise comparison

matrix A can be expressed as

$$A = \begin{array}{c} \\ d_1 \\ d_2 \\ \vdots \\ d_q \end{array} \begin{array}{c} \begin{array}{cccc} d_1 & d_2 & \dots & d_q \end{array} \\ \begin{pmatrix} \frac{w_1}{w_1} & \frac{w_1}{w_2} & \dots & \frac{w_1}{w_q} \\ \frac{w_2}{w_1} & \frac{w_2}{w_2} & \dots & \frac{w_2}{w_q} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{w_q}{w_1} & \frac{w_q}{w_2} & \dots & \frac{w_q}{w_q} \end{pmatrix} \end{array} = \begin{pmatrix} 1 & a_{12} & \dots & a_{1q} \\ a_{21} & 1 & \dots & a_{2q} \\ \vdots & \vdots & \ddots & \vdots \\ a_{q1} & a_{q2} & \dots & 1 \end{pmatrix}$$

where the entry a_{ij} indicates the strength of the step direction d_i when compared with the step direction d_j. These entries are obtained from the well-established comparison scale used by the AHP in [127] and [126]. Such a scale is shown in Table 4.1.

Numerical value	**Interpretation**
1	requirements i and j are of equal value
3	requirement i has a slightly higher value than j
5	requirement i has a strongly higher value than j
7	requirement i has a very strongly higher value than j
9	requirement i has an absolutely higher value than j
2,4,6,8	intermediate values between two adjacent judgments

Table 4.1: Graduation scale for comparing alternatives

We note that the above comparison matrix is a reciprocal matrix, where $a_{ij} = 1/a_{ij}$, $a_{ij} > 0$ and $a_{ij} = 1$ for $i = j$. After filling out and

obtaining the comparison matrix, the matrix is then normalized. The normalized principal eigen-vector herein referred to as the preference or priority vector, p, is obtained by averaging across the rows of the matrix. The components of the priority vector are then used as coefficients of a convex combination of the q interior directions that yields a combined direction that enables one to move from the current iterate to the next, (more details can be seen in [127] and [126]).

4.3 Interactive Affine Scaling Interior MOLP Algorithm

We present the Interactive ASIMOLP which is another examplar algorithm from the Affine Scaling Interior MOLP Algorithm in this section. It is developed in [15]. It works in the decision space and also returns only one efficient solution, or an efficient face of the feasible region. The difference between Algorithms 6 and this one is that at each iteration a set of points is presented to the DM to decide that which is most preferred. The difference can also be seen in the assessment of the relative priority vector p which is used to derive the combined step direction vector dx. In Algorithm 6, AHP was used

to derive p by filling a comparison matrix, while in here, the implicitly known DM's utility function $u(x)$ is used to approximate vector p. At each iteration, the current iterate and the boundary point are presented to the DM to choose the most preferred efficient solution.

Notation: A, b, C form the problem data; x^0 is the initial interior feasible solution; δ is a stopping tolerance; ρ is the step size factor; D, the diagonal and scaling matrix; y_i is an estimate of the dual vector $(1 \leq i \leq q)$; $u(x)$ is the implicitly known utility function; z_i are the q reduced cost step direction vectors $(1 \leq i \leq q)$; dx_i are the individual q interior step direction vectors $(1 \leq i \leq q)$; du is the change in utility function; dv is the change in objective values; du_x is an approximate gradient of the utility function; dx is the combined step direction vector; α_i is the step size $(1 \leq i \leq q)$; p_i is the derived priority vector $(1 \leq i \leq q)$; x_{new} is the new feasible point; f_{new} is the new objective values; x_{end} is the most preferred efficient solution at the boundary of the feasible region at termination and f_{end} is the corresponding objective values at termination.

Algorithm 7 Interactive Affine Scaling Interior MOLP Algorithm

0: Input: A, b, C
1: Initialize: *Choose* $x^0 > 0$, *Stopping tolerance* $\sigma\,(0 < \sigma < 1)$, *Step size factor* $\rho\,(0 < \rho < 1)$, $u(x)$,
Converged $= 0$, $k \leftarrow 0$;
2: while *converged* $\neq 1$ **do**
3: $k \leftarrow k + 1$
4: $D \leftarrow diag(x^0)$
5: $y_i(k) \leftarrow (AD^2A^T)^{-1}AD^2c_i, 1 \leq i \leq q$
6: $z_i(k) \leftarrow (c_i^T - A^Ty_i(k)), 1 \leq i \leq q$
7: $dx_i(k) \leftarrow D^2(z_i(k)), 1 \leq i \leq q$
8: **if** $dx_i(k) \geq 0$, $1 \leq i \leq q$, **stop**
9: **else**
10: $du \leftarrow [p_i - p_0]$
11: $dv(k) \leftarrow C[\rho\alpha_i dx_i(k)]$
12: $du_x(k) \leftarrow du(dv(k))^{-1}C$
13: $y \leftarrow (AD^2A^T)^{-1}AD^2(du_x(k))^T$
14: $dx(k) \leftarrow D^2(du_x(k))^T - A^Ty$
15: $\alpha_i = Min[\frac{-X_i}{dx_i(k)}, \forall dx_i(k) < 0], 1 \leq i \leq q$
16: $x_i(k+1) \leftarrow x(k) + \rho\alpha_i dx_i(k), 1 \leq i \leq q$
17: $x_{end} \leftarrow x(k) + \rho dx(k)$
18: **if** $k > 1$, **do**
19: $du \leftarrow [p_i - p_0,\ p_b - p_0], 1 \leq i \leq q$
20: $dv(k) \leftarrow C[\rho\alpha_i dx_i(k),\ dx_b], 1 \leq i \leq q$
21: $du_x(k) \leftarrow du(dv(k))^T(dv(k)(dv(k))^T)^{-1}C$
22: $y \leftarrow (du_x(k))^T - A^T(AD^2A^T)^{-1}AD^2(du_x(k))^T$
23: $dx \leftarrow D^2y$
24: $x_{new} \leftarrow x(k+1) + \rho dx(k)$
25: $f_{new} \leftarrow C^Tx_{new}$
26: $x_{end} \leftarrow x(k+1) + dx(k)$
27: $dx_{end} \leftarrow x_{end} - x_{new}$
28: **else**
29: **if** $\|dx_{end}\| \leq \sigma$, **stop**
30: **else**
31: $x^0 \leftarrow x_{new}$
32: **Go to step 4**
33: **endif**
34: **endif**
35: **endif**
36: **endwhile**
37 Output: x_{end}
f_{end}

4.3.1 Illustration of Interactive ASIMOLP

We also developed the pseudo-code of Algorithm 7, implemented the algorithm in Matlab and applied to Problem 3.1 of section 3.2.1. The most preferred efficient solution found is $x^1 = (3.7146, 3.7094)^T$ and the corresponding values of objective functions are $f^1 = (-3.7146, -3.7094)^T$. Notice how the solution is quite similar to that obtained with Algorithm 6. The search path as generated by Algorithm 7 is shown in Figure 4.2

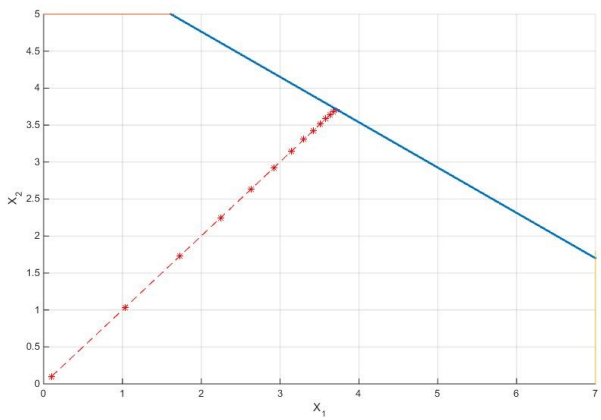

Figure 4.2: Interactive ASIMOLP search path showing convergence to the efficient frontier.

4.4 Selection of the Most Preferred Nondominated Point

This issue has been alluded to in Section 4.1. To determine the MPNP, we employ the technique of Compromise Programming (CP) introduced by Zeleny [161] and compute the ideal objective point which would serve as a reference point in each case. CP is a mathematical programming method that is based on the notion of distance of a most preferred solution from the ideal point y^*, [163]. CP can be used to find the best nondominated point by determining the minimum distance to the ideal point, [164]. Ehrgott & Tenfelde-Podehl [60] note that the ideal point is an essential component of CP, and the idea is to find a nondominated point which is as close as possible to it. This is a point in the objective space whose components are the optimal values of the objective functions when they are individually optimized, [3]. It was also noted in [163] that the ideal point serves as a rationale directing and facilitating human choice and decision making. To find the ideal point, we simply solve q sin-

gle objective problems

$$\begin{aligned} \min \quad & c_k^T x, \quad k = 1, 2, ..., q \\ \text{subject to} \quad & x \in X. \end{aligned} \tag{4.1}$$

We note here that, the ideal point itself is not an element of the nondominated set ($y^* \notin Y_N$). Otherwise, this would mean that the objective functions are not conflicting. It always exists in the objective space, but its corresponding point in the feasible region of the decision space may not exist [3].

For our numerical illustration above (problem 3.1 of Section 3.2.1), solving each of the objective function individually over the feasible region X yields the ideal objective point $y^* = (-7.0, -5.0)^T$. Clearly $y^* \notin Y_N$ where $Y_N = \{(-7.0,\ -1.8)^T, (-1.6,\ -5.0)^T\}$.

Having computed the ideal objective point y^*, we now determine the minimum distance of each nondominated point $\hat{y}$ from it by finding

$$\min \quad \{\|\hat{y}_1 - y^*\|,\ \ \|\hat{y}_2 - y^*\|, \ldots, \|\hat{y}_n - y^*\|\}$$

where $\hat{y} \in Y_N$ has already been found either by BOA or EMSA, $\| \ .\|$ is the Euclidean norm on $\mathbb{R}^q$ and y^* is the ideal objective point.

Using the nondominated points f^1 and f^2 of problem (3.1) yields

$$\|f^1 - y^*\| = 3.2 \qquad \text{and} \qquad \|f^2 - y^*\| = 5.4.$$

Since, the relative distance of f^1 from the ideal point y^* is 3.2 which is the smallest of the two, it therefore means that $f^1 = (-7.0,\ -1.8)^T$ is the closest of the two nondominated points to the ideal point $y^* = (-7.0, -5.0)^T$. Hence, f^1 is selected as the DM's most preferred nondominated point.

Next, we measure the distance of the nondominated point $f^1 = (-3.6418 - 3.7913)^T$ returned by ASIMOLP in Section 4.2.1 for the same numerical illustration (Problem 3.1 of Section 3.2.1) from the ideal point $y^* = (-7.0, -5.0)^T$, as was done with those returned by BOA and EMSA for the same example. It turned out that, the distance

$$\|f^1 - y^*\| = 3.5691$$

is bigger than 3.2 which was the closest when measuring the points returned by BOA and EMSA, thereby making the nondominated points returned by BOA and EMSA closer to the ideal point and of higher quality.

The following more substantial illustrative MOLP adapted from Zeleny [163] with three objectives makes the point.

$$
\begin{aligned}
\min f_1 &= -x_1 - 2x_2 + x_3 - 3x_4 - 2x_5 - x_7 \\
\min f_2 &= -x_2 - x_3 - 2x_4 - 3x_5 - x_6 \\
\min f_3 &= -x_1 - x_3 + x_4 + x_6 + x_7
\end{aligned}
$$

Subject to

$$
\begin{aligned}
x_1 + 2x_2 + x_3 + x_4 + 2x_5 + x_6 + 2x_7 &\leq 16 \\
-2x_1 - x_2 + x_4 + 2x_5 + x_7 &\leq 16 \\
-x_1 + x_3 + 2x_5 - 2x_7 &\leq 16 \\
x_2 + 2x_3 - x_4 + x_5 - 2x_6 - x_7 &\leq 16 \\
x_1, \quad x_2, \quad x_3, \quad x_4, \quad x_5, \quad x_6, \quad x_7 &\geq 0
\end{aligned}
\tag{4.2}
$$

Again, optimizing each of the objective functions individually over the feasible region yields the ideal objective point $y^* = (-48.0, -32.0, -16.0)^T$. Solving (4.2) with BOA and EMSA, the set of nondominated points found is $Y_N = \{(-48.0, -32.0, 16.0)^T, (-16.0, 0.0, -16.0)^T, (0.0, -8.0, -16.0)^T, (-5.33, -21.33, -5.33)^T, (-16.0, -24.0, 0.0)^T\}$ with $y^* \notin Y_N$. By determining the minimum distance of each of these non-

dominated points from the ideal point y^*, it was found that the point $(-48.0, -32.0, 16.0)^T$ is the closest. Its distance from it is 32. It is selected as the DM's MPNP.

For problem (4.2), the MPNP returned by ASIMOLP is $f^1 = (-7.65, -13.80, -7.75)^T$ as shown in Table 4.2, Problem 10. Again, we measure its distance from the ideal point $y^* = (-48.0, -32.0, -16.0)^T$. It was found that the distance of the point $(-7.65, -13.80, -7.75)^T$ from $y^* = (-48.0, -32.0, -16.0)^T$ is 45.0269, which is also larger than the corresponding values of BOA and EMSA, thereby making the MPNPs returned by BOA and EMSA to be of higher quality.

We have used this method to choose the MPNP from the nondominated sets returned by BOA and EMSA for comparison. There is no selection of a MPNP in ASIMOLP as the algorithm computes a most preferred efficient solution and also returns the corresponding most preferred nondominated point.

To determine the quality of the MPNP returned by ASIMOLP, we simply measure its distance from the ideal point in each case and compare with the distances of those returned by BOA and EMSA in order to determine that which is the closest to the ideal point and of

higher quality.

4.5 Discussion of Experimental Results

In this section, we provide numerical results to compare the quality of a Most Preferred Nondominated Point (MPNP) and the efficiency of Algorithms 4, 5 and 6. Table 4.3 shows the numerical results for a collection of 61 existing problems ranging from small to medium and realistic MOLP instances. Problem 1 is taken from Ehrgott [56], and Problems 2 to 10 are from Zeleny [163]. Problems 11 to 21 are test problems from the interactive MOLP explorer (iMOLPe) of Alves *et al.* [3]. Problems 22 to 47 are taken from Steuer [143]. Problem 48 is a test problem in Bensolve-1.2 of Löhne [101], while problems 49 and 53 are test problems in Bensolve-2.0 of Löhne and Weißing [104]. Problems 50 to 52 are obtained using a script in Bensolve-2.0 of Löhne and Weißing [104] that was also used to generate problem 53 with the same number of variables and constraints. Finally, problems 54 to 61 are from MOPLIB [103] which stands for Multi-Objective Problem Library.

We have added six (6) additional instances to our collection of prob-

lems in this chapter. These problems are larger in size and have more difficult structures. They include Problems 54 to 56 and 59 to 61. Problem 54 is such that the constraint and criterion matrices are sparse while the components of the RHS vector are all zeros except for a one (1) at the centre as the only non-zero entry. Problems 55 and 60 have dense RHS vectors while the constraint and criterion matrices are sparse. In Problems 56 and 59, the constraint matrices are sparse, the criterion matrices are dense and all the elements in the RHS vectors are ones. Problem 61 is such that the constraint and criterion matrices are sparse while the components of the RHS vector are all zeros except for a ninety (90) at the end as the only non-zero entry.

We modified and extended Algorithm 3 of Evans and Steuer [63] into Algorithm 4 or EMSA the Extended Multi-objective Simplex Algorithm. We have implemented it in Matlab in the same way as in [124] and experimented with it on a set of MOLP's. We have also implemented Algorithm 6 in Matlab and used an existing Matlab implementation of Algorithm 5 known as Bensolve-1.2, [101]. The current version, Bensolve-2.0, of Löhne and Weißing [104] is implemented in the C programming language. We employed Bensolve-1.2 [101] which

is implemented in Matlab to test the algorithms with the same tools and for a more meaningful comparisons. In all test, m is the number of constraints, n the number of variables and q the number of objectives. Algorithm 4 is EMSA, Algorithm 5 is BOA as presented in [135] and Algorithm 6 is Arbel's ASIMOLP [6]. All algorithms were executed on an Intel Core i5-2500 CPU at 3.30GHz with 16.0GB RAM. We recorded the CPU times (in seconds) for each problem and acted as the DM by choosing a most preferred (best) nondominated point (whose components are as close as possible to the ideal objective point as explained in Section 4.4) from the nondominated set $Y_N = \{Cx : x \in X_E\}$ returned by BOA to compare with the MPNP returned by EMSA and with that returned by ASIMOLP.

As can be seen from Table 4.3, the CPU times for all algorithms increase as the problem sizes increase. It was observed that ASIMOLP returns a CPU time of less than a second for most of the test problems it solves, thereby making it computationally more efficient than BOA and EMSA. However, BOA was found to be computationally more efficient than EMSA for all the test problems considered. We noticed that ASIMOLP did not solve problem 54 as there exists no initial and strictly positive starting solution ($x^0 > 0$ such that $Ax^0 = b$) due

to singularity issues which indicates that either the initial solution does not exist or it is not unique. We suspect that the difficulty this problem pose to ASIMOLP is due to its matrix structure and the way interior-point methods work. A diagonal matrix whose diagonal elements are the elements of an initial positive starting solution is required. Once the diagonal elements are not strictly positive ($x^0 > 0$), ASIMOLP exhibits this difficulty. We even employed a decomposition approach which did not yield the required initial positive starting solution.

In terms of the quality of a MPNP returned by the algorithms, it was observed that EMSA and BOA return the same MPNPs for all test problems considered. This makes these two algorithm comparable and the nondominated points they returned are of higher quality than those returned by ASIMOLP in all cases. We also observed in Table 4.3 that EMSA and BOA could not produce results for some of the test problems considered despite the long running time allowed (3 days); they were aborted. If allowed to run further, they would potentially return a huge number of nondominated points or run out of memory as earlier explained in Section 3.6. We note here that, some of these problems most especially from problem 48 to 61 are

numerically ill-posed and highly challenging MOLP instances with difficult structures.

4.6 Summary of Results

In this section, we present the summary of experimental results discussed in the previous section in Table 4.2. We have also presented the CPU time of EMSA, ASIMOLP and BOA for 48 out of the 61 instances (which represent 78.69 %) of the total problems solved by all the algorithms in Figure 4.3.

	Criteria for Evaluation	
Algorithms	**Computing Efficiency**	**Quality of MPNP returned**
EMSA	Efficient on small to medium size instances	Return high quality MPNP as BOA which is superior to that returned by ASIMOLP
ASIMOLP	Computationally more efficient than EMSA and BOA	Quality of MPNP returned not so good
BOA	Computationally more efficient than EMSA	Return the same MPNP as EMSA for all the problems considered

Table 4.2: Summary of experimental results

4.7 Summary

In this chapter, we have presented ASIMOLP and its interactive version. We have also illustrated them on a small MOLP instance. We then proceeded to investigate their computational efficiency and compare the quality of a most preferred nondominated point they returned on a collection of 61 existing problems ranging from small to moderate and large MOLP instances. In the next chapter, we shall carry out a detailed comparison of BOA with the recently introduced PSA of Rudloff *et al.* [124] using small, medium and realistic MOLP instances.

