11-12-2008

Modified predator-prey (MPP) algorithm for single-and multi-objective optimization problems

Souma Chowdhury
Florida International University

DOI: 10.25148/etd.FI14060822

FLORIDA INTERNATIONAL UNIVERSITY

Miami, Florida

MODIFIED PREDATOR-PREY (MPP) ALGORITHM FOR SINGLE- AND MULTI-OBJECTIVE OPTIMIZATION PROBLEMS

A thesis submitted in partial fulfillment of the

requirements for the degree of

MASTER OF SCIENCE

in

MECHANICAL ENGINEERING

by

Souma Chowdhury

2008

To: Interim Dean Amir Mirmiran
College of Engineering and Computing

This thesis, written by Souma Chowdhury, and entitled Modified Predator-Prey (MPP) Algorithm for Single- and Multi-Objective Optimization Problems, having been approved in respect to style and intellectual content, is referred to you for judgment.

We have read this thesis and recommend that it be approved.

Yiding Cao

Igor Tsukanov

George S. Dulikravich, Major Professor

Date of Defense: November 12, 2008

The thesis of Souma Chowdhury is approved.

Interim Dean Amir Mirmiran
College of Engineering and Computing

Dean George Walker
University Graduate School

Florida International University, 2008

DEDICATION

I dedicate this thesis to my parents. Without their nurturing and encouragement this work would have never been possible. I also dedicate this work to all my close friends for their continuous support and unwavering love.

ACKNOWLEDGMENTS

I wish to thank the members of my committee for their support, patience and good humor. Their gentle but firm direction has been most appreciated. Dr. Cao was particularly helpful in introducing me to various aspects of quantitative analysis. Dr. Tsukanov has been helpful in arming me with the necessary methods of computational analysis. Finally, I would like to thank my major professor, Dr. George S. Dulikravich. From the beginning, he had confidence in my abilities to not only complete the degree, but to complete it with excellence. His continuous advice and inspiration will always be appreciated.

I would like to thank my fellow researcher, Dr. Ramon J. Moral for all the ideas and help he gave me. I would also like to thank my close friend Himangi Marathe and my friend and fellow researcher Stephen Wood for their help with various graphs and schematic diagrams involved in this work.

ABSTRACT OF THE THESIS

MODIFIED PREDATOR-PREY (MPP) ALGORITHM FOR SINGLE- AND MULTI-OBJECTIVE OPTIMIZATION PROBLEMS

by

Souma Chowdhury

Florida International University, 2008

Miami, Florida

Professor George S. Dulikravich, Major Professor

The aim of this work is to develop an algorithm that can solve multidisciplinary design optimization problems. In predator-prey algorithm, a relatively small number of predators and a much larger number of prey are randomly placed on a two dimensional lattice with connected ends. The predators are partially or completely biased towards one or more objectives, based on which each predator kills the weakest prey in its neighborhood. A stronger prey created through evolution replaces this prey. In case of constrained problems, the sum of constraint violations serves as an additional objective.

Modifications of the basic predator-prey algorithm have been implemented in this study regarding the selection procedure, apparent movement of the predators, mutation strategy, dynamics of the Pareto convergence, etc. Further modifications have been made making the algorithm capable of handling equality and inequality constraints. The final modified algorithm is tested on standard constrained/unconstrained, single and multi-objective optimization problems.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

FIGURE PAGE

LIST OF ACRONYMS

EMO	Evolutionary Multi-Objective Optimization
GA	Genetic Algorithm
IOSO	Indirect Optimization on the basis of Self-Organization
MDO	Multidisciplinary Design Optimization
MOHO	Multi-Objective Hybrid Optimization
MPP	Modified Predator-Prey
NSDE	Non-Dominated Sorting Differential Evolution
NSGA	Non-Dominated Sorting Genetic Algorithm
PP	Predator-Prey
SOMPP	Single-Objective Modified Predator-Prey
SPEA	Strength Pareto Evolutionary Algorithm

CHAPTER I – INTRODUCTION

1.1 Research Objective

The Predator-Prey (PP) algorithm is an Evolutionary Multiobjective Optimization (EMO) algorithm, which utilizes the dynamics of predator and prey interactions existing in nature in search for optimal solutions. There are different forms of the Predator Prey algorithm available in literature, but most of them prove to be relatively incapable of solving complex problems, when compared to other popular evolutionary optimization algorithms. Consequently, there exist very few instances of application of any form of the PP algorithm to real world problems. Nevertheless, since the *modus operandi* of the PP algorithm is significantly different from other standard EMOs, there is sufficient basis to believe that the potentials of this algorithm have not been fully realized.

This research is directed towards the development of a robust and computationally inexpensive Modified Predator-Prey (MPP) optimization algorithm capable of handling complex design optimization problems, through the assimilation of special features of existing PP models, modifications of the same and addition of certain new features. Specific objectives are as follows:

1. Validation of MPP with standard test cases consisting of two or more objectives, as well as cases with large number of design variables. Test cases analyzed are taken from the multi-objective optimization comparison by Zitzler *et al.* [1] and the design of scalable test problems by Deb *et al.* [2].
2. Development of a single objective version of MPP (SOMPP) and subsequent validation with standard single objective test cases.

3. Formulation and inclusion of legitimate constraint handling modules into both MPP and SOMPP and subsequent validation with standard constrained multiobjective optimization problems and constrained single objective problems such as developed by Hock and Schittkowskii [3] and Schittkowskii [4], respectively.
4. Application of MPP to a practical problem, preferably in the fields of fluid and thermal systems simulated using Computational Fluid Dynamics (CFD).

1.2 Multidisciplinary Design Optimization (MDO)

Typical real world systems, be it engineering, scientific, social or financial are comprised of a large number of variables and multiple output parameters. Skilled designers and systems analysts use their knowledge, experience and intuition to assign values to these variables in order to extract the most desirable performance from the process or the system in concern. However, due to the size and complexity of the design task and likely involvement of different disciplines, it becomes increasingly difficult even for the most competent designers to account for all the variables and constraints involved simultaneously. This calls for the application of relevant, efficient and robust mathematical models. Multidisciplinary Design Optimization (MDO) is the application of numerical algorithms for designing systems with or without inherent coupling between various disciplines, in order to achieve optimal performance in terms of desired parameter outcomes, cost and reliability.

Before the 1980's, design optimization was mainly dominated by gradient based techniques, currently referred to as classical techniques that are a combination of optimality criterion and mathematical programming. Most practical systems/processes demand multi-objective optimization that is searching for feasible solutions corresponding to extreme values of one or more objectives (output parameters). In the case of multiple objectives, decision makers and designers would prefer a set of most suitable trade-off solutions, better termed as non-dominated solutions [5]. However, gradient based algorithms follow a point-by-point approach in search for better solutions, consequently leading to a single optimized solution. The last few decades have seen the development of optimization algorithms inspired by the principles of natural evolution. These algorithms, often termed as Evolutionary Optimization Algorithms (EOA), utilize a set of multiple candidate solutions (population space) to follow an iterative procedure producing a final set of the best compromise solutions, the graphical representation of these which is termed Pareto front [5]. In case of single objective problems, the Pareto front reduces to a single optimal solution known as the global minimum or global maximum. Genetic algorithm, differential evolution, particle swarm, ant colony, and predator-prey algorithms are some of the most prominent EOAs.

Most real world systems that demand optimized design are often subject to configurational and operational restrictions which should be taken into consideration during the process of optimization. This necessitates optimization algorithms capable of producing solutions that are both optimum as well as feasible with respect to the system constraints. These system constraints can be modeled as mathematical constraint functions.

1.3 PP – Literature Review

In 1998, Hans Paul Schwefel proposed a new optimization algorithm [6] to search for Pareto-optimal solutions from a randomly generated initial population of candidate solutions. This algorithm imitates the natural phenomena that a predator kills the weakest prey in its neighborhood, and the next generations of preys that evolve are relatively stronger and more immune to such predator attacks. However, this initial PP optimization algorithm seemed to have difficulty in producing well distributed non-dominated solutions along the Pareto front. Since then, several modifications of the above algorithm have appeared in literature. Deb [5] suggested an improved version of the algorithm which included certain new features, namely, the 'elite preservation operator", the 'recombination operator' and the 'diversity preservation operator'. A further modified version of the algorithm was proposed by Li [7], where a dynamic spatial structure of the predator-prey population was used. It involved the movement of both predators and preys and changing population strength of prey. Some other versions of the algorithm have been presented by Grimme *et al.* [8] and Silva *et al.* [9]. The former used a modified recombination and mutation model. The latter, predominantly a particle swarm optimization algorithm, introduces the concept of predator-prey interactions in the swarm to control the balance between exploration and exploitation, hence improving both diversity and rate of convergence.

However, most of the above versions find it difficult to produce well distributed set of Pareto optimal solutions in a limited number of function evaluations especially when dealing with problems with more than two objectives or significantly high number of decision (design) variables. In most practical applications of optimization, the calculation time for evaluating model functions dominate. This demands optimization algorithms capable of producing dependable solutions while investing the minimum number of function evaluations possible. Moreover, the forms of the PP algorithm available in literature do not have the ability to handle constraints, which form an integral part of most practical problems.

1.4 Optimization in Aerodynamics

Such multi-objective evolutionary techniques have been widely employed in the aerospace industry for optimizing design and performance such as using genetic algorithms in the conceptual phase of aerospace vehicle design and satellite constellation design [10] and aerodynamic shape design with minimization of 'drag to lift' ratio [11, 12]. The same also finds several applications in the field of gas dynamics such as using evolutionary hybrid optimization for the design of internal convectively cooled 3-D axial gas turbine blades [13], optimizing hub and shroud geometry and inlet/exit flow parameters for each row of blades in a multistage axial flow turbine [14] and aerodynamic shape design of turbine blades involving minimization of total pressure loss across the 2D linear-airfoil cascade row [15].

CHAPTER II – MULTI-OBJECTIVE PREDATOR-PREY ALGORITHM

2.1 Overview

Any multi-objective optimization problem can be stated in its general form as follows:

$$\begin{aligned}
&Min / Max \; f_i(X), \quad i = 1,2,\ldots,Nf \\
&subject \; to \\
&g_i(x) \le 0, \quad i = 1,2,\ldots,P \\
&h_i(x) = 0, \quad i = 1,2,\ldots,Q \\
&x_i^{(L)} \le x_i \le x_i^{(U)}, \quad i = 1,2,\ldots,m
\end{aligned} \tag{1}$$

A solution X is a vector of m decision (design) variables that is $x = (x_1, x_2, \ldots, x_m)^T$, which is not unique in case of multiple objectives. The last set of constraints is called the variables bounds/limits, confining each decision variable x_i to take a value within a lower $x_i^{(L)}$ and an upper $x_i^{(U)}$ limit. They determine the boundaries of the decision variable space D (also known as design variable space in case of MDO problems). The objective space is constricted by P inequality and Q equality constraints and need not span over the whole region mapped onto by the bounded decision variable. The terms $g_j(x)$ and $h_k(x)$ are called the constraint functions. Figure 1 shows the mapping between the decision space and the objective space for a general unconstrained 3-variable/2-objective problem. However, either the variable space or the objective space need not be continuous. They may be discontinuous or even discrete as is the case with integer problems. At the same time, in certain problems the variable space is likely to be unbounded, in which case it becomes substantially difficult to search for optimal solutions without prior knowledge of a favourable starting region.

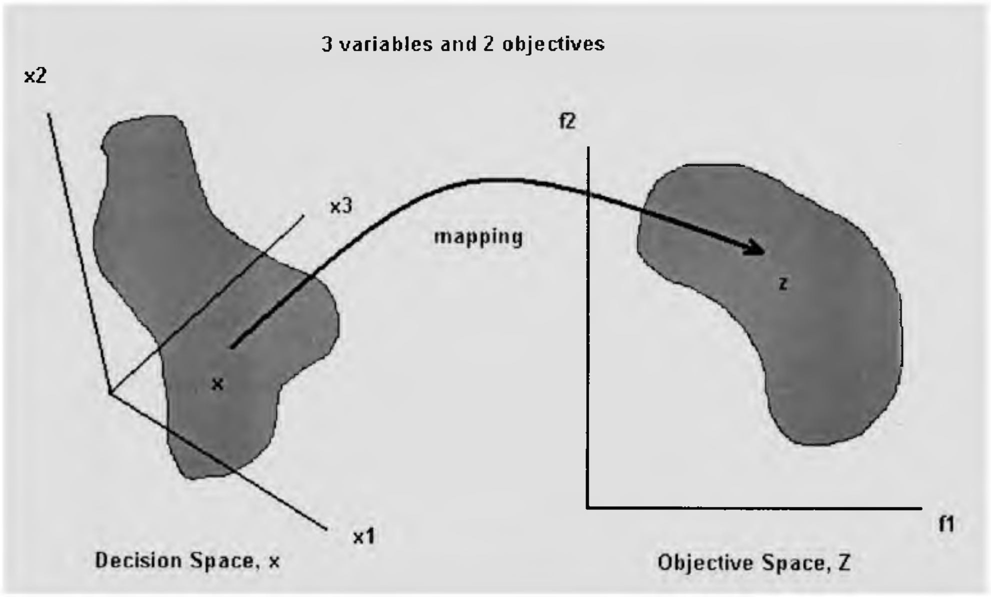

Figure 1 Variable space mapping onto the objective space

2.1.1 Feasible Space: In case of constrained problems, this objective space is curtailed to a smaller region called the feasible space. A solution should be within this feasible space in order to be a valid solution (feasible solution). Solutions lying outside the feasible space are called infeasible solutions. The above is illustrated in Figure 1.

2.1.2 Pareto Front: In most multi-objective problems the search for optimal solutions and the intermediate selection of solutions are driven by the concept of dominance [5]. When two solutions are compared, the dominance criterion decides the better solution, taking into account all the problem objectives simultaneously. It can be stated as follows; Solution A is said to weakly dominate solution B if,

1. The solution A is no worse than B in all the objectives.
2. The solution A is strictly better than B in at least one objective.

If either of two competing solutions is not better than the other on the basis of the above criterion they are termed as non-dominated with respect to each other. The concept of dominance is illustrated in Figure 2.

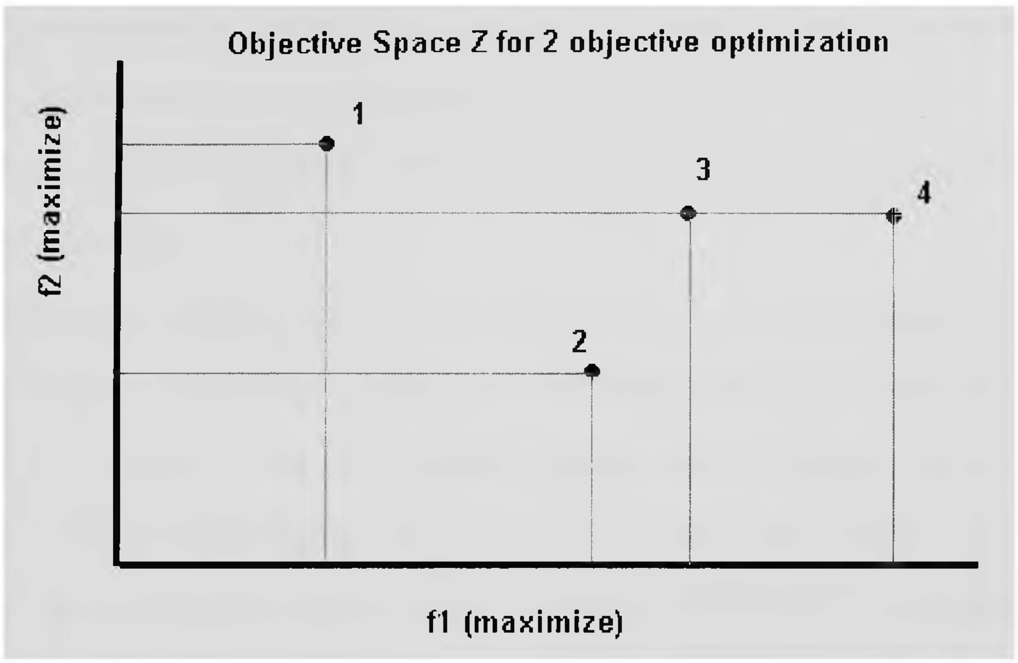

Figure 2 Concept of dominance in a maximization problem

Figure 2 shows that solution 1 is non-dominating with respect to the other three solutions, as it is worse than the other three in objective f1 but better in objective f2. Solution 2 is dominated by solutions 3 and 4 as it is worse than the latter two in both the objectives. Solution 3 is dominated by solution 4 as it is worse than 4 in objective f1 though equal in objective f2. The above figure manifests another important dominance characteristic. When two solutions are independently non-dominated w.r.t. a third solution, it is not necessary that the former two solutions be non-dominated w.r.t to each other. Thus, non-domination demonstrates the apparent comparability of trade-off solutions in case of multiobjective problems.

Many multi-objective optimization algorithms use a population of decision variable sets in search for optimal solutions. This population can be divided into two major sets during any generation.

1. The non-dominated set, which is comprised of solutions that are not dominated by any other solution in the whole population (*i.e.* local optimal solutions), and
2. The dominated set, which is comprised of all the solutions excluded from the non-dominated set.

The solutions belonging to the non-dominated set during a particular generation form a hyper-surface in the objective space, called the Pareto front [5]. The Pareto front manifests as a curve in case of a 2-objective problem, and a 3D surface in case of a 3-objective problem. Solutions which are not dominated by any other solution in the whole feasible space are termed as globally optimal solutions, and the Pareto front constituted of these globally optimal solutions is called the global Pareto front. However, in case of non-conflicting objectives this hypersurface reduces to a single optimal point. Four different forms of Pareto fronts have been illustrated in Figure 3, for a 2-objective problem.

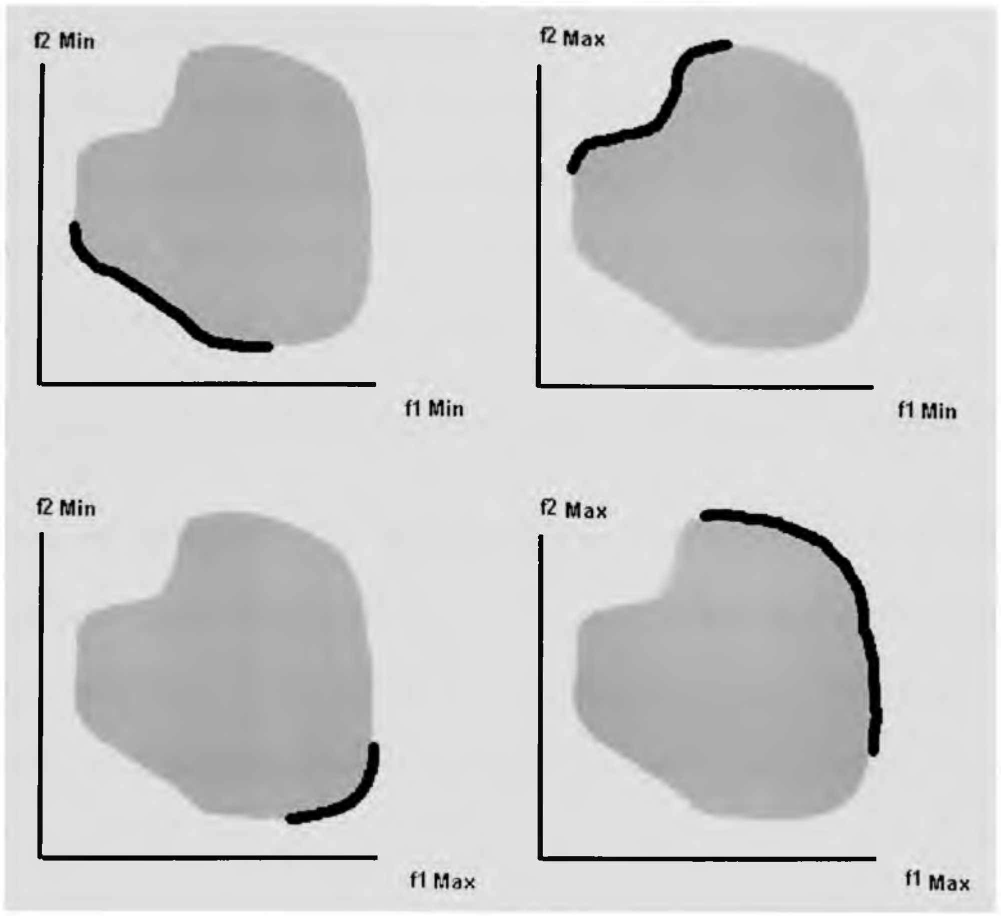

Figure 3 Pareto fronts for different forms of 2-objective optimization problem

In order to have flexibility in an optimal system design it is necessary to compute a set of best compromise solutions which are distinctly biased towards one or more objectives. Consequently, achieving a practically uniform distribution of solutions over the whole span of the global Pareto front is as important as converging to the global Pareto front. Such efficient coverage of the Pareto front demands prominent presence of diversity among the members of the non-dominated population set, which becomes progressively unmanageable with increasing number of objectives.

2.2 Classical Methods and Their Drawbacks

Classical methods which are mostly gradient based search techniques, have long dominated the stage of multi-objective optimization until mid 1990s. One of the most popular classical approaches is the weighted sum method, where the objectives are linearly combined to form a single composite objective function, *i.e.* $f(X)=\sum_{i=1}^{Nf} w_i f_i(X)$. The objectives have to be normalized in this case, which requires some prior knowledge of their range of magnitude. One combination of weights, w_i, yields only one single solution search [5]. Thus, the weighted sum method reduces to multiple explorations for single-objective optimal points, each corresponding to a particular combination of weights. This technique has severe shortcomings, which are as follows.

1. Successful convergence to a point on the global Pareto front depends on the selection of the initial solution.
2. Diversity among the Pareto solutions is highly sensitive to the user's choice of weights.
3. this approach has an inherent tendency to converge to sub-optimum solutions (local optima), especially in case of multi modal problems.
4. Are unable to handle problems with non convex Pareto fronts.
5. Are unable to handle problems with discontinuous search space.
6. Computational cost escalates exponentially with increasing number of decision variables and increasing non-linearity of the objective functions or constraint functions.

7. Due to lack of communication between different solutions, these classical algorithms rarely benefit when run on parallel machines.

2.3 Evolutionary Multi-Objective Optimization Algorithms (EMOs)

Evolutionary algorithms provide a stochastic approach towards optimization. They mimic the principles coined by Darwin and Mendel, which is, evolution occurs through selection and adaptation [11]. Evolutionary algorithms initiate with a randomly generated population of candidate solutions that evolve (improve) over generations through activities such as 'selection', 'recombination/crossover' and 'mutation'. The decision variable values associated with a solution constitute the genotype of the solution, and the corresponding objective function values for that solution constitute the phenotype. Depending upon the number system in which the genotype is represented, an evolutionary algorithm can be a binary-coded or a real-coded algorithm.

Since EMOs operate with a population of solutions, the outcome at the end of each generation is also a population of solutions which gives them the ability to converge to the multiple optimal solutions in one single simulation run. Ample communication between the solutions with different genotype leads to more efficient coverage of the problem space. EMO algorithms can run on parallel machines for design optimization of complex systems/processes. There exist different types of multi-objective hybrid optimizers that make use of such parallel computing. Two such well known multi-objective hybrid optimizers are MOHO [12] developed by Moral and Dulikravich and AMALGAM [13] developed by Vrugt and Robinson, both of which apply a combination of more than one EMO to solve a multi-objective optimization problem. MOHO (Multi-

Objective Hybrid Optimization software) consists of (i) Strength Pareto Evolutionary Algorithm (SPEA-2), (ii) Multi-Objective Implementation of the Single-Objective Particle Swarm algorithm (MOPSO) and (iii) Non-Sorting Differential Evolution (NSDE). They are applied in series controlled by a built in automatic switching algorithm that swaps the operating optimization algorithm based on several performance criterion w.r.t the problem being solved. On the other hand, in AMALGAM (Genetically Adaptive Multi-Objective Method), each constituent algorithm is employed in parallel, which contribute a portion of the next generation's population. The extent of the share (of population) for each contributing algorithm is dependent on their success of solution exploration in the preceding generations, w.r.t. the problem being solved.

The Modified Predator-Prey (MPP) algorithm is an EMO that imports certain features of the weighted sum approach as well. In addition MPP was enabled to deal with both linear/non-linear equality and inequality constraints.

2.4 Modified Predator-Prey (MPP) Algorithm [14]

Any general constrained multiobjective problem involving Nf objectives and m design variables can be reformulated as follows.

$$\begin{aligned}
&\text{Minimize } f_i = f_i(X), \quad i = 1,2,\dots,Nf \\
&\text{subject to} \\
&g_i \le 0, \quad i = 1,2,3,\dots,p \\
&h_i = 0, \quad i = p+1, p+2,\dots,p+q \\
&p,q \in N
\end{aligned} \tag{2}$$

Where, X is the design vector i.e. $X = (x_1, x_2, x_3, \dots, x_m)$,$x_i \in R$

The constraints are added up to form the $(Nf+1)^{\text{th}}$ objective in the following way,

$$\text{Minimize } f_{Nf+1} = \sum_{i=1}^{p} \max(g_i, 0) + \sum_{i=p+1}^{p+q} \max((h_i - \varepsilon), 0) \qquad (3)$$

where ε is the tolerance for equality objectives.

It should be noted that in case of maximization the corresponding objective function is multiplied by '-1', to convert it into a general minimization problem. Also, a 'greater than equal to' inequality constraint is converted into a 'less than equal to' constraint by multiplying with '-1'.

The overall structure of the modified predator-prey algorithm developed in this study is presented below in sequential steps.

1. A population of N candidate solutions/preys ("antelopes") are initialized using Sobol's [15] quasi random sequence generator.
2. The preys are placed on a two dimensional grid with connected ends hence having a toroidal nature as shown in figure 4. The grid is allowed to adjust its size dynamically according to the prey population size maintaining the dimensions I x J, where typically J = 5. Random members of the prey populations are cloned and placed on the grid when $N < I \times J$ in order to ensure all integer grid points are occupied by preys.

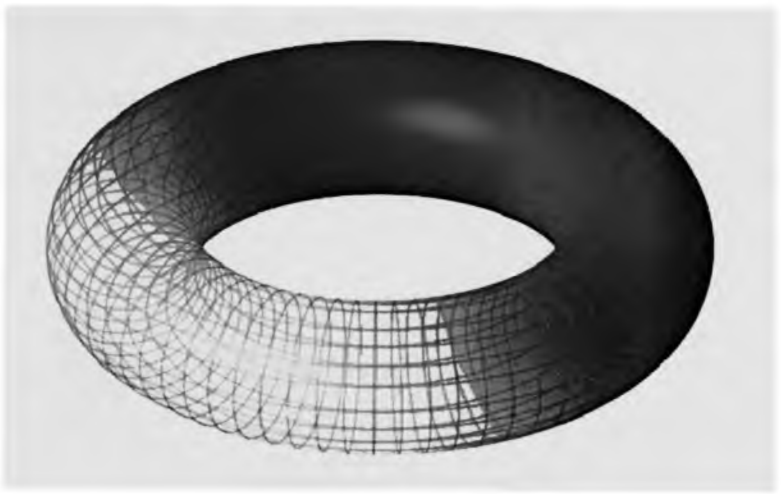

Figure 4 Toroidal grid – 2D grid wrapped around in both directions

3. M number of predators ("lions") is placed on the same 2D grid such that they occupy random cell centers. M is determined by the following empirical formula,

$$M = \left\lceil \frac{N}{20} \right\rceil \times Nf \qquad (4)$$

where, $\lceil r \rceil$ is the lowest integer greater than r, $r \in R^+$. Each predator is associated with a weighted value of the objectives as follows.

$$f = \sum_{i=1}^{Nf} w_i f_i \qquad (5)$$

Here, w_i is the weight associated with the ith objective function, f_i is the ith objective function. The weights are distributed uniformly in case of two-objectives problems ($0 \leq w_1 \leq 1$, $w_2 = 1 - w_1$) and using Sobol's quasi random sequence generator [15] in case of problems with more than two objectives.

