DEPARTMENT OF MATHEMATICS

Systematic Evolutionary Algorithm for a general Multilevel Stackelberg Problems with bounded decision variables (SEAMSP)

By: **Ashenafi Teklay Woldemariam**

Advisor: **Semu Mitiku Kassa (PhD)**

A Thesis Submitted to the Department of Mathematics of Addis Ababa University in Partial Fulfillment of the Requirements for the Master of Science Degree in Mathematics

July 1, 2013

Declaration

This thesis is my original work and has not been presented for a degree in any other University and that all sources of information used for the thesis have been fully acknowledged.

Ashenafi Teklay Woldemariam Signature: _____

This thesis is submitted for examination with my approval as a university advisor.

Dr. Semu Mitiku Kassa Signature: _____

© Copyright by Ashenafi Teklay Woldemariam, 2013

Acknowledgement

First and foremost, I would like to thank my advisor Dr. Semu, for being all that I need from an advisor. I feel myself very lucky to get his constant assistance and encouragement throughout my thesis. I would also like to thank Administration for Refugee-Retunee Affairs (ARRA), for providing this opportunity and Mathematics Department of Addis Ababa University (AAU), for accepting my application.

I thankfully acknowledge the financial support from International Science Program (ISP), Sweden through the Department of Mathematics, AAU and I am very grateful to all the staff members of the Department for their help over the courses of my study.

Special thanks go to my family for their love and encouragement over the years of my study.

This work is dedicated to my brother, Berhane Teklay Woldemariam.

Contents

Abstract

Multilevel Stackelberg Problems (MSPs) are nested optimization problems which reply hierarchical decisions of subproblems. Each decision maker (DM) in the hierarchy admits the decision of those above its level (if exist), observes the response of those below (if exist) for each possible value of its decision variable and returns the best variable value/s of its interest. These kind of problems are known to be common in distinct areas of study. Linear MSPs are shown to be NP-hard problems by different authors. The inclusion of non-linear, non-convex, non-differentiable and other undisciplined property of functions add further complexity to the problem. Unfortunately, real life situations are crowded with such kind of functions. Most existing algorithms in MSPs are proposed for bilevel stackelberg problems (BSPs), specially the linear version of BSPs. Systematic evolutionary algorithm for a general multilevel stackelberg problems (SEAMSP) having bounded decision spaces, has been proposed in this work. A unique feature of the algorithm is that it is not affected by the behavior of the objective and constraint functions involved in a problem. The proposed algorithm apply evolutionary algorithm concepts to MSPs, with systematic way of selecting initial populations at each iteration and a newly constructed mutation operator, which is suitable to the selection of populations. In SEAMSP, each decision space is controlled by **"intelligent spies"** having a nice cooperation with the spies on the other decision spaces, for representing the whole constraint region in a random, unique, diverse and systematic way. The numerical results on various problems demonstrated that the proposed algorithm is very much promising to MSPs without any limitation of the included functions, and it can be used as a benchmark for a comparison of approximate results by other algorithms.

Notations and Abbreviations

DM	Decision Maker
MPP	Multilevel Programming Problem
MSP	Multilevel Stackelberg Problem
BPP	Bilevel Programming Problem
BSP	Bilevel Stackelberg Problem
TPP	Trilevel Programming Problem
TSP	Trilevel Stackelberg Problem
R	The set of all real numbers
SS	Stackelberg Solution set
AS	Approximate Solution set by SEAMSP
$a \lesssim b$	maximum natural number a which is less than or equal to b and different from 1
$a - b \approx 0$	possible natural numbers a and b with minimum value of $\|a - b\|$
$\|x\|$	Euclidean norm of $x = (x_1, \ldots, x_n) \in R^n$, i.e $\|x\| = (\sum_{i=1}^{n} x_i^2)^{\frac{1}{2}}$
$d(x, y)$	euclidean distance between x and y, i.e if $x, y \in R^n$, $x = (x, \ldots, x_n)$ and $y = (y_1, \ldots, y_n)$ then, $d(x, y) = (\sum_{i=1}^{n} \|x_i - y_i\|^2\|)^{\frac{1}{2}}$
$\beta(x, \mu)$	an open ball of radius μ about the point x
$\|H\|$	number of elements in set H

Chapter 1

Introduction

1.1 Background

Many organizational decisions are made in a multilevel hierarchical structures, and although each decision level has no direct control over the decisions made by the others, the actions taken at one-decision level do affect those taken at all other levels. Decision-makers at all levels attempt to optimize individual objectives, but their decisions are affected by the optimal objective values present at other levels. This hierarchical decision process arises in many fields, including decentralized resource planning, highway pricing, the power market, logistics, economics, manufacturing, and road network management. Such types of decisions are referred to as multilevel decision problems. Multilevel programming is typically used to solve these kinds of problems.[63]

Decision making problems in decentralized organizations are often modeled as Stackelberg games. In the context of two-level programming, the DM at the upper level first specifies a strategy, and then the DM at the lower level specifies a strategy so as to optimize the objective with full knowledge of the action of the DM at the upper level. Assuming that the lower level DM behaves rationally, that is, optimally responds to the decision of the upper level DM, the upper level DM also specifies the strategy so as to optimize its objective. Assuming that, there is no communication among decision makers, or they do not make any binding agreement even if there exists such commu-

nication, we will refer to the problem described above as a **stackelberg problem** and the optimal solution of the problem as a **stackelberg solution**.[35]
Algorithms for function optimization are generally limited to convex regular functions and the solution methods proposed for multilevel optimization problems are mainly for bilevel optimization problems with linear or convex property. Moreover, the linear bilevel problem has been shown to be NP-hard in [7]. Refer to [33] and [34] for details of NP-hard problems. However, many functions are multi-modal, discontinuous, and non-differentiable. Evolutionary algorithms have been proposed to optimize these functions in a single optimization problems. Whereas traditional search techniques use characteristics of the functions involved in the problem to determine the next sampling point (e.g. gradients, Hessians, linearity, and continuity), evolutionary algorithms make no such assumptions. Instead, the next sampled point is selected randomly from a certain region rather than a set of deterministic decision rules.

1.2 Mathematical Formulation of a Multilevel Stackelberg Problem

An optimization problem which has other optimization problems in the constraint set and has a decision maker for each objective function controlling part of the variables is called **multilevel optimization problem**. The problem may have k decision makers with their own objective functions and each controlling part of the variables. [49]

The general k-level programming problems can be expressed as:

$$\begin{aligned}
&\min_{x_1 \in X_1} f_1(x_1, \ldots, x_k) \\
&s.t. \quad (x_1, \ldots, x_k) \in S_1 \\
&\qquad \min_{x_2 \in X_2} f_2(x_1, \ldots, x_k) \\
&\qquad s.t. \quad (x_1, \ldots, x_k) \in S_2 \\
&\qquad\qquad . \\
&\qquad\qquad\quad . \\
&\qquad\qquad\qquad . \\
&\qquad\qquad \min_{x_k \in X_k} f_k(x_1, \ldots, x_k) \\
&\qquad\qquad s.t. \quad (x_1, \ldots, x_k) \in S_k
\end{aligned} \tag{1.1}$$

For a mathematical formulation of a multilevel stackelberg problems, consider the MPP in (1.1) composed of k-levels each characterized by individual objective functions f_i for $i = 1, 2, \ldots, k$, which are to be minimized by the respective decision makers. Let the decision variable space R^n be partitioned among k-levels, such that $(x_1, x_2, \ldots, x_k) \in X_1 \times X_2 \times \cdots \times X_k = S \subseteq R^n$, where $X_i \subseteq R^{n_i}$, $\sum_{i=1}^{k} n_i = n$ and S be a nonempty set. Assume that decisions are made sequentially beginning with DM_1 (called the leader DM) who has control over a vector $x_1 \in X_1$, followed by DM_2 who has control over a vector $x_2 \in X_2$ down through DM_k who has control over a vector $x_k \in X_k$.
A Multilevel Stackelberg Problem of the MPP in (1.1) can be defined as follows:-

- Constraint region of the MPP
  $\Omega = \{(x_1, \ldots, x_k) \in S : (x_1, \ldots, x_k) \in S_1 \cap S_2 \cdots \cap S_k\}$

- For each given vector $(x_1^*, \ldots, x_p^*) \in X_1 \times \cdots \times X_p$, $1 \leq p < k$, feasible region of $p+1$, and lower order levels is given by:-
  $\Omega(x_1^*, \ldots, x_p^*) = \{(x_{p+1}, \ldots, x_k) \in X_{p+1} \times \cdots \times X_k : (x_1^*, \ldots, x_p^*, x_{p+1}, \ldots, x_k) \in S_{p+1} \cap \cdots \cap S_k\}$

- Projection of Ω onto the $1^{st}, \ldots, p^{th}$, $1 \leq p < k$, levels decision space is given by:-
  $\Omega(X_1, \ldots, X_p) = \{(x_1, \ldots, x_p) : \exists (x_{p+1}, \ldots, x_k), (x_1, \ldots, x_k) \in \Omega\}$

- For each $(x_1^*, \ldots, x_p^*) \in \Omega(X_1, \ldots, X_p)$, $1 \leq q < k$, $p = k - q$ rational reaction set for the $p+1$ and lower order levels is given by:-
  $M(x_1^*, \ldots, x_p^*) = \{(x_{p+1}^*, \ldots, x_k^*) : (x_{p+1}^*, \ldots, x_k^*) \in argmin\{f_{p+1}(x_1^*, \ldots, x_p^*, x_{p+1}, \ldots, x_k) : (x_{p+1}, \ldots, x_k) \in \Omega(x_1^*, \ldots, x_p^*), (x_{p+2}, \ldots, x_k) \in M(x_1^*, \ldots, x_p^*, x_{p+1})\}\}$

- The Induced Region at level one is :
  $IR = \{(x_1, \ldots, x_k) \in S_1 : x_1 \in X_1, (x_2, \ldots, x_k) \in M(x_1)\}$

- Using the Induced Region one can describe the MSP at the level one of the MPP in (1.1) as:

$$\begin{aligned} & min\ f_1(x_1, \ldots, x_k) \\ & \text{s.t. } (x_1, \ldots, x_k) \in IR \end{aligned}$$

In our definition of MPP in (1.1), if there are only two nested objective functions then it is called a Bilevel Programming Problem. For the sake of simplicity, we define the particular class of MPP, the BPP , in the next subsection.

1.2.1 Bilevel Stackelberg Problem

Consider a case where there are two decision makers, in which the top decision maker first makes a decision and the bottom one who knows the decision of the opponent makes a decision next. Such a situation is formulated as a bilevel programming problem.
Let $(x, y) \in X \times Y = S \subseteq R^{n_1} \times R^{n_2}$, $\quad f, F : R^{n_1} \times R^{n_2} \to R$,
$S_2 = \{(x, y) : g(x, y) \leq 0, h(x, y) = 0\}$, $S_1 = \{(x, y) : G(x, y) \leq 0, H(x, y) = 0\}$,
$g = [g_1, \ldots, g_r] : R^{n_1} \times R^{n_2} \to R^r$, $h = [h_1, \ldots, h_{r'}] : R^{n_1} \times R^{n_2} \to R^{r'}$
$G = [G_1, \ldots, G_p] : R^{n_1} \times R^{n_2} \to R^p$, $H = [H_1, \ldots, H_{p'}] : R^{n_1} \times R^{n_2} \to R^{p'}$

The general BPP can then be defined as:

$$\begin{aligned} &\min_{x\in X} F(x,y) \\ &s.t. \quad G(x,y) \leq 0 \\ &\qquad H(x,y) = 0 \\ &\qquad \min_{y\in Y} f(x,y) \\ &\qquad s.t. \quad g(x,y) \leq 0 \\ &\qquad\qquad h(x,y) = 0 \end{aligned} \tag{1.2}$$

We call the top DM who first makes a decision the leader and the other DM the follower. In non-cooperative situations, for each decision made by the leader, the follower with a full knowledge of the decision of the leader, reacts with an optimal solution of its problem. According to this rule the leader also makes a decision which optimizes its objective function. The optimal decision of the leader with an optimal reaction of the follower after the leader's decision is called a Stackelberg equilibrium in the fields of game theory and economics, and we refer to it as a Stackelberg solution. [35] The whole situation described above is refereed to us a Stackelberg problem.
A Bilevel Stackelberg Problem of the BPP in (1.2) can be defined as [51]:

- Constraint region of the BPP
  $\Omega = \{(x,y) : x \in X, y \in Y,\ G(x,y) \leq 0,\ H(x,y) = 0,\ g(x,y) \leq 0,\ h(x,y) = 0\}$
- For each given $x^* \in X$, the follower's feasible region is :
  $\Omega(x^*) = \{y : y \in Y,\ g(x^*,y) \leq 0,\ h(x^*,y) = 0\}$
- Projection of Ω onto the leader's decision space gives :
  $\Omega(X) = \{x : \exists y, (x,y) \in \Omega\}$
- For each given $x^* \in \Omega(X)$, the follower reaction set is :
  $M(x^*) = \{y : y \in argmin\{f(x^*,y) : y \in \Omega(x^*)\}\}$
- The Induced Region at the upper level is :
  $IR = \{(x,y) : x \in X, G(x,y) \leq 0, H(x,y) = 0, y \in M(x)\}$.

- The BSP at the the upper level of the BPP in (1.2) can be defined as:

$$\begin{aligned} &\min F(x,y) \\ s.t. \quad &(x,y) \in IR \end{aligned} \tag{1.3}$$

In case, if all the functions (F, f, $G_1,\ldots,G_p$, $H_1,\ldots,H_{p'}$, $g_1,\ldots,g_r$, $h_1,\ldots,h_{r'}$) in the BPP (1.2) are linear, we call it a linear-BPP.
Now let us describe the BSP by looking at the follower's reaction to the leader's action. Given that a strategy $x^* \in X$ is chosen by the leader, the follower reacts with its optimal decisions in $M(x^*) \subseteq Y$ by solving the optimization problem given by:-

$$\begin{aligned} &\min f(x^*,y) \\ s.t \quad &g(x^*,y) \leq 0 \\ &h(x^*,y) = 0 \\ &y \in Y \end{aligned} \tag{1.4}$$

If for each $x^* \in X$, (1.4) has a unique optimal solution ,$M(x^*)$, then we can reformulate the BSP at the upper level in (1.3) as:

$$\begin{aligned} &\min F(x,M(x)) \\ s.t \quad &G(x,M(x)) \leq 0 \\ &H(x,M(x)) = 0 \\ &x \in X \end{aligned} \tag{1.5}$$

Now if for each $x^* \in X$, $(x^*,M(x^*)) \in \Omega$ then,
$IR = \{(x,M(x)) : G(x,M(x)) \leq 0, H(x,M(x)) = 0, x \in X\} = \{(x,M(x)) : x \in X\}$
In this case the BSP (1.3) can be written as:-

$$\begin{aligned} &\min F(x,M(x)) \\ s.t \quad &x \in X \end{aligned} \tag{1.6}$$

In-addition, if F is continuous on $(x,M(x)), \forall x \in X$ and X is nonempty compact set then the problem in (1.6) attains its optimal solution (Weirstrass Theorem).

Example 1.1. *Consider the linear-BSP, given below:-*

$$\begin{aligned} \min \quad & x + 2y \\ s.t \quad & -1 \leq x \leq 1 \\ & \min \quad -x - y \\ & s.t \quad x + y \leq 1 \\ & \qquad -x + y \leq 1.5 \\ & \qquad -1 \leq y \leq 1 \end{aligned} \tag{1.7}$$

$\forall x \in [-1, 1]$, the followers reaction can be described as follows:-

$$M(x) = \begin{cases} 1.5 + x \ , & for \quad -1 \leq x \leq -0.5 \\ 1 \ , & for \quad -0.5 \leq x \leq 0 \\ 1 - x \ , & for \quad 0 \leq x \leq 1 \end{cases} \tag{1.8}$$

Hence, the linear-BSP in (1.10) is to find an optimal strategy for the leader DM and it can be expressed as:-

$$\min_x F(x, M(x)) = \begin{cases} 3x + 3 \ , & for \quad -1 \leq x \leq -0.5 \\ x + 2 \ , & for \quad -0.5 \leq x \leq 0 \\ -x + 2 \ , & for \quad 0 \leq x \leq 1 \end{cases} \tag{1.9}$$

We can see from (1.9) that, even the simplest case of MSP, the linear-BSP, is a non-convex and non-differentiable optimization problem. The Stackelberg Solution is found at the point $(-1, 0.5)$, with an optimal function values $F(-1, 0.5) = 0$ and $f(-1, 0.5) = 0.5$.

In literature, the linear-BSP has been shown to be NP-hard so that, we don't expect to have a polynomial time exact algorithms for any kind of MSP.

Example 1.2. *Consider the BSP, given below:-*

$$\begin{aligned} \min \quad & x^2 + y^2 + 2|x + y| \\ s.t \quad & -x + y \leq 0 \\ & -1 \leq x \leq 1 \\ & \min \quad |x - y| + 2 \\ & s.t \quad x + y \leq 1 \\ & \qquad -1 \leq y \leq 1 \end{aligned} \tag{1.10}$$

For each given $x \in [-1, 1]$*, the followers reaction can be described as follows:-*

$$M(x) = \begin{cases} x \ , & for \quad -1 \leq x \leq 0.5 \\ 1 - x \ , & for \quad 0.5 \leq x \leq 1 \end{cases} \tag{1.11}$$

Hence, for each $x \in [-1, 1]$ the follower reacts with a unique optimal solution and $(x, M(x)) \in \Omega$.

So the BSP in (1.10), can be reformulated in a similar fashion to (1.6) as follows:-

$$\begin{aligned} \min \quad & x^2 + M^2(x) + 2|x + M(x)| \\ s.t \quad & x \in [-1, 1] \end{aligned} \tag{1.12}$$

which is equivalent to:-

$$\min_x \begin{cases} 2x^2 + 4|x| \ , & for \quad -1 \leq x \leq 0.5 \\ 2x^2 - 2x + 3 \ , & for \quad 0.5 < x \leq 1 \end{cases} \tag{1.13}$$

The stackelberg solution is found at the point $(0, 0)$, with optimal value of the leader's objective function 0 and the follower's objective function 2.