Table 4.3: Comparative results for small, medium and large instances

Algorithm					EMSA		ASIMOLP		BOA	
Prob.	**Origin**	**n**	**m**	**q**	**MPNP**	**CPU (s)**	**MPNP**	**CPU (s)**	**MPNP**	**CPU (s)**
1	Ehrgott 2006	3	3	3	f1 = -2.00 f2 = 10.00 f3 = -5.00	0.169	f1 = -1.74 f2 = 5.56 f3 = -2.75	0.036	f1 = -2.00 f2 = 10.00 f3 = -5.00	0.038
2	Zeleny 1982	2	2	2	f1 = -25000.00 f2 = -66667.00	0.027	f1 = -30626.00 f2 = -64132.00	0.111	f1 = -25000.00 f2 = -66667.00	0.021
3	”	2	4	2	f1 = -9.00 f2 = -15.00	0.072	f1 = 4.00 f2 = -18.42	0.031	f1 = -9.00 f2 = -15.00	0.026
4	”	2	4	3	f1 = -3.00 f2 = -7.50 f3 = 4.00	0.182	f1 = -3.50 f2 = -2.74 f3 = 4.89	0.027	f1 = -3.00 f2 = -7.50 f3 = 4.00	0.161
5	”	2	6	2	f1 = -24.00 f2 = -16.00	0.399	f1 = -21.29 f2 = -17.29	0.032	f1 = -24.00 f2 = -16.00	0.212
6	”	3	3	3	f1 = 3.00 f2 = -6.00 f3 = -12.00	0.075	f1 = 1.33 f2 = -6.20 f3 = -9.68	0.028	f1 = 3.00 f2 = -6.00 f3 = -12.00	0.046
7	”	5	3	3	f1 = 0.00 f2 = -4.00 f3 = -24.00	0.041	f1 = -1.38 f2 = -2.77 f3 = -10.04	0.034	f1 = 0.00 f2 = -4.00 f3 = -24.00	0.043
8	”	5	2	2	f1 = -52.0 f2 = -52.0	0.055	f1 = -4.11 f2 = -29.30	0.019	f1 = -52.0 f2 = -52.0	0.016
9	”	6	4	2	f1 = 0.00 f2 = 0.00	0.229	f1 = -0.02 f2 = -0.00	0.043	f1 = 0.00 f2 = 0.00	0.017
10	”	7	4	3	f1 = -48.00 f2 = -32.00 f3 = 16.00	0.317	f1 = -7.65 f2 = -13.80 f3 = -7.75	0.035	f1 = -48.00 f2 = -32.00 f3 = 16.00	0.163
11	iMOLPe	2	3	2	f1 = -21.00 f2 = -7.00	0.052	f1 = -11.87 f2 = -10.22	0.032	f1 = -21.00 f2 = -7.00	0.047
12	”	3	3	4	f1 = -10.00 f2 = -20.00 f3 = -100.00 f4 = -10.00	0.074	f1 = -5.59 f2 = -18.62 f3 = -34.83 f4 = -42.23	0.037	f1 = -10.00 f2 = -20.00 f3 = -100.00 f4 = -10.00	0.033
13	”	3	5	3	f1 = -21.00 f2 = -4.50	0.551	f1 = -10.48 f2 = -3.62	0.035	f1 = -21.00 f2 = -4.50	0.042

Continued on next page

Table 4.3 – continued from previous page

Algorithm					EMSA		ASIMOLP		BOA	
Prob.	**Origin**	**n**	**m**	**q**	**MPNP**	**CPU (s)**	**MPNP**	**CPU (s)**	**MPNP**	**CPU (s)**
					f3 = -4.00		f3 = -2.14		f3 = -4.00	
14	”	3	3	3	f1 = -2.66 f2 = -2.00 f3 = -0.33	0.089	f1 = -1.10 f2 = -1.22 f3 = -1.57	0.041	f1 = -2.66 f2 = -2.00 f3 = -0.33	0.035
15	”	4	3	3	f1 = -48.50 f2 = -19.50 f3 = -37.00	0.097	f1 = -35.80 f2 = -43.97 f3 = -29.82	0.051	f1 = -48.50 f2 = -19.50 f3 = -37.00	0.038
16	”	4	2	3	f1 = -20.00 f2 = -80.00 f3 = -40.00	0.041	f1 = -31.71 f2 = -49.12 f3 = -38.69	0.046	f1 = -20.00 f2 = -80.00 f3 = -40.00	0.036
17	”	4	4	3	f1 = -40.00 f2 = -50.00 f3 = -10.00	0.243	f1 = -32.22 f2 = 32.50 f3 = -36.27	0.041	f1 = -40.00 f2 = -50.00 f3 = -10.00	0.186
18	”	3	3	3	f1 = 0.00 f2 = -2.00 f3 = -4.00	0.178	f1 = -1.12 f2 = -2.14 f3 = 2.63	0.042	f1 = 0.00 f2 = -2.00 f3 = -4.00	0.033
19	”	15	10	2	f1 = -363.82 f2 =-33.70	1.581	f1 = -137.09 f2 =-198.96	0.142	f1 = -363.82 f2 =-33.70	0.195
20	”	15	10	3	f1 = -363.82 f2 = -33.70 f3 = -136.71	4.852	f1 = -107.15 f2 = -169.94 f3 = -16.26	0.223	f1 = -363.82 f2 = -33.70 f3 = -136.71	0.476
21	”	10	15	3	f1 = -132.60 f2 = -236.42 f3 = -279.67	0.569	f1 = 59.42 f2 = -257.21 f3 = -256.48	0.145	f1 = -132.60 f2 = -236.42 f3 = -279.67	0.259
22	Steuer 1986	5	5	2	f1 = -10.00 f2 = -3.00	0.197	f1 = -6.30 f2 = -6.90	0.033	f1 = -10.00 f2 = -3.00	0.036
23	”	4	4	3	f1 = 3.42 f2 = -10.28 f3 = -3.42	0.065	f1 = -3.79 f2 = 11.38 f3 = -2.96	0.035	f1 = 3.42 f2 = -10.28 f3 = -3.42	0.015
24	”	5	5	4	f1 = 1.02 f2 = -25.46 f3 = 24.44 f4 = -28.32	1.432	f1 = 2.28 f2 = -22.58 f3 = 25.30 f4 = -25.47	0.037	f1 = 1.02 f2 = -25.46 f3 = 24.44 f4 = -28.32	0.098
25	”	10	8	4	f1 = 106.29	125.406	f1 = 80.00	0.048	f1 = 106.29	1.973

Continued on next page

Table 4.3 – continued from previous page

Algorithm					EMSA		ASIMOLP		BOA	
Prob.	**Origin**	**n**	**m**	**q**	**MPNP**	**CPU (s)**	**MPNP**	**CPU (s)**	**MPNP**	**CPU (s)**
					f2 = -462.13 f3 = 175.57 f4 = -33.41		f2 = -54.36 f3 = -163.73 f4 = -23.82		f2 = -462.13 f3 = 175.57 f4 = -33.41	
26	”	5	4	3	f1 = -52.07 f2 = 31.50 f3 = -17.35	0.196	f1 = -4.44 f2 = -13.17 f3 = -14.37	0.041	f1 = -52.07 f2 = 31.50 f3 = -17.35	0.054
27	”	6	8	4	f1 = -6.94 f2 = -5.38 f3 = 6.83 f4 = -9.16	14.846	f1 = -6.69 f2 = -2.25 f3 = 6.77 f4 = -8.83	0.045	f1 = -6.94 f2 = -5.38 f3 = 6.83 f4 = -9.16	0.065
28	”	7	6	4	f1 = -31.53 f2 = -26.48 f3 = -26.57 f4 = -0.34	0.451	f1 = -25.90 f2 = -23.94 f3 = -19.06 f4 = -8.62	0.032	f1 = -31.53 f2 = -26.48 f3 = -26.57 f4 = -0.34	0.286
29	”	7	6	4	f1 = 26.80 f2 = -37.73 f3 = -24.33 f4 = -59.60	1.601	f1 = 4.03 f2 = -29.03 f3 = -18.07 f4 = -28.17	0.033	f1 = 26.80 f2 = -37.73 f3 = -24.33 f4 = -59.60	0.192
30	”	8	8	6	f1 = -74.00 f2 = -107.50 f3 = -41.25 f4 = -27.25 f5 = -9.00 f6 = -30.75	36.125	f1 = -15.46 f2 = -38.73 f3 = -43.30 f4 = -30.95 f5 = -8.30 f6 = -26.72	0.084	f1 = -74.00 f2 = -107.50 f3 = -41.25 f4 = -27.25 f5 = -9.00 f6 = -30.75	73.963
31	”	8	8	3	f1 = -36.57 f2 = -22.28 f3 = -14.00	2.726	f1 = -32.03 f2 = -20.03 f3 = -17.73	0.036	f1 = -36.57 f2 = -22.28 f3 = -14.00	0.168
32	”	8	8	3	f1 = -14.03 f2 = -18.00 f3 = -4.93	0.706	f1 = -8.77 f2 = -10.56 f3 = -5.13	0.036	f1 = -14.03 f2 = -18.00 f3 = -4.93	0.135
33	”	5	5	4	f1 = -21.50 f2 = -39.25 f3 = -16.25	0.618	f1 = -20.83 f2 = -21.78 f3 = -16.05	0.049	f1 = -21.50 f2 = -39.25 f3 = -16.25	0.277

Continued on next page

Table 4.3 – continued from previous page

Algorithm					EMSA		ASIMOLP		BOA	
Prob.	**Origin**	**n**	**m**	**q**	**MPNP**	**CPU (s)**	**MPNP**	**CPU (s)**	**MPNP**	**CPU (s)**
					f4 = 27.00		f4 = 14.45		f4 = 27.00	
34	”	6	6	3	f1 = -12.65 f2 = 0.00 f3 = -30.15	3.907	f1 = 12.69 f2 = -3.21 f3 = -28.39	0.046	f1 = -12.65 f2 = 0.00 f3 = -30.15	0.212
35	”	5	5	4	f1 = -14.66 f2 = -21.06 f3 = 35.73 f4 = -16.00	2.016	f1 = -6.33 f2 = -14.44 f3 = 20.77 f4 = -14.63	0.033	f1 = -14.66 f2 = -21.06 f3 = 35.73 f4 = -16.00	0.183
36	”	10	10	4	f1 = 46.50 f2 = 19.21 f3 = -27.07 f4 = -27.07	325.555	f1 = 50.69 f2 = 18.98 f3 = -23.38 f4 = -23.85	0.057	f1 = 46.50 f2 = 19.21 f3 = -27.07 f4 = -27.07	0.333
37	”	8	8	3	f1 = -14.48 f2 = -4.74 f3 = 6.93	13.251	f1 = -2.46 f2 = -3.22 f3 = -1.93	0.042	f1 = -14.48 f2 = -4.74 f3 = 6.93	0.217
38	”	6	7	4	f1 = -2.61 f2 = -12.63 f3 = 9.70 f4 = -2.37	7.001	f1 = -1.80 f2 = -4.00 f3 = 2.78 f4 = -2.07	0.039	f1 = -2.61 f2 = -12.63 f3 = 9.70 f4 = -2.37	0.386
39	”	12	16	4	*	-	f1 = -5.09 f2 = -9.83 f3 = -9.53 f4 = -6.18	0.062	f1 = -5.25 f2 = -14.25 f3 = -8.25 f4 = -1.00	31.034
40	”	10	14	5	*	-	f1 = -1.07 f2 = -3.83 f3 = -5.53 f4 = -16.87 f5 = -8.42	0.051	f1 = -5.16 f2 = -2.79 f3 = -4.38 f4 = -18.70 f5 = -9.69	102.952
41	”	7	6	3	f1 = -29.40 f2 = -65.30 f3 = -39.30	0.447	f1 = -10.74 f2 = -32.20 f3 = -24.39	0.044	f1 = -29.40 f2 = -65.30 f3 = -39.30	0.165
42	”	7	7	3	f1 = -62.18 f2 = -93.50	0.583	f1 = -47.39 f2 = -86.99	0.039	f1 = -62.18 f2 = -93.50	0.153

Continued on next page

Table 4.3 – continued from previous page

Algorithm					EMSA		ASIMOLP		BOA	
Prob.	**Origin**	**n**	**m**	**q**	**MPNP**	**CPU (s)**	**MPNP**	**CPU (s)**	**MPNP**	**CPU (s)**
					f3 = -52.00		f3 = -54.78		f3 = -52.00	
43	”	6	6	4	f1 = -37.50 f2 = -11.25 f3 = -7.50 f4 = -20.25	1.311	f1 = -10.40 f2 = -6.52 f3 = -5.91 f4 = -0.34	0.032	f1 = -37.50 f2 = -11.25 f3 = -7.50 f4 = -20.25	0.158
44	”	6	6	4	f1 = 34.50 f2 = -7.50 f3 = -56.00 f4 = -31.50	1.856	f1 = 28.39 f2 = -6.38 f3 = -45.43 f4 = -25.83	0.029	f1 = 34.50 f2 = -7.50 f3 = -56.00 f4 = -31.50	0.211
45	”	10	14	5	*	-	f1 = 3.35 f2 = -3.18 f3 = -2.48 f4 = -2.50 f5 = 2.10	0.043	f1 = 1.03 f2 = -2.19 f3 = 2.01 f4 = -8.13 f5 = -7.22	307.611
46	”	10	14	5	*	-	f1 = 2.35 f2 = 0.73 f3 = -11.72 f4 = -1.90 f5 = -10.33	0.057	f1 = -4.95 f2 = -3.42 f3 = -4.38 f4 = -18.91 f5 = -9.27	105.344
47	”	7	7	3	f1 = -3.83 f2 = -76.46 f3 = -49.57	0.601	f1 = -6.82 f2 = -68.93 f3 = -25.08	0.038	f1 = -3.83 f2 = -76.46 f3 = -49.57	0.159
48	Bensolve 1.2	100	101	2	*	-	f1 = -4.89 f2 = -106.50	0.037	f1 = -8.42 f2 = -116.65	0.503
49	Bensolve 2.0	5	31	5	*	-	f1 = 0.00 f2 = 0.00 f3 = 0.00 f4 = 0.00 f5 = 0.01	0.021	f1 = 0.00 f2 = -1.00 f3 = 0.00 f4 = 0.00 f5 = -2.00	2.877
50	Script of Prob. 53	36	36	2	f1 = -5.00 f2 = -26.00	1.921	f1 = -8.41 f2 = -20.12	0.018	f1 = -5.00 f2 = -26.00	0.211
51	”	64	64	2	f1 = -63.00 f2 = -7.00	14.455	f1 = -52.24 f2 = -10.56	0.028	f1 = -63.00 f2 = -7.00	0.403
52	”	100	100	2	f1 = -124.00	125.495	f1 = -119.51	0.031	f1 = -124.00	0.621

Continued on next page

Table 4.3 – continued from previous page

Algorithm					EMSA		ASIMOLP		BOA	
Prob.	**Origin**	**n**	**m**	**q**	**MPNP**	**CPU (s)**	**MPNP**	**CPU (s)**	**MPNP**	**CPU (s)**
					f2 = -9.00		f2 = -11.72		f2 = -9.00	
53	Bensolve 2.0	343	343	3	x	-	f1 = -35.32 f2 = -101.41 f3 = -9.63	0.651	f1 = -42.00 f2 = -294.00 f3 = -6.00	55.302
54	MOPLIB	30	21	12	f1=5.0E-12, f2=5.0E-12 f3=5.0E-12, f4=5.0E-12 f5=5.0E-12, f6=5.0E-12 f7=5.0E-12, f8=5.0E-12 f9=5.0E-12, f10=5.0E-12 f11=5.0E-12, f12=-5.5E-11	7.235	+	-	f1=5.0E-12, f2=5.0E-12 f3=5.0E-12, f4=5.0E-12 f5=5.0E-12, f6=5.0E-12 f7=5.0E-12, f8=5.0E-12 f9=5.0E-12, f10=5.0E-12 f11=5.0E-12, f12=-5.5E-11	0.598
55	”	4492	1003	4	x	-	f1 = 42.70 f2 = 83.10 f3 = 0.00 f4 = -725.90	18.281	x	-
56	”	100	20	3	f1 = -168.00 f2 = -124.00 f3 = -143.00	5.281	f1 = -61.18 f2 = -73.93 f3 = -96.38	0.011	f1 = -168.00 f2 = -124.00 f3 = -143.00	4.291
57	”	53	221	3	*	-	f1 = -19.23 f2 = -54.56 f3 = 0.00	0.725	f1 = 0.00 f2 = -2.00 f3 = -13959.00	1663.803
58	”	53	226	3	f1 = -188.00 f2 = -123.00 f3 = 16842.00	28.116	f1 = -50.22 f2 = -47.01 f3 = 0.00	0.196	f1 = -188.00 f2 = -123.00 f3 = 16842.00	6.551
59	”	900	60	4	*	-	f1 = -283.50 f2 = -352.74 f3 = -268.85 f4 = -372.16	0.262	*	-
60	”	1143	1211	3	x	-	f1 = 2.11 f2 = 51.82 f3 = 0.61	0.725	x	-
61	”	218	28	27	*	-	f1=0.52,f2=0.01,f3=6.95 f4=2.41,f5=4.94,f6=-5.53 f7=-9.71,f8=2.23,f9=-0.31 f10=2.67,f11=-3.19,f12=-5.55	0.481	*	-

Continued on next page

Table 4.3 – continued from previous page

Algorithm					EMSA		ASIMOLP		BOA	
Prob.	**Origin**	**n**	**m**	**q**	**MPNP**	**CPU (s)**	**MPNP**	**CPU (s)**	**MPNP**	**CPU (s)**
							f13=-5.42,f14=-4.46,f15=-4.35 f16=6.01,f17=-3.36,f18=1.71 f19=-8.01,f20=8.90,f21=8.01 f22=-5.35,f23=5.35,f24=5.35 f25=5.35,f26=-5.37,f27=-4.35			

(*) Aborted after 3 days of running time

(+) No initial starting solution

(x) Out of memory

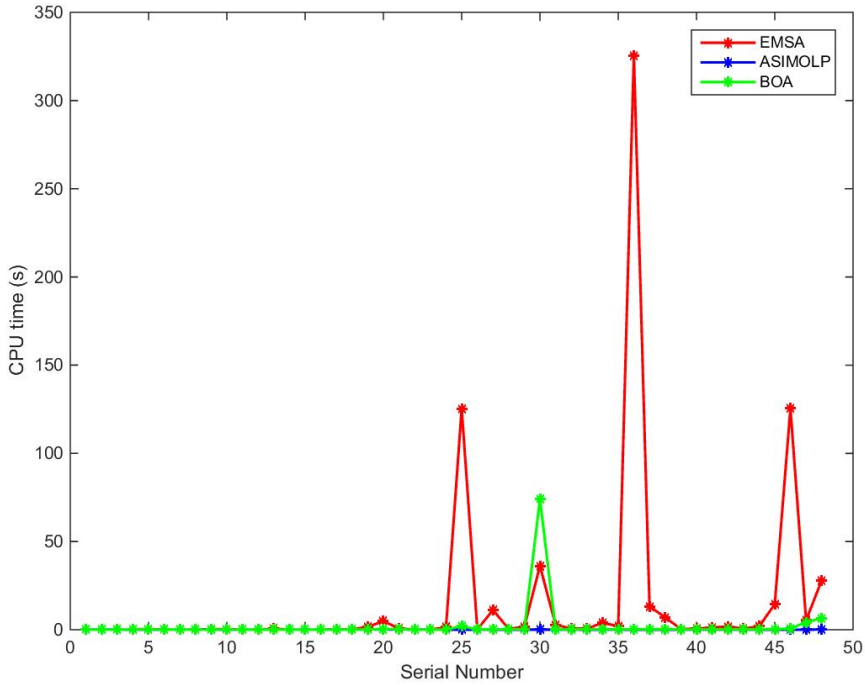

Figure 4.3: Running time of EMSA, ASIMOLP and BOA for the 48 instances solved.

Chapter 5

COMPARATIVE STUDY OF TWO KEY ALGORITHMS IN MOLP

5.1 Introduction

MOLP has been an active area of research since the 60s. During this period, various algorithms have been developed for generating either the entire efficient or nondominated set, or a subset of it, or a most preferred efficient or nondominated point for the problem. Most of these approaches are decision space-based. However, objective space-based methods are becoming more and more prominent.

This chapter presents a detailed computational investigation of two MOLP algorithms namely the PSA of Rudloff *et al.* [124] and Algo-

rithm 5 which is the primal variant of BOA [30]. These two algorithms are based on the same solution concept introduced by Lohne [100]. They are similar in output but different in philosophy since one of them, BOA, is an objective space-based search algorithm, while PSA is a decision space-based algorithm. They are prominent examples of MOLP algorithms. Unfortunately, there is no empirical evidence in the literature, as far as we can tell, that separates them in terms of the quality of a most preferred nondominated point they returned and robustness. This chapter intends to fill this gap. It considers all available test problems ranging from small size to large size and solves them with Matlab implementations of the two algorithms on the same machine. The extensive results are recorded and discussed. It is hoped that these results would be a valuable guideline for the potential users to choose between the two depending on the problems they have to solve. Note that to the best of our knowledge, no one tested these algorithms as extensively as we have done here in terms of the size and variety of problems.

The PSA [124] work in the decision space and does not intend to find the set of all efficient solutions of the problem, rather it finds a subset of efficient solutions that allows to generate the whole efficient

frontier and also returns the corresponding nondominated points and directions. BOA on the other hand, works in the objective space to find the set of all nondominated points as well as directions of the problem.

In this chapter, we shall also act as the DM and choose a Most Preferred Nondominated Point (MPNP) whose components are as close as possible to an unattainable ideal objective point from the nondominated set returned by PSA to compare with a MPNP returned by BOA. The rest of the chapter is organized as follows. The PSA is presented in Section 5.2. We present another illustration of BOA in Section 5.3. Section 5.4 presents experimental results on the two algorithms to compare their computing efficiency, robustness and the quality of a MPNP. We present the summary of experimental results in Section 5.5. The chapter summary is presented in Section 5.5.

5.2 The Parametric Simplex Algorithm

In this section, we presents the PSA of Rudloff, Ulus and Vanderbei [124]. This algorithm is one of the current solution approaches for the problem. It can be viewed as a variant of the algorithm in [63],

with a similar structure. It is different in the sense that it does not find all the efficient extreme points and unbounded efficient edges (extreme rays) as is being done in [63]. As mentioned earlier, the algorithm works in the decision space and finds a solution based on the idea of Löhne [100] i.e., it finds a finite subset of efficient extreme points and directions that allows to generate the whole efficient frontier. The algorithm is initialized by solving an LP to find a weight vector, such that the weighted sum problem using this weight vector yields an optimal solution that is efficient. The corresponding optimal dictionary is used to construct an initial dictionary D^0 and an index set of entering variables J^{D^0}. A dictionary contains a set of basic and nonbasic variables at each iteration. The optimal solution is used as an initial efficient basic feasible solution x^0. Its implementation stores a set of Boundary Dictionaries (BD) containing dictionaries that are not yet visited and a set of Visited Dictionaries (VD) that contains dictionaries that are already visited. At each iteration, the algorithm moves from one dictionary to another, collecting their basic solutions into a set of efficient solutions $\bar{X}$ for output until all the dictionaries are visited. More specifically, the algorithm performs pivoting for only one leaving variable among the set of all possible

leaving variables and picks only one entering variable thereby making it computationally efficient. In addition, rather than solving a vertex enumeration problem which is more costlier as is being done in [30], the algorithm finds a set of parameters that guarantees the efficiency of the current vertex and eliminates the redundant inequalities thereof. This is done by solving a parametrized LP to check if an inequality is defining or redundant; if redundant it would be eliminated which also improves computation time. For unbounded problems where there is no leaving variable for an entering variable, a corresponding homogenous problem is solved and the solution found forms the extreme directions of the problem. When all the dictionaries are visited, the algorithm stops and returns the set of efficient extreme points $\bar{X}$ and directions $\bar{X}^h$ as well as the corresponding nondominated vertices $\bar{Y}$ and directions $\bar{Y}^h$ of the problem. Before we describe the pseudo-code form of PSA, we first explain the used notation.