4. Predators are randomly located in the toroidal grid. Each neighborhood that contains a predator can be termed as an 'active locality' as shown in figure 5. In each of these localities/cells, the value of 'f' as defined by equation (2) corresponding to the local

predator, is calculated for each prey. The weakest prey (*i.e.* having the maximum value of f) is selected to be killed and replaced by a new prey produced by the crossover of the two strongest local preys and subsequent mutation of the crossover child.

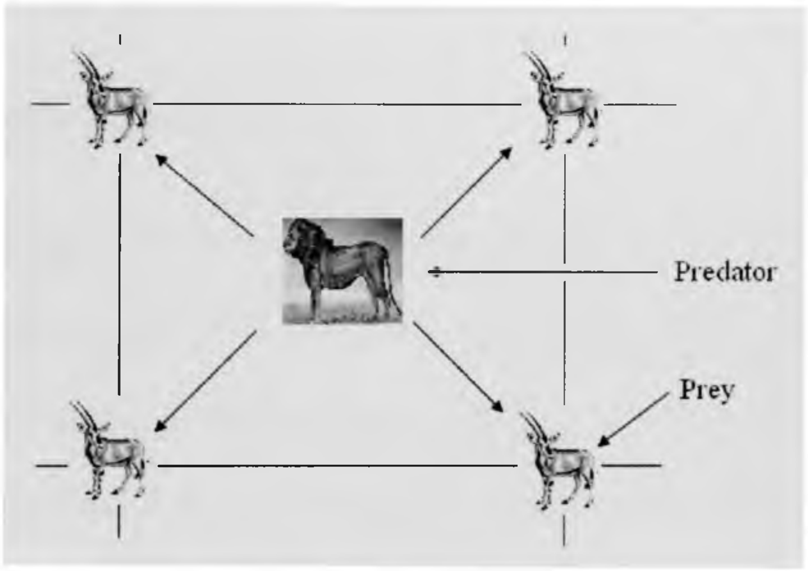

Figure 5 An active 4 prey locality/neighborhood in the toroidal grid

5. However, this phylogenetic child prey qualifies to be accepted only if it fulfills the following three criteria,

   (i) The child is stronger than the worst local prey (based on f calculated by equation 2),

   (ii) The child is non-dominated [5] with respect to the other three local preys, and

   (iii) the child is not within the objective space hypercube [5] of the other three preys of this locality.

Ten trials are allowed to produce a qualified child that satisfies the three criteria, failing which the weakest prey is retained.

6. Upon completion of the above predator-prey interactions in each active locality, the predators are relocated randomly. A probability based relocation criterion has been introduced here, which favours a fairly even distribution of the 'number of visitations' to each cell/locality by a predator. The relocation criterion is defined as follows:

$$\begin{array}{ll} \text{if } \mathit{cellcount}(i,j) > \mathit{cellcount}_{avg}+1, & \mathit{locate}=\mathit{no} \\ \text{else} & , \mathit{locate}=\mathit{yes} \end{array} \tag{6}$$

   Here, $\mathit{cellcount}(i,j)$ is the cumulative number of times predators have visited the cell (i,j) in previous generations, $\mathit{cellcount}_{avg}$ is the average of all $\mathit{cellcount}(i,j)$ and (i,j) is the randomly generated location on the 2D lattice. This new feature ensures that every member of the prey population irrespective of its fixed location in the 2D lattice gets a reasonable opportunity of improvement.

7. After each generation, the non-dominated solutions in the prey population are copied to a secondary set called the 'elite set'. Certain number of randomly selected elite solutions are incorporated into the main population (preys in the toroidal space) at the cost of some of the dominated solutions (dominated by atleast one other prey).

Specific features that have been modified or added to the PP algorithm during this research in order to enhance its efficiency and applicability are as follows.

2.4.1 Evolution: The generation of new solutions in each active locality is initiated by the crossover of the strongest two local preys (with respect to the corresponding f value). The blend crossover (BLX-α), initially proposed by Eshelman and Schaffer [16] for real-coded genetic algorithms (later improved by Deb [5]), is used in this algorithm. It is defined as follows,

$$\begin{aligned} x_i^{(1,t+1)} &= (1-\gamma_i)\,x_i^{(1,t)} + \gamma_i x_i^{(2,t)} \\ \gamma_i &= (1+2\alpha)u_i - \alpha \end{aligned} \tag{7}$$

where, $x_i^{(1,t)}$ and $x_i^{(2,t)}$ are the parent solutions, $x_i^{(1,t+1)}$ is the child solution and u_i is the random number between 0 and 1. A value of 0.5 is used for α as suggested by Deb [5]. With BLX-α, the location of the offspring in the decision space depends upon the difference between the parent solutions [5]. This facilitates genetic recombination that is adaptive to the existing diversity in the parent population; a desirable characteristic for Pareto convergence.

This crossover child prey is then subjected to non-uniform mutation originally introduced by Michalewicz [17], and mathematically formulated as,

$$\begin{aligned} \beta &= 10^{-\left(t/t_{\max}\right)} \\ y_i^{(1,t+1)} &= x_i^{(1,t+1)} + \tau\left(x_i^{(U)} - x_i^{(L)}\right)\left(1 - r_i^{\left(1-t/t_{\max}\right)^b}\right) \end{aligned} \tag{8}$$

where $y_i^{(1,t+1)}$ is the child solution produced from the parent solution $x_i^{(1,t+1)}$, by mutation of the ith variable, $x_i^{(U)}$ and $x_i^{(L)}$ are upper and lower limits of the ith variable, τ takes a Boolean value -1 or 1, each with a probability of 0.5, r_i is a random number between 0 and 1, t and $t_{\max}$ are the number of generations already executed and the maximum allowed number of generations, respectively, while b is a user defined parameter.

Non-uniform mutation favours creation of child solutions in the vicinity of the parent solution, and the probability of creating a child solution closer to the parent increases with increasing number of generations. This provides a uniformly distributed search in the earlier generations and a relatively focused search in the later ones. A modified version of this non-uniform mutation has been applied in MPP, which is as follows,

$$\beta = 10^{-\left(t/t_{\max}\right)}$$
$$y_i^{(1,t+1)} = x_i^{(1,t+1)} + \tau\left(x_i^{(U)} - x_i^{(L)}\right)\left(1 - r_i^{\left(1-t/t_{\max}\right)^b}\right) \times \beta \tag{9}$$

Here t and $t_{\max}$ are the number of function evaluations performed until then and maximum allowed number of function evaluations, respectively, (b = 1.5 determined empirically) and β is the scaling parameter. The latter two factors monitor the order of magnitude, or in other words, the extent of mutation.

Both the crossover and mutation techniques employed here establish an adaptive search, which makes the MPP algorithm more economical with respect to function evaluations.

2.4.2 Dominance and Constraint handling: The concept of weak dominance [5] is applied here, according to which in case of an unconstrained optimization problem, solution i is said to weakly dominate solution j if solution i is better than solution j in atleast one objective and equal in all other objectives. However, in case of a constrained optimization, the theory of dominance is altered to give preference to feasible solutions or relatively less infeasible solutions. The modified definition of dominance is the same as used in NSGA-II [18], which is as follows,

Solution i is said to constraint-dominate solution j if

1. Solution i is feasible and solution j is not.

2. Solutions i and j are both infeasible, while solution i has a smaller net constraint violation than solution j, *i.e.* $f_{Nf+1}{}^{i} < f_{Nf+1}{}^{j}$ (considering function minimization).
3. Solutions i and j are both feasible, while solution i weakly dominates solution j.

Due to the absence of any penalty function method, the normal objectives (f_k, $\forall k < Nf + 1$) and the net constraint violation objective (f_{Nf+1}), get similar quantitative importance. This, together with the constraint-dominance criterion, favour feasible solutions, but also helps retain genetic traits of infeasible solutions with substantially better objective values as well. This speeds up convergence to the Pareto front especially when it is located at the boundary of the feasible region. Nevertheless, it should be noted that unless the whole prey population lies in the infeasible region (in the objective space) the progressing Pareto front will always constitute of feasible solutions, because the Pareto front is formed by the non-dominated elite solutions.

2.4.3 Diversity Preservation: A multiobjective problem prefers a reasonably uniform distribution of solutions along the whole span of the Pareto front. This calls for preservation of diversity in the objective space. In other words, an efficient multi-objective optimization algorithm is expected to promote generation of new solutions (evolution) that do not closely resemble their parents or other nearby solutions (in the objective space). Here, the concept of objective space hypercube is used as a qualifying criterion for new preys to assure diversity preservation. Each old local prey is considered to be at the centre of its hypercube, the size of which is dynamically updated with generations and could be determined by the following equation [14]

$$\begin{aligned} &\omega = 10^{-\left(2+ t/t_{\max}\right)} \\ &\eta_i = \omega \times \min\left(f_i^{new\ prey}, f_i^{old\ prey}\right) \end{aligned} \tag{10}$$

Here, ω is the window size of the hypercube and η_i is the half side length of the hypercube corresponding to the ith objective.

2.4.4 Sectional Convergence (Biased Weighing of Objectives): A prominent drawback of the original predator-prey algorithm is its tendency to converge to a small section of the Pareto front due to absence of local selection pressure chiefly based on non-dominance. A new and innovative concept of sectional convergence has been introduced [14] to deal with this possible lack of effective variation in the prey population. Instead of the running the algorithm throughout for the same initial specified distribution of weights, there is redistribution of weights within a small biased range (<1.0) after certain number of function evaluations. The redistribution is governed by the following equations in case of two-objective optimization problems.

$$\begin{aligned} &w_1^i = \frac{(iterp-1)M+i}{iterp_{\max}M+1} \\ &w_2^i = 1-w_1 \end{aligned} \tag{11}$$

Here, $iterp_{\max}$ is the maximum allowed number of primary iterations, *i.e.*, maximum number of times redistribution is allowed, $iterp$ is the present primary iteration, and i is the ith predator. In case of multi-objective optimization with more than two objectives, a different formula could be used [14] as shown below.

$$j = \frac{iterp - 1}{\frac{iterp_{\max}}{Nf}} + 1$$

$$w_{k\max}^{i} = \frac{\left(iterp - \frac{iterp_{\max}}{Nf}(j-1) - 1\right)M + i}{\frac{iterp_{\max}}{Nf}M + 1} \times 0.75 \tag{12}$$

$$w_{j}^{i} = 1 - w_{k\max}^{i}$$

Here, w_j^i is the weight associated with the jth objective function for the ith predator and $w_{k\max}^{i}$ is the maximum allowable weight associated with any kth objective function ($k \neq j$) for the ith predator. The weights (w_k^i) associated with the objective functions other than the jth objective are distributed using Sobol's [15] within the range $0 - w_{k\max}^{i}$. However in this case (i.e. problems with $Nf > 2$), $\frac{iterp_{\max}}{Nf} \in N$ is an essential condition.

This added feature involving biased distribution of weights does away with the often observable drawback of PP which is its tendency to converge to a small section of the Pareto due to absence of selection pressure chiefly based on non-dominance. Nevertheless, such sectional convergence comes at the cost of an increased number of function evaluations which might be necessary only in case of complex problems such as sharp discontinuities or mixed convex-concave Paretos or orders of magnitude difference between the objective functions.

2.4.5 Elitism: In order to retain the genetic traits of the best solutions it is necessary to introduce some form of elite preservation mechanism into the algorithm. This, when judiciously applied, accelerates the rate of convergence to the Pareto front. In MPP the secondary set (elite set) consisting of the non dominated solutions from each generation is maintained at a fixed strength Ne using the clustering technique designed by Deb [5]. After each generation, certain randomly selected solutions/preys (from the main population), if found to be dominated, are replaced from the 2D lattice by randomly selected elite solutions. This new additional attribute boosts the speed of convergence of this algorithm. However, the allowed number of such replacements should be carefully chosen to avoid introducing excessive elitism. Here the total number of allowed replacements is always kept below $N/2$.

2.4.6 Additional Features: During the course of development of MPP a few other alterations/additional features were also implemented, but not included in the final version of the algorithm. This was due to certain drawbacks associated with each one of them. A couple of them are being presented here, keeping in mind that a more judicious application of any of these features, in the future, might help to improve the dependability or performance of MPP or other similar evolutionary optimization algorithms. They are as follows:

- Controlled killing in active localities: Instead of killing exactly one prey (the weakest) at each active locality during a generation, the predator was allowed to kill 'κ' number of the weakest local preys depending on the 'non-domination' quality of the locality. The value of κ for each locality was computed according to the following formula,

$$\kappa = \begin{cases} 0 & \text{if } ne \geq 3 \\ 1 & \text{if } 3 > ne > 0 \\ 2 & \text{if } ne = 0 \end{cases} \tag{13}$$

where ne is the number of preys from that locality that qualified for the elite set when last updated ($ne \in \{1,2,3,4\}$). This reduced the required number of function evaluations, but severely hindered further progress when solutions converged to a local Pareto front.

- Relocating preys: Like predators, preys were also relocated randomly within the same 2D lattice after every 'nm' iterations, where $nm = \frac{N}{M}$. A favourable genetic mixing was observed, leading to greater diversity, but at the cost of noticeably increased number of function evaluations.

2.5 Numerical Experiments

MPP was implemented using C++ programming language. The objective functions were evaluated by the corresponding executable files. The C++ code simulating MPP is known as 'mpp_cnstrnt.cpp'. It compiles and runs successfully on both Windows and Linux workstations using Microsoft Visual C++ .NET for the former and KDevelop 3.1.1 for the latter operating systems.

2.5.1 Unconstrained 2-Objective Test Cases: MPP was tested to evaluate its performance by running it on some well known unconstrained two-objective test problems, with known analytical solution. The first six test cases analyzed are taken from the multi-objective optimization comparison by Zitzler *et al.* [1] namely the ZDT test cases. Two other popular test cases with known analytical solutions for the Pareto front which are the Fonseca and Fleming multiobjective problem no. 2 [19] and the Coello multiobjective problem [11] have also been used. All the eight test cases involve two-objective optimizations where both objectives are to be minimized. They are summarized in Table 1.

Table 1 Details of the unconstrained 2-objective optimization test cases.

Problem	m	Variable limits	Objective Functions	Analytical Solution
ZDT1	30	$x_i \in [0,1]$	$f_1 = x_1$ $g = 1 + 9\sum_{i=2}^{m} \frac{x_i}{m-1}, \quad h = 1 - \sqrt{\frac{f_1}{g}}$ $f_2 = h.g$	Set $g = 1$
ZDT2	30	$x_i \in [0,1]$	$f_1 = x_1$ $g = 1 + 9\sum_{i=2}^{m} \frac{x_i}{m-1}, \quad h = 1 - \left(\frac{f_1}{g}\right)^2$ $f_2 = h.g$	Set $g = 1$
ZDT3	30	$x_i \in [0,1]$	$f_1 = x_1$ $g = 1 + 9\sum_{i=2}^{m} \frac{x_i}{m-1}$ $h = 1 - \sqrt{\frac{f_1}{g}} - \left(\frac{f_1}{g}\right)\sin(10\pi f_1)$ $f_2 = h.g$	Set $g = 1$
ZDT4	10	$x_1 \in [0,1]$ $x_i \in [-5,5]$	$f_1 = x_1$ $g = 1 + 10(m-1)$ $+\sum_{i=2}^{m}\left(x_i^2 - 10\cos(4\pi x_i)\right)$ $h = 1 - \sqrt{\frac{f_1}{g}}$ $f_2 = h.g$	Set $g = 1$
ZDT5	11	$x_1 \in [0,1]$, 30 bit resolution $x_i \in [0,1]$, 5 bit resolution	$f_1 = 1 + u(x_1)$ $u(x_i)$ = the number of ones in the bit vector form of x_i	Set $g = 10$

			$g=\sum_{1=2}^{m} v(u(x_i)),\ h=\frac{1}{f_1}$ $v(u(x_i))=\begin{cases}2+u(x_i) & \text{if } u(x_i)<5\\ 1 & \text{if } u(x_i)=5\end{cases}$ $f_2=h.g$	
ZDT6	10	$x_i\in[0,1]$	$f_1=1-e^{-4x_1\sin^6(6\pi x_1)}$ $g=1+9\left(\frac{\sum_{i=2}^{m} x_i}{m-1}\right)^{0.25},\ h=1-\left(\frac{f_1}{g}\right)^2$ $f_2=h.g$	Set $g=1$
Fonseca-Fleming	3	$x_i\in[-4,4]$	$f_1=1-e^{\left(-\sum_{i=1}^{m}\left(x_i-\frac{1}{\sqrt{m}}\right)^2\right)}$ $f_2=1-\exp\left(-\sum_{i=1}^{m}\left(x_i+\frac{1}{\sqrt{m}}\right)^2\right)$	$f_2=1-e^{-\left(2-\sqrt{-\ln(1-f_1)}\right)^2}$
Coello	2	$x_i\in[0,1]$	$f_1=x_1$ $f_2=(1+10x_2)\left(1-\left(\frac{x_1}{1+10x_2}\right)^2-\frac{x_1}{1+10x_2}\sin(8\pi x_1)\right)$	$f_2=1-f_1^2$ $-f_1\sin(8\pi f_1)$

[Note: m = number of variables]

To compensate for performance fluctuations induced by random generators driving the initial population and other genetic operators, the algorithm was run 30 times for 25,000 function evaluations each in case of the six ZDT test cases and 2,000 function evaluations each in case of the Fonseca-Fleming and Coello test problems. The concept of sectional convergence was not implemented during these runs. The non-dominated plots are generated by making a union of the elite set (non-dominated set) of the first five runs for each test case. The non-dominated set of the unions is then extracted and plotted as shown in figures 6-13. The user defined MPP parameters used for these test cases were as follows.

Table 2 User defined MPP parameters for unconstrained 2-objective test cases.

Parameter	*Value*
Population size (# preys)	100
# Predators	10
Elite strength	40
Crossover probability	1.0
Mutation probability	0.05

