Fall 2017

Using Optimality Theory and Reference Points to Improve the Diversity and Convergence of a Fuzzy-Adaptive Multi-Objective Particle Swarm Optimizer

Amit A. Kulkarni
Old Dominion University

USING OPTIMALITY THEORY AND REFERENCE POINTS TO IMPROVE THE DIVERSITY AND CONVERGENCE OF A FUZZY-ADAPTIVE MULTI-OBJECTIVE PARTICLE SWARM OPTIMIZER

by

Amit A. Kulkarni
B.E. July 2006, Shivaji University, India
M.S. May 2011, Old Dominion University

A Dissertation Submitted to the Faculty of
Old Dominion University in Partial Fulfillment of the
Requirements for the Degree of

DOCTOR OF PHILOSOPHY

MECHANICAL ENGINEERING

OLD DOMINION UNIVERSITY

December 2017

Approved by:

Miltiadis Kotinis (Director)

Gene J.-W.Hou (Member)

Charles Keating (Member)

ABSTRACT

USING OPTIMALITY THEORY AND REFERENCE POINTS TO IMPROVE THE DIVERSITY AND CONVERGENCE OF A FUZZY-ADAPTIVE MULTI-OBJECTIVE PARTICLE SWARM OPTIMIZER

Amit A. Kulkarni
Old Dominion University, 2017
Director : Dr. Miltiadis Kotinis

Particle Swarm Optimization (PSO) has received increasing attention from the evolutionary optimization research community in the last twenty years. PSO is a metaheuristic approach based on collective intelligence obtained by emulating the swarming behavior of bees. A number of multi-objective variants of the original PSO algorithm that extend its applicability to optimization problems with conflicting objectives have also been developed; these multi-objective PSO (MOPSO) algorithms demonstrate comparable performance to other state-of-the-art metaheuristics. The existence of multiple optimal solutions (Pareto-optimal set) in optimization problems with conflicting objectives is not the only challenge posed to an optimizer, as the latter needs to be able to identify and preserve a well-distributed set of solutions during the search of the decision variable space. Recent attempts by evolutionary optimization researchers to incorporate mathematical convergence conditions into genetic algorithm optimizers have led to the derivation of a point-wise proximity measure, which is based on the solution of the achievement scalarizing function (ASF) optimization problem with a complementary slackness condition that quantifies the violation of the Karush-Kuhn-Tucker necessary conditions of optimality. In this work, the aforementioned KKT proximity measure is incorporated into the original Adaptive Coevolutionary Multi-Objective Swarm Optimizer (ACMOPSO) in order to monitor the convergence of the sub-swarms towards the Pareto-optimal front and provide feedback to Mamdani-type fuzzy logic controllers (FLCs) that are utilized for online adaptation of the algorithmic parameters. The proposed Fuzzy-Adaptive Multi-Objective Optimization Algorithm with the KKT proximity measure (FAMOPSOkkt) utilizes a set of reference points to cluster the computed nondominated solutions. These clusters interact with

their corresponding sub-swarms to provide the swarm leaders and are also utilized to manage the external archive of nondominated solutions. The performance of the proposed algorithm is evaluated on benchmark problems chosen from the multi-objective optimization literature and compared to the performance of state-of-the-art multi-objective optimization algorithms with similar features.

To Rashmi, my family and my friends.

ACKNOWLEDGMENTS

It is a pleasure to thank the many people who made this thesis possible. First and foremost I would like to express my gratitude and heartfelt appreciation to my advisor and mentor Dr. Miltiadis Kotinis. His technical expertise and guidance has helped me throughout my research. He was always there to help answer my questions and has always channeled me in the right direction. Throughout my thesis- writing period, he provided encouragement, sound advice and good teaching. I would like to thank my committee members, Dr. Gene J.-W. Hou, and Dr. Charles Keating for their time in serving on my thesis advisory committee and for their editorial comments.

I would like to thank my wife, Rashmi who has stood by me through all my travails, long hours of absence from home and through my fits of impatience and frustration. She gave me support, encouragement and unwavering love throughout this journey. I thank my parents for their faith in me and allowing me to be as ambitious as I wanted. I would like to express my gratitude to my parents-in-laws for their support. I could not have come far without them. To all of them, I dedicate this thesis.

I also extend my special thanks to Dr. Sebastiab Bawab, Chairman of the Mechanical and Aerospace Engineering, Dr. Ramamurthy Prabhakaran, Dr. Stacie Ringleb for their continuous support and encouragement. In addition, Ms. Diane Mitchell, Ms. June Blount for assisting me in many different ways and handling the paperwork.

I thank all my friends, Ajay Panditrao, Deep Kantaria, Pushkar Dixit, Ayush Khandelwal, Avinash Gosavi, Rohit Lambi, Ganesh Reddy, Chetan Jain , Dwiti Kantaria, and Francis Hauris for their encouragement and support.

Last but not the least; I would like to thank the Almighty for providing me with the opportunity to attain this level. Thanks again to everyone who has helped me throughout my journey.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

LIST OF ALGORITHMS

CHAPTER 1

INTRODUCTION

"Owing to this struggle for life, any variation, however slight and from whatever cause proceeding, if it be in any degree profitable to an individual of any species, in its infinitely complex relations to other organic beings and to external nature, will tend to the preservation of that individual, and will generally be inherited by its offspring" - Charles Darwin

1.1 Background

By definition, optimization is a process of making the best or most effective use of a situation or resource [1]. Mathematically, it means finding the minimum or the maximum value of a function. Most of our decisions in life are based on trade-offs between conflicting objectives. For example, when booking a flight we look for things like minimum cost, shortest possible route with minimum number of stops, enough transit time between airports, food quality etc. The ultimate choice depends on what we treat as our objectives and what we treat as constraints, i.e. flight time might be a constraint for one person, while cost might be for another. The bottom line is, everyone has to deal with multiple conflicting objectives and, thus, needs to make decisions considering the trade-offs that might arise in a particular situation.

Mathematical definitions of multi-objective thinking date back to the 18^{th} century, when Francis Y. Edgeworth (1845-1926) and Vilfredo Pareto (1848-1923) introduced the concepts of non-inferiority and trade-offs in the context of economics [2]. It was Edgeworth who introduced the concept of equilibrium by comparing two functions using the same axis, which were later brought into more understandable form by Pareto, commonly known as *Edgeworth box* [3] . Once an equilibrium is achieved, resource allocation is then performed by the efficiency criteria proposed

by Pareto, now known as *Pareto optimality*. *"The optimum allocation of the resources of a society is not attained so long as it is possible to make at least one individual better off in his own estimation while keeping others as well off as before in their own estimation"* [4].

According to Pareto's efficiency criteria, finding solutions to a multi-objective optimization problem (MOOP) involves computing all the globally non-dominated solutions.

The task of finding the globally non-dominated solutions, the *Pareto-optimal set* (PS), poses a number of challenges to an optimization algorithm. When the Pareto-optimal solutions cannot be obtained analytically, a numerical optimization algorithm is utilized, which provides a set of discrete non-dominated solutions when the stopping criteria have been met; typically, a prespecified number of iterations. These solution vectors need to be located as near as possible to the global Pareto-optimal front (PF), i.e., the representation of the PS in the objective function space, but also be well-distributed, i.e., cover the entire front in a uniform manner. In this way, the decision maker would be able to consider several design alternatives and select the solution that best matches their preferences. Therefore, solution diversity is an equally important requirement as solution optimality when evaluating the effectiveness of a multi-objective optimization algorithm.

Usually, a decision maker is involved in the process of selecting a solution among the available set of solutions if any decision-making is required. For this purpose, several multi-objective optimization methods are available and can be broadly classified into four categories, *no-preference*, *a priori*, *a posteriori*, and *interactive methods*. As the name suggests, *no-preference* methods do not call for the decision maker's involvement. *A priori* methods require preference input from the decision maker to adjust the search based on these preferences. In *A posteriori* methods, the decision maker chooses from the given set of solutions. Lastly, in *interactive* methods, optimal solutions are improved during each iteration based on the decision maker's feedback.

1.2 Evolutionary Algorithms

Rosenberg's work [5, 6] was seminal in identifying the potential of evolutionary algorithms to solve multi-objective problems, while Schaffer introduced the first working multi-objective evolu-

tionary algorithm [7], VEGA - 'Vector Evaluated Genetic Algorithm'. Since then, several methods have been developed to solve multi-objective optimization problems. Evolutionary computation techniques like Genetic Algorithms (GA) [5, 6, 7], Ant Colony Optimization (ACO) [8, 9], Evolutionary Algorithms (EA) [10, 11, 12, 13], Differential Evolution (DE) [14, 15, 16], and Particle Swarm Optimization(PSO) [17, 18] are the most prominent methods. Most of these are population based methods which emulate natural processes, i.e., evolution and swarm intelligence.

Typically, an evolutionary algorithm starts with a randomly generated initial population, i.e., set of solution vectors, whose fitness, i.e., the objective function is evaluated on each solution vector. Subsequently, it is subjected to biological functions like *reproduction* and *mutation*, to generate an evolved population with potentially increased fitness. The iterative process also incorporates a selection mechanism based on Pareto optimality, which may be implemented at the end of each generation (iteration) as a survival criterion, in addition to a means to maintain diversity among the population members. Both of these methods ensure that the algorithm converges to a diverse set of non-dominated solutions.

1.3 Reference Set Based Algorithms

The goal of any MOO algorithm is to to provide a diverse set of Pareto-optimal solutions to the decision maker; the latter is going to utilize their preferences/criteria in order to select one optimal solution. As mentioned in section 1.1, *preference based* or *interactive* certain solution methods have been developed that involve a decision maker, a task that actually helps to ease the difficulty in modeling a practical problem in a precise mathematical form [19]. These methods use a set of reference solutions to measure the quality of solutions (convergence and diversity). There are two main challenges in this approach. First, how to generate the reference set, and second, how to evaluate the quality of the solutions obtained using the reference set. Ample literature is available on preference-based multi-objective optimization algorithms that are capable of finding a preferred set of solutions near the reference point supplied by the decision maker [20, 21, 22, 23, 24]. It should be mentioned that there is no information available about the decision vectors of the reference set,

as they are predefined and supplied.

Another application of reference sets is to guide the population members along specific directions and towards the Pareto-optimal front, without focusing on a specific region of the front based on a decision maker's preferences. In this way, the reference set can be utilized as a diversity inducing, and preserving, mechanism of the multi-objective algorithm.

NSGA III [25, 26], one of the most recently developed evolutionary algorithms, replaces the crowding distance diversity-preservation operator used in NSGA II by a reference-set-based niche strategy. A predefined set of reference points is supplied and a scalarization function-*achievement scalarizing function* [19] is used to find the perpendicular distance to the reference lines joining the origin, i.e., the ideal point in a normalized objective function space, and the reference points. Each population member is, thus, associated with a reference point. (and the ones with fewer population members are preferred)

RVEA [24] is another recently proposed algorithm where the search is guided using predefined reference vectors. Similar to NSGA III, this algorithm first decomposes the objective function space and then implements a scalarizarion function known as *angle penalized distance* (APD), to maintain balance between solution convergence and diversity.

1.4 Achievement Scalarizing Function (ASF)

Scalarizing functions are classified under *a posteriori* methods for generating Pareto-optimal solutions. In this approach, a multi-objective problem is reformulated into a single objective optimization problem by means of scalarizing parameters or through a non-scaling approach that maintains the vector-valued objective function and uses other optimality concepts. A detailed survey on these methods is available in [27]. *Weighted sums* method can generate a special class of solutions, which appear at corner points of the set of available solutions [28]. *Achievement Scalarizing Functions* on the other hand can generate any nondominated solution.

1.5 Swarm Intelligence

Particle swarm optimization (PSO) algorithms emulate the social behavior of animals/insects: a swarm of bees, an ant colony, a flock of birds, a school of fish. Similar to evolutionary algorithms, they also maintain a population of solutions (particles) that perform a search of the decison variable space by solving mathematical equations involving the position and velocity vectors of the particles. The search is typically guided towards the optimal solution(s) using the personal best position (solution) found by each particle up to the current iteration and by the global or local best solution found by the entire swarm or subswarms, respectively. The original single objective PSO algorithm is attributed to Kennedy, Eberhart and Shi [29, 30]. Since then several variants of the PSO extending to multi-objective optimization have been developed [18, 31, 32, 33, 34, 35, 36, 37]. A recent comprehensive survey on PSO and its applications can be found in [38]. The additional challenges faced by multi-objective PSO algorithms (MOPSOs) include the update of the personal best and the global best solution, since there is no single optimal solution but a set of nondominated solutions to choose from. The search direction is typically governed by the swarm leaders that guide the particles towards the Pareto-optimal front. The selection of leaders affects both the convergence and diversity of the solutions. Various methods that address the leader selection process are available in the literature [39, 40, 35, 41, 42]. Furthermore, hybrid evolutionary/swarm intelligence algorithms have also been proposed. These algorithms implement evolutionary operators, e.g. a mutation operator can be utilized in order to compensate for premature convergence and maintain solution diversity [43, 44].

In this dissertation, the Adaptive Coevolutionary Multi-objective Particle Swarm Optimizer (ACMOPSO) [36] is utilized as the baseline algorithm and its performance with respect to solution convergence and diversity is improved. Two main mechanisms are used to achieve those goals. First, a set of reference points is used to cluster the non-dominated solutions and acts primarily as a diversity-preserving operator. Each cluster provides leaders to the associated swarm and is also utilized to evaluate the convergence of each members through appropriate metrics. The latter are

used as input to fuzzy logic controllers (FLCs) that adapt the PSO algorithmic parameters. The second mechanism corresponds to a Pareto-optimal-front proximity measure, which is utilized to measure the convergence of the computed nondominated solutions to the Pareto-optimal front.

1.6 Dissertation Outline

A review of the concepts of multi-objective optimization is given in Chapter 2, Section 2.1, followed by a literature review on multi-objective optimization algorithms in Section 2.2. A description of the ACMOPSO algorithm is provided in Section 2.3. Chapter 3 describes the computation of the Pareto optimal-front proximity measure based on the Karush-Kuhn-Tucker optimality conditions, following its derivation for single and multi-objective optimization problems [45, 46]. The performance of the ACMOPSO algorithm in terms of the aforementioned Pareto-optimal front proximity measure is evaluated on a selection of problems from the multi-objective literature and is presented in Chapter 4. A novel method enhancing the performance of the ACMOPSO algorithm is proposed and evaluated in Chapter 5. In Chapter 6, a comparison between the performance of the proposed algorithm and the performance of state-of-the-art multi-objective optimization algorithms is presented. Conclusions and directions for future research are provided in Chapter 7.

CHAPTER 2

MULTI-OBJECTIVE OPTIMIZATION

2.1 Multi-Objective Optimization

In case of single objective optimization problems we are concerned with finding the optimal decision variable(s) which minimize(or maximize) the objective function. In other words the focus is on the decision variable space. This is not the case when dealing with multi-objective optimization problems.We have to focus on the objective space, which is often of lower dimension than the decision variable space [19]. As mentioned earlier, MOOP has a set of solutions instead of one single solution due to the conflicting nature of the objectives. As proposed by Edgeworth [3] and later developed by Vilfredo Pareto [4], in multi-objective scenarios we cannot improve one objective without deteriorating the other at the same time. A formal definition of *Edgeworth-Pareto optimality* is given below.

When the objectives in a vector optimization problem are conflicting with each other, the resulting multiple optimal solutions correspond to a trade-off between these objectives. A constrained multi-objective optimization problem with M objectives, L inequality constraints, K equality constraints, I decision variables, and J parameters is formulated as (assuming minimization of all the objective functions):

$$\begin{aligned} \underset{m}{\text{minimize}} \quad & f(x) \\ \text{subject to} \quad & g_l(x) \leq 0, \quad l = 1, \ldots, M, \\ & h_k(x) = 0, \quad k = 0, \ldots, K \end{aligned} \tag{2.1}$$

The set of optimal solutions is called the globally Pareto-optimal set of solutions of the feasible search space S. The members of this set correspond to solutions that are not dominated by any other

solution inside S. The concept of dominance can be defined as follows [47]: Solution x_1 dominates solution x_2 if solution x_1 is not worse than solution x_2 in all objectives and is strictly better than solution x_2 in at least one objective. When an optimization algorithm solves a constrained multi-objective problem with M objectives and L+K constraints, it produces a set of non-dominated solutions. In order a solution vector **x** to be globally Pareto-optimal, it is necessary that it satisfies the Karush-Kuhn-Tucker conditions (KKT). If the feasible search space and the objective functions are convex, then the KKT conditions are also sufficient for Pareto optimality [48].

2.1.1 Pareto dominance

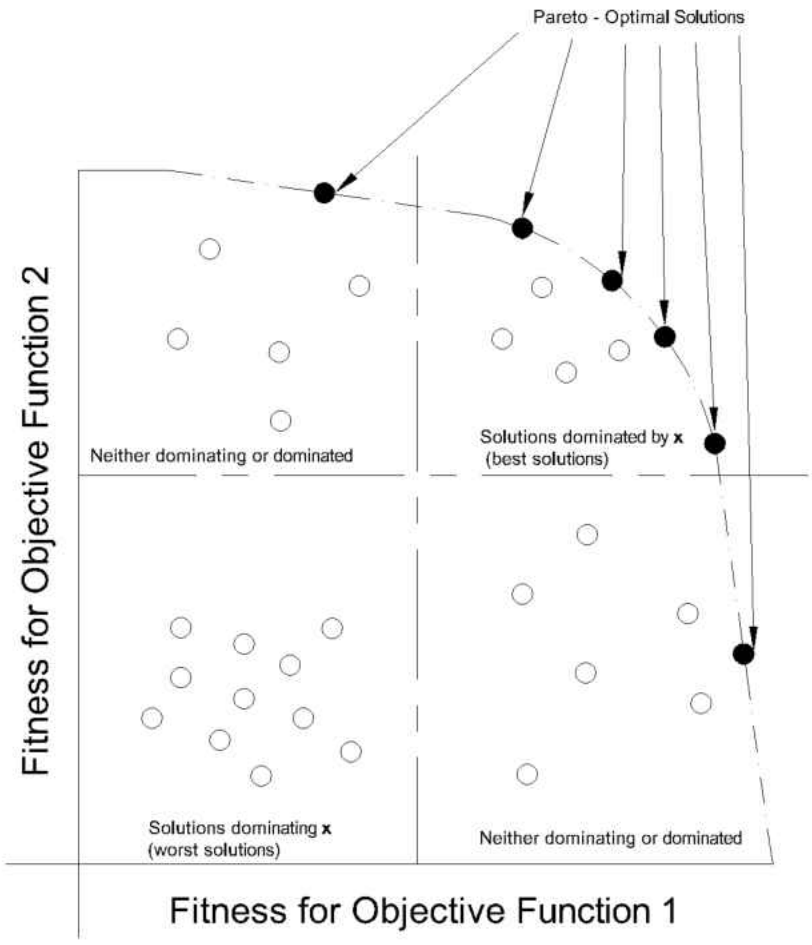

Figure 2.1: Concept of Pareto Optimality.

In unconstrained multi-objective optimization problems, a solution vector A dominates another solution vector B, if and only if the following two conditions for Pareto dominance are satisfied:

- The objective vector that corresponds to solution $A(x_A)$ is no worse than the objective vector that corresponds to solution $B(x_B)$ in all objectives
- Solution A is strictly better than solution B in at least one objective

If x_A is not dominated by any other solution, it is called a Pareto-optimal solution. The corresponding objective function vector belongs to the set of vectors that comprise the Pareto-optimal front. In constrained problems, x_A constraint-dominates x_B if any of the following conditions are satisfied [16]:

- Both x_A and x_B are feasible and x_A dominates x_B, based on the aforementioned conditions for Pareto dominance.
- Both x_A and x_B are infeasible but x_A has smaller constraint violation
- Solution x_A is feasible and solution x_B is not

All Pareto-optimal solutions are feasible, but not all feasible solutions are Pareto-optimal. It should be mentioned that the above definition is of *global Pareto optimality*. Unless any specific requirement is met, all obtained Pareto solutions are only locally optimal, i.e. if the objective functions and constraints meet the convexity criteria, they are globally Pareto-optimal. This set of solutions is called the *Pareto-Optimal Set* (PS). Figure 2.1 gives a graphical representation of these concepts.

2.1.2 Ideal and nadir vectors

An *ideal vector* minimizes each of the objective functions simultaneously, whereas the upper bounds of the PS form the *nadir vector*. It should be noted that the nadir vector might be feasible or infeasible. The ideal vector is usually unattainable and depends on the nature of problem at hand (convex or nonconvex).