Notation: A, b, C are the problem data; D^0 is the initial dictionary; J^{D^0} is an initial index set of entering variables; x^0, an initial efficient basic feasible solution corresponding to the initial dictionary; E^{D^0} is the set of explored pivots for the initial dictionary; R, the recession

cone of the image; J^D is the index set of entering variables; BD is the set of boundary dictionaries; N, the set of nonbasic variables; B, the set of basic variables; the indices i, j correspond to basic variable $x_i \in B$ and nonbasic variable $x_j \in N$ respectively; B^{-1}, is the inverse of the basic matrix; e^j is a unit column vector corresponding to the nonbasic variable x_j; x^h is a direction of the recession cone in the decision space; $P^T[\bar{X}^h]$ is the image of the direction; $-Z_N^T e^j$ is the objective value of the homogeneous problem; E^D is the set of explored pivots for the current dictionary; $\bar{D}$ is the new dictionary; $\bar{N}$ is the updated nonbasic variables; $E^{\bar{D}}$ is the set of all explored pivots of the new dictionary $\bar{D}$; $\bar{x}$ is the basic feasible solution for the new dictionary $\bar{D}$; VD, the set of visited dictionaries; $\bar{X}$ is the set of efficient extreme points in the decision space; $\bar{X}^h$, the set of extreme directions in the decision space; $\bar{Y}$ is the set of all nondominated vertices in the objective space and $\bar{Y}^h$, is the set of extreme directions in the objective space.

Algorithm 8 Parametric Simplex Algorithm

Input: A, b, C

Initialize: $Find\ D^0\ and\ the\ index\ set\ of\ entering\ variables\ J^{D^0}$;

$BD \leftarrow \{D^0\}, \quad \bar{X} \leftarrow \{x^0\}, \bar{Y^h} \leftarrow \emptyset, VS \leftarrow \emptyset, \quad \bar{X}^h \leftarrow \emptyset, \quad E^{D^0} \leftarrow \emptyset.$

while $BD \neq \emptyset$ **do**

  $Let\ D \in BD\ with\ nonbasic\ variables\ N\ and\ index\ set\ of\ entering\ variables\ J^D$;

  for $j \in J^D$ **do**

    $Let\ x_j\ be\ the\ entering\ variable$;

    if $B^{-1}Ne^j \leq 0$ **then**

      $Let\ x^h\ be\ such\ that\ x^h_B = -B^{-1}Ne^j\ and\ x^h_N = e^j$;

      $\bar{X}^h \leftarrow \bar{X}^h \cup \{x^h\}$

      $\bar{Y}^h \leftarrow P^T[\bar{X}^h] \cup \{-Z^T_N e^j\}$

    else

      $Pick\ i \in argmin_{i \in B, (B^{-1}N)_{ij} > 0} \frac{(B^{-1}b)_i}{(B^{-1}N)_{ij}}$;

      if $(j, i) \notin E^D$ **then**

        $Perform\ the\ pivot\ with\ entering\ variable\ x_j\ and\ leaving\ variable x_i$;

        $Call\ the\ new\ dictionary\ \bar{D}\ with\ nonbasic\ variables\ \bar{N} = N \cup \{i\} \backslash \{j\}$;

        if $\bar{D} \notin VS$ **then**

          if $\bar{D} \in BD$ **then**

          $E^{\bar{D}} \leftarrow E^{\bar{D}} \cup \{(i, j)\}$;

          else

            $Let\ \bar{x}\ be\ the\ basic\ solution\ for\ \bar{D}$;

            $\bar{X} \leftarrow \bar{X} \cup \{\bar{x}\}$;

            $\bar{Y} \leftarrow P^T[\bar{X}] \cup \{P^T\bar{x}\}$;

            $Compute\ the\ index\ set\ of\ entering\ variables\ J^{\bar{D}}\ of\ \bar{D}$;

            $Let\ E^{\bar{D}} = \{(i, j)\}$;

            $BD \leftarrow BD \cup \{\bar{D}\}$;

          endif

        endif

      endif

    endif

  endfor

  $VS \leftarrow VS \cup \{D\}, \quad BD \leftarrow BD \backslash \{D\}$

endwhile

Output: $(\bar{X}, \bar{X^h})$: $Set\ of\ Efficient\ solutions\ and\ directions$;

$(\bar{Y}, \bar{Y^h})$: $Nondominated\ set\ and\ directions.$

5.2.1 Illustration of the PSA

Consider the following MOLP adapted from Alves *et al.* [3], which we solved using a Matlab implementation of PSA.

$$\min f_1 = -3x_1 - x_2$$

$$\min f_2 = -x_1 - 4x_2$$

Subject to

$$\begin{aligned} -x_1 + x_2 &\leq 2 \\ x_1 + x_2 &\leq 7 \\ x_1 + 2x_2 &\leq 10 \\ x_1, \quad x_2 &\geq 0 \end{aligned} \tag{5.1}$$

The efficient extreme points found are $x^1 = (2.0,\ 4.0)^T$, $x^2 = (4.0,\ 3.0)^T$ and $x^3 = (7.0,\ 0.0)^T$. The corresponding nondominated points are $f^1 = (-10.0,\ -18.0)^T$, $f^2 = (-15.0,\ -16.0)^T$ and $f^3 = (-21.0,\ -7.0)^T$ respectively, where $x^1 = (x_1^1, x_2^1)^T, x^2 = (x_1^2, x_2^2)^T, x^3 = (x_1^3, x_2^3)^T \in X_E$ and $f^1 = (f_1^1, f_2^1)^T$, $f^2 = (f_1^2, f_2^2)^T$, $f^3 = (f_1^3, f_2^3)^T \in Y_N$. The feasible region in the decision space is shown in Figure 5.1.

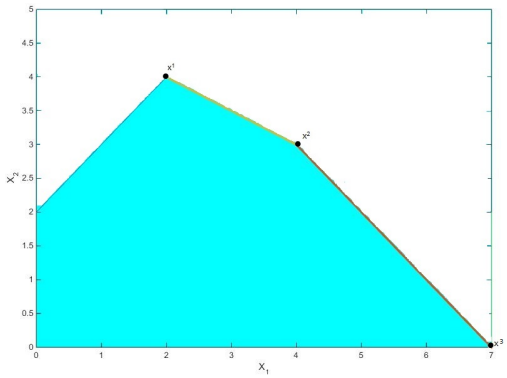

Figure 5.1: Efficient edges joining the three points in the decision space

5.3 Additional illustration of BOA

We now provide another illustration of BOA using Problem 5.1 of Section 5.2.1. The nondominated points found using Bensolve-1.2 [101], are $f^1 = (-21.0, -7.0)^T$, $f^2 = (-15.0, -16.0)^T$ and $f^3 = (-10.0, -18.0)^T$ respectively. These points are shown in Figure 5.2.

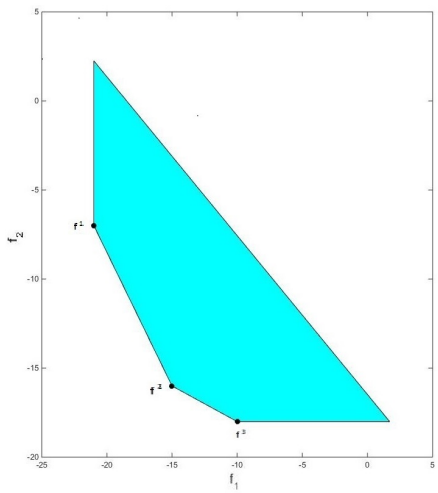

Figure 5.2: Nondominated edges connecting the three points in the objective space.

To select the MPNP using Problem 5.1, solving each of the objective function individually over the feasible region X as was done earlier yields the ideal objective point $y^* = (-21.0, -18.0)^T$. Again, $y^* \notin Y_N$ where $Y_N = \{(-10.0,\ -18.0)^T, (-15.0,\ -16.0)^T, (-21.0,\ -7.0)^T\}$.

Having computed the ideal objective point y^*, we now determine the minimum distance of each nondominated point $\hat{y}$ from it by finding

$$\min \quad \{\|\hat{y}_1 - y^*\|,\ \ \|\hat{y}_2 - y^*\|, \ldots, \|\hat{y}_n - y^*\|\}$$

where $\hat{y} \in Y_N$ has already been found either by PSA or BOA and y^* is the ideal objective point.

Using the nondominated points f^1, f^2 and f^3 of Problem 5.1 yields

$$\|f^1 - y^*\| = 11.0, \quad \|f^2 - y^*\| = 6.3 \qquad \text{and} \qquad \|f^3 - y^*\| = 11.0.$$

Since, the relative distance of f^2 from the ideal point y^* is 6.3 which is the smallest of the three, it therefore means that $f^2 = (-15.0,\ -16.0)^T$ is the closest of the three nondominated points to the ideal point $y^* = (-21.0, -18.0)^T$. Hence, f^2 is selected as the DM's MPNP.

We now used Problem 4.2 of Section 4.4. For PSA, the set of nondominated points found is $Y_N = \{(-8.0, -4.0, 12.0)^T,$ $(-16.0, 0.0, -16.0)^T, (0.0, -8.0, 8.0)^T, (-8.0, 0.0, 8.0)^T\}$ also with $y^* \notin Y_N$.

Next, we measure the distances of each of these points from the ideal point $y^* = (-48.0, -32.0, -16.0)^T$ as was done with those returned by BOA in Section 4.4. It turned out that, the nondominated point $(-16.0, 0.0, -16.0)^T$ is the closest to the ideal point y^* and is selected as the DM's MPNP as shown in Table 5.2, Problem 10. Its distance from it is 55.42 which is bigger than 32 which was the closest when measuring the points returned by BOA in Section 4.4, thereby making the MPNP returned by BOA closer to the ideal point and of higher

quality for this problem.

5.4 Discussion of Experimental Results

In this section, we provide numerical results to compare the computing efficiency, robustness and the quality of a Most Preferred Nondominated Point (MPNP) returned by Algorithms 5 and 8.

Table 5.2 shows the numerical results for a collection of 53 problems, from the existing literature. Problem 1 is taken from Ehrgott [56], Problems 2 to 10 were taken from Zeleny [163]. Problems 11 to 21 are test problems from the interactive MOLP explorer (iMOLPe) of Alves *et al.* [3]. Problems 22 to 47 are taken from Steuer [143]. Problems 48 and 53 are test problem in Bensolve-2.0 of Löhne and Weißing [104], while Problem 52 is a test problem in Bensolve-1.2 of Löhne [101]. Finally, Problems 49 to 51 are obtained using a script in Bensolve-2.0 of Löhne and Weißing [104] that was used to generate problem 53 with the same number of variables and constraints.

We have added twenty-five (25) additional realistic instances to our collection of problems in this chapter. These problems are recorded

in Table 5.3. They include Problems 54 to 72, 78, 80 to 84 and 86. Problems 54 to 72 are such that their constraint and criterion matrices are sparse while all the components of the RHS vectors are zeros except for a one (1) as the only non-zero entry. Problem 78 is such that the RHS vector is dense while the constraint and criterion matrices are sparse. In Problems 80 and 82, the constraint matrices are sparse, criterion matrices are dense and all the elements in the RHS vectors are ones. Problems 83 and 84 are such that the constraint and criterion matrices are sparse, all the components of the RHS vectors are zeros except for a one (1) at the begining as the only non-zero element. Finally, problem 86 is such that, the constraint and criterion matrices as well as the RHS vector are all sparse.

Results for Algorithm 8 were obtained using a Matlab implementation of this algorithm provided by Rudloff *et al.* [124]. We also used Bensolve-1.2 of Löhne [101] here to test their performance with the same tools and for a meaningful comparison. All problems were executed on Intel Core i5-2500 CPU at 3.30GHz with 16.0GB RAM. In all tests, n is the number of variables, m the number of constraints, q the number of objectives, NNP the number of nondominated points returned by the algorithms for each problem and the CPU times (in

seconds). Algorithm 8 is PSA of Rudloff *et al.* [124] and Algorithm 5 is BOA [30] and as presented in [135]. We recorded the CPU times (in seconds) returned by the algorithms for each problem and also acted as the DM by choosing the MPNP (whose components are as close as possible to the ideal objective point as explained in Section 4.4 and 5.3) from the nondominated set $Y_N = \{Cx : x \in X_E\}$ returned by PSA to compare with the MPNP returned by BOA.

As can be seen in Table 5.2, as usual, the CPU times increase as the problem dimension increases. We can also infer from Tables 5.2 and 5.3 that the CPU times also depend to some extent on the total number of nondominated points returned by the algorithm for a given problem. That is to say, the more the number of nondominated points in a given problem the more computational effort would be required to obtain them. We note here that all the problems in Table 5.2 are non-degenerate For these problems, PSA appears to have computational advantage over BOA, most especially for those problems with more nondominated points as it returns only a subset of them; see problems 20, 25, 30, 39, 40 and 46. We noticed that for those problems where both algorithms return the same number of nondominated points, there is a slight difference in CPU time which is in favour of PSA.

We also observed that PSA returns more nondominated points for some of the problems than BOA; this is not supposed to happen as it is meant to return a subset of these points. Some of the nondominated points returned are repeated.

In terms of the quality of a MPNP returned by these algorithms, we observed in Table 5.2 that, both algorithms returned the same MPNP points for most of the problems considered. However, for a few of these problems where the MPNP are not the same, BOA returned higher quality MPNP than PSA as illustrated in page 151.

Next, we use practical size MOLP instances from Csirmaz [42] which is an MOLP solver called Inner, and MOPLIB [103] which stands for Multi-Objective Problem Library. These test problems were also executed on the same machine and the results are reported in Table 5.3. Problems 54 to 72 are from Csirmaz [42] while problems 73 to 86 are from MOPLIB. Note that Problems 54 to 73 are highly degenerate. In fact, the RHS vector is such that all the components are zeros except for a one (1) at the begining, at the end or at the centre as the only none zero element. For these highly degenerate problems, it was observed that BOA is computationally superior to

PSA which confirms what was reported by Rudloff *et al.* [124], that BOA outperforms PSA on highly degenerate problems. Even the nondominated points returned by PSA for these problems are also of lower quality than those returned by BOA.

In terms of robustness we noticed in Tables 5.2 and 5.3 that, PSA could not solve problems 38, 45, 52, 68 and 81. It returns the image which is the whole region which indicates that none of the vertices in the image is nondominated, meaning that no solution is returned thereby making BOA more robust.

For those problems which were solved by BOA in Table 5.3, it was also observed that the MPNPs returned are of higher quality than those returned by PSA. However, for the non-degenerate problems, PSA was found to be computationally superior to BOA.

5.5 Summary of Results

In this section, we present the summary of experimental results discussed in the previous section in Table 5.1. We have also presented the CPU time of BOA and PSA for 70 out of the 86 instances (which

represent 81.40 %) of the total problems solved by both methods in Figure 5.3.

Algorithms	**Criteria for Evaluation**			
	Computing Efficiency		**Robustness**	**Quality of MPNP**
	Degenerate Problems	**Non-degenerate Problems**		
BOA	Computationally superior to PSA on highly degenerate problems	Computationally inferior to PSA on non-degenerate problems	Outperforms PSA in terms of robustness of methods	Returned high quality MPNP than PSA on highly degenerate problems
PSA	Computationally inferior to BOA on highly degenerate problems	Computationally more efficient than BOA on non-degenerate problems	Not so robust as compared to BOA	Returned high quality MPNP as BOA for most of the problems considered

Table 5.1: Summary of experimental results

5.6 Summary

We have presented PSA, illustrated it and BOA on small MOLP instance. We have also presented a detailed computational experience to compare the efficiency, robustness and the quality of MPNP's re-

turned by these algorithms. The CPU times and quality of MPNP's returned by these algorithms for a collection of 86 existing MOLP problems ranging from small to medium and practical size instances is reported. In the next chapter, we shall apply the strawberry multi-objective plant propagation algorithm (MOPPA) [67] and NSGA-II [49,50] which were originally designed to solve multi-objective non-linear programming problems to solve MOLP.

Table 5.2: Comparative results for small to medium instances

Algorithms					**BOA**			**PSA**		
Prob.	**Origin**	**n**	**m**	**q**	**NNP**	**MPNP**	**CPU (s)**	**NNP**	**MPNP**	**CPU (s)**
1	Ehrgott 2006	3	3	3	3	f1 = -2.00 f2 = 10.00 f3 = -5.00	0.038	3	f1 = -2.00 f2 = 10.00 f3 = -5.00	0.031
2	Zeleny 1982	2	2	2	3	f1 = -25000 f2 = -66667	0.021	3	f1 = -25000 f2 = -66667	0.015
3	”	2	4	2	2	f1 = -9.00 f2 = -15.00	0.026	2	f1 = -9.00 f2 = -15.00	0.019
4	”	2	4	3	3	f1 = -3.00 f2 = -7.50 f3 = 9.00	0.161	3	f1 = -3.00 f2 = -7.50 f3 = 9.00	0.121
5	”	2	6	2	3	f1 = -24.00 f2 = -16.00	0.212	3	f1 = -24.00 f2 = -16.00	0.181
6	”	3	3	3	5	f1 = 3.00 f2 = -6.00 f3 = -12.00	0.046	5	f1 = 3.00 f2 = -6.00 f3 = -12.00	0.032
7	”	5	3	3	4	f1 = 0.00 f2 = -4.00 f3 = -24.00	0.043	4	f1 = 0.00 f2 = -4.00 f3 = -23.62	0.041
8	”	5	2	2	1	f1 = -52.00 f2 = -52.00	0.016	1	f1 = -52.00 f2 = -52.00	0.011
9	”	6	4	2	1	f1 = 0.00 f2 = 0.00	0.017	1	f1 = 0.00 f2 = 0.00	0.012
10	”	7	4	3	5	f1 = -48.00 f2 = -32.00 f3 = 16.00	0.163	4	f1 = -16.00 f2 = 0.00 f3 = -16.00	0.152

Continued on next page

Table 5.2 – continued from previous page

Algorithms					BOA			PSA		
Prob.	**Origin**	**n**	**m**	**q**	**NNP**	**MPNP**	**CPU (s)**	**NNP**	**MPNP**	**CPU (s)**
11	iMOLPe	2	3	2	3	f1 = -21.00 f2 = -7.00	0.047	3	f1 = -21.00 f2 = -7.00	0.035
12	”	3	3	4	3	f1 = -10.00 f2 = -20.00 f3 = -100.00 f4 = -10.00	0.033	3	f1 = -10.00 f2 = -20.00 f3 = -100.00 f4 = -10.00	0.021
13	”	3	5	3	10	f1 = -21.00 f2 = -4.50 f3 = -4.00	0.042	10	f1 = -21.00 f2 = -4.50 f3 = -4.00	0.033
14	”	3	3	3	7	f1 = -2.66 f2 = -2.00 f3 = -0.33	0.035	7	f1 = -2.66 f2 = -2.00 f3 = -0.33	0.032
15	”	4	3	3	8	f1 = -48.50 f2 = -19.50 f3 = -37.00	0.038	8	f1 = -48.50 f2 = -19.50 f3 = -37.00	0.036
16	”	4	2	3	6	f1 = -20.00 f2 = -80.00 f3 = -40.00	0.036	6	f1 = -20.00 f2 = -80.00 f3 = -40.00	0.035
17	”	4	4	3	11	f1 = -40.00 f2 = -50.00 f3 = -10.00	0.186	11	f1 = -40.00 f2 = -50.00 f3 = -10.00	0.162
18	”	3	3	3	5	f1 = 0.00 f2 = -2.00 f3 = -4.00	0.033	5	f1 = 0.00 f2 = -2.00 f3 = -4.00	0.031
19	”	15	10	2	11	f1 = -363.82 f2 = -33.70	0.195	7	f1 = -229.18 f2 = -35.31	0.125

Continued on next page

Table 5.2 – continued from previous page

Algorithms					BOA			PSA		
Prob.	**Origin**	**n**	**m**	**q**	**NNP**	**MPNP**	**CPU (s)**	**NNP**	**MPNP**	**CPU (s)**
20	”	15	10	3	37	f1 = -363.82 f2 = -33.70 f3 = -136.71	0.476	7	f1 = -134.17 f2 = -32.88 f3 = -135.82	0.301
21	”	10	5	3	14	f1 = 226.40 f2 = -501.86 f3 = -351.14	0.623	14	f1 = 223.09 f2 = -496.23 f3 = -246.64	0.589
22	Steuer 1986	5	5	2	5	f1 = -10.00 f2 = -3.00	0.036	5	f1 = -10.00 f2 = -3.00	0.034
23	”	4	4	3	3	f1 = 3.42 f2 = -10.28 f3 = -3.42	0.015	3	f1 = 3.42 f2 = -10.28 f3 = -3.42	0.012
24	”	5	5	4	14	f1 = 1.02 f2 = -25.46 f3 = 24.44 f4 = -28.32	0.098	14	f1 = 1.02 f2 = -25.46 f3 = 24.44 f4 = -28.32	0.081
25	”	10	8	4	72	f1 = 106.29 f2 = -462.13 f3 = 175.57 f4 = -33.41	1.973	65	f1 = 183.36 f2 = -424.26 f3 = 117.29 f4 = -4.03	0.921
26	”	5	4	3	9	f1 = -52.07 f2 = 31.50 f3 = -17.35	0.054	8	f1 = -52.07 f2 = 31.50 f3 = -17.35	0.045
27	”	6	8	4	14	f1 = -6.94 f2 = -5.38 f3 = 6.83 f4 = -9.16	0.065	6	f1 = -6.94 f2 = -5.38 f3 = 6.83 f4 = -9.16	0.053