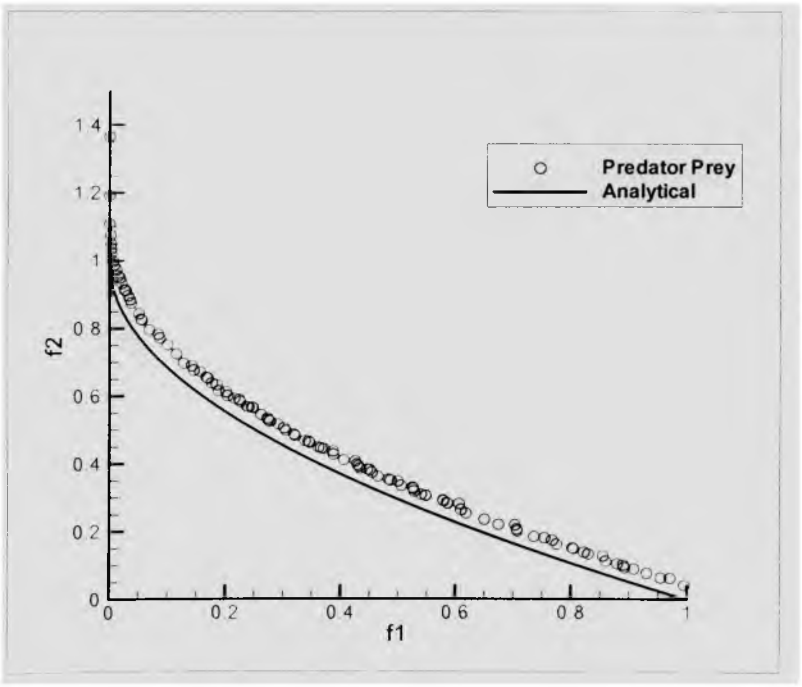

Figure 6 Two-objective test case ZDT 1

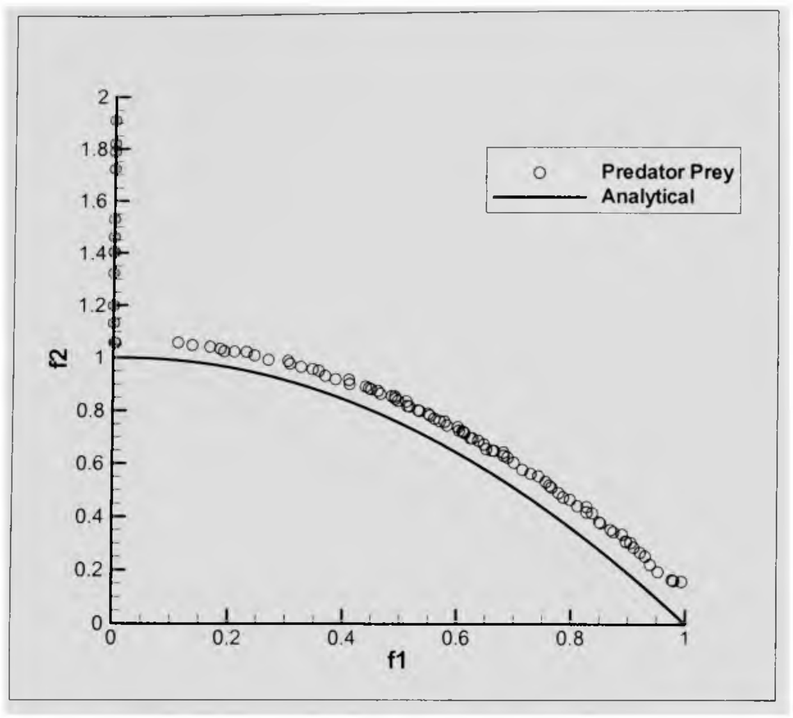

Figure 7 Two-objective test case ZDT 2

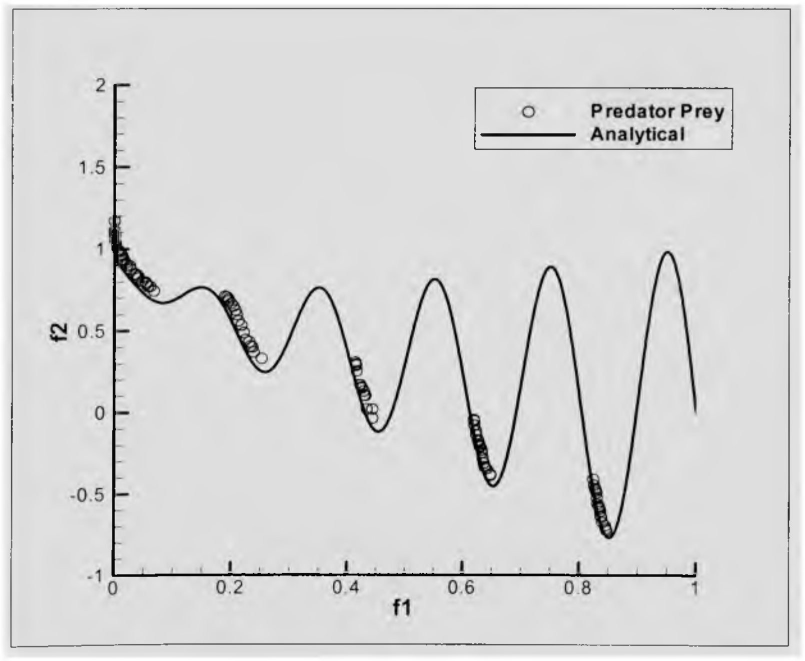

Figure 8 Two-objective test case ZDT 3

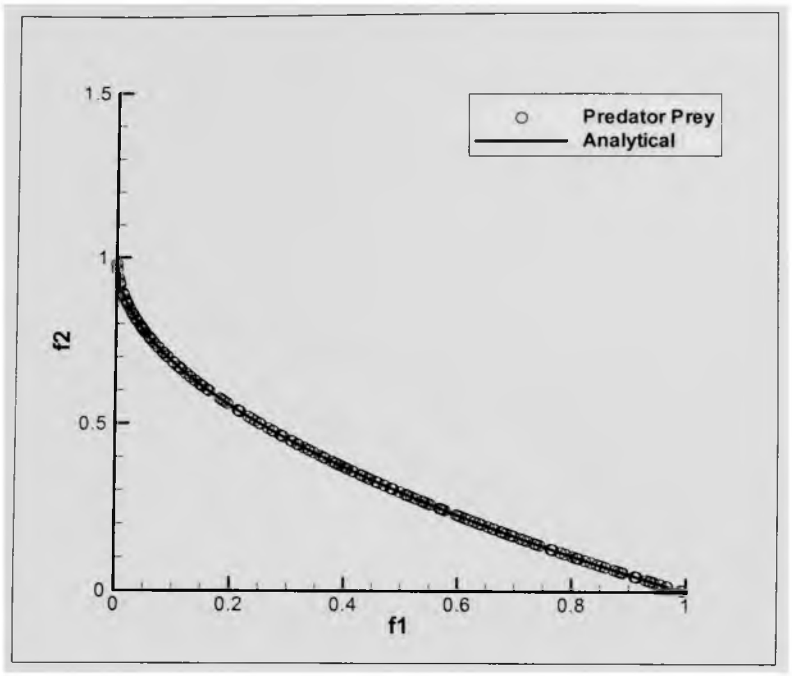

Figure 9 Two-objective test case ZDT 4

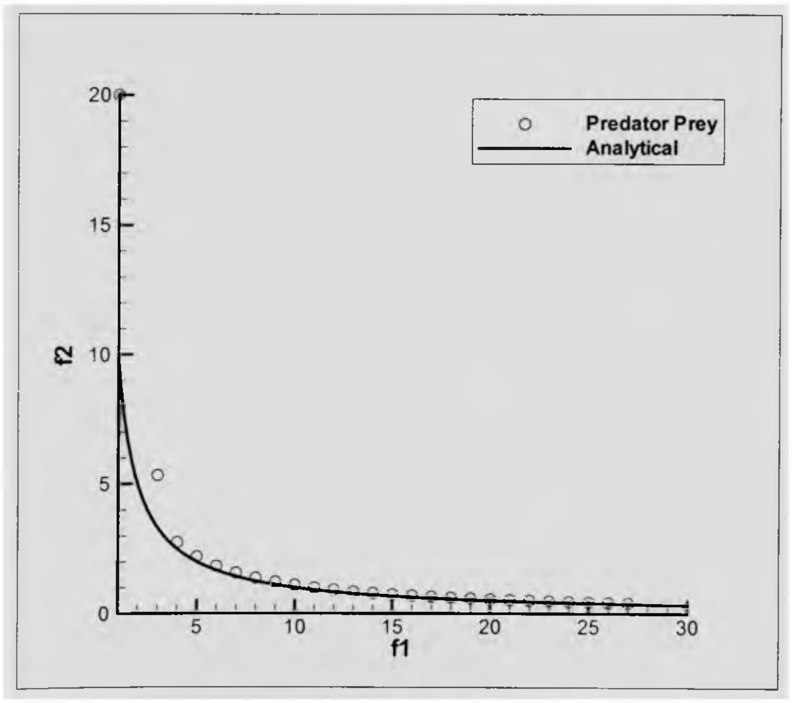

Figure 10 Two-objective test case ZDT 5

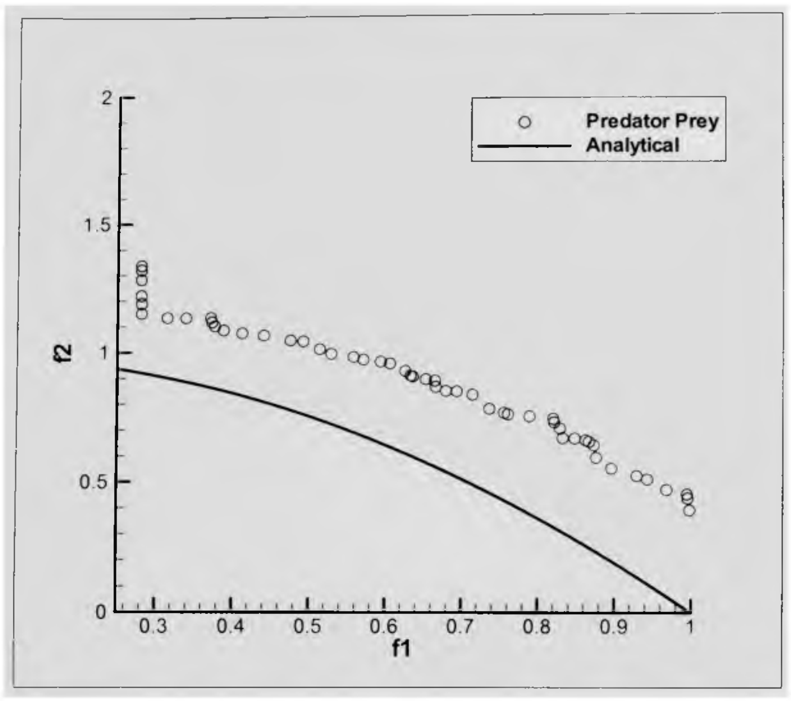

Figure 11 Two-objective test case ZDT 6

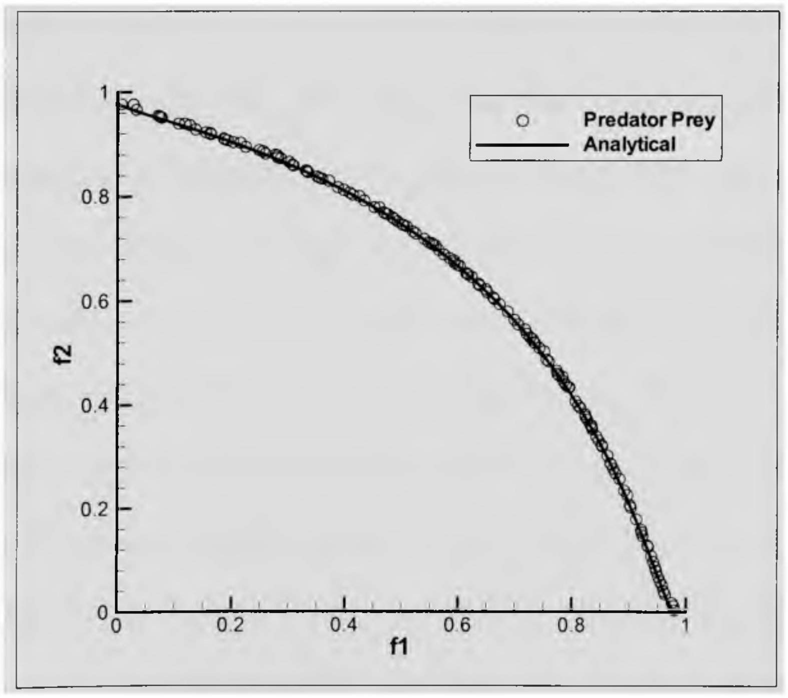

Figure 12 Two-objective test case of Fonseca & Fleming problem 2

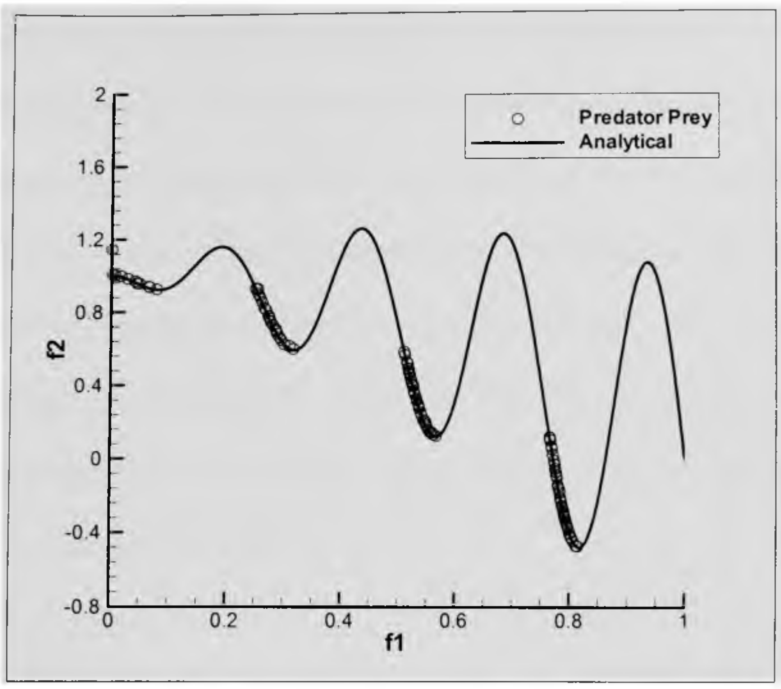

Figure 13 Two-objective test case of Coello

It is observed from figures 6-13, that MPP performs very well on the ZDT test cases when compared with the performances of some other well known algorithms as shown by Moral *et al.* [12] (given in Appendix A) as well as with one of the more popular previous versions of the predator-prey algorithm (given in Appendix B – conditions being much relaxed). The same is exhibited in case of the Fonseca-Fleming and Coello test problems as shown in figures 12 and 13 respectively. In certain cases as in ZDT 1, 2 and 6 the solutions do not completely converge to the global Pareto. This is due to significant slowing down of the rate of convergence as the solutions approach the global Pareto. Nevertheless, it is evident from the above figures that the algorithm consistently produces a desirable spread of non-dominated solutions irrespective of the nature of the Pareto front and without using the concept of sectional convergence.

Two performance measures for evaluating the performance of multiobjective optimization algorithms have been developed by Deb *et al.* [18]. The first performance metric, the gamma (γ) parameter, is a measure of the extent of convergence. The minimum of the Euclidean distances of each computed non-dominated solution from H uniformly distributed points on the ideal Pareto front (H=500) is calculated, the average of which gives the value of the gamma parameter. The other performance metric, namely the delta (Δ) parameter, gives a measure of the spread of solutions along the computed Pareto front. It is calculated as follows,

$$\Delta = \frac{d_f + d_l + \sum_{i=1}^{N-1} \left| d_i - \bar{d} \right|}{d_f + d_l + (N-1)\bar{d}} \qquad (14)$$

where, d_f and d_l - the respective Euclidean distances between the two extreme solutions and the corresponding extremities of the analytical Pareto front, d_i- Euclidean distance between consecutive solutions and $\bar{d}$ - mean of all d_i (i = 1,2,3..., N). A perfectly uniform distribution of solutions along the computed Pareto front with existence of exact extreme solutions will give a delta value of zero. However, inspite of accurate convergence, the gamma parameter need not be zero, due to possible lack of coincidence of computed solutions and uniformly distributed analytical Pareto points.

Table 3 shows the values of these two parameters calculated for the eight cases studied here, and also the comparison of some them with that calculated by Deb *et al.* [18] for NSGA-II. The same conditions have been used, i.e. a population of 100 solutions, subjected to 25000 function evaluations, for the six ZDT test cases. However, the Fonseca-Fleming and the Coello test cases involve 2000 function evaluations and hence the former has not been compared with the corresponding data of Deb *et al.* [18], all of which are with respect to 25000 function evaluations.

Table 3 Performance parameters

Algorithm	NSGA-II (real)		NSGA-II (binary)		MPP	
Parameter / Problem	γ	Δ	γ	Δ	γ	Δ
ZDT 1	0.0335	0.39	0.0009	0.46	0.0447	0.59
ZDT 2	0.0724	0.43	0.0009	0.44	0.1181	0.78
ZDT 3	0.1145	0.73	0.0434	0.58	0.0198	0.73
ZDT 4	0.5130	0.70	3.2276	0.48	0.6537	1.48
ZDT 5	NA	NA	NA	NA	0.4282	1.49
ZDT 6	0.2966	0.67	7.8068	0.64	0.2334	0.71
Fonseca-Fleming	NA	NA	NA	NA	0.0082	0.42
Coello	NA	NA	NA	NA	0.0498	1.17

As seen from table 3, the performance of MPP compares well with that of real coded NSGA-II, except in the case of ZDT 2. The latter may be attributed to the vertical congregation of points near the left boundary of the Pareto front where an abrupt change in the value of f_2 corresponding to very small values of f_1 poses difficulty in properly distributing ideal Pareto points in this region. However, in the case of ZDT 3, the MPP seems to outperform both the real coded and the binary coded NSGA, in accuracy. As seen from figure 9, a fairly accurate and well distributed non-dominated solution set is computed by MPP in the case of ZDT 4. Due to the high density of solutions along the computed Pareto front, the deviation in d_is exceed the average, $\bar{d}$, which accounts for the relatively high value of Δ (>1), calculated in case of ZDT 4.

Difficulties encountered in converging to the ideal Pareto front in the case of ZDT 5, by other standard optimization algorithms have been claimed to be not trivial, as also confirmed by Deb *et al.* [18]. However during the course of this study, it has been found that achieving acceptable accuracy in the case of ZDT 5 to be relatively manageable as evident from figure 10 and table 3. But the above is true only when the correct order of precision is used in representing the decision variables and computing the objective functions. Failure to do so might be the very reason behind the relatively low accuracy of solutions computed by other optimization algorithms while dealing with ZDT 5.

Test case results presented in this work are generated without considering the concept/module of sectional convergence. However, sectional convergence was experimented on during the study of MPP and a visual representation is exhibited in figure 9.

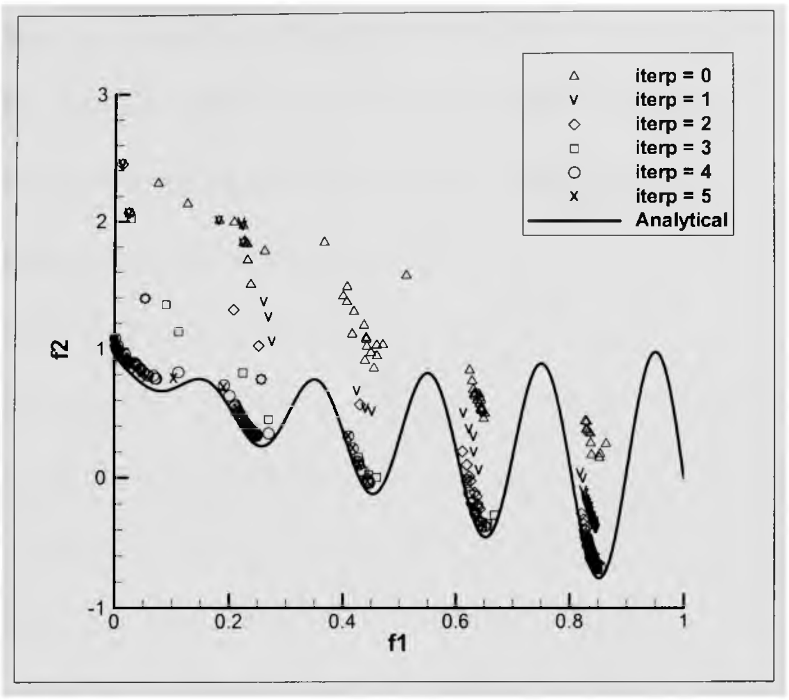

Figure 14 Sectional convergence for ZDT 3 ($iterp_{\max} = 5$)

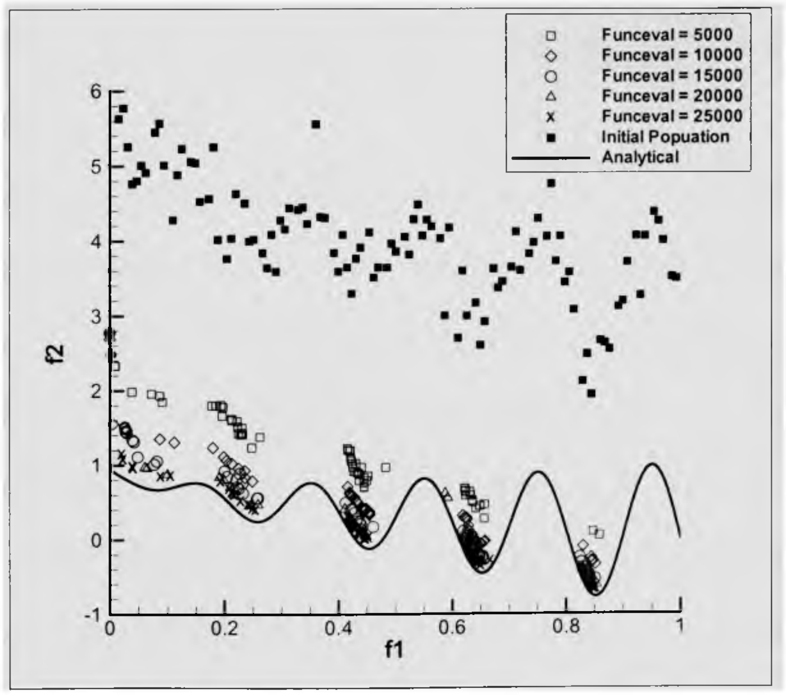

Figure 15 General convergence for ZDT 3 ($iterp_{\max} = 0$)

Figure 14 shows the location of solutions in the objective space at the end of each primary iteration ($iterp$), where $iterp = 0$ represents the initial global progression of solutions, and $iterp > 0$ represents the sequential sectional convergence of solutions to parts of the Pareto. It is observed that though the progress of solutions is biased towards sections of the objective space going from right to left, the solution set as a whole always keeps moving towards the ideal Pareto. This is desirable and eventually leads to a well distributed set of non-dominated solutions along the final computed Pareto front.

Figure 15 shows the progress of solutions towards the ideal Pareto, in absence of the sectional convergence module. The solutions are plotted after intervals of 5000 (approx.) function evaluations. It is observed that the solutions converge noticeably faster during the initial stages of MPP to form an intermediate Pareto. The subsequent progress of this intermediate front becomes more and more exhaustive in terms of function evaluations as it nears the global Pareto front.

2.5.2 Unconstrained 3-Objective Test Cases: Multi-objective optimization algorithms often demonstrate different behavior when working on problems with more than two objectives. The Pareto front is just a planar curve in two-objective problems which proliferates into a surface in three-objective problems, and then to a hypersurface of increasing dimensionality with every additional problem objective. This intensifies the necessity for careful preservation of diversity. Selection procedure based on either weighted sum of objectives and weak domination criterion work very differently. For example, say in the case of a problem with Nf objectives ($Nf > 2$), solution A has one objective better than solution B, while in all other objectives solution B ranks higher. Weighted sum would most likely recognize solution B as the better solution whereas according to the principles of weak dominance both solutions are non-dominated w.r.t. each other. Predator-Prey is unique in utilizing the principles of both selection procedures. However, the performance gain of such a characteristic can be appreciated only when the algorithm is tested on optimization problems with more than two objectives. Therefore, MPP is tested on two standard scalable 3-objective minimization problems developed by Deb *et al.* [2]. They are summarized in Table 4.

Table 4 Details of the unconstrained 3-objective optimization test cases

Problem	m	Variable limits	Objective Functions	Analytical Solution
DTLZ1	7	$x_i \in [0,1]$	$g = 100\left(5 + \sum_{i=3}^{7}\left(\begin{array}{l}(x_i - 0.5)^2 \\ -\cos(20\pi(x_i - 0.5))\end{array}\right)\right)$ $f_1 = \frac{1}{2}x_1 x_2(1+g)$ $f_2 = \frac{1}{2}x_1(1-x_2)(1+g)$ $f_3 = \frac{1}{2}(1-x_1)(1+g)$	$x_j = 0$ $j = 3,4,..,7$ $\sum_{k=1}^{3} f_k = 0.5$
DTLZ2	12	$x_i \in [0,1]$	$g = \sum_{3}^{12}(x_i - 0.5)^2$ $f_1 = (1+g)\cos(x_1\pi/2)\cos(x_2\pi/2)$ $f_2 = (1+g)\cos(x_1\pi/2)\sin(x_2\pi/2)$ $f_3 = (1+g)\sin(x_1\pi/2)$	$x_j = 0.5$ $j = 3,4,..,12$ $\sum_{k=1}^{3} f_k^2 = 1$

Due to similar reasons as in case of the ZDT test cases, both DTLZ test cases were run 30 times, 30000 function evaluations each, and the final Pareto front is formed by extracting the non dominated set from the union of the final elite sets of the first five runs. The user defined parameters specified in the algorithm for these test cases are presented in table 5.

Table 5 General MPP parameters for unconstrained 3-objective test cases

Parameter	*Value*
Population size (# preys)	100
# Predators	10
Elite strength	40
Crossover probability	1.0
Mutation probability	0.05

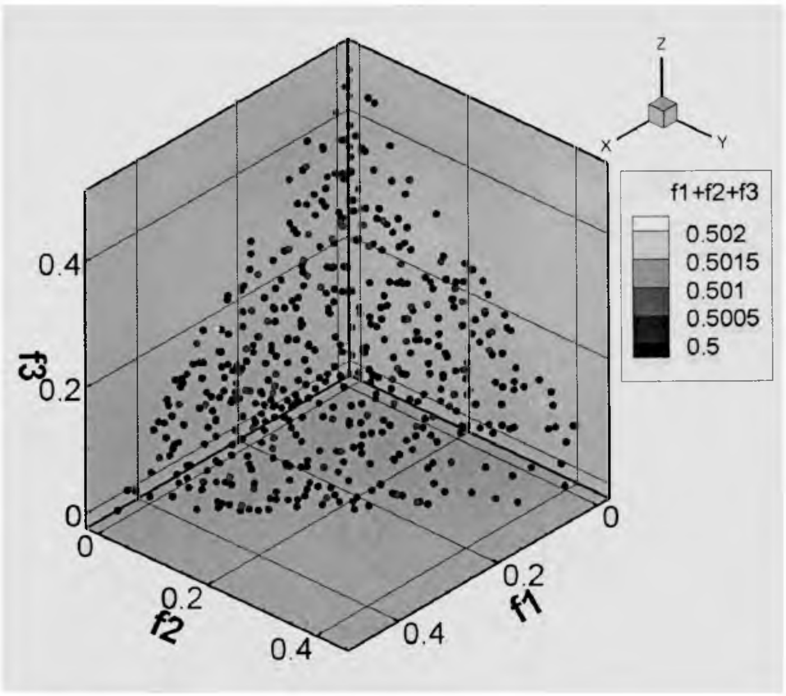

Figure 16a 3-objective test case DTLZ1 with $iterp_{\max} = 0$: view 1

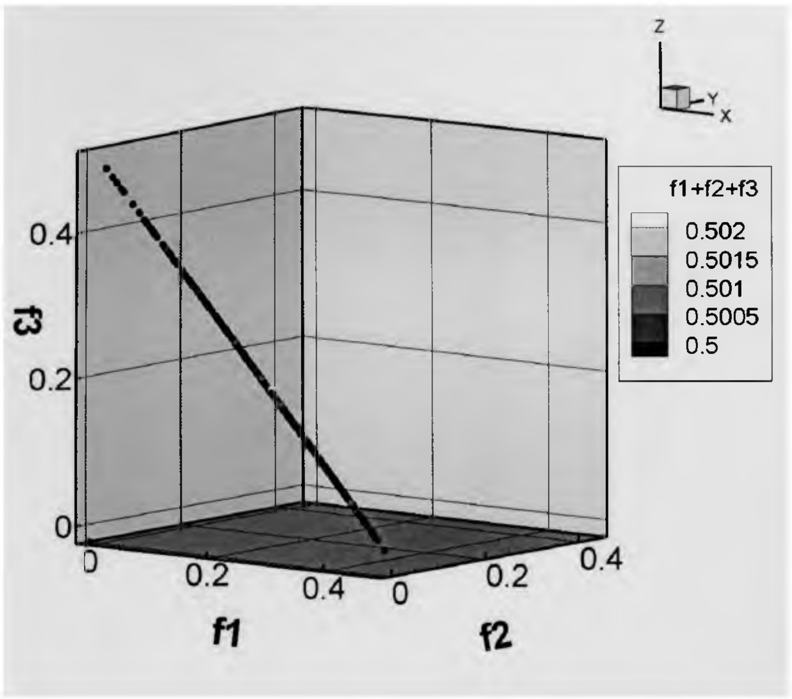

Figure 16b 3-objective test case DTLZ1 with $iterp_{\max} = 0$: view 2

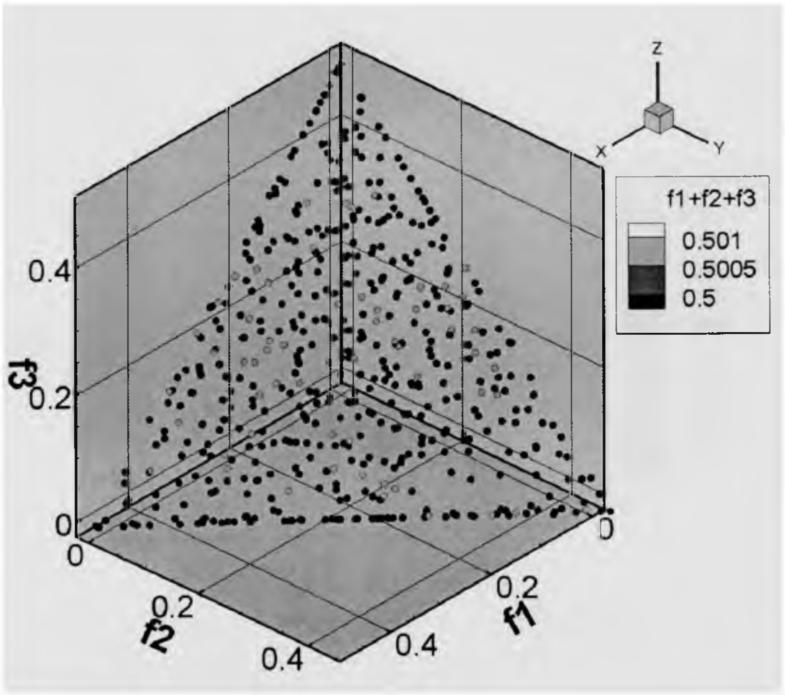

Figure 17a 3-objective test case DTLZ1 with $iterp_{\max} = 3$: view 1

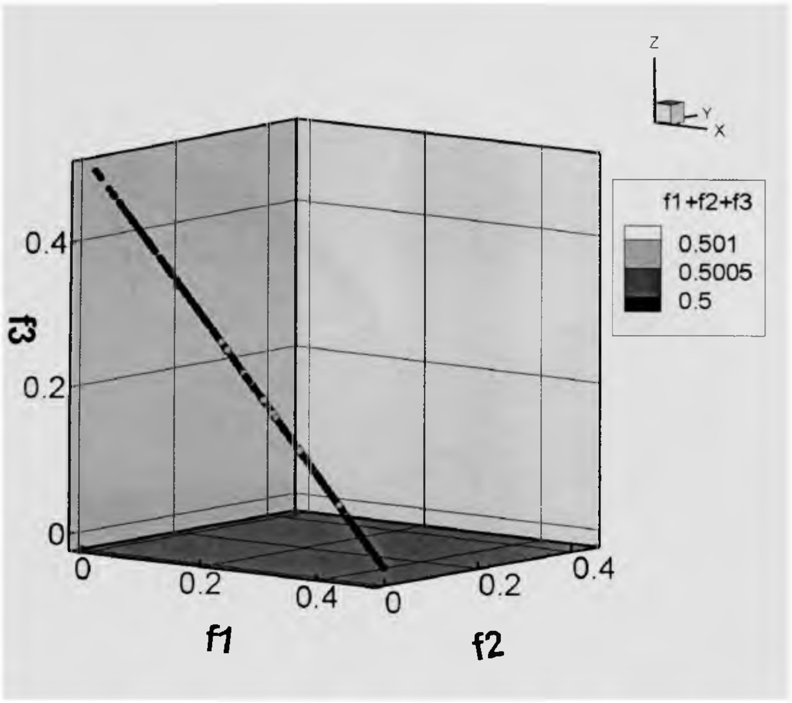

Figure 17b 3-objective test case DTLZ1 with $iterp_{\max} = 3$: view 2

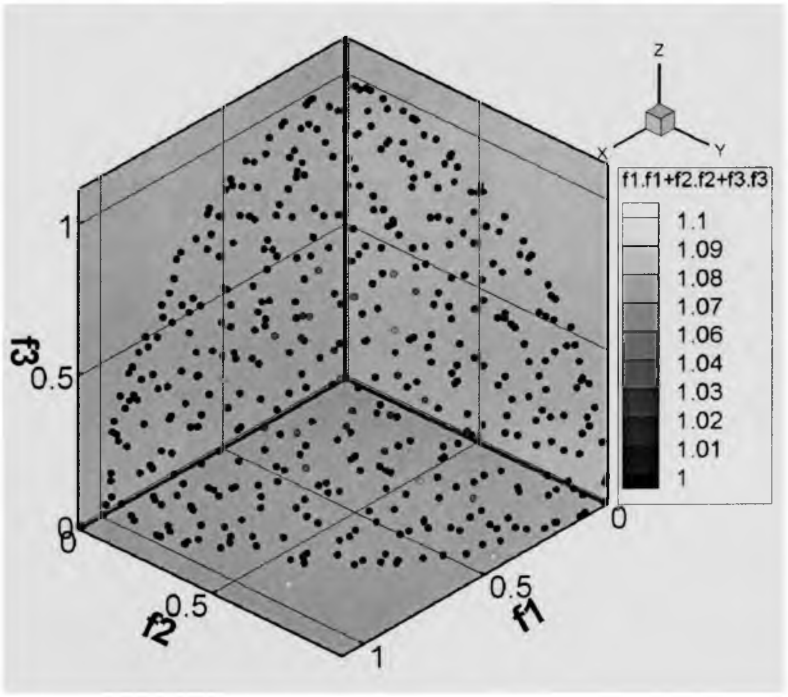

Figure 18a 3-objective test case DTLZ2 with $iterp_{\max} = 0$: view 1

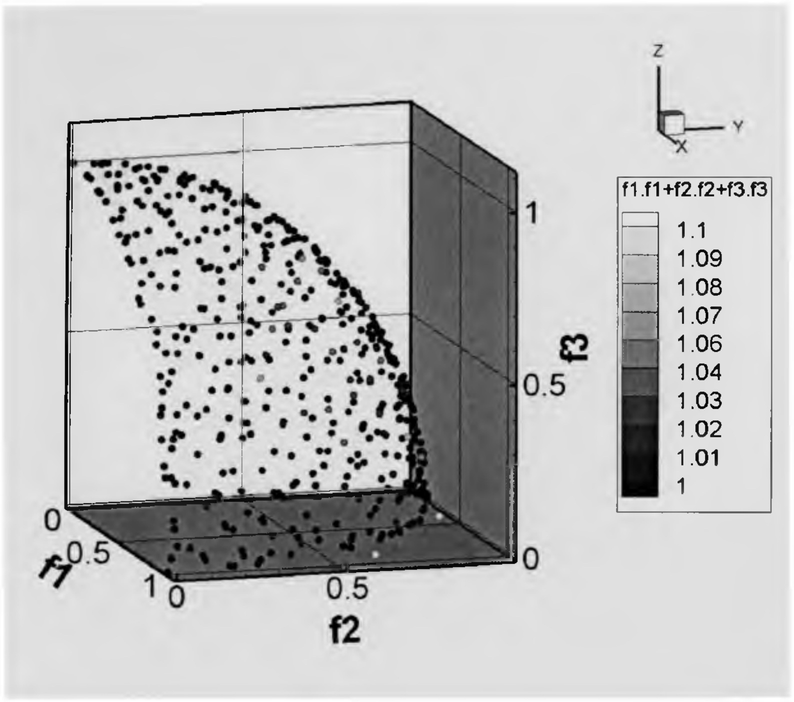

Figure 18b 3-objective test case DTLZ2 with $iterp_{\max} = 0$: view 2

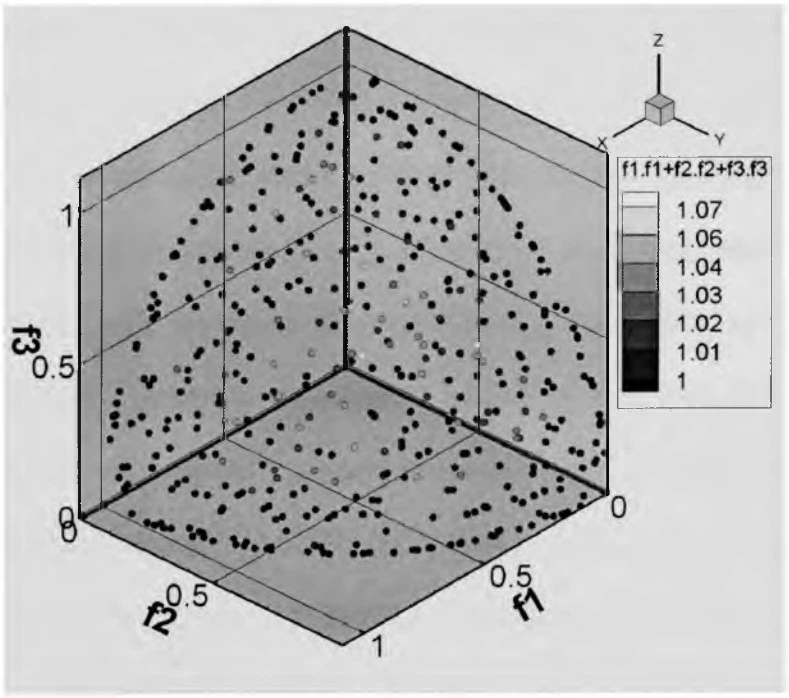

Figure 19a 3-objective test case DTLZ2 with $iterp_{\max} = 3$: view 1

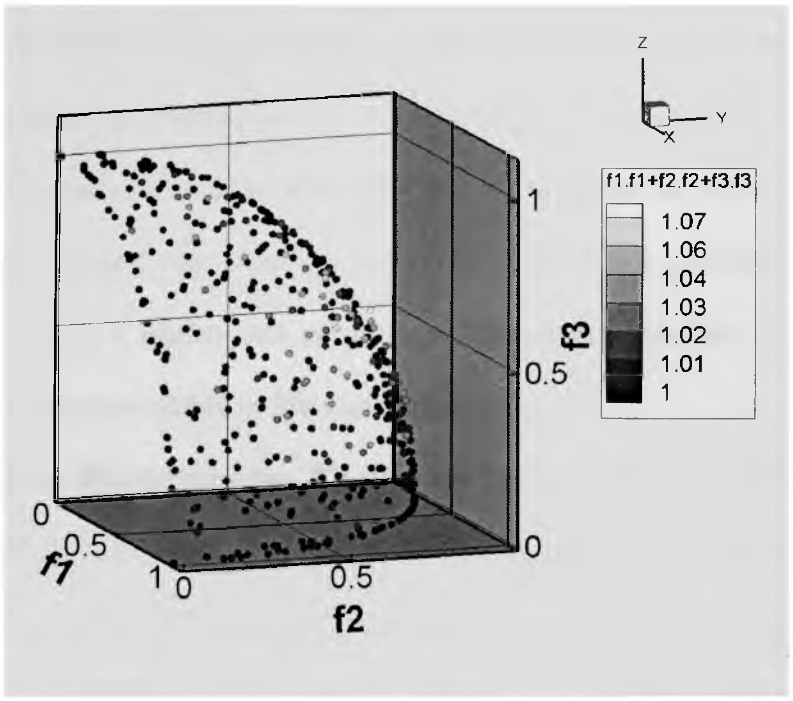

Figure 19b 3-objective test case DTLZ2 with $iterp_{\max} = 3$: view 2

Different views of the final Pareto front computed for the 3-objective problems DTLZ1 and DTLZ2 have been illustrated in figures 16 to 19. It is observed that MPP performs very well in producing Pareto solutions that are both reasonably accurate and well distributed along the Pareto surface. This is further evident when results from figures 16-19 are compared with the performance of NSGA-II and SPEA on these problems (given in Appendix C). Sectional convergence scheme as seen from figures 17 and 19 helps in covering the whole global Pareto front more effectively. Hence the boundaries of the global Pareto computed by MPP are crisply defined when using sectional convergence. However in case of DTLZ2, sectional convergence proves to be computationally more expensive leading to relatively lower accuracy as seen from figure 19.