Chapter 2

Literature Review and Scientific Motivation

2.1 Literature Review

Multilevel programming was first proposed by Bracken and McGill [12] to model a decentralized non-cooperative decision system with one leader and multiple followers of equal status in 1973. It was in [14] that first used the designation bilevel and multilevel programming. However, it was not until the early eighties that these problems started receiving the attention they deserve. Motivated by the game theory of H. Stackelberg, several authors studied bilevel programming intensively and contributed to its proliferation in the mathematical programming community [51]. It finds many applications in daily life such as strategic-force planning [11], resource allocation [58], transportation problems [35], engineering design [51] and water regulation [3]. However with all its significance, MPP are hard to solve. The simplest form of MPP, the linear BPP, has been shown to be an NP-hard problem in [7].

To solve the model numerically, many algorithms have been proposed such as extreme point algorithms [13], kth best algorithm [8], [56], [63], branch and bound algorithm [6], [23], [62], descent method [38] and penalty functions method [32], [52], [57]. Parametric optimization approaches has also been proposed to solve $BPPs$ in [19], [26], .

Extreme point search or vertex enumeration algorithms are used to solve linear bilevel problems. In [8], it is shown that a solution of linear BPP must be a vertex or extreme point of the constraint region of the BPP. Motivated by this result, extreme point search algorithms are limited in searching all extremes of the constraint region of the BPP. Different algorithms, such as Kth best algorithms, are proposed to efficiently search the vertices. In [63], Kth best algorithm is used to find a solution for a linear trilevel programming problem. It is clear that enumerating the adjacent extreme points is equivalent to enumerating all the feasible solutions for the decision problem, and based on this statement, the proposed trilevel Kth best algorithm consists of two sub-algorithms. First the simplex algorithm which addresses the problem of determining an optimal solution for the first level linear programming in the constraint region of the trilevel programming problem. The optimal solution of the first level in the constraint region is tested using bilevel kth-best algorithm and if it coincides with the optimal solution of the bilevel problem while fixing the first variable of the solution and ignoring the first level of the TPP, it becomes a global solution to the TPP else, it continues to the next algorithm, i.e, the algorithm for finding the adjacent extreme points of a selected extreme point. The iteration terminates at the optimal solution of the TPP.
In [62] an extended branch and bound algorithm is proposed to solve a linear BPP. The proposed algorithm in the paper highly depends on a theorem which specifies the necessary and sufficient condition to solve a linear BPP. In the theorem a linear BPP has been transfered in to equivalent nonlinear programming problem using KKT conditions. The reformation of the linear BPP is relatively easy to solve because all but one constraint is linear. Omitting or relaxing the nonlinear constraint leaves a standard linear programming that can be solved using simplex algorithm. At each iteration, the proposed algorithm solves the equivalent programming problem ignoring the non-linear constraint (complementary slackness condition). If the solution of the relaxed problem satisfies the nonlinear constraint, the corresponding point is in the inducible region and hence a potential solution to the original problem. If not, a branch and bound scheme is used to implicitly examine all combinations of complementarity slackness.
Descent direction algorithms are proposed for BPP by defining a descent direction, along

which the objective function of BPP is decreased. Once a descent direction is defined for BPP, a positive step size is determined such that an acceptable decrease is obtained in the objective. At the same time the new point obtained by applying the descent direction and the step size must be feasible to the BPP.

In [15], a penalty function algorithm is proposed to solve nonlinear bilevel problems. Assuming the lower level reaction set M(x) is convex with a convex lower level objective function, concave lower level constraint functions and the optimal solution satisfies a constraint qualification condition for the lower level problem, the BPP is transformed into an equivalent one level problem using the KKT conditions. Generally, the transformed problem is non-linear, non-convex and non-differentiable, so that finding a feasible point is nontrivial. To avoid the difficulty of finding an initial feasible point, the proposed algorithm allow the iterate to be infeasible and attempts to minimize the objective function at the same time as moving toward the feasibility. The proposed algorithm uses a penalty function to solve the transformed problem. The penalty function includes a penalty parameter which is used to balance the possibility of conflicting goals of minimizing the objective function and obtaining feasibility, and unscaled, nonnegative penalty terms which takes the magnitude of the violation of the constraint.

Multilevel programming involving fuzzy variable was first proposed by Lai [27] in 1996, and then developed by Shih [46], and Lee [28]. A fuzzy multilevel programming model is proposed in [4], [20], [35], [50], [40].

In [40], the objective function and the DM at each level are transformed in to fuzzy goals by means of assigning an aspiration level to each of them. The optimal solution of each DM when calculated in isolation are considered as the best solution and the associated objective value are considered as the aspiration level of the corresponding fuzzy goal. A linear membership function for each objective function and decision maker is defined to describe the level of satisfaction alloted between 0 and 1. The aim of each DM is to maximize his or her membership function by making them as close as possible to unity by minimizing its negative deviational variables. Therefore, in effect, it will optimize all the objective functions.

Multilevel programming involving random variables was first proposed by Patriksson

and Wynter [39] in 1999. After that, Gao and Liu [21] proposed new stochastic multilevel programming models in 2004. Evolutionary algorithms has been proposed for single level and multiobjective optimization problems in many variant research papers. In the papers [31], [36], [54], [53], [29], solution methods for BSPs, has been proposed by transferring the two level problem into single objective optimization problems and to solve the transformed problems, they use EA as a tool.

In [54], assuming that the lower problem is a differentiable convex programming for each decision variable of the leader problem, a nonlinear bilevel programming problem is transformed in to an equivalent non-linear optimization problem with a single non-differentiable and non-covex objective function. In order to solve the equivalent problem, a specific optimization problem with two objective functions is constructed, such that, a pareto optimal solution for the two objective programming is an optimal solution for the single level programming and thus an optimal solution (if it exist) for the nonlinear BPP. To solve the two objective programming problem, an evolutionary algorithm is proposed including a new constraint handling method to improve the offsprings generated by crossover and mutation operators. The method consists of handling of linear constraints and nonlinear constraints separately to enhance the search ability of algorithm such that it can force the infeasible solution to become feasible solution, improve the quality of individuals and speed the movement of the individuals toward optimal solution.
In [29], assuming the follower has only one solution y for each selected leader's decision maker x, the follower's problem is replaced by its complementary system based on KKT conditions and the bases of the linear systems are encoded as individuals of populations for an evolutionary algorithm. Each randomly selected individual l which corresponds to a bases B is encoded as a binary string and a new fitness function is given, which provides a local search for the variable values of the leader. At first, according to each l, the complementary system is solved and get the complementary basic feasible solution (CBFS), in which the follower's solution $y(x)$ is obtained as a linear function of x. Then, in the BPP y is replaced by y(x), as a result, the leader problem is converted to an optimization problem only involving x. At last, the leader problem is solved to obtain the optimal value F and the value is taken as the fitness of l. Parents are selected

from the individuals according to their fitness values. Crossover (recombination) and mutation operators are applied to the parents and get offspring sets. The fitness value of each offspring is evaluated and the selection is hold from the parents and the offsprings according to their fitness value to select individuals which are going to be included to the next population. Those selected individuals are always less than the randomly selected initial population and the number is filled by selecting randomly from the remaining individuals in the population. The procedure repeats again and again until the termination condition is satisfied.

In the above, we have tried to give short description of some of the algorithms proposed to solve multilevel programming problems, but all of them are done for a special types of problems specified on their assumptions and no one have proposed for a general multilevel problem. An algorithm proposed in [49] seems quite more general-purpose. The algorithm in the paper calls a (1+1)-Evolutionary Strategy (in which a parent gives a birth to one child) to solve a single objective optimization problem several times in such a way that: first the leaders level is solved alone for all the variables in the problem under all the constraint set, as if it controls all the variables. Then by fixing the leader's DM the second level problem is solved alone for all the variables below the level under the common constraints and all the constraints below the level. Generally for any level i, fixing the DMs of 1^{st} to $(i-1)^{th}$ level, the i^{th} level problem is solved alone for all the variables below the level, under the common constraint and all the constraints below the level. Continuing the same way, once the last level problem is solved, the second iteration starts to solve the leader problem with only one variable (the leader's DM) to control, under the leaders constraint and common constraints. This process continue until a termination criterion is fulfilled. But calling evolutionary strategy at every step of movement decreases the efficiency of the algorithm exponentially and it will be hard to find a result even for small problems. Furthermore the convergence of the algorithm, even with large number of initial populations is not guaranteed.

2.2 Scientific Motivation

As we see in the literature review, there is no efficient algorithm proposed for a general MSPs. Most existing algorithms work with BPPs specially with the linear BPPs and BPPs having 'nice' properties of the involved functions, such as continuity, differentiablity, convexity,... etc. But even for such kind of problems, no existing algorithms have been proposed to solve the problems in a polynomial time. Our interest to apply the concept of EAs for a general MSPs, starts from that point of view. We all need to get near optimal solution within acceptable time, instead of waiting an exact solution for thousands of years, in which the problem of our interest may not be feasible at that time. The most significant advantage of EAs over other traditional algorithms is that its non-dependent characteristic on the functions included in optimization problems. The success of EAs to solve complex real world problems in diverse areas have been discussed in different research papers such as [61],[16],[48]. Some authors have tried to apply EAs partially to solve BSPs, so that they only receive partial benefit from it. Until now, no existing algorithms have proposed the whole concept of EAs to MSPs and no one acquires its full benefit. So the huge hierarchical structures of the real world problems which mostly contains unfamiliar functions, the lack of efficient algorithms for general MSPs and the performance of EAs in complex single level optimization problems, motivated us to apply the concept of EAs by some how modifications of the operators in the existing algorithms, for efficiently implementing it to general MSPs.

Chapter 3

Systematic Evolutionary Algorithm for a general Multilevel Stackelberg Problems with bounded decision variables (SEAMSP)

3.1 Description of the Algorithm

The proposed algorithm is designed for efficiently searching a stackelberg solution of MPP having a common constraint set S, that contains the minimum and maximum values of the decision variables.

SEAMSP search stackelberg solutions of a given MPP in a hierarchical way. Within an iteration, the leader DM observes the decision of other's by passing different values of the variable under its control. We call those values, **"spies"** of the leader DM. Similarly for each value passed by the DMs above their level, the other DMs observe the decisions of those below their level by passing their own spies. For each value obtained from above, by recording the results gained from each "spy" of their own, the DMs in the interior of the MPP selects a best spy of their interest (which results a better value of their objective function). The leader DM record the feedbacks of all of its spies and

selects the best of all based on its objective. The best spy of the leader DM with all the hierarchical decisions of the other DMs is placed temporarily in a stock, being the best spies of the iteration. In the next iterations, the above steps are repeated again and again until the requirements of a termination is fulfilled. Best of the best hierarchical spies of the iterations, regarding the leaders objective function are proposed as an approximate stackelberg solution of the MPP.

As the number of spies for each level increases, the probablity of getting better results will increase, but accordingly the number of function evaluations will increase exponentially so that the computational time of the algorithm will not be acceptable. Instead, in SEAMSP each DM send few but **"intelligent spies"** having a nice cooperation with the spies of the other DMs, for representing the whole constraint region in a random, unique, diverse and systematic way. The decision space of a variable is divided in to equal regions of the number of spies, where each spy represents a unique region. Once representing the decision spaces using systematic sampling, the DMs select candidates for a decision from the representatives of their spaces in a hierarchical way. If a spy is selected as a candidate for a decision by a DM of any level below the leader, a mutation operator is applied to search a better representative on the region controlled by the selected one. The best among a candidate (parent) and its offsprings is/are selected by the DM. Every iteration comes with new spies of the leader DM from a region that does not have a representative in the first decision space and tour the same hierarchical paths for a decision. The flow chart in Figure-3.1 shows the procedures followed by the proposed algorithm for solving k-level stackelberg problems. Refer to the next section for the details. The main idea of the proposed algorithm, more or less depends on :-

1. Systematic and Fair distribution of spies on S
2. Action of the leader DM and Reactions of the other DMs
3. Definition of Mutation operator on the reaction sets
4. Selection of initial solution set and adaptation of the environment
5. Maximum iteration and a Tolerance of the initial solutions

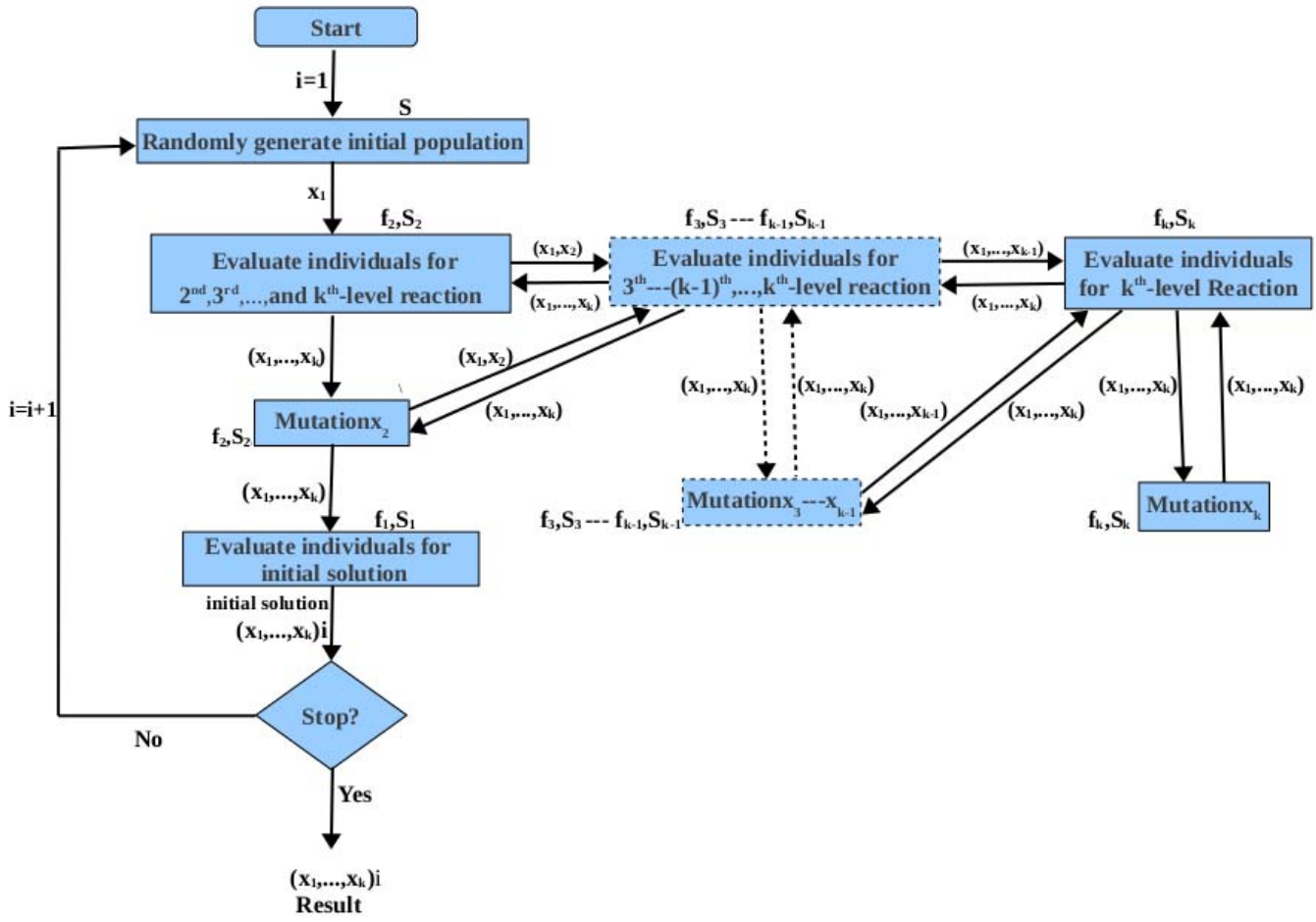

Figure 3.1: SEAMSP flowchart for MSPs

3.2 SEAMSP for a k-level stackelberg problems

Every discussion on MPP of this section considers the minimization of k-level optimization problem defined in the previous chapter.

Input:-

- ▲ Minimum and Maximum values of the j^{th} decision variable x_j:- m_j and M_j respectively, S_j, and f_j, $j = 1, 2, \ldots, k$
- ▲ Vector dimension of the variables, n_1, n_2, ... and n_k respectively

Notations:-

$d_j = M_j - m_j :=$ range of decision space X_j, $j = 1, 2, \ldots, k$

$X_j = [m_j, M_j]^{n_j} :=$ x_j's decision space, $j = 1, 2, \ldots, k$

$X_{ji} := \{x^* : x^*$ is a sample point ("spy") on X_j, at iteration $i\} \subseteq X_j$, $j = 1, 2, \ldots, k$

$2b_j :=$ number of spies on a single dimension space of X_j

$maxiter :=$ maximum number of iterations before termination

$d := \prod_{j=1}^{k} d_j^{n_j}$

$d' := \prod_{j=2}^{k} d_j^{n_j}$

$n := \sum_{j=1}^{k} n_j$

$\pi_j :=$ the number of functions in j-level problem

$floor(a) :=$ the maximum integer which is less than a

Parameters of the algorithm:-

$\alpha :=$ a maximum number of hierarchical spies on $S = X_1 \times \cdots \times X_k$, in one iteration

$\gamma :=$ a minimum number of iterations with nonempty initial solution, if not terminated by $maxiter$

$t(< \gamma) :=$ number of nonempty initial solutions required before a start of computing tolerance of initial solutions of an iteration based on the leader objective function

$\epsilon_1 :=$ tolerance of a reaction, from spies on X_j, $j = 2, 3, \ldots k$

$\epsilon :=$ maximum tolerance, with respect to the leaders problem, of consecutive $(\gamma - t)$ initial solutions of iterations

$\beta_1 :=$ minimum number of spies on X_1 for $d_1 = 1$, if not terminated by tolerance

$\beta_2 :=$ minimum number of iterations, if not terminated by tolerance

1. The maximum number of spies on a single space of X_j, a_j, are selected as follows:- To be fair on distributions of $x_j's$ in a single iteration, the proposed algorithm consider the following k equations :-

$$\prod_{j=1}^{k} a_j^{n_j} \lesssim \alpha \tag{3.1}$$

$$\gamma \left(\frac{a_1}{d_1}\right)^{n_1} - \left(\frac{a_j}{d_j}\right)^{n_j} \approx 0 \ , \qquad where \quad j = 2, 3, ..., k \tag{3.2}$$

Equation (3.1) indicates that the total number of hierarchical spies, $\prod_{j=1}^{k} |X_{ji}|$, on any iteration, i, is less than or equal to α and for a fair distribution of different problems, we recommend α to be directly proportional to d. The number $\left(\frac{a_j}{d_j}\right)^{n_j}$, represents, the maximum density of spies on the region X_j, where $j = 1, 2, \ldots, k$. Equation (3.2) indicates that, $|X_{ji}|$ is directly proportional to d_j, for $j = 1, \ldots, k$. Since each iteration comes with new spies on X_1 and the minimum number of iterations are given to be γ, the density of spies on X_1 (for the same purpose of minimizing the leaders objective function) are multiplied at least by γ times. Once the leader spy is changed, the objective functions of the other levels changes accordingly, hence even if their DMs send new spies on X_j, $j = 2, 3, \ldots, k$ for the next iterations (after the first), they will come-on different objectives for the reason that the decision of those below the leaders level DMs depend on their "ancestors" (the decision of those above their level). So the number γ in equation (3.2) uses to minimize the difference between the density of spies on X_1 and the other decision spaces.