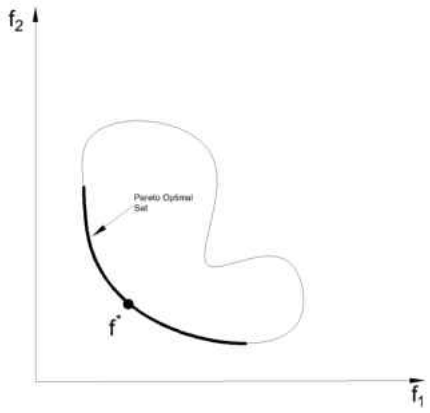

Figure 2.2: Objective Space.

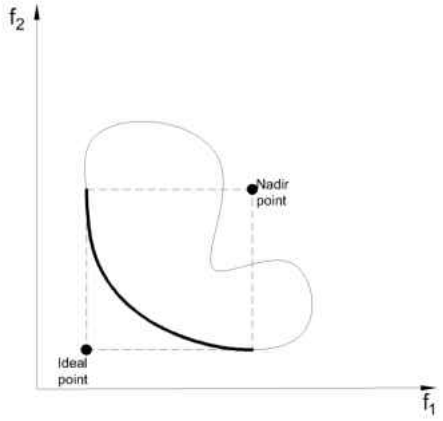

Figure 2.3: Ideal and Nadir Objective Vector.

Constrained or unconstrained, one can solve each objective function individually and then form the ideal vector.

2.2 Multi-Objective Optimization Algorithms

Several algorithms based on Evolutionary Computation(EC) have been developed in the past few decades. These algorithms mimic natural processes like evolution and collective intelligent behavior.

2.2.1 Evolutionary algorithms

Evolutionary algorithms (EA), as the name suggests are nature-inspired algorithms based on Darwinian principle of *"Survival of the fittest"*. Classified as a part of computational intelligence, they are population based, derivative free approaches for global optimization. Since their inception (late 1950's), EA have been widely used in diverse disciplines to solve single objective and multi-objective optimization problems. Although, easy to implement and configure, they are still critiqued for the theoretical aspects of convergence or the optimality of obtained solution(s).

2.2.2 Differential evolution

A standard Differential Evolution algorithm (DE) maintains and evolves a population of individuals (solutions). A new solution can be generated by performing crossover, where some components of an individual are replaced with a linear combination of three other members in the population. Selection or acceptance criterion is imposed for overall improvement of the population, where a current solution is replaced by a better or improved one. Although, DE algorithms have been shown to converge onto a single point for strictly convex objective functions, the converged point need not be a global minimizer. For more general functions, it might occur that the entire population gets stuck in a local optima, which would prevent the further progress of the population towards a global minimizer. Following is a pseudo-code for an standard DE.

Algorithm 1 The Classic Differential Evolution Algorithm.

Generate initial population of individuals
while termination criteria not met **do**
  For each individual j in the population
  Choose three random vectors r_1,r_2, and r_3 such that, $1 \leq r_1, r_2, r_3 \leq N$ with $r_1 \neq r_2 \neq r_3 \neq j$
  Generate a random integer $k_{rand} \in (1, N)$
  for each parameter k **do**
    if $k_{rand} \leq Cr$ or $k = k_{rand}$ **then**
      $u_{j,i} = v_{j,i} = x_{j,r0} + F * (x_{j,r1} - x_{j,r2})$
    else
      $u_{j,i} = x_{j,i}$
    end if
  end for
  Select the next generation by a tournament selection between parent and child
end while

$F \in (0, 1+)$ is the scale factor and $Cr \in [0, 1]$ is the control variable. The scale factor controls the mutation step size and in turn the population's convergence speed. The control variable is the crossover probability which is determined by the average number of parameters that the trial vector $u_{i,j}$ inherits from the mutant vector $v_{j,i}$. It was found in that rotating the coordinate system relocates some trial vectors, but the position of mutant trial vector is rotationally invariant, i.e., when $Cr = 1$, essentially there would be no crossover. Without crossover, the classic DE algorithm

performs poorly in multimodal functions [49]. In our investigation, we implement a rotationally invariant version of the DE, the *DE/rand/1/either-or* developed in [50] in order to compute the KKT proximity measure following the procedure described in Chapter 3.

2.2.3 The DE/rand/1/either-or algorithm

In this DE version, trial vectors that are pure mutants occur with a probability p_F and pure recombinants occur with a probability $1-p_F$. The scheme for trial vector generation is given below. Price *et al.* [49] suggest a value of K as $K = 0.5(1 + F)$, for a given value of F.

Algorithm 2 The Either/Or Differential Evolution algorithm.

if $rand_i(0, 1) \leq P_F$ **then**
  $\mathbf{u}_i = \mathbf{x}_{r0} + F * (\mathbf{x}_{r1} - \mathbf{x}_{r2})$
else
  $\mathbf{u}_i = \mathbf{x}_{r0} + K * (\mathbf{x}_{r1} + \mathbf{x}_{r2} - 2 * \mathbf{x}_{r0})$
end if

2.2.4 Swarm intelligence

Swarm Intelligence (SI) is the collective behavior of decentralized, self-organized systems, natural or artificial [31]. SI algorithms derive inspiration from natural processes like the behavior of bee swarms while searching for food source. The bees do not follow orders from any centralized system, but rely on interaction between each other. Similar behaviors are observed in ant colonies, flock of birds, school of fish and many other natural systems.

Boids [51] was the first program to mimic swarm behavior. The first published literature on agent based global search and optimization technique mimicking swarm behavior was Stochastic Diffusion Search (SDS) [52]. Ant Colony Optimization (ACO) was proposed by Dorigo [9] which was a collection of optimization algorithms modeled on interactions within an ant colony. Particle Swarm Optimization (PSO) algorithm, an global optimization method based on bird flocking was proposed by Kennedy and Eberhart [53]. The latter two have gained much popularity in single objective and multi-objective optimization. A multi-objective version of PSO based on Pareto

dominance, MOPSO was proposed by [54]. A relaxed form of Pareto dominance relation was implemented in [55] to increase pressure selection towards the true Pareto front. NSGA II was combined with Control of Dominance Area of Solutions (CDAS), a modified Pareto dominance relation was developed in [56]. Influence of CDAS was then investigated in [57] on two other versions of MOPSO - the SMPSO [58] and SigmaMOPSO[59]. ACMOPSO proposed in [36] uses co-evolution of multiple swarms along with mutation and elitism to efficiently explore and exploit the search space using only the social components as guides. Another class of recently proposed MOPSOs make use of reference set to guide the swarm towards the Pareto front with an effort to preserve the diversity [60, 61].

The MaOPSO proposed in [60] is an archive based PSO which uses a set of dynamically generated reference points to guide the search process to converge to the true Pareto front. Its working is similar to the evolutionary algorithm [25, 26] which is the third version of NSGA III using a clustering operator instead of the crowding distance operator as in NSGA II. The difference in both algorithms being, NSGA III does not employ any explicit reproduction selection operation, while MaOPSO employs a Pareto dominance and information about density and proximity to push the particles towards the Pareto front. A detailed explanation about this algorithm is discussed as both of them use reference points, some sort of scalarizing function, and preference information for guiding solutions towards the true Pareto front. The proposed method in this study uses ACMOPSO as a base algorithm since it has been investigated as an efficient algorithm for parallelization.

2.3 Description of the ACMOPSO Algorithm

Solutions of the multidisciplinary design problem were obtained using the ACMOPSO optimizer [50]. ACMOPSO, which is a co-evolutionary PSO algorithm, explores the design variable space using the search mechanism of PSO combined with random mutation. In every iteration, the swarm is divided into a number of sub-swarms; each sub-swarm focuses on a specific region of the computed Pareto-optimal front, which consists of the non-dominated solutions stored in the external archive. The latter of which, is divided into segments equal to the number of sub-swarms.

Two members of the Pareto set within each segment are randomly selected to act as leaders of the k^{th} swarm particle for each design variable j:

$$v_{kj}(t+1) = w.v_{kj}(t) + c_1.(x_{ndlr1,j}(t) - x_{kj}(t) + c_1.rand_j(0,1).(x_{ndlr2,j}(t)) - x_{kj}(t)) \tag{2.2}$$

where v_{kj} is the movement of particle k along position coordinate j ,and $x_{ndlr2,j}(t)$ are the j^{th} position coordinates of the two randomly selected leaders, $x_{kj}(t)$ is the corresponding j^{th} position coordinate of the k^{th} particle, w is the inertia weight, c_1 is the social coefficient, and $rand_j(0,1)$ is a random number uniformly distributed in (0,1). The k^{th} particle's position is updated in every iteration $(t+1)$ by adding the velocity vector over a single time increment to the current position vector :

$$\mathbf{x}_k(t+1) = \mathbf{x}_k(t) + \mathbf{v}_k(t+1) \tag{2.3}$$

This procedure is followed for 80% of the swarm particles. The remaining particles are substituted with randomly selected non-dominated archived solutions, which are subsequently mutated with a mutation rate $p_{mut} = 10\%$. The inclusion of a mutation operator, combined with particle substitution, enhances the algorithm's capability to solve problems with multiple local Pareto-optimal fronts, and also counterbalances the high selective pressure due to the utilization of a Pareto ranking procedure to manage the external archive. It needs to be mentioned that if the number of non-dominated solutions exceeds the nominal capacity of the archive, a crowding distance operator [16] is employed to maintain the solutions that reside in the least crowded areas.

The values of algorithmic parameters, w and c_1, are adapted on-line for each sub-swarm using the feedback provided by two metrics. The first metric takes into account the effectiveness of each sub-swarm in producing new non-dominated solutions during the current iteration. Second metric represents the ability of the entire swarm to produce solutions capable of entering the external archive.

In the current research project, a parallelized version of ACMOPSO with five sub-swarms

was utilized. The parallelization, which involved the computation of the objective functions and constraints, was done through the OpenMP interface using a work-sharing *paralleldo* construct. The results reported in [50] regarding an application of the parallelized version of ACMOPSO to an engineering design optimization problem demonstrated near-linear speedup and high parallel efficiency.

Algorithm 3 The ACMOPSO algorithm.

Initialize positions and velocities of particles(use a random seed for each sub-swarm)
Initialize iteration counter ($j = 0$) and external archive
do
for each sub-swarm **do**
    Evaluate objective functions and constraints
end for
Combine the archived solutions with the new solutions generated by the sub-swarms
Rank the non-dominated solutions and *update* the external archive
Partition the Pareto-optimal front into a number of segments equal to the number of sub-swarms
if the number of archived solutions within a segment exceeds threshold value **then**
    Compute the crowding distance of the archived solutions and *use* it as a leader selection criterion
end if
Generate the set of leaders for each sub-swarm by selecting archived solutions from the corresponding segment
for each sub-swarm **do**
    for each particle in the sub-swarm **do**
        if $rand(0, 1) \leq p_f$ **and** more than one leaders are available **then**
            Select two different leaders(randomly)
            for each decision variable j **do**
                Update the corresponding velocity component
                Compute the new position coordinate
            end for
        else
            Select one leader (randomly, if more than one are available)
            Select a decision variable , j_{rand}(randomly)
            for each decision variable j ,***if*** $(rand)(0, 1) \leq \eta$ **or** $j = j_{rand}$ **do**
                Mutate the j_{th} component of the leaders
            end for
            Replace particle with mutated leader
        end if
    end for
    Update the values of the adaptation parameters
    Adapt the inertia weight and social coefficient values
end for
$t = t + 1$
until stopping criterion is satisfied

CHAPTER 3

KKT PROXIMITY MEASURE

3.1 Necessary Optimality Conditions

The convergence criteria for multi-objective optimization algorithms is based on Pareto dominance, and their termination criteria is based on either the number of iterations or the quality indicators like hypervolume (HV), generational distance (GD) or the inverted generational distance (IGD). The hypervolume requires a pre-defined reference point for its computation, which cannot be set for an unknown problem, which makes it difficult to use it as a termination criteria. IGD, requires the knowledge of true Pareto-optimal solutions and their corresponding objective values, and again cannot be used in case of an unknown problem. Along with achieving convergence, an optimization algorithm should be able to maintain diversity among the obtained solutions.

The Karush-Kuhn-Tucker (KKT) conditions are able to check whether the obtained solution(s) is truly an optimal solution or not. Any point that satisfies these conditions is called as an optimal solution. It should be mentioned that any KKT point is an optimal point, but not every optimal point is a KKT point. Violation of the KKT conditions does not give any information regarding the solution's proximity to the optimal solution. The KKT conditions are singluar conditions that require the first-order derivatives of the objective functions and the constraints.

Dutta et al. proposed a KKT proximity measure for a single objective optimization problem [62]. In their work they performed meticulous calculations to prove the theoretical correctness of the proposed metric which calculates the proximity of the obtained solution to the optimal solution. Deb et al. extended this work to multi-objective optimization combined with achievement scalarizing functions and proposed a faster way of calculating KKT proximity measure [45, 46]. The KKT proximity measure is capable to identify relative closeness of any point from the theoretical optimum

point without actually knowing any information about the exact location of the optimum point. A smaller value of KKT proximity measure for an iterate indicates its closeness to the optimal point, a large value of KKT proximity measure indicates slow convergence in the region [63].

For nondominated solutions their closeness to the true Pareto front are likely to be different. Not all nondominated solutons are closer to the Pareto front, and even if they are, they might not be truly Pareto optimal. In another case, the solutions can be away from the true Pareto front and still be nondominated to the rest of the solutions. This enables the use of KKTPM as a quantitive measure for solutions on a nondominated front parallel to the true efficient front, therby providing an equal metric value near Pareto-optimal solutions.

The theoretical optimality of the solutions obtained by evolutionary algorithms (EA) is an open topic for criticism. It is required for every Pareto-optimal solution to satisfy the KKT conditions. Dutta et al, defined an approximate KKT solution to compute a KKT proximity measure for any iterate x^k for a multi objective constrained optimization problem of the type (2.1) for which, the Karush-Kuhn Tucker (KKT) optimality conditions are given as

$$\sum_{k=1}^{M} \lambda_k \nabla f_k(x^k) + \sum_{i=1}^{m} u_i \nabla g_i(x) = 0 \tag{3.1}$$

$$g_i(x^k) \leq 0, \quad \forall_i \tag{3.2}$$

$$u_i g_i(x^k) = 0, \quad \forall_i \tag{3.3}$$

$$u_i \geq 0, \quad \forall_i \tag{3.4}$$

$$\lambda_k \geq 0, \quad \forall_k, \quad and \quad \lambda \neq 0 \tag{3.5}$$

where, u_i is the Lagrange multiplier for the i-th constraint. Equation (3.1) is the equilibrium condition or gradient condition, equation (3.2) ensures feasibility for $\bar{x}$, equation (3.3) is the complimentary slackness condition, and equation (3.4) tells that the Lagrange multipliers are non-negative.

A Pareto-optimal solution must always be a KKT point, but a KKT point needs not to be a

Pareto-optimal solution. A KKT point becomes a Pareto-optimal solution if it does not violate the above convexity condition i.e. the feasible search space is convex and all objective functions are convex.

Information about the proximity and direction of the optimum from a given point, and using some metric derived from the KKT condition violations could be very helpful for devising a theoretically motivated termination condition which could be used in the actual design of the algorithms. Although, it is tempting to use the KKT conditions as a performance criterion for the algorithms, the extent of violations of these conditions close to the KKT point is not smooth, which makes it difficult to implement. To overcome this problem, Dutta *et al* proposed a new KKT proximity measure, based on the KKT error which can be computed by solving the following optimization problem for optimumul values of λ and μ. Doing so, enable equations (3.2), (3.3), (3.4), and (3.5) to be satisfied and the equation (3.1) to be least violated. This new proximity measure is defined by relaxing the complimentary slackness and equilibrium equations of KKT conditions defining a modified ϵ-KKT point. Dutta *et al* , defined an approximate KKT solution to compute a KKT proximity measure for any iterate x_k

$$\begin{aligned} \text{minimize} \quad & \| \sum_{k=1}^{M} \lambda_k \nabla f_k(x^k) + \sum_{i=1}^{m} u_i \nabla g_i(x) \| \\ \text{subject to} \quad & g_i(x^k) \leq 0, \quad \forall_i, \\ & u_i g_i(x^k) = 0, \quad \forall_i, \\ & u_i \geq 0, \quad \forall_i, \\ & \lambda_k \geq 0, \quad \forall_k, and \quad \lambda \neq 0 \end{aligned} \tag{3.6}$$

The KKT error for a feasible iterate x^k is then defined as

$$KKT \quad Error \quad (x^k) = \| \sum_{k=1}^{M} \lambda_k \nabla f_k(x^k) + \sum_{i=1}^{m} u_i \nabla g_i(x) \| \tag{3.7}$$

where λ_k^* and μ_k^* are the optimal solution to the problem stated in the above equation.

Drawbacks of this naive KKT proximity measure were highlighted in [45], proving that the KKT error does not monotonically reduces as the iterate approaches the KKT point, and this method is unreliable for estimating the convergence pattern to the Pareto-optimal solutions.

3.2 Scalarization Functions

Several methods are used to solve multi-objective optimization problems. There are *a priori, a posteriori, no preference, interactive* and *hybrid* methods. Scalarizing falls under the *a priori method* where a multi-objective problem is formulated as single-objective optimization problem whose solutions are Pareto-optimal solutions to the multi-objective optimization problem. Scalarization functions were introduced by Wierzbicki [64, 65, 66, 67, 68, 69, 70].

The weighted Tchebycheff metric shown in equation (3.8) minimizes the distance between an ideal objective vector and a feasible objective vector, but has few limitations.First,an unknown global ideal vector will result in failure of producing (weakly) Pareto-optimal solutions. Second,if a reference point used in place of the ideal vector lies inside the feasible objective region, the minimal distance between them might be zero and we will not be able to use it in generating (weakly) Pareto-optimal solutions.

$$\begin{aligned} &\text{minimize} \quad max \sum_{i=1}^{M} w_i |(f_i(x) - z_i)| \\ &\text{subject to} \quad x \in S \end{aligned} \tag{3.8}$$

In general one can obtain weakly Pareto-optimal solutions by solving above problem, by not including the absolute signs. Getting rid of the absolute sign helps to generate weakly Pareto-optimal solutions that are independent of the feasibility or infeasibility of the reference point in use. An achievement scalarizing function (ASF) is a function $s_{\bar{\mathbf{z}}} : Z \rightarrow \mathbf{R}$ where $\bar{\mathbf{z}} \in \mathbf{R}^k$, is an arbitrary reference point in the M dimensional objective space, which is fixed to the utopian point $z_i = z_i^{ideal} - \epsilon_i$, and ϵ_i can be set as 0.001 or even 0 in the formulation. The idea of the ASF is to minimize the distance between the ideal objective vector and the feasible objective region. For a

reference point **z**, and a weight vector **w**, the ASF problem is given as

$$\begin{aligned} &\text{minimize} \quad ASF(x, z, w) = max \sum_{i=1}^{M} (f_i(x) - z_i)/w_i \\ &\text{subject to} \quad g_j(x) \leq 0 \quad \forall_i \end{aligned} \tag{3.9}$$

Here, $w \in R^M$ is an M dimensional weight vector, such that $w_i \geq 0$ and $\|w\| = 1$ and the weight value for the i-th objective function is

$$w_i = \frac{(f_i(x) - z_i)}{\sqrt{\sum_{i=1}^{M} (f_k(x) - z_k)^2}} \tag{3.10}$$

The biggest advantage of ASF over other weighted-metric methods is that it is independent of the global ideal objective vector, and hence is more reliable. It has been proved in [19] that with above conditions for **w** and **z**, solution to the equation (3.9) is always Pareto-optimal. By keeping the reference point **z** fixed and by changing the weight vector **w**, different points on the efficient front can be generated by the ASF procedure given by equation (3.1).