Though it might seem over-optimistic to extrapolate the performance appreciation of MPP from 3 objectives to N objectives ($N > 3$), Pareto fronts computed by MPP in case of the DTLZ test cases do indicate that MPP has the potential to achieve reasonably accurate well distributed Pareto solutions in case of optimization problems with higher number of objectives; a quality not so common among the standard multi-objective optimization algorithms available in literature and practice.

2.5.3 Constrained Multi-Objective Test Cases: To examine the constraint handling capability of MPP, it was tested was three well known constrained 2-objective test cases studied by Deb *et al.* [18]. Two standard test cases with known analytical solutions namely Binh multi-objective optimization problem no. 2 [20] and the Osyczka multiobjective optimization problem no. 2 [21] have also been tested for. All these test cases are 2-objective minimization problems and are summarized in table 6.

Table 6 Details of the constrained 2-objective optimization test cases

Problem	m	Variable limits	Objective Functions	Constraints
CONSTR	2	$x_1 \in [0.1,1]$ $x_2 \in [0,5]$	$f_1 = x_1$ $f_2 = (1+x_2)/x_1$	$x_2 + 9x_1 \geq 6$ $-x_2 + 9x_1 \geq 1$
SRN	2	$x_i \in [-20,20]$	$f_1 = (x_1-2)^2 + (x_2-1)^2 + 2$ $f_2 = 9x_1 - (x_2-1)^2$	$x_1^2 + x_2^2 \leq 225$ $x_1 - 3x_2 \leq -10$
TNK	2	$x_i \in [0,\pi]$	$f_1 = x_1$ $f_2 = x_2$	$-x_1^2 - x_2^2 + 1 + 0.1\cos\left(16\tan^{-1}\left(\frac{x_1}{x_2}\right)\right) \leq 0$ $(x_1-0.5)^2 + (x_2-0.5)^2 \leq 0.5$
Binh	2	$x_1 \in [0,5]$ $x_i \in [0,3]$	$f_1 = 4x_1^2 + 4x_2^2$ $f_2 = (x_1-5)^2 + (x_2-5)^2$	$(x_1-5)^2 + x_2^2 - 25 \geq 0$ $-(x_1-8)^2 - (x_2+3)^2 + 7.7 \geq 0$
Osyczka	6	$x_1 \in [0,10]$ $x_1 \in [0,10]$ $x_1 \in [1,5]$ $x_1 \in [0,6]$ $x_1 \in [1,5]$ $x_1 \in [0,10]$	$f_1 = -\begin{pmatrix} 25(x_1-2)^2 \\ +(x_2-2)^2 \\ +(x_3-1)^2 \\ +(x_4-4)^2 \\ +(x_5-1)^2 \end{pmatrix}$ $f_2 = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 + x_6^2$	$x_1 + x_2 - 2 \geq 0$ $6 - x_1 - x_2 \geq 0$ $2 + x_1 - x_2 \geq 0$ $2 - x_1 + 3x_2 \geq 0$ $4 - (x_3-3)^2 - x_4 \geq 0$ $(x_5-3)^2 + x_6 - 4 \geq 0$

Each constrained test case given in table 6 was run 30 times and the final Pareto front is formed in the same way as in the ZDT and the DTLZ test cases. However, the final Pareto fronts in case of all the five constrained test cases (table 6) are constructed of only those elite set solutions that do not violate any of the problem constraints, *i.e.* for the final Pareto solutions

$$f_{Nf+1} = \sum_{i=1}^{p} \max(g_i, 0) + \sum_{i=p+1}^{p+q} \max((h_i - \varepsilon), 0) = 0). \tag{15}$$

It is worth mentioning that MPP achieved full strength elite set, constituted of such feasible global Pareto solutions in each of these test cases. The user defined parameters in the algorithm pertinent to the constrained test cases are presented in tables 7 and 8.

Table 7 General MPP parameters for first three constrained 2-objective test cases

Parameter	*Value*
Population size (# preys)	100
# Predators	10
Elite strength	40
Crossover probability	1.0
Mutation probability	0.05
# Primary iterations (sections)	0, 3

Table 8 General MPP parameters for last two constrained 2-objective test cases

Parameter	*Value*
Population size (# preys)	100
# Predators	10
Elite strength	100
Crossover probability	1.0
Mutation probability	0.05
# Primary iterations (sections)	0, 6

A higher number of primary iterations and greater elite set strength were used in case of the Binh and the Osyczka problems as seen from table 8. This is to counteract the relatively greater difficulty in covering the whole Pareto front in these two test problems. The converged Pareto fronts computed by MPP in each of these test cases are shown in figures 20 to 24. The global Pareto front computed by Deb *et al.* [18], using NSGA-II and corresponding analytical solutions for the first three test cases are given in Appendix D.

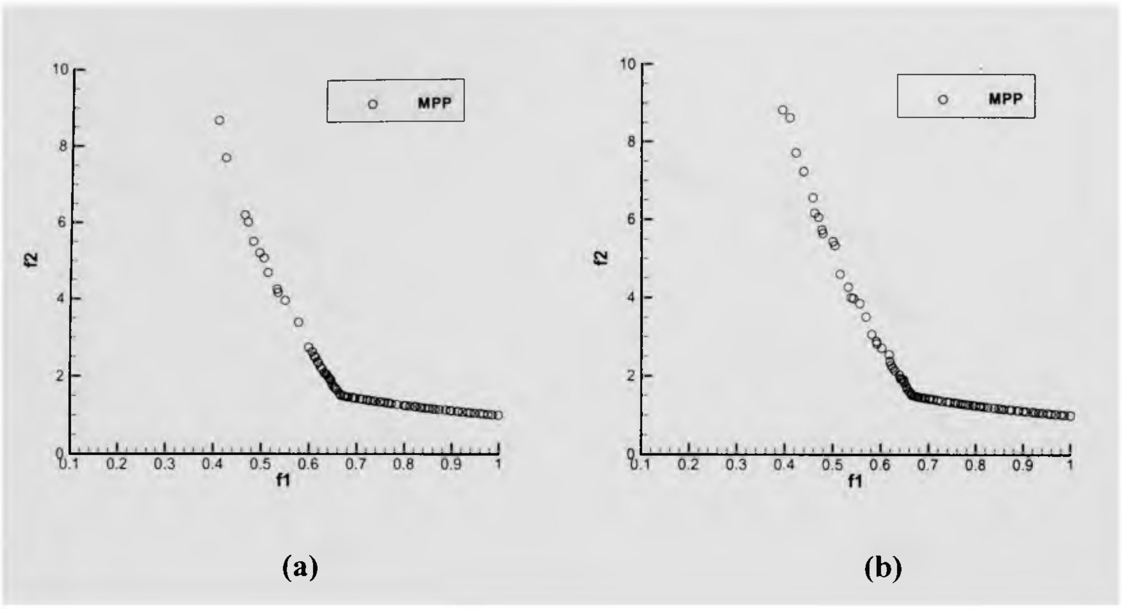

Figure 20 Constrained 2-objective test case CONSTR with (a) $iterp_{\max} = 0$, (b) $iterp_{\max} = 3$

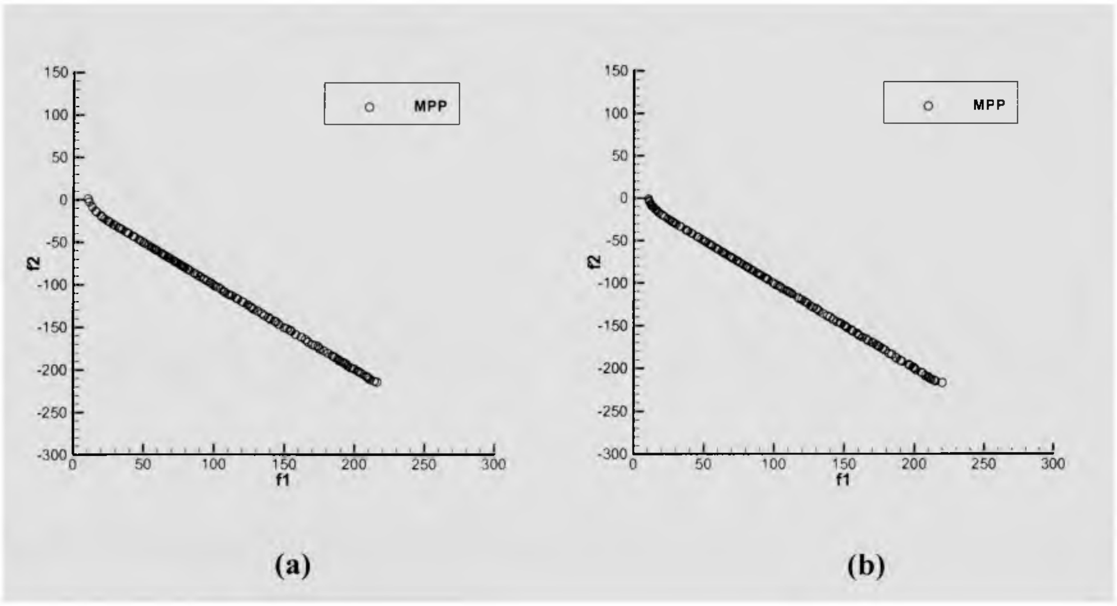

Figure 21 Constrained 2-objective test case SRN with (a) $iterp_{\max} = 0$, (b) $iterp_{\max} = 3$

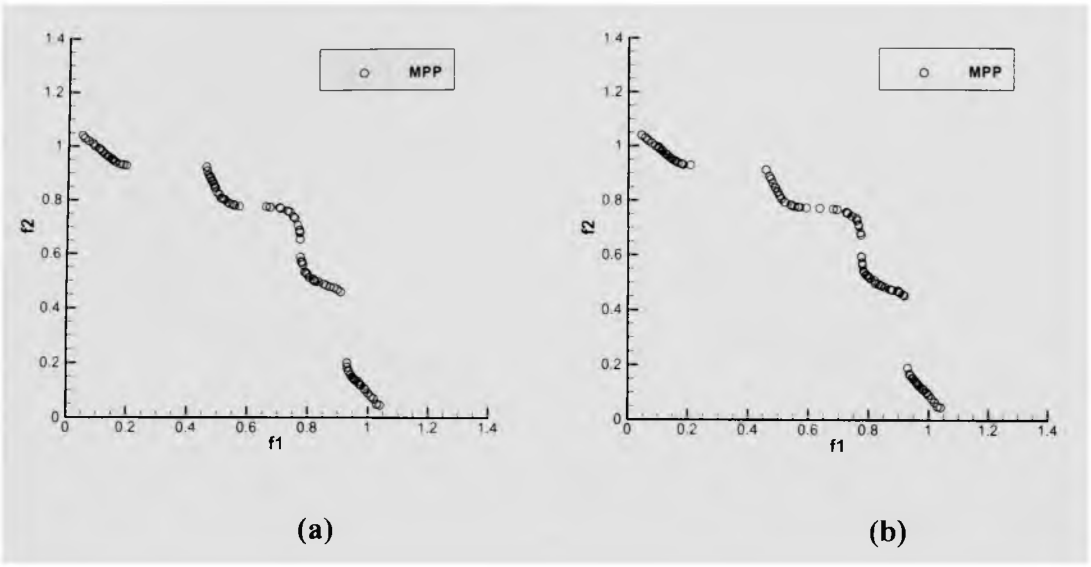

Figure 22 Constrained 2-objective test case TNK with (a) $iterp_{\max} = 0$, (b) $iterp_{\max} = 3$

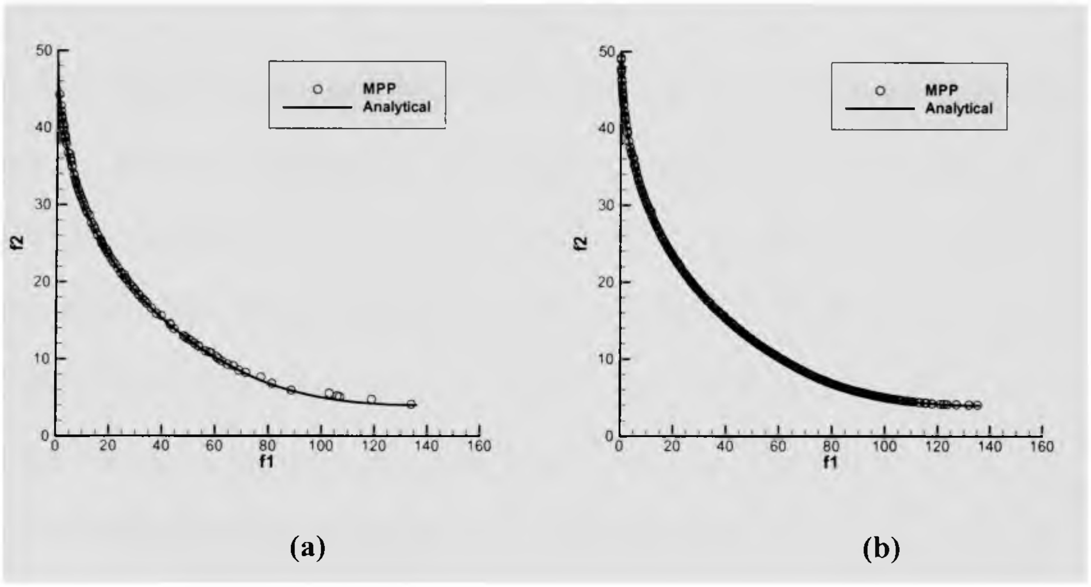

Figure 23 Constrained 2-objective test case of Binh with (a) $iterp_{\max} = 0$, (b) $iterp_{\max} = 6$

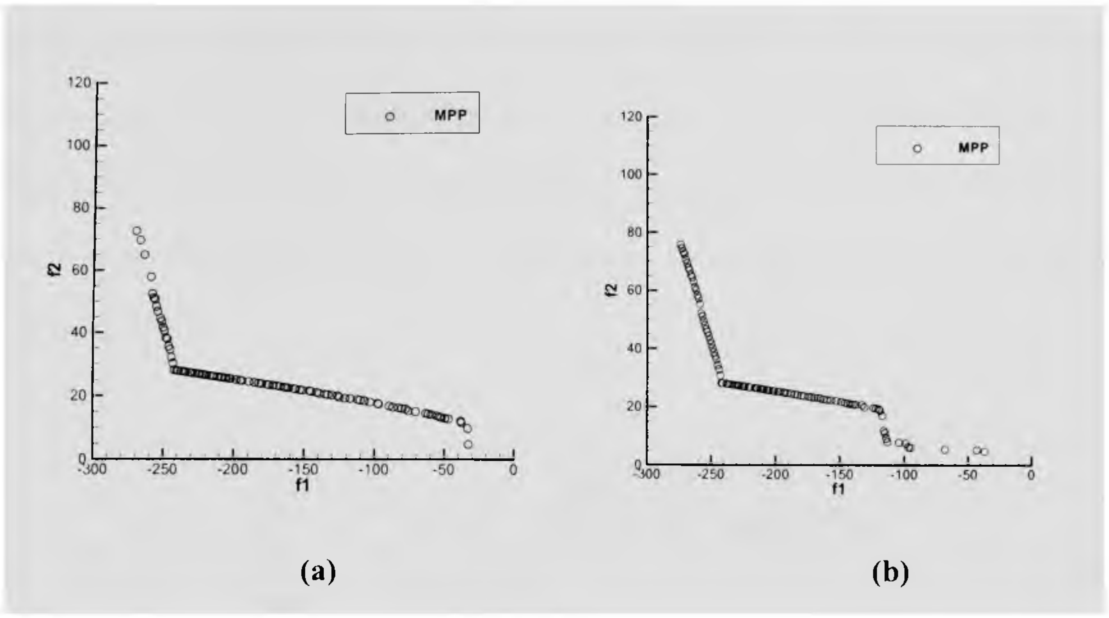

Figure 24 Constrained 2-objective test case of Osyczka with (a) $iterp_{\max} = 0$, (b) $iterp_{\max} = 6$

The final Pareto fronts computed for SRN, TNK and Binh constrained multi-objective problems as shown in figures 21, 22 and 23 respectively are fairly accurate and well distributed. However, in the Binh problem there is significant improvement in performance when using the sectional convergence scheme (figures 23 a and b). In case of CONSTR and Osyczka constrained multi-objective problems (figures 20 and 24), though solutions converge to the global Pareto front, their distribution on the final Pareto is not uniform, even with the sectional convergence scheme. Overall, MPP compares well in performance, with other popular algorithms such as NSGA-II [18] and IOSO [22] (illustrated in Appendix D) in solving similar constrained multi-objective problems at the expense of limited number of function evaluations. Nevertheless, appropriate implementation of the sectional convergence scheme is necessary for certain problems, in order to attain a reasonable spread of solutions along the final Pareto front.

The remarkable feature of MPP is its ability to consistently produce *feasible* Pareto solutions, irrespective of the number or nature (*i.e.* linear or non-linear) of problem constraints involved. This is accomplished without normalization of any objective functions or constraint functions, or application of computationally costly penalty function methods.

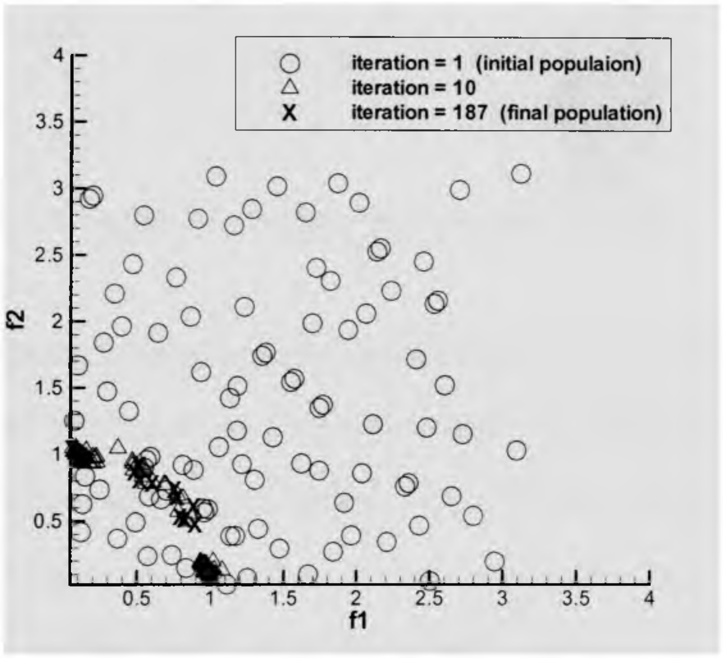

Figure 25 Progress of solutions towards the final Pareto front for TNK problem

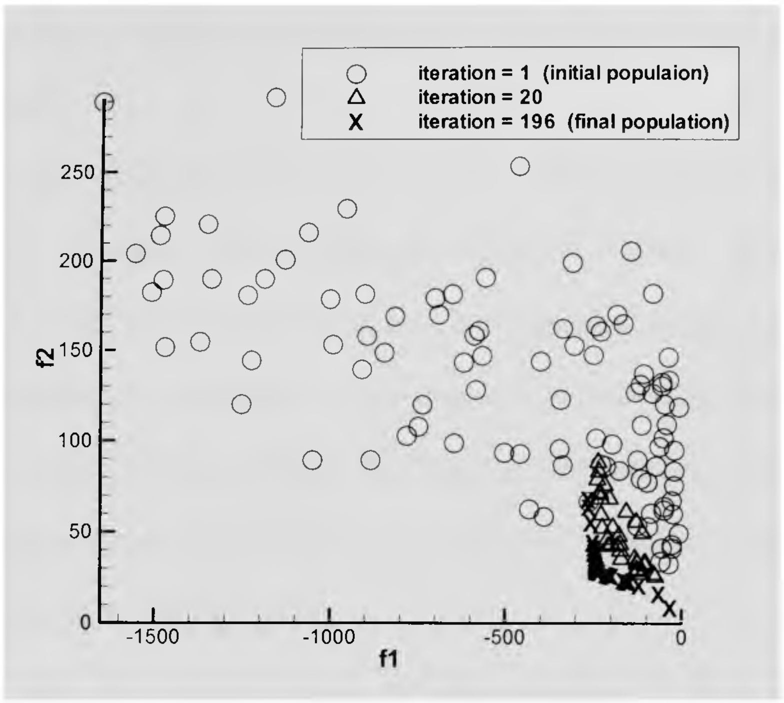

Figure 26 Progress of solutions towards the final Pareto front for Osyczka problem

Figures 25 and 26 demonstrate an immediate migration of solutions into the feasible region and concomitant advancement towards the global Pareto front during the initial stages of the algorithm. Hence, the pace at which MPP drives the population into the feasible domain and subsequently converges to the global Pareto front is appreciable – a quality which may be attributed to the simultaneous application of the added constraint objective (to be minimized) and constraint dominance criterion introduced by Deb *et al.* [18].

CHAPTER III – SINGLE-OBJECTIVE PREDATOR PREY ALGORITHM

3.1 Overview

Constrained single objective optimization can be defined as the maximization or minimization of a single system parameter subject to certain geometric/process constraints, both of which are dependent on a set of independent system variables (design variables). Depending on the nature of the objective function that maps the system variables to the dependent parameter and the nature of the constraints, a single objective optimization problem may be classified into several categories as represented by the schematic diagram shown in Figure 27.

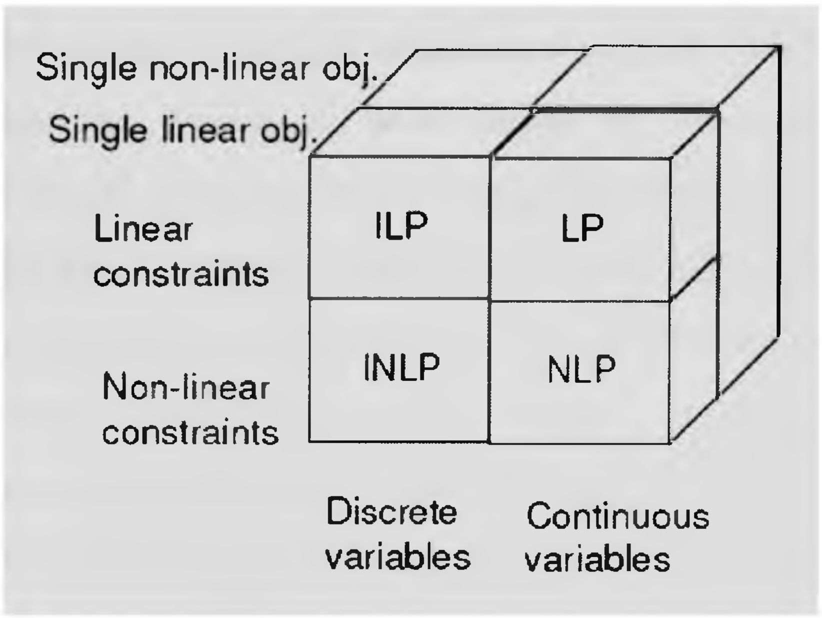

Figure 27 Classification of constrained single objective optimization problems

[Note: LP – Liner Programming, NLP – Non-Linear Programming, ILP – Integer Linear Programming, INLP – Integer Non-Linear Programming]

3.1.1 Classical Methods and Their Drawbacks: Classical single-objective optimization algorithms use gradient-based and heuristic-based search techniques [23, 24]. In contrast to such deterministic search principles, evolutionary algorithms follow stochastic search principles that mimic the process of natural evolution. Classical algorithms are relatively computationally inexpensive and reliable when solving single objective optimization problems with few design variables. However in case of problems with

(i) large number of design variables

(ii) large number of constraints

(iii) severe non linearity of the objective function or constraint functions

(iv) multimodal objective functions (i.e. having multiple local extrema)

(v) discontinuous search space,

classical algorithms prove to be both unreliable and inefficient when compared to stochastic algorithms. Typical real world systems are often simulated using computational models instead of a definite mathematical function mapping decision variables to the problem objective, in which case it becomes increasingly difficult to calculate gradients at different locations of the problem space. Evolutionary algorithms on the other hand, use a set of random multiple solutions that gradually approach the global extrema over generations based on relative fitness and subsequent evolution. They do not require any gradient estimation. Consequently evolutionary algorithms are free from the inherent drawbacks of classical algorithms when dealing with complex single objective optimization problems. Hybrid optimizers of the likes of H1 and H2 developed by Colaco *et al.* [25] and Colaco and Dulikravich [26] employ a combination of the deterministic and stochastic/evolutionary algorithms hence utilizing the advantages of both types of optimization techniques.