From equations (3.1) and (3.2), we reach to the following approximations:-

$$a_1 \lesssim \left[\frac{\alpha d_1^{n_1(k-1)}}{\gamma^{k-1} d'}\right]^{\frac{1}{n_1 k}} \tag{3.3}$$

$$a_j \lesssim d_j \left[\gamma \left(\frac{a_1}{d_1}\right)^{n_1}\right]^{\frac{1}{n_j}} \ , \quad where \quad j = 2, 3, ..., k \tag{3.4}$$

Now natural numbers $a_j > 1$ are chosen from (3.3) and (3.4), for sufficiently large algorithm parameter α.

▼ Let $b_j = floor\left(\frac{a_j}{2}\right)$, $j = 1, 2, \ldots k$, the step-size of odd and even distributions , δ_j, on X_j is calculated as follows:-

$\delta_j = \frac{d_j}{b_j}$, where the range $[m_j, M_j]$ is divided in to b_j intervals of step-size δ_j and each interval contains two subintervals (containing an odd and even distributed spies on X_j) of step-size $\frac{\delta_j}{2}$. For $j > 1$, from the two subintervals of the first interval two spies are selected randomly and the rest are distributed with a step-size of δ_j from the two randomly selected spies, by using systematic sampling.

▼ To protect repetition of the represented regions on X_1, in the first iteration each subinterval of the first interval is divided into distinct $maxiter$ sub-subintervals of n_1 dimensions, with step size δ_{11}, where $\delta_{11} = \frac{\delta_1}{2maxiter}$. At each iteration n_1 spies are randomly selected from randomly selected unique sub-subintervals of the first subinterval in n_1 dimensions, and distributed with the step size δ_1. The n_1 randomly selected sub-subintervals depend on a random permutation of integers in $[1, maxiter]^{n_1}$. For instance, if a first number in the random permutation is $(c_1, c_2, \ldots, c_{n_1})$, where $1 \leq c_i \leq maxiter$, $i \in \{1, 2, \ldots, n_1\}$, then in the first iteration the c_i^{th} sub-subinterval of the first subinterval and the $(maxiter - c_i + 1)^{th}$ sub-subinterval of the second subinterval, in the i^{th} dimension, are selected from the first interval and distributed by systematic sampling with a step size of δ_1, $\forall i \in \{1, 2, \ldots, n_1\}$.

Figures-3.2,3.3,3.4 and 3.5, demonstrates possible distributions and density of spies on X_j, $j > 1$, and on X_1 in different iterations.

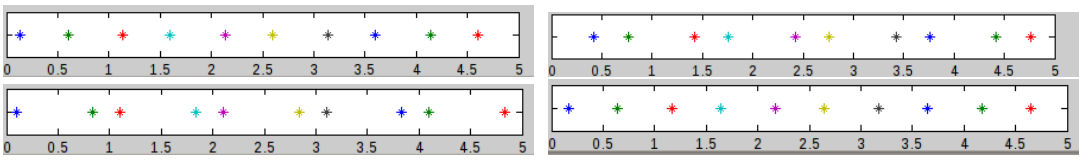

Figure 3.2: Possible spies on $X_j = [0, 5]$, $b_j = 5$, $j > 1$, in iterations 1 to 4 resp.

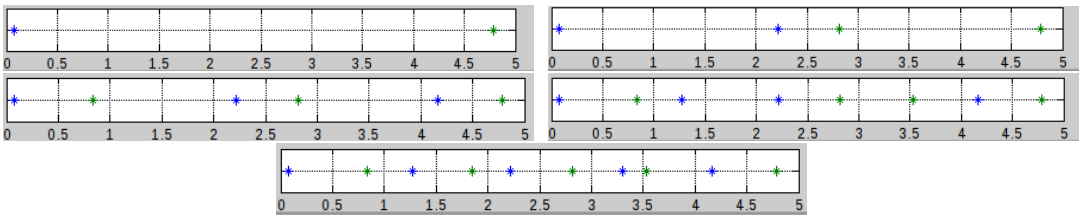

Figure 3.3: Possible spies on $X_1 = [0, 5]$, $b_1 = 1$, $maxiter = 5$, in iterations 1 to 5 resp.

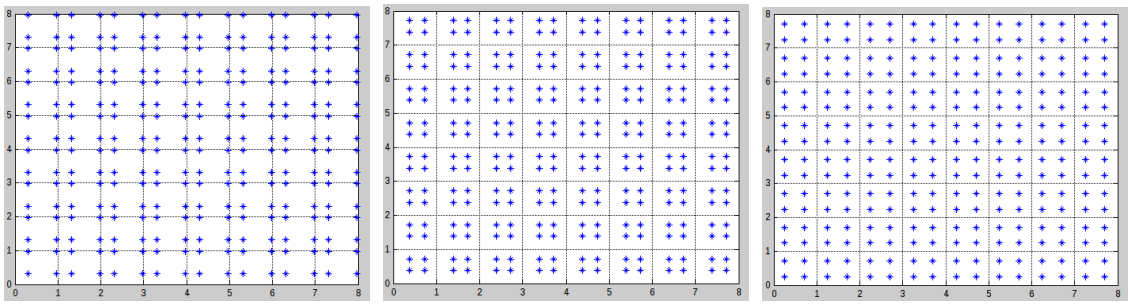

Figure 3.4: Possible spies on $X_j = [0, 8]^2$, $b_j = 8$, $j > 1$, in iterations 1 to 3 resp.

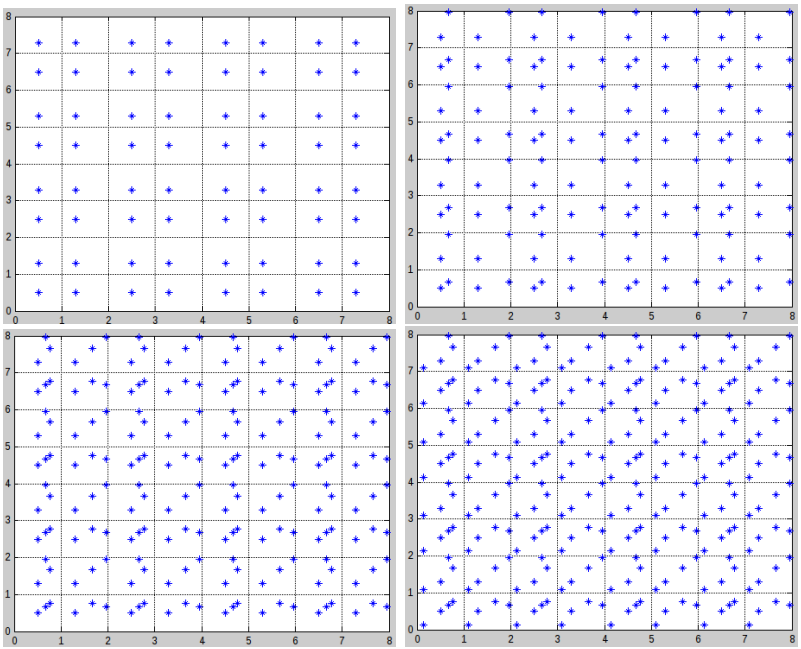

Figure 3.5: Possible spies on $X_1 = [0, 8]^2$, $b_1 = 4$, $maxiter = 4$, in iterations 1 to 4 resp.

Figures-3.2,3.3,3.4 and 3.5, show the possible distributions on the given decision spaces of dimension one and two. The distributions on X_1 increase its density (of the same purpose) with the number of iterations. We can see from Figures-3.3 and 3.5 that, the newcomers on X_1 control the unrepresented sub-regions of X_1 at each iteration. We can also observe from the figures that, for any a_j $(j > 1)$ which are related with a_1 by equations (3.3) and (3.4), the number of representatives on X_1 and the other decision spaces becomes equivalent at the end of the fixed iterations. Any further iterations (if larger *maxiter* is given) increase the control capacity of the spies on decision space X_1 with out any repetition.

The above distributions are designed to fill the decision spaces fairly and to tackle any objective functions, including the most difficult once, where the function values decline suddenly within a short range of the variables. The selection of spies in any other decision spaces are done in a similar fashion. At each iteration the DMs at different level send new representatives of their region. The selection of new representatives at every iterations in the reactions, helps to minimize the missing of very important points from the decision variables of the levels below the leader level. Similarly the selection of new spies for the decision variable x_1 minimizes the missing of important points. In-addition to this we may not be certain to get good points in one iteration but in a number of iterations by improving the initial solutions obtained from each iteration we can reach some how to believable result in which the improvement of the initial solutions would be negligible in the next iterations.

2. After the representatives are chosen, every spy on X_1 is sent to search good reactions from the spies on X_j, where $j > 1$.

   ▶ In general, finding stackelberg solutions in a reasonable number of time for most multilevel problems is not a simple task and finding approximate stackelberg solution in MPP becomes hard due to a missing of important points for the problem in the search of reactions. For each given ancestors $(x_1, \ldots, x_i)$ from a distribution on $X_1 \times \cdots \times X_i$, to find approximate rational reaction sets from the distributions on $X_{i+1} \times \cdots \times X_k$, if we only consider the reactions with a full interest of the

j-level ($j > i$) problems, we possibly fail to cache points (with a slight deficit from the value interested by the j-level problems in the given distributions on X_j's) but drastically needed by those above the $j-$level problems.

▶ To protect such a missing in SEAMSP, we define a tolerance of reactions (depending on the function value) such that all points having the levels objective function values within the given tolerance are considered as good reactions and hence candidates for a reaction in the respected level.

Example 3.1. *Fixing* $(x_1^*, \ldots, x_{j-1}^*)$, *if* $f_j(x_1^*, \ldots, x_{j-1}^*, x_j^*, \ldots, x_k^*) = v_j$ *denotes the objective function values at the hierarchical decisions from the representatives on* $X_j \times \cdots \times X_k$, *then for a given algorithm parameter* $\epsilon_1 > 0$, *the points* $(x_i^{'}, \ldots, x_k^{'})$ *on* $X_i \times \cdots \times X_k$ *such that,* $(x_1^*, \ldots, x_{i-1}^*, x_i^{'}, \ldots, x_k^{'}) \in S_i$, $j \leq i \leq k$ *and,*

$$f_i(x_1^*, \ldots, x_{i-1}^*, x_i^{'}, x_{i+1}^*, \ldots, x_k^*) \leq v_i + \epsilon_1(1 + |f_i(x_1^*, \ldots, x_{i-1}^*, x_i^{'}, x_{i+1}^* \ldots, x_k^*)|) \tag{3.5}$$

are considered to be good reactions of $(x_1^*, x_2^*, \ldots, x_{i-1}^*)$. *The selection of approximate reactions in the distribution of* $(x_i, x_{i+1}, \ldots, x_k)$, *depends on a tolerance parameter, and the value of the function at a given point in a sense that, an approximate values of absolutely large numbers may have large distance from the number, than approximate values of absolutely small numbers have. (For instance,* 1000 *approximate to* 999 (*with fixed* $\epsilon_1 = 0.001$), *doesn't mean that* 2 *approximate to* 1 (*with fixed* $\epsilon_1 = 0.001$).

In a normal situation, where for every ancestor $(x_1^*, \ldots, x_{j-1}^*)$, $1 < j \leq k$, each $j-$level problem reacts uniquely from the representatives in X_j, the search of candidates for a rational reaction set by the j-level problems ($j > 1$) evaluates each function in the $j-$level problem (objective function and constraint functions) for $\prod_{i=1}^{j}(2b_i)^{n_i}$ times. Hence, the total function evaluations for the candidates of rational reaction sets is :-

$$\sum_{j=2}^{k} \pi_j \left(\prod_{i=1}^{j} (2b_i)^{n_i} \right), \tag{3.6}$$

where π_j denotes the number of functions in the $j-$level problem.

3. Each spy on X_j, $1 \leq j < k$, is sent to secretly watch the reactions of the DMs below its level and all DMs in $i-$level, $j < i \leq k$ use their maximum effort to react rationally based on their level problem by ignoring the hidden intension of those above their levels. For each fixed ancestors $(x_1, \ldots, x_j)$, $1 \leq j < k$, the first candidates for reactions by the DMs in $i-$level, $j < i \leq k$, are selected from the representatives on X_i. Once a spy, λ_i, on X_i is selected as a candidate, the $i-$level DM focus its search around the region controlled by λ_i and a mutation operator is applied for the betterness of the $i-$level problem. Every candidate of a reaction on X_i are assumed to be parents and mutated to find better representatives of the region bounded by the movement scale of the operator. The fitness of an offspring ϱ_j in level$-j$, when ordered with the decisions of the other levels, ϱ, is evaluated based on f_j and $S \cap S_j$. If $\varrho \in S \cap S_j$ then the value $f_j(\varrho)$ is compared with the f_j value, v_j, at the original hierarchical decision and the defined tolerance in equation-(3.5). The search area of offsprings at level$-j$ increase with dimension of the decision space, X_j, and hence the number of offsprings growth with the dimension (up to 2^{n_j-1}) depending on the fulfillment of the requirements (constraint region) of the j-level problem. In the normal situation of (3.6), the total number of parents in j-level problem, $j \in \{2, \ldots, k\}$, is $\prod_{i=1}^{j-1}(2b_i)^{n_i}$. Hence, the total function evaluations by the mutation operator is :-

$$\sum_{j=2}^{k} \left(\pi_j 2^{n_j-1} \prod_{i=1}^{j-1} (2b_i)^{n_i} \right) \tag{3.7}$$

Around good points of most functions, there could be another good points. Similarly, around good reactions of MSPs there could be another good reactions. A mutation operator of j-level problem, search reactions around a known good reactions. But a good reaction is not necessarily good for MSPs, because the final decision of MSP is made by the leaders DM and reactions doesn't consider the leaders problem. What makes mutation of good reactions in j-level problem important for approximation of stackelberg solutions is the assumption that:-

- Offsprings (after mutation of a parent in j-level problem) may continue to be good (or better) for $j-$level problem.

♦ Parents (which are assumed to be good reactions of the j-level) may not be feasible for an i-level problem (where $i < j$), but their offsprings probably results a drastic change in the $i-$level problems.

Mutation operator of SEAMSP is applied to all parents by considering the following things:-

- random movements of the parents
- against any repetition
- coverage of the large unrepresented region around the parent

Consider a parent

$$\lambda_j = \begin{pmatrix} x(1,i) \\ . \\ . \\ . \\ x(n_j,i) \end{pmatrix} \in R^{n_j}, \qquad i \in \{1,2,...,(2b_j)^{n_j}\}, \quad 2 \leq j \leq k$$

i denotes the order of λ_j in the distribution of the spies on n_j dimension of X_j. Before we apply a mutation operator on each parent λ_j, the movement scale of the operator, δ, is calculated as:

$\star$ $r_1 = x(n_j, 2) - x(n_j, 1)$;

(nonrepresented region of the $n_j's$ direction among consecutive odd distributed spy and even distributed spy)

$\star$ $r_2 = \delta_j - r_1$;

(nonrepresented region of the $n_j's$ direction among consecutive even distributed spy and odd distributed spy)

$\star$ $\delta = max(r_1, r_2)$;

(a) If $\delta = r_1 > 2 * r_2$ / $\delta = r_2 > 2 * r_1$ then the mutation operator search points to the right / left direction of $x(n_j, i)$, for an odd number i and to the left / right direction of $x(n_j, i)$, for an even number i as follows:-

i. For odd distributed parents (i.e when i is odd number)

If $n_j > 1$

$$\lambda_j^{mut_m} = \begin{pmatrix} x(1,i) + (A(m,1) * \delta * rand(1)) \\ x(2,i) + (A(m,2) * \delta * rand(1)) \\ . \\ . \\ . \\ x(n_j - 1, i) + (A(m, n_j - 1) * \delta * rand(1)) \\ x(n_j, i) \pm \frac{\delta}{2} * rand(1) \end{pmatrix}_{m \in \{1,2,....,2^{n_j - 1}\}}$$

If $n_j = 1$

$$\lambda_1^{mut} = \left(\lambda_j \pm \tfrac{\delta}{2} * rand(1)\right)$$

ii. For even distributed parents (i.e when i is even number)

If $n_j > 1$

$$\lambda_j^{mut_m} = \begin{pmatrix} x(1,i) + (A(m,1) * \delta * rand(1)) \\ x(2,i) + (A(m,2) * \delta * rand(1)) \\ . \\ . \\ . \\ x(n_j - 1, i) + (A(m, n_j - 1) * \delta * rand(1)) \\ x(n_j, i) \mp \frac{\delta}{2} * rand(1) \end{pmatrix}_{m \in \{1,2,....,2^{n_j - 1}\}}$$

If $n_j = 1$

$$\lambda_1^{mut} = \left(\lambda_j \mp \tfrac{\delta}{2} * rand(1)\right)$$

(b) If $r_2 \leq \delta = r_1 \leq 2 * r_2$ or $r_1 \leq \delta = r_2 \leq 2 * r_1$ then

If $n_j > 1$

$$\lambda_j^{mut_m} = \begin{pmatrix} x(1,i) + (A(m,1) * \delta * rand(1)) \\ x(2,i) + (A(m,2) * \delta * rand(1)) \\ . \\ . \\ . \\ x(n_j - 1, i) + (A(m, n_j - 1) * \delta * rand(1)) \\ x(n_j, i) + \frac{\delta}{4} * (rand(1) - rand(1)) \end{pmatrix}_{m \in \{1,2,....,2^{n_j - 1}\}}$$

If $n_j = 1$

$$\lambda_1^{mut} = \left(\lambda_j + \frac{\delta}{4} * (rand(1) - rand(1))\right)$$

, where A is a $(2^{n_j - 1}) \times (n_j - 1)$ matrix having a permutation of rows of entries 1 and -1, (i.e $A = rowexch(n_j - 1, 2^{n_j - 1})$). The formulation of matrix A assures the uniqueness of the offsprings of a parent. $rand(1)$ denotes for any random number between 0 and 1.