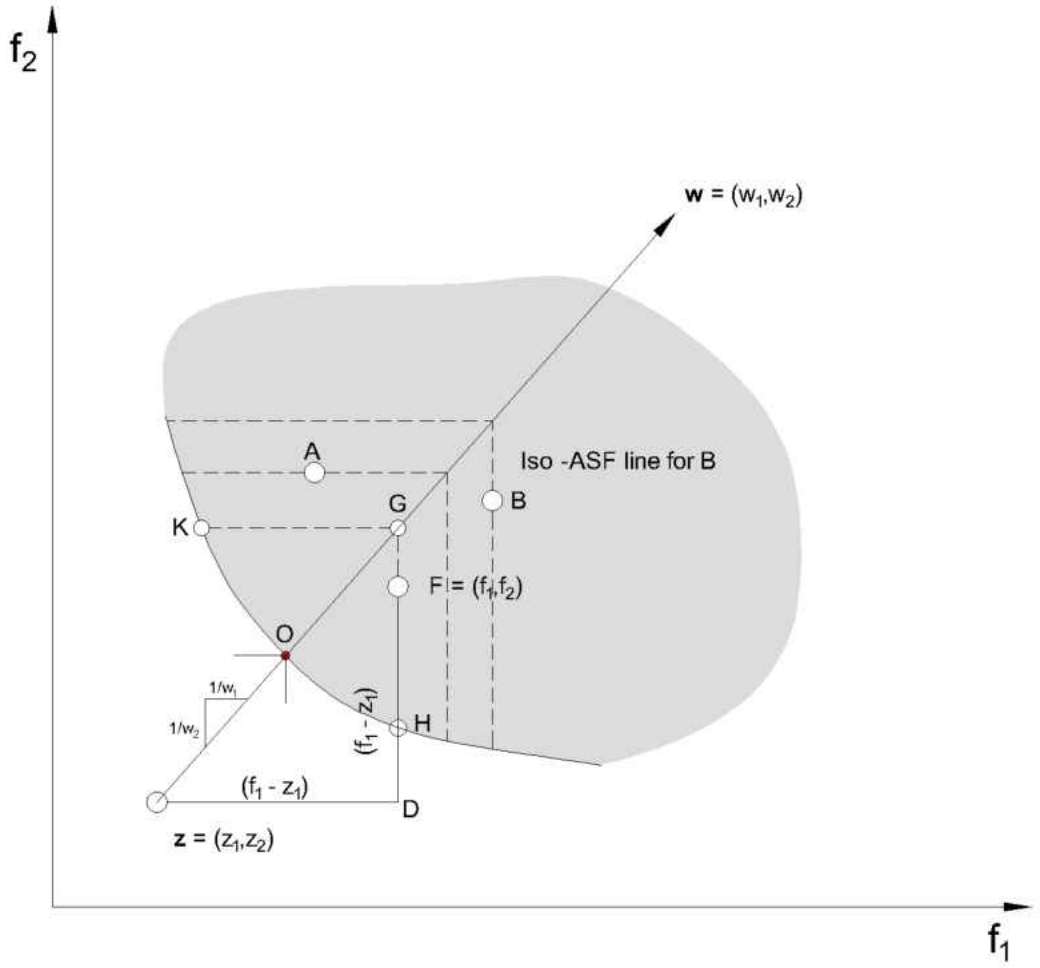

Figure 3.1: ASF Procedure

3.2.1 Notes on the ASF

- For a feasible reference point, the minimization of ASF must produce a solution that maximizes the distance to the Pareto-optimal set.
- For an infeasible reference point, the minimization of ASF must produce a solution that minimizes the distance to the Pareto-optimal set.
- By moving the reference point only, one can obtain any arbitrary weakly Pareto-optimal or Pareto optimal solution.

3.3 Karush Kuhn Tucker Proximity Measure for Multi-Objective Optimization Problem

The ASF formulation makes the objective function non-differentiable, a smooth transition of the ASF problem is required, which is made by introducing a slack variable x_{n+1} and reformulating the original ASF problem as follows:

$$\begin{aligned} \text{minimize} \quad & F(x, x_{n+1}) = x_{n+1} \\ \text{subject to} \quad & \frac{f_i(x) - z_i}{w_i^k} - x_{n+1} \leq 0 \quad i = 1, 2 \ldots M, \\ & g_j(x) \leq 0 \quad j = 1, 2 \ldots J \end{aligned} \tag{3.11}$$

Considering only the feasible solutions, the KKT proximity measure for above smooth single objective problem for $\mathbf{y} = (\mathbf{x}; x_{n+1})$ can be determined by solving the following problem [45]

$$\underset{\epsilon_k,\, x_{n+1},\, u}{\text{minimize}} \quad \epsilon_k + \sum_{j=1}^{J} (u_M + g_j(x^k))^2 \tag{3.12a}$$

$$\text{subject to} \quad \|\nabla F(y) + \sum_{j=1}^{M+j} u_j \nabla G_j(\mathbf{y})\|^2 \leq \epsilon_k,, \tag{3.12b}$$

$$\sum_{j=1}^{M+j} u_j \nabla G_j(\mathbf{y}) \geq -\epsilon_k, \tag{3.12c}$$

$$\frac{f_j(x) - z_j}{w_j^k} - x_{n+1} \leq 0 \quad j = 1 \ldots M, \tag{3.12d}$$

$$u_j \geq 0 \quad j = 1 \ldots (M + J) \tag{3.12e}$$

The optimal value ϵ_k^* to the above problem corresponds to the proposed KKT proximity measure. For infeasible iterates, the KKT proximity measure is computed by measuring the constraint violations.

KKT proximity measure for any iterate x^k is calculated as

$$KKT\ \ proximity\ \ measure = \epsilon_k^*\ \ if\ \ x^k\ \ is\ \ feasible$$
$$= 1 + \sum_{j=1}^{J} < g_j(x^k) >^2 \quad otherwise$$

and,

$$\epsilon_k^k = 1 - \sum_{j=1}^{M} u_j^* - \sum_{j=1}^{J} (u_M^* + g_j(x^k)^2) \tag{3.13}$$

For a KKT point, $x^k = x^*$, the complimentary slackness condition $u_M^* + g_j(x^*) = 0$ for all constraints, and $\epsilon_k \geq (1 - \sum_{j=1}^{M} u_j)^2$

THe KKT proximity measure has been shown to converge monotonically towards the Pareto-optimal front [45].

CHAPTER 4

APPLYING THE KKT PROXIMITY MEASURE TO A MULTI-OBJECTIVE PARTICLE SWARM OPTIMIZER

In this chapter, a parametric investigation is performed in order to access the capabilities of the ACMOPSO algorithm using traditional quality indicators but also the KKT proximity measure on a number of benchmark problems. Several test suites are available to test complex multi-objective optimization algorithms. A detailed review of multi-objective test problems can be found in [71]. Seven unique benchmark problems are selected to test the performance of ACMOPSO. These problems will also be utilized in the following chapter to evaluate the performance of the proposed MOPSO algorithm.

4.1 Definitions

- Pareto-Optimal Set and Pareto-Optimal Front

  As discussed in Chapter 2, a multi-objective optimization problem has a set of solutions, which is called a Pareto-optimal front. A *Pareto-optimal set* (PS) is the collection of all Pareto optimal decision variables and a *Pareto Front* (PF) is the collection of all Pareto optimal objective vectors. PS gives the mapping in search space, while PF maps the objective space. Pareto-optimality criteria defined earlier is used to obtain a *nondominated* PS and PF.

- Fitness landscape

  It is the mapping between the search space and the objective space, i.e. between PS and PF. They might share a One-to-One mapping or Many-to-One mapping. In One-to-One mapping each decision variable vector corresponds to one objective vector, while in Many-to-One

Table 4.1: Test Problem Properties.

Name	Objective	Modality	Dissimilar parameter domain	Dissimilar trade-off range	Optima known	Geometry	Many-to-one	Flat regions
ZDT2	f1	U	x	x	✓	concave	−	−
	f2	U						
ZDT6	f1	M	x	✓	✓	concave	+	−
	f2	M						
OSY	f1	M	✓	✓	✓	convex	+	+
	f2	M	✓	✓	✓	convex	+	+
TNK	f1	M	x	x	✓	concave	+	−
	f2	M	x	x	✓	concave	+	−
mTNK	f1	M	x	x	✓	concave	+	−
	f2	M	x	x	✓	concave	+	−
WBD	f1	M	✓	✓	✓	convex	+	+
	f2	M	✓	✓	✓	convex	+	+
Viennet	f1	M	x	✓	✓	concave	+	−
	f2	M	x	✓	✓	concave	+	−
	f3	M	x	✓	✓	concave	+	−

more than one decision variable vectors correspond to identical objective vector. In the later case, difficulty arises while selecting between these two decision variable vectors using non-dominance criteria.

- Modality

  A fitness landscape can be *unimodal* or *multimodal*. An objective function with single optima has a unimodal landscape, while the one with multiple optima has a multimodal landscape. Unimodal problems are easy to solve as there is no danger for the algorithm to get stuck in any local optima, which is not the case for multimodal problems.

- Dissimilar parameter domain

  An algorithm is forced to adjust its parameters for problems with design variables that lie in different magnitude of domain. For example x_a may lie in [0,1] , while x_b might lie in [10,100].

- Dissimilar trade-off range Similar to design variables, objective functions may have a different range too. For example in one objective may define quality, while the other might define price which have different scales of measurement. Both, dissimilar parameter domain and trade-off range force an algorithm to adjust its parameters and find well spread solutions that cover the entire search domain.

- Pareto Optimal Geometry

  For MOOPs, the PF can have a wide variety of geometry. It can be *convex, concave, linear, degenerate, connected, and disconnected*. By definition if the convex hull is covered by the set,it is *convex*, and if the convex hull covers the set then it is *concave*. A problem having both sets, convex and concave, is linear. A lower dimension PF in a higher dimension objective vector space is a *degenerate* front. *Disconnected* and *connected* front refers to discontinuous set and continuous set. It should be noted that a discontinuous set might not always map a disconnected front.

4.2 Multi-Objective Optimization Test Problems

We select seven problems from the multi-objective optimization literature for investigating the performance of ACMOPSO. All of the selected problems have objective functions and constraints

that are differentiable. It has to be mentioned that the KKT optimality analysis requires the gradients of objective functions and constraints.

4.2.1 Test problem Zitler-Deb-Thiele - 2 (ZDT2)

ZDT2 is a high dimensional (30 design variable), bi-objective unconstrained problem having a non-convex Pareto-optimal front. Below is the problem formulation.

$$\text{Minimize} \begin{cases} f_1(x) = x_1 \\ g(x) = 1 + \frac{9}{n-1} \sum_{i=2}^{n} x_i \\ f_2(x, y) = 1 - (f_1(x)/g(x))^2 \end{cases} \tag{4.1}$$

4.2.2 Test problem Zitler-Deb-Thiele - 6 (ZDT6)

ZDT6 is a 10-variable, unconstrained problem having a non-convex Pareto-optimal set. It has a non-uniformly distributed PS with sparse density of solutions near the optimal front, which makes it a difficult problem to solve.

$$\text{Minimize} \begin{cases} f_1(x) = 1 - \exp(-4x_1)\sin^6(6\pi x_1) \\ g(x) = 1 + 9((\sum_{i=2}^{n} x_i)/9)^{0.25} \\ f_2(x) = 1 - (f_1(x)/g(x))^2 \end{cases} \tag{4.2}$$

4.2.3 Test problem Osyczka and Kundu (OSY)

The Osyczka and Kundu (OSY) problem is a constrained six variable test problem. The true Pareto front of this problem consists of five disjoint Pareto-optimal regions, with atleast one active constraint per region. The optimal solutions lie on the intersection of constraint boundaries and hence it is important that an algorithm properly explores and exploits these regions.

$$
\begin{aligned}
\underset{x}{\text{minimize}} \quad & f_1(x) = -[25(x_1-2)^2 + (x_2-2)^2 + (x_3-1)^2 + (x_4-4)^2 + (x_5-1)^2] \\
\underset{x}{\text{minimize}} \quad & f_2(x) = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 + x_6^2 \\
\text{subject to} \quad & g_1(x) = x_1 + x_2 - 2 \geq 0 \\
& g_2(x) = 6 - x_1 - x_2 \geq 0 \\
& g_3(x) = 2 - x_2 + x_1 \geq 0 \\
& g_4(x) = 2 - x_1 + 3x_2 \geq 0 \\
& g_5(x) = 4 - (x_3-3)^2 - x_4 \geq 0 \\
& g_6(x) = (x_5-3)^2 + x_6 - 4 \geq 0 \\
& 0 \leq x_1, x_2, x_6 \geq 10, 1 \leq x_3, x_5 \geq 5, 0 \leq x_4 \geq 6,
\end{aligned}
\tag{4.3}
$$

4.2.4 Test problem Tanaka (TNK)

The original TNK problem is a two variable constrained test problem. It has a globally Pareto-optimal front with three disconnected fronts alternating between feasible and infeasible regions of the search space.

$$
\begin{aligned}
\underset{x}{\text{minimize}} \quad & f_1(x) = x_1 \\
\underset{x}{\text{minimize}} \quad & f_2(x) = x_2 \\
\text{subject to} \quad & g_1(x) = x_1^2 + x_2^2 - 1 - 0.1\cos\ (16\arctan\frac{x_1}{x_2}) \geq 0 \\
& g_2(x) = (x_1-0.5)^2 + (x_2-0.5)^2 \leq 0.5 \\
& 0 \leq x_1 \geq \pi \\
& 0 \leq x_2 \geq \pi
\end{aligned}
\tag{4.4}
$$

4.2.5 Test problem modified Tanaka (mTNK)

The modified TNK is similar to the original TNK problem with a slight change in one of the constraints. It has a globally Pareto-optimal front with eight disconnected smaller fronts.

$$\begin{aligned}
\underset{x}{\text{minimize}} \quad & f_1(x) = x_1 \\
\underset{x}{\text{minimize}} \quad & f_2(x) = x_2 \\
\text{subject to} \quad & g_1(x) = x_1^2 + x_2^2 - 1 - 0.1\cos\ (32\arctan\frac{x_1}{x_2}) \geq 0 \\
& g_2(x) = (x_1 - 0.5)^2 + (x_2 - 0.5)^2 \leq 0.5 \\
& 0 \leq x_1 \geq \pi \\
& 0 \leq x_2 \geq \pi
\end{aligned} \tag{4.5}$$

4.2.6 Test problem welded beam design (WBD)

The welded beam design problem has four variables and four nonlinear constraints. The objective function and the constraints are differentiable. The first objective function corresponds to the cost of the weld assembly and the second objective function corresponds to the deflection of the free end of the beam.

$$
\begin{aligned}
\underset{x}{\text{minimize}} \quad & f_1(x) = 1.10471 x_1^2 x_2 + 0.04811 x_3 x_4 (14 + x_2) \\
\underset{x}{\text{minimize}} \quad & f_2(x) = \frac{4PL^3}{E x_4 x_3^3} \\
\text{subject to} \quad & g_1(x) = x_1 - x_4 \leq 0 \\
& g_2(x) = \frac{4PL^3}{E x_4 x_3^3} - 0.25 \leq 0 \\
& g_3(x) = \sqrt{t_p^2 + \frac{t_p t_{dp} x_2}{R} + t_{dp}^2} - \tau_{max} \leq 0 \\
& g_4(x) = \frac{6PL}{(x_4 x_3^2)} - \sigma_{max} \leq 0 \\
& g_5(x) = P - P_c \leq 0 \\
& t_p = \frac{P}{(\sqrt{2} x_1 x_2)} \\
& R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2} \\
& M = P\left(L + \frac{x_2}{2}\right) \\
& J = 2\left(\left(\frac{x_1 x_2}{\sqrt{2}}\right) * \left(\frac{x_2^2}{12} + \left(\frac{x_1 + x_3}{2}\right)^2\right)\right) \\
& t_{dp} = \frac{MR}{J} \\
& P_c = 4.013\sqrt{EG}\left(\frac{x_3 x_4^3/6}{L^2}\right)\left(1 - \frac{x_3\sqrt{\frac{E}{4G}}}{2L}\right) \\
& P = 6,000 \ \ lbf \\
& L = 14 \ \ inches \\
& E = 30,000,000 \ \ psi \\
& \tau_{max} = 13,600 \ \ psi \\
& \sigma_{max} = 30,000 \ \ psi \\
& G = 12,000,000 \ \ psi
\end{aligned}
\tag{4.6}
$$

4.2.7 Test problem Viennet (VNT)

$$\begin{aligned}
\underset{x}{\text{minimize}} \quad & f_1(x) = 0.5(x_1^2 + x_2^2) + \sin(x_1^2 + x_2^2) \\
\underset{x}{\text{minimize}} \quad & f_2(x) = 15 + \frac{(3x_1 - 2x_2 + 4)^2}{8} + \frac{(x_1 - x_2 + 1)^2}{27} \\
\underset{x}{\text{minimize}} \quad & f_3(x) = \frac{1}{x_1^2 + x_2^2 + 1} - 1.1\exp(-x_1^2 + x_2^2) \\
\text{subject to} \quad & -3 \leq x_1, x_2 \geq 3
\end{aligned} \tag{4.7}$$

4.3 Performance Evaluation of ACMOPSO

The ACMOPSO algorithm developed in [36] is used as the baseline algorithm for this preliminary investigation. The algorithm was coded in Intel Fortran 2017, and its performance is evaluated using selected benchmark problems for multi-objective optimization. ACMOPSO maintains an external archive to store the nondominated solutions found in all iterations. A swarm particle keeps track of the optimal position that has found thus far and sends its new position to the archive for evaluation if and only if it is not dominated by its personal best position. Leaders are selected among the archived solutions based on the cone separation method. *Inertia weight* is adjusted by adapting the *social* component of the velocity update equation. A mutation operator is also employed in the algorithm as recommended in [33]. The KKT proximity measure as defined in (3.12a) is calculated for the nondominated solutions found by ACMOPSO. The quality of the obtained solutions is evaluated using the Generational Distance (GD) [72] and the Inverse Generational Distance (IGD). The KKT proximity measure is computed at each iteration during the experiments. The Generational Distance quantifies the average distance of the computed solutions from the true Pareto-optimal front, while the Inverted Generational Distance measures how well the Pareto-optimal front is represented by the computed nondominated set of solutions. Similar to GD, the KKT proximity measure quantifies the proximity of the obtained solutions from the true Pareto-optimal front [45]. The GD and IGD indicators are calculated as follows:

$$\text{GD} = \frac{\sqrt{\sum_{i=1}^{I} d_i^2}}{|I|} \tag{4.8}$$

$$\text{IGD} = \frac{\sqrt{\sum_{j=1}^{J} d_j^2}}{|J|} \tag{4.9}$$

where $|I|$ is the cardinality of the computed Pareto-optimal set, $|J|$ is the cardinality of the globally Pareto-optimal set of solutions provided that a set of discrete Pareto-optimal solutions is available, d_i is the Euclidean distance between the computed Pareto-optimal solution i and its nearest available globally Pareto-optimal solution, and d_j is the Euclidean distance between a globally Pareto-optimal solution j and its nearest computed Pareto-optimal solution.

Table 4.2: Simulation Parameters.

Test problem	MOO iterations	SO iterations	Processor threads
ZDT2	50	5000	1
ZDT6	50	5000	8
OSY	300	5000	4
TNK	100	3000	8
mTNK	100	3000	8
WBD	300	5000	8
Viennet	100	3000	8

A swarm with 100 particles which is divided into five sub-swarms is utilized. These sub-swarms explore the search space which is partitioned into equal segments using the information about the ideal and the nadir points. Objective functions, constraints and their gradients are normalized and used in the KKT proximity calculation. The procedure is repeated 31 times with randomly generated initial swarm positions the mean, median and standard deviation of all of the quality

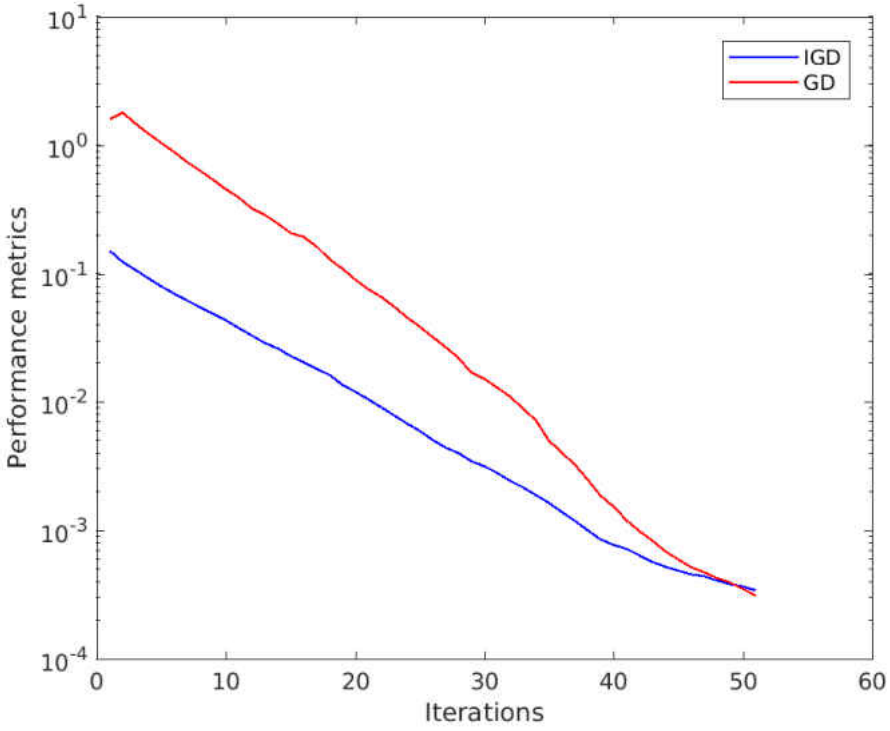

Figure 4.1: GD and IGD Values for the Problem ZDT2.

indicators mentioned above are reported. The simulation parameters are listed in Table 4.2. The MOO iterations correspond to the number of iterations performed by ACMOPSO, and in Chapter 5 by the proposed MOPSO. The SO iterations correspond to the number of iterations performed by the single-objective DE/rand/1/either-or algorithm (see Section 2.2.3) in order to calculate the KKT proximity measure for each nondominated solution vector generated by ACMOPSO.