Continued on next page

Table 5.2 – continued from previous page

Algorithms					BOA			PSA		
Prob.	**Origin**	**n**	**m**	**q**	**NNP**	**MPNP**	**CPU (s)**	**NNP**	**MPNP**	**CPU (s)**
28	”	7	6	4	15	f1 = -31.53 f2 = -26.48 f3 = -26.57 f4 = -0.34	0.286	12	f1 = -31.53 f2 = -26.48 f3 = -26.57 f4 = -0.34	0.555
29	”	7	6	4	9	f1 = 26.80 f2 = -37.73 f3 = -24.33 f4 = -59.60	0.192	9	f1 = 26.80 f2 = -37.73 f3 = -24.33 f4 = -59.60	0.142
30	”	8	8	6	286	f1 = -74.00 f2 = -107.50 f3 = -41.25 f4 = -27.25 f5 = -9.00 f6 = -30.75	73.963	40	f1 = -77.00 f2 = -52.00 f3 = -16.00 f4 = -52.40 f5 = 26.00 f6 = -20.00	0.699
31	”	8	8	3	5	f1 = -36.57 f2 = -22.28 f3 = -14.00	0.168	5	f1 = -36.57 f2 = -22.28 f3 = -14.00	0.156
32	”	8	8	3	12	f1 = -14.03 f2 = -18.00 f3 = -4.93	0.135	1	f1 = -6.50 f2 = -11.00 f3 = -7.50	0.121
33	”	5	5	4	12	f1 = -21.50 f2 = -39.25 f3 = -16.25 f4 = 27.00	0.277	8	f1 = -8.00 f2 = -23.87 f3 = -7.62 f4 = 27.00	0.216
34	”	6	6	3	17	f1 = -12.65 f2 = 0.00	0.212	17	f1 = 13.62 f2 = -9.75	0.210

Continued on next page

Table 5.2 – continued from previous page

Algorithms					BOA			PSA		
Prob.	**Origin**	**n**	**m**	**q**	**NNP**	**MPNP**	**CPU (s)**	**NNP**	**MPNP**	**CPU (s)**
						f3 = -30.15			f3 = -26.25	
35	”	5	5	4	9	f1 = -14.66 f2 = -21.06 f3 = 35.73 f4 = -16.00	0.462	2	f1 = -14.00 f2 = 0.00 f3 = 27.00 f4 = 0.00	0.345
36	”	10	10	4	6	f1 = 46.50 f2 = 19.21 f3 = -27.07 f4 = -27.07	0.333	6	f1 = 46.50 f2 = 19.21 f3 = -27.07 f4 = -27.07	0.241
37	”	8	8	3	13	f1 = -14.48 f2 = -4.74 f3 = 6.93	0.217	13	f1 = -14.48 f2 = -4.74 f3 = 6.93	0.201
38	”	6	7	4	21	f1 = -2.61 f2 = -12.63 f3 = 9.70 f4 = 2.37	0.386	*	-	-
39	”	12	16	4	601	f1 = -5.25 f2 = -14.25 f3 = -8.25 f4 = -1.00	31.034	23	f1 = -5.13 f2 = -3.38 f3 = 1.83 f4 = -1.18	0.982
40	”	10	14	5	132	f1 = -5.16 f2 = -2.79 f3 = -4.38 f4 = -18.70 f5 = -9.69	102.952	9	f1 = -18.00 f2 = 70.60 f3 = -3.20 f4 = 8.00 f5 = -4.30	1.395
41	”	7	6	3	3	f1 = -29.40	0.165	3	f1 = -29.40	0.151

Continued on next page

Table 5.2 – continued from previous page

Algorithms					BOA			PSA		
Prob.	**Origin**	**n**	**m**	**q**	**NNP**	**MPNP**	**CPU (s)**	**NNP**	**MPNP**	**CPU (s)**
						f2 = -65.30 f3 = -39.30			f2 = -65.30 f3 = -39.30	
42	”	7	7	3	7	f1 = -62.18 f2 = -93.50 f3 = -52.00	0.036	7	f1 = -62.18 f2 = -93.50 f3 = -52.00	0.036
43	”	6	6	4	5	f1 = -37.50 f2 = -11.25 f3 = -7.50 f4 = -20.25	0.158	5	f1 = -37.50 f2 = -11.25 f3 = -7.50 f4 = -20.25	0.134
44	”	6	6	4	10	f1 = 34.50 f2 = -7.50 f3 = -56.00 f4 = -31.50	0.211	10	f1 = 34.50 f2 = -7.50 f3 = -56.00 f4 = -31.50	0.201
45	”	10	14	5	471	f1 = 1.03 f2 = -2.19 f3 = 2.01 f4 = -8.13 f5 = 7.22	307.611	*	-	-
46	”	10	14	5	128	f1 = -4.95 f2 = -3.42 f3 = -4.38 f4 = -18.91 f5 = -9.27	105.344	1	f1 = -4.93 f2 = -5.57 f3 = -2.83 f4 = -16.28 f5 = -6.13	0.291
47	”	7	7	3	6	f1 = -3.83 f2 = -76.46 f3 = -49.57	0.045	9	f1 = -3.83 f2 = -76.46 f3 = -49.57	0.031

Continued on next page

Table 5.2 – continued from previous page

Algorithms					BOA			PSA		
Prob.	**Origin**	**n**	**m**	**q**	**NNP**	**MPNP**	**CPU (s)**	**NNP**	**MPNP**	**CPU (s)**
48	Bensolve-2.0	5	31	5	22	f1 = 0.00 f2 = -1.00 f3 = 0.00 f4 = 0.00 f5 = -2.00	2.877	1	f1 = 0.00 f2 = 0.00 f3 = 0.00 f4 = 0.00 f5 = 0.00	0.125
49	”	36	36	2	8	f1 = -5.00 f2 = -26.00	0.211	82	f1 = -5.00 f2 = -25.50	0.772
50	”	64	64	2	14	f1 = -63.00 f2 = -7.00	0.403	292	f1 = -34.50 f2 = -9.50	5.167
51	”	100	100	2	20	f1 = -124.00 f2 = -9.00	0.621	1102	f1 = -123.50 f2 = -9.00	36.323
52	Bensolve-1.2	100	101	2	32	f1 = -8.42 f2 = -116.65	0.503	*	-	-
53	Bensolve-2.0	343	343	3	1,368	f1 = -42.00 f2 = -294.00 f3 = -6.00	55.302	x	-	-

(*) The image is the whole region, implying that the problem has no solution

(x) Out of memory

Table 5.3: Comparative results for large instances (NNP stands for Number of Nondominated Points)

Algorithms					BOA			PSA		
Prob.	**Origin**	**n**	**m**	**q**	**NNP**	**MPNP**	**CPU (s)**	**NNP**	**MPNP**	**CPU (s)**
54	inner	844	12	10	1	f1 = -1.00, f2 = -1.00, f3= 0.00 f4 = 0.00, f5 = 0.00, f6 = 0.00 f7 = 0.00, f8 = 0.00, f9 = 0.00 f10 = 0.00	0.871	1	f1 = 0.00, f2 = 0.00, f3= 0.00 f4 = 0.00, f5 = 0.00, f6 = 0.00 f7 = 0.00, f8 = 0.00, f9 = 0.00 f10 = 0.00	2.835
55	”	853	12	10	1	f1 = -1.00, f2 = -1.00, f3= 0.00 f4 = 0.00, f5 = 0.00, f6 = 0.00 f7 = 0.00, f8 = 0.00, f9 = 0.00 f10 = 0.00	0.892	1	f1 = 0.00, f2 = 0.00, f3= 0.00 f4 = 0.00, f5 = 0.00, f6 = 0.00 f7 = 0.00, f8 = 0.00, f9 = 0.00 f10 = 0.00	2.888
56	”	857	12	10	1	f1 = -1.00, f2 = -1.00, f3= 0.00 f4 = 0.00, f5 = 0.00, f6 = 0.00 f7 = 0.00, f8 = 0.00, f9 = 0.00 f10 = 0.00	0.871	1	f1 = 0.00, f2 = 0.00, f3= 0.00 f4 = 0.00, f5 = 0.00, f6 = 0.00 f7 = 0.00, f8 = 0.00, f9 = 0.00 f10 = 0.00	2.922
57	”	873	12	10	1	f1 = 0.00, f2 = -1.00, f3= -1.00 f4 = 0.00, f5 = 0.00, f6 = 0.00 f7 = 0.00, f8 = 0.00, f9 = 0.00 f10 = 0.00	0.884	1	f1 = 0.00, f2 = 0.00, f3= 0.00 f4 = 0.00, f5 = 0.00, f6 = 0.00 f7 = 0.00, f8 = 0.00, f9 = 0.00 f10 = 0.00	3.041
58	”	877	12	10	1	f1 = 0.00, f2 = -1.00, f3= -1.00 f4 = 0.00, f5 = 0.00, f6 = 0.00 f7 = 0.00, f8 = 0.00, f9 = 0.00 f10 = 0.00	0.935	1	f1 = 0.00, f2 = 0.00, f3= 0.00 f4 = 0.00, f5 = 0.00, f6 = 0.00 f7 = 0.00, f8 = 0.00, f9 = 0.00 f10 = 0.00	3.071
59	”	880	12	10	1	f1 = 0.00, f2 = -1.00, f3= -1.00 f4 = 0.00, f5 = 0.00, f6 = 0.00 f7 = 0.00, f8 = 0.00, f9 = 0.00 f10 = 0.00	0.968	1	f1 = 0.00, f2 = 0.00, f3= 0.00 f4 = 0.00, f5 = 0.00, f6 = 0.00 f7 = 0.00, f8 = 0.00, f9 = 0.00 f10 = 0.00	3.113
60	”	882	12	10	1	f1 = -1.00, f2 = 0.00, f3= -1.00 f4 = 0.00, f5 = 0.00, f6 = 0.00 f7 = 0.00, f8 = 0.00, f9 = 0.00 f10 = 0.00	1.009	1	f1 = 0.00, f2 = 0.00, f3= 0.00 f4 = 0.00, f5 = 0.00, f6 = 0.00 f7 = 0.00, f8 = 0.00, f9 = 0.00 f10 = 0.00	3.115
61	”	886	12	10	2	f1 = 0.00, f2 = 0.00, f3= 0.00 f4 = 1.00, f5 = -1.00, f6 = -1.00 f7 = 0.00, f8 = 0.00, f9 = 0.00 f10 = 0.00	1.341	1	f1 = 0.00, f2 = 0.00, f3= 0.00 f4 = 0.00, f5 = 0.00, f6 = 0.00 f7 = 0.00, f8 = 0.00, f9 = 0.00 f10 = 0.00	3.118

Continued on next page

Table 5.3 – continued from previous page

Algorithms					BOA			PSA		
Prob.	**Origin**	**n**	**m**	**q**	**NNP**	**MPNP**	**CPU (s)**	**NNP**	**MPNP**	**CPU (s)**
62	”	888	12	10	1	f1 = -1.00, f2 = -1.00, f3= 0.00 f4 = 0.00, f5 = 0.00, f6 = 0.00 f7 = 0.00, f8 = 0.00, f9 = 0.00 f10 = 0.00	1.104	1	f1 = 0.00, f2 = 0.00, f3= 0.00 f4 = 0.00, f5 = 0.00, f6 = 0.00 f7 = 0.00, f8 = 0.00, f9 = 0.00 f10 = 0.00	3.119
63	”	1009	12	10	1	f1 = 0.00, f2 = -1.00, f3= -1.00 f4 = 0.00, f5 = 0.00, f6 = 0.00 f7 = 0.00, f8 = 0.00, f9 = 0.00 f10 = 0.00	1.281	1	f1 = 0.00, f2 = 0.00, f3= 0.00 f4 = 0.00, f5 = 0.00, f6 = 0.00 f7 = 0.00, f8 = 0.00, f9 = 0.00 f10 = 0.00	3.911
64	”	1956	12	10	1	f1 = -1.00, f2 = -1.00, f3= 0.00 f4 = 0.00, f5 = 0.00, f6 = 0.00 f7 = 0.00, f8 = 0.00, f9 = 0.00 f10 = 0.00	4.288	1	f1 = 0.00, f2 = 0.00, f3= 0.00 f4 = 0.00, f5 = 0.00, f6 = 0.00 f7 = 0.00, f8 = 0.00, f9 = 0.00 f10 = 0.00	13.374
65	”	1983	12	10	1	f1 = 0.00, f2 = 0.00, f3= 0.00 f4 = -1.00, f5 = -1.00, f6 = 0.00 f7 = 0.00, f8 = 0.00, f9 = 0.00 f10 = 0.00	4.418	1	f1 = 0.00, f2 = 0.00, f3= 0.00 f4 = 0.00, f5 = 0.00, f6 = 0.00 f7 = 0.00, f8 = 0.00, f9 = 0.00 f10 = 0.00	13.805
66	”	3722	338	10	55	f1 = -0.25, f2 = -0.50, f3= -2.75, f4 = 1.87, f5 = -0.62, f6 = 0.00, f7 = 0.00, f8 = 0.00, f9 = 0.00 f10 = 0.00	22.167	1	f1 = 0.00, f2 = 0.00, f3= 0.00 f4 = 0.00, f5 = 0.00, f6 = 0.00 f7 = 0.00, f8 = 0.00, f9 = 0.00 f10 = 0.00	56.198
67	”	3725	338	10	61	f1 = -0.20, f2 = -0.40, f3= -2.40, f4 = -2.20, f5 = -0.6, f6 = -0.20 f7 = 0.00, f8 = 0.00, f9 = 0.00 f10 = 0.00	24.605	1	f1 = 0.00, f2 = 0.00, f3= 0.00 f4 = 0.00, f5 = 0.00, f6 = 0.00 f7 = 0.00, f8 = 0.00, f9 = 0.00 f10 = 0.00	56.458
68	”	3897	362	10	+	-	-	*	-	-
69	”	5646	492	10	1575	f1 = -0.38, f2 = 0.00, f3= -2.55, f4 = -1.66, f5 = -0.16, f6=-0.44 f7 = 0.00, f8 = 0.00, f9 = 0.00 f10 = 0.00	125.488	1	f1 = 0.00, f2 = 0.00, f3= 0.00 f4 = 0.00, f5 = 0.00, f6 = 0.00 f7 = 0.00, f8 = 0.00, f9 = 0.00 f10 = 0.00	130.159
70	”	8891	707	10	13	f1 = -0.20, f2 = 0.00, f3= -2.20, f4 = -2.20, f5 = -0.20, f6=-0.20 f7 = 0.00, f8 = 0.00, f9 = 0.00	228.312	1	f1 = 0.00, f2 = 0.00, f3= 0.00 f4 = 0.00, f5 = 0.00, f6 = 0.00 f7 = 0.00, f8 = 0.00, f9 = 0.00	329.701

Continued on next page

Table 5.3 – continued from previous page

Algorithms					BOA			PSA		
Prob.	**Origin**	**n**	**m**	**q**	**NNP**	**MPNP**	**CPU (s)**	**NNP**	**MPNP**	**CPU (s)**
						f10 = 0.00			f10 = 0.00	
71	”	9472	707	10	+	-	-	1	f1 = 0.00, f2 = 0.00, f3= 0.00 f4 = 0.00, f5 = 0.00, f6 = 0.00 f7 = 0.00, f8 = 0.00, f9 = 0.00 f10 = 0.00	362.449
72	”	10017	779	10	31	f1=-0.11, f2 = 0.00, f3= -2.44, f4 = -2.44, f5 = -0.55, f6=-0.55 f7 = 0.00, f8 = 0.00, f9 = 0.00 f10 = 0.00	260.494	1	f1 = 0.00, f2 = 0.00, f3= 0.00 f4 = 0.00, f5 = 0.00, f6 = 0.00 f7 = 0.00, f8 = 0.00, f9 = 0.00 f10 = 0.00	412.929
73	MOPLIB	30	21	12	1	f1=5.0E-12, f2=5.0E-12 f3=5.0E-12, f4=5.0E-12 f5=5.0E-12, f6=5.0E-12 f7=5.0E-12, f8=5.0E-12 f9=5.0E-12, f10=5.0E-12 f11=5.0E-12, f12=-5.5E-11	0.598	1	f1 = 0, f2 = 0 f3 = 0, f4 = 0 f5 = 0, f6 = 0 f7 = 0 , f8 = 0 f9 = 0, f10 = 0 f11 = 0, f12 = 0	0.167
74	”	100	20	3	291	f1 = -168.00 f2 = -124.00 f3 = -143.00	4.291	1	f1 = -168.00 f2 = -124.00 f3 = -143.00	0.122
75	”	53	221	3	2552	f1 = 0.00 f2 = -2.00 f3 = -13959.00	1663.803	1	f1 = 0.00 f2 = 0.00 f3 = -13461.00	0.682
76	”	53	226	3	552	f1 = -180.00 f2 = -123.00 f3 = 16842.00	6.551	74	f1 = -144.00 f2 = -79.00 f3 = 13360.00	1.628
77	”	1143	1211	3	x	-	-	1	f1 = -85.00, f2 = 0, f3 = 0	16.248
78	”	36939	4608	3	x	-	-	1	f1 = 0, f2 = 0, f3 = 0	18927.102
79	”	900	60	4	+	-	-	1	f1 = -434.00, f2 = -452.00 f3 = -497.00, f4 = -463.00	3.005
80	”	729	729	4	+	-	-	x	-	-
81	”	4492	1003	4	x	-	-	*	-	-
82	”	900	60	10	x	-	-	1	f1 = -394.00, f2 = -429.00 f3 = -415.00, f4 = -428.00 f5 = -447.00, f6 =-417.00	4.306

Continued on next page

Table 5.3 – continued from previous page

Algorithms					BOA			PSA		
Prob.	**Origin**	**n**	**m**	**q**	**NNP**	**MPNP**	**CPU (s)**	**NNP**	**MPNP**	**CPU (s)**
									f7 = -401.00 , f8 = -414.00 f9 = -402.00, f10 = -429.00	
83	”	779	10174	10	+	-	-	1	f1 = 0.00, f2 = 0.00 f3 = 0.00, f4 = 0.00 f5 = 0.00, f6 = 0.00 f7 = 0.00, f8 = 0.00 f9 = 0.00, f10 = 0.00	424.58
84	”	376	1917	19	x	-	-	1	f1 = 2, f2 = -1, f3 = -1 f4 = -1 , f5 = 0, f6= -1, f7 = 0, f8 = -1, f9= 0, f10 = -1, f11 = -2, f12= -2 f13 = 0 , f14 =0, f15 = 0 f16 = 0, f17 = 0, f18 = 0, f19 = 0 .	69.765
85	”	218	28	27	+	-	-	1	f1 = -360, f2 = 0, f3 = 0 f4 = 90 , f5 =180, f6=180, f7 = 180, f8 = 180, f9=270, f10 = 0, f11 = 360, f12=90 f13 = 180 , f14 =0, f15=90 f16 = 90, f17 =0, f18 = -90, f19= 90 , f20 = -90, f21=-90, f22= 90 , f23 = -90, f24=-90, f25= -90 , f26=90, f27 = 0.	14.746
86	”	295056	24586	2	x		-	x	-	-

(*) The image is the whole region, implying that the problem has no solution

(+) Aborted after three days of running time

(x) Out of memory

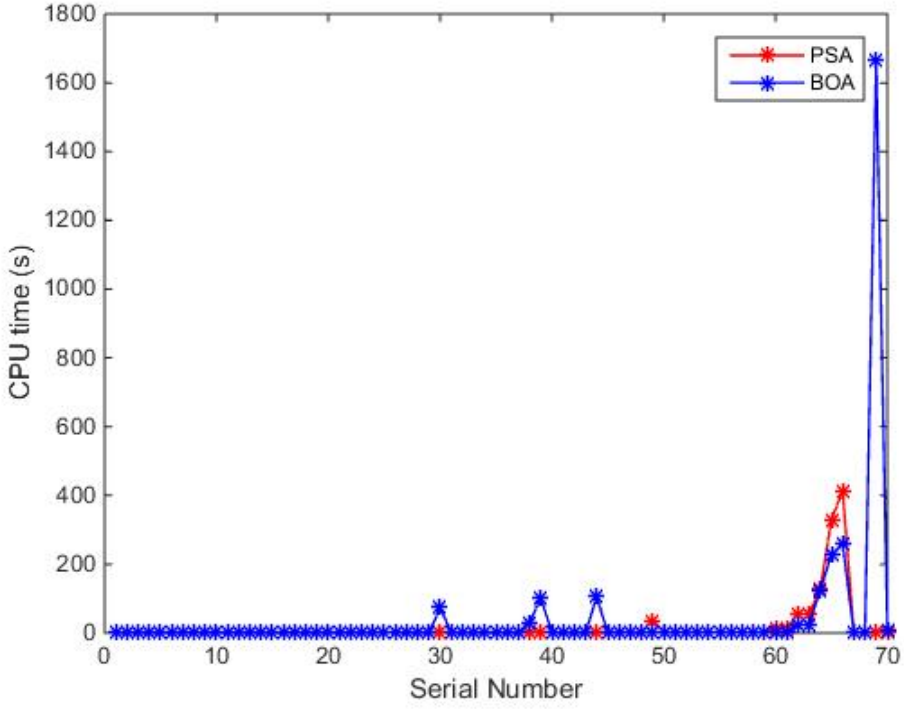

Figure 5.3: Running time of PSA and BOA for the 70 instances solved.

Chapter 6

A HEURISTIC APPROACH TO MULTI-OBJECTIVE LINEAR PROGRAMMING

6.1 Introduction

Multiple Objective Linear Programming (MOLP) problems are usually solved with exact methods as stated in previous chapters.

Given that MOLP is NP-hard [34, 90], it is surprising that only one approximate approach has been applied to it. Such approaches also called heuristics have been commonly used in nonlinear and discrete multi-objective optimisation (MOO). It is, therefore, worthwhile in-

vestigating representatives of this class of approaches and comparing them with exact methods.

Heuristic approaches are numerous and varied. Perhaps the most common such algorithms are those of the population-based evolutionary type such as the Genetic Algorithm, Ant Colony Algorithm and Tabu Search [22]. In the context of MOO, Nondominated Sorting Genetic Algorithm II or NSGA-II [49, 50] is arguably the most popular. There are new nature-inspired population based stochastic algorithms which have shown a lot of promise on nonlinear MOO problems. One such algorithm is the strawberry multi-objective plant propagation algorithm or MOPPA [67].