The above mutation operation search unique points near the parent biased to the largest unrepresented region near $x(n_j, i)$. In two dimensional space, we can see the three cases of δ, (the two cases in (a) and the third one of (b)), from Figure-3.4.

Example 3.2. *Let $x(n_j, i) \in [0, 5]$, $\delta_j = 1$ and $a_j = 10$, then if we apply the mutation operator defined above to all representatives of the distribution on X_j, a possible movements of $x(n_j, i)$ are shown in the figures below:*

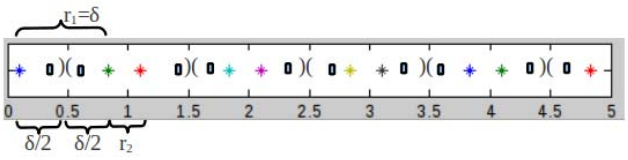

Figure 3.6: $\delta = r_1 > 2 * r_2$

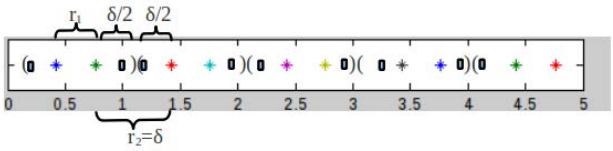

Figure 3.7: $\delta = r_2 > 2 * r_1$

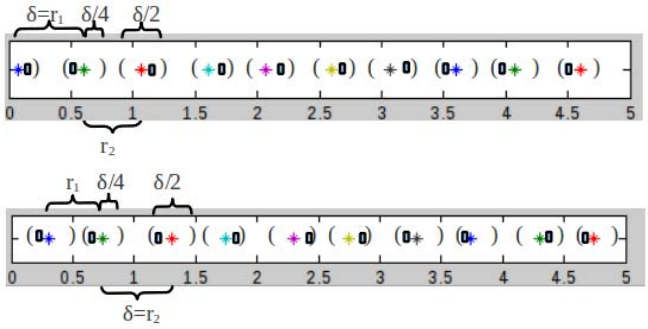

Figure 3.8: $r_2 \leq \delta = r_1 \leq 2 * r_2$ and $r_1 \leq \delta = r_2 \leq 2 * r_1$, resp.

where the rectangular points indicate possible movements of $x(n_j, i)$ *after mutation, the left and right open brackets indicate the maximum movements of the offsprings.*

Generally offsprings of a parent are reproduced from distinct and equal rectangular neighbors (line-segments, rectangles, rectangular-boxes, ...etc) of their parents. To cover the search spaces the operator uses random movements in the restricted regions.

4. From all the good reactions (after mutation), those satisfying the leaders constraint region, S_1, are selected to be members of approximate inducible region and become candidates for initial stackelberg solutions in a given iteration.
   The initial solution set is selected from all points in the inducible region, based on the interest of the leaders problem. For a better adaptation of the environment, the initial solution is updated at any iteration having a better leaders objective function value. Under the normal situation of (3.6) and if all the reactions are in S_1, the total function evaluations for initial solution set is $\pi_1(2b_1)^{n_1}$. Hence, under

these assumptions the total function evaluations in one iteration is:-

$$\sum_{j=1}^{k} \pi_j \left(\prod_{i=1}^{j} (2b_i)^{n_i} \right) + \sum_{j=2}^{k} \left(\pi_j 2^{n_j - 1} \prod_{i=1}^{j-1} (2b_i)^{n_i} \right) \tag{3.8}$$

5. Some times, specially if there is no an open ball subset of the constraint region Ω (eg. if all members of Ω are discrete or Ω is contained in a hyperplane of R^n) then only the very luckiest hierarchical representatives of the decision spaces or very luckiest mutations will get entrance to the inducible region. Searching these luckiest points may result in an infinite iterations and hence we need to bound the number of iterations with finite number.

   For a better approximation of solutions, problems with large range of decision variables need more iterations or large number of initial representatives.

   $*$ Maximum iteration is calculated by considering the above statement as:

$$maxiter = max(\beta_1 \left(\tfrac{d_1}{2b_1} \right)^{n_1}, \beta_2),$$

   where β_1 and β_2 are the algorithm parameters which are fixed by considering the following statement. If the algorithm does not terminate with tolerance value then, β_1 represents the minimum number of spies on X_1 for a unit range of the decision space X_1 and β_2 represents the minimum number of iterations, required before termination. $\left(\frac{2b_1}{d_1} \right)^{n_j}$ describes the density of the distribution of representatives of X_1 on the given range of decision space, d_1.

   $*$ The tolerance of the algorithm starts to calculate after some fixed t non-empty initial stackelberg solutions and determine the tolerances of two consecutive initial stackelberg solutions for every $\gamma - t$ iterations. Each tolerance of the $\gamma - t$ iterations are determined in the same sense as in equation-(3.5), where f_i and v_i, are replaced by initial leaders objective function value at iteration i (i.e. $LV_i = f_1(x_1^i, ..., x_k^i)$, where $(x_1^i, ..., x_k^i)$ is an initial solution at iteration i) and initial leaders objective function value at iteration $i+1$ (i.e. $LV_{i+1} = f_1(x_1^{i+1}, ..., x_k^{i+1})$, where $(x_1^{i+1}, ..., x_k^{i+1})$ is an initial solution at iteration $i+1$), respectively as:-

$$\epsilon^i = \frac{LV_i - LV_{i+1}}{1 + |LV_i|}$$

The program terminates whenever the number of iterations reach the *maxiter* or $tol^{(p)} < \epsilon$, where

$$tol^{(p)} = \sum_{q=t+(\gamma-t)(p-1)+1}^{t+(\gamma-t)p} \epsilon^q \qquad , for \quad p = 1, 2, 3, \ldots$$

Hence, the exact number of iterations before termination is :-

$$I = \min(maxiter, \Delta),$$

where $\Delta = t + (\gamma - t)r$, $r = min\{p : tol^{(p)} < \epsilon\}$ and the minimum number (γ) holds at $r = 1$.

In general:-

$$\lim_{\epsilon=0,\alpha\to\infty} AS = \lim_{\epsilon=0,a_i\to\infty} AS = SS,$$

but sampling infinite numbers is impractical.

Tolerance of a solution and *maxiter* are used to tackle the problem in a practical and acceptable manner. If an initial solution by the algorithm satisfies the tolerance definition (i.e the initial solutions doesn't improve much for several iterations), then the algorithm accepts the last initial solution as an approximation to the desired solution (stackelberg solution). The acceptance of the last initial solution (which satisfy the tolerance definition) as an approximation depends on a believe that:-

♣ The advantage (interms of time) of exiting from the iterations is more visible than the advantage of getting better results on a move to next iterations.

Table 3.1: The proposed algorithm is summarized in the following table:

Step	Description
1	Consider the MPP in (1.1) with bounded decision variables, Input:- $n_j, m_j, M_j, S_j, f_j, \forall j \in \{1,\ldots,k\}$ Algorithm Parameters:- $\alpha, \gamma, t, \beta_1, \beta_2, \epsilon, \epsilon_1$
2	$\forall j \in \{1,\ldots,k\}$, $r \in \{1,\ldots,n_1\}$ and $p \in \{2,\ldots,k\}$, set:- $d_j = M_j - m_j$, $d' = \prod_{j=2}^{k} d_j^{n_j}$, $a_1 = floor\left(\left[\frac{\alpha d_1^{n_1(k-1)}}{\gamma^{k-1}d'}\right]^{\frac{1}{n_1 k}}\right)$, $a_p = floor\left(d_p\left[\gamma\left(\frac{a_1}{d_1}\right)^{n_1}\right]^{\frac{1}{n_p}}\right)$ $b_j = floor\left(\frac{a_j}{2}\right)$, $c = ceil\left(\beta_1\left(\frac{d_1}{2b_1}\right)^{n_1}\right)$, $maxiter = max(c, \beta_2), i = 1$, $\delta_j = \frac{d_j}{b_j}$, $\delta_{11} = \frac{\delta_1}{2maxiter}$, $perm_r = randperm(maxiter)$, $X_{ji} = []$
3	At each iteration, i, generate a random initial population for each decision variable from two radomly selected points and step-size δ_j, as shown below:- for $r = 1 : n_1$ $x_1^r(1,1) = m_j + (perm_r(1,i) - 1)\delta_{11} + rand(1)\delta_{11}$ $x_1^r(1,2) = m_1 + \frac{\delta_1}{2} + (maxiter - perm_r(1,i))\delta_{11} + rand(1)\delta_{11}$ for $q = 3 : 2 : 2b_1 - 1$ $x_1^r(1,q) = x_1^r(1,q-2) + \delta_1$ $x_1^r(1,q+1) = x_1^r(1,q-1) + \delta_1$ end for $z_1(r,:) = x_1^r(1,:)$ end for for j=2:k for $r = 1 : n_j$ $x_j^r(1,1) = m_j + rand(1)\frac{\delta_j}{2}$ $x_j^r(1,2) = m_j + \frac{\delta_j}{2} + rand(1)\frac{\delta_j}{2}$ for $q = 3 : 2 : 2b_j - 1$ $x_j^r(1,q) = x_j^r(1,q-2) + \delta_j$ $x_j^r(1,q+1) = x_j^r(1,q-1) + \delta_j$ end for end for $z_j(r,:) = x_j^r(1,:)$ for $r = 1 : n_j$ $X_{ji} = [X_{ji}; reshape(repmat([repmat(z_j(r,:)', 1, (2b_j)^{n_j - r})]', 1, (2b_j)^{r-1}), 1, (2b_j)^{n_j})]$ end for end for
4	For each $(x_1^*,\ldots,x_{j-1}^*,x_j,\lambda_{j+1}^{*mut},\ldots,\lambda_k^{*mut})$, select parents λ_j from all $x_j \in X_{ji}$, where $f_j(x_1^*,\ldots,x_{j-1}^*,\lambda_j,\lambda_{j+1}^{*mut},\ldots,\lambda_k^{*mut}) \le f_j(x_1^*,\ldots,x_{j-1}^*,x_j,\lambda_{j+1}^{*mut},\ldots,\lambda_k^{*mut}) + \epsilon_1 v$ where $v = 1 + \lvert f_j(x_1^*,\ldots,x_{j-1}^*,\lambda_j,\lambda_{j+1}^{*mut},\ldots,\lambda_k^{*mut})\rvert$, and $(x_1^*,\ldots,x_{j-1}^*,\lambda_j,\lambda_{j+1}^{*mut},\ldots,\lambda_k^{*mut}) \in S_j$
5	Find offsprings, λ_j^*, as described on pages 25-28, and select the best of a parent and offsprings, λ_j^{*mut}.
6	$\forall x_1 \in X_{1i}$ select $x_1^i \in X_{1i}$ where $f_1(x_1^i,\lambda_2^{imut},\ldots,\lambda_k^{imut}) \le f_1(x_1,\lambda_2^{mut},\ldots,\lambda_k^{mut})$, initial solution $(x_1^i,\lambda_2^{imut},\ldots,\lambda_k^{imut}) \in S_1$ and update the initial solution based on f_1
7	i=i+1, terminate if the termination criteria on pages 29-30 is fullfilled else go to step-3
8	**Output:-** $(x_1^i,\lambda_2^{imut},\ldots,\lambda_k^{imut})$

Chapter 4

Approximation of Stackelberg Solutions

Due to the conflicting behavior of the objectives in multilevel problems, specially Stackelberg problem, it is not easy to compare two approximate results by just looking at the values of the objective functions and the constraint region.
To compare approximate stackelberg solutions of MPP, one needs to consider the following three basic properties of the solutions:

1. the feasibility of the results to the given MPP
2. the approximation of the results to the reaction sets
3. the fitness of the results, on evaluation of the leader's objective function

While using probabilistic method to solve optimization problems, we don't expect to get exact solutions, instead the algorithms try to approach a solution in a finite time. But how much the result obtained approaches the exact solution is always under a question mark.
Most existing algorithms compare their results by just looking the numerical values of the objective functions and the constraint region which is generally not correct for approximate stackelberg solutions, since all the objective function values at a non-stackelberg

solution may even be better than the same function values at a known stackelberg solution. In other words, Stackelberg solution doesn't obey pareto optimality.
To show this, consider the following problem:-

$$\begin{aligned} \min_{-10\leq x\leq 10} F(x,y) &= |x| - y^3 \\ s.t. \quad & x - y \leq 0 \\ & \min_{-10\leq y\leq 10} f(x,y) = |y| - x^3 \\ & s.t. \quad x + y \leq 10 \end{aligned} \tag{4.1}$$

We can find Stackelberg solution of the above problem by just applying the definition of the BSP directly. For each fixed leader's decision variable, $x \in [-10, 10]$, it is clear that the reaction of the followers problem is $y = 0$. Therefore, the DM at the leader level selects its best strategy, i.e $x = 0$. Hence, $(0, 0)$ is a stackelberg solution for the given problem and the objective function values at the optimal solution are both *zero*. We can see that the point $(5, 5)$ is also located in the constraint region of the problem and the objective function values at the point are both -120, which is numerically much better than the values at the stackelberg solution.

Eventhough there is no common agreement on what an approximate solution to MSPs mean, according to the definition of a stackelberg solution, in this chapter we have tried to define different terms in a logical manner.

4.1 (ε, δ) - Approximation and Comparison Methods

Definition 4.1. *A given point $(x_1^*, \ldots, x_k^*) \in \Omega$ is said to be an (ε, δ)-**approximate stackelberg solution** to the MPP in (1.1), where $\delta, \varepsilon \geq 0$, if :-*

1. $f_1(x_1^*, \ldots, x_k^*) \leq f_1(x_1', \ldots, x_k') + \varepsilon(1 + |f_1(x_1^*, \ldots, x_k^*)|)$, $\forall (x_1', \ldots, x_k') \in IR$
2. $f_j(x_1^*, \ldots, x_k^*) \leq f_j(x_1^*, \ldots, x_{j-1}^*, x_j, \ldots, x_k) + \delta(1 + |f_j(x_1^*, \ldots, x_k^*)|)$,
   $\forall j \in \{2, 3, \ldots, k\}, \quad (x_j, \ldots, x_k) \in M(x_1^*, \ldots, x_{j-1}^*)$
   *(a point $(x_1^*, \ldots, x_k^*)$ satisfying the second condition is called **in a δ-reaction**)*

Definition 4.2. *A point* $(x_1^*, \ldots, x_k^*) \in \Omega$ *is said to be* ***first-rank better*** *(or simply better) than* $(x_1^{'}, \ldots, x_k^{'})$ *if either* $(x_1^{'}, \ldots, x_k^{'}) \notin \Omega$*, or*

- $\forall(\varepsilon_1, \delta_1)$ *such that* $(x_1^{'}, \ldots, x_k^{'})$ *is an* $(\varepsilon_1, \delta_1)$*-approximate stackelberg solution,* $\exists(\varepsilon, \delta)$ *such that* $\varepsilon \leq \varepsilon_1$*,* $\delta < \delta_1$ *(* or $\varepsilon < \varepsilon_1$ *,* $\delta \leq \delta_1$*) and* $(x_1^*, \ldots, x_k^*)$ *is an* (ε, δ)*-approximate stackelberg solution,*

holds true.

Definition 4.3. *A point* $(x_1^*, \ldots, x_k^*) \in \Omega$ *is said to be* ***second-rank better*** *than* $(x_1^{'}, \ldots, x_k^{'})$ *if either* $(x_1^{'}, \ldots, x_k^{'}) \notin \Omega$*, or*

- $\forall(\varepsilon_1, \delta_1)$ *such that* $(x_1^{'}, \ldots, x_k^{'})$ *is an* $(\varepsilon_1, \delta_1)$*-approximate stackelberg solution,* $\exists(\varepsilon, \delta)$ *such that* $\varepsilon + \delta < \varepsilon_1 + \delta_1$ *and* $(x_1^*, \ldots, x_k^*)$ *is an* (ε, δ)*-approximate stackelberg solution,*

holds true.

Definition 4.4. *A point* $(x_1^*, \ldots, x_k^*) \in \Omega$ *is first-rank better (or second-rank better) than a set H, if it is first-rank (or second-rank) better than each element in H.*

Proposition 4.1.

1. *Any stackelberg solution is first-rank better than a non-stackelberg point and any point can never be second-rank better than a stackelberg solution.*

2. *(****Transitivity****) If a point* $(x_1^*, \ldots, x_k^*)$ *is first-rank (or second-rank) better than* $(x_1^{'}, \ldots, x_k^{'})$ *and* $(x_1^{'}, \ldots, x_k^{'})$ *is first-rank (or second-rank) better than* $(x_1^{''}, \ldots, x_k^{''})$*, then* $(x_1^*, \ldots, x_k^*)$ *is first-rank (or second-rank) better than* $(x_1^{''}, \ldots, x_k^{''})$*.*

3. *first-rank better* $\Rightarrow$ *second-rank better, but not the converse*

4. *If a point* $(x_1^*, \ldots, x_k^*)$ *is first-rank (or second-rank) better than* $(x_1^{'}, \ldots, x_k^{'})$ *and* $(x_1^{'}, \ldots, x_k^{'})$ *is second-rank (or first-rank) better than* $(x_1^{''}, \ldots, x_k^{''})$*, then* $(x_1^*, \ldots, x_k^*)$ *is second-rank (but not necessarily first-rank) better than* $(x_1^{''}, \ldots, x_k^{''})$*.*

Proof.