4.3.1 Problem ZDT2

Figure 4.1 shows the convergence of the median value of GD and IGD for problem ZDT2 across 31 runs and Table 4.3 shows the mean, median and standard deviation values of the KKT proximity measure after 50 iterations across 31 runs. ACMOPSO is able to find a good distribution of solutions near the true Pareto-optimal front within approximately 45 iterations as demonstrated by the convergence of GD and IGD. Figure 4.2 shows the true Pareto-optimal front as a solid line

Table 4.3: ZDT2 KKT Proximity Measure.

Metric	Mean	Median	Std. Dev
KKTPM	3.09E-04	4.69E-006	6.36E-004
GD	3.11E-04	2.18E-004	2.88E-004
IGD	3.42E-04	2.80E-004	1.51E-004

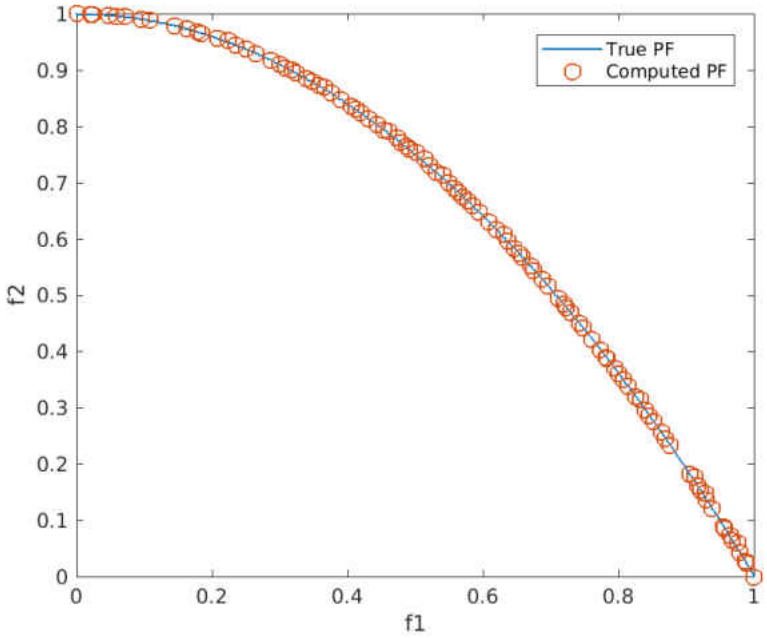

Figure 4.2: Pareto Front for the Problem ZDT2.

and the computed solutions represented as circles for the run that corresponds to the median value of the KKT proximity measure. The observed values of the latter validate the fact that the obtained nondominated solutions are Pareto optimal.

4.3.2 Problem ZDT6

Problem ZDT6 is a 10-variable problem with a nonconvex Pareto-optimal set. The characteristic feature of this problem is its non-uniform density of solutions across the Pareto-optimal region, which cause convergence issues for many multi-objective optimization algorithms. Figures 4.3

and 4.4 show that ACMOPSO requires on average approximately 30 iterations to converge to the Pareto-optimal front based on the IGD value. The presence of one nondominated solution near the upper left-hand corner of the Pareto-optimal front, as shown in Figure 4.5 at the final iteration demonstrates the inability of the algorithm to focus the search on that region and bring the non-dominated solutions closer to the Pareto-optimal front. That conclusion is corroborated by the high GD value shown in Figure 4.3 and by the worst KKT proximity measure value being approximately equal to one.

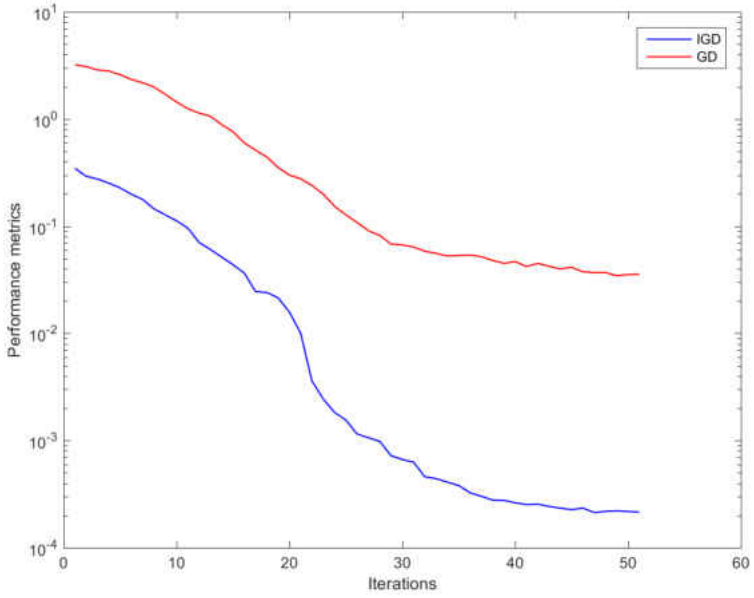

Figure 4.3: GD and IGD Values for the Problem ZDT6.

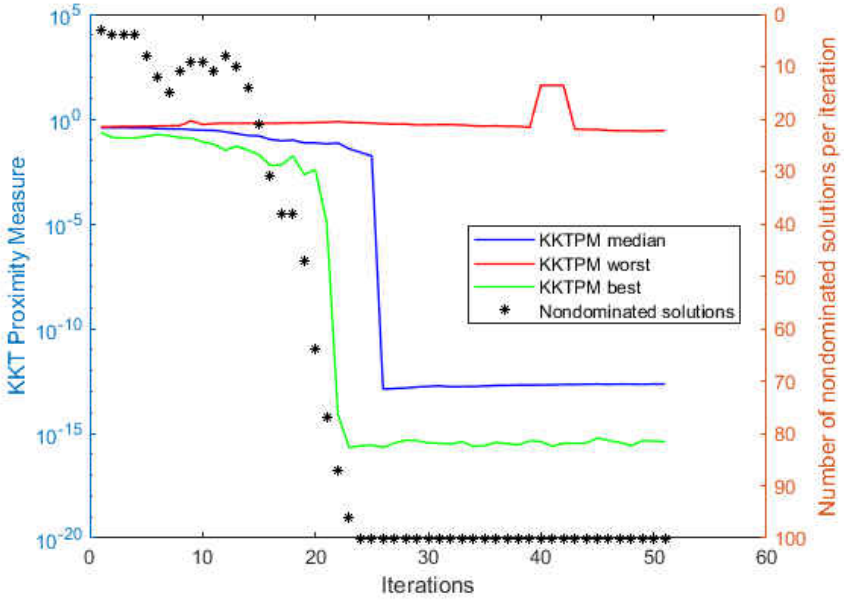

Figure 4.4: KKT Proximity Measure for the Problem ZDT6.

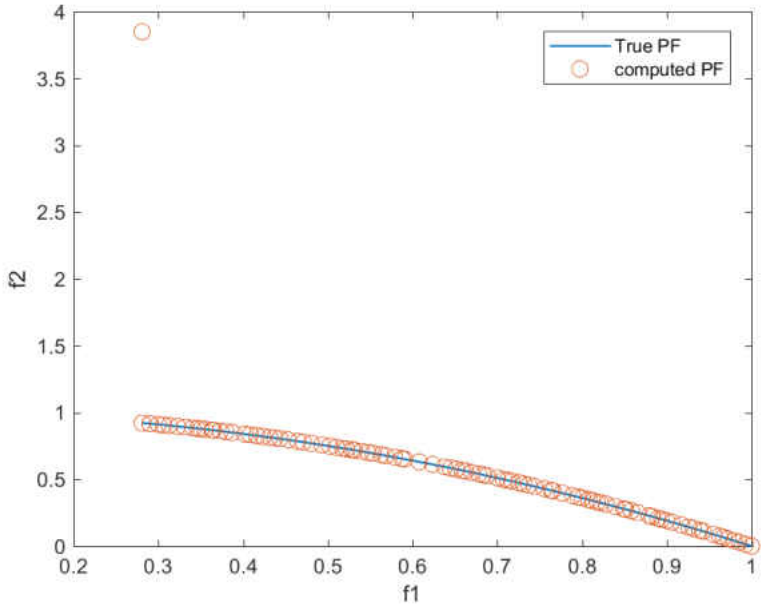

Figure 4.5: Pareto Front for the Problem ZDT6.

4.3.3 Problem OSY

It can be observed from Figure 4.8 that the first objective function does not vary significantly in the regions AB, BC, and EF, which makes OSY a difficult problem to solve. Figure 4.6 shows the quality indicators and Figure 4.7 shows the KKT proximity measure for the corresponding Pareto-optimal front. At about 200 iterations with a swarm size of 100, ACMOPSO is able to reach the vicinity of the true Pareto-optimal front. The GD, IGD, and the KKT proximity measure corroborate this claim. It should be mentioned that the median value of the KKT proximity measure decreases from 0.9885 to 0.1921 across the iterations for 31 runs. High KKT proximity measure values at certain points indicate that the solution has not converged to the true Pareto-optimal front. This is a scenario where it is demonstrated that the KKT proximity measure offers a means to detect inability to converge to the true Pareto-optimal front. [63].

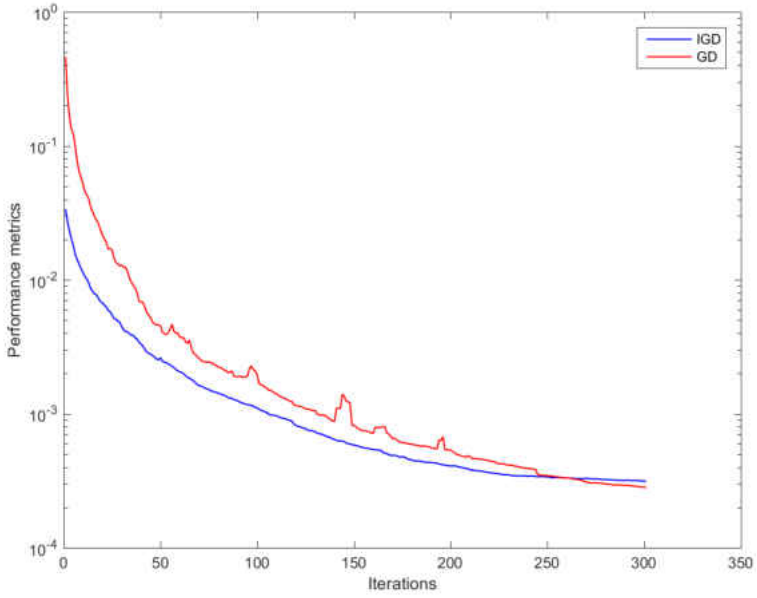

Figure 4.6: GD and IGD Values for the Problem OSY.

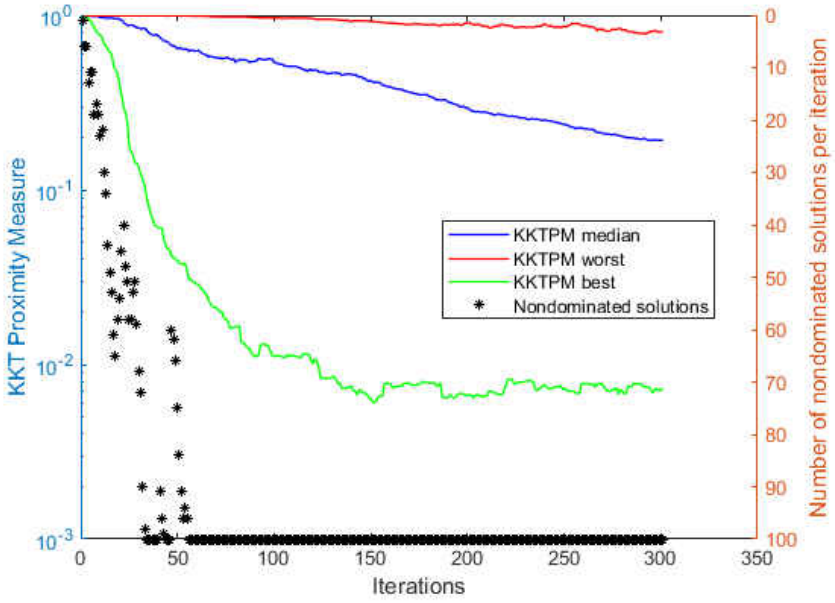

Figure 4.7: KKT Proximity Measure for the Problem OSY.

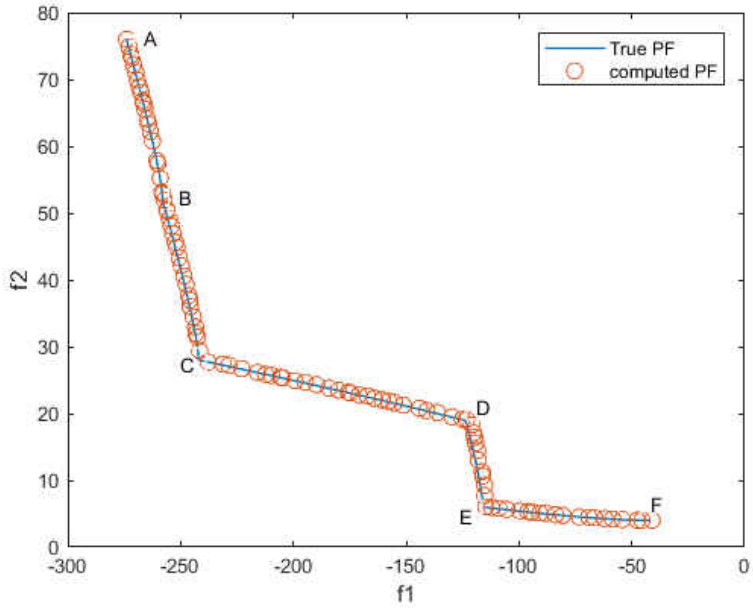

Figure 4.8: Pareto Front for the Problem OSY.

4.3.4 Problem TNK

Figure 4.11 shows the disconnected Pareto-optimal front for the original TNK problem. An optimization algorithm has difficulty exploring the search space since all of the solutions lie on a nonlinear constraint surface. Figures 4.9 and 4.10 show a steady convergence of ACMOPSO to the true Pareto-optimal front. ACMOPSO is able to find a well-distributed set of solutions in about 50 iterations when the values of GD and IGD drop below 10^{-3}.

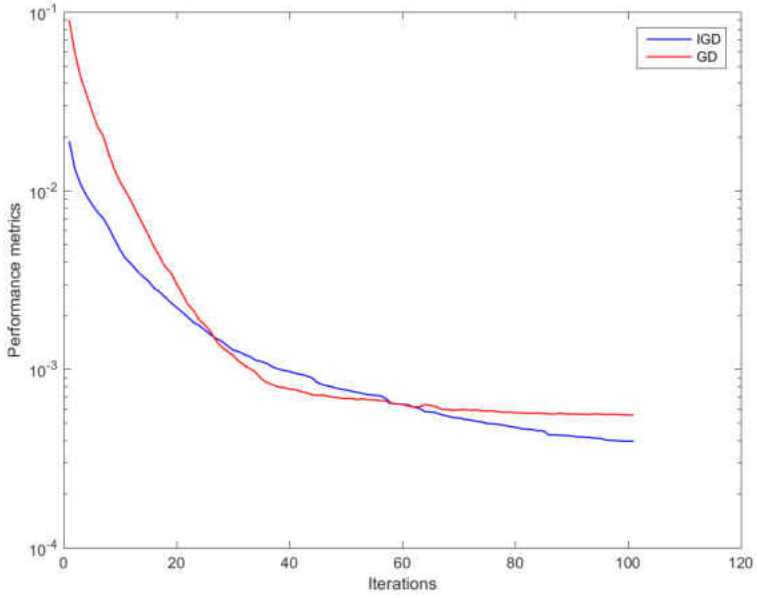

Figure 4.9: GD and IGD Values for the Problem TNK.

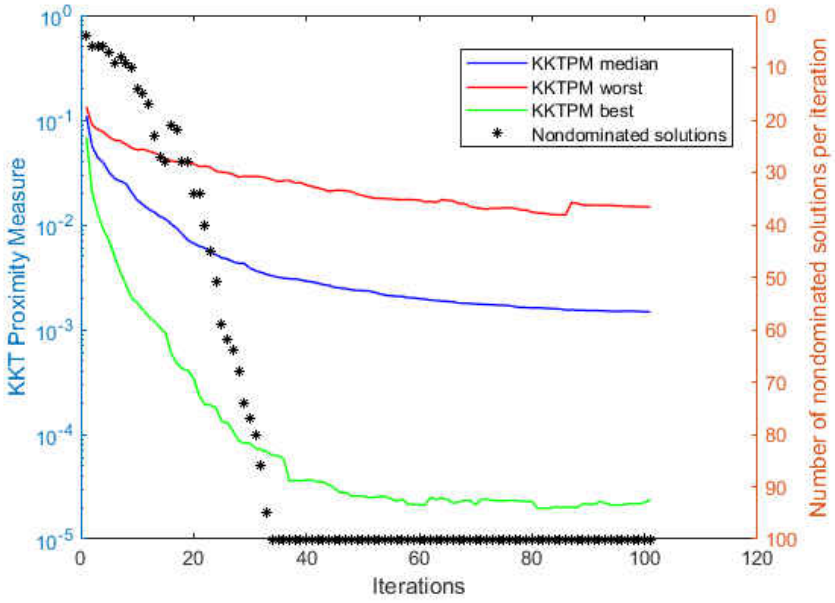

Figure 4.10: KKT Proximity Measure for the Problem TNK.

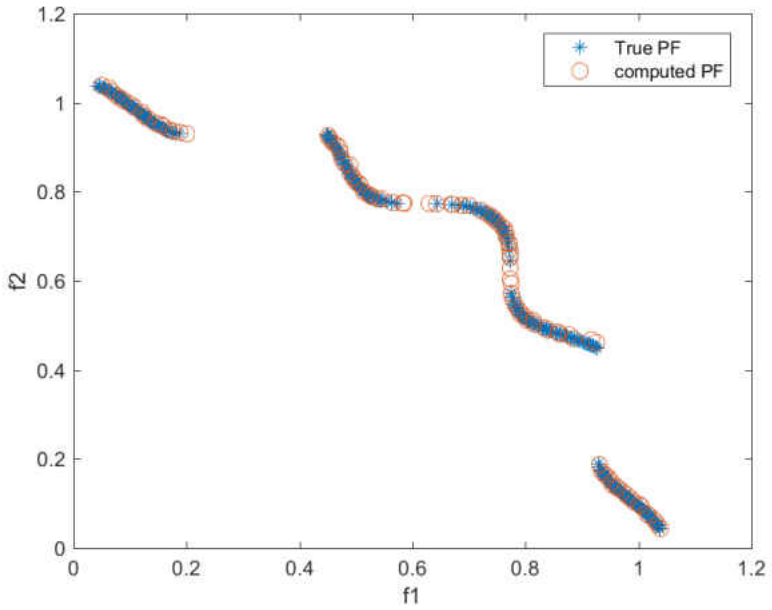

Figure 4.11: Pareto Front for the Problem TNK.

4.3.5 Problem mTNK

The difference between the original and the modified version is just the value of a coefficient in the first constraint, which results in a Pareto-optimal front with eight disconnected segments. The performance of ACMOPSO on the modified TNK problem is similar to its performance on the original TNK. Figures 4.12 and 4.13 show the same steady convergence of the algorithm to the true Pareto-optimal front in about 60 iterations. The GD and IGD values remain steady with minor fluctuations and the median value KKT proximity measure converges to 10^{-3}.