PPA in its multi-objective version has not been applied to MOLP, yet. We intend to do that here. MOPPA [67] is the implemented version. We shall apply MOPPA and NSGA-II [49, 50] to our collection of representative MOLP problems and then compare their outcomes with those of the exact methods. Before embarking into these experiments, we first briefly present MOPPA. We then apply MOPPA and NSGA-II to a set of well known test problems. The results will be compared with those obtained with the exact methods EMSA,

ASIMOLP, BOA and PSA.

6.2 Multi-objective Plant Propagation Algorithm

The Multi-objective Plant Propagation Algorithm (MOPPA) [67] is an extension of PPA [128] to multi-objective nonlinear optimisation problems. Apart from the extension, the algorithm was also equipped with a novel fitness function whose aim is to emphasize the end-points and drive the population based stochastic algorithm towards a good approximation to the nondominated front. In both PPA and MOPPA, each member of the population can generate a number of runners proportional to that member's fitness and to define a new point, a distance away proportional to 1 minus the fitness with all values randomly chosen, [67]. Both approaches implements exploitation or concentration by sending many short runners from good solutions and implements diversification or exploration by sending fewer but longer runners from those solutions that are not so good. The algorithm was successfully applied to the integrated energy systems design for off-grid mining operations which is a bicriteria dynamic optimisation problem. Most recently, the algorithm was also applied

to the industrial beer fermentation process in [123] for which the non-dominated front was successfully approximated. Given the successes of the algorithm recorded so far for single and multi-objective non-linear programming problems, we intend to apply MOPPA to MOLP problems. The pseudo-code of MOPPA adapted from [67] is given as Algorithm 9.

Algorithm 9 The Multi-Objective Plant Propagation Algorithm

```
Given: f(x), a vector function; n_g, number of generations to perform; n_p, the
    propagation size; n_r, maximum number of runners to propagate.
Output: z, vector approximation to Nondominated frontier.
    p ← initial random population of size n_p
    for n_g generations do
        prune population p, removing similar solutions
        N ← fitness(p)                          ▷ Use rank based fitness
        p̄ ← ∅                                   ▷ Empty set
        for i ← 0 ... n_p do
          x ← select(p, N)                      ▷ Tournament fitness based selection
          for each runner to generate do        ▷ Number proportional to fitness rounded up
              x̄ ← new solution(x, 1 − N)        ▷ Distance inversely proportional to fitness
              p̄ ← x̄ ∪ p̄                         ▷ Add to new population
          endfor
          p ← p\x                               ▷ Remove from old population
        endfor
        p ← p̄ ∪ Nondominated(p)                 ▷ New population with elitism
    endfor
    z ← Nondominated(p)
```

6.3 Solution Procedure

In order to apply MOPPA to MOLP, we use the penalty function method [110] to handle the constraints. The penalty function method is the most popular constraint handling technique in evolutionary

algorithms and many other optimisation frameworks [110, 155]. It penalizes each objective or fitness function by reducing its fitness values in proportion to the degree of constraint violation [139]. In other words, a penalty term is added to each of the objective function penalizing the function values that are not in the feasible region. To use this method, MOLP problem (1.1) is reformulated as follows:

$$\begin{aligned} \min \quad & f_1(x) + Kp_1(x) \\ & \vdots \\ & f_q(x) + Kp_q(x) \\ \text{subject to } & x \in X = \{[a,b]^n \subset \mathbb{R}^n : g_j(x) \leq 0, j = 1, ..., m\}, \end{aligned} \tag{6.1}$$

where X is the search space or feasible region which is described by box constraints, a and b are the lower and upper bounds on all variables, the scalar quantity K is a constant which is called the penalty parameter and the function $p_k(x), k = 1, ..., q$ is the penalty function. Equation (6.1) is now our new MOLP penalty program.

The penalty function $p_k(x)$ satisfies the following

- $p_k(x) = 0, \ if \ g_j(x) \leq 0$
- $p_k(x) > 0, \ if \ g_j(x) \nleq 0,$

that is to say, the penalty function is zero if no violation of the constraint occurs and is positive if a constraint is violated; the penalty parameter term would be added to the objective function such that the solution is pushed back towards the feasible region. A large penalty value prevents searching the infeasible region and enables the method to converge to a feasible solution quickly, [155].

6.3.1 Illustration of MOPPA

We implemented the MOLP penalty program 6.1 in Matlab and applied a Matlab implementation of MOPPA which can be found in [66] to solve Problem 3.1 of Section 3.2.1. With $x_1 \in [0, 7]$, $x_2 \in [0, 5]$ as variable bounds, a population size of 50, maximum number of runners 5 and the number of generations to perform at 200 are chosen. The nondominated front approximated by the algorithm is shown in Figure 6.1.

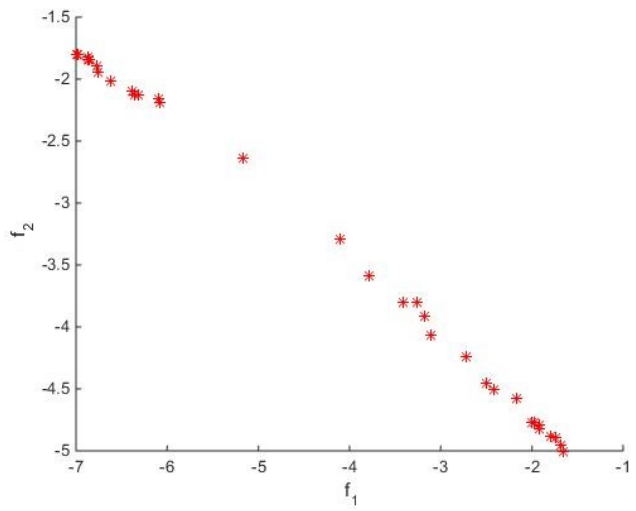

Figure 6.1: Nondominated frontier approximated by MOPPA.

The nondominated points and their corresponding fitness values are shown in Table 6.1. These are sorted in decreasing order from left to right of the table according to the rank based fitness function incorporated in the algorithm.

f1	-6.98	-1.64	-1.79	-7.00	-6.76	-2.50	-6.87	-6.32	-3.17	-6.61	-2.73	-3.26	-1.93	-1.97	-3.42	-1.68	-6.86	-1.93	-3.11	-6.08	-6.77	-3.78	-2.42	-4.10	-1.74	-2.00	-6.87	-6.36
f2	-1.80	-5.00	-4.89	-1.79	-1.94	-4.46	-1.83	-2.13	-3.92	-2.01	-4.24	-3.81	-4.79	-4.77	-3.80	-4.96	-1.85	-4.82	-4.07	-2.19	-1.90	-3.59	-4.51	-3.29	-4.90	-4.77	-1.84	-2.12
fitness	0.96	0.95	0.93	0.91	0.90	0.88	0.86	0.84	0.82	0.81	0.79	0.77	0.75	0.73	0.71	0.69	0.67	0.65	0.63	0.61	0.59	0.57	0.55	0.53	0.51	0.49	0.47	0.45

Table 6.1: Nondominated Points and their corresponding fitness values

We are interested in the nondominated point with the best fitness value which will serve as the Best Nondominated Point (BNP). From Table 6.1, the BNP is $f^1 = (-6.98, -1.80)^T$ with a fitness value of 0.96, where $f^1 = (f_1^1, f_2^1)^T \in Y_N$. Note that when solving Problem 3.1 of Section 3.2.1 with exact methods, the MPNP was found to be $f^1 = (-7.0,\ -1.80)^T$ in Section 4.4.

In Figure 6.1, it can be seen that MOPPA is able to approximate the nondominated frontier using the penalty function method. The nondominated points are not evenly distributed on the frontier but tend to concentrate more towards the end-points of the front.

We have also solved Problem 3.1 using NSGA-II [49, 50] which is another approximate multi-objective evolutionary algorithm with the same settings as in Equation 6.1. We use the same variable bounds with a population size of 50, 200 generations, mutation rate of 0.02 and crossover rate of 0.8. The nondominated frontier as approximated by NSGA-II is shown in Figure 6.2.

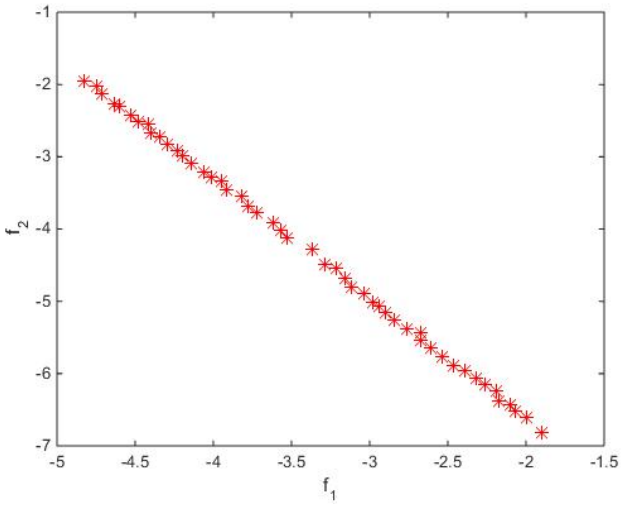

Figure 6.2: Nondominated frontier approximated by NSGA-II.

The nondominated points approximated by NSGA-II are in Table 6.2. Each of these points is assigned a rank or fitness value according to its domination level and sorted in descending order from left to right of the table according to their crowding distances. The end-points which are fitter than other points are assigned an infinite distance value.

f1	-6.82	-1.96	-4.28	-4.12	-6.60	-4.49	-3.78	-3.91	-5.77	-5.64	-5.26	-5.39	-3.55	-4.88	-5.88	-3.10	-3.45	-5.97	-2.13	-4.55
f2	-1.90	-4.82	-3.36	-3.53	-2.00	-3.29	-3.72	-3.62	-2.54	-2.61	-2.84	-2.76	-3.82	-3.03	-2.47	-4.14	-3.92	-2.39	-4.72	-3.21
Crowding Distance	Inf	Inf	0.1582	0.1232	0.1171	0.1066	0.1032	0.1014	0.0974	0.0953	0.0943	0.0933	0.0925	0.0917	0.0916	0.0912	0.0905	0.0894	0.0863	0.0842

Table 6.2: Nondominated Points and their corresponding crowding distances

From Table 6.2, the end-point $f^1 = (-6.82, -1.90)^T$ which is the first to be listed and therefore highest in ranking of the two end-points, is selected as the BNP.

In Figure 6.2, it can be seen that NSGA-II did not only approximate the nondominated frontier, but also distribute the points evenly on the nondominated front for the problem.

In terms of the quality of nondominated points approximated by these two algorithms, it can be seen in Tables 6.1 and 6.2 that the points returned by MOPPA are of higher quality than those returned by NSGA-II. This can easily be seen that the BNP $f^1 = (-6.98, -1.80)^T$ selected from Table 6.1 is of higher quality and closer to the MPNP $f^1 = (-7.0,\ -1.80)^T$ found by the exact methods of Section 4.4 than the BNP $f^1 = (-6.82, -1.90)^T$ selected from Table 6.2.

We compare these two approximate methods with the four exact methods (EMSA, ASIMOLP, BOA and PSA) whose results are presented in Tables 4.3, 5.2 and 5.3 respectively.

6.4 Discussion of Experimental Results

In this section, we provide numerical results to compare the quality of the BNP returned by MOPPA and NSGA-II which find approximate points to the MPNP computed by exact methods as shown in Tables 4.3, 5.2 and 5.3. Since heuristic approaches are best tested on problems for which the solutions are known [50]. Table 6.4 shows the numerical results for a collection of 51 existing problems from the literature ranging from small to medium and realistic instances. The structure of these problems have been described in previous chapters.

All algorithms were implemented in Matlab and executed on an Intel Core i5-2500 CPU at 3.30GHz with 16.0GB RAM. In all tests, m is the number of constraints, n the number of variables and q the number of objectives. We used the MPNPs computed from the nondominated set returned by EMSA, PSA, BOA and ASIMOLP as explained in Section 4.4 to compare with the BNP returned by MOPPA and NSGA-II.

As can be seen from Table 6.4, MOPPA compared favourably in terms of the quality of the BNP it returns. The algorithm returns BNPs

which are of higher quality and closer to those returned by exact methods than NSGA-II, which confirms what was reported in [67]. Of particular interest is Problems 3, 9, 11 and 23 where the points returned are exactly the same with those of the exact methods. In terms of diversity (spread) between the two approximated methods, as can be seen from Section 6.4.1, NSGA-II returns nondominated points that are more uniformly or evenly distributed than the non-dominated front of MOPPA whose points tend to concentrate more towards the end-points of the front.

We also observed in Table 6.4 that some of the exact methods could not produce results for some of the numerically ill-posed and highly challenging test problems considered due to one reason or the other as stated in the table. The approximate methods on the other hand, were able to solve all these difficult MOLP instances approximately.

6.5 Summary of Results

In this section, we present the summary of experimental results discussed in the previous section in Table 6.3.

Algorithms	Criteria for Evaluation	
	Diversity (Spread)	**Quality of BNP returned**
MOPPA	The nondominated points are not evenly distributed on the front but concentrate more towards the end-points	Return high quality BNP than NSGA-II which is closer to those returned by the exact methods
NSGA-II	The nondominated points are uniformly distributed on the nondominated front	The quality of BNP is not as good as that returned by MOPPA

Table 6.3: Summary of experimental results

6.6 Summary

In this chapter, we have applied two heuristic approaches to MOLP. One, namely NSGA-II is well established and popular heuristic for continuous and discrete MOO. The other, MOPPA, is fairly recent addition to NIA's which has shown a lot of promise on continuous MOO, and continuous and discrete single objective optimisation. Our experimental investigation using Matlab implementations of both approaches applied to an extensive and representation set of MOLP instances has shown that the methods found on the whole good nondominated fronts. That of NSGA-II is more uniformly spread while

the BNP's returned by MOPPA tend to be of better quality. The methods compare well with the exact ones especially on the large instances which the exact methods failed to solve even when given generous amounts of computation times. Constraints have been handled using a penalty approach.

Table 6.4: Comparative results for individual problem

Algorithm					EMSA	ASIMOLP	BOA	PSA	NSGA-II	MOPPA
Prob.	**Origin**	**n**	**m**	**q**	**MPNP**	**MPNP**	**MPNP**	**MPNP**	**BNP**	**BNP**
1	Ehrgott 2006	3	3	3	f1 = -2.00 f2 = 10.00 f3 = -5.00	f1 = -1.74 f2 = 5.56 f3 = -2.75	f1 = -2.00 f2 = 10.00 f3 = -5.00	f1 = -2.00 f2 = 10.00 f3 = -5.00	f1 = -0.35 f2 = 3.27 f3 = -1.98	f1 = -0.70 f2 = 7.12 f3 = -3.56
2	Zeleny 1982	2	2	2	f1 = -25000 f2 = -66000	f1 = -30626 f2 = -64132	f1 = -25000 f2 = -66000	f1 = -25000 f2 = -66667	f1 = -22302 f2 = -34750	f1 = -24880 f2 = -37320
3	”	2	4	2	f1 = -9.00 f2 = -15.00	f1 = 4.00 f2 = -18.42	f1 = -9.00 f2 = -15.00	f1 = -9.00 f2 = -15.00	f1 = -9.18 f2 = -11.21	f1 = -9.00 f2 = -15.00
4	”	2	4	3	f1 = -3.00 f2 = -7.50 f3 = -9.00	f1 = -3.50 f2 = -2.74 f3 = 4.89	f1 = -3.00 f2 = -7.50 f3 = -9.00	f1 = -3.00 f2 = -7.50 f3 = -9.00	f1 = -3.10 f2 = -2.21 f3 = -4.13	f1 = -3.10 f2 = -2.50 f3 = -4.40
5	”	2	6	2	f1 = -24.00 f2 = -16.00	f1 = -21.29 f2 = -17.29	f1 = -24.00 f2 = -16.00	f1 = -24.00 f2 = -16.00	f1 = -14.61 f2 = -9.32	f1 = -21.00 f2 = -17.00
6	”	3	3	3	f1 = 3.00 f2 = -6.00 f3 = -12.00	f1 = 1.33 f2 = -6.20 f3 = -9.68	f1 = 3.00 f2 = -6.00 f3 = -12.00	f1 = 3.00 f2 = -6.00 f3 = -12.00	f1 = -0.17 f2 = -3.11 f3 = -3.97	f1 =0.00 f2 = -6.00 f3 = -3.00
7	”	5	3	3	f1 = 0.00 f2 = -4.00 f3 = -24.00	f1 = -1.38 f2 = -8.77 f3 = -10.04	f1 = 0.00 f2 = -4.00 f3 = -24.00	f1 = 0.00 f2 = -4.00 f3 = -23.62	f1 =-1.40 f2 = -3.00 f3 = -5.08	f1 = -0.10 f2 = -4.00 f3 = -9.00
8	”	5	2	2	f1 = -52.00 f2 = -52.00	f1 = -4.11 f2 = -29.30	f1 = -52.00 f2 = -52.00	f1 = -52.00 f2 = -52.00	f1 = -17.15 f2 = -29.72	f1 = -51.00 f2 = 54.00
9	”	6	4	2	f1 = 0.00 f2 = 0.00	f1 = -0.02 f2 = 0.00	f1 = 0.00 f2 = 0.00	f1 = 0.00 f2 = 0.00	f1 = -0.03 f2 = -0.03	f1 = 0.00 f2 = 0.00
10	”	7	4	3	f1 = -48.00 f2 = -32.00 f3 = 16.00	f1 = -7.65 f2 = -13.80 f3 = -7.75	f1 = -48.00 f2 = -32.00 f3 = 16.00	f1 = -16.00 f2 = 0.00 f3 = -16.00	f1 = -10.85 f2 = -11.66 f3 = -2.52	f1 = -13.90 f2 = -10.76 f3 = 2.65
11	iMOLPe	2	3	2	f1 = -21.00 f2 = -7.00	f1 = -11.87 f2 = -17.22	f1 = -21.00 f2 = -7.00	f1 = -21.00 f2 = -7.00	f1 = -20.02 f2 = -8.44	f1 = -21.00 f2 = -7.00
12	”	3	3	4	f1 = -10.00 f2 = -20.00 f3 = -100.00	f1 = -5.59 f2 = -18.62 f3 = -34.83	f1 = -10.00 f2 = -20.00 f3 = -100.00	f1 = -10.00 f2 = -20.00 f3 = -100.00	f1 = -8.86 f2 = -17.15 f3 = -20.95	f1 = -12.00 f2 = -19.00 f3 = -35.00

Continued on next page

Table 6.4 – continued from previous page

Algorithm					EMSA	ASIMOLP	BOA	PSA	NSGA-II	MOPPA
Prob.	**Origin**	**n**	**m**	**q**	**MPNP**	**MPNP**	**MPNP**	**MPNP**	**BNP**	**BNP**
					f4 = -10.00	f4 = -42.23	f4 = -10.00	f4 = -10.00	f4 = -12.55	f4 = -12.00
13	”	3	5	3	f1 = -21.00 f2 = -4.50 f3 = -4.00	f1 = -10.48 f2 = -3.62 f3 = -2.14	f1 = -21.00 f2 = -4.50 f3 = -4.00	f1 = -21.00 f2 = -4.50 f3 = -4.00	f1 = -14.40 f2 = -4.57 f3 = -3.83	f1 = -16.50 f2 = -4.50 f3 = -5.00
14	”	3	3	3	f1 = -2.66 f2 = -2.00 f3 = -0.33	f1 = -1.10 f2 = -1.22 f3 = -1.57	f1 = -2.66 f2 = -2.00 f3 = -0.33	f1 = -2.66 f2 = -2.00 f3 = -0.33	f1 = -2.22 f2 = -1.07 f3 = -0.40	f1 = -2.00 f2 = -1.50 f3 = -1.50
15	”	4	3	3	f1 = -48.50 f2 = -19.50 f3 = -37.00	f1 = -35.80 f2 = -43.97 f3 = -29.82	f1 = -48.50 f2 = -19.50 f3 = -37.00	f1 = -48.50 f2 = -19.50 f3 = -37.00	f1 = -30.37 f2 = -21.88 f3 = -35.70	f1 = -35.00 f2 = -30.00 f3 = -35.00
16	”	4	2	3	f1 = -20.00 f2 = -80.00 f3 = -40.00	f1 = -31.71 f2 = -49.12 f3 = -69.69	f1 = -20.00 f2 = -80.00 f3 = -40.00	f1 = -20.00 f2 = -80.00 f3 = -40.00	f1 = -20.48 f2 = -24.56 f3 = -27.11	f1 = -35.00 f2 = -30.00 f3 = -35.00
17	”	4	4	3	f1 = -40.00 f2 = -50.00 f3 = -10.00	f1 = -32.22 f2 = -32.50 f3 = -36.27	f1 = -40.00 f2 = -50.00 f3 = -10.00	f1 = -40.00 f2 = -50.00 f3 = -10.00	f1 = -21.90 f2 = -20.58 f3 = -28.34	f1 = -35.00 f2 = -30.00 f3 = -35.00
18	”	3	3	3	f1 = 0.00 f2 = -2.00 f3 = -4.00	f1 = -1.12 f2 = -2.14 f3 = -2.63	f1 = 0.00 f2 = -2.00 f3 = -4.00	f1 = 0.00 f2 = -2.00 f3 = -4.00	f1 = -1.93 f2 = -1.22 f3 = -1.87	f1 = -2.00 f2 = -1.92 f3 = -2.00
19	”	15	10	2	f1 = -363.82 f2 =-33.70	f1 = -137.09 f2 =-198.96	f1 = -363.82 f2 =-33.70	f1 = -229.18 f2 = -35.31	f1 = -73.53 f2 = -31.14	f1 = -143.05 f2 = -32.11
20	”	15	10	3	f1 = -363.82 f2 = -33.70 f3 = -136.71	f1 = -107.15 f2 = -169.94 f3 = -166.26	f1 = -343.50 f2 = -42.43 f3 = -158.75	f1 = -134.17 f2 = -32.88 f3 = -135.82	f1 = -74.85 f2 = -36.96 f3 = -55.35	f1 = -133.77 f2 = -45.05 f3 = -76.54
21	”	10	5	3	f1 = 226.40 f2 = -501.86 f3 = -351.14	f1 = 59.42 f2 = -357.21 f3 = -356.48	f1 = 226.40 f2 = -501.86 f3 = -351.14	f1 = 223.09 f2 = -496.23 f3 = -246.64	f1 = -111.85 f2 = -47.52 f3 = -58.96	f1 =-164.65 f2 = -47.33 f3 = -58.37
22	Steuer 1986	5	5	2	f1 = -10.00 f2 = -3.00	f1 = -6.30 f2 = -6.90	f1 = -10.00 f2 = -3.00	f1 = -10.00 f2 = -3.00	f1 = -6.36 f2 = -3.33	f1 = -6.50 f2 = -3.50
23	”	4	4	3	f1 = 3.42 f2 = -10.28	f1 = -3.79 f2 = 11.38	f1 = 3.42 f2 = -10.28	f1 = 3.42 f2 = -10.28	f1 = -3.09 f2 = 9.25	f1 = -3.50 f2 = 10.50