1. Let $(x_1^*, \ldots, x_k^*)$ be a stackelberg solution and $(x_{1*}, \ldots, x_{k*})$ be non-stackelberg point
   $\Rightarrow (x_1^*, \ldots, x_k^*)$ is $(0,0)$-approximate solution and $\forall \varepsilon, \delta \geq 0$, if $(x_{1*}, \ldots, x_{k*})$ is a (ε, δ)-approximate solution then $\varepsilon > 0$ or $\delta > 0$ (or both $\varepsilon > 0$ and $\delta > 0$), hence the result

2. If $(x_1^{''}, \ldots, x_k^{''}) \notin \Omega$, it is clear.
   Let $(x_1^{''}, \ldots, x_k^{''}) \in \Omega$, let $(\varepsilon_1, \delta_1)$, $(\varepsilon_2, \delta_2)$ and $(\varepsilon_3, \delta_3)$ denote the minimum possible values representing the (ε, δ)-approximations of $(x_1^*, \ldots, x_k^*)$, $(x_1^{'}, \ldots, x_k^{'})$ and $(x_1^{''}, \ldots, x_k^{''})$, respectively.
   $\Rightarrow \varepsilon_1 < \varepsilon_2 \leq \varepsilon_3$, $\delta_1 \leq \delta_2 < \delta_3$/ $\varepsilon_1 \leq \varepsilon_2 < \varepsilon_3$, $\delta_1 < \delta_2 \leq \delta_3$ (or $\varepsilon_1 + \delta_1 < \varepsilon_2 + \delta_2 < \varepsilon_3 + \delta_3$)
   $\Rightarrow \varepsilon_1 < \varepsilon_3$, $\delta_1 < \delta_3$ (or $\varepsilon_1 + \delta_1 < \varepsilon_3 + \delta_3$), hence the result

3. Let $(x_1^*, \ldots, x_k^*)$ be first-rank better than $(x_1^{'}, \ldots, x_k^{'})$, where $(\varepsilon_1, \delta_1)$ and $(\varepsilon_2, \delta_2)$ denote the minimum possible values representing the (ε, δ)-approximations of $(x_1^*, \ldots, x_k^*)$ and $(x_1^{'}, \ldots, x_k^{'})$, respectively.
   $\Rightarrow \varepsilon_1 < \varepsilon_2$, $\delta_1 \leq \delta_2$ or $\varepsilon_1 \leq \varepsilon_2$, $\delta_1 < \delta_2$
   $\Rightarrow \varepsilon_1 + \delta_1 < \varepsilon_2 + \delta_2$,
   $\Rightarrow (x_1^*, \ldots, x_k^*)$ is second-rank better than $(x_1^{'}, \ldots, x_k^{'})$
   To show that the converse is not necessarily true, consider the cases where

   $$\varepsilon_1 = 0.01, \varepsilon_2 = 0.02, \delta_1 = 0.001 \text{ and } \delta_2 = 0.0005$$

   hence the result

4. If $(x_1^{''}, \ldots, x_k^{''}) \notin \Omega$, it is clear.
   Let $(x_1^{''}, \ldots, x_k^{''}) \in \Omega$, let $(\varepsilon_1, \delta_1)$, $(\varepsilon_2, \delta_2)$ and $(\varepsilon_3, \delta_3)$ denote the minimum possible values representing the (ε, δ)-approximations of $(x_1^*, \ldots, x_k^*)$, $(x_1^{'}, \ldots, x_k^{'})$ and $(x_1^{''}, \ldots, x_k^{''})$, respectively.
   From 3. and 2., it is clear that, $(x_1^*, \ldots, x_k^*)$ is second-rank better than $(x_1^{''}, \ldots, x_k^{''})$,

to show that $(x_1^*, \ldots, x_k^*)$ is not necessarily first-rank better than $(x_1^{''}, \ldots, x_k^{''})$ in both cases, consider situations where
$\delta_1 = 0.1$, $\delta_2 = 0.2$, $\delta_3 = 0.5$, $\varepsilon_1 = 0.1$, $\varepsilon_2 = 0.15$, $\varepsilon_3 = 0.08$ (or $\delta_1 = 0.1$, $\delta_2 = 0.2$, $\delta_3 = 0.5$, $\varepsilon_1 = 0.1$, $\varepsilon_2 = 0.05$, $\varepsilon_3 = 0.08$) , hence the result

□

Proposition 4.2. *If* $f_1(x_1^*, \ldots, x_k^*) < f_1(x_1^{'}, \ldots, x_k^{'})$ *(or* $f_1(x_1^*, \ldots, x_k^*) \leq f_1(x_1^{'}, \ldots, x_k^{'})$*) , and* $\forall \delta_2$ *such that* $(x_1^{'}, \ldots, x_k^{'})$ *is in a* δ_2*-reaction,* $\exists \delta_1 \leq \delta_2$ *(or* $\exists \delta_1 < \delta_1$*), such that* $(x_1^*, \ldots, x_k^*)$ *is in a* δ*-reaction, then* $(x_1^*, \ldots, x_k^*)$ *is first-rank better than* $(x_1^{'}, \ldots, x_k^{'})$.

Proof. Let ε_1 and ε_2 denotes the minimum possible values representing the (ε, δ)-approximations of $(x_1^*, \ldots, x_k^*)$ and $(x_1^{'}, \ldots, x_k^{'})$, respectively.
$f_1(x_1^*, x_2^*, \ldots, x_k^*) < f_1(x_1^{'}, \ldots, x_k^{'})$ (or $f_1(x_1^*, x_2^*, \ldots, x_k^*) \leq f_1(x_1^{'}, \ldots, x_k^{'})$)
$\Rightarrow \varepsilon_1 < \varepsilon_2$ $(\varepsilon_1 \leq \varepsilon_2)$
$\varepsilon_1 + \delta_1 < \varepsilon_2 + \delta_2$ and hence the result □

4.2 (ε, δ)-convergence of SEAMSP

Convergence of evolutionary algorithms for single-level and multiobjective optimization problems has been studied in different research papers of Rudolph and others, such as in [47], [22], [42],[43], [44] and [41]. In all, the convergence has been described relating with the limit behaviors of the random variables used as well as, the absolute global convergence of a random sequence of values obtained by subtracting the hidden optimal value of the problem from the result expected to be obtained by the algorithm at each iteration. The conflicting multiparamertic behavior of the lower-level problems in the MSPs make it hard to apply the results of the papers for a convergence study of our algorithm, SEAMSP. We recommend further research on (ε, δ)-convergence of evolutionary algorithms of MSPs. In this section, according to definition-4.1, for any given $\varepsilon, \delta > 0$, an (ε, δ)-convergence of SEAMSP to a stackelberg solution of (1.1) has been studied for problems satisfying the assumptions in Lemma-4.1 below. The result in Theorem-4.1 shows the performance of SEAMSP in searching the entire feasible region

of a given MPP. The mutation operator of the decision variables defined in SEAMSP search better points (if exist) within the given step sizes of the decision variables, so that it doesn't affect the results obtained, instead it helps to get better reactions (possibly smaller δ) of each level. Similarly, the iterations of SEAMSP helps to find better points for the leaders problem, without affecting the fixed δ-reaction of problems below the leader's level.

Lemma 4.1. *Consider the k-level stackelberg problem, defined in section-1.2, where* $X_i = [m_i, M_i]^{n_i}$, $i \in \{1, 2, \ldots, k\}$ *and* $S = X_1 \times \cdots \times X_k$. *Let,* $\forall j \in \{1, \ldots, k\}$, f_j *be a continuous function and* S_j *be a convex set. Let* $\forall j \in \{2, \ldots, k\}$, f_j *be a strictly convex function w.r.t* x_j *and* $\forall (x_1^*, \ldots, x_{j-1}^*) \in X_1 \times \cdots \times X_{j-1}$, $M(x_1^*, \ldots, x_{j-1}^*) \neq \emptyset$. *Let* $\forall (x_1, x_2, \ldots, x_k)$, *such that* $(x_2, \ldots, x_k) \in M(x_1)$, $\exists \mu_1 > 0 : \beta((x_1, \ldots, x_k), \mu_1) \cap S \subseteq \Omega$. *In-addition* $\forall j \in \{2, \ldots, k\}$, $\exists \mu_j > 0 : (x_1^*, \ldots, x_{j-1}^*, x_{j_*}, \ldots, x_{k_*}) \in \Omega$, $\forall (x_{j_*}, \ldots, x_{k_*}) \in \beta((x_j^{'}, \ldots, x_k^{'}), \mu_j) \cap S$ *and* $(x_j^{'}, \ldots, x_k^{'}) \in M(x_1^*, \ldots, x_{j-1}^*)$. *Then* $\forall \varepsilon, \delta > 0$ *there exist step sizes,* δ_j $(\forall j \in \{1, \ldots, k\})$, *of SEAMSP such that each iteration of the algorithm results with an* (ε, δ)*-approximate solutions.*

Proof. :

- Step-1 (existence of solution)
  by definition $IR = \{(x_1, x_2, \ldots, x_k) \in S_1 : x_1 \in X_1, (x_2, \ldots, x_k) \in M(x_1)\}$, and the MSP is:
  $$\min_{(x_1, \ldots, x_k) \in IR} f_1(x_1, \ldots, x_k) \tag{4.2}$$
  but, from the assumption: $\forall x_1 \in X_1, \exists \mu_2 > 0, : (x_1, x_2^{'}, \ldots, x_k^{'}) \in \Omega, \forall (x_2^{'}, \ldots, x_k^{'}) \in \beta((x_2, \ldots, x_k), \mu_2) \cap S$, and $(x_2, \ldots, x_k) \in M(x_1) \neq \emptyset$
  $\Rightarrow (x_1, x_2, \ldots, x_k) \in \Omega$, $\forall x_1 \in X_1$ and $(x_2, \ldots, x_k) \in M(x_1)$
  $\Rightarrow (x_1, x_2, \ldots, x_k) \in S_1$, $\forall x_1 \in X_1$ and $(x_2, \ldots, x_k) \in M(x_1)$
  So, problem-(4.2) is equivalent to:
  $$\begin{aligned} &\min f_1(x_1, \ldots, x_k) \\ &s.t \quad x_1 \in X_1 \\ &\quad\;\; (x_2, \ldots, x_k) \in M(x_1) \end{aligned} \tag{4.3}$$

which is a parametric optimization problem of a continuous function over a point-to-set map: $M : X_1 \to X_2 \times \cdots \times X_k$

From assumption, f_j is strictly convex function w.r.t x_j and,
$\forall (x_1^*, \ldots, x_{j-1}^*) \in X_1 \times \cdots \times X_{j-1},\ \exists \mu_j > 0 :\ (x_1^*, \ldots, x_{j-1}^*, x_{j_*}, \ldots, x_{k_*}) \in \Omega,$
$\forall (x_{j_*}, \ldots, x_{k_*}) \in \beta((x_j^{'}, \ldots, x_k^{'}), \mu_j) \cap S$ and $(x_j^{'}, \ldots, x_k^{'}) \in M(x_1^*, \ldots, x_{j-1}^*) \neq \emptyset$.

$\Rightarrow f_j(x_1^*, \ldots, x_{j-1}^*, x_j, x_{j+1}^{'}, \ldots, x_k^{'})$ is strictly convex function, where
$(x_j^{'}, x_{j+1}^{'}, \ldots, x_k^{'}) \in M(x_1^*, \ldots, x_{j-1}^*) \neq \emptyset$, $(x_1^*, \ldots, x_{j-1}^*, x_j, x_{j+1}^{'}, \ldots, x_k^{'}) \in S \cap S_j \cap \cdots \cap S_k$, $\forall (x_j, x_{j+1}^{'}, \ldots, x_k^{'}) \in \beta((x_j^{'}, \ldots, x_k^{'}), \mu_j) \cap S$

$\Rightarrow$ the j^{th}-level problem, $j \in \{2, \ldots, k\}$, has at least one decision, $x_j^{'}$, for any decision of the other levels.
From theorem-A.1, regardless of the other decisions, the j^{th}-level problem, $j \in \{2, \ldots, k\}$, reacts with a unique solution.

$\Rightarrow \forall x_1 \in X_1$, $M(x_1)$ has a unique solution
Therefore, problem-(4.3) is equivalent to:

$$\begin{aligned} \min\ & f_1(x_1, M(x_1)) \\ s.t\quad & x_1 \in X_1, \end{aligned} \tag{4.4}$$

but $X_1 = [m_1, M_1]^{n_1}$ is a compact set by Definition-A.3. So problem-4.4 is an optimization problem of a continuous function over a compact set. Hence the existence of a solution follows from a well known Weirstrass theorem.

- Step-2 (selection of step-sizes)
  Let $M(x_1, \ldots, x_{j-1}) = (x_j^{'}, \ldots, x_k^{'})$, for arbitrary $(x_1, \ldots, x_{j-1}) \in X_1 \times \cdots \times X_{j-1}$.
  From assumption, f_j is continuous on $(x_1, \ldots, x_{j-1}, x_j^{'}, \ldots, x_k^{'})$
  $\Rightarrow \forall \delta > 0, \exists \delta_j^{'} > 0 :$
  $f_j(x_1, \ldots, x_k) \in \beta(f_j(x_1, \ldots, x_{j-1}, x_j^{'}, \ldots, x_k^{'}), \delta)$ whenever,
  $(x_1, \ldots, x_k) \in \beta((x_1, \ldots, x_{j-1}, x_j^{'}, \ldots, x_k^{'}), \delta_j^{'})$

Let $\delta^* = min\{\delta'_j : j \in \{2, \ldots, k\}\}$, then
$(x_1, \ldots, x_k) \in \beta((x_1, \ldots, x_{j-1}, x_j^{'}, \ldots, x_k^{'}), \delta^*)$
$\Rightarrow |f_j(x_1, \ldots, x_{j-1}, x_j^{'}, \ldots, x_k^{'}) - f_j(x_1, \ldots, x_k)| < \delta, \forall j \in \{2, \ldots, k\}$
but, $(x_1, \ldots, x_k) \in \beta((x_1, \ldots, x_{j-1}, x_j^{'}, \ldots, x_k^{'}), \delta^*)$
$\Rightarrow \|x_j - x_j^{'}\|^2 + \ldots \|x_k - x_k^{'}\|^2 < \delta^{*2}$
choose $\|x_i - x_i^{'}\| = \|x_{i'} - x_{i'}^{'}\|, \forall i, i' \in \{j, \ldots, k\}$
$\Rightarrow (k - j + 1)\|x_j - x_j^{'}\|^2 < \delta^{*2}$
$\Rightarrow \|x_j - x_j^{'}\|^2 < \frac{\delta^{*2}}{k-j+1} \Rightarrow \|x_j - x_j^{'}\| < \frac{\delta^*}{(k-j+1)^{\frac{1}{2}}}$
$\Rightarrow \sum_{i=1}^{n_j} |x_{ji} - x_{ji}^{'}|^2 < \frac{\delta^*}{(k-j+1)^{\frac{1}{2}}}$, where $x_j = (x_{j1}, \ldots, x_{jn_j})$
choose $|x_{ji} - x_{ji}^{'}| = |x_{ji'} - x_{ji'}^{'}|, \forall i, i' \in \{1, \ldots, n_j\}$
$\Rightarrow$

$$|x_{ji} - x_{ji}^{'}| < \left(\frac{\delta^*}{n_j(k-j+1)^{\frac{1}{2}}}\right)^{\frac{1}{2}} \tag{4.5}$$

Hence any feasible movement of the sample points from their optimal reaction, with a step size of $\left(\frac{\delta^*}{n_j(k-j+1)^{\frac{1}{2}}}\right)^{\frac{1}{2}}$ results in a δ-reaction.
but, from assumption $\forall j \in \{2, \ldots, k\}$, $\exists \mu_j > 0$:

$$(x_j, \ldots, x_k) \in \beta((x_j^{'}, \ldots, x_k^{'}), \mu_j) \cap S \Rightarrow (x_1, \ldots, x_{j-1}, x_j^{'}, \ldots, x_k^{'}) \in \Omega \tag{4.6}$$

choose $\mu^* = min\{\mu_j : j \in \{2, \ldots, k\}\}$ then,
$(x_j, \ldots, x_k) \in \beta((x'_j, \ldots, x'_k), \mu^*) \cap S \Rightarrow (x_1, \ldots, x_{j-1}, x_j, \ldots, x_k) \in \Omega, \forall j \in \{2, \ldots, k\}$
Let, $r_j = min\left\{\left(\frac{\delta^*}{n_j(k-j+1)^{\frac{1}{2}}}\right)^{\frac{1}{2}}, \left(\frac{\mu^*}{n_j(k-j+1)^{\frac{1}{2}}}\right)^{\frac{1}{2}}\right\}, j \in \{2, \ldots, k\}$

$\Rightarrow$ any choose of step sizes, $\delta_j : 0 < \delta_j \leq r_j$, $j \in \{2, \ldots, k\}$, results with in a δ-reaction.

Let $(x_1^*, \ldots, x_k^*)$ be a stackelberg solution,
f_1 is continuous $\Rightarrow \forall \varepsilon > 0, \exists \varepsilon^* > 0$:
$(x_1, \ldots, x_k) \in \beta((x_1^*, \ldots, x_k^*), \varepsilon^*) \Rightarrow f_1(x_1, \ldots, x_k) \in \beta(f_1(x_1^*, \ldots, x_k^*), \varepsilon)$
similar to the result in (4.5) we can reach to $|x_{ji} - x_{ji}^*| < \left(\frac{\varepsilon^*}{n_j k^{\frac{1}{2}}}\right)^{\frac{1}{2}}$

$\Rightarrow$ any feasible movement of the sample points, x_j, from the optimal point with a step size of $\left(\frac{\varepsilon^*}{n_j k^{\frac{1}{2}}}\right)^{\frac{1}{2}}$ results in less than ε difference of the leaders objective function value from the optimal value.

From assumption $\exists \mu_1 > 0$:

$$(x_1, \ldots, x_k) \in \beta((x_1^*, \ldots, x_k^*), \mu_1) \cap S \Rightarrow (x_1, \ldots, x_k) \in \Omega \tag{4.7}$$

Let, $r_j^* = min\left\{\left(\frac{\mu_1}{n_j k^{\frac{1}{2}}}\right)^{\frac{1}{2}}, \left(\frac{\varepsilon^*}{n_j k^{\frac{1}{2}}}\right)^{\frac{1}{2}}\right\}$

Now, choose step-size of the distributions, δ_j, $j \in \{1, 2, \ldots, k\}$ as:-

$$\delta_1 = r_1^*, \quad \delta_j = min\{r_j, r_j^*\}, \quad where \;\; j \in \{2, \ldots, k\} \tag{4.8}$$

□

Theorem 4.1. *Let a MSP satisfy all the assumptions in lemma-4.1, then $\forall \varepsilon, \delta > 0$, there exist an algorithm parameter, $\alpha < \infty$, of SEAMSP such that, each iteration of the algorithm results with an (ε, δ)-approximate solutions.*

Proof. :

From lemma-4.1, selecting step-sizes δ_j, $j \in \{1, \ldots, k\}$, results in an (ε, δ)-approximate results. So the number of sample points on a single space of x_j, $b_j = \frac{d_j}{\delta_j}$, for $j \in \{1, \ldots, k\}$. Now select α, safely large number which results in $a_j \geq 2b_j, \forall j \in \{1, \ldots, k\}$.