3.1.2 Parent algorithm of SOMPP: The Single-Objective Modified Predator-Prey (SOMPP) algorithm has been derived from the parent algorithm Modified Predator-Prey (MPP) developed by Chowdhury *et al.* [14]. Any unconstrained single-objective optimization problem is treated as a two-objective optimization problem, where the second objective is just a clone of the first one. In case of the constrained problems, all the equality and inequality constraints are collaged together to form a third objective and the problem is solved as a three-objective optimization problem. Nevertheless, the concerted constrained objective does not conform to the requirements of a general Pareto convergence. Therefore, this three-objective scenario is distinctly different from a generic three-objective optimization problem and treated accordingly by SOMPP.

3.2 Single Objective Modified Predator-Prey Algorithm (SOMPP)

Any general constrained single objective test problem is reformulated as follows.

$$\begin{aligned}
&\text{Minimize } f_1 = f(X) \\
&\text{Minimize } f_2 = f_1 \\
&\text{subject to} \\
&g_i \le 0, \quad i = 1,2,3,\dots,p \\
&h_i = 0, \quad i = p+1, p+2, \dots, p+q \\
&p, q \in N
\end{aligned} \tag{16}$$

Here, X is the vector of design variables, *i.e.* $X = (x_1, x_2, x_3, \dots, x_m)$ $, x_i \in R$

The constraints are added up to form the third objective

$$\begin{aligned}
\text{Minimize } f_3 = &\sum_{i=1}^{p} \max(g_i, 0) \\
&+ \sum_{i=p+1}^{p+q} \max((h_i - \varepsilon), 0)
\end{aligned} \tag{17}$$

where ε is the tolerance for equality objectives.

3.2.1 SOMPP Version-1: The initialization and subsequent steps executed by the algorithm in each generation in solving a single-objective optimization problem are sequentially presented below. It should be noted that in the case of a maximization problem the function is multiplied by '-1', to convert it into a general minimization problem.

First, a population of N candidate solutions (prey) is created using Sobol's [15] quasi random sequence generator to generate their vectors of design variables. Using these values of design variables, objective functions for each candidate solution are evaluated.

Then, the prey are placed at nodes of a two dimensional grid with connected ends hence having a toroidal nature. The grid is allowed to adjust its size dynamically according to the population size maintaining the dimensions I x J, where typically J = 5. Random members of the prey population are cloned and placed on the grid when $N < I \times J$ in order to ensure that all grid points (all have integer co-ordinates) are occupied by prey.

Similarly, M predators are placed on the same 2D grid such that they occupy random cell centers (Figure 1). The value of M is determined by the following empirical formula.

$$M = \max\left(\left(\left\lceil \frac{N}{20} \right\rceil \times Nf\right), 4\right) \tag{18}$$

where, $\lceil r \rceil$ is the lowest integer greater than r, $r \in R^{+}$, and Nf is the number of objectives. Each predator is associated with a weighted value of the objectives as follows.

$$f = \sum_{i=1}^{Nf} w_i f_i \tag{19}$$

Here, w_i is the weight associated with the ith objective function, and f_i is the ith objective function. The weights are distributed uniformly in case of two-objective problems (from (0,1) to (1,0)) and using Sobol's [15] algorithm in case of problems with more than two objectives (constrained problems). Predators are randomly located at the centers of quadrilateral cells drawn on an unfolded toroidal surface. Each neighborhood that contains a predator can be termed as an 'active locality' as shown in figure 1. In each of these localities/cells, the value of f as defined by equation 5 corresponding to the local predator, is calculated for each prey. The weakest prey, that is, the prey having the maximum value of f is selected to be killed and replaced by a new prey produced by the crossover of the two strongest neighboring prey and a subsequent mutation of the crossover child.

The blend crossover (BLX-α) [5] was used in this case.

$$\begin{aligned} x_i^{(1,t+1)} &= (1-\gamma_i)x_i^{(1,t)} + \gamma_i x_i^{(2,t)} \\ \gamma_i &= (1+2\alpha)u_i - \alpha \end{aligned} \tag{20}$$

Here, $x_i^{(1,t)}$ and $x_i^{(2,t)}$ are the parent solutions, $x_i^{(1,t+1)}$ is the child solution and u_i is the random number between 0 and 1. A value of 0.5 was used for α as suggested by Deb [5].

Non-uniform mutation [5], as defined below, was used in this algorithm.

$$\begin{aligned} \beta &= 10^{-\left(1+K\, t/t_{\max}\right)} \\ y_i^{(1,t+1)} &= x_i^{(1,t+1)} + \tau\left(x_i^{(U)} - x_i^{(L)}\right)\left(1 - r_i^{\left(1-t/t_{\max}\right)^b}\right) \times \beta \end{aligned} \tag{21}$$

Here, 10^{-K} is the terminal order of magnitude of the extent of mutation, $y_i^{(1,t+1)}$ is the child produced by mutation of the ith variable, $x_i^{(U)}$ and $x_i^{(L)}$ are upper and lower limits of the ith variable, r_i is the random number between 0 and 1, t and $t_{\max}$ are the number of function evaluations performed until then and maximum allowed number of function evaluations, respectively, while b is the user defined parameter (b = 1.5 determined empirically) and β is the scaling parameter.

The child prey produced by crossover and mutation qualifies to be accepted only if it fulfills the following three criteria:

(i) The child is stronger than the worst local prey based on f calculated by equation 2,

(ii) The child is non-dominated (Deb 2002) with respect to the other three local prey, and

(iii)The child is not within the objective space hypercube [5] of the remaining three neighboring prey.

The basis for determining relative dominance between two solutions (solutions i and j) is the same as used in NSGA-II [18], which is as follows. Solution *i* is said to dominate solution *j* if:

(i) Both solutions are infeasible, and solution *i* has lower value of constraint violation than solution j (i.e. $f_3^i < f_3^j$)

(ii) Solution *i* is feasible and solution *j* is infeasible.

(iii)Both solutions are feasible (or problem is unconstrained) and solution *i* has a lower objective value than solution *j* (that is, $f_1^i < f_1^j$).

In case of the third criterion, each old local prey is considered to be at the centre of its hypercube, the size of which is dynamically updated with generations and is determined by the following novel equation.

$$\begin{aligned} \omega &= 10^{-\left(2+L\frac{t}{t_{\max}}\right)} \\ \eta_i &= \omega \times \min\left(f_i^{new\ prey}, f_i^{old\ prey}\right) \end{aligned} \tag{22}$$

Here, 10^{-L} is the terminal order of magnitude of relative window size, ω is the window size of the hypercube and η_i is the half side length of the hypercube corresponding to the i^th objective. The first two criteria promote convergence towards the global minimum. The third criterion helps in maintaining diversity in the solution space in order to avoid converging to a local minimum. Ten trials were allowed to produce a qualified child that satisfies these three criteria, failing which the worst prey was retained.

Upon completion of the above predator-prey interactions in each active locality, the predators were relocated randomly. A probability based relocation criterion was introduced here, which ensures that each cell is visited, therefore favoring an even distribution of the number of visitations by a predator to each cell. The predator relocation criterion is defined as follows:

$$\begin{array}{ll} \text{if } \ cellcount(i,j) > cellcount_{avg}+1, & locate = no \\ \text{else} & , \ locate = yes \end{array} \tag{23}$$

Here, $cellcount(i,j)$ is the number of times predators have visited the cell (i,j) in previous generations, $cellcount_{avg}$ is the average of all $cellcount(i,j)$ and (i,j) is the randomly generated location on the 2D lattice. This new feature ensures that every member of the prey population irrespective of its location in the 2D lattice gets fair opportunity of improvement.

At the end of each generation the objective value of the strongest prey (based on dominance criterion) is found and the algorithm checks for termination. The convergence or termination criteria are as follows:

(i) Maximum allowed number of function evaluations (*fcallmax*) has been exhausted, or

(ii) The best objective value searched by the algorithm has not changed during the last 100 generations.

The dynamic reduction of the window size of the hypercube and the mean extent of mutation along the course of generations introduces the desirable attribute of 'adaptive shrinkage of the search radii' as solutions converge towards the global optimum.

The above steps summarize the basic version of SOMPP which can be termed as SOMPP Version-1. During the course this research, further alterations/additional techniques were also implemented causing minor to significant improvements in its performance. The improved versions of SOMPP are described in detail as follows

3.2.2 SOMPP Version-2 (Rank Based Predator Relocation): Localities with relatively stronger prey were designed to have a higher affinity of attracting predators. The probability '$cellprob_{i,j}$' of locating a predator in a particular locality (co-ordinates i, j generated by a random number generator) is determined as follows.

$$cellrank_{i,j} = \min\begin{pmatrix} rank_{i,j} & rank_{i+1,j} \\ rank_{i+1,j+1} & rank_{i,j+1} \end{pmatrix} \qquad (24)$$

$$cellprob_{i,j} = \frac{N - cellrank_{i,j}}{N}$$

Here, $cellrank_{i,j}$ is the rank of the cell/locality (i,j) and $rank_{i,j}$ is the rank of the prey located at the grid point (i,j), ranking being determined on the basis of dominance. N is the total number of prey, hence equal to the maximum rank in the population. This feature speeds up convergence, but limits the domain of search in certain cases.

3.2.3 SOMPP Version-3 (Nine Prey Neighbourhood): Instead of the predator being located at the center of a four-vertex quadrilateral cell, the predator is now located on the same grid nodes as prey and allowed to have access to all 8 preys around it as well as the prey at that very grid location (Figure 28).

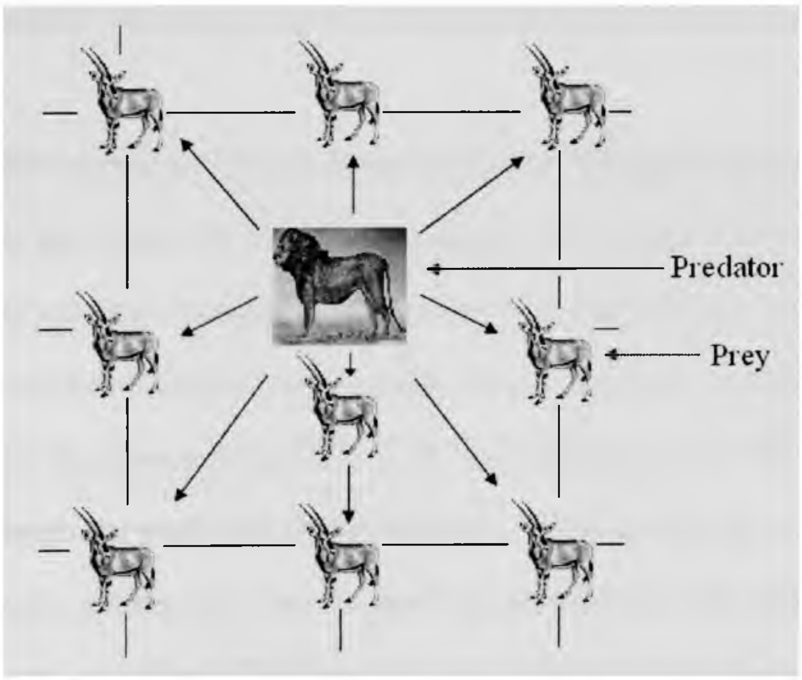

Figure 28 An active 9 prey locality/neighbourhood on the grid drawn on an unfolded toroidal surface.

This increases the neighbourhood scope of the predator from four to nine. Since prey are not relocated in SOMPP, this modification facilitates faster communication of genetics among prey irrespective of their location on the unfolded toroidal surface grid, which in turn accelerates the rate of improvement of the prey population as a whole. However, this modification instills a tendency to converge to a local minimum.

3.2.4 SOMPP Version-4 (Global Elitist Crossover): Here, the worst prey in each active neighbourhood is replaced by the crossover of the strongest two prey in the entire prey population, instead of the strongest two local prey. Strength of the prey in this case is determined on the basis of the objective value. This significantly decreased the number of function evaluations necessary, but promoted convergence to local minima. This might be avoided by selecting the parents for crossover out of the top '*frac*' fraction of the prey population based on dominance, instead of the two global prey with minimum objective values.

3.2.5 SOMPP Version-5 (Version-2 and Version-3 Combined with an Epidemical Operator): In this version of SOMPP, the concepts of nine-prey active neighborhoods and rank based relocation of predators are implemented simultaneously to promote faster convergence and better communication among the prey. However, the rank for each cell is calculated as the average of the ranks of all the local prey in that cell. In addition to that, to counteract the possibility of convergence to a local minimum, a concept of an epidemic genetic operator was introduced as implemented by Cuco *et al.* [27] in the Epidemic Genetic Algorithm. If the objective value of the strongest prey does not change over a certain number of consecutive iterations, a part of the prey population is discarded and replaced with new population generated using Sobol's [15] quasi-random sequence generator. This is implemented as follows.

if $Nchng > 10$,

1. Rank prey population by dominance.

2. Discard weakest $0.0 < f_w < 1.0$ fraction of the prey population.
3. Set variable limits suitable to the order of magnitude of the remaining prey and generate $N \times f_w$ new prey to replace the discarded ones.

Here, $Nchng$ is the consecutive number of generations without any change in the objective value of the strongest prey by a relative tolerance of 10e-03.

3.2.6 SOMPP Version-6 (Version-5 with dominance based selection in active neighbourhoods): Here, the relative strength of the prey in an active locality is determined on the basis of the dominance criterion instead of the weighted f value given by equation 5. In case of unconstrained problems, this has no additional influence because the dominance is merely based on the actual objective value. However, in case of constrained problems, this modification helps significantly in directing solutions into the feasible region first, before the process of minimization takes over. This is because the dominance criterion [5] was designed so that feasibility has a preference over minimization. This in turn substantially reduces the domain of search at the later stages making the algorithm more robust and efficient.

3.3 Numerical Experiments

All six versions of SOMPP are implemented using C++ programming language. The objective functions are evaluated by the corresponding external executable files. The C++ code simulating SOMPP is called 'PPsingle_cnstrnt.cpp'. It compiles and runs successfully on both Windows and Linux workstations using Microsoft Visual C++ .NET for the former and KDevelop 3.1.1 for the latter operating systems.

3.3.1 Unconstrained Single Objective Test Functions: The basic SOMPP (Version-1) and the final SOMPP (Version-6) were both tested on ten well known unconstrained single objective test problems [28]. Details about these functions are given in table 9.

Table 9 Details of ten unconstrained single-objective test cases

Test Function	m	Objective Function 1	Analytical Solution
Griewank	2	$f(X)=\sum_{i=1}^{m}\frac{x_i^2}{4000}-\prod_{i=1}^{m}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$ $x_i \in [-600,600]$	$f(X)=0,\ x_i=0$
Rosenbrock	2	$f(X)=100\left(x_2-x_1^2\right)^2+\left(1-x_1\right)^2$ $x_i \in [-2.048,2.048]$	$f(X)=0,\ x_i=1$
Miele-Cantrell	4	$f(X)=\left(e^{(x_2-x_1)}\right)^4+100\left(x_2-x_1\right)^6$ $+\left(\tan^{-1}\left(x_3-x_4\right)\right)^4+x_1^2$ $x_i \in [-10,10]$	$f(X)=0,\ x_1=0,$ $x_2=x_3=x_4=1$
De Jong1	2	$f(X)=\sum_{i=1}^{m}x_i^2$ $x_i \in [-5.12,5.12]$	$f(X)=0,\ x_i=0$
Rastrigin	2	$f(X)=10m-\sum_{i=1}^{m}\left(x_i^2-10\cos\left(2\pi x_i\right)\right)$ $x_i \in [-5.12,5.12]$	$f(X)=0,\ x_i=0$

Schwefel	2	$f(X)=\sum_{i=1}^{m}\left(-x_i\sin\left(\sqrt{\lvert x_i\rvert}\right)\right)$ $x_i\in[-500,500]$	$f(X)=418.9829,$ $x_i=420.9687$
Ackley's path	2	$f(X)=\sum_{i=1}^{m}\left(\begin{matrix}-ae^{-b\sqrt{\frac{\sum_{i=1}^{m}x_i}{m}}}-e^{\frac{\sum_{i=1}^{m}\cos(cx_i)}{m}}\\+a+e\end{matrix}\right)$ $a=20,\ b=0.2,\ c=2\pi$ $x_i\in[-1,1]$	$f(X)=0,\ x_i=0$
Michalewicz	10	$f(X)=-\sum_{i=1}^{m}\left(\sin(x_i)\left(\sin\left(\frac{ix_i^2}{\pi}\right)\right)^{2p}\right)$ $x_i\in[0,\pi]$	$f(X)=-9.66$
Easom	2	$f(X)=-\cos(x_1)\cos(x_2)e^{-\left((x_1-\pi)^2+(x_2-\pi)^2\right)}$ $x_i\in[-100,100]$	$f(X)=-1,\ x_i=\pi$
Goldstein-Price	2	$f(X)=\left(\begin{matrix}1+(x_1+x\ +12)^2\\ \left(\begin{matrix}19-14x_1+3x_1^2-14x_2\\+6x_1x_2+3x_2^2\end{matrix}\right)\end{matrix}\right)\cdot$ $\left(\begin{matrix}\left(30+(2x_1-3x_2)^2\right)^2\\ \left(\begin{matrix}18-32x_1+12x_1^2+48x_2\\-36x_1x_2+27x_2^2\end{matrix}\right)\end{matrix}\right)$ $x_i\in[-2,2]$	$f(X)=3,$ $x_1=0,\ x_2=-1$

[Note: $m=$ number of variables]

The user-defined parameters used in the SOMPP Version-1 and Version-6 algorithms in case of the above test problems are summarized in table 10 respectively.

Table 10 SOMPP Version-6 user-defined parameters for three single-objective test cases

Parameter	Value
Population size (# prey)	10 x m
Crossover probability	1.0
Mutation probability	0.25
Maximum allowed function evaluations	10000
K (mutation)	6
L (hypercube)	10
f_w (epidemical operator for Version-6)	0.9

The test functions were run until one of the following termination criterion was satisfied – (i) 'relative error in the computed minima' ≤ 10e-10 or (ii) the change in magnitude of the instant computed minima for 100 consecutive generations ≤ a relative tolerance of 10e-03 or (iii) the maximum allowed number of function evaluations was exhausted. The relative error is calculated as follows.

$$relative\ error = \begin{cases} \dfrac{\left|Min_{comp} - Min_{anal}\right|}{Min_{anal}}, & if\ Min_{anal} \neq 0 \\ \left|Min_{comp} - Min_{anal}\right|, & if\ Min_{anal} = 0 \end{cases} \tag{25}$$

The history of convergence for SOMPP Version-1 and Version-6 working on the above test problems are shown in figures 29 to 34.

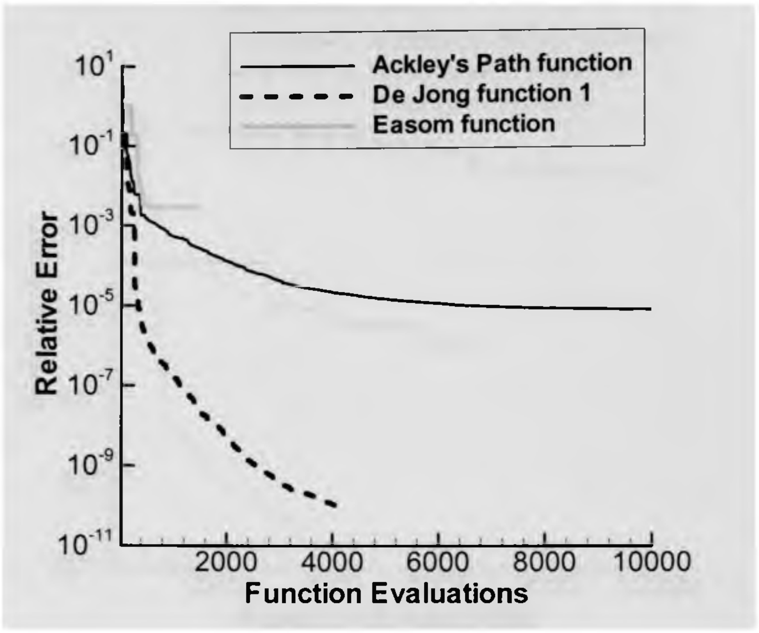

Figure 29 Convergence histories of the Ackley's Path function, De Jong's function 1 and Easom function using SOMPP Version-1

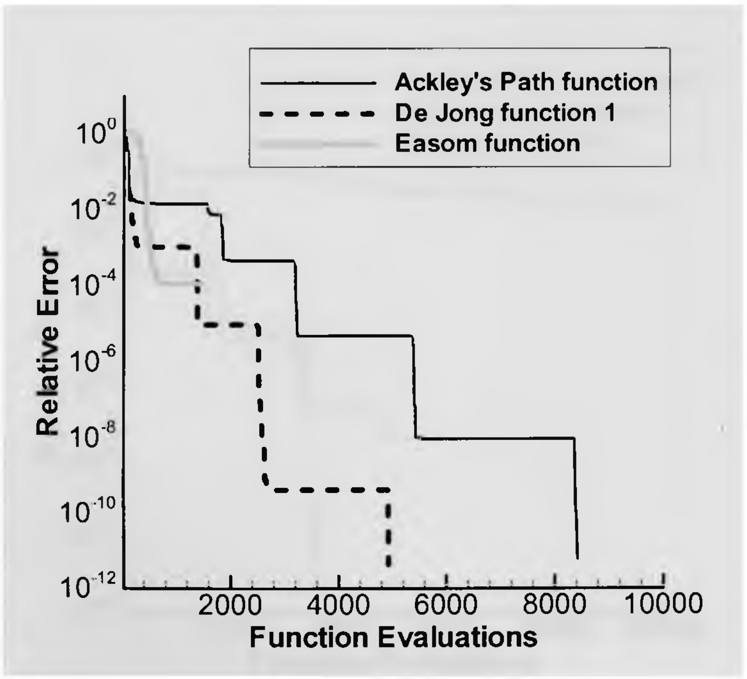

Figure 30 Convergence histories of the Ackley's path function, De Jong's function 1 and Easom function using SOMPP Version-6

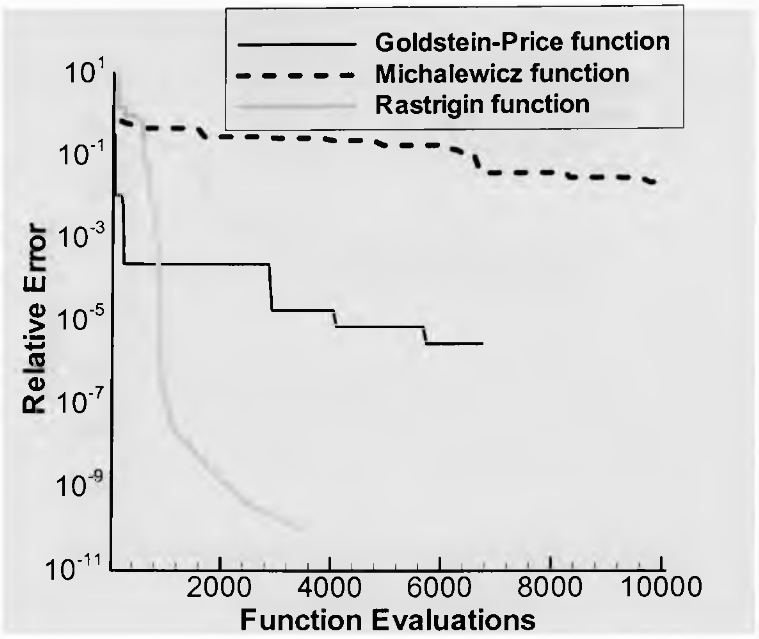

Figure 31 Convergence histories of the Goldstein-Price's function, Michalewicz's function and Rastrigin's function using SOMPP Version-1

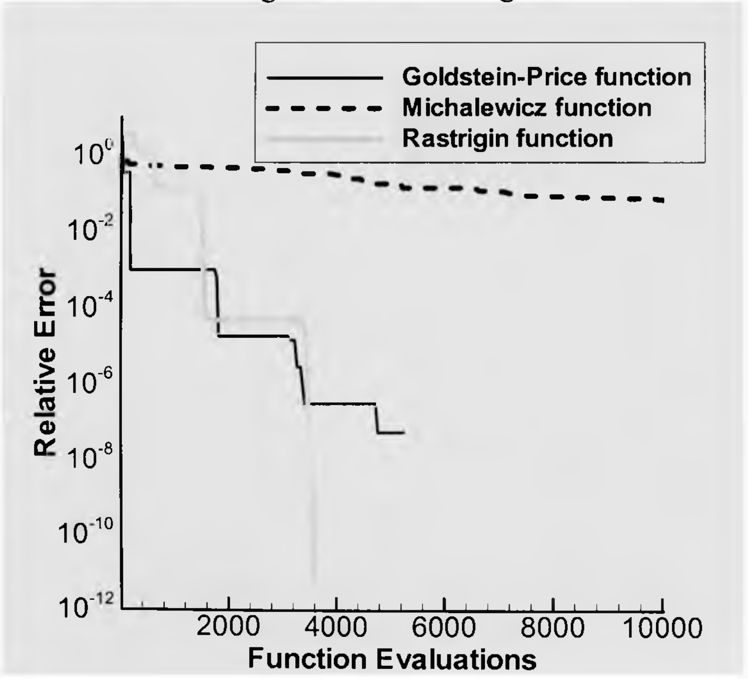

Figure 32 Convergence histories of the Goldstein-Price's function, Michalewicz's function and Rastrigin's function using SOMPP Version-6

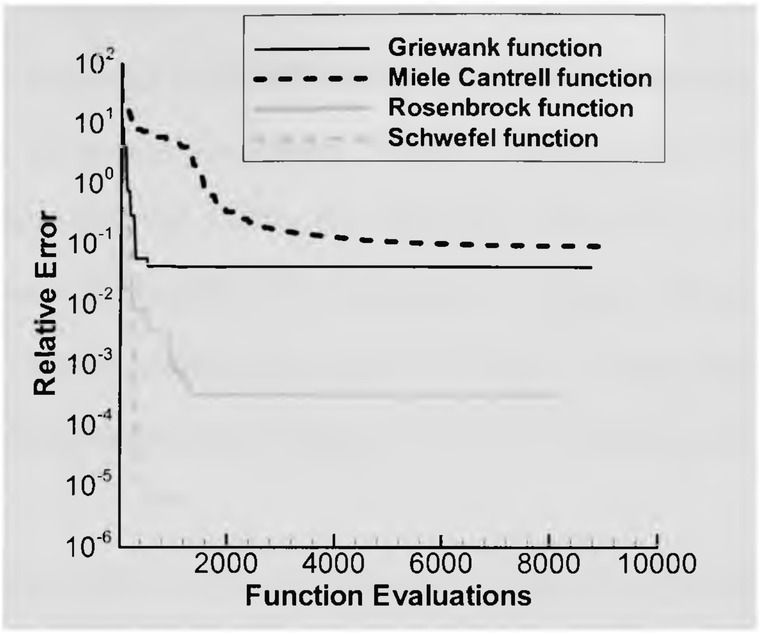

Figure 33 Convergence histories of the Griewank's function, Miele-Cantrell's function, Rosenbrock's function and Schweffel's function using SOMPP Version-1

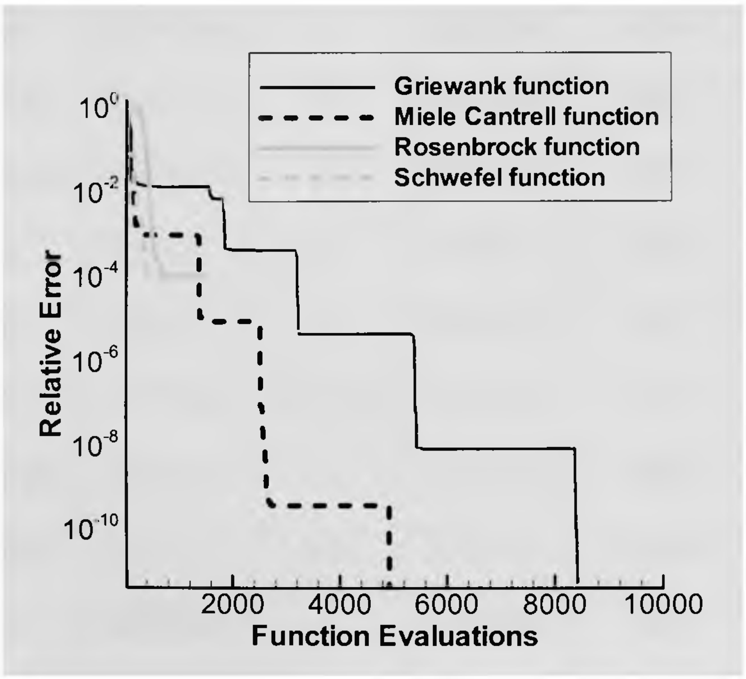

Figure 34 Convergence histories of the Griewank's function, Miele Cantrell's function, Rosenbrock's function and Schewel's function using SOMPP Version-6

Figures 29 to 34 demonstrate that the numerous modifications introduced in SOMPP Version-6 made it superior to SOMPP Version-1. The rate of convergence has increased, and so has the accuracy of the problem. Version-1 performs better than Version-6 for only Michalewicz's function. Further fine calibration of the extent of mutation and the relative hypercube size together with allowing more function evaluations is likely to achieve better accuracy in finding the global minimum with both versions of SOMPP. The various output parameters resulting from these runs are summarized in table 10 and 11.