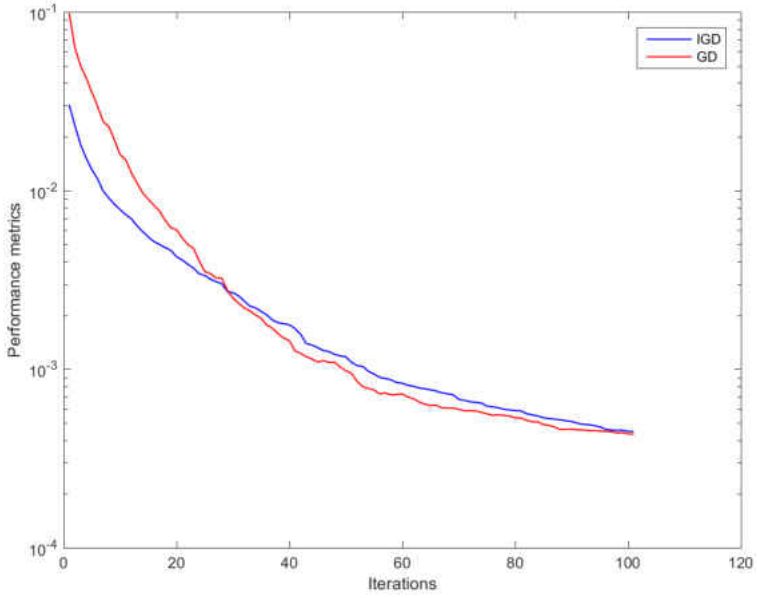

Figure 4.12: GD and IGD Values for the Problem mTNK.

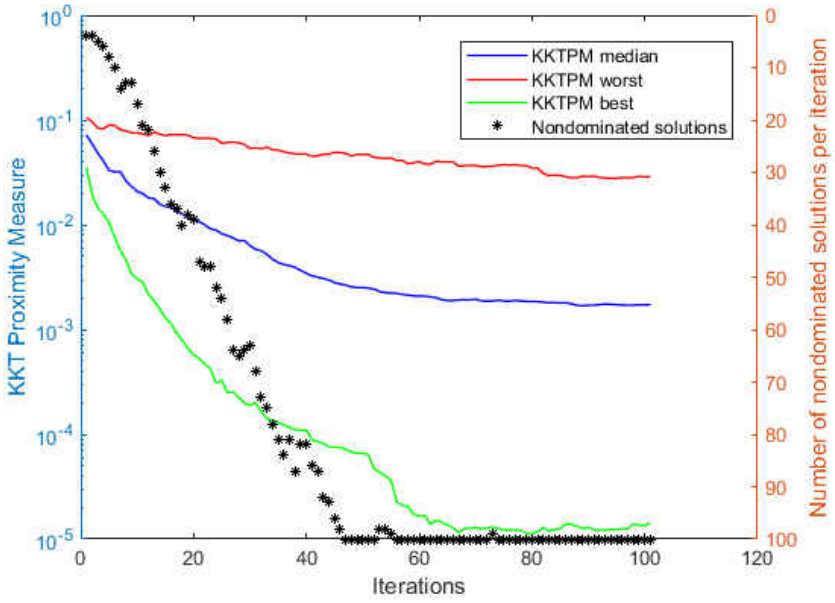

Figure 4.13: KKT Proximity Measure for the Problem mTNK.

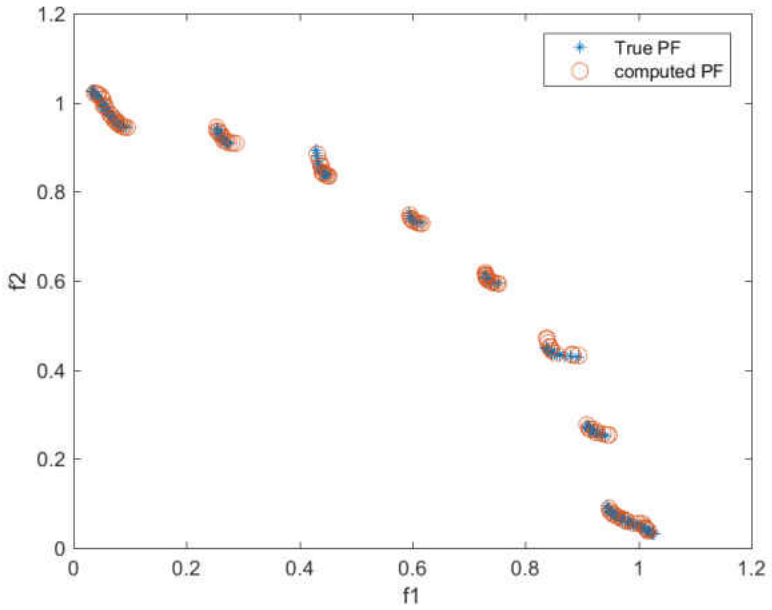

Figure 4.14: Pareto Front for the Problem mTNK.

4.3.6 Problem WBD

The main challenge in this problem is to find the Pareto-optimal front near the minimum-cost solution (first objective function) due to the number of active constraints present in that region. The Pareto-optimal front near the minimum-deflection solution (second objective function) is easily obtained as there is only one active constraint in that region [73]. GD and IGD manage to converge to the order of 10^{-2} and 10^{-3} respectively indicating that ACMOPSO is able to find well-distributed nondominated solutions which are near the global Pareto-optimal front, with the exception of the minimum-cost region, where not all solutions are found. Figures 4.15 and 4.16 show the convergence of ACMOPSO near the true Pareto-optimal front within approximately 150 iterations.

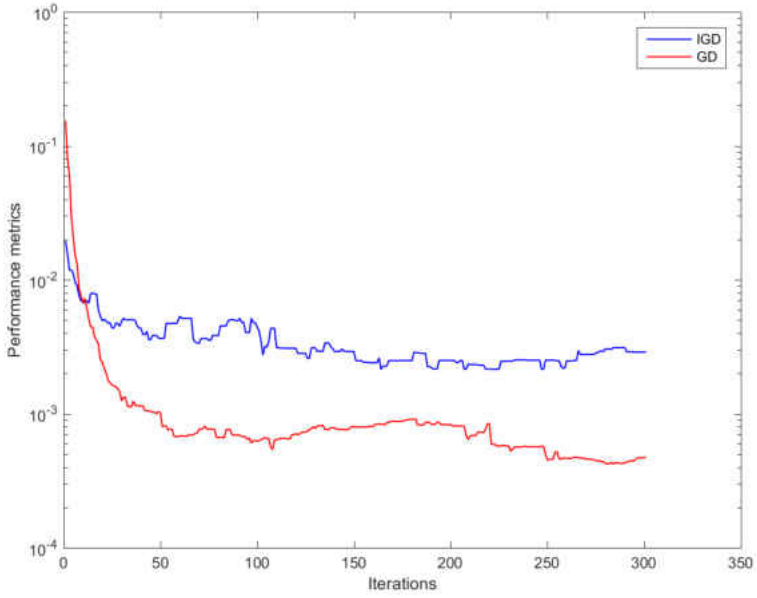

Figure 4.15: GD and IGD Values for the Problem WBD.

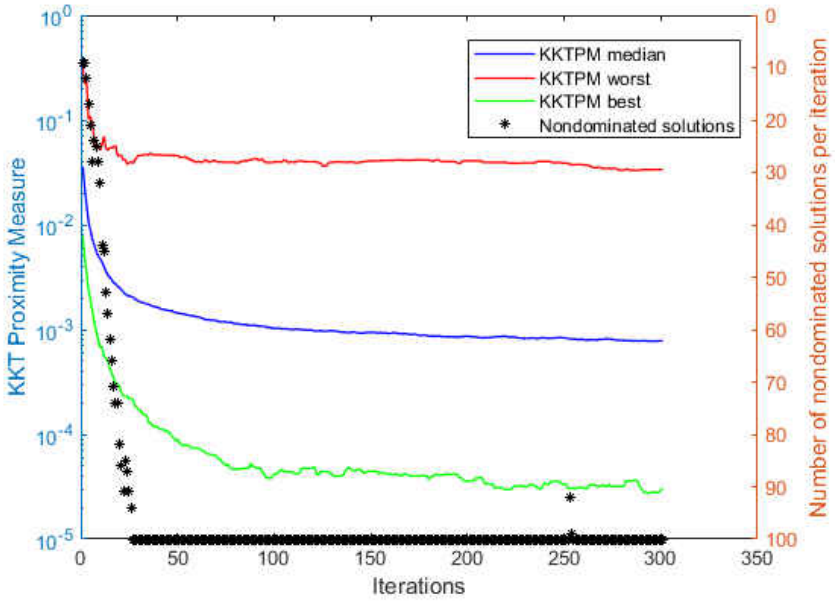

Figure 4.16: KKT Proximity Measure for the Problem WBD.

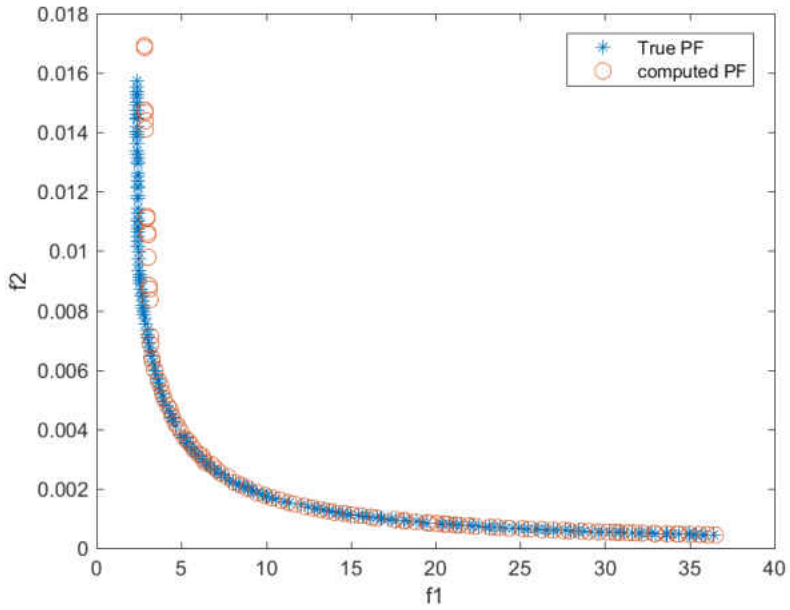

Figure 4.17: Pareto Front for the Problem WBD.

4.3.7 Problem VNT

The Viennet problem is a three-objective, constrained problem that also has a disconnected set of Pareto-optimal fronts. Figures 4.18 and 4.19 show the capability of ACMOPSO to reach the vicinity of the true Pareto-optimal front within 40 iterations, however, the lack of local search operators prevent the algorithm from obtaining a well-distributed set of solutions that is even closer to the true Pareto-optimal front. Furthermore, as shown in Figure 4.19, certain solutions remain far from the front considering that their KKT proximity measure value is approximately equal to one.

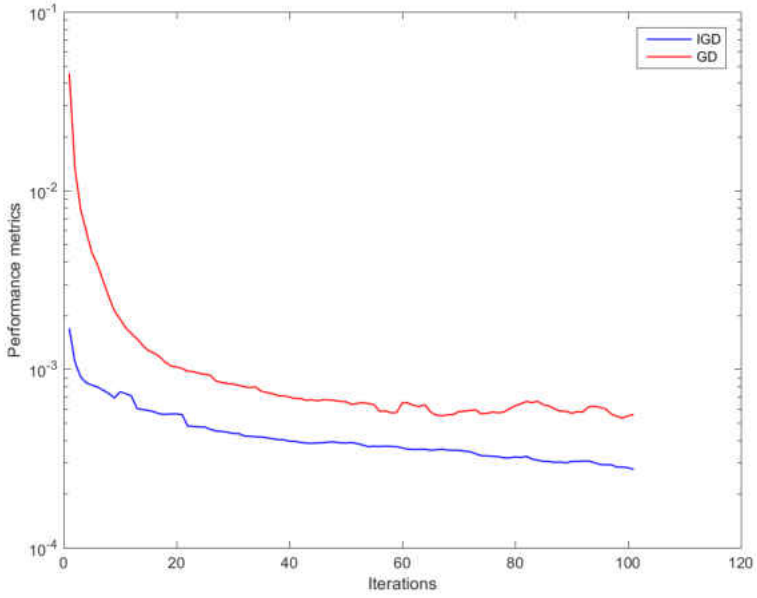

Figure 4.18: GD and IGD Values for the Problem VNT.

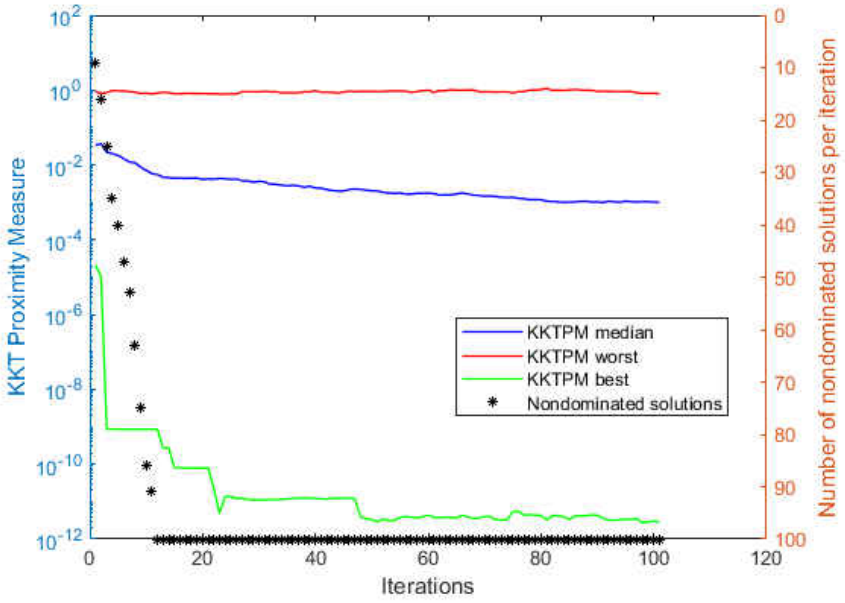

Figure 4.19: KKT Proximity Measure for the Problem VNT.

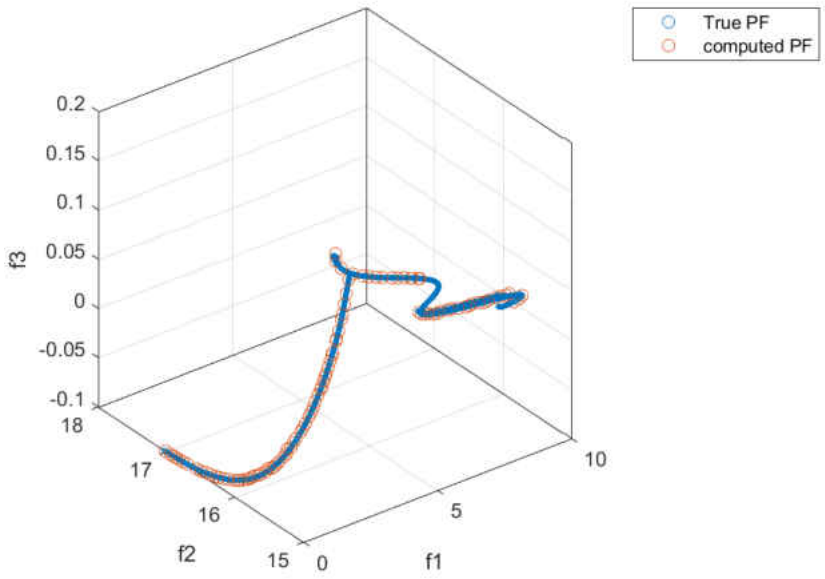

Figure 4.20: Pareto Front for the Problem VNT.

CHAPTER 5

DEVELOPMENT OF A FUZZY-ADAPTIVE MULTI-OBJECTIVE PARTICLE SWARM OPTIMIZER

The proposed optimization framework uses the swarm intelligence operators and multi-objective optimization principles utilized in the ACMOPSO algorithm, which is described in Section 2.3. The main new features consist of the calculation of the KKT proximity measure and the utilization of a set of reference vectors in order to cluster the non-dominated solutions in the external archive in a way that each sub-swarm utilizes its own set of leaders and it is, thus, able to focus on a more localized search when the particles are in the vicinity of the Pareto-optimal front. Furthermore, convergence statistics regarding the clustered non-dominated solutions are utilized in order to adapt the PSO algorithmic parameters, i.e., inertia weight w and social coefficient c_1. The adaptation is performed via two fuzzy logic controllers (FLCs). A description of the proposed Fuzzy-Adaptive Multi-Objective Particle Swarm Optimization (FAMOPSOkkt) algorithm is provided in the following sections.

5.1 Using Reference Vectors for Swarm Guidance and Archive Management

The first modification in the ACMOPSO algorithm corresponds to the utilization of a set of reference points to guide the swarms towards the Pareto-optimal front. The reference points are generated as an evenly distributed set of points on a unit hyperplane that lies in the first quadrant. This requires the normalization of the population members in every iteration, which is performed by transforming each objective function value as follows: The corresponding ideal point coordinate is subtracted from the objective function value and the outcome is divided by the difference of the maximum objective function value among the solutions in the external archive and the ideal point

coordinate,

$$f_j^n(\mathbf{x}) = \frac{f_j(\mathbf{x}) - {z_j}^i}{{z_j}^{\max} - {z_j}^i}, \quad j \in \{1, \ldots, J\} \tag{5.1}$$

where z_j is the ideal point coordinate in objective j, ${z_j}^{\max}$ is the maximum value in objective j among the solutions in the archive, and f_j^n is the normalized objective function value. The ideal point is obtained by utilizing the single-objective differential evolution DE/rand/1/either-or optimizer (see Section 2.2.3).

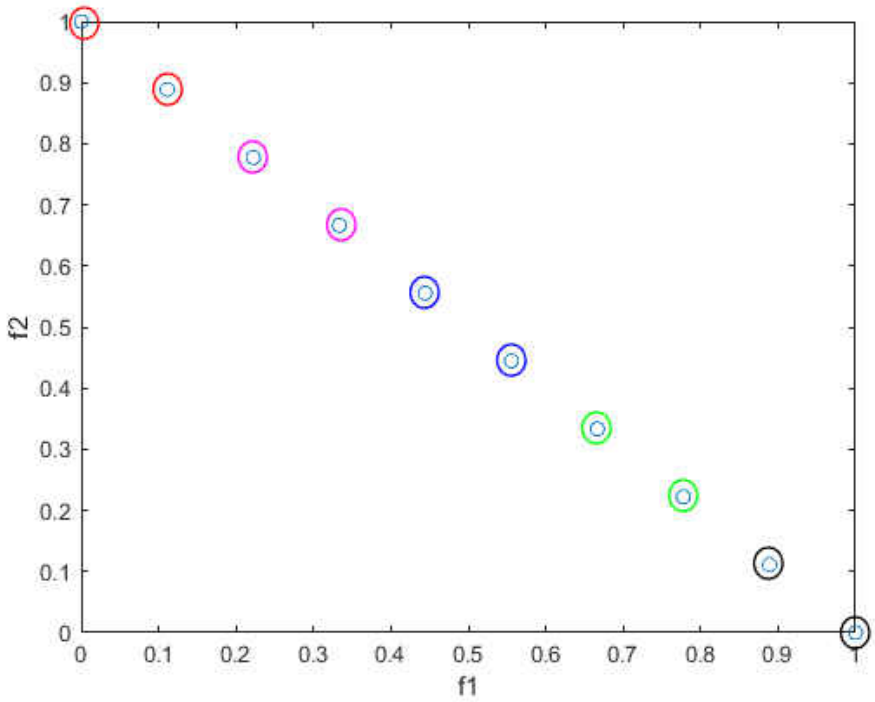

Figure 5.1: Reference Set in 2-D.