□

Corollary 4.1. *For any $(\varepsilon_*, \delta_*)$-approximate solution,$(x_1^*, \ldots, x_k^*)$, of MSP satisfying all the assumptions in lemma-4.1 where $\varepsilon_*, \delta_* > 0$, $(\varepsilon_*, \delta_*) = inf\{(\varepsilon, \delta) : (x_1^*, \ldots, x_k^*)$ is an (ε, δ)-approximate solution to the MSP$\}$, $\exists \alpha$ such that each iteration of the SEAMSP results with a first-rank better solution than $(x_1^*, \ldots, x_k^*)$.*

Proof. : Let $\varepsilon = \frac{\varepsilon_*}{2}, \delta = \frac{\delta_*}{2}$

By Theorem-4.1, $\exists \alpha$ such that each iteration of the algorithm results in an (ε, δ)-approximate solution.

Hence, each iteration results with a first-rank better solution than $(x_1^*, ..., x_k^*)$. □

The assumptions in Lemma-4.1, could be satisfied by different MPPs, containing non-convex, non-linear and non-differentable objective functions in all levels.
For instance, consider a BSP given below:-

$$\min_{-5\leq x_1\leq 5} f_1(x_1, x_2) = 5sin(x_1^2 + 1) - |cos(x_2 - 1)| + |x_1 - x_2|$$

$$\begin{aligned} s.t. \quad & x_1 + x_2 \leq 6 \\ & x_1 - x_2 \leq 6 \\ & \min_{-5\leq x_2\leq 5} f_2(x_1, x_2) = |-x_1^3 + 2x_1^2| + x_2^2 \\ & s.t. \quad -x_1 + x_2 \leq 6 \end{aligned}$$

, where $X_1 = [-5, 5]$, $X_2 = [-5, 5]$, $S_1 = \{(x_1, x_2) : x_1 + x_2 \leq 6, x_1 - x_2 \leq 6\}$, $S_2 = \{(x_1, x_2) : -x_1 + x_2 \leq 6\}$, $S = X_1 \times X_2$, and $\Omega = S \cap S_1 \cap S_2$.

The feasible regions of the leader and follower problems are shown in figure-4.2:

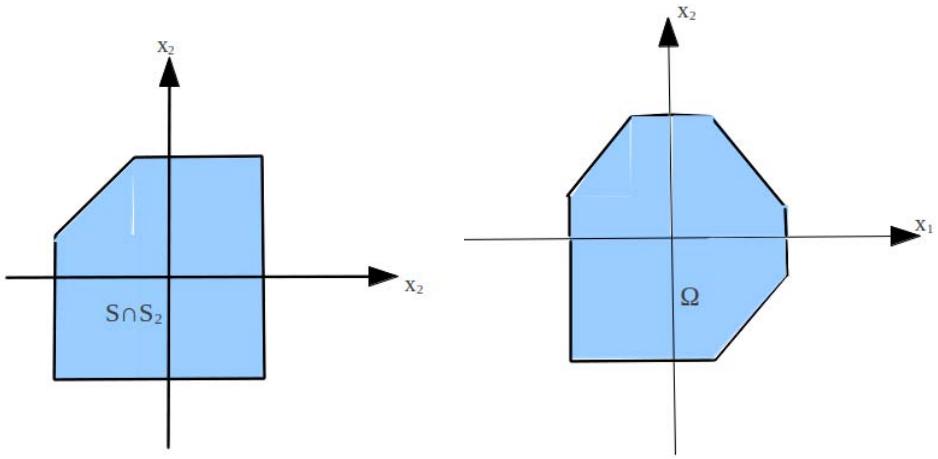

Figure 4.1:

But f_2 is continuous over the convex set $S \cap S_2$, $\forall x_1^* \in [-5, 5]$, $M(x_1^*) = \{0\}$ and $f_2(x_1^*, x_2) = |-x_1^{*3} + 2x_1^{*2}| + x_2^2$ is strictly convex function.
$\forall x_1^* \in [-5, 5]$ we can observe from figure-4.2 that $\exists \mu_2 > 0$ (any $\mu_2 : 0 < \mu_2 < 1$) such that $(x_1^*, x_2) \in \Omega$, $\forall x_2 \in \beta(0, \mu_2) \cap S$ and $\forall (x_1^*, 0), \exists \mu_1 > 0$ (any $\mu_1 : 0 < \mu_1 < 1$) such that $(x_1, x_2) \in \Omega$, $\forall (x_1, x_2) \in \beta((x_1^*, 0), \mu_1) \cap S$.

Chapter 5

Numerical Results

To show the efficiency, effectiveness and stability of the proposed algorithm for solving MSPs with various types of objective functions, we conduct some numerical experiments. For each test problem, we execute the proposed algorithm for 10 independent runs and we record the following data:-

- The solution by the algorithm
- The objective function values at each solution
- The CPU time (in seconds) required before execution
- The number of iterations routed before execution

In-addition to this we record (ε, δ)-approximation for those already have a known exact solution and easily determined reactions. A "safe-(ε, δ)-approximation" are also recorded for those having a known boundary values of the objective functions. The (ε, δ)-approximation we defined in the introduction is important to compare approximate results from different algorithms as well as to select a best result from independent runs of a problem by any algorithm.
The numerical results of the test problems are arranged in three different point of views as listed below:-

1. The first three problems are taken from the literature. They have known exact (stackelberg) solutions and for every fixed leaders variable value one can easily determine the reactions. We find (ε, δ)-approximation for each result from the reference as well as for all 10 independent runs of our proposed algorithm (SEAMSP).

2. The next five problems are newly constructed problems. These problems contains non-differentiable, non-convex and even non-continuous functions in all levels, which are beyond the scope of most existing algorithms. Eventhough we are not certain for the exact solutions, but we know the limit of the objective function values, so we record a "safe-(ε, δ)-approximation" to all the results of the independent runs. A safe-(ε, δ)-approximation of any result we record for these problems implies the exact (ε, δ)-approximation of the result is better (less) than or equal to what we have recorded.

3. The final problem of our simulation is taken from the literature. For each fixed leaders variable value, it is not easy to find the exact reaction and the limit of the objective functions value is not trivial to determine. This makes hard to predict an (ε, δ)-approximation of any result. We record the results of each independent run by SEAMSP as well as the results obtained from references and we left the choose of best solution to the reader.

The details of the problems and the simulation results are shown in the next section.

5.1 Numerical Results of Test Problems

In all the test problems the algorithm parameters are selected as follows:-
$\gamma = 10$, $t = 5$, $\beta_1 = 5$, $\beta_2 = 50$, $\epsilon_1 = 0.005$, and $\alpha = 50,000$ for all the examples,
except for Examples-5.2,5.4,5.5, $\alpha = 10,000$, for Example-5.7, $\alpha = 70,000$, for Example-5.8, $\alpha = 30,000$, and for all the TPPs $\gamma = 6$, $t = 3$, $\beta_1 = 10$, $\beta_2 = 50$, $\epsilon_1 = 0.01$.
The best solution obtained by [53] for Example-5.1 is (0,30,-10,10) where $F = 0$, and $f = 100$, which is a global optimal solution (stackelberg solution) to the given BPPs, so from Proposition-4.1, we don't expect to get better solution for the given problem.

The (ε, δ)-approximation of the results in all the independent runs by SEAMSP are recorded to show the closeness of the results to the global optimal solution. Furthermore, for Examples-5.2 and 5.3, we compare the results obtained by SEAMSP with those presented in the corresponding references.

Example 5.1.

$$\min_{0\leq x\leq 50} F(x,y) = |sin(2x_1 + 2x_2 - 3y_1 - 3y_2 - 60)|$$

$$s.t. \quad x_1 + x_2 + y_1 - 2y_2 \leq 40$$

$$\min_{-10\leq y\leq 20} f(x,y) = (y_1 - x_1 + 20)^2 + (y_2 - x_2 + 20)^2$$

$$s.t. \quad x_1 - 2y_1 \geq 10$$
$$x_2 - 2y_2 \geq 10$$

No.	Solution	F	f	ε	δ	CPU(Sec.)	iter
1	(11.0281,11.0281,-8.9517,-8.9105)	3.1223e-5	0.0042	3.123e-5	4.183e-3	18.8450	15
2	(11.1888,11.1888,-8.8228,-8.8228)	0.0073	0.0003	7.258e-3	3.000e-4	12.3450	10
3	(1.9306,24.8787,-9.9125,4.6448)	0.0029	66.5887	2.892e-3	0.0219	12.2850	10
4	(11.6728,45.0446,-9.0863,17.4730)	6.6472e-4	57.9058	6.643e-4	0.0225	12.4420	10
5	(17.2424,17.2302,-2.6929,-2.6929)	0.0302	0.0101	0.0294	0.0100	12.1780	10
6	(21.8722,26.4172,1.7206,6.2990)	0.0462	0.0370	0.0442	0.0357	12.0570	10
7	(20.0594,20.0594,0.2298,0.2298)	0.0090	0.0581	8.920e-3	0.0533	18.7120	15
8	(21.8981,45.3253,2.2913,17.2913)	0.0092	64.6997	9.117e-3	0.0911	18.3060	15
9	(9.0032,8.9778,-9.7659,-9.7659)	5.7001e-5	3.0933	5.700e-5	0.2577	12.2390	10
10	(18.8155,35.4808,-1.1507,12.1107)	0.0048	11.3587	4.778e-3	0.3115	18.4380	15

Table 5.1: SEAMSP

From Table-5.1,we show that all the independent runs of SEAMSP results in an (ε, δ)-approximation to example-5.1 for small values of ε and δ. The mean CPU time of SEAMSP for the 10 independent runs of Example-5.1 is 14.7847 sec.,which is much better than the mean CPU time by the algorithm in [53], which is 33.354 sec.

Example 5.2.

$$\min_x F(x,y) = x^2 + (y-10)^2$$

$$s.t. \quad -x + y \leq 0$$

$$0 \leq x \leq 15$$

$$\min_y f(x,y) = (x + 2y - 30)^2$$

$$s.t. \quad x + y \leq 20$$

$$0 \leq y \leq 20$$

For the problem in Example-5.2, one can easily determine the reaction (of any fixed x) and the value of the leaders objective function at a stackelberg solution is known to be 100. So we can compare any results for the problem using a first-rank better with minimum possible (ε, δ)-approximation of a given approximate result of the problem.

Best Solution	Best F	Best f	ε	δ
(10.03,9.969)	100.58	0.001	5.710e-3	9.99e-5

Table 5.2: [37]

No.	Solution	F	f	ε	δ	CPU(Sec.)	iter
1	(10.0002,9.9981)	100.0035	1.3216e-005	3.466e-5	1.323e-5	2.2880	10
2	(9.9989,9.9964)	99.9776	6.9390e-005	0	6.939e-5	2.6250	10
3	(10.0012,9.9768)	100.0265	0.0020	2.624e-4	1.997e-3	3.4280	15
4	(9.9962,9.9900)	99.9244	5.6589e-004	0	5.656e-4	1.6590	10
5	(9.9931,9.9907)	99.8614	6.4871e-004	0	6.483e-4	1.6910	10
6	(9.9931,9.9905)	99.8620	6.6738e-004	0	6.734e-4	2.2720	10
7	(9.9927,9.9902)	99.8538	7.2756e-004	0	7.271e-4	3.3850	15
8	(9.9909,9.9896)	99.8189	8.9032e-004	0	8.896e-4	2.2630	10
9	(9.9919,9.9825)	99.8389	0.0019	0	1.897e-3	5.6340	25
10	(10.065,9.9198)	101.3134	0.0091	0.0129	9.018e-3	2.2640	10

Table 5.3: SEAMSP

From Tables-5.2 and 5.3 we can show that the best two results of SEAMSP for Example-5.2 are first-rank better than the best result in [37] and the next seven results of SEAMSP are second-rank better than the result in [37].

Example 5.3.

$$\min_{0\le x\le 0.5} f_1(x,y,z) = -x + 4y$$

$$s.t. \quad x + y \le 1$$

$$\min_{0\le y\le 1} f_2(x,y,z) = 2y + z$$

$$s.t. \quad -2x + y \le -z$$

$$\min_{0\le z\le 1} f_3(x,y,z) = -z^2 + y$$

$$s.t. \quad z \le x$$

In Example-5.3, for any fixed x, we can easily determine the reaction and it is clear that the leaders objective function value can never be less than -0.5, so we can safely determine the values of ε and δ to describe the (ε, δ)-approximation of the results in all the independent runs. In fact, we can easily verify that, the function values, f_1, f_2, f_3, at the exact solution are, $-0.5, 0.5, -0.25$, respectively. Table-5.6 shows the results of all the independent runs of SEAMSP for Example-5.3 and the (ε, δ)-approximation of each.

Best Solution	Best f_1	Best f_2	Best f_3	ε	δ
(0.5,0,0.0095)	-0.5	0.0127	-0.2499	0	8.001e-5

Table 5.4: [49]

Best Solution	Best f_1	Best f_2	Best f_3	ε	δ
(0.5,1,1)	3.5	3	0	0.8889	0.5

Table 5.5: [26]

No.	Solution	f_1	f_2	f_3	ε	δ	CPU(Sec.)	iter
1	(0.4994,0.0016,0.4988)	-0.4929	0.5020	-0.2472	4.756e-3	1.732e-3	5.3650	6
2	(0.4977,0.0004,0.4919)	-0.4959	0.4928	-0.2415	2.741e-3	4.677e-3	5.0110	6
3	(0.4957,0.0005,0.4916)	-0.4938	0.4925	-0.2412	4.151e-3	3.238e-3	5.0760	6
4	(0.4996,0.0026,0.4858)	-0.4879	0.4917	-0.2331	8.133e-3	0.0113	4.8790	6
5	(0.4965,0.0023,0.4936)	-0.4873	0.4983	-0.2414	8.539e-3	2.266e-3	5.1450	6
6	(0.4992,0.0044,0.4929)	-0.4814	0.5018	-0.2386	0.0126	5.007e-3	4.9770	6
7	(0.4883,0.0005,0.4765)	-0.4862	0.4775	-0.2265	9.286e-3	9.325e-3	5.1530	6
8	(0.4938,0.0026,0.4878)	-0.4835	0.4929	-0.2353	0.0112	5.939e-3	4.9780	6
9	(0.4916,0.0027,0.4727)	-0.4807	0.4781	-0.2207	0.0131	0.0183	5.1430	6
10	(0.4900,0.0014,0.4774)	-0.4846	0.4801	-0.2266	0.0104	9.865e-3	4.9810	6

Table 5.6: SEAMSP

From Tables-5.4, 5.5, 5.6 and Definition-4.2, we just observe that the result in Table-5.4 is first-rank better than all the results by SEAMSP and the result obtained from [26], but the objective function values (of middle and bottom-levels) and the best solution, (0.5,0,0.0095), in the paper are not relevant. Instead, the leader, middle and bottom objective function values at that point are -0.5, 0.0095 and -9.025e-5 respectively. At these objective function values we can get $\varepsilon = 0$ and $\delta = 0.2499$ in which the results of SEAMSP in all the independent runs are second-rank better than the best solution obtained in [49]. We can also observe from Tables-5.5 and 5.6, all results of the independent runs by SEAMSP, are first-rank better than the result obtained from [26]. In-fact the result obtained from [26] is not in the constraint region of the given problem.

Example 5.4.

$$\min_x F(x,y) = \begin{cases} |x_1^2 - x_2^2| + |x_2^2 - x_3^2| + |y_1^2 - y_2^2| + 5 \ , & if \quad x_1 < x_2 \quad and \quad y_1 < y_2 \\ sin^2(x_2^2 - 2y_1) + 5cos^2(2y_2 - 3x_1) + 5 \ , & if \quad x_1 < x_3 \quad and \quad x_2 \leq x_1 \\ |y_3^2 - y_2^2 + 9| \ , & else \end{cases}$$

$$\begin{aligned} s.t. \quad & x_1 - 2x_2^2 y_1 + y_2 \leq 0 \\ & x_1 - 4x_2 + 4y_3 \leq 0 \\ & -x_1 + 4x_2 - 2y_3 \leq 0 \\ & \min_y f(x,y) = (y_3 - y_1)^2 - cos(x_2 - x_1) \\ & \quad s.t. \quad -x_1 + x_2^2 - y_3^2 \leq 0 \\ & \qquad -2 \leq x, y \leq 2 \end{aligned}$$

For any fixed $x \in [-2, 2]$ of Example-5.4, we can easily determine the reaction and it is clear that the leaders objective function value can never be less than 5, so we can safely determine the values of ε and δ to describe the (ε, δ)-approximation of the results in all the independent runs. Table-5.7 shows the results of all the independent runs by SEAMSP and the safe-(ε, δ)-approximations of each result, for Example-5.4.

No.	Solution	F	f	ε	δ	CPU(S.)	iter
1	(0.6889,-0.2638,0.7045,-0.7970,-1.3449,-0.7970)	5.0022	-0.5795	3.666e-4	0	23.0290	15
2	(0.3060,-0.3325,0.3231,-0.7278,-1.8707,-0.7278)	5.0031	-0.8030	5.164e-4	0	15.3250	10
3	(0.2910,-0.4952,0.3695,-1.1020,-1.9020,-1.0635)	5.0074	-0.7051	1.232e-3	8.211e-4	22.4350	15
4	(0.3766,-0.0893,0.5106,-0.3218,-1.7907,-0.3218)	5.0178	-0.8934	2.958e-3	0	22.4030	15
5	(0.5139,-0.1670,0.6592,-0.4908,-1.6337,-0.4908)	5.0303	-0.7770	5.025e-3	0	23.0140	15
6	(0.2991,-0.6262,0.4749,-1.3285,-1.8999,-1.3285)	5.0308	-0.6016	5.108e-3	0	29.5080	20
7	(0.1991,0.0033,0.3497,-0.1549,-0.5329,-0.0785)	5.0310	-0.9751	5.141e-3	2.937e-3	14.6970	10
8	(0.5860,-0.6061,0.7563,-0.9582,-1.4530,-0.8677)	5.0311	-0.3615	6.023e-3	6.023e-3	16.1220	11
9	(-0.3082,-0.4945,-0.1333,-0.8544,-1.3120,-0.8267)	5.0467	-0.9819	7.724e-3	4.037e-4	14.9150	10
10	(0.9276,-0.0895,1.0807,-0.4915,-0.9368,-0.6064)	5.0264	-0.5127	8.661e-3	8.661e-3	22.4080	15

Table 5.7: SEAMSP

From the results in Table-5.7, we observe that all the independent runs are in an (ε, δ)-approximation, with sufficiently small values of ε and δ. Moreover five out of the ten independent runs results are almost in the exact inducible region, IR.