Table 11 Output for the test problems discussed in table 9 using SOMPP Version-1

TP	Computed Minima	Actual Minima	Relative Error	# Function Evaluations	Computing Time (s)
Griewank	0.0395046	0	0.0395046	8798	452
Rosenbrock	0.0003079	0	0.0003079	8140	417
Miele-Cantrell	0.0873523	0	0.0873523	9082	485
De Jong1	9.85E-11	0	9.85E-11	4078	217
Rastrigin	9.83E-11	0	9.83E-11	3481	186
Schwefel	-837.961	-837.966	5.7E-06	1312	69
Ackley's path	7.86E-06	0	7.86E-06	10000	608
Michalewicz	-9.45616	-9.66	0.021101	10070	586
Easom	-0.997203	-1	0.002796	1436	77
Goldstein-Price	3.00001	3	2.81E-06	6774	370

Table 12 Output for the test problems discussed in table 1 using SOMPP Version-6

TP	Computed Minima	Actual Minima	Relative Error	# Function Evaluations	Computing Time (s)
Griewank	5.28E-12	0	5.28E-12	5590	281
Rosenbrock	0.0003965	0	0.0003965	5485	288
Miele-Cantrell	3.82E-06	0	3.82E-06	6064	306
De Jong1	4.26E-12	0	4.26E-12	4926	278
Rastrigin	6.59E-12	0	6.59E-12	3576	200
Schwefel	-837.961	-837.966	5.23E-06	1422	72
Ackley's path	5.92E-12	0	5.92E-12	8422	466
Michalewicz	-9.05829	-9.66	0.0622889	10020	506
Easom	-999892	-1	0.0001079	1465	76
Goldstein-Price	3	3	5E-08	5247	263

It should be noted that in predator-prey algorithms the number of function evaluations made during each generation is not restricted to the population size. Hence, at times the total number of function evaluations made slightly exceeds the maximum allowed number of function evaluations as seen from table 6. It is seen from the table 4 that SOMPP performs well on all the unconstrained single objective test problems from table 1, with the exception of Michalewicz's function.

3.3.2 Constrained/Unconstrained Single Objective Test Problems by Hock & Schittkowskii: In order to thoroughly examine the potentials of SOMPP, the algorithm in its original version (SOMPP Version-1) was tested on the 293 constrained and unconstrained single objective test cases with known analytic solutions. These 293 test cases were derived from the collection of 395 linear/nonlinear test cases (actually 295 test problems) formulated by Hock and Schittkowskii [3] and Schittkowskii [4]. The number of variables involved in these 293 cases ranges from 2 to 100 as shown in figure 5. The number of inequality and equality constraints range from 0 to 38 and 0 to 6, respectively.

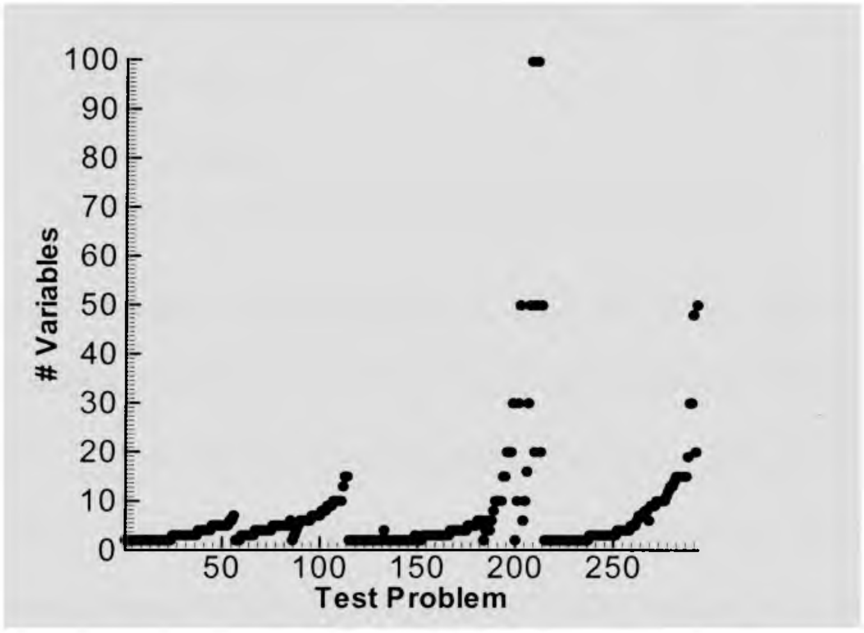

Figure 35 Number of variables for each of the 293 test cases.

The user-defined parameters used in the SOMPP Version-1 algorithm in case of the above 293 test problems are summarized in table 13.

Table 13 SOMPP Version-1 user-defined parameters for the 293 test cases

Parameter	Value
Population size (# prey)	10 x *m*
Crossover probability	1.0
Mutation probability	0.1
Maximum allowed function evaluations	20000
K (mutation)	2
L (hypercube)	4

The prey population size used here is termed as 'small set' which is equal to ten times the number of design variables. A tolerance of 10e-03 was used for equality constraints, that is, $\varepsilon = 10^{-3}$. To compensate for performance fluctuations induced by random generators used in creating the initial population and other genetic operators, the algorithm was run 5 times for each of the 293 test problems resulting in a total of 1465 test runs. An explicit termination criterion was also implemented when relative error became less than 0.001. The final relative error for the computed minimum and the number of function evaluations exhausted in doing so for each of these test runs can be seen in figures 6 and 7.

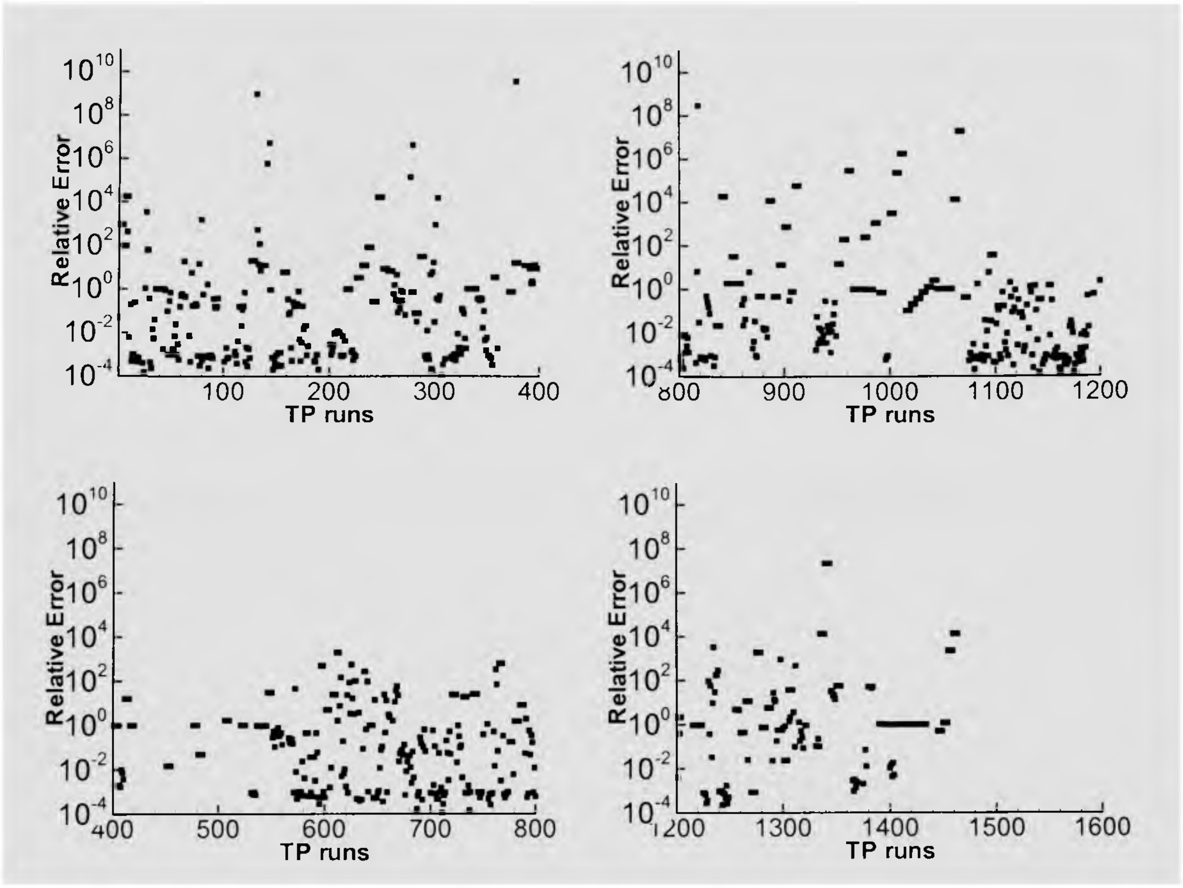

Figure 36 Relative errors of computed minima for the 293 test problems (SOMPP Version-1).

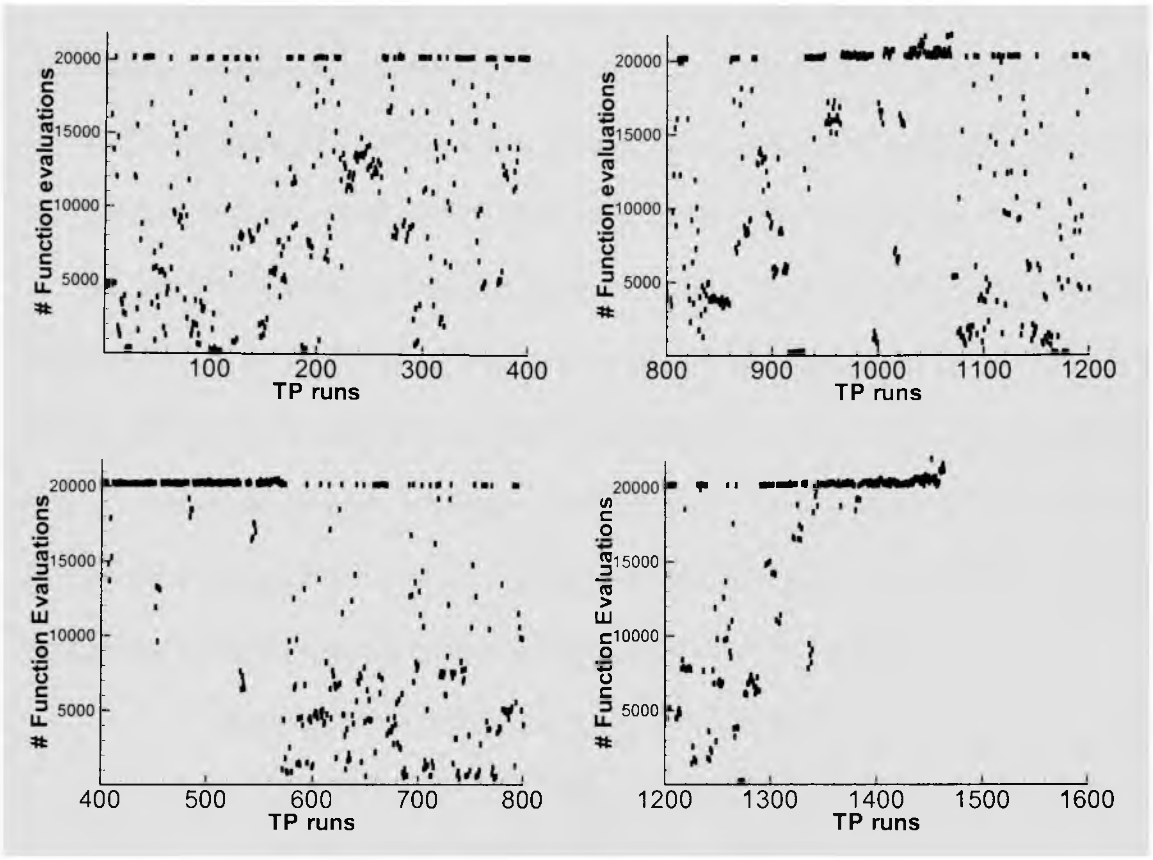

Figure 37 Number of function evaluations made for each of the 293 test problems (SOMPP Version-1).

It is evident from figure 36 that some of the test cases exhibit partial convergence with a relative error of the order of around 1.0. This can be attributed to the presence of either multiple equality or inequality constraints (linear /nonlinear) or both in most of these test problems ([3], [4]). Some of the test cases do not converge at all leading to a relative error of orders above unity. This is primarily due to the lack of any specified variable ranges for some of the design variables in the original publications. In such cases, a comprehensive range of -10e10 to +10e10 was assigned for each design variable. The number of function evaluations varied significantly from problem to problem as seen from figure 37. Test problems (TP) from TP-80 onwards till TP-118 (test runs 400-590) have relatively high number of constraints leading to a higher number of function evaluations. Whereas test problems ranging from TP-190 to TP-210 as well as from TP-260 to TP-293 have a relatively high number of design variables leading also to a higher consumption in terms of the number of the objective function evaluations.

Running all 293 test problems in series is extremely computationally time consuming. Consequently, a set of 13 test problems were chosen from among these 293 cases. These 13 test cases involve number of variables ranging from 2 to 50 (with or without specified limits), number of equality constraints ranging from 0 to 6 and number of inequality constraints ranging from 0 to 38, thereby exhibiting varying degree and nature of complexity. Details pertinent to these test problems are given in table 14.

Table 14 Details of the 13 test problems from the set of 293

#	TP	m	q	p		#	TP	m	q	p
1	1	2	0	0		8	118	15	0	29
2	37	3	0	2		9	246	3	0	0
3	44	4	0	6		10	251	3	0	1
4	55	6	6	0		11	301	50	0	0
5	75	4	3	2		12	393	48	2	1
6	110	10	0	0		13	395	50	1	0
7	112	10	3	0						

Here, p = number of inequality constraints, q = number of equality constraints.

All the latter 5 versions of SOMPP (version 2 to 6) were tested on these 13 test problems. Each of these test problems was run 5 times on a small population size (10 x *m*) as before. The user-defined parameters used in the SOMPP algorithm in case of these 13 test problems are summarized in table 15.

Table 15 SOMPP User-defined parameters for the 13 test cases

Parameter	Value
Population size (# prey)	10 x *m*
Crossover probability	1.0
Mutation probability	0.25
Maximum allowed function evaluations	20000
K (mutation)	3
L (hypercube)	6

It should be noted that, compared to table 3, a higher mutation probability was used to prevent intermediate convergence to local minima and subsequent stagnancy in the region of the local minima. Higher values of K and L were used to allow for higher accuracy.

The relative error of the computed minima, the constraint violation of the computed minima, and the number of function evaluations exhausted for each of the 5 versions of SOMPP running on each of the 13 test problems thus resulting in 65 runs can be seen in figures 38, 39 and 40 respectively.

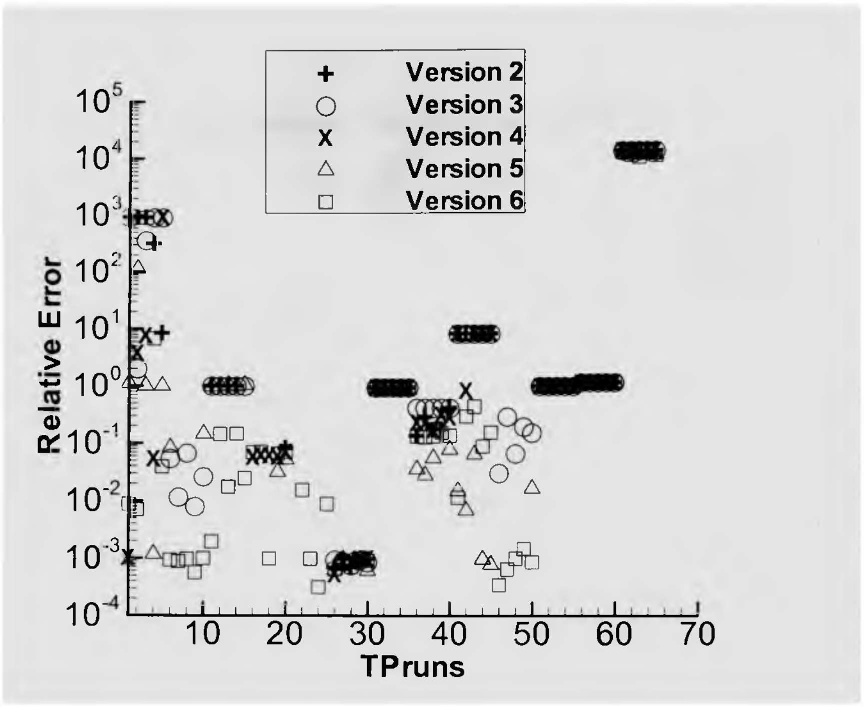

Figure 38 Relative errors of computed minima for the 13 test problems.

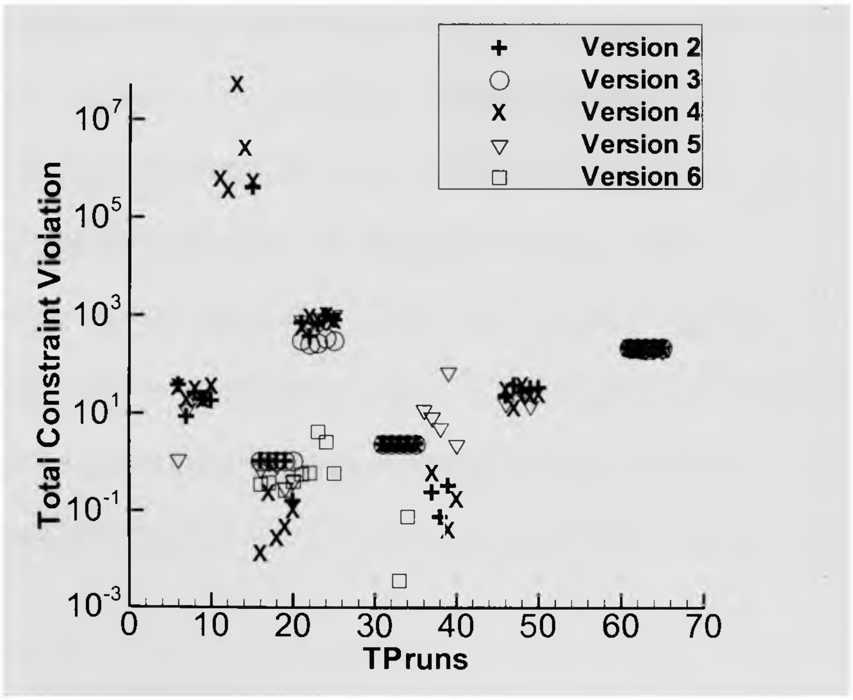

Figure 39 Total constraint violation for each of the 13 test problems that are constrained.

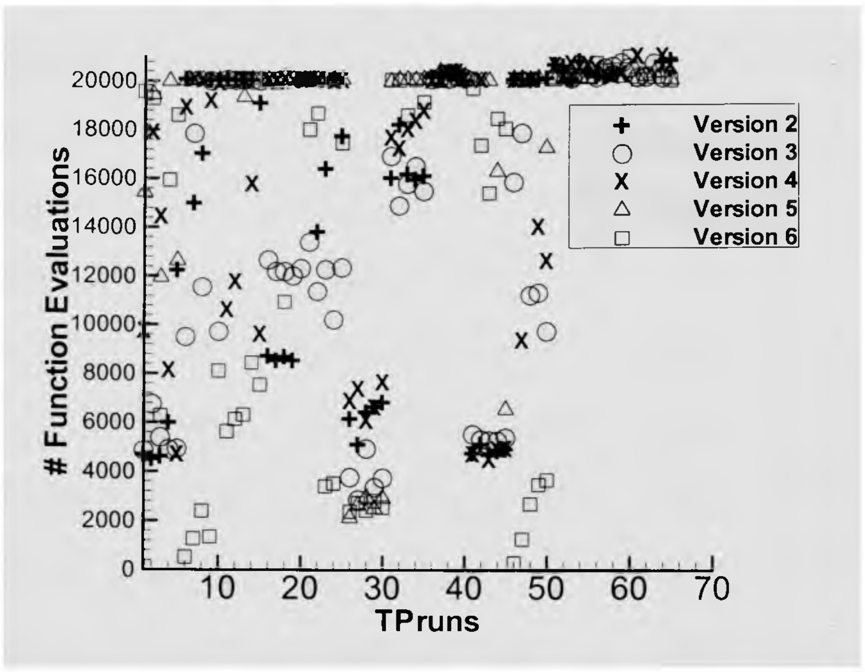

Figure 40 Number of function evaluations for the 13 test problems.

It can be observed from figure 38 that SOMPP Version-6 performs better than the other versions of SOMPP in approaching the global minima. It also has the maximum potential in driving solutions into the feasible domain as seen from figure 39. In case of some of the constrained problems the data points are not visible in figure 39. This is because the constraint violation is zero, which means the final computed minima in these cases are feasible solutions, and hence cannot be represented in a logarithmic plot of figure 39. The pertinent output parameters relating to the most accurate solution (of the 5 runs for each problem) for SOMPP Version-6 running on the 13 cases are summarized in table 16.

Table 16 Output for the 13 test problems with SOMPP Version-6

TP	Computed Minima	Actual Minima	Relative Error	Constraint Violation	# Function Evaluations	Computing Time (s)
1	0.00701	0	0.00701		19291	989
37	-3454.06	-3456	0.00056	0	1347	69
44	-14.9708	-15	0.00195	0	5635	290
55	6.33959	6.3333	0.00098	0.996963	10952	568
75	5176.05	5174.41	0.00031	2.45536	3522	182
110	-45.7493	-45.7785	0.00064		2385	123
112	-0.05151	-0.47761	0.89215	0	20059	1045
118	751.617	664.82	0.130556	0	20031	1045
246	0.011518	0	0.011518		19696	1021
251	-3454.81	-3456	0.000345	0	294	15
301	0	-50	1		20052	1062
393	1.8623	0.86338	1.15699	0	20712	1192
395	19990.6	1.91667	10428.9	163.789	20150	1071

The significantly low accuracy and inability to find feasible solutions in case of TP-395 can be attributed to the fact that there were no specified variable limits for any of the 50 design variables involved in this problem provided in the original publications [4].

SOMPP Version-6 being the most efficient and robust of all the different forms of the SOMPP, was then tested on the entire set of 293 single objective test problems ([3], [4]) run 5 times each. The various user-defined parameters used were the same as given in table 5. The relative error of the computed minima, the number of function evaluations exhausted and the constraint violation of the computed minima for all the (293 x 5) test runs are displayed in Figures 41 to 43.

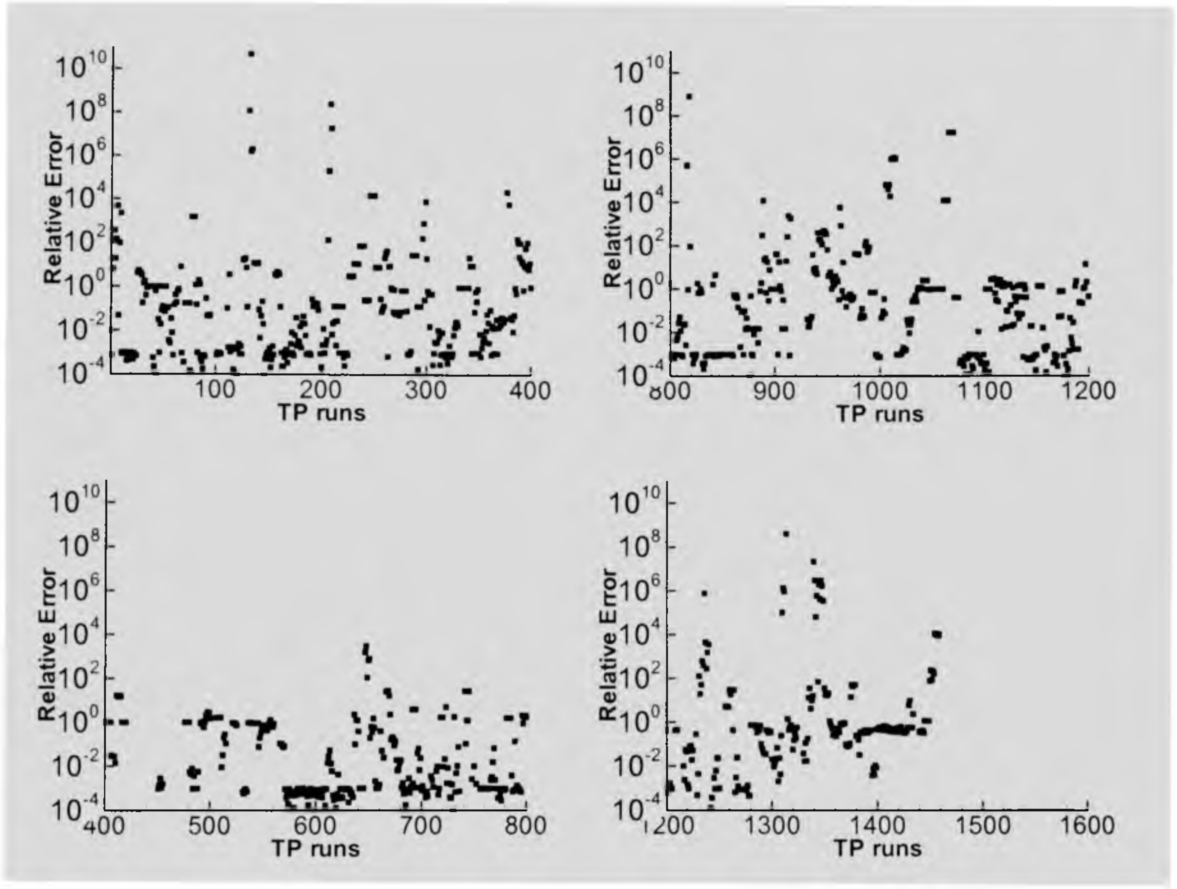

Figure 41 Relative errors of computed minima for the 293 test problems (SOMPP Version-6)

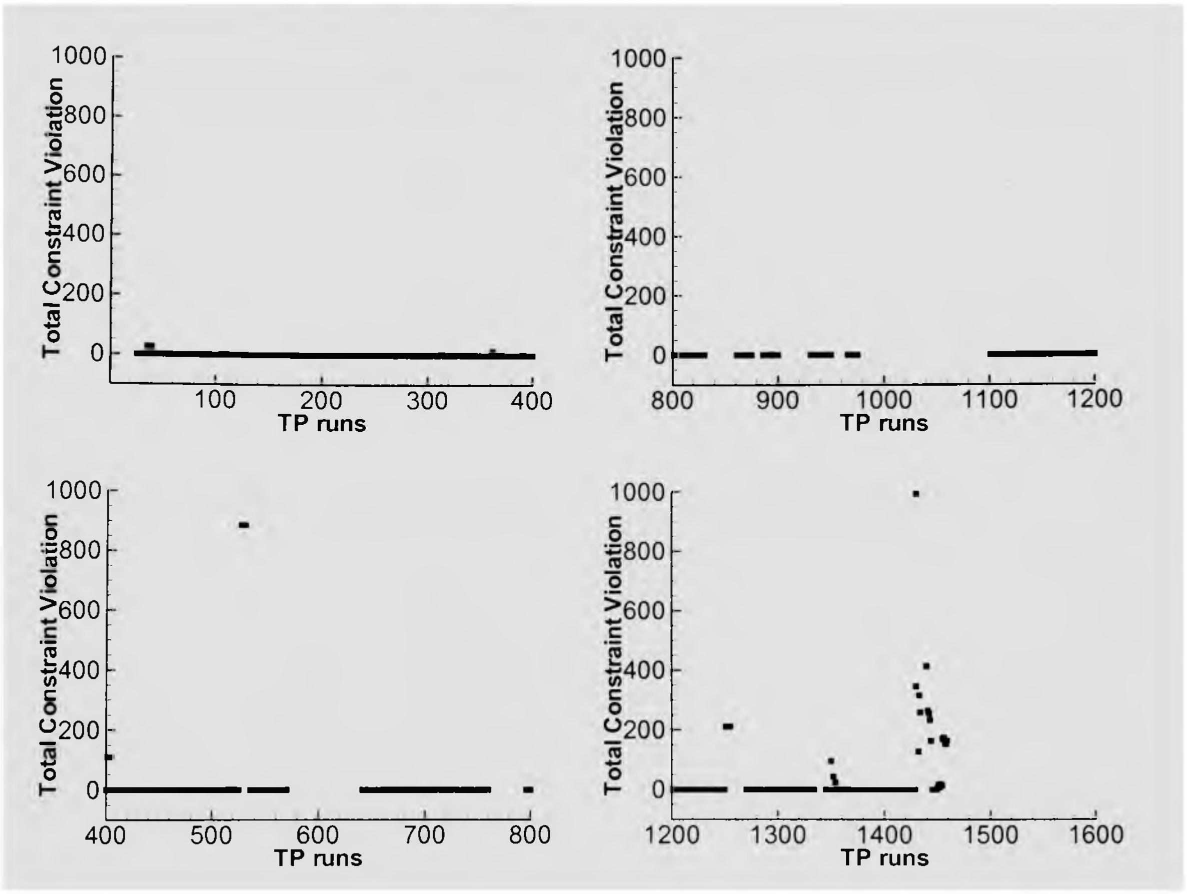

Figure 42 Total constraint violation for each of the 293 test problems that are constrained (SOMPP Version-6).