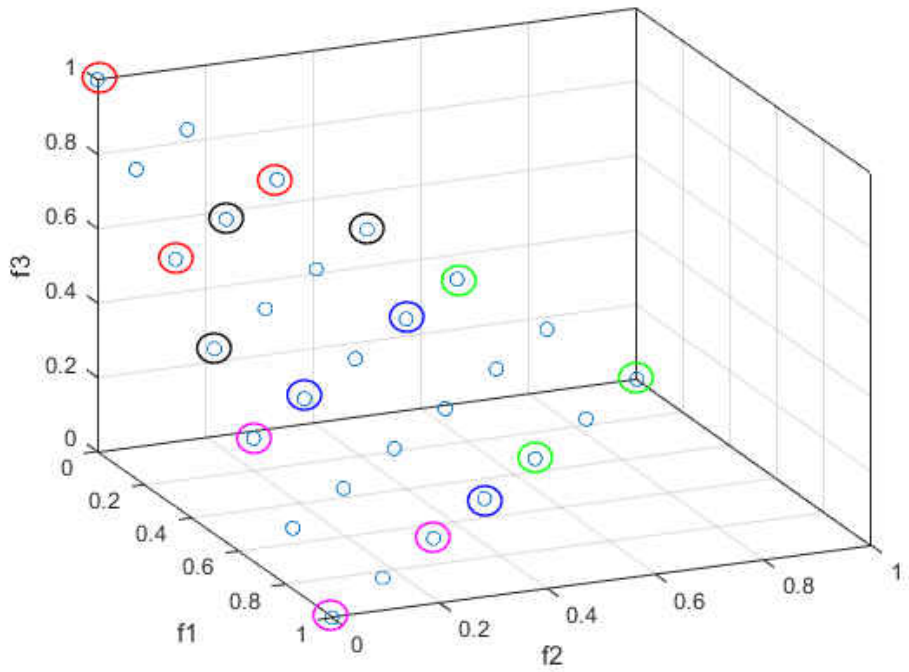

Figure 5.2: Reference Set in 3-D.

For a two-objective problem, the reference points lie on the line that connects points (0, 1) and (1, 0). For a problem with three objectives, the hyperplane corresponds to a triangle with apex at (1, 0, 0), (0, 1, 0), and (0, 0, 1). For the purposes of this work, a swarm with five equisized sub-swarms is utilized, with two reference points associated with each sub-swarm in problems with two objectives, as shown in Figure 5.1, and three reference points in problems with three objectives, as shown in Figure 5.2.

The nondominated solutions in the current iteration, which are obtained after applying the Pareto dominance criteria (see Section 2.1.1) to the solutions already stored in the external archive and the solutions that were sent to the archive by the swarm, are clustered by determining their association with each of the reference points. Specifically, for each normalized nondominated solution vector $\mathbf{f}_k^n$, the cosine of the angle formed with each reference vector (see Figure 5.3) $\mathbf{r}_l$ is calculated as:

$$\cos\theta_{k,l} = \frac{\mathbf{f}_k^n \cdot \mathbf{r}_l}{|| \mathbf{f}_k^n || || \mathbf{r}_l ||} \tag{5.2}$$

In this way, the nondominated solution vector is assigned to the reference vector that forms the smallest angle with the solution vector. Once all the nondominated solutions have been assigned to a reference vector, the solution density of the corrresponding cluster is determined by first calculating the maximum angle formed by the solution vectors that belong to the particular cluster and then dividing the angle by the cardinality of the cluster. The solution density metric quantifies the crowdedness of the cluster and is utilized to assign the probability of selection of the cluster in order to provide leaders to the sub-swarm. The selection probability is calculated as the ratio of the solution density metric of the reference point over the sum of the density metric of the group of reference points assigned to the corresponding sub-swarm.

If the number of nondominated solutions exceeds the capacity of the external archive, the following approach is utilized in order to select solution vectors that occupy the least crowded areas. First, the two extreme solutions in each cluster, i.e., the solutions that form the largest angle with the reference vector are always kept in the archive. The remaining solutions in each cluster are ranked based on their average distance from solution vectors that belong to the same cluster. A threshold regarding the number of nondominated solutions is specified for each cluster; if all the available slots are not filled, they become available to the other cluster(s) associated with the same sub-swarm. In this way, the capacity of the external archive is not exceeded at the end of each iteration.

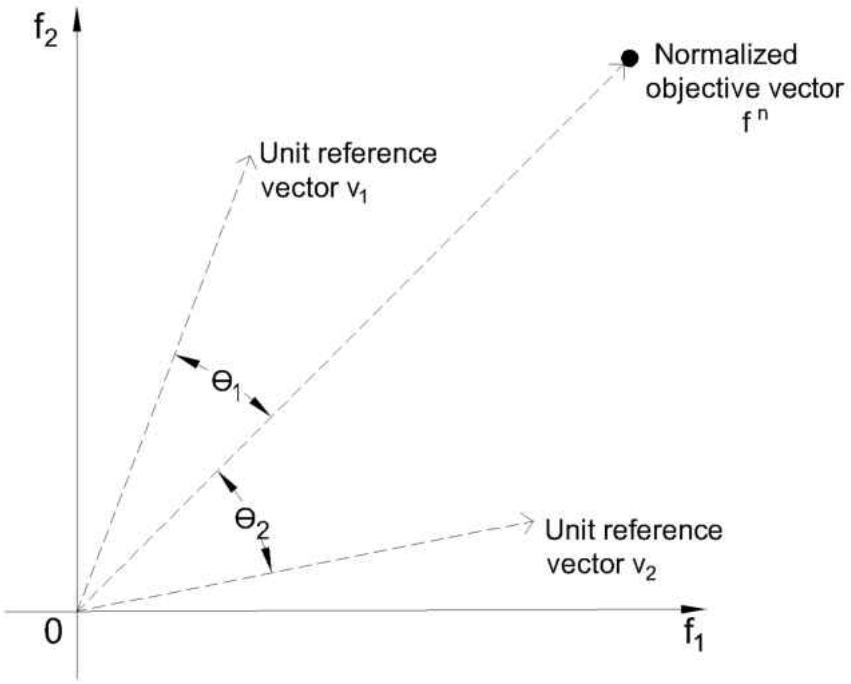

Figure 5.3: Calculating the Angle Between a Solution Vector and Each of the Reference Vectors.

5.2 Using the KKT Proximity Measure for the Selection of Swarm Leaders

The KKT proximity measure is computed for each of the nondominated solutions following the methodology described in Section 3.3. The selection of a set of leaders is performed in the following manner: If the number of leaders in each cluster does not exceed a prespecified threshold value, all the nondominated solutions in the cluster are added to the set of leaders. Otherwise, the nondominated solutions in each cluster are ordered based on their KKT proximity measure value. The solution vectors with the lowest values, up to a number of solutions equal to the threshold value, are selected and placed in the set of leaders of the sub-swarm.

The update of the velocity of each swarm particle is performed using Equation (2.2). The two leaders are obtained by first selecting the cluster of the sub-swarm using the probability assigned to each cluster (see Section 5.2) and, subsequently, by randomly choosing the leaders from the

selected cluster.

The values of the inertia weight and the social coefficient in Equation (2.2) are adapted for each reference point and corresponding cluster at the end of each iteration using feedback from the computed KKT proximity measure values. Two metrics are utilized to quantify the feedback: The first metric corresponds to the average value of the KKT proximity measure in the cluster, which reflects the average convergence of the nondominated solutions associated with that cluster. The second metric is the improvement of the average convergence relative to the previous iteration. This feedback is utilized as the input to two FLCs as described in the following section.

5.3 Fuzzy Logic Controllers

Two Mamdani-type FLCs [74] are utilized in order to adapt the inertia weight w and the social coefficient $c1$ dynamically throughout the optimization process. The adaptation is performed for each cluster of nondominated solution vectors in the external archive, i.e., each cluster has its own set of parameters, w_k and $c_{1,k}$. Two fuzzy variables are used as input to the FLCs:

- The first variable, $fz^{(1,k)}$, corresponds to the average value of the KKT proximity measure in cluster k during iteration: $fz^{(1,k)} \in [0,1]\ \forall k = 1, ..., K$
- The second variable, $fz^{(2,k)}$, is the improvement of the average convergence of the cluster solution vectors relative to the previous iteration. The computed value is normalized by the maximum improvement in all the clusters during the current iteration, thus, $fz^{(2,k)} \in [0,1]$.

The output variable, $fz^{(3,k)}$, of the first FLC is the inertia weight $w_k \in [0.05, 0.9]$ and the output variable of the second FLC, $fz^{(4,k)}$, is the social coefficient $c_{1,k} \in [1.0, 3.0]$. A fuzzy set consisting of three linguistic subsets is constructed for each input and output variable: $zs_1 = low$, $zs_2 = medium$, and $zs_3 = high$. Therefore, the fuzzy set for $fz^{(1,k)}$ is $\{zs_1^{(1)}, zs_2^{(1)}, zs_3^{(1)}\}$, for $fz^{(2,k)}$ is $\{zs_1^{(2)}, zs_2^{(2)}, zs_3^{(2)}\}$, for $fz^{(3,k)}$ is $\{zs_1^{(3)}, zs_2^{(3)}, zs_3^{(3)}\}$, and for $fz^{(4,k)}$ is $\{zs_1^{(4)}, zs_2^{(4)}, zs_3^{(4)}\}$.

Table 5.1: Centers and Shapes of Membership Functions

i	j			
	1	2	3	4
1	{0.10, 0.15}	{0.10, 0.15}	{0.10, 0.10}	{1.10, 0.40}
2	{0.50, 0.20}	{0.50, 0.20}	{0.40, 0.15}	{2.00, 0.40}
3	{0.90, 0.15}	{0.90, 0.15}	{0.85, 0.25}	{2.90, 0.40}

The fuzzy subsets are represented by Gaussian membership functions and defined as:

$$\mu_i(fz^{(j)}) = \mathrm{e}^{\frac{-(fz^{(j)}-\alpha_{i,j})^2}{2\sigma_{i,j}^2}}, i \in \{1,2,3\},\ j \in \{1,2,3,4\}. \tag{5.3}$$

where $\alpha_{i,j}$ and $\sigma_{i,j}$ are the center and the corresponding shape of the membership function of fuzzy subset i within the fuzzy set of variable $fz^{(j)}$, respectively. The values of $\{\alpha_{i,j}, \sigma_{i,j}\}$ are listed in Table 5.1.

The membership functions for the average convergence, convergence improvement, inertia weight, and social coefficent are plotted in Figures 5.4 through 5.7, respectively.

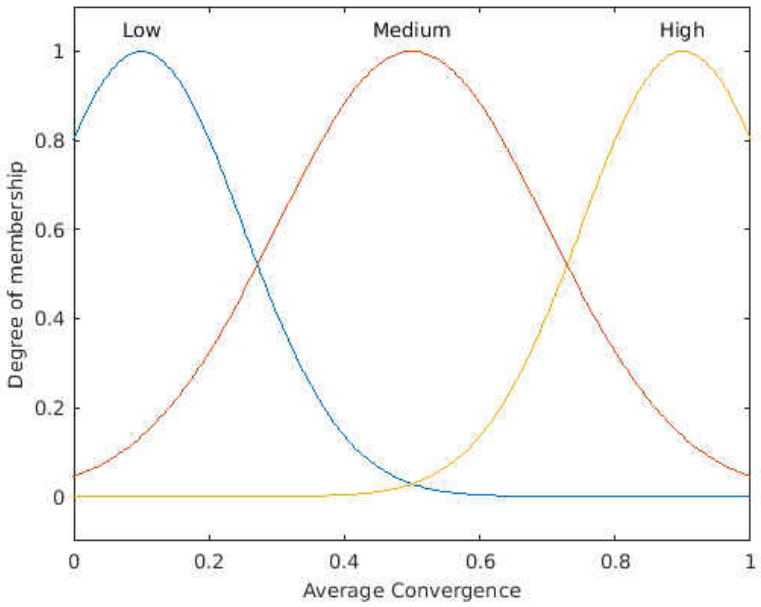

Figure 5.4: Degree of Membership vs. Average Convergence.

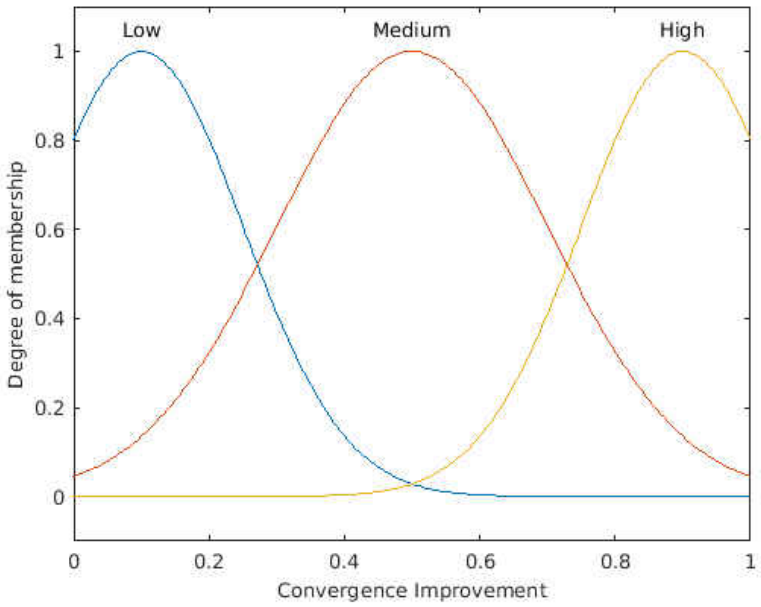

Figure 5.5: Degree of Membership vs. Convergence Improvement.

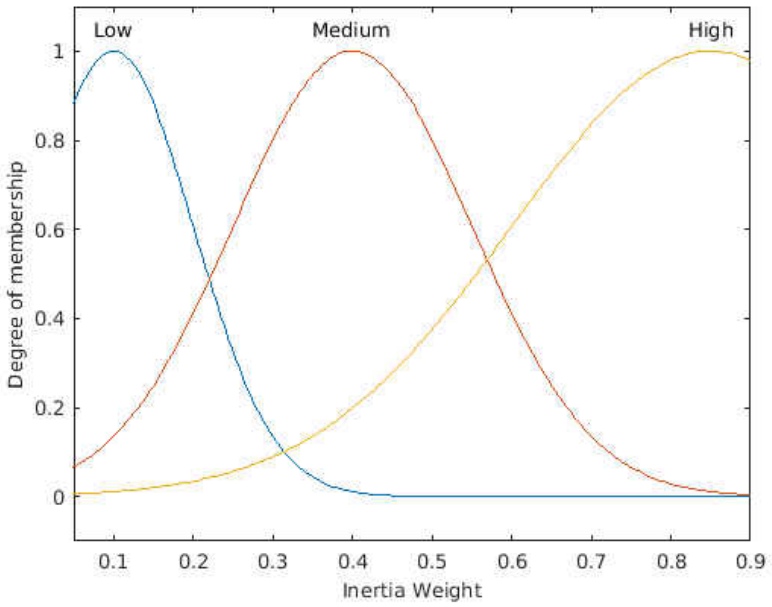

Figure 5.6: Degree of Membership vs. Inertia Weight.

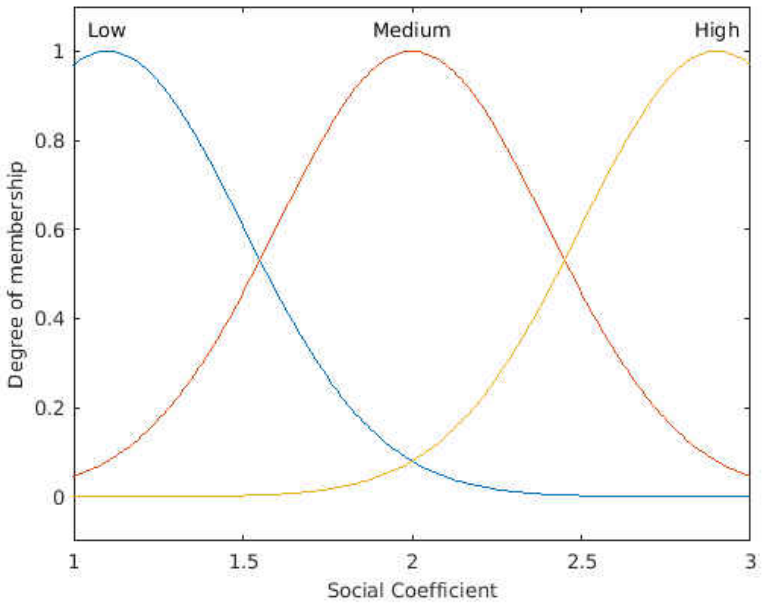

Figure 5.7: Degree of Membership vs. Social Coefficient.

Table 5.2: Fuzzy Rules

Rule	Fuzzy subset			
	$fz^{(1,k)}$	$fz^{(2,k)}$	$fz^{(3,k)}$	$fz^{(4,k)}$
1	L	L	L	L
2	L	M	L	L
3	L	H	M	M
4	M	L	M	M
5	M	M	M	M
6	M	H	H	H
7	H	L	M	M
8	H	M	H	H
9	H	H	H	H

The next step is to establish a set of inference rules in order to implement the fuzzification process. The fuzzy rules utilized in this work are based on Mamdani's direct method [75]; These rules, which correspond to $IF-THEN$ statements between the input and output fuzzy sets, are listed in Table 5.2. The antecedent sets (fuzzy input sets) and their respective membership functions are combined via a fuzzy AND (intersection) operator. For instance, rule $\#1$ in Table 5.2 specifies that if $fz^{(1,k)}$ is $zs_1^{(1)}$ and $fz^{(2)}$ is $zs_1^{(2)}$, then $fz^{(3,k)}$ is $zs_1^{(3)}$. In this case, rule $\#1$ is said to have fired. The fuzzy logic is implemented by using the membership functions of the fuzzy variables:

$$\mu_3(fz^{(3,k)})^{(r\#1)} = \mu_1(fz^{(1,k)}) \cap \mu_1(fz^{(2)}) \Rightarrow$$

$$\Rightarrow \mu_3(fz^{(3,k)})^{(r\#1)} = \min\{\mu_1(fz^{(1,k)}), \mu_1(fz^{(2,k)})\} \tag{5.4}$$

If the resulting membership function value is not zero, rule $\#1$ is said to have been activated.

The final task performed by the FLCs is to defuzzify the membership function values to obtain crisp values for w_k and $c_{1,k}$. For this purpose, the $centroid-of-area$ approach [76] is employed. The centroid, $fz_*^{(j)}$, (with $j \in \{3,4\}$), which corresponds to a crisp output variable value, is calculated as:

$$fz_*^{(j)} = \frac{\int \mu(fz^{(j)}) fz^{(j)}}{\int \mu(fz^{(j)})} \tag{5.5}$$

5.4 Performance Evaluation of FAMOPSOkkt

The FAMOPSOkkt algorithm developed in Intel Fortran 2017 is tested in the benchmark problems used for the evaluation of the ACMOPSO algorithm in Chapter 4. The quality indicators that are utilized for the performance evaluation are the Generational Distance (GD), the Inverse Generational Distance (IGD), and the KKT proximity measure.

A swarm consisting of 100 particles and divided into five sub-swarms is utilized. The capacity of the external archive is limited to 100 for all problems and the threshold of the number of leaders per cluster is set equal to ten. The procedure is repeated 31 with randomly generated initial swarm positions. The median values of the quality indicators among the 31 runs for each of the seven benchmark problems for ACMOPSO and FAMOPSOkkt are listed in Tables 5.3 and 5.4, respectively.

Table 5.3: ACMOPSO Quality Indicators.

Test problem	Generational Distance	Inverse Generational Distance	KKT proximity measure
ZDT2	2.18E-04	2.80E-04	4.69E-06
ZDT6	3.18E-02	2.11E-04	2.24E-13
TNK	5.44E-04	2.90E-04	2.30E-03
mTNK	4.25E-04	4.07E-04	2.50E-03
OSY	2.54E-04	3.03E-04	1.83E-01
WBD	4.72E-04	2.88E-03	9.31E-04
Viennet	3.15E-04	2.01E-04	9.66E-04

Algorithm 4 The FAMOPSOkkt Algorithm.

Initialize positions and velocities of particles (use a random seed for each sub-swarm)
Initialize iteration counter ($j = 0$) and external archive
for each sub-swarm **do**
    Evaluate objective functions and constraints
end for
for each particle **do**
    if the particle's current position is not dominated by its personal best position **then**
        Calculate the KKT proximity measure and *send* to the external archive for evaluation
    end if
end for
Combine the archived solutions with the new solutions generated by the sub-swarms
Use Pareto dominance criteria to find the nondominated solutions
Cluster the nondominated solutions using the set of reference points
Calculate the solution density of each cluster and the corresponding probability of selection
for each cluster **do**
    if the number of nondominated solutions within a cluster exceeds a threshold value **then**
        Select the solutions with the smallest KKT proximity values
    else
        Add all nondominated solutions within the cluster to the set of leaders
    end if
    Calculate the average convergence and the convergence improvement (input to the FLCs)
end for
if the number of nondominated solutions exceeds the capacity of the external archive **then**
    Remove solutions that reside in crowded regions
end if
for each sub-swarm **do**
    for each particle in the sub-swarm **do**
        if at least one leader is available **then**
            Select a leader from an associated cluster
            if $rand(0,1) \leq p_f$ **and** more than one leaders are available **then**
                Select a leader (randomly, but different than the first leader)
                for each decision variable j **do**
                    Update the corresponding velocity component and *compute* the new position
                end for
            else
                Select one leader(randomly) and a decision variable, j_{rand} (also randomly)
                for each decision variable j ,***if*** $(rand)(0,1) \leq \eta$ **or** $j = j_{rand}$ **do**
                    Mutate the j_{th} component of the leader
                end for
                Replace particle with mutated leader
            end if
        end if
    end for
end for

45: *Adapt* the inertia weight and social coefficient of each cluster using the corresponding FLCs
46: $t = t + 1$
47: until stopping criterion is satisfied

Table 5.4: FAMOPSOkkt Quality Indicators.