Example 5.5.

$$\min_x F(x,y) = 2|sin(x_1 - y_2 - x_2 + y_4)|$$

$$\begin{aligned} s.t. \quad & x_1 - 2x_2^2 y_4 + y_2 \leq 0 \\ & x_1 - 4x_2 + y_3 \leq 0 \\ & -x_1 + 4x_2 - 2y_3 \leq 0 \end{aligned}$$

$$\min_y f(x,y) = x_1^2 y_1^2 - 2x_1^2 y_1 y_2 + x_1^2 y_2^2 + 5|(y_4 - y_1)^3|$$

$$\begin{aligned} s.t. \quad & -x_1 + x_2^2 - y_3^2 \leq 0 \\ & -2 \leq x, y \leq 2 \end{aligned}$$

Table-5.8 shows the results of all the independent runs by SEAMSP and the safe-(ε, δ)-approximations of each result of the problem in Example-5.5.

No.	Solution	F	f	ε	δ	CPU	iter
1	(-1.0522,-1.0522,1.6504,1.6504,-1.5496,1.6504)	0	-8.8818e-16	0	0	93.413	10
2	(-1.0356,-1.0356,1.2373,1.2373,-1.4820,1.2373)	0	-4.4409e-16	0	0	131.324	15
3	(-1.4671,-1.4671,1.7367,1.7367,-1.9170,1.7367)	0	0	0	0	137.805	15
4	(-1.0000,-1.0000,1.7507,1.7507,-1.4493,1.7507)	0	0	0	0	89.791	10
5	(-1.0284,-1.0284,1.6975,1.6975,-1.5025,1.6975)	0	0	0	0	88.137	10
6	(-1.1977,-1.1977,0.7623,0.7623,-1.6377,0.7623)	0	-2.2204e-16	0	0	91.742	10
7	(-1.1013,-1.1013,-0.3385,-0.4050,-1.5614,-0.4144)	0.0188	0.0076	0.0185	7.543e-3	126.421	15
8	(-1.2076,-1.2893,0.1794,0.2028,-1.8235,0.1345)	0.0268	0.0013	0.0262	1.299e-3	94.979	10
9	(-0.2267,-0.3637,-0.0082,0.1931,-0.6082,0.0724)	0.0327	0.0047	0.0317	4.679e-3	88.241	10
10	(0.0284,-0.2279,-1.3746,-1.1480,-0.3480,-1.3746)	0.0594	4.1335e-05	0.0561	4.134e-5	131.653	15

Table 5.8: SEAMSP

From Table-5.8, we show that the best six out of the ten independent run results are almost exact solutions of the problem and the rest are in a safe-(ε, δ)-approximation, with sufficiently small values of ε and δ.

Example 5.6.

$$\min_x f_1(x,y,z) = x^2 + 4y^2 + sin^2(y+z) - 6$$

$$\begin{aligned} s.t. \quad & 3x - 2y - z \leq 0 \\ & 2x - y^2 + z^3 \leq 1 \\ & 2|x| - 3y \leq 2 \\ & \min_y f_2(x,y,z) = 0.2sin^2 y + x^2 \\ & \quad s.t. \quad x - y^2 \leq -z \\ & \quad \min_z f_3(x,y,z) = z^2 + y^2 \\ & \quad s.t. \quad x + y \leq z \\ & \quad -2 \leq x,y,z \leq 2 \end{aligned}$$

In Examples-5.6,5.7 and 5.8, for any fixed leaders decision variable, x, determining the reactions may not be easy or it may take time to do that, but we can safely determine the values of ε and δ from the global optimal point of each level separately. We record the safe-(ε,δ)-approximation of the results in Table-5.9.

No.	Solution	f_1	f_2	f_3	ε	δ	CPU(Sec.)	iter
1	(-0.0290,0.0062,0.0041)	-5.9989	8.4293e-004	5.4521e-005	1.572e-4	8.423e-4	16.1310	6
2	(-0.0469,-0.0132,0.0011)	-5.9969	0.0022	1.7515e-004	4.431e-4	2.196e-3	15.4840	6
3	(-0.0715,-0.0002,-0.0078)	-5.9948	0.0051	6.0735e-005	7.435e-4	5.075e-3	15.8110	6
4	(-0.0875,0.0330,0.0222)	-5.9849	0.0077	0.0016	2.162e-3	7.642e-3	14.5540	6
5	(-0.1062,0.0320,-0.0186)	-5.9844	0.0113	0.0014	2.234e-3	0.0112	15.3050	6
6	(-0.0945,-0.0513,0.0148)	-5.9792	0.0090	0.0028	2.981e-3	8.920e-3	14.3770	6
7	(-0.1909,-0.0234,-0.0187)	-5.9596	0.0364	8.9459e-004	5.805e-3	0.0352	23.3570	9
8	(-0.2422,0.0169,-0.0106)	-5.9402	0.0587	3.9734e-004	8.617e-3	0.0555	22.8870	9
9	(-0.1368,-0.0723,0.0319)	-5.9587	0.0188	0.0062	5.936e-3	6.162e-3	15.1780	6
10	(-0.2750,0.0050,-0.0066)	-5.9243	0.0756	6.8373e-005	0.0120	0.0703	15.8290	6

Table 5.9: SEAMSP

One can observe from Table-5.9 that, in all the independent runs for the problem in

Example-5.6 a safe-(ε, δ)-approximation has been recorded with sufficiently small values of ε and δ.

Example 5.7.

$$\min_x f_1(x,y) = x_1^2(y_1^2+2y_3^2)+3x_2^2(y_2+y_4)^2+y_1^2(x_2^2-2x_1x_2)^2+2y_3^2(x_4^2-2x_1x_4)^2+x_3^2(x_3y_4-2x_2y_4)^2$$

$$\begin{aligned} s.t.\ & 2|x_3-4|-3y_2 \le 2 \\ & 2x_4-y_2^2+y_3^2 \le 1 \\ & 3x_1-2y_2-y_4 \le 0 \end{aligned}$$

$$\min_y f_2(x,y) = 2x_1^2y_3^2|y_3-y_4|+x_3^2(x_1y_2-y_1x_2)^2+3x_1^2y_4^2-5$$

$$\begin{aligned} s.t.\quad & x_1-y_2^2 \le -y_3 \\ & -2 \le x \le 2, \quad -4 \le y \le 4, \end{aligned}$$

Tables-5.10 and 5.11, contains the records from SEAMSP in 10 independent runs corresponding to problems in Examples-5.7 and 5.8, respectively. The ε, δ values in Table-5.12 shows safe-(ε, δ)-approximations of the results obtained in Tables-5.10 and 5.11, where $(\varepsilon_1, \delta_1)$, $(\varepsilon_2, \delta_2)$ are safe-(ε, δ)-approximations corresponding to the results in Tables-5.10 and 5.11, respectively, of the same order as the corresponding tables.

No.	Solution	F	f	CPU(Sec.)	iter
1	(-0.0202,0.0075,-0.0202,-0.0202, -0.2300,2.4367,-0.2300,-1.9308)	1.0986e-004	-4.9954	14.5840	12
2	(-0.0318,-0.0318,0.0573,0.0573, -0.0207,2.0913,-0.3417,-0.6924)	0.0062	-4.9984	4.3120	6
3	(0.0316,-0.0197,0.0316,-0.0197, -0.0063,2.0374,-0.0127,-2.0431)	2.1423e-005	-4.9875	10.1310	12
4	(0.0007,-0.0155,0.0007,0.0007 -0.0029,2.3595,1.5546,-2.3726)	3.1016e-006	-5.0000	13.77530	6
5	(-0.1074,-0.0055,0.0703,-0.1074, 0.0686,3.6320,-0.0671,0.3630)	0.0016	-4.9946	10.5440	15
6	(0.0598,-0.0110,-0.0342,0.0598 -0.0546,2.6121,-0.0546,-0.0546)	0.0024	-4.9999	4.5320	6
7	(-0.1386,-0.0100,0.1343,-0.1386 0.1326,2.4656,0.1326,-0.2011)	0.0026	-4.9954	10.847	15
8	(-0.0002,-0.0002,-0.0002,-0.0002, 0.0183,2.3210,0.3506,-1.9040)	1.8665e-008	-5.0000	12.4430	6
9	(-0.0703,-0.0228,0.1213,-0.0703, -0.2688,2.3978,-0.2688,-0.2688)	0.0082	-4.9985	4.1490	6
10	(-0.0123,-0.0123,-0.0123,0.0354, -0.0042,2.5659,0.2496,-2.7759)	3.9878e-005	-4.9964	10.3750	12

Table 5.10: SEAMSP

Example 5.8.

$$\min_x f_1(x,y,z) = x_1^2(y_1^2+2z_1^2)+3x_2^2(y_2+z_2)^2+y_1^2(x_2^2-2x_1x_2)^2+2z_1^2(x_4^2-2x_1x_4)^2+x_3^2(x_3z_2-2x_2z_2)^2-50$$

$$\begin{aligned} s.t.\quad & 2|x_3-4|-3y_2 \le 2 \\ & 2x_4-y_2^2+z_1^2 \le 1 \\ & 3x_1-2y_2-z_2 \le 0 \end{aligned}$$

$$\min_y f_2(x,y,z) = 2x_1^2z_1^2|z_1-z_2|+x_3^2(x_1y_2-y_1x_2)^2+3x_1^2z_2^2-5$$

$$s.t.\quad x_1-y_2^2 \le -z_1$$

$$\min_z f_3(x,y,z) = (x_2-x_4)^2+|y_1-y_2|+|z_1-z_2|$$

$$s.t.\quad x_2+y_2 \le z_1$$

$$-2 \le x \le 2, \quad -4 \le y \le 4 \quad , -5 \le z \le 5$$

No.	Solution	f_1	f_2	f_3	CPU	iter
1	(-0.0022,-0.0022,-0.0022,-0.0022,3.1394,3.1394,3.2195,3.2195)	-49.9993	-4.9999	0	84.564	7
2	(0.0243,-0.0968,-0.0968,-0.0968,2.4872,2.4872,2.5038,2.5038)	-49.2840	-4.9881	0	95.516	8
3	(-0.0049,-0.0049,-0.0049,-0.0049,3.5912,3.5912,3.6568,3.6568)	-49.9853	-4.9990	0	82.947	7
4	(-0.0326,-0.0326,-0.0326,-0.3227,2.8148,2.8148,3.0887,3.0887)	-49.7286	-4.9697	0.0842	177.037	16
5	(-0.0240,-0.0240,-0.0240,-0.3376,3.7012,3.7012,3.8759,3.8759)	-49.5883	-4.9740	0.0983	70.942	8
6	(-0.0063,-0.0063,-0.0063,-0.1497,3.5814,3.5814,3.7221,3.7221)	-49.9803	-4.9983	0.0206	105.774	10
7	(-0.0427,-0.0427,-0.0427,-0.0427,2.2802,2.2802,2.3382,2.3382)	-49.8537	-4.9701	0	180.062	16
8	(-0.0030,-0.0030,-0.0030,-0.0030,2.5926,2.5926,2.7795,2.7795)	-49.9991	-4.9998	0	152.764	15
9	(-0.0171,-0.0171,-0.0171,-0.0171,3.6616,3.6616,3.6490,3.6490)	-49.9413	-4.9883	0	197.164	18
10	(-0.0274,-0.0274,0.6272,-0.0274,1.6793,1.6793,1.9061,1.9061)	-49.2986	-4.9918	0	81.365	7

Table 5.11: SEAMSP

The safe-(ε,δ)-approximations in Table-5.12, shows that almost all results obtained by SEAMSP are near the optimal solution.

No.	ε_1	δ_1	ε_2	δ_2
1	1.099e-4	7.673e-4	1.373e-5	1.667e-5
2	6.162e-3	2.668e-4	0.0143	1.988e-3
3	2.143e-5	2.088e-3	2.884e-4	1.667e-4
4	3.102e-6	0	5.351e-3	0.0777
5	1.598e-3	9.009e-4	8.139e-3	0.0896
6	2.395e-3	1.667e-5	3.865e-4	0.0202
7	2.594e-3	7.673e-4	2.877e-3	5.009e-3
8	1.9e-8	0	1.765e-5	3.334e-5
9	8.134e-3	2.501e-4	1.153e-3	1.954e-3
10	3.988e-6	6.004e-4	0.0140	1.369e-3

Table 5.12: SEAMSP

Example 5.9.

$$\max_x F(x,y) = \frac{(x_1+y_1)(x_2+y_2)}{1+x_1y_1+x_2y_2}$$

$$s.t. \qquad x_1^2 + x_2^2 \leq 100$$

$$\max_y f(x,y) = -F(x,y)$$

$$s.t. \quad 0 \leq y_1 \leq x_1$$

$$0 \leq y_2 \leq x_2$$

The reactions and the exact solution of our last simulation Example-5.9 is not simple to determine, this makes hard to find possible minimum (ε, δ) values to describe an (ε, δ)-approximation of a result. Tables-5.13,5.14,5.15 shows the best result obtained by [53] and [30] as well as the result of 10 independent runs by SEAMSP, respectively.

Best Solution	Best F	Best f	Mean CPU(Sec.)
(7.0709,7.0713,7.0709,7.0713)	1.9802	-1.9802	177.672

Table 5.13: [53]

Best Solution	Best F	Best f
(7.0854,7.0291,7.0854,0)	1.9760	-1.9454

Table 5.14: [30]

No.	Solution	F	f	CPU(Sec.)	iter
1	(6.5783,6.5742,6.2282,0.7813)	1.9996	-1.9996	13.2101	10
2	(6.3017,6.3017,5.9340,1.1572)	1.9976	-1.9976	14.0020	10
3	(6.90850,6.9085,0.2195,6.7436)	1.9817	-1.9817	14.0580	10
4	(6.2830,6.2830,5.9316,1.2651)	1.9949	-1.9949	14.2390	10
5	(6.4552,6.4369,6.0935,0.1763)	2.0012	-2.0012	20.8910	15
6	(6.2003,6.2003,5.8529,0.9055)	1.9963	-1.9963	14.0231	10
7	(5.6983,5.6983,5.3729,0.3315)	1.9924	-1.9924	24.1250	10
8	(5.7018,5.7018,5.3617,0.8854)	1.9901	-1.9901	14.1350	10
9	(6.8612,6.8612,6.5674,0.0885)	1.9998	-1.9998	20.7170	15
10	(6.2573,6.2573,5.9259,0.4272)	1.9983	-1.9983	27.6590	20

Table 5.15: SEAMSP

We left the choose of best solution to the reader, but one can observe from Tables-5.13 and 5.15, the efficiency of SEAMSP over the algorithm in [53].

Note that :-

- The use of safe-(ε, δ)-approximation for a comparison of different solutions is totally wrong, it is only used to describe the goodness of a solution which is near to the global optimal solutions of individual problems in a MSP. To show this, consider problem-(4.1) in which the function values at $(5, 5)$ better approaches than the known stackelberg solution approaches, to the global optimal solutions of the individual problems (i.e -1000).

Conclusion

In all the test problems, SEAMSP results in an (ε, δ)-approximation of all the independent runs, with sufficiently small values of ε and δ. SEAMSP has been tested with MSPs including complex objective functions, like discontinuous, non-convex, non-linear non-differentiable...etc without any restriction, in which no existing algorithm has been worked with such kind of problems. The proposed algorithm performs well in all the test problems including the difficult once and all the results shows as the proposed algorithm can find near global optimal solutions relatively in a short period of time. This indicates that the proposed algorithm is efficient and effective. The closeness of the results in each independent runs shows us the proposed algorithm is stable. The definitions of (ε, δ)-approximation and the new comparison methods might be used as a referee for better approximate stackelberg solutions of a given problem. The newly constructed problems could be used as a benchmark for a comparison of Monte Carlo algorithms.

We believe that, the theoretical and numerical parts of our proposed algorithm constitutes a new and useful addition to the topic. Strengths of the algorithm include that it can be extended to any optimization problems, including cooperative MPPs and multiobjective programming problems. Our further research will include extension of SEAMSP to cooperative MPPs as well as multiobjective optimization problems.

Appendix A

A.1 Definitions and Optimal Conditions

Definition A.1. [10]

- *A set C is convex if for any $x_1, x_2 \in C$ and any λ with $0 \leq \lambda \leq 1$, we have $\lambda x_1 + (1-\lambda)x_2 \in C$.*
- *A function $f : C \to R$, $C \subseteq R^n$, is convex if C is a convex set and if for all $x, y \in C$ and λ with $0 \leq \lambda \leq 1$, we have:*

$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda)f(y) \tag{A.1}$$

- *A function f is strictly convex if strict inequality holds in (A.1) whenever $x \neq y$ and $0 < \lambda < 1$.*

Definition A.2. [24]

- *If $x_0 = (x_1, \ldots, x_n) \in R^n$ and $\mu > 0$, the open ball of radius μ about the point x_0 is, $\beta(x_0, \mu) = \{x = (x_1^*, \ldots, x_n^*) : d(x, x_0) < \mu\} = \{x : \sum_{i=1}^{n} |x_i - x_i^*|^2 < \mu^2\}$*
- *A neighborhood of x_0 is a subset of R^n which contains an open ball about x_0.*
- *A set $U \subseteq R^n$ is open if it is a neighborhood of each of its points.*
- *A set $S \subseteq R^n$ is closed if and only if its complement is open.*

- *A set $B \subseteq R^n$ is bounded if there exist a number $L \in R : d(x, y) < L, \forall x, y \in B$*

Definition A.3. [24]
A set $S \subseteq R^n$ is compact if and only if it is closed and bounded.

Definition A.4. [24] *A function $f : D \to R$ is continuous at $x_0 \in D \subseteq R^n$ if $\forall \epsilon > 0, \exists \delta > 0$:*

$$x \in \beta(x_0, \delta) \Rightarrow f(x) \in \beta(f(x_0), \epsilon)$$

A function f is said to be continuous on a set C, if it is continuous at all points in C.