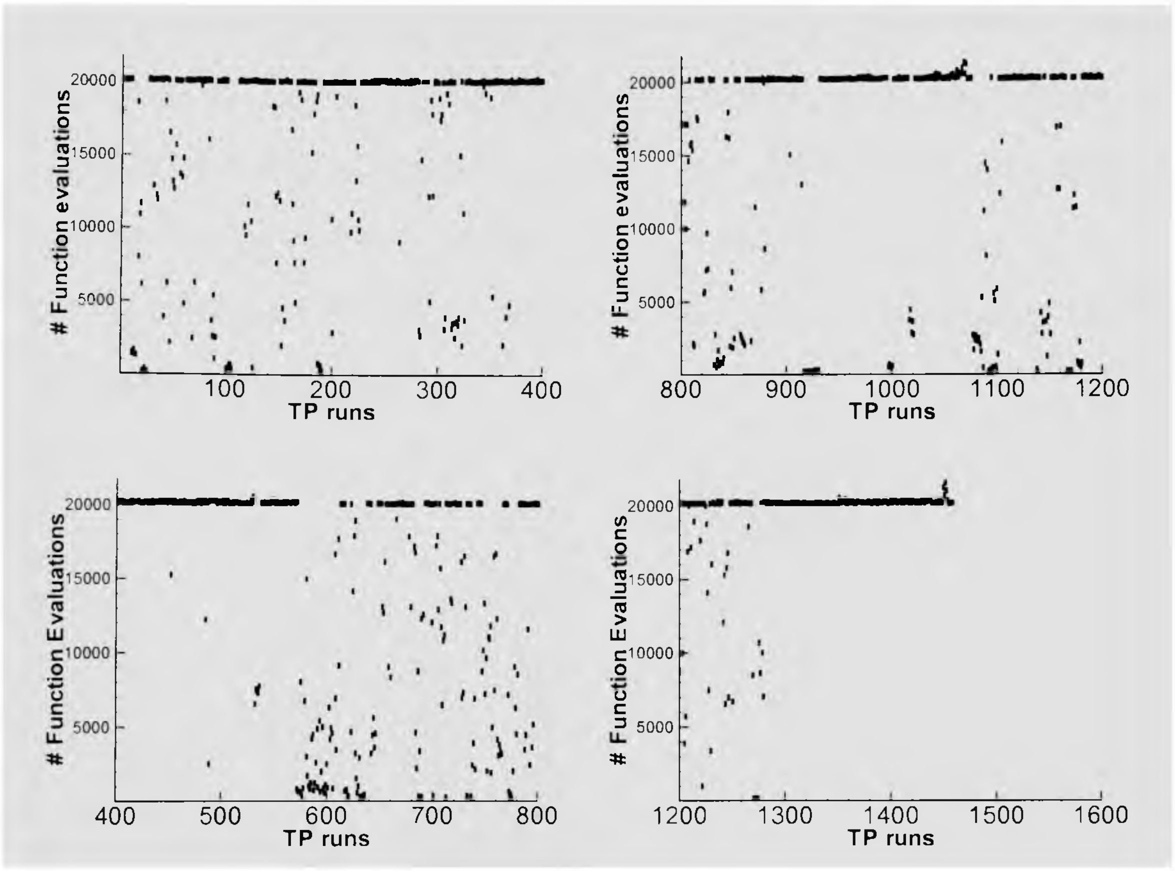

Figure 43 Number of function evaluations made for the 293 test problems (SOMPP Version-6).

It is seen from figure 41 that SOMPP Version-6 performs well in achieving relative errors of the order of less than 1.0, except for in cases which have a high number of design variables with unspecified variable limits. However, the most prominent improvement of this version of SOMPP is its ability to find the feasible space in case of constrained problems (as shown in figure 42) irrespective of the number and complexity of the inequality and equality constraints (whether linear or nonlinear). It should be noted that in many of these constrained problems the initial population is completely in the infeasible space. The inability to converge to the feasible space in case of the last few test problems can be attributed to the involvement of relatively high number of unbounded design variables (from 20 to 50) as seen from figure 5. The number of function evaluations exhausted by SOMPP Version-6 is relatively high as shown in figure 43, which is expected as a substantial amount of functions evaluations are consumed in successfully searching for the feasible space in case of constrained problems.

The improved performance of SOMPP Version-6 becomes more evident from the histogram presented in figure 44.

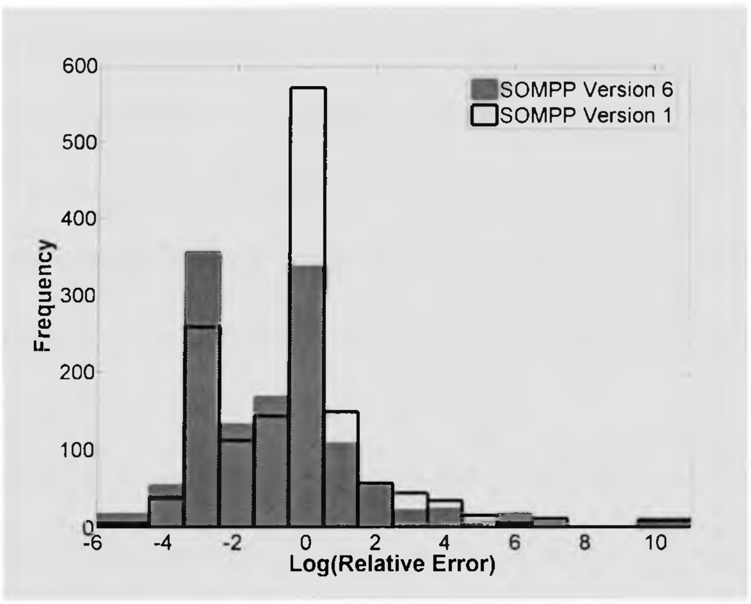

Figure 44 Comparison of the frequency of occurrence of different orders of magnitude of relative error in the computed minima between SOMPP Version-1 and Version-6

Here frequency relates to the number of test runs that converged to that particular order of magnitude of relative error. It is seen from figure 44 that in case of COMPP Version-6, noticeably more test cases have converged to relative errors of orders of magnitude less than 1.0 (higher histogram bars for $\log(\textit{relative error}) \leq 0$).

CHAPTER IV – CONCLUSIONS

Technological advancements in recent years have necessitated the efficient design of systems and processes in order to be competitive in the global market. However, design and performance optimization implemented during the same time has been limited to the use of experience and intuition of research personnel/field technicians and application of existing classical models in design. Optimization techniques/models on the other hand have undergone radical improvements, with the rise of evolutionary [5] and hybrid [12, 25] optimization algorithms and robust modeling/interpolating/pattern searching techniques such as response surfaces [29, 30], artificial neural networks etc. Application of such techniques to real life systems whether engineering/scientific systems or financial systems, demands efficient optimization concepts that are simplistic in execution, provide reliable solutions and are computationally inexpensive.

The modified predator-prey algorithm provides one such means of searching for optimal solutions. This algorithm, both in its multi-objective version and the single objective version, with added constraint handling modules, has been tried and thoroughly validated against test problems of different types. The pertinent analysis results show that this algorithm is competent in producing dependable optimal solutions, and for certain cases even does better than most well known algorithms presently available in literature. Performance of the constraint handling technique in driving solutions into the feasible domain at the expense of a reasonable number of function evaluations is also appreciable.

MPP employs the concept of weighted sum of objectives without any normalization of the objectives, which leads to relatively poor distribution of Pareto solutions in certain complex multi-objective cases. Nevertheless, the inclusion of the concept of sectional convergence using biased weighing of objectives and careful hypercube sizing ensures a desirable distribution of the Pareto solutions even for these poorly behaved cases.

Single-objective optimization problems posed without explicit decision variable limits (*i.e.*, unbounded problems) are likely to diverge. This issue was addressed by the relatively nascent concept of epidemic operator [27], where a significant portion of the candidate solution population is replaced by new randomly generated candidate solutions within a practical range depending on the problem. Equality constraints pose severe threats against convergence, especially in problems with high number of design variables, because they create an extremely constricted feasible region in a multi-dimensional search domain of high order. However, SOMPP handles such problems with acceptable accuracy, without the application of a computationally expensive penalty function method.

The modified predator-prey algorithm presents a concordant application of the basic traits of evolutionary algorithms, classical weighed sum approach and certain ingenious techniques such as sectional convergence, hypercube operator, epidemic operator, *etc* to single- and multi-objective problems (constrained and unconstrained). A combination of such distinct features is rare in optimization literature and provides a foundation to construct robust composite optimization algorithms with features adaptive to both the problem and the progress of the algorithm through the function space towards the Pareto front.

Future Work

The unconstrained multi-objective version of the modified predator-prey algorithm, *i.e.* MPP, is due to be incorporated into the hybrid optimizer MOHO developed by Moral and

Dulikravich [12]. The *modus operandi* of MPP is conceptually different from the existing algorithms in MOHO. As discussed before, the performance of MPP is comparable to the performance of MOHO on the ZDT test problems. Thereby, MPP is expected to contribute significantly to the versatility of MOHO in tackling complex real world optimization problems. This might demand some minor, but necessary changes to the C++ code simulating MPP. Addition of MPP to MOHO will be followed by testing the new MOHO with the same ZDT test cases to demonstrate the expected performance gain due to MPP and compute the percentage contribution of MPP in terms of function evaluations and execution time.

MPP is also in line to be applied on several real design problems presently under analysis within the MAIDROC research group. One of them is 'COOLNET', a project which involves generation of three-dimensional cooling networks [31] for cooling of electronic components. An efficient operation of such a thermo-fluids system calls for a design that incurs minimum pressure drop of the cooling fluid flowing through the network and ensures maximum heat extraction from the electronic components. This poses a multi-objective problem which will be addressed accordingly by MPP where choice of branching/sub-branching configuration, number of levels/branches, lengths of each of the branches, and cross sectional area of different branches are likely to be the design variables.

A complete multi-disciplinary design optimization package demands a combination of optimization algorithm(s) and a modeling/interpolating/pattern searching technique. Self Organizing Map (SOM) concept [32] is a type of artificial neural network that is trained using unsupervised learning to produce a low-dimensional (typically 2D), discretized map of the design space of the training samples. This makes SOM useful for visualizing low-dimensional views of high-dimensional data, akin to multidimensional scaling, thereby seeking to preserve the topological properties of the design space. The synergy of SOM and MPP has the potential to form a complete MDO package, capable of addressing real world design problems. Such hybrid software will be developed and validated with standard test problems. Subsequently, it'll also be applied to practical problems of the likes of alloy optimization using available experimental data [33], weather prediction using measured/recorded field data, *etc.*

REFERENCES

1. Zitzler, E., Deb, K. and Thiele, L., "Comparison of Multiobjective Evolutionary Algorithms: Empirical Results," Evolutionary Computation, Vol. 8, No. 2, 2000, pp. 173-195.

2. Deb, K., Thiele, L., Laumanns, M. and Zitzler, E., "Evolutionary Multiobjective Optimization," Advanced Information and Knowledge Processing, Springer, Berlin Heidelberg, 2006, pp. 105-145.

3. Hock, W. and Schittkowski, K., "Test Examples for Nonlinear Programming Codes," Lecture Notes in Economics and Mathematical Systems, Vol. 187, Springer Verlag, Berlin, Heidelberg, New York, 1981.

4. Schittkowski, K., "More Test Examples for Nonlinear Programming," Lecture Notes in Economics and Mathematical Systems, Vol. 282, Springer Verlag.

5. Deb, K., "*Multi-Objective Optimization Using Evolutionary Algorithms*," Chichester, UK, Wiley, 2002.

6. Laumanns, M., Rudolph, G. and Schwefel, H. P., "A Spatial Predator-Prey Approach to Multi-Objective Optimization: A Preliminary Study," in Proceedings of the Parallel Problem Solving from Nature, V, 1998, pp. 241-249.

7. Li, X., "A Real-Coded Predator-Prey Genetic Algorithm for Multi-Objective Optimization," Lecture Notes in Computer Science, Vol. 2632/2003, pp. 69.

8. Grimme, C. and Schmitt, K., "Inside a Predator-Prey Model for Multiobjective Optimization – A Second Study," In Proc. Genetic and Evolutionary Computation Conf.(GECCO 2006), Seattle WA, *ACM* Press, New York, 2006, pp. 707-714.

9. Silva, A., Neves, E. and Costa, E., "An Empirical Comparison of Particle Swarm and Predator-Prey Optimization," Lecture Notes in Computer Science; Vol. 2464, pp. 103-110.

10. Crossley, W. A., "Optimization for Aerospace Conceptual Design Through the Use of Genetic Algorithms," Proceedings of the First NASA/DoD Workshop on Evolvable Hardware, July 1999, pp. 200-207.

11. Coello Coello, C. A., Veldhuizen, D. A. V. and Lamont, G. B., "Evolutionary Algorithms for Solving Multi-Objective Problems," Genetic Algorithms and Evolutionary Computation, Vol. 5, Kluwer Academic Publishers, May 2002.

12. Moral, R. J. and Dulikravich, G. S., "Multi-Objective Hybrid Evolutionary Optimization with Automatic Switching among Constituent Algorithms," AIAA Journal, Vol. 46, No. 3, pp. 673-700, March 2008.

13. Vrugt, J. A. and Robinson, B. A., "Improved Evolutionary Optimization from Genetically Adaptive Multimethod Search," Proceedings of the National Academy of Sciences (U.S.A.), Vol. 104, No. 3, pp. 708–711, 2007.

14. Chowdhury, S., Moral, R. J. and Dulikravich, G. S., "Predator-Prey Evolutionary Multi-Objective Optimization Algorithm: Performance and Improvements", 7th ASMO-UK/ISSMO International Conference on Engineering Design Optimization, Bath, UK, July 7/8, 2008.

15. Sobol I. M., "Uniformly Distributed Sequences with an Additional Uniform Property," USSR Computational Mathematics and Mathematical Physics, Vol. 16, pp. 236-242, 1976.

16. Eshelman, L. J. and Schaffer J. D., "Real Coded Genetic Algorithms and Interval Schemata," In Foundations of Genetic Algorithms *2 (FOGA 2)*, pp. 187-202, 1993.

17. Michalewicz, J., "Genetic Algorithms + Data Structures = Evolutionary Programs," Berlin, Springer-Verlag, 1992.

18. Deb, K., Pratap, A., Agarwal, S. and Meyarivan T., "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II," IEEE Transactions on Evolutionary Computation, Vol. 6, No. 2, pp. 182-197, April 2002.

19. Fonseca, C. M. and Fleming, P. J., "An Overview of Evolutionary Algorithms in Multiobjective Optimization," Evolutionary Computation, Vol. 3, No. 1995, pp. 1-16.

20. Binh, T. T. and Korn, U., "MOBES: A Multiobjective Evolution Strategy for Constrained Optimization Problems," In The Third International Conference on Genetic Algorithms (Mendel 97), Brno, Czech Republic, pp. 176-182, 1997.

21. Osyczka, A. and Kundu, S., "A New Method to Solve Generalized Multicriteria Optimization Problems Using the Simple Genetic Algorithm," Structural Optimization, Vol. 10, pp. 94-99, 1995.

22. "User Guide", IOSO NM Version 1.0, IOSO Technology Center, 2003.

23. Rao, S., "Engineering Optimization: Theory and Practice," Third Edition, John Wiley Interscience, New York, 1996.

24. Dulikravich, G. S., Martin, T. J., Dennis, B. H., and Foster, N. F., "Multidisciplinary Hybrid Constrained GA Optimization," Chapter 12 in EUROGEN'99 - Evolutionary

Algorithms in Engineering and Computer Science: Recent Advances and Industrial Applications, Jyvaskyla, Finland, May 30 - June 3 1999, 231-260.

25. Colaço, M. J., Dulikravich, G. S., Orlande, H. R. B. and Martin, T. J., "Hybrid Optimization with Automatic Switching among Optimization Algorithms," in: Handbooks on Theory and Engineering Applications of Computational Methods: Evolutionary Algorithms and its Applications, eds. Onate, E. Annicchiarico, W. Periaux, Barcelona, Spain, CIMNE, 2005.

26. Colaço, M. J. and Dulikravich, G. S., "Solidification of Double-Diffusive Flows using Thermo-Magneto-Hydrodynamics and Optimization," Materials and Manufacturing Processes, Vol. 22, 2007, pp. 594-606.

27. Cuco, A. P. C., Neto, A. J. S., Velho, H. F. C. and de Sousa, F. L. D., "Solution of an Inverse Adsorption Problem with an Epidemic Genetic Algorithm and the Generalized Extremal Optimization Algorithm," Inverse Problems in Science and Engineering, 2008, 16(8):901-914.

28. http://www.geatbx.com/docu/fcnindex-01.html#P89_3085, The Genetic and Evolutionary Algorithm Toolbox for MATLAB (GETAbx) Documentation, version 3.8, Hartmut Pohlheim, 2006.

29. Colaco, M. J., Sahoo, D. and Dulikravich, G. S., "A Response Surface Method-Based Hybrid Optimizer," Inverse Problems in Science and Engineering, Vol. 16, issue 6, 2008, pp. 717-741.

30. Moral. R. J. and Dulikravich, G. S., "A Hybrid Self-Organizing Response Surface Methodology," paper AIAA-2008-5891, 12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Victoria, British Columbia, Canada, September 10-12, 2008.

31. Gonzalez, M. J., Jelisavcic, N., Moral, R. J., Sahoo, D., Dulikravich, G. S. and Martin, T. J. M., "Multi-Objective Design Optimization of Topology and Performance of Branching Networks of Cooling Passages," International Journal of Thermal Sciences, Vol. 46, pp. 1191-1202, 2007.

32. Kohonen, T., *Self-Organizing Maps*, Springer-Verlag, Berlin, 1995.

33. Egorov-Yegorov, I. N. and Dulikravich, G. S., "Chemical Composition Design of Superalloys for Maximum Stress, Temperature and Time-to-Rupture Using Self-Adapting Response Surface Optimization," Materials and Manufacturing Processes, Vol. 20, No. 3, May 2005, pp. 569-590.

34. Deb, K. and Rao, U. N., "Investigating Predator-Prey Algorithms for Multi-Objective optimization," KanGAL Report Number 2005010, IIT Kanpur, India.

APPENDICES

Appendix A: Performance of standard algorithms on the ZDT test cases [12]