Continued on next page

Table 6.4 – continued from previous page

Algorithm					EMSA	ASIMOLP	BOA	PSA	NSGA-II	MOPPA
Prob.	**Origin**	**n**	**m**	**q**	**MPNP**	**MPNP**	**MPNP**	**MPNP**	**BNP**	**BNP**
					f3 = -3.42	f3 = -2.96	f3 = -3.42	f3 = -3.42	f3 = -2.74	f3 = -3.40
24	”	5	5	4	f1 = 1.02 f2 = -25.46 f3 = 24.44 f4 = -28.32	f1 = 2.28 f2 = -22.58 f3 = 20.30 f4 = -25.47	f1 = 1.02 f2 = -25.46 f3 = 24.44 f4 = -28.32	f1 = 1.02 f2 = -25.46 f3 = 24.44 f4 = -28.32	f1 = 1.98 f2 = -2.73 f3 = 10.75 f4 = -5.32	f1 = 1.20 f2 = -3.81 f3 = 2.61 f4 = -4.55
25	”	10	8	4	f1 = 106.29 f2 = -462.13 f3 = 175.57 f4 = -33.41	f1 = 80.00 f2 = -54.36 f3 = -163.73 f4 = -23.82	f1 = 106.29 f2 = -462.13 f3 = 175.57 f4 = -33.41	f1 = 183.36 f2 = -424.26 f3 = 117.29 f4 = -4.03	f1 = 13.22 f2 = -36.00 f3 = 6.07 f4 = -1.80	f1 = -54.50 f2 = -20.64 f3 = 46.59 f4 = 20.97
26	”	5	4	3	f1 = -52.07 f2 = 31.50 f3 = -17.35	f1 = -4.44 f2 = -13.17 f3 = -14.37	f1 = -52.07 f2 = 31.50 f3 = -17.35	f1 = -52.07 f2 = 31.50 f3 = -17.35	f1 = -21.00 f2 = 4.69 f3 = -13.87	f1 = -21.00 f2 = 8.79 f3 = -14.79
27	”	6	8	4	f1 = -6.94 f2 = -5.38 f3 = 6.83 f4 = -9.16	f1 = -6.69 f2 = -2.25 f3 = 6.77 f4 = -8.83	f1 = -6.94 f2 = -5.38 f3 = 6.83 f4 = -9.16	f1 = -6.94 f2 = -5.38 f3 = 6.83 f4 = -9.16	f1 = -2.90 f2 = -2.29 f3 = 1.12 f4 = -3.94	f1 = -2.50 f2 = -4.27 f3 = 5.74 f4 = -4.41
28	”	7	6	4	f1 = -31.53 f2 = -26.48 f3 = -26.57 f4 = -0.34	f1 = -25.90 f2 = -23.94 f3 = -19.06 f4 = -8.62	f1 = -31.53 f2 = -26.48 f3 = -26.57 f4 = -0.34	f1 = -31.29 f2 = -30.08 f3 = -26.33 f4 = -0.82	f1 = -17.42 f2 = -12.29 f3 = -9.58 f4 = -7.61	f1 = -20.02 f2 = -16.41 f3 = -14.00 f4 = -4.41
29	”	7	6	4	f1 = 26.80 f2 = -37.73 f3 = -24.33 f4 = -59.60	f1 = 4.03 f2 = -29.03 f3 = -18.07 f4 = -28.17	f1 = 26.80 f2 = -37.73 f3 = -24.33 f4 = -59.60	f1 = 26.80 f2 = -37.73 f3 = -24.33 f4 = -59.60	f1 = -4.05 f2 = -6.97 f3 = -4.92 f4 = -6.13	f1 = -4.79 f2 = -7.34 f3 = -6.11 f4 = -10.55
30	”	8	8	6	f1 = -74.00 f2 = -107.50 f3 = -41.25 f4 = -27.25 f5 = -9.00 f6 = -30.75	f1 = -15.46 f2 = -38.73 f3 = -43.30 f4 = -30.95 f5 = -8.30 f6 = -26.72	f1 = -74.00 f2 = -107.50 f3 = -41.25 f4 = -27.25 f5 = -9.00 f6 = -30.75	f1 = -77.00 f2 = -52.00 f3 = -16.00 f4 = -52.40 f5 = 26.00 f6 = -20.00	f1 = -12.11 f2 = -15.40 f3 = -3.90 f4 = -4.79 f5 = 4.37 f6 = -4.10	f1 = -14.41 f2 = -22.72 f3 = -10.93 f4 = -13.79 f5 = -5.36 f6 = -19.30
31	”	8	8	3	f1 = -36.57 f2 = -22.28	f1 = -32.03 f2 = -20.03	f1 = -36.57 f2 = -22.28	f1 = -36.00 f2 = -23.00	f1 = -9.53 f2 = -3.97	f1 = -10.43 f2 = -4.02

Continued on next page

Table 6.4 – continued from previous page

Algorithm					EMSA	ASIMOLP	BOA	PSA	NSGA-II	MOPPA
Prob.	**Origin**	**n**	**m**	**q**	**MPNP**	**MPNP**	**MPNP**	**MPNP**	**BNP**	**BNP**
					f3 = -14.00	f3 = -17.73	f3 = -14.00	f3 = -15.00	f3 = -4.32	f3 = -3.05
32	”	8	8	3	f1 = -14.03 f2 = -18.00 f3 = -4.93	f1 = -8.77 f2 = -10.56 f3 = -5.13	f1 = -14.03 f2 = -18.00 f3 = -4.93	f1 = -6.50 f2 = -11.00 f3 = -7.50	f1 = -3.85 f2 = -3.32 f3 = -2.79	f1 = -6.30 f2 = -3.82 f3 = -3.06
33	”	5	5	4	f1 = -21.50 f2 = -39.25 f3 = -16.25 f4 = 27.00	f1 = -20.83 f2 = -21.78 f3 = -16.05 f4 = 14.45	f1 = -21.50 f2 = -39.25 f3 = -16.25 f4 = 27.00	f1 = -8.00 f2 = -23.87 f3 = -7.62 f4 = 27.00	f1 = -3.44 f2 = -15.76 f3 = -4.97 f4 = 5.97	f1 = -17.00 f2 = -13.00 f3 = -5.00 f4 = 8.00
34	”	6	6	3	f1 = -12.65 f2 = 0.00 f3 = -30.15	f1 = 12.69 f2 = -3.21 f3 = -28.39	f1 = -12.65 f2 = 0.00 f3 = -30.15	f1 = 13.62 f2 = -9.75 f3 = -26.25	f1 = -4.81 f2 = 2.06 f3 = 3.25	f1 = -6.00 f2 = 4.00 f3 = 2.00
35	”	5	5	4	f1 = -14.66 f2 = -21.06 f3 = 35.73 f4 = -16.00	f1 = -6.33 f2 = -14.44 f3 = 20.77 f4 = -15.63	f1 = -14.66 f2 = -21.06 f3 = 35.73 f4 = -16.00	f1 = -14.00 f2 = 0.00 f3 = 27.00 f4 = 0.00	f1 = -6.64 f2 = -11.42 f3 = 18.07 f4 = -8.71	f1 = -9.00 f2 = -11.21 f3 = 20.21 f4 = -9.00
36	”	10	10	4	f1 = 46.50 f2 = 19.21 f3 = -27.07 f4 = -27.07	f1 = 50.69 f2 = 18.98 f3 = -23.38 f4 = -23.85	f1 = 46.50 f2 = 19.21 f3 = -27.07 f4 = -27.07	f1 = 46.50 f2 = 19.21 f3 = -27.07 f4 = -27.07	f1 = -1.46 f2 = -2.51 f3 = -4.09 f4 = -3.50	f1 = 12.98 f2 = 8.84 f3 = -4.04 f4 = -8.09
37	”	8	8	3	f1 = -14.48 f2 = -4.74 f3 = 6.93	f1 = -2.46 f2 = -3.22 f3 = -1.93	f1 = -14.48 f2 = -4.74 f3 = 6.93	f1 = -14.48 f2 = -4.74 f3 = 6.93	f1 = -1.13 f2 = -2.12 f3 = 0.09	f1 = -2.06 f2 = -4.68 f3 = 4.02
38	”	6	7	4	f1 = -2.61 f2 = -12.63 f3 = 9.70 f4 = 2.37	f1 = -1.80 f2 = -4.00 f3 = 2.78 f4 = -2.07	f1 = -2.61 f2 = -12.63 f3 = 9.70 f4 = 2.37	+	f1 = -2.07 f2 = -0.27 f3 = 0.45 f4 = -0.90	f1 = -2.60 f2 = -2.44 f3 = 1.66 f4 = -1.04
39	”	12	16	4	*	f1 = -5.09 f2 = -9.83 f3 = -9.53 f4 = -6.18	f1 = -5.25 f2 = -14.25 f3 = -8.25 f4 = -1.00	f1 = -5.13 f2 = -3.38 f3 = 1.83 f4 = -1.18	f1 = -2.99 f2 = -2.02 f3 = -2.57 f4 = -1.98	f1 = -5.75 f2 = -2.19 f3 = -3.05 f4 = -1.90
40	”	10	14	5	*	f1 = -1.07 f2 = -3.83	f1 = -5.16 f2 = -2.79	f1 = -18.00 f2 = 70.60	f1 = -2.20 f2 = -3.53	f1 = -4.15 f2 = -2.86

Continued on next page

Table 6.4 – continued from previous page

Algorithm					EMSA	ASIMOLP	BOA	PSA	NSGA-II	MOPPA
Prob.	**Origin**	**n**	**m**	**q**	**MPNP**	**MPNP**	**MPNP**	**MPNP**	**BNP**	**BNP**
						f3 = -5.53 f4 = -16.87 f5 = -8.42	f3 = -4.38 f4 = -18.70 f5 = -9.69	f3 = -3.20 f4 = 8.00 f5 = -4.30	f3 = -1.72 f4 = -5.09 f5 = -1.50	f3 = -3.14 f4 = -5.36 f5 = -1.73
41	”	7	6	3	f1 = -29.40 f2 = -65.30 f3 = -39.30	f1 = -10.74 f2 = -32.20 f3 = -24.39	f1 = -29.40 f2 = -65.30 f3 = -39.30	f1 = -29.40 f2 = -65.30 f3 = -39.30	f1 = -26.09 f2 = -51.35 f3 = -31.97	f1 = -30.00 f2 = -52.00 f3 = -24.00
42	”	7	7	3	f1 = -62.18 f2 = -93.50 f3 = -52.00	f1 = -47.39 f2 = -86.99 f3 = -54.78	f1 = -62.18 f2 = -93.50 f3 = -52.00	f1 = -62.18 f2 = -93.50 f3 = -52.00	f1 = -36.80 f2 = -40.01 f3 = -14.05	f1 = -34.50 f2 = -50.50 f3 = -22.00
43	”	6	6	4	f1 = -37.50 f2 = -11.25 f3 = -7.50 f4 = -20.25	f1 = -10.40 f2 = -6.52 f3 = -5.91 f4 = -0.34	f1 = -37.50 f2 = -11.25 f3 = -7.50 f4 = -20.25	f1 = -37.50 f2 = -11.25 f3 = -7.50 f4 = -20.25	f1 = -11.04 f2 = -2.47 f3 = -6.11 f4 = 0.13	f1 = -17.58 f2 = -3.04 f3 = -0.24 f4 = -10.86
44	”	6	6	4	f1 = 34.50 f2 = -7.50 f3 = -56.00 f4 = -31.50	f1 = 28.39 f2 = -6.38 f3 = -45.43 f4 = -25.83	f1 = 34.50 f2 = -7.50 f3 = -56.00 f4 = -31.50	f1 = 34.50 f2 = -7.50 f3 = -56.00 f4 = -31.50	f1 = 6.02 f2 = -6.95 f3 = -9.14 f4 = -3.30	f1 = 10.00 f2 = -7.00 f3 = -13.00 f4 = -6.50
45	”	10	14	5	*	f1 = 3.35 f2 = -3.18 f3 = -2.48 f4 = -2.50 f5 = 2.10	f1 = 1.03 f2 = -2.19 f3 = 2.01 f4 = -8.13 f5 = 7.22	+	f1 = 2.62 f2 = -2.73 f3 = 0.27 f4 = -2.18 f5 = 2.69	f1 = 3.03 f2 = -2.89 f3 = -0.16 f4 = -2.64 f5 = 2.55
46	”	10	14	5	*	f1 = 2.35 f2 = 0.73 f3 = -11.72 f4 = -1.90 f5 = -10.33	f1 = -4.9 f2 = -3.42 f3 = -4.38 f4 = -18.91 f5 = -9.27	f1 = -14.93 f2 = -5.57 f3 = -2.83 f4 = -16.28 f5 = -6.13	f1 = 1.73 f2 = -3.43 f3 = -1.32 f4 = 3.35 f5 = -1.26	f1 = 2.04 f2 = -4.01 f3 = -1.86 f4 = 4.04 f5 = -1.34
47	”	7	7	3	f1 = -3.83 f2 = -76.46 f3 = -49.57	f1 = -6.82 f2 = -68.93 f3 = -25.83	f1 = -3.83 f2 = -76.46 f3 = -49.57	f1 = -3.83 f2 = -76.46 f3 = -49.57	f1 = -1.83 f2 = -29.69 f3 = -19.70	f1 = -2.00 f2 = -30.00 f3 = -20.00
48	Bensolve 2.0	5	31	5	*	f1 = 0.00 f2 = 0.00	f1 = 0.00 f2 = -1.00	f1 = 0.00 f2 = 0.00	f1 = -0.17 f2 = -0.17	f1 = -0.20 f2 = -0.20

Continued on next page

Table 6.4 – continued from previous page

Algorithm					EMSA	ASIMOLP	BOA	PSA	NSGA-II	MOPPA
Prob.	**Origin**	**n**	**m**	**q**	**MPNP**	**MPNP**	**MPNP**	**MPNP**	**BNP**	**BNP**
						f3 = 0.00 f4 = 0.00 f5 = 0.01	f3 = 0.00 f4 = 0.00 f5 = -2.00	f3 = 0.00 f4 = 0.00 f5 = 0.00	f3 = -0.17 f4 = -0.17 f5 = -0.17	f3 = -0.20 f4 = -0.20 f5 = -0.20
49	MOPLIB	100	20	3	f1 = -168.00 f2 = -124.00 f3 = -143.00	f1 = -61.18 f2 = -73.93 f3 = -96.38	f1 = -168.00 f2 = -124.00 f3 = -143.00	f1 = -168.00 f2 = -124.00 f3 = -143.00	f1 = -100.12 f2 = -99.09 f3 = -109.69	f1 = -121.82 f2 = -120.09 f3 = -126.84
50	”	30	21	12	f1=5.0E-12,f2=5.0E-12 f3=5.0E-12,f4=5.0E-12 f5=5.0E-12,f6=5.0E-12 f7=5.0E-12,f8=5.0E-12 f9=5.0E-12,f10=5.0E-12 f11=5.0E-12,f12=-5.5E-11	-	f1=5.0E-12,f2=5.0E-12 f3=5.0E-12,f4=5.0E-12 f5=5.0E-12,f6=5.0E-12 f7=5.0E-12,f8=5.0E-12 f9=5.0E-12,f10=5.0E-12 f11=5.0E-12,f12=-5.5E-11	f1 = 0, f2 = 0 f3 = 0, f4 = 0 f5 = 0, f6 = 0 f7 = 0 , f8 = 0 f9 = 0, f10 = 0 f11 = 0, f12 = 0	f1=8.0E+15,f2=8.0E+15 f3=8.0E+15,f4=8.0E+15 f5=8.0E+15,f6=8.0E+15 f7=8.0E+15,f8=8.0E+15 f9=8.0E+15,f10=8.0E+15 f11=8.0E+15,f12=8.0E+15	f1=8.0E+15,f2=8.0E+15 f3=8.0E+15,f4=8.0E+15 f5=8.0E+15,f6=8.0E+15 f7=8.0E+15,f8=8.0E+15 f9=8.0E+15,f10=8.0E+15 f11=8.0E+15,f12=8.0E+15
51	”	218	28	27	x	f1=0.52,f2=0.01, f3=6.95,f4=2.14 f5=4.94,f6=-5.53, f7=-9.71, f8=2.23, f9=-0.31,f10=2.67 f11=-3.19,f12=-5.55 f13=-5.42 , f14=-4.46, f15=-4.35,f16=6.01 f17=-3.36,f18=1.71, f19=-8.01 ,f20=8.90, f21=8.01,f22=-5.35 f23=5.35,f24=5.35, f25=5.35, f26=-5.37, f27=-4.35	x	f1=-360.00,f2=0.00, f3=0.00,f4=90.00 f5=180.00,f6=180.00, f7=180.00,f8=180.00, f9=270.00,f10=0.00, f11=360.00,f12=90.00 f13=180.00,f14 =0.00, f15=90.00,f16=90.00 f17=0.00,f18=-90.00, f19=90.00,f20=-90.00, f21=-90.00,f22=90.00 f23=-90.00,f24=-90.00, f25=-90.00,f26=90.00, f27=0.00	f1=2.0E+15, f2=2.0E+15, f3=2.0E+15,f4=2.0E+15, f5=2.0E+15,f6=2.0E+15 f7=2.0E+15,f8=2.0E+15, f9=2.0E+15,f10=2.0E+15, f11=2.0E+15,f12=2.0E+15 f13=2.0E+15,f14=2.0E+15, f15=2.0E+15,f16=2.0E+15, f17=2.0E+15,f18=2.0E+15, f19=2.0E+15,f20=2.0E+15 f21=2.0E+15,f22=2.0E+15, f23=2.0E+15,f24=2.0E+15 f25=2.0E+15,f26=2.0E+15, f27=2.0E+15	f1=2.0E+15, f2=2.0E+15, f3=2.0E+15,f4=2.0E+15, f5=2.0E+15,f6=2.0E+15 f7=2.0E+15,f8=2.0E+15, f9=2.0E+15,f10=2.0E+15, f11=2.0E+15,f12=2.0E+15 f13=2.0E+15,f14=2.0E+15, f15=2.0E+15,f16=2.0E+15, f17=2.0E+15,f18=2.0E+15, f19=2.0E+15,f20=2.0E+15 f21=2.0E+15,f22=2.0E+15, f23=2.0E+15,f24=2.0E+15 f25=2.0E+15,f26=2.0E+15, f27=2.0E+15

(x) Out of memory

(-) No initial starting solution

(*) Aborted after 3 days of running time

(+) The image is the whole region, implying that none of the vertices is nondominated

Chapter 7

CONCLUSION AND FUTURE RESEARCH PLAN

7.1 Introduction

This thesis is concerned with the state-of-the-art technology for MOLP. It considers prominent exact algorithms, understanding them, implementing and modifying some of them, as well as comparing them comprehensively on a series of existing test problems. Above all, the thesis also extends the MSA of Evans and Steuer [63] to generate the set of all nondominated points devoid of redundant ones, presents a procedure for computing the MPNP for the problem and applies nature-inspired population-based MOPPA [67] and NSGA-II [49, 50]

to MOLP. This chapter concludes the thesis and provides it findings. The chapter also provides further research directions.

7.2 Contributions

In Chapter 2, we carried out a comprehensive review of MOLP literature spanning over five decades and presented a state-of-the-art survey of all relevant algorithms. The survey classifies MOLP algorithms into two broad classes: Non-Interactive and Interactive algorithms. The Non-Interactive methods include the simplex, interior-point and objective space based algorithms. The Interactive ones include only the simplex and interior-point based algorithms. A tabulated list of all algorithms is included in the appendices section. It was observed that between 1964 and the early 90s, MOLP problems were solved mostly by the simplex based algorithms and it variants. Interior-point based methods for MOLP appeared in the literature from the early 90s, following the appearance of the Ellipsoid algorithm of Khachiyan [91] and Karmarkar's algorithm [89]. The objective space methods appeared first in the mid 90s. They are gaining

in notoriety.

Chapter 3 extends the MSA of Evans and Steuer [63] to generate the set of all nondominated points of the problem devoid of redundant ones. We also compared the extended version with the original version and with the primal variant of BOA [30] which is an objective space based method that also computes the set of all nondominated points of the problem. It was observed in Table 3.2 of Section 3.6 that there is a slight difference in computational time between EMSA, introduced here and the original version. This was expected as a consequence of the extension, more computational efforts would be required to generate the corresponding nondominated points, after which they are sorted in each case. It was also observed that the total number of nondominated points returned by EMSA is the same with that returned by BOA in most of the problems considered.