Definition A.5. *(Optimization Problem)* [25]
Let $\emptyset \neq X \subseteq R^n$ and $f : X \to R$ be a function, then an optimization problem ,O, is to determine a vector $x \in X$ that solves

$$\min_x f(x) \qquad or \qquad \max_x f(x) \tag{A.2}$$

The function ,f, is called an objective function of the optimization problem ,O.

Theorem A.1. *A strictly convex function $f : C \to R$ has at most one local and global minimizer on the convex set C* [45].

Theorem A.2. ***Weierstrass Theorem***
A continuous real-valued function on a non-empty compact set S attains its maximum and minimum values.

A.2 Classification of Optimization Algorithms

Generally optimization algorithms can be divided in two basic classes: deterministic and probabilistic algorithms. Deterministic algorithms are mostly used if there is a clear relationship among the characteristics of the possible solutions and their objective function properties. Then, the search space can efficiently be explored using for example a divide and conquer, cutting-plane scheme, simplex method...etc. If the relation between a solution candidate and its function value are not so obvious or too complicated,

or the dimensionality of the search space is very high, it becomes harder to solve a problem deterministically. Trying it would possibly result in exhaustive enumeration of the search space, which is not feasible (in terms of time) even for relatively small problems. In each execution step of a deterministic algorithm, there exists at most one way to proceed. If no way to proceed exists, the algorithm has terminated [55]. Deterministic algorithms can be defined in terms of a state machine: a state describes what a machine is doing at a particular instant in time. State machines pass in a discrete manner from one state to another. Just after we enter the input, the machine is in its initial state or start state. If the machine is deterministic, this means that from this point onwards, its current state determines what its next state will be; its course through the set of states is predetermined. Note that a machine can be deterministic and still never stop or finish, and therefore fail to deliver a result. [60]

Probabilistic algorithm includes at least one instruction that acts on the basis of random numbers. There are two general classes of probabilistic algorithms: Las Vegas and Monte Carlo algorithms. A Las Vegas algorithm is a probabilistic algorithm that never returns a wrong result. Either it returns the correct result, reports a failure, or does not return at all. If a Las Vegas algorithm returns, its outcome is deterministic (but not the algorithm itself). The termination, however, cannot be guaranteed. However a Monte Carlo algorithms, always terminates. They trade in guaranteed correctness of the solution for a shorter runtime. This does not mean that the results obtained using them are incorrect but they may just not be the global optima. On the other hand, a solution a little bit inferior to the best possible one is better than one which needs 10^{100} years to be found. [55]

A heuristic is part of optimization algorithm that uses the information currently gathered by the algorithm to help to decide which solution candidate should be tested next or how the next individual can be produced. Heuristics used in global optimization are functions that help decide which one of a set of possible solutions is to be examined next. On one hand, deterministic algorithms usually employ heuristics in order to define the processing order of the solution candidates. Probabilistic methods, on the other hand, may only consider those elements of the search space in further computations that have

been selected by the heuristic [55]. A metaheuristic is an iterative generation process which guides a subordinate heuristic by combining intelligently different concepts for exploring and exploiting the search space [9].

A.2.1 Evolutionary Algorithms

Evolutionary algorithms (EAs) are population-based, Monte-Carlo metaheuristic optimization algorithms that use biology-inspired mechanisms like mutation, crossover, natural selection, and survival of the fittest in order to refine a set of solution candidates iteratively [55]. As the history of the field suggests there are many different variants of evolutionary algorithms. The common underlying idea behind all these techniques is the same: given a population of individuals the environmental pressure causes natural selection (survival of the fittest) and this causes a rise in the fitness of the population. Given a quality function to be optimized we can randomly create a set of candidate solutions, i.e., elements of the function's domain, and apply the quality function as an abstract fitness measure – the higher/lower the better. Based on this fitness, some of the better candidates are chosen to seed the next generation by applying recombination and/or mutation to them. Recombination is an operator applied to two or more selected candidates (the so- called parents) and results in one or more new candidates (the children). Mutation in evolutionary algorithms corresponds to small and random variation/s of an individual. Executing recombination and mutation leads to a set of new candidates (the offspring) that compete based on their fitness for a place in the next generation. This process can be iterated until a candidate with sufficient quality (a solution) is found or a previously set computational limit is reached. In this process there are two fundamental forces that form the basis of evolutionary systems.

- Variation operators (recombination and mutation) create the necessary diversity and thereby facilitate novelty, while
- Selection acts as a force pushing quality.

The combined application of variation and selection generally leads to improving fitness values in consecutive populations. Alternatively, evolution is often seen as a process

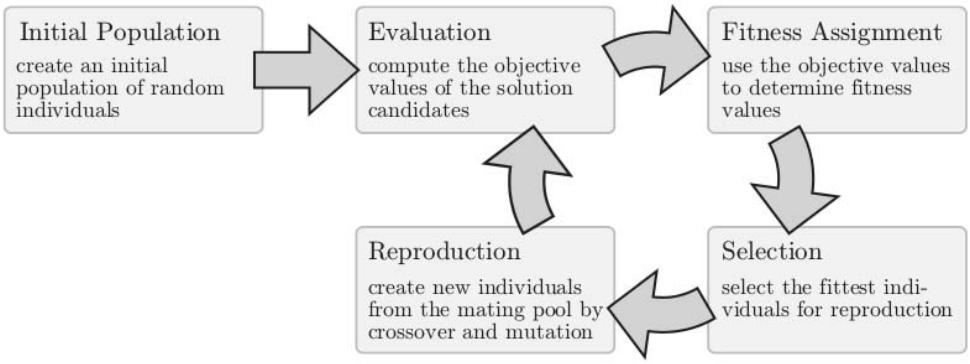

Figure A.1: The basic cycle of evolutionary algorithm [55]

of adaptation. From this perspective, the fitness is not seen as an objective function to be optimized, but as an expression of environmental requirements. Matching these requirements more closely implies an increased viability, reflected in a higher number of offsprings. The evolutionary process makes the population adapt to the environment better and better [18].

A.3 Systematic Sampling

Systematic sampling is a statistical method involving the selection of elements from an ordered sampling frame. It is a method of selecting sample members from a larger population according to a random starting point and a fixed, periodic interval. Systematic sampling is still thought of as being random, as long as the periodic interval is determined beforehand and the starting point is random.

The most common form of systematic sampling is an equal-probability method. The sampling starts by selecting an element from the list at random and then every k^{th} element in the frame is selected, where k is the sampling interval: this is calculated as:

$k = \frac{N}{n}$, where n is the sample size, and N is the population size.

The main advantage of systematic random sampling over simple random sampling is the assurance that the population will be evenly sampled. There exists a chance in simple random sampling that allows a clustered selection of subjects. This is systematically

eliminated in systematic sampling.

Since the starting point is chosen at random, each point in the population have equal chance of selection. [59]

Example A.1. *Consider choosing a systematic sample of* 9 *points from a population list in* $z_1 = [0, 9]$. *To find k, divide* $(9-0)$ *by* 9 *to get* 1. *Randomly select a number from* 0 *to* 1, *say* 0.5. *Start at the point* 0.5 *and then choose the next point by adding* 1.
The sample is made up of those points: $0.5, 1.5, 2.5, 3.5, 4.5, 5.5, 6.5, 7.5, 8.5$

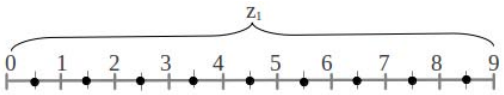

Figure A.2: Representatives of the region $[0, 9]$, chosen by systematic sampling

Similarly we can apply the concept of systematic sampling in any dimensional spaces as shown in figures below, for 2 *and* 3 *dimensional spaces .*

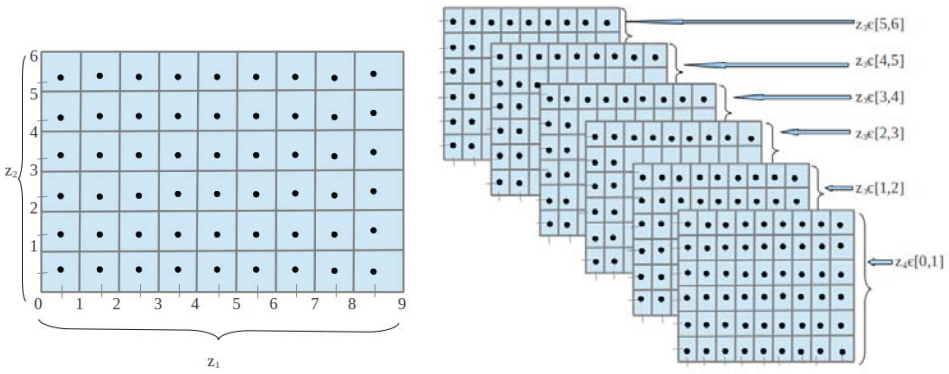

Figure A.3: Representatives of the regions $[0, 9] \times [0, 6]$ and $[0, 9] \times [0, 6] \times [0, 6]$

Bibliography

[1] Acevedo, J. and Pistikopoulos, E. N. *, Stochastic optimization based algorithms for process synthesis under uncertainty*, Computer and chemical engineering, 1998

[2] Aiyoshi, E. and Shimuzu, K.,*"A solution method for the static constrained Stackelberg problem via penalty method",* IEEE Trans. Autom. Control, 1984.

[3] Akhmouch, A., *Water Governance in Latin America and the Caribbean: A Mulitleve Approach,* OECD Regional Development Working Papers, 2012

[4] Arora, S. and Gaur, A., *A fuzzy algorithm for multilevel programming problems,* operational research society of India, 2010

[5] Bard, J., *Some properties of the bilevel programming problem,* Journal of Optimization Theory and Applications, 1991

[6] Bard, J. and Moore, J., *An algorithm for the discrete bilevel programming problem,* Naval Research Logistics, 1992

[7] Ben-Ayed, O., *Bilevel linear programming.* Computers and Operations Research,1993

[8] Bialas, W. and Karwan, M., *On two-level optimization,* IEEE Transactions on Automatic Control, 1982

[9] Blum, C. and Roli, A., *Hybrid Metaheuristics,* Springer-Verlag, 2008

[10] Boyd, S. and Vandenberghe, L.,*Convex Optimization,* Cambridge University Press, 2009

[11] Bracken, J. and McGill J., *Defense applications of mathematical programs with optimization problem in the constraints,* Operations Research, 1974.

[12] Bracken, J. and McGill, J., *Mathematical programs with optimization problems in the constraints,* Operational Research, 1973

[13] Campelo, M. and Scheimberg, S., *A note on a modified simplex approach for solving bilevel linear programming problems,* European Journal of Operational Research, 2000

[14] Candler, W, and Towersley R., *A linear two-level programming problem,* Computers and Operations Research, 1982

[15] *Case, L., An l_1 Penalty Function Approach to the Nonlinear Bilevel Programming Problem,* PhD Thesis, Waterloo University, Department of Computer Science, 1997

[16] Coelho, L., and Mariani, V., *Combining of Chaotic Differential Evolution and Quadratic Programming for Economic Dispatch Optimization With Valve-Point Effect,* IEEE , 2006

[17] Dempe, S., *Foundations of Bilevel Programming,* Kluwer Academic Publishers, 2002

[18] Eiben, A.,Smith, J., *Introduction to Evolutionary Computing,* Springer, 2003

[19] *Faisca, N., Dua, V., Rustem, B., Saraiva, P. and Pistikopoulos, E. Parametric global optimisation for bilevel programming,* Springer Science+Business Media B.V., 1997

[20] Gao, J., and Liu B., *Fuzzy multilevel programming with a hybrid intelligent algorithm,* Computers and Mathematics with Applications, 2005

[21] Gao, J, Liu B, and Gen M., *A hybrid intelligent algorithm for stochastic multilevel programming,* IEEJ Transction on Electronics, Information and Systems, 2004

[22] Greenwood, G. and Zhu, Q., *Convergence in Evolutionary Programs with Self-adaptation,* Evolutionary Computation, 1999

[23] Hansen, P., Jaumard, B., and Savard, G., *New branch-and-bound rules for linear bilevel programming,* SIAM Journal on Scientific and Statistical Computing, 1992

[24] Hoffman, K., *Analysis in Euclidean Space,* PRINCE-HALL, INC, 1975

[25] Jahn, J., *Introduction to the theory on nonlinear optimization,* Springer-Verlag, 2007

[26] Kassa, A., *A Multiparametric Programming Approach for Multilevel Optimization,* M.Sc Thesis, Addis Ababa University, Department of Mathematics, 2011

[27] Lai, Y., *Hierachical optimization: A satisfactory solution,* Fuzzy Sets and Systems, 1996

[28] Lee, E., *Fuzzy multiple level programming,* Applied Mathematics and Computation, 2001

[29] Li, H. and Wang, Y., *An Evolutionary Algorithm with Local Search for Convex Quadratic Bilevel Programming Problems,* Applied Mathematics and Information Sciences, 2011

[30] Liu, B., *"Stackelberg–Nash equilibrium for multilevel programming with multiple followers using genetic algorithms"* Elsevier Science Ltd., 1998

[31] Liu, D. and Chen, D., *An Efficient Evolutionary Algorithm for A Kind of Nondifferentiable Nonlinear Bilevel Programming* International Conference on Computational Intelligence and Security Workshops, 2007

[32] Liu, G., Han, J., and Zhang, J., *Exact penalty functions for convex bilevel programming problems,* Journal of Optimization Theory and Applications, 2001

[33] Mattson, P., *Physical systems for the solution of hard computational problems,* M.Sc Thesis, Edinburgh University, Cognitive Science and Natural Language Processing, 2003

[34] Manyem, P. and Ugon, J., *Computational Complexity, NP Completeness and Optimization Duality: A Survey,* Electronic Colloquium on Computational Complexity, Report No. 9, 2012

[35] Nishizaki, I. and Sakawa, M., *cooperative and non-cooperative multilevel programming.* Springer, 2009

[36] Nishizaki, I. and Sakawa, M., *Stackelberg solutions to multiobjective two-level linear programming problems,* Journal of Optimization Theory and Applications, 1999

[37] Oduguwa, V and Roy, R., *"Bi-level optimization using genetic algorithm,"* in Proc. IEEE Int. Conf. Artificial Intelligence Systems, 2002

[38] Patriksson, M. and Rockafellar, R., *A mathematical model and descent algorithm for bilevel traffic management,* Technical report, Department of Mathematics, Chalmers University of Technology, 2001

[39] Patriksson, M, and Wynter, L., *Stochastic mathematical programs with equilibrium constraints,* Operations Research Letters, 1999

[40] Pramanik, S., and Roy, T.K., *Fuzzy goal programming approach to multilevel programming problems,* European Journal of Operational Research, 2005

[41] Rudolph, G., *Convergence Analysis of Canonical Genetic Algorithms,* IEEE, 1994

[42] Rudolph, G.,*Convergence of evolutionary algorithms in general search spaces,* In Proceedings of the Third IEEE Conference on Evolutionary Computation, 1996.

[43] Rudolph, G., *Finite Markov Chain Results in Evolutionary Computation:A Tour d'Horizon* Fundamenta Informaticae, 1998

[44] Rudolph, G., *On a multi-objective evolutionary algorithm and its convergence to the pareto set,* In Proceedings of the 1998 IEEE International Conference on Evolutionary Computation, IEEE Press, Piscataway (NJ), 1998

[45] Simchi-Levi, D., Chen, X. and Bramel, J., *The Logic of Logistics,* Springer, 2005

[46] Shih, H., Lai Y., and Lee, E., *Fuzzy approach for multilevel programming problems,* Computer and Operations Research, 1996

[47] Stark, D. and Spall, J., *Computable Rate of Convergence in Evolutionary Computation,* ISIF, 2002

[48] Tang, K., *A Memetic Algorithm for Multi-Level Redundancy Allocation,* IEEE , 2010

[49] Tilahun, S., Kassa, S., & Ong, H., *A New Algorithm for Multilevel Optimization Problems Using Evolutionary Strategy Inspired by Natural Adaptation*, Proceeding of 12th Pacific Rim International Conference on Artificial Intelligence, 2012

[50] Tilahun, S. and Ong, H., *On Fuzzy Preference of Decision Makers in Multiobjective and Multilevel Decision Making,* Preceding of 2nd International Conference on Management, 2012

[51] Vicente, L., and Calamai, P., *Bilevel and Multilevel Programming: A Bibliography Review,* Journal of Global Optimization, 1994

[52] Wan, Z. and Zhou, S., *The convergence of approach penalty function method for approximate bilevel programming problem,* Acta mathematica scientia, Series B, English Edition, 2001

[53] Wang, Y. and Li, H., *An Evolutionary Algorithm Based on a New Decomposition Scheme for Nonlinear Bilevel Programming Problems,* Int. J. Communications, Network and System Sciences, 2009

[54] Wang, Y., Jiao, Y and Li, H., *An Evolutionary Algorithm for Solving Nonlinear Bilevel Programming Based on a New Constraint-Handling Scheme* IEEE transactions on systems, man, and cybernetics : Applications and Reviews, 2005

[55] Weise, T., *Global Optimization Algorithms -Theory and Application*, http://www.it-weise.de/, 2009

[56] Wen, U., *The "Kth-Best" algorithm for multilevel programming,* Technical report, Department of Operations Research, State University of New York at Buffalo, 1981

[57] *White, D., Penalty function approach to linear trilevel programming,* Journal of Optimization Theory and Applications, 1997

[58] Whitford, D. and David, W., *The Generalized Hierarchical model: A new approach to Resource allocation within Multilevel organizations,* BEBR, 1981

[59] Wikipedia. *Wikipedia-the free encyclopedia,* Online available at http://en.wikipedia.org/wiki/Systematic_Sampling, [accessed 2013-01-26]

[60] Wikipedia. *Wikipedia-the free encyclopedia,* Online available at http://http://en.wikipedia.org/wiki/Deterministic_algorithm, [accessed 2013-05-23]

[61] Yogeswaran, M.,Ponnambalam, S. and Tiwari, M., *An efficient hybrid evolutionary heuristic using genetic algorithm and simulated annealing algorithm to solve machine loading problem in FMS,* Taylor & Francis, 2009

[62] Zhang, G., Lu, J. and Shi, C., *An extended branch and bound algorithm for linear bilevel programming,* Applied Mathematics and Computation, 2007

[63] Zhang, G., Lu, J., Montero, J. and Zeng, Y., *Model, solution concept, and Kth-best algorithm for linear trilevel programming,* Information Sciences, 2010