Test problem	Generational Distance	Inverse Generational Distance	KKT proximity measure
ZDT2	1.02E-04	2.70E-04	1.50E-05
ZDT6	7.20E-04	2.41E-04	9.06E-14
TNK	4.95E-04	3.49E-04	3.20E-03
mTNK	3.86E-04	3.16E-04	2.80E-03
OSY	2.21E-04	2.87E-04	1.74E-01
WBD	4.79E-04	8.16E-03	9.28E-04
Viennet	2.81E-04	2.71E-04	2.82E-05

5.4.1 Problem ZDT2

Figure 5.8 shows plots of the median GD and IGD values across the 31 runs for problem ZDT2 and Table 5.5 shows the mean, median, and standard deviation of the KKT proximity measure for 50 iterations across 31 runs. A comparison between the GD and IGD plots in Figures 4.1 and 5.8 for ACMOPSO and FAMOPSOkkt, respectively, reveals that FAMOPSOkkt is able to find a well-distributed set of solutions near the true Pareto-optimal front within twenty iterations, on an average basis, which is significantly faster than the ACMOPSO algorithm. Figure 5.9 shows the true Pareto-optimal front as a solid line and the computed front as circles for the ZDT2 problem plotted for the run that corresponds to the median value of the KKT proximity measure. It needs to be emphasized that the median values for both GD and IGD are smaller for the FAMOPSOkkt, in addition to a much smaller standard deviation.

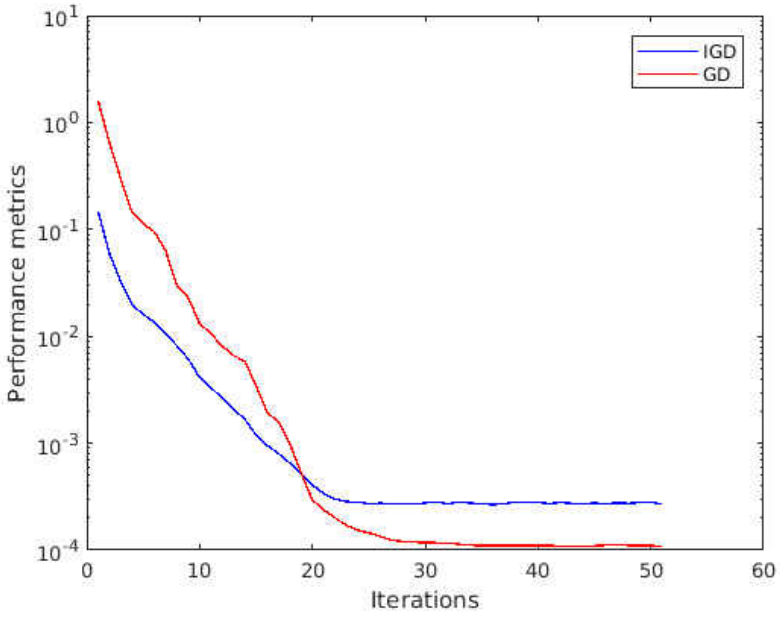

Figure 5.8: GD and IGD Values for the Problem ZDT2.

Table 5.5: ZDT2 KKT Proximity Measure.

Metric	Mean	Median	Std. Dev
KKTPM	2.17E-04	1.50E-05	7.96E-04
GD	1.08E-04	1.02E-04	1.77E-05
IGD	2.69E-04	2.70E-04	1.71E-05

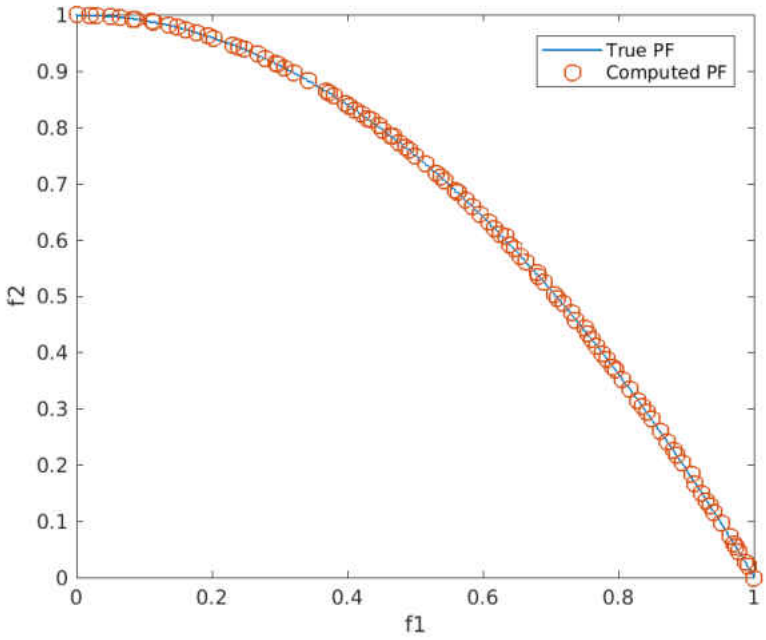

Figure 5.9: Pareto Front for the Problem ZDT2.

5.4.2 Problem ZDT6

In contrast to the performance of the ACMOPSO algorithm, FAMOPSOkkt is able to perform a local search and attract all the nondominated solutions near the Pareto-optimal front as is evident in Figures 5.10 and 5.12.A similar conclusion can be drawn by observing the convergence of the KKT proximity measure in 5.11, where the worst value drops to 10^{-10} after approximately 40 iterations. FAMOPSOkkt requires approximately 15 iterations to reach IGD values in the order of 10^{-4}, while ACMOPSO requires approximately 30 iterations.

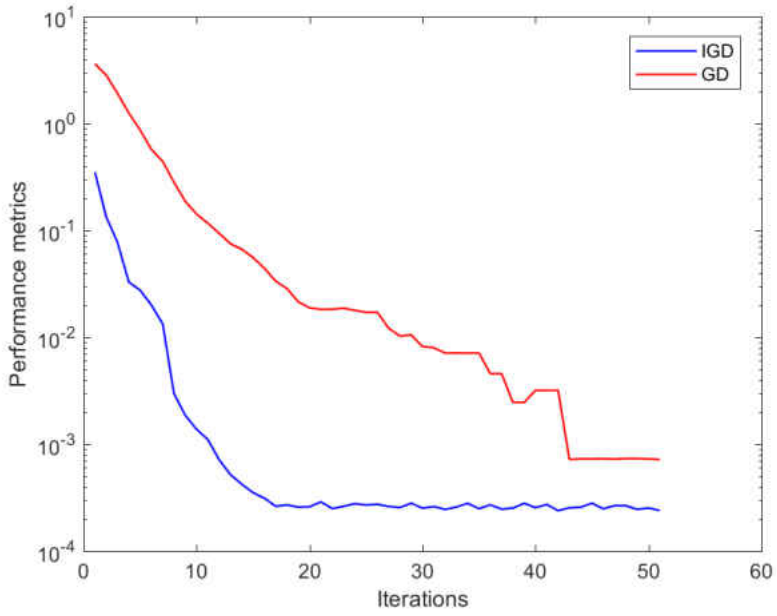

Figure 5.10: GD and IGD Values for the Problem ZDT6.

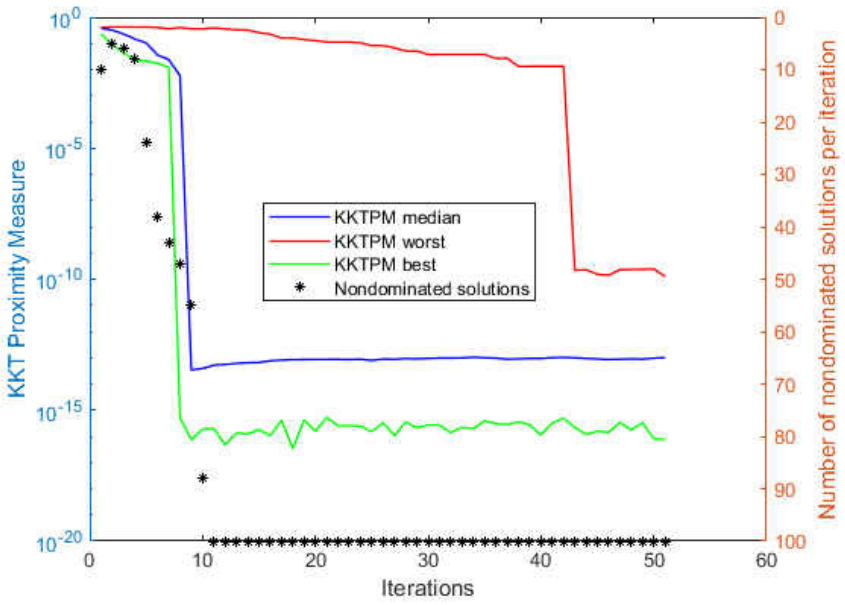

Figure 5.11: KKT Proximity Measure for the Problem ZDT6.

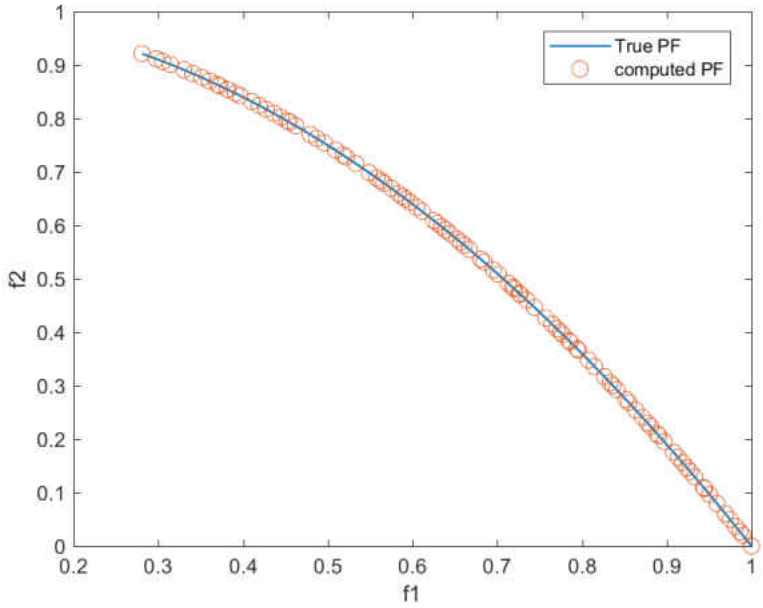

Figure 5.12: Pareto Front for the Problem ZDT6.

5.4.3 Problem OSY

The convergence of the GD and IGD quality indicators is demonstrated in Figures 5.13 and 5.14. A comparison with Figures 4.6 and 4.7 shows similar behavior for the two optimizers, but the median value of the KKT proximity measure is smaller for the FAMOPSOkkt algorithm. The computed Pareto-optimal front that corresponds to the run with the median value of the KKT proximity measure is plotted in Figure 5.15.

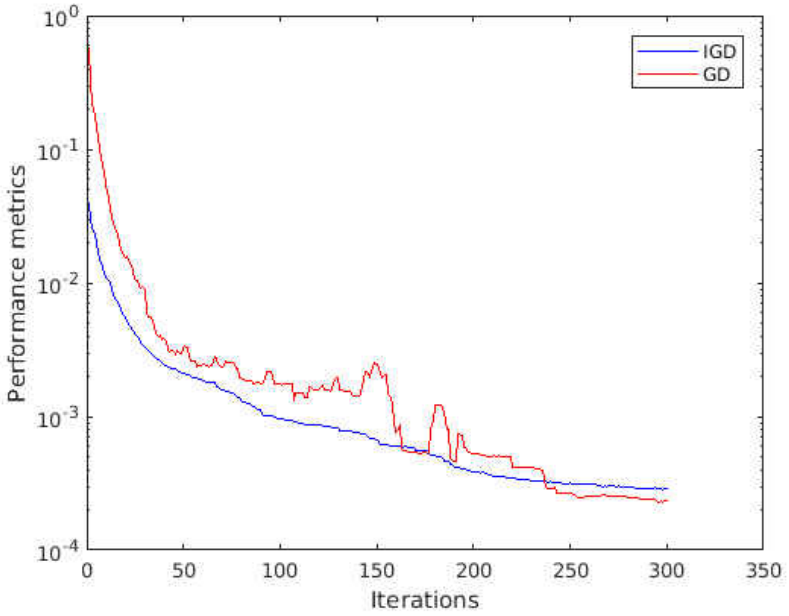

Figure 5.13: GD and IGD Values for the Problem OSY.

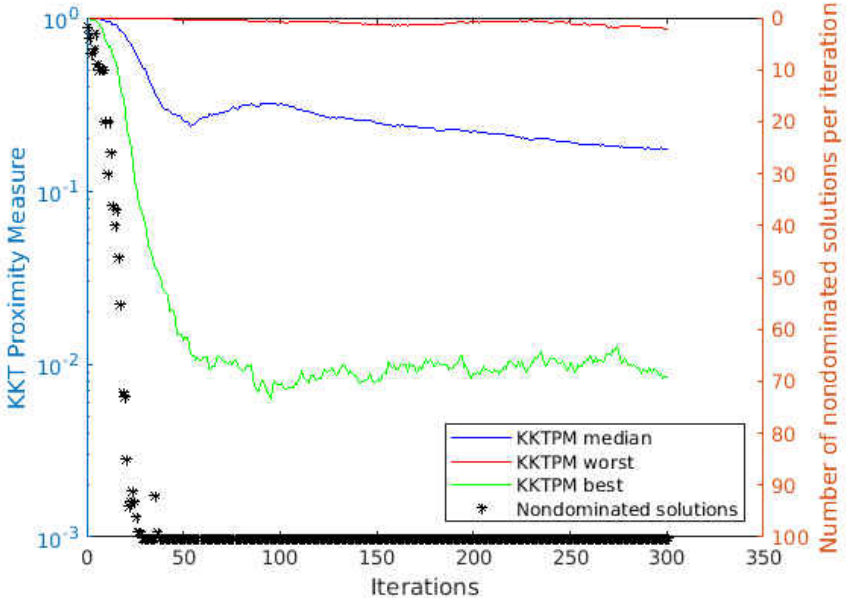

Figure 5.14: KKT Proximity Measure for the Problem OSY.

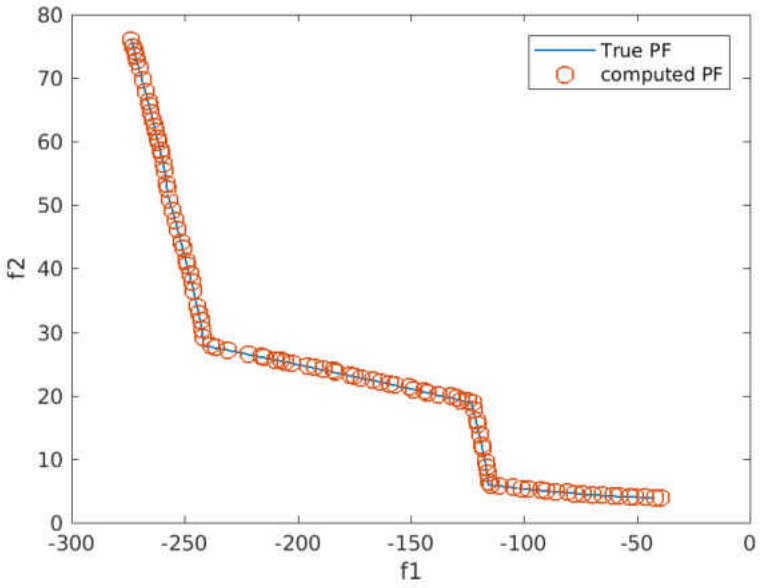

Figure 5.15: Pareto Front for the Problem OSY.

5.4.4 Problem TNK

Figure 5.18 shows the disconnected Pareto front for the original TNK problem. The FAMOP-SOkkt algorithm is capable of finding near Pareto-optimal solutions in all the disconnected segments of the front in a similar manner to the ACMOPSO algorithm as is revealed through a comparison between Figures 5.16 and 5.17 and Figures 4.9 and 4.10, respectively. The Pareto-optimal front that corresponds to the run with the median value of the KKT proximity measure is shown in 5.18.

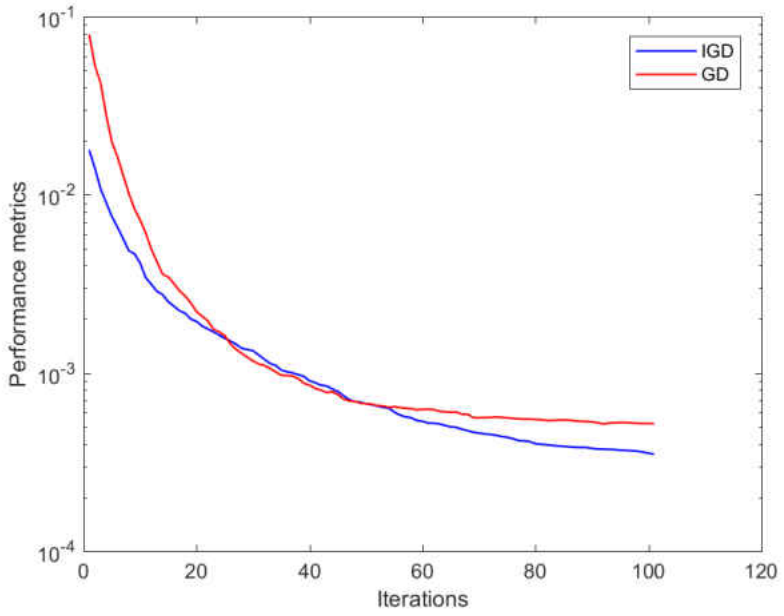

Figure 5.16: GD and IGD Values for the Problem TNK.

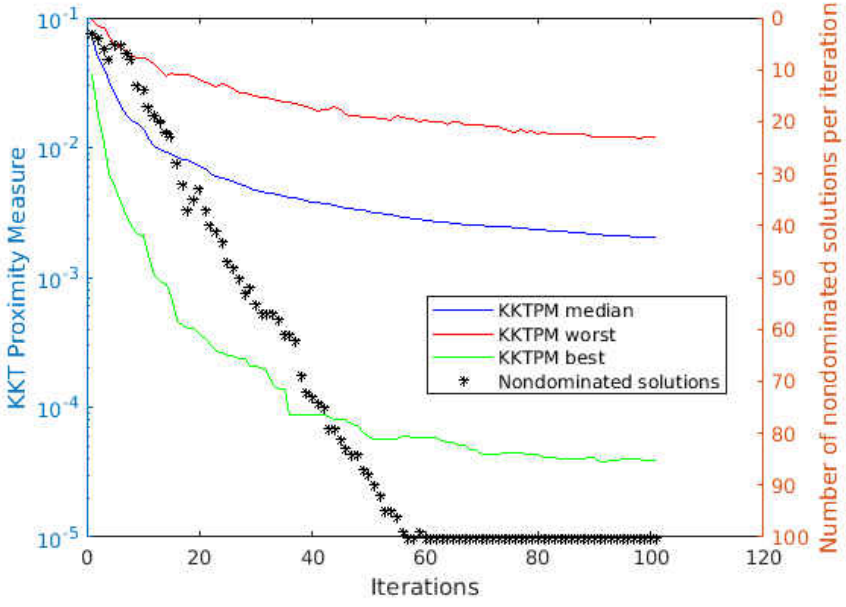

Figure 5.17: KKT Proximity Measure for the Problem TNK.