Chapter 4 further investigates EMSA, BOA [30] and Arbel's ASIMOLP [6]. In this chapter, we proposed a procedure to compute the Most Preferred Nondominated Point (MPNP) and compared the computing efficiency as well as the quality of a MPNP returned by these three algorithms. In terms of the quality of a MPNP, it was ob-

served in Table 4.3 of Section 4.5 that BOA and EMSA are superior to ASIMOLP in this regard while the latter outperforms BOA and EMSA in terms of computing efficiency. However, on the test problems considered, BOA was also found to be computationally superior to EMSA.

Chapter 5 compares BOA [30] comprehensively with the recently introduced PSA [124]. In this chapter, we compared the computing efficiency, robustness and the quality of MPNP's returned by these two algorithms. It was seen in Tables 5.2 and 5.3 that BOA is superior to PSA in terms of robustness, the quality of MPNP it returns and is also computationally more efficient than PSA on highly degenerate problems. However, PSA outperforms BOA computationally on non-degenerate problems.

Finally, Chapter 6 applies nature-inspired population-based stochastic algorithms MOPPA [67] and NSGA-II [49, 50] to MOLP. In this chapter, we compared the quality of Best Nondominated Points (BNPs) returned by MOPPA and NSGA-II with those returned by the exact methods. It was observed in Table 6.4 that MOPPA compared favourably in terms of the quality of BNPs it returns. They are closer

to those returned by exact methods. However, NSGA-II returns nondominated points that are evenly distributed on the nondominated front.

7.3 Future Research

To further improve the solution processes to the problem, the following ideas are suggested.

- Dual Affine Scaling Interior-point MOLP algorithm: Based on our extensive review of the literature, it was observed that this algorithm is yet to be introduced as a solution approach to the problem. It would be interesting if efforts were made to adapt this algorithm as it was done in the single objective linear programming.

- Hybridization: It would be worthwhile to look at the possibility of combining the interior-point method with multi-objective simplex pivoting algorithm in order to generate a more effective algorithm.

- As was noted in Chapter 2, ASIMOLP suffers from the zigzag-

ging phenomenon. It would, therefore, be worthwhile to modify it in order to resolve the issue and potentially speed up its convergence.

Bibliography

[1] Abhyankar, S., Morin, T., and Trafalis, T. Efficient faces of polytopes: Interior point algorithms, parametrization of algebraic varieties, and multiple objective optimization. *Contemporary Mathematics 114* (1990), 319–341.

[2] Aghezzaf, B., and Ouaderhman, T. An interactive interior point algorithm for multiobjective linear programming problems. *Operations Research Letters 29*, 4 (2001), 163–170.

[3] Alves, M. J., Antunes, C. H., and Clímaco, J. Interactive MOLP explorer: a graphical-based computational tool for teaching and decision support in multi-objective linear programming models. *Computer Applications in Engineering Education 23*, 2 (2015), 314–326.

[4] Alves, M. J., and Costa, J. P. An exact method for com-

puting the nadir values in multiple objective linear programming. *European Journal of Operational Research 198*, 2 (2009), 637–646.

[5] Arbel, A. An interior multiobjective linear programming algorithm. *Computers & operations research 20*, 7 (1993), 723–735.

[6] Arbel, A. A weighted-gradient approach to multi-objective linear programming problems using the analytic hierarchy process. *Mathematical and computer modelling 17*, 4 (1993), 27–39.

[7] Arbel, A. Anchoring points and cones of opportunities in interior multiobjective linear programming. *Journal of the Operational Research Society* (1994), 83–96.

[8] Arbel, A. Interior-point methods for multiobjective linear programming problems. In *Multiple Criteria Decision Making*. Springer, 1994, pp. 27–36.

[9] Arbel, A. Using efficient anchoring points for generating search directions in interior multiobjective linear programming. *Journal of the Operational Research Society* (1994), 330–344.

[10] Arbel, A. An interior multiple objective primal-dual linear

programming algorithm using efficient anchoring points. *Journal of the Operational Research Society* (1995), 1121–1132.

[11] Arbel, A. An interior multiobjective primal-dual linear programming algorithm based on approximated gradients and efficient anchoring points. *Computers & operations research 24*, 4 (1997), 353–365.

[12] Arbel, A., and Korhonen, P. Using aspiration levels in an interior primal-dual multiobjective linear programming algorithm. *Journal of Multi-Criteria Decision Analysis 5*, 1 (1996), 61–71.

[13] Arbel, A., and Korhonen, P. Using objective values to start multiple objective linear programming algorithms. *European Journal of Operational Research 128*, 3 (2001), 587–596.

[14] Arbel, A., and Oren, S. S. Generating search directions in multiobjective linear programming using the analytic hierarchy process. *Socio-Economic Planning Sciences 20*, 6 (1986), 369–373.

[15] Arbel, A., and Oren, S. S. Generating interior search

directions for multiobjective linear programming. *Journal of Multi-Criteria Decision Analysis 2*, 2 (1993), 73–86.

[16] Arbel, A., and Oren, S. S. A modification of Karmarkars algorithm to multiple objective linear programming problems. In *Multiple Criteria Decision Making*. Springer, 1994, pp. 37–46.

[17] Arbel, A., and Oren, S. S. Using approximate gradients in developing an interactive interior primal-dual multiobjective linear programming algorithm. *European Journal of Operational Research 89*, 1 (1996), 202–211.

[18] Arbel, A., and Sadka, R. Weighted euclidean centers. *Optimization 54*, 3 (2005), 239–251.

[19] Armand, P. Finding all maximal efficient faces in multiobjective linear programming. *Mathematical Programming 61*, 1-3 (1993), 357–375.

[20] Armand, P., and Malivert, C. Determination of the efficient set in multiobjective linear programming. *Journal of Optimization Theory and Applications 70*, 3 (1991), 467–489.

[21] Bagchi, T. P. *Multiobjective scheduling by genetic algorithms.* Springer Science & Business Media, 1999.

[22] Balicki, J., and Kitowski, Z. Multicriteria evolutionary algorithm with tabu search for task assignment. In *International Conference on Evolutionary Multi-Criterion Optimization* (2001), Springer, pp. 373–384.

[23] Ban, V. T. A finite algorithm for minimizing a concave function under linear constraints and its applications. In *Proceedings of IFIP Working Conference on Recent Advances on System Modelling and Optimization* (1983).

[24] Belenson, S. M., and Kapur, K. C. An algorithm for solving multicriterion linear programming problems with examples. *Journal of the Operational Research Society 24*, 1 (1973), 65–77.

[25] Benayoun, R., De Montgolfier, J., Tergny, J., and Laritchev, O. Linear programming with multiple objective functions: Step method (stem). *Mathematical programming 1*, 1 (1971), 366–375.

[26] Benson, H., Lee, D., and McClure, J. Applying mul-

tiple criteria decision making in practice: The citrus rootstock selection problem in Florida. Tech. rep., Discussion Paper, University of Florida, Department of Decision and Information Sciences, Gainesville, Florida, 1992.

[27] Benson, H. P. Finding an initial efficient extreme point for a linear multiple objective program. *Journal of the Operational Research Society* (1981), 495–498.

[28] Benson, H. P. Further analysis of an outcome set-based algorithm for multiple objective linear programming. *Journal of Optimization Theory and Applications 97*, 1 (1998), 1–10.

[29] Benson, H. P. Hybrid approach for solving multiple-objective linear programs in outcome space. *Journal of Optimization Theory and Applications 98*, 1 (1998), 17–35.

[30] Benson, H. P. 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 (1998), 1–24.

[31] Benson, H. P., Lee, D., and McClure, J. P. Global

optimization in practice: an application to interactive multiple objective linear programming. *Journal of Global Optimization 12*, 4 (1998), 353–372.

[32] Benson, H. P., and Sayin, S. A face search heuristic algorithm for optimizing over the efficient set. *Naval Research Logistics (NRL) 40*, 1 (1993), 103–116.

[33] Benson, H. P., and Sun, E. A weight set decomposition algorithm for finding all efficient extreme points in the outcome set of a multiple objective linear program. *European Journal of Operational Research 139*, 1 (2002), 26–41.

[34] Blanco, V., Puerto, J., and Ali, S. E. H. B. A semidefinite programming approach for solving multiobjective linear programming. *Journal of Global Optimization 58*, 3 (2014), 465–480.

[35] Burton, B. A., and Ozlen, M. Projective geometry and the outer approximation algorithm for multiobjective linear programming. *arXiv preprint arXiv:1006.3085* (2010).

[36] Cardoso, D. M., and Clímaco, J. C. Efficient frontier

scanning in molp using a new tool. In *Multiple Criteria Decision Making*. Springer, 1994, pp. 229–238.

[37] Chakraborty, M., and Ray, A. Parametric approach and genetic algorithm for multi objective linear programming with imprecise parameters. *Opsearch 47*, 1 (2010), 73–92.

[38] Choi, Y. S., and Kim, S. H. Approximation of the set of efficient objective vectors for large scale molp. In *Multiple Criteria Decision Making*. Springer, 1994, pp. 301–310.

[39] Clímaco, J., and Antunes, C. H. Trimap- an interactive tricriteria linear programming package. *Found. Control Eng. 12*, 3 (1987), 101–119.

[40] Climaco, J. C., and Antunes, C. H. Implementation of a user-friendly software packagea guided tour of trimap. *Mathematical and Computer Modelling 12*, 10-11 (1989), 1299–1309.

[41] Cohon, J. L. *Multiobjective Programming and Planning*, vol. 140. Courier Corporation, 2004.

[42] Csirmaz, L. *inner: MOLP solver, https://github.com/lcsirmaz/inner*. 2016.

[43] Csirmaz, L. Using multiobjective optimization to map the entropy region. *Computational Optimization and Applications 63*, 1 (2016), 45–67.

[44] Dauer, J. P. Analysis of the objective space in multiple objective linear programming. *Journal of Mathematical Analysis and Applications 126*, 2 (1987), 579–593.

[45] Dauer, J. P., and Gallagher, R. J. A combined constraint-space, objective-space approach for determining high-dimensional maximal efficient faces of multiple objective linear programs. *European Journal of Operational Research 88*, 2 (1996), 368–381.

[46] Dauer, J. P., and Liu, Y.-H. Solving multiple objective linear programs in objective space. *European Journal of Operational Research 46*, 3 (1990), 350–357.

[47] Dauer, J. P., and Saleh, O. Constructing the set of efficient objective values in multiple objective linear programs. *European Journal of Operational Research 46*, 3 (1990), 358–365.

[48] De, P., and Yadav, B. An algorithm for obtaining optimal

compromise solution of a multi objective fuzzy linear programming problem. *International Journal of Computer Applications 17*, 1 (2011), 20–24.

[49] Deb, K., Agrawal, S., Pratap, A., and Meyarivan, T. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In *International Conference on Parallel Problem Solving From Nature* (2000), Springer, pp. 849–858.

[50] Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE transactions on evolutionary computation 6*, 2 (2002), 182–197.

[51] Dell, R. F., and Karwan, M. H. An interactive mcdm weight space reduction method utilizing a Tchebycheff utility function. *Naval Research Logistics (NRL) 37*, 2 (1990), 263–277.

[52] Dhanalakshmi, S., Kannan, S., Mahadevan, K., and Baskar, S. Application of modified NSGA-II algorithm to combined economic and emission dispatch problem. *Interna-*

tional Journal of Electrical Power & Energy Systems 33, 4 (2011), 992–1002.

[53] Ecker, J., Hegner, N. S., and Kouada, I. Generating all maximal efficient faces for multiple objective linear programs. *Journal of Optimization Theory and Applications 30*, 3 (1980), 353–381.

[54] Ecker, J., and Kouada, I. Finding all efficient extreme points for multiple objective linear programs. *Mathematical Programming 14*, 1 (1978), 249–261.

[55] Ecker, J. G., and Kouada, I. Finding efficient points for linear multiple objective programs. *Mathematical Programming 8*, 1 (1975), 375–377.

[56] Ehrgott, M. *Multicriteria optimization.* Springer Science & Business Media, 2006.

[57] Ehrgott, M., Löhne, A., and Shao, L. A dual variant of Bensons outer approximation algorithm for multiple objective linear programming. *Journal of Global Optimization 52*, 4 (2012), 757–778.

[58] Ehrgott, M., Puerto, J., and Rodriguez-Chia, A. Primal-dual simplex method for multiobjective linear programming. *Journal of optimization theory and applications 134*, 3 (2007), 483–497.

[59] Ehrgott, M., Shao, L., and Schöbel, A. An approximation algorithm for convex multi-objective programming problems. *Journal of Global Optimization 50*, 3 (2011), 397–416.

[60] Ehrgott, M., and Tenfelde-Podehl, D. Computation of ideal and nadir values and implications for their use in mcdm methods. *European Journal of Operational Research 151*, 1 (2003), 119–139.

[61] Eiselt, H. A., and Sandblom, C.-L. *Linear programming and its applications.* Springer Science & Business Media, 2007.

[62] Erlanger, P. *Louis XIV.* Weidenfeld & Nicolson, 1970.

[63] Evans, J. P., and Steuer, R. A revised simplex method for linear multiple objective programs. *Mathematical Programming 5*, 1 (1973), 54–72.

[64] Fliege, J. An efficient interior-point method for convex mul-

ticriteria optimization problems. *Mathematics of Operations Research 31*, 4 (2006), 825–845.

[65] Foroughi, A., and Jafari, Y. A modified method for constructing efficient solutions structure of molp. *Applied mathematical modelling 33*, 5 (2009), 2403–2410.

[66] Fraga, E. S. *http://www.ucl.ac.uk/ uce-cesf/strawberry.html#orgec5771e*. 2018.

[67] Fraga, E. S., and Amusat, O. Understanding the impact of constraints: a rank based fitness function for evolutionary methods. In *Advances in Stochastic and Deterministic Global Optimization*. Springer, 2016, pp. 243–254.

[68] Fraga, E. S., Salhi, A., Zhang, D., and Papageorgiou, L. G. Optimisation as a tool for gaining insight: An application to the built environment. *Journal of Algorithms & Computational Technology 9*, 1 (2015), 13–26.

[69] Gal, T. A general method for determining the set of all efficient solutions to a linear vectormaximum problem. *European Journal of Operational Research 1*, 5 (1977), 307–322.

[70] Gal, T., Stewart, T., and Hanne, T. *Multicriteria decision making: advances in MCDM models, algorithms, theory, and applications*, vol. 21. Springer Science & Business Media, 2013.

[71] Gallagher, R. J., and Saleh, O. A. A representation of an efficiency equivalent polyhedron for the objective set of a multiple objective linear program. *European Journal of Operational Research 80*, 1 (1995), 204–212.

[72] Gao, Y., Xu, C., and Yang, Y. An outcome-space finite algorithm for solving linear multiplicative programming. *Applied Mathematics and Computation 179*, 2 (2006), 494–505.

[73] Geoffrion, A. M. Solving bicriterion mathematical programs. *Operations Research 15*, 1 (1967), 39–54.

[74] Ghosh, A., and Das, M. K. Non-dominated rank based sorting genetic algorithms. *Fundamenta Informaticae 83*, 3 (2008), 231–252.

[75] Grünbaum, B. Convex polytopes, volume 221 of graduate texts in mathematics, 2003.

[76] Haksever, C., and Ringuest, J. L. Computational efficiency and interactive molp algorithms: an implementation of the simolp procedure. *Computers & Operations Research 17*, 1 (1990), 39–50.

[77] Hamel, A. H., Löhne, A., and Rudloff, B. Benson type algorithms for linear vector optimization and applications. *Journal of Global Optimization 59*, 4 (2014), 811–836.

[78] Hartley, R. *Linear and nonlinear programming: An introduction to linear methods in mathematical programming.* Halsted Press, 1985.

[79] Heyde, F., and Löhne, A. Geometric duality in multiple objective linear programming. *SIAM Journal on Optimization 19*, 2 (2008), 836–845.

[80] Heyde, F., Löhne, A., and Tammer, C. Set-valued duality theory for multiple objective linear programs and application to mathematical finance. *Mathematical Methods of Operations Research 69*, 1 (2009), 159–179.

[81] Holland, J. H. *Adaptation in natural and artificial systems:*

An introductory analysis with applications to biology, control, and artificial intelligence. The University of Michigan Press, 1975.

[82] Hu, Y., Bie, Z., Ding, T., and Lin, Y. An NSGA-II based multi-objective optimization for combined gas and electricity network expansion planning. *Applied energy 167* (2016), 280–293.

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

[84] Isermann, H., and Naujoks, G. Operating manual for the EFFACET multiple objective linear programming package. *Fakultaet fuer Wirtschaftswissenschaften, University of Bielefeld, Bielefeld, Germany* (1984).

[85] Jones, D. F., Mirrazavi, S. K., and Tamiz, M. Multi-objective meta-heuristics: An overview of the current state-of-the-art. *European journal of operational research 137*, 1 (2002), 1–9.

[86] Junior, H., and Lins, M. P. E. A win-win approach to multiple objective linear programming problems. *Journal of the Operational Research Society 60*, 5 (2009), 728–733.

[87] Kalyanmoy, D. *Multi objective optimization using evolutionary algorithms.* John Wiley and Sons, 2001.

[88] Kannan, S., Baskar, S., McCalley, J. D., and Murugan, P. Application of NSGA-II algorithm to generation expansion planning. *IEEE Transactions on Power systems 24*, 1 (2009), 454–461.

[89] Karmarkar, N. A new polynomial-time algorithm for linear programming. In *Proceedings of the sixteenth annual ACM symposium on Theory of computing* (1984), ACM, pp. 302–311.

[90] Khachiyan, L., Boros, E., Borys, K., Elbassioni, K., and Gurvich, V. Generating all vertices of a polyhedron is hard. *Discrete & Computational Geometry 39*, 1-3 (2008), 174–190.

[91] Khachiyan, L. G. A polynomial algorithm for linear programming. *Soviet Mathematics Doklady 20* (1979), 191–194.

[92] Kim, N. T. B. Efficiency equivalent polyhedra for the feasible set of multiple objective linear programming. *Acta Mathematica Vietnamica 27*, 1 (2002), 77–85.

[93] Kim, N. T. B., and Thien, N. T. Generating all efficient extreme points in multiple objective linear programming problem and its application, 2007.

[94] Kim, N. T. B., Thien, N. T., and Thuy, L. Q. Generating all efficient extreme solutions in multiple objective linear programming problem and its application to multiplicative programming. *East-West Journal of Mathematics 10*, 1 (2008).

[95] Korhonen, P. Multiple objective linear programming in supporting forest management. In *Multiple Use of Forests and Other Natural Resources*. Springer, 1999, pp. 85–95.

[96] Korhonen, P., and Wallenius, J. A pareto race. *Naval Research Logistics (NRL) 35*, 6 (1988), 615–623.

[97] Korhonen, P. J., and Laakso, J. A visual interactive method for solving the multiple criteria problem. *European Journal of Operational Research 24*, 2 (1986), 277–287.

[98] Küfer, K.-H. On the asymptotic average number of efficient vertices in multiple objective linear programming. *Journal of Complexity 14*, 3 (1998), 333–377.

[99] Lin, C., Chen, C., Chen, P., et al. On the modified interior point algorithm for solving multi-objective linear programming problems. *International Journal of Information and Management Sciences 17*, 1 (2006), 107.

[100] Löhne, A. *Vector optimization with infimum and supremum.* Springer Science & Business Media, 2011.

[101] Löhne, A. *Bensolve: VLP solver, version 1.2, www.bensolve.org.* 2012.

[102] Löhne, A., Rudloff, B., and Ulus, F. Primal and dual approximation algorithms for convex vector optimization problems. *Journal of Global Optimization 60*, 4 (2014), 713–736.

[103] Löhne, A., and Schenker, S. *MOPLIB: Multi-Objective Problem Library, http://moplib.uni-jena.de/.* 2015.

[104] Löhne, A., and Weibing, B. *Bensolve: VLP solver, version 2.0.x, www.bensolve.org.* 2015.

[105] Lotfi, V., Yoon, Y. S., and Zionts, S. Aspiration-based search algorithm (absalg) for multiple objective linear programming problems: theory and comparative tests. *Management Science 43*, 8 (1997), 1047–1059.

[106] Luc, D. T. *Multiobjective Linear Programming: An Introduction.* 2016.

[107] Malakooti, B., and Ravindran, A. Experiments with an interactive paired comparison simplex method for molp problems. *Annals of Operations Research 5*, 1-4 (1986), 575–597.

[108] Martinez-Vargas, A., Domínguez-Guerrero, J., Andrade, Á. G., Sepúlveda, R., and Montiel-Ross, O. Application of NSGA-II algorithm to the spectrum assignment problem in spectrum sharing networks. *Applied Soft Computing 39* (2016), 188–198.

[109] Massobrio, R., Fagúndez, G., and Nesmachnow, S. Multiobjective taxi sharing optimization using the NSGA-II evolutionary algorithm. In *11th Metaheuristic International Conference* (2015).

[110] Michalewicz, Z., and Schoenauer, M. Evolutionary algorithms for constrained parameter optimization problems. *Evolutionary computation 4*, 1 (1996), 1–32.

[111] Michalowski, W., and Szapiro, T. A bi-reference procedure for interactive multiple criteria programming. *Operations Research 40*, 2 (1992), 247–258.

[112] Nandasana, A. D., Ray, A. K., and Gupta, S. K. Applications of the non-dominated sorting genetic algorithm (NSGA) in chemical reaction engineering. *International Journal of Chemical and Reactor Engineering 1* (2003), 1018.

[113] Oliveira, C., and Antunes, C. H. Multiple objective linear programming models with interval coefficients–an illustrated overview. *European Journal of Operational Research 181*, 3 (2007), 1434–1463.

[114] Pandian, P., and Jayalakshmi, M. Determining efficient solutions to multiple objective linear programming problems. *Applied Mathematical Sciences 7*, 26 (2013), 1275–1282.

[115] Pareto, V. *Manuale di economia politica*, vol. 13. Societa

Editrice, 1906.

[116] Pei, Y., and Hao, J. Non-dominated sorting and crowding distance based multi-objective chaotic evolution. In *International Conference in Swarm Intelligence* (2017), Springer, pp. 15–22.