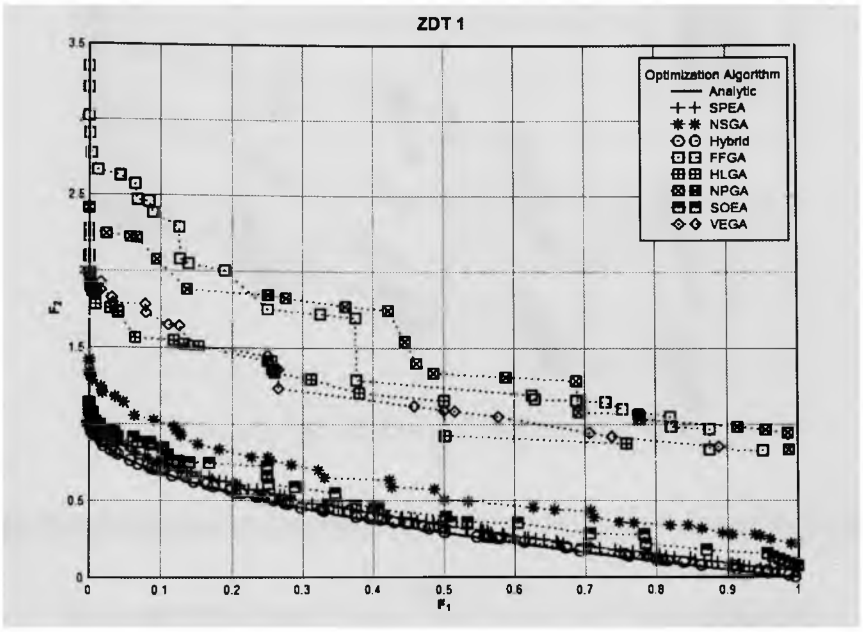

Figure 45 Results for test problem ZDT 1 using various standard algorithms

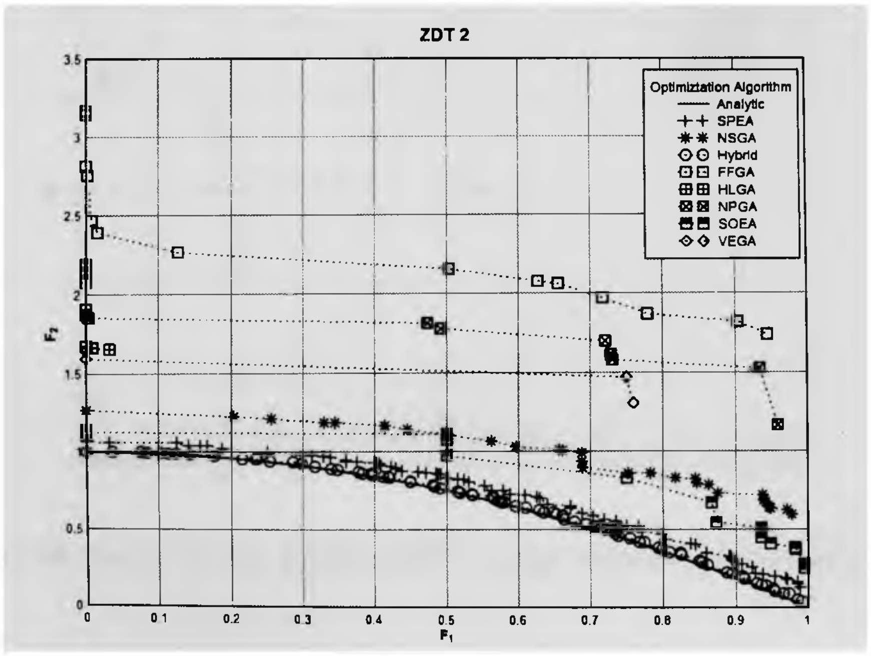

Figure 46 Results for test problem ZDT 2 using various standard algorithms

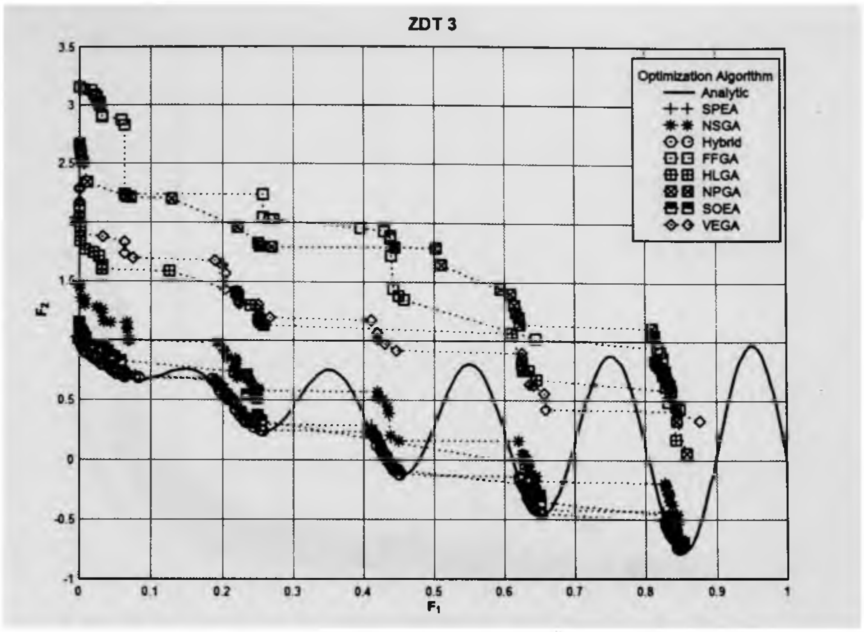

Figure 47 Results for test problem ZDT 3 using various standard algorithms

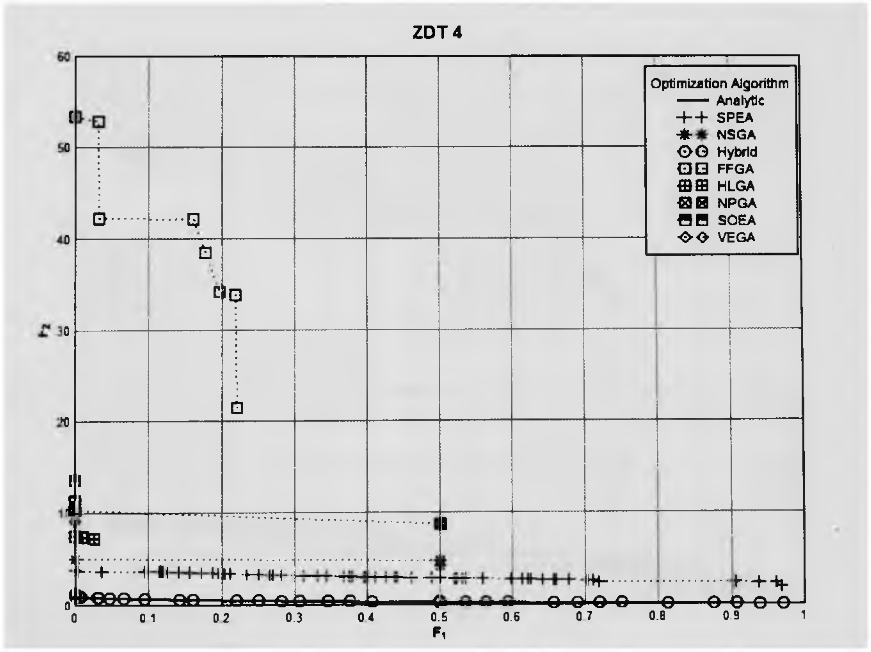

Figure 48 Results for test problem ZDT 4 using various standard algorithms

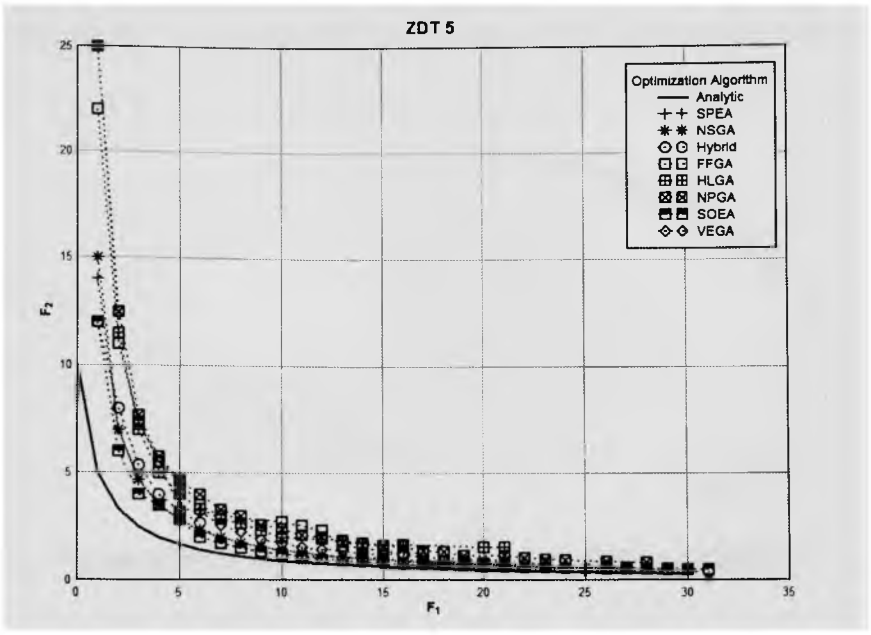

Figure 49 Results for test problem ZDT 5 using various standard algorithms

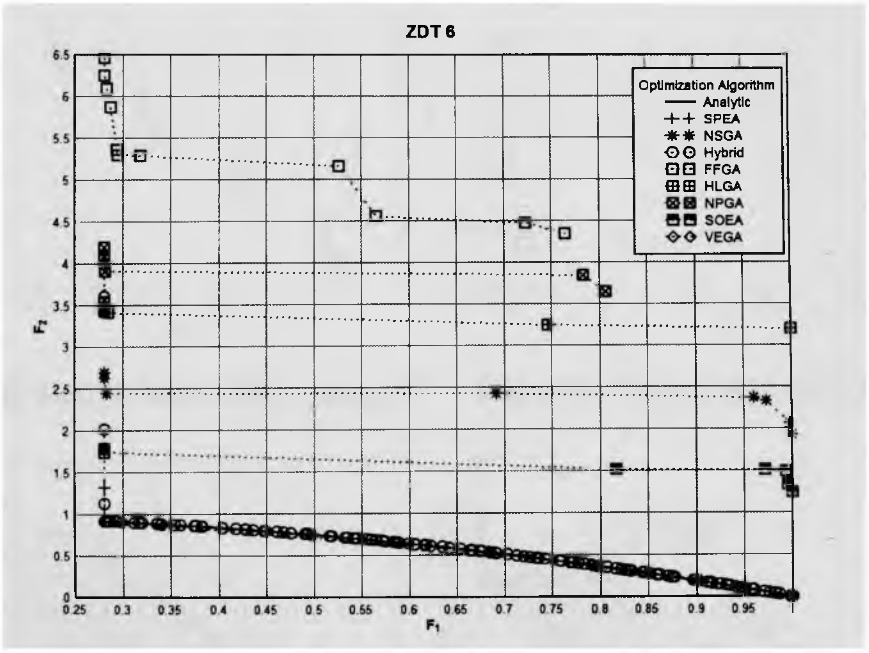

Figure 50 Results for test problem ZDT 6 using various standard algorithms

Appendix B: Performance of PP by Deb *et al.* [34] on standard test cases

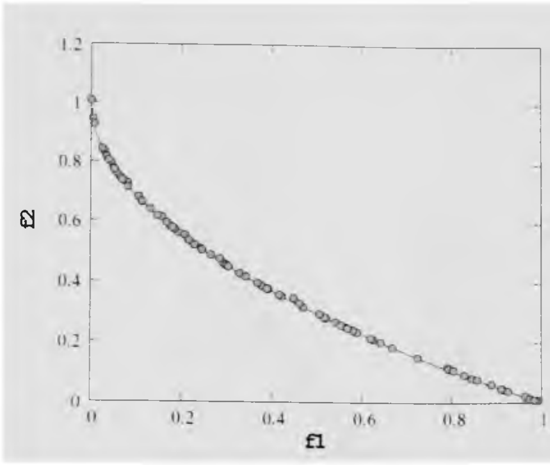

Figure 51 Test problem ZDT 1 using PP

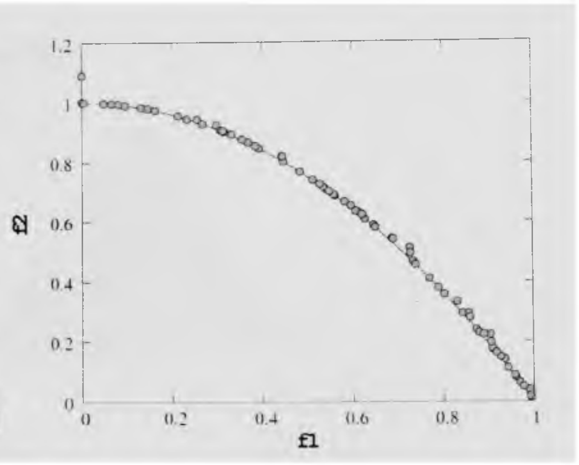

Figure 52 Test problem ZDT 2 using PP

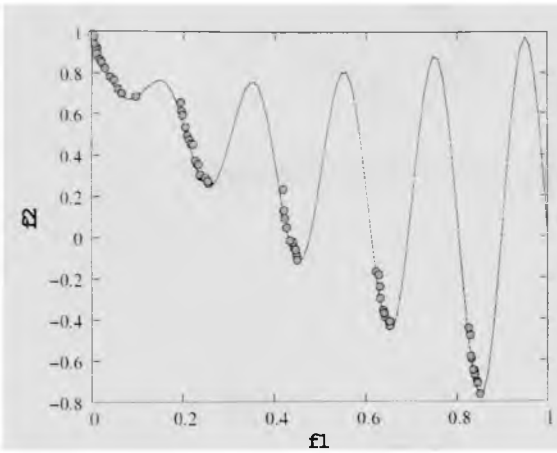

Figure 53 Test problem ZDT 3 using PP

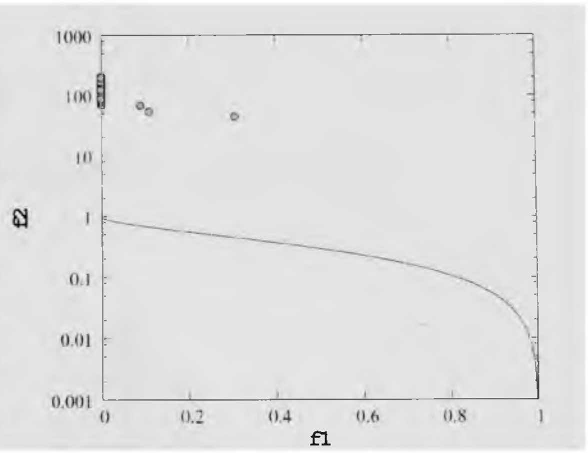

Figure 54 Test problem ZDT 4 using PP

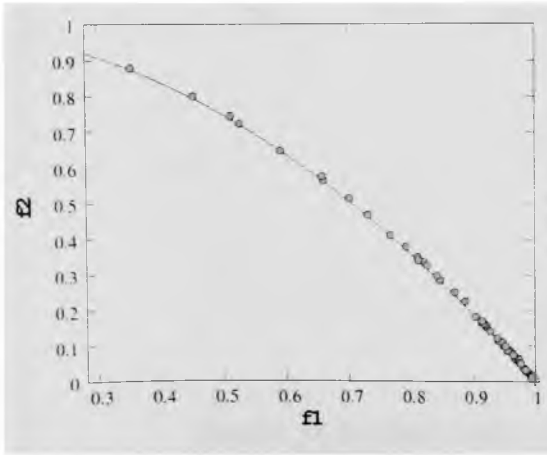

Figure 55 Test problem ZDT 6 using PP

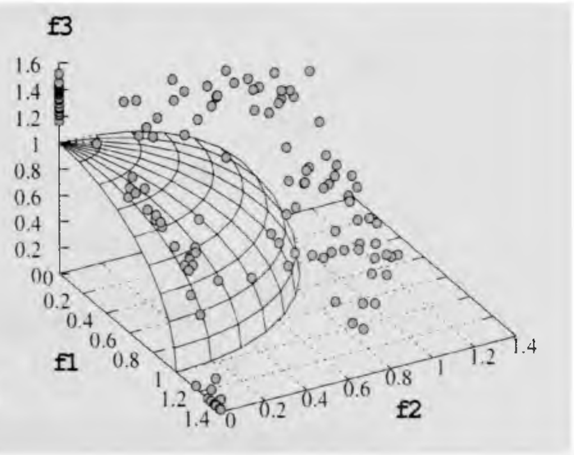

Figure 56 Test problem DTLZ 2 using PP

Appendix C: Performance of NSGA-II and SPEA on unconstrained 3-objective test cases [2]

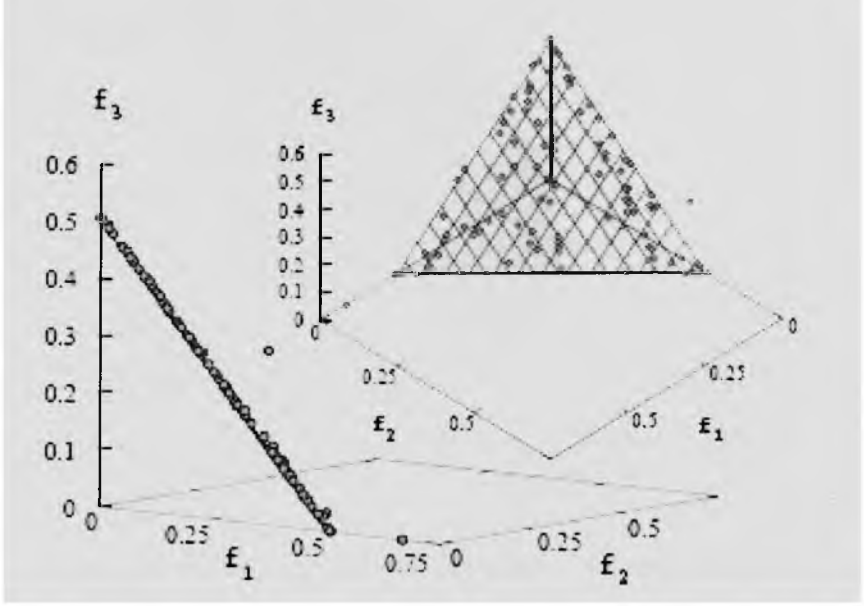

Figure 57 Results for test problem DTLZ 1 using NSGA-II

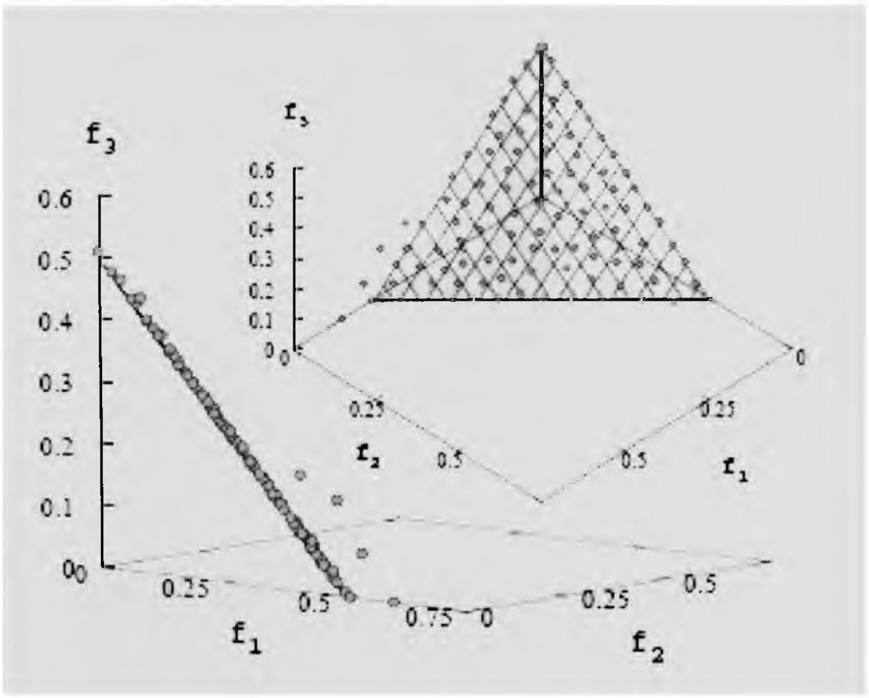

Figure 58 Results for test problem DTLZ 1 using SPEA

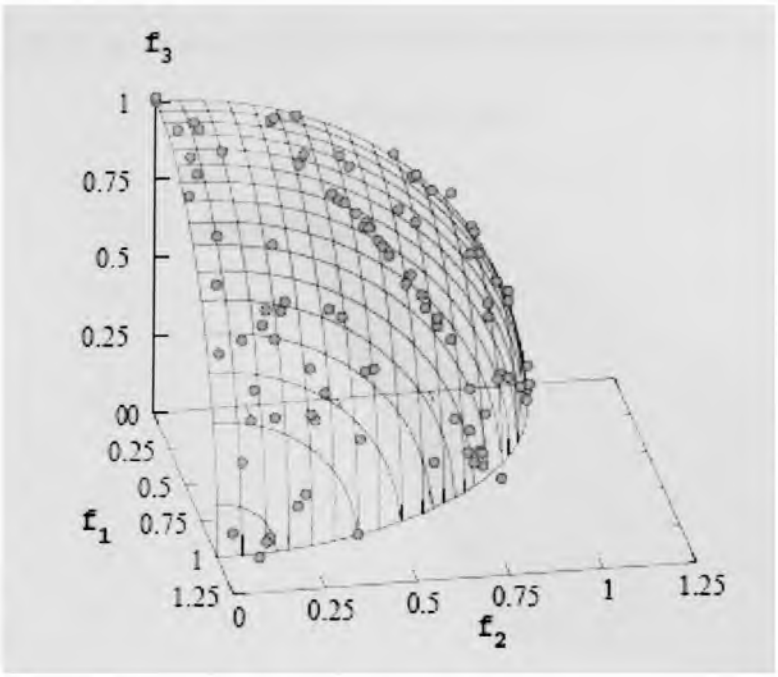

Figure 59 Results for test problem DTLZ 2 using NSGA-II

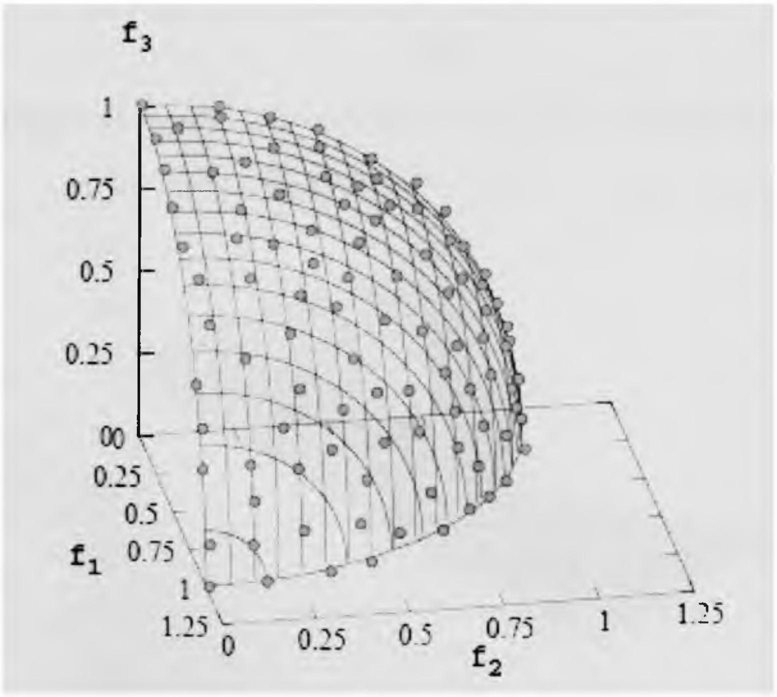

Figure 60 Results for test problem DTLZ 2 using SPEA

Appendix D: Performance of NSGA-II [18] and IOSO [22] on constrained 2-objective test cases

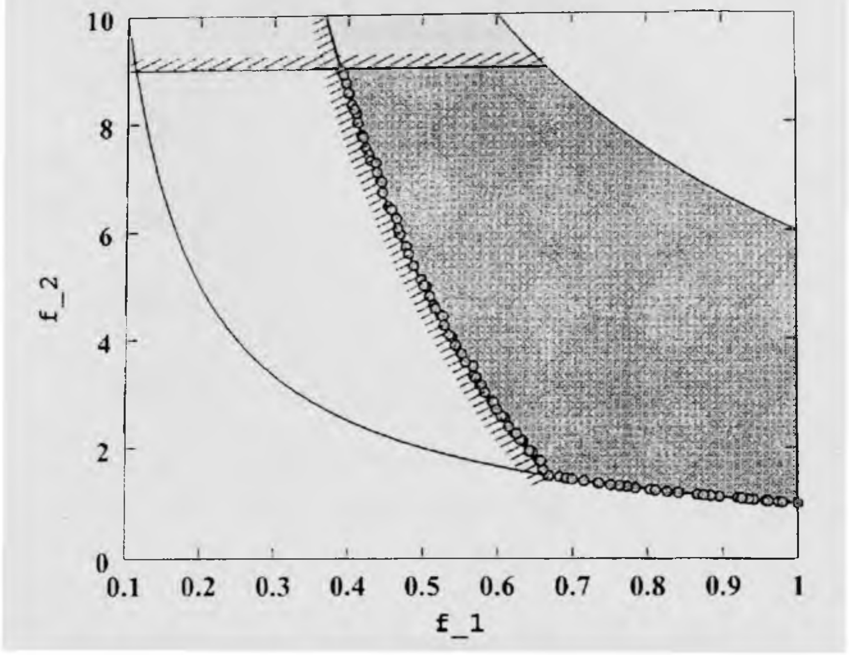

Figure 61 Results for test problem CONSTR using NSGA-II

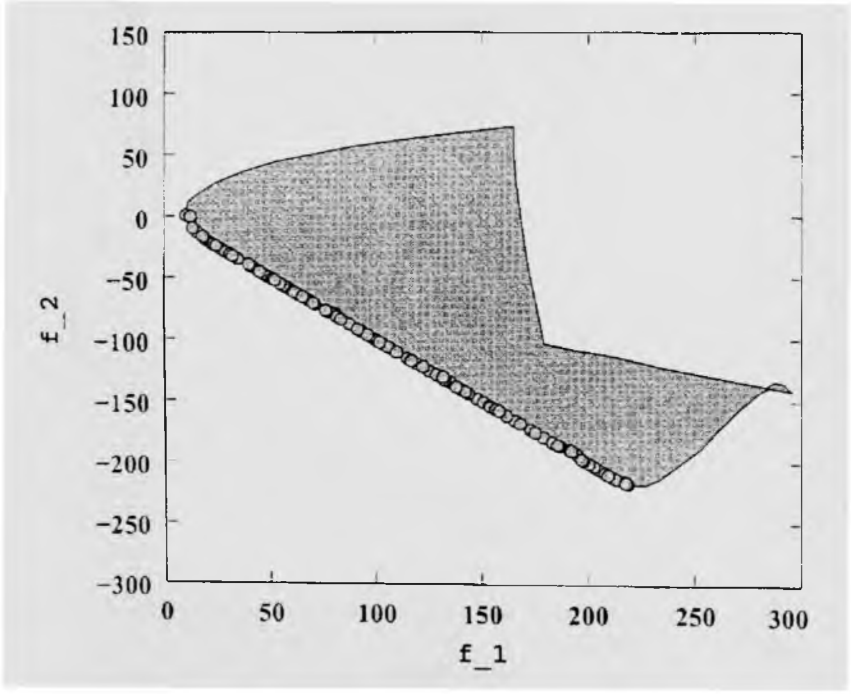

Figure 62 Results for test problem SRN using NSGA-II

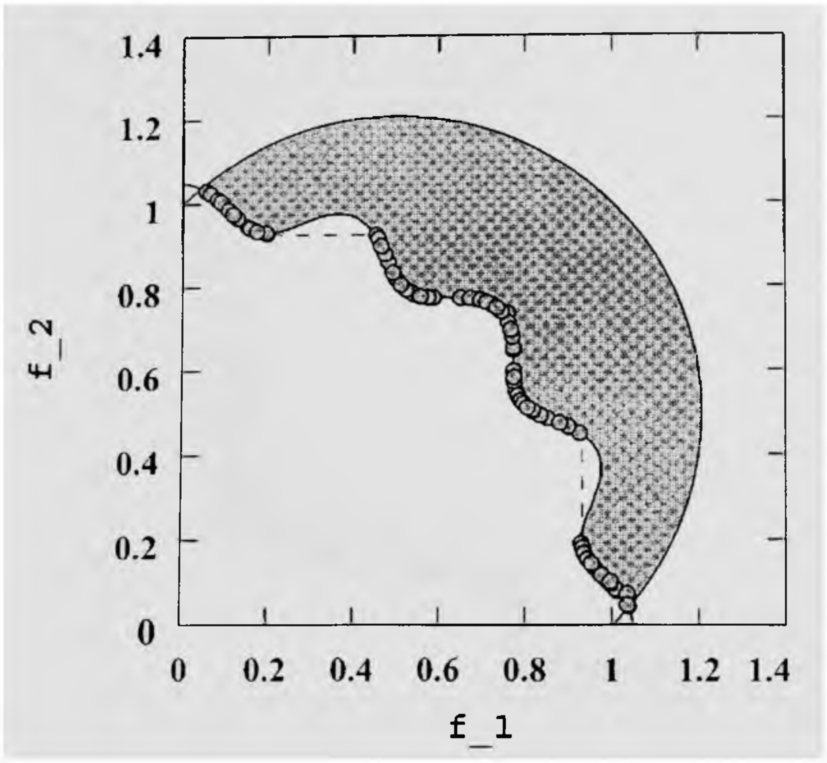

Figure 63 Results for test problem TNK using NSGA-II

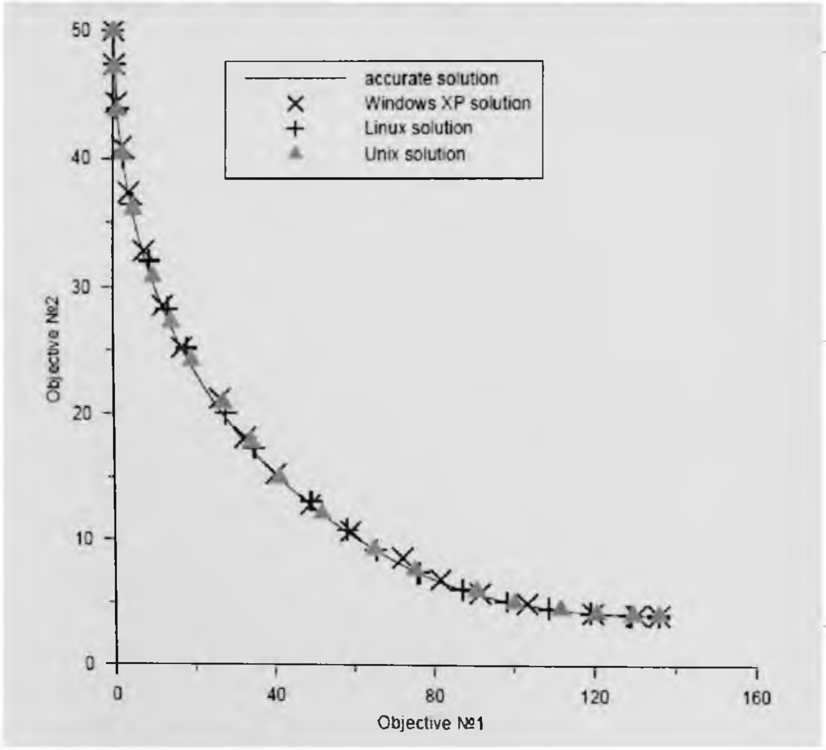

Figure 64 Results for Binh's multi-objective problem no. 2 using IOSO

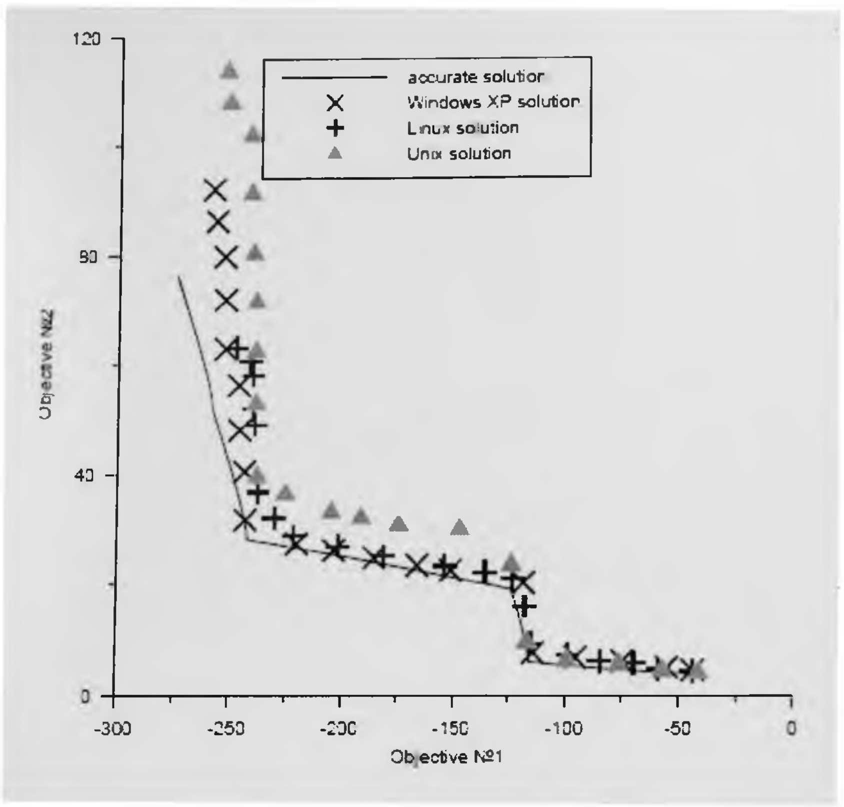

Figure 65 Results for Osyczka multi-objective problem no. 2 using IOSO