[117] Philip, J. Algorithms for the vector maximization problem. *Mathematical programming 2*, 1 (1972), 207–229.

[118] Philip, J. Vector maximization at a degenerate vertex. *Mathematical Programming 13*, 1 (1977), 357–359.

[119] Pourkarimi, L., Yaghoobi, M., and Mashinchi, M. Determining maximal efficient faces in multiobjective linear programming problem. *Journal of Mathematical Analysis and Applications 354*, 1 (2009), 234–248.

[120] Quaddus, M., and Holzman, A. Imolp: an interactive method for multiple objective linear programs. *IEEE transactions on systems, man, and cybernetics 16*, 3 (1986), 462–468.

[121] Razali, N. M., Geraghty, J., et al. Genetic algorithm performance with different selection strategies in solv-

ing TSP. In *Proceedings of the world congress on engineering* (2011), vol. 2, International Association of Engineers Hong Kong, pp. 1134–1139.

[122] Reeves, G. R., and Franz, L. S. A simplified interactive multiple objective linear programming procedure. *Computers & operations research 12*, 6 (1985), 589–601.

[123] Rodman, A. D., Fraga, E. S., and Gerogiorgis, D. On the application of a nature-inspired stochastic evolutionary algorithm to constrained multi-objective beer fermentation optimisation. *Computers & Chemical Engineering 108* (2018), 448–459.

[124] Rudloff, B., Ulus, F., and Vanderbei, R. A parametric simplex algorithm for linear vector optimization problems. *Mathematical Programming* (2015), 1–30.

[125] Ruszczyński, A., and Vanderbei, R. J. Frontiers of stochastically nondominated portfolios. *Econometrica* (2003), 1287–1297.

[126] Saaty, T., and Vargas, L. G. The logic of priorities: Appli-

cation in business, energy, health, and transportation. *Nijhoff, Boston* (1982).

[127] Saaty, T. L. The analytic hierarchy process: planning, priority setting, resources allocation. *New York: McGraw* (1980).

[128] Salhi, A., and Fraga, E. S. Nature-inspired optimisation approaches and the new plant propagation algorithm. In *Proceedings of the International Conference on Numerical Analysis and Optimisation (ICeMATH'11), Yogyakarta, Indonesia* (2011).

[129] Sayin, S. An algorithm based on facial decomposition for finding the efficient set in multiple objective linear programming. *Operations Research Letters 19*, 2 (1996), 87–94.

[130] Schaffer, J. D. Multiple objective optimization with vector evaluated genetic algorithms. In *Proceedings of the 1st International Conference on Genetic Algorithms, Pittsburgh, PA, USA, July 1985* (1985), pp. 93–100.

[131] Schechter, M., and Steuer, R. E. A correction to the connectedness of the Evans-Steuer algorithm of multiple objective

linear programming. *Foundations of Computing and Decision Sciences 30*, 4 (2005), 351–360.

[132] Schönfeld, K. P. *Effizienz und Dualität in der Aktivitätsanalyse (Diss.)*. Freie Universität Berlin, Germany, 1964.

[133] Seiford, L., and Yu, P.-L. Potential solutions of linear systems: The multi-criteria multiple constraint levels program. *Journal of Mathematical Analysis and Applications 69*, 2 (1979), 283–303.

[134] Selamoğlu, B. İ., and Salhi, A. The plant propagation algorithm for discrete optimisation: The case of the travelling salesman problem. In *Nature-inspired computation in engineering*. Springer, 2016, pp. 43–61.

[135] Shao, L., and Ehrgott, M. Approximately solving multiobjective linear programmes in objective space and an application in radiotherapy treatment planning. *Mathematical Methods of Operations Research 68*, 2 (2008), 257–276.

[136] Shao, L., and Ehrgott, M. Approximating the nondominated set of an molp by approximately solving its dual problem.

Mathematical Methods of Operations Research 68, 3 (2008), 469–492.

[137] Shao, L., and Ehrgott, M. An objective space cut and bound algorithm for convex multiplicative programmes. *Journal of Global Optimization 58*, 4 (2014), 711–728.

[138] Shao, L., and Ehrgott, M. Primal and dual multi-objective linear programming algorithms for linear multiplicative programmes. *Optimization* (2015), 1–17.

[139] Smith, A. E., and Coit, D. W. Constraint handling techniques: penalty functions. *Handbook of evolutionary computation* (1997), 5–2.

[140] Srinivas, N., and Deb, K. Multi-objective function optimization using non-dominated sorting genetic algorithms. *Evolutionary computation 2*, 3 (1995), 221–248.

[141] Steuer, R. E. Multiple objective linear programming with interval criterion weights. *Management Science 23*, 3 (1976), 305–316.

[142] Steuer, R. E. An interactive multiple objective linear pro-

gramming procedure. *TIMS Studies in the Management Sciences 6* (1977), 225–239.

[143] Steuer, R. E. *Multiple criteria optimization: theory, computation, and applications.* Wiley, 1986.

[144] Steuer, R. E. Adbase: A multiple objective linear programming solver for all efficient extreme points and all unbounded efficient edges. *Terry college of Business, University of Georgia, Athens* (2003).

[145] Stewart, T. J. An interactive multiple objective linear programming method based on piecewise-linear additive value functions. *Systems, Man and Cybernetics, IEEE Transactions on 17*, 5 (1987), 799–805.

[146] Strijbosch, L. W., Van Doorne, A. G., and Selen, W. J. A simplified MOLP algorithm: the MOLP-S procedure. *Computers & operations research 18*, 8 (1991), 709–716.

[147] Sulaiman, M., Salhi, A., Selamoglu, B. I., and Kirikchi, O. B. A plant propagation algorithm for constrained engineering optimisation problems. *Mathematical*

Problems in Engineering 2014 (2014).

[148] Suprajitno, H. Solving multiobjective linear programming problem using interval arithmetic. *Applied Mathematical Sciences 6*, 80 (2012), 3959–3968.

[149] Tantawy, S. Detecting non-dominated extreme points for multiple objective linear programming. *Journal of Mathematics and Statistics 3*, 3 (2007), 77–79.

[150] Trafalis, T. B., and Alkahtani, R. M. An interactive analytic center trade-off cutting plane algorithm for multiobjective linear programming. *Computers & industrial engineering 37*, 3 (1999), 649–669.

[151] Wen, U.-P., and Weng, W.-T. An interior algorithm for solving multiobjective linear programming problem. *Institute for Operations Research and the Management Sciences International Meeting: Tel Aviv - Israel* (1998).

[152] Weng, W.-T., and Wen, U.-P. An interior point algorithm for solving linear optimization over the efficient set problems. *Journal of the Chinese Institute of Industrial Engineers 18*, 3

(2001), 21–30.

[153] Wierzbicki, A. P. The use of reference objectives in multiobjective optimization. In *Multiple criteria decision making theory and application*. Springer, 1980, pp. 468–486.

[154] Yan, H., Wei, Q., and Wang, J. Constructing efficient solutions structure of multiobjective linear programming. *Journal of mathematical analysis and applications 307*, 2 (2005), 504–523.

[155] Yeniay, Ö. Penalty function methods for constrained optimization with genetic algorithms. *Mathematical and computational Applications 10*, 1 (2005), 45–56.

[156] Yu, P., and Zeleny, M. The techniques of linear multiobjective programming. *Revue française d'automatique, d'informatique et de recherche opérationnelle. Recherche opérationnelle 8*, 3 (1974), 51–71.

[157] Yu, P., and Zeleny, M. The set of all nondominated solutions in linear cases and a multicriteria simplex method. *Journal of Mathematical Analysis and Applications 49*, 2 (1975), 430–

468.

[158] Yu, P., and Zeleny, M. Linear multiparametric programming by multicriteria simplex method. *Management Science 23*, 2 (1976), 159–170.

[159] Yuen, T. J., and Ramli, R. Comparision of compuational efficiency of MOEA\D and NSGA-II for passive vehicle suspension optimization. *ECMS 2010* (2010), 219–225.

[160] Zadeh, L. Optimality and non-scalar-valued performance criteria. *IEEE transactions on Automatic Control 1*, 8 (1963), 59–60.

[161] Zeleny, M. Compromise programming. In *Cochrane JL, Zeleny M (eds), Multiple criteria decision making.* University of South Carolina Press, Columbia, SC, 1973, pp. 262–301.

[162] Zeleny, M. *Linear multiobjective programming*, vol. 95. Springer Science & Business Media, 1974.

[163] Zeleny, M. *Multiple criteria decision making*, vol. 25. McGraw-Hill New York, 1982.

[164] Zhang, W. A compromise programming method using multi-bounds formulation and dual approach for multicriteria structural optimization. *International journal for numerical methods in engineering 58*, 4 (2003), 661–678.

[165] Zhong, Y., and Shi, Y. An interior-point approach for solving MC2 linear programming problems. *Mathematical and computer modelling 34*, 3 (2001), 411–422.

[166] Zionts, S., and Wallenius, J. An interactive programming method for solving the multiple criteria problem. *Management science 22*, 6 (1976), 652–663.

[167] Zionts, S., and Wallenius, J. An interactive multiple objective linear programming method for a class of underlying nonlinear utility functions. *Management Science 29*, 5 (1983), 519–529.

Appendices

Appendix A

Non-Interactive Simplex based methods

SN	Author(s)	Year	Title
1	Schonfeld, K. P.	1964	Effizienz und Dualitat in der Aktivitatsanalyse
2	Geoffrion, A. M.	1967	Solving bicriterion mathematical programs
3	Philip, J.	1972	Algorithms for the vector maximization problem
4	Evans & Steuer	1973	A revised simplex method for linear multiple objective programming
5	Belenson & Kapur	1973	An algorithm for solving multicriterion linear programming problems with examples
6	Zeleny, M.	1973	Compromise programming
7	Zeleny, M.	1974	Linear multiobjective programming
8	Yu and Zeleny	1974	The techniques of linear multiple objective programming
9	Yu and Zeleny	1975	The set of all nondominated solutions in linear cases and a multicriteria simplex method
10	Ecker and Kouada	1975	Finding efficient points for linear multiple objective programs
11	Yu and Zeleny	1976	Linear multiparametric programming

Continued on next page

Table A.1 – continued from previous page

SN	Author(s)	Year	Title
			by multicriteria simplex method
12	Gal, T.	1977	A general method for determining the set of all efficient solutions to a linear vector maximization problem
13	Isermann, H.	1977	The enumeration of the set of all efficient solutions for a linear multiple objective program
14	Philip, J.	1977	Vector optimization at a degenerate vertex
15	Ecker and Kouada	1978	Finding all efficient extreme points for multiple objective linear programs
16	Seiford and Yu	1979	Potential solutions of linear systems: The multicriteria multiple constraint levels program
17	Ecker et al.	1980	Generating all maximal efficient faces for multiobjective linear programs
18	Benson, H. P	1981	Finding an initial efficient extreme point for linear multiobjective program
19	Zeleny, M	1982	Multiple criteria decision making
20	Steuer, R. E.	1986	Multiple criteria optimization: theory, computation & applications
21	Strijbosch et al.	1991	A simplified MOLP algorithm: The MOLP-S procedure
22	Armand and Malivert	1991	Determination of the efficient set in multiobjective linear programming
23	Armand, P.	1993	Finding all maximal efficient faces in multiobjective linear programming
24	Cardoso and Climaco	1994	Efficient frontier scanning in MOLP using a new tool
25	Sayin, S.	1996	An algorithm based on facial decomposition for finding the efficient set in MOLP
26	Ruszczynski and Vanderbei	2003	Frontiers of stochastically nondominated portfolios
27	Yan et al.	2005	Constructing efficient solutions

Continued on next page

Table A.1 – continued from previous page

SN	Author(s)	Year	Title
			structure of multiobjective linear programming
28	Kim and Thien	2007	Generating all efficient extreme points in multiple objective linear programming problem and its application
29	Ehrgott et al.	2007	Primal-dual simplex method for multiobjective linear programming
30	Kim et al.	2008	Generating all efficient extreme solutions in MOLP and its application to multiplicative programming
31	Pourkarimi et al.	2009	Determining maximal efficient faces in multiobjective linear programming problem
32	Foroughi & Jafari	2009	A modified method for construction efficient solutions structure of MOLP
33	Suprajitno, H	2012	Solving multiobjective linear programming problem using interval arithmetic
34	Pandian and Jayalakshmi	2013	Determining efficient solutions to multiple objective linear programming problems
35	Rudloff et al.	2015	A parametric simplex algorithm for linear vector optimization problems An interior point multiple criteria
36	Luc, D. T.	2016	Multiobjective Linear Programming: An Introduction

Appendix B

Interactive Simplex based methods

SN	**Author(s)**	**Year**	**Title**
SN	Author(s)	Year	Title
1	Benayoun et al	1971	Linear programming with multiple objective functions
2	Zionts and Wallenius	1976	An interactive programming method for solving the multiple criteria problem
3	Steuer, R. E.	1976	Multiple objective linear programming with interval criterion weights
4	Steuer, R. E.	1977	An interactive multiple objective linear programming procedure
5	Zionts and Wallenius	1983	An interactive multiple objective linear programming method for a class of underlying nonlinear utility functions
6	Reeves and Franz	1985	A simplified interactive multiple objective linear programming procedure
7	Korhonen and Laakso	1986	A visual interactive method for

Continued on next page

Table B.1 – continued from previous page

SN	Author(s)	Year	Title
			solving the multiple criteria problem
8	Quaddus and Holzman	1986	Imolp: an interactive method for multiple objective linear programs
9	Arbel and Oren	1986	Generating serach directions in MOLP using the analytic hierarchy process
10	Malakooti and Ravindran	1986	Experiments with an interactive paired comparison simplex method for molp problems
11	Stewart, T. J.	1987	An interactive MOLP method based on piecewise linear additive value functions
12	Climaco and Antunes	1987	Trimap - an interactive tricriteria linear programming package
13	Korhonen and Wallenius	1988	A Pareto race
14	Dell and Karwan	1990	An interactive mcdm weight space reduction method utilizing a Tcheby-cheff utility function
15	Michalowski and Szapiro	1992	A bi-reference procedure for interactive multiple criteria programming
16	Choi and Kim	1994	Approximation of the set of efficient objective vectors for large scale molp
17	Lotfi et al.	1997	Aspiration-based search algorithm for multiple objective linear programming problems: theory and comparative tests
18	Benson et al.	1998	Global optimization in practice: an application to interactive multiple

Continued on next page

Table B.1 – continued from previous page

SN	Author(s)	Year	Title
			objective linear programming
19	Alves et al.	2015	Interactive MOLP explorer: A graphical-based computational tool for teaching and decision support in multi-objective linear programming models

Appendix C

Objective space based methods

SN	Author(s)	Year	Title
1	Dauer, J. P.	1987	Analysis of the objective space in Multiobjective linear programming
2	Dauer & Liu	1990	Solving multiple objective linear programs in objective space
3	Dauer & Saleh	1990	Consrtucting the set of efficient objective in multiple objective linear programs
4	Gallagher & Saleh	1995	A representation of an efficient equivalent polyhedron for the objective set of a MOLP
5	Dauer & Gallagher	1996	A combined constraint space, objective space approach for determining high dimensional efficient faces of MOLP
6	Benson, H. P.	1998a	An outer approximation algorithm for generating all efficient extreme points in the outcome set of MOLP problem
7	Benson, H. P.	1998b	Further analysis of an outcome set-based algorithm

Continued on next page

Table C.1 – continued from previous page

SN	Author(s)	Year	Title
			for MOLP
8	Benson, H. P.	1998c	Hybrid approach for solving multiple objective linear programs in outcome space
9	Benson & Sun	2002	A weight set decomposition algorithm for finding all extreme points in the outcome set of a MOLP
10	Kim, N. T. B.	2002	Efficiency equivalent polyhedra for feasible set of MOLP
11	Yan *et al.*	2005	Constructing efficient solutions structure of multiobjective linear programming
12	Tantawy, S.	2007	Detecting nondominated extreme points for MOLP
13	Shao & Ehrgott	2008a	Approximately solving MOLPs in objective space and an application in radiotherapy treatment planning
14	Shao & Ehrgott	2008b	Approximating the nondominated set of an MOLP by approximately solving its dual problem
15	Heyde & Löhne	2008	Geometric duality in Multiple objective linear programming
16	Heyde *et al.*	2009	Set-valued duality theory for MOLP & application to mathematical finance
17	Burton & Ozlen	2010	Projective geometry and the outer approximation algorithm for MOLP
18	Ehrgott *et al.*	2011	An approximation algorithm for convex MOLP problem
19	Ehrgott *et al.*	2012	A dual variant of Benson's outer approximation algorithm for MOLP
20	Lohne, A.	2012	Vector optimization with infimum and supremum
21	Csirmaz, L.	2016	Using multiobjective optimization to map the entropy region.
22	Hamel *et al.*	2014	Benson type algorithms for linear vector

Continued on next page

Table C.1 – continued from previous page

SN	Author(s)	Year	Title
			optimization & applications
23	Lohne *et al.*	2014	Primal and dual approximation algorithms for convex vector optimization problems
24	Shao & Ehrgott	2015	Primal and dual MOLP algorithms for linear multiplicative programming

Appendix D

Non-Interactive Interior-point based methods

SN	Author(s)	Year	Title
1	Abhyankar et al.	1990	Efficient faces of polytopes: interior point algorithms, parametrization of algebraic varieties and multiple objective optimization
2	Arbel, A.	1993a	An interior multiobjective linear programming algorithm
3	Arbel, A.	1993b	A weighted-gradient approach to MOLP problems using the analytic hierarchy process
4	Arbel and Oren	1994	A modification of Karmarkar's algorithm to multiple objective linear programming problems
5	Arbel, A.	1994a	Anchoring points and cones of opportunities in interior multiple objective linear porgramming

Continued on next page

Table D.1 – continued from previous page

SN	Author(s)	Year	Title
6	Arbel, A.	1994b	Using efficient anchoring points for generating search directions in interior MOLP
7	Arbel, A.	1994c	Interior point methods for multiobjective linear programming problems: In multiple criteria decision making
8	Wen and Weng	1998	An interior algorithm for solving multiobjective linear programming problem
9	Bufardi, A.	1999	Multicriteria decision making: and advances in MCDM models, algorithms, theory, and applications
10	Arbel and Korhonen	2001	Using objective values to start multiple objective linear programming algorithms
11	Weng and Wen	2001	An interior point algorithm for solving linear optimization over the efficient set problems
12	Zhong and Shi	2001	An interior point approach for solving mc2 linear programming problems
13	Arbel and Sadka	2005	Weighted euclidean centers
14	Lin et al.	2006	On the modified interior point algorithm for solving MOLP problems
15	Fliege, J.	2006	An efficient interior point method for convex multicriteria optimization problems

Appendix E

Interactive Interior-point based methods

SN	Author(s)	Year	Title
1	Arbel and Oren	1993	Generating interior search directions for multiobjective linear programming
2	Arbel, A.	1995	An interior multiple objective primal-dual linear programming algorithm using efficient anchoring points
3	Arbel and Oren	1996	Using approximate gradients in developing an interactive primal-dual MOLP algorithm
4	Arbel & Korhonen	1996	Using aspiration levels in an interior primal-dual multiple objective linear programming algorithm
5	Arbel, A.	1997	An interior multiobjective primal dual linear programming algorithm based on approximated gradients

Continued on next page

Table E.1 – continued from previous page

SN	Author(s)	Year	Title
			& efficient anchor points
6	Trafalis and Alkahtani	1999	An interactive analytic centre trade off cutting plane algorithm for multiobjective linear programming
7	Junior and Lins	2009	A win-win approach to multiple objective linear programming problems
8	Aghezzaf and Ouaderhmann	2001	An interactive interior point algorithm for MOLP problems

Appendix F

Script used in generating Problem 50 to 52.

The script in Bensolve-2.0 [104] that was used to generate Problems 50 to 52 in Table 3.2 is given below:

MOLP with q number of objectives, $n = (q + 2 * m)^q$ as the number of variables and constraints. One can adopt q and m to generate other larger instances as was done in bensolve-2.0.1 $\backslash ex \backslash example10.m$

$n = (q + 2 * m)^q$, is the number of variables and constraints
A, the constraint matrix is given by
$A = eye(n)$, where n is the number of variables.
b, the RHS vector is given by
$b = ones(n, 1)$.

To generate the criterion matrix C;
$C = zeros(n, q)$;
for i=1:n
    $line = dec2base(i - 1, q + 2 * m, q) - \text{`0`}$;
    $line = line - (q + 2 * m - 1)/2$;
    $C(i, :) = line$;
end

$C = C'$.

If $q = m = 2$, Problem 50 in Table 3.2 is generated with 36 number of variables, 36 constraints and 2 objective functions.
When $q = 2$ and $m = 3$, Problem 51 in Table 3.2 is generated with 64 number of variables, 64 constraints with 2 objective functions.
When $q = 2$ and $m = 4$, Problem 52 in Table 3.2 is generated with 100 variables, 100 constraints and 2 objective functions.
Finally, when $q = 3$ and $m = 2$, Problem 53 in Table 3.2 is generated with 343 number of variables, 343 constraints and 3 objective functions.

Appendix G

List of papers submitted/awaiting submission to Journals

1. On the Simplex, Interior-point, and Objective space approaches to Multi-objective Linear Programming: Revised version resubmitted to Journal of the Operational Research Society (JORS).
2. A Comparative study of two key algorithms in Multi-objective Linear Programming: Revised version awaiting resubmission to Journal of Algorithms and Computational Technology (JACT).
3. A Comparison of Benson's Outer-approximation Algorithm with an Extended version of Multi-objective Simplex Algorithm: Submitted to Asia-Pacific Journal of Operational Research (APJOR).
4. Multi-objective Linear Programming: A Survey: Awaiting second submission.
5. Application of Plant Propagation Algorithm and NSGA-II to Multi-objective Linear Programming: Awaiting submission.