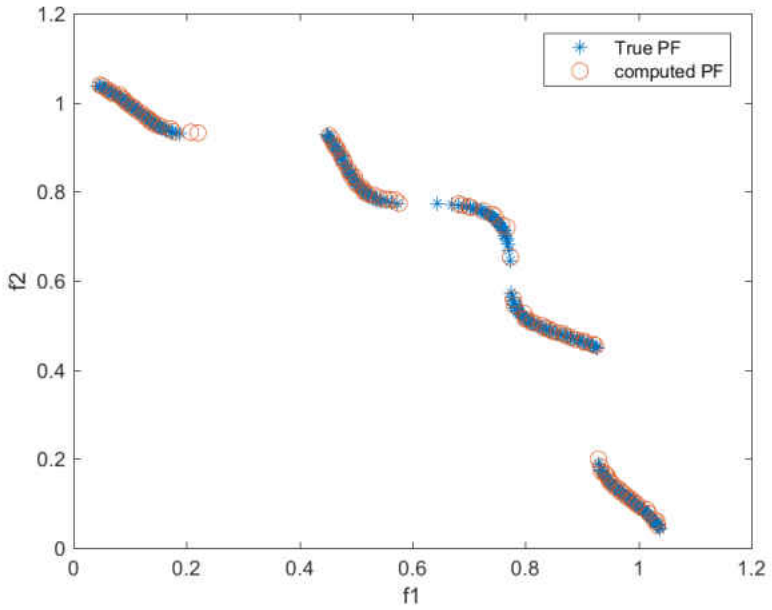

Figure 5.18: Pareto Front for the Problem TNK.

5.4.5 Problem mTNK

Figure 5.21 shows the eight disconnected segments of the Pareto-optimal front for the modified TNK problem. The FAMOPSOkkt algorithm is capable of finding near Pareto-optimal solutions in all the disconnected segments of the front in a similar manner to the ACMOPSO algorithm. However, the median values of both the GD and IGD metrics are smaller for the FAMOPSOkkt algorithm as revealed through a comparison between Figures 5.19 and 4.12 and by the values listed in Tables 5.3 and 5.4. The Pareto-optimal front that corresponds to the run with the median value of the KKT proximity measure is shown in 5.21.

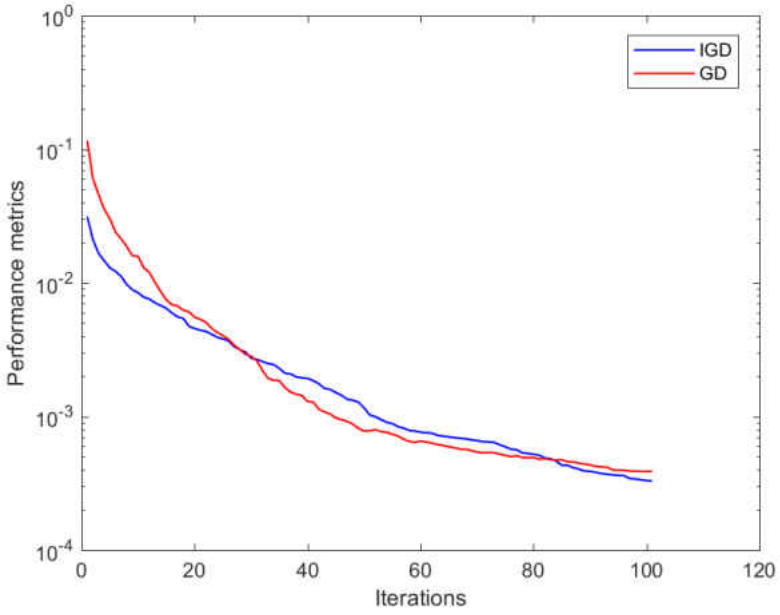

Figure 5.19: GD and IGD Values for the Problem mTNK.

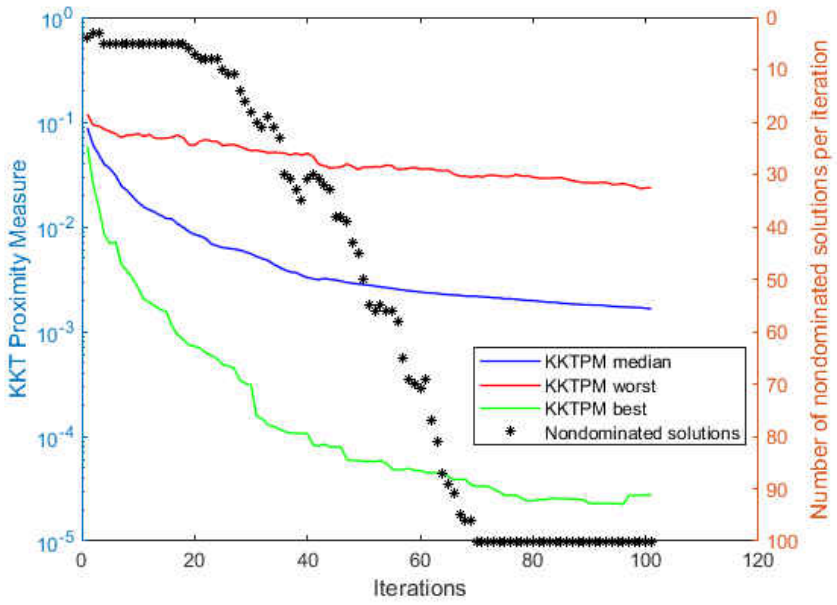

Figure 5.20: KKT Proximity Measure for the Problem mTNK.

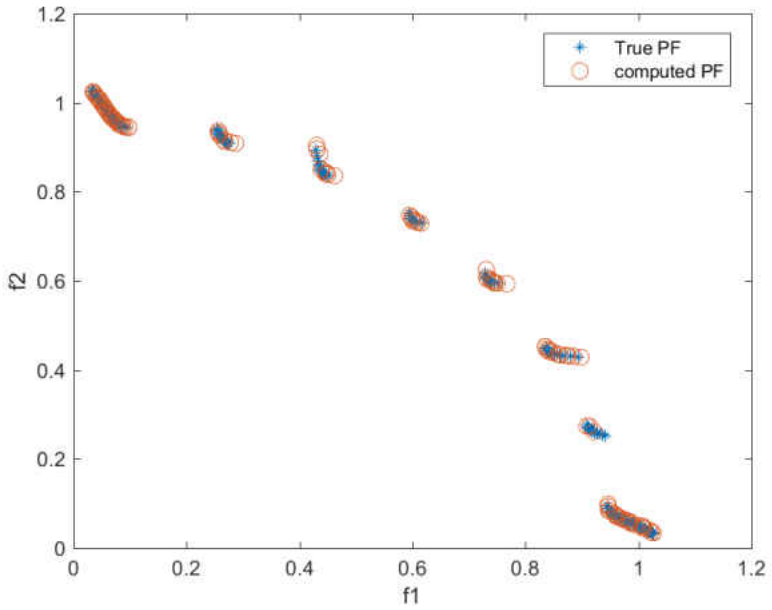

Figure 5.21: Pareto Front for the Problem mTNK.

5.4.6 Problem WBD

The FAMOPSOkkt algorithm is capable of finding near Pareto-optimal solutions for this engineering design problem in a similar manner to the ACMOPSO algorithm. However, the median values of both the GD and IGD metrics are smaller for the FAMOPSOkkt algorithm as revealed through a comparison between Figures 5.22 and 4.15 and by the values listed in Tables 5.3 and 5.4. The Pareto-optimal front that corresponds to the run with the median value of the KKT proximity measure is shown in 5.24.

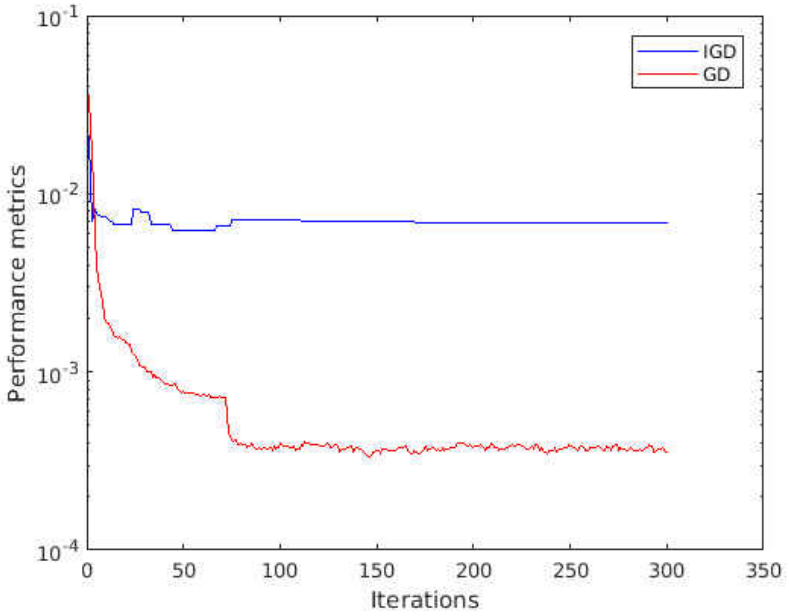

Figure 5.22: GD and IGD Values for the Problem WBD.

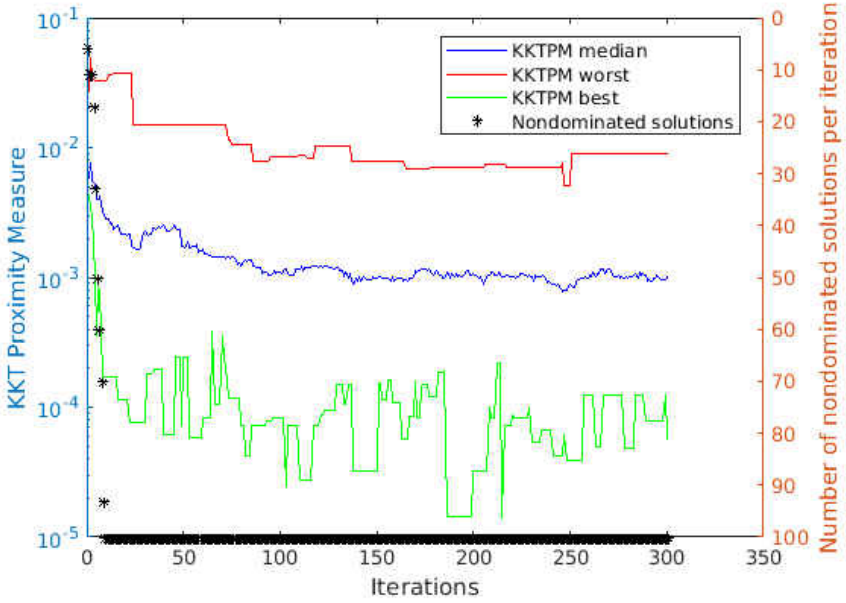

Figure 5.23: KKT Proximity Measure for the Problem WBD.

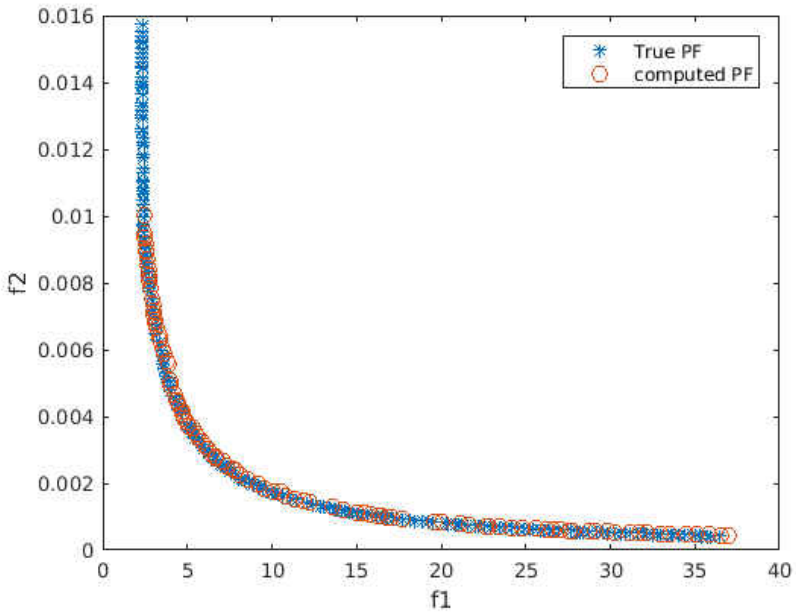

Figure 5.24: Pareto Front for the Problem WBD.

5.4.7 Problem VNT

The performance of FAMOPSOkkt in the Viennet problem is demonstrated in Figures 5.25 and 5.26. The obtained GD and IGD values are similar to the values obtained by ACMOPSO. However, FAMOPSOkkt is able to find solutions with lower KKT proximity measure values as shown in 5.26 compared to 4.19. The Pareto-optimal front that corresponds to the run with the median value of the KKT proximity measure is shown in 5.27.

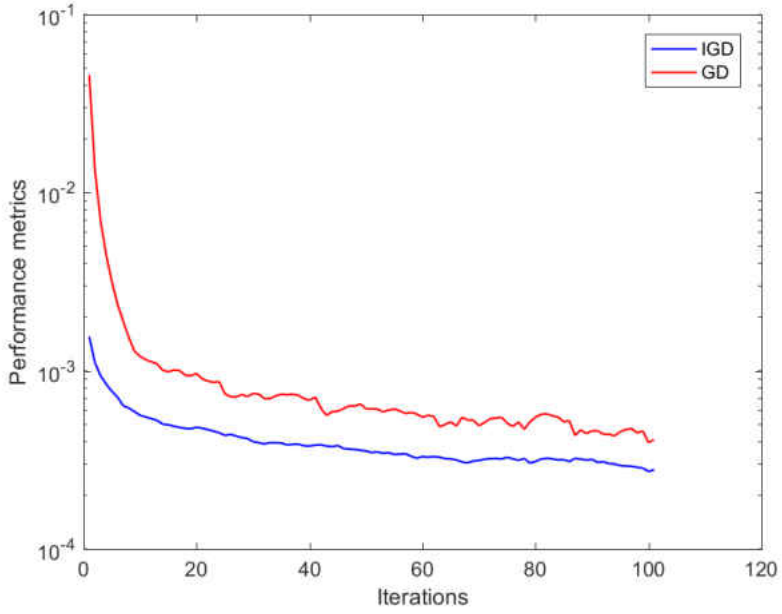

Figure 5.25: GD and IGD Values for the Problem VNT.

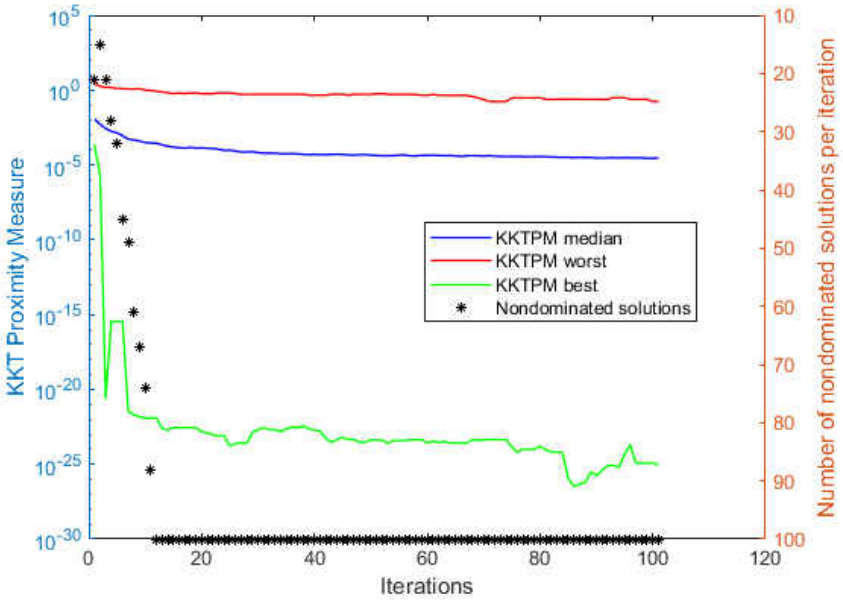

Figure 5.26: KKT Proximity Measure for the Problem VNT.

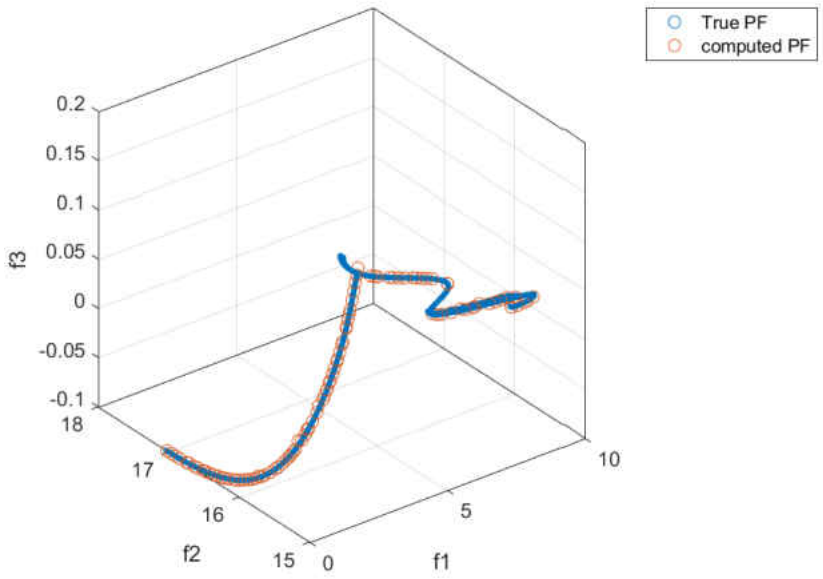

Figure 5.27: Pareto Front for the Problem VNT.

CHAPTER 6

BENCHMARKING FAMOPSOkkt

In this section, we benchmark the FAMOPSOkkt algorithm against state-of-the-art multi-objective optimization algorithms. FAMOPSOkkt is tested against GDE3 [77], SMPSO [58, 78], MOCell [79, 80], and NSGA III's [81] implementation in jMetal. For these experiments, a redesigned version of jMetal [82, 83, 84] is employed. jMetal stands for Metaheuristic Algorithms in Java, and it is an object-oriented Java-based framework for multi-objective optimization with metaheuristics. A brief description of the chosen algorithms is given below for completeness.

6.1 GDE3

GDE3 is the third version of the Generalized Differential Evolution which uses the standard version - *DE/rand/1/bin* version [77]. Pareto dominance is used to make comparisons between target and trial vectors to select new population members. The population size is maintained using Pareto ranking and crowding distance.

6.2 SMPSOhv

Speed-Constrained Multi-Objective Particle Swarm Optimizer (SMPSO) [58, 78] implements a velocity constriction mechanism, polynomial mutation, and maintains an external archive to store the non-dominated solutions found during the search process. Crowding distance is also calculated and used as a density estimator and as a leader selection criterion. In this comparison, a specific version, SMPSOhv, is utilized; SMPSOhv uses hypervolume as a density estimator and takes its contribution into account while selecting leaders from the external archive using binary tournament selection. The particles in the archive contributing the most to the hypervolume have higher chances

to be chosen as a leader, while the particle in the archive contributing the least to the hypervolume is removed.

6.3 MOCellhv

Multi-objective Cellular Genetic Algorithm (MOCell) is a family of cellular genetic algorithms which uses an external archive to store non-dominated solutions and a feedback mechanism in which solutions from this archive randomly replace existing individuals in the population after each iteration [79, 80]. Like SMPSO, this algorithm implements polynomial mutation and a crowding distance mechanism to maintain solution diversity in the archive.

6.4 NSGA-III

Non-dominated sorting genetic algorithm, the third version (NSGA -III) is a revision to NSGA-II which uses a modified elitist selection mechanism and a set of pre-defined reference points. All the objective vectors and the supplied reference points are normalized and every population member is associated with a particular reference point based on a proximity measure. Niching of accepted population members is done to ensure a diverse set of solutions. However, jMetal implements an improved version of NSGA-III, called θ-NSGA-III. [81]. This implementation retains most of the salient features of the original NSGA-III like adaptive normalization and diversity-preservation aided by well-spread reference points, while replacing the Pareto dominance by a new θ-dominance. In θ-dominance the solutions are allocated into different clusters represented by the well-distributed reference points. Solutions belonging to the same cluster are subjected to a competitive relationship defining a fitness function similar to penalty-based boundary intersection function; and the ones with the better fitness are selected and kept in the population.

6.5 Results

Quality indicators like hypervolume, GD, and IGD are also available in the jMetal framework. Two implementations of hypervolume (HV) calculation are available: PISA hypervolume [85] and WFH hypervolume [86]. The hypervolume quality indicator is capable of assessing the convergence of the computed Pareto-optimal front to the true Pareto-optimal front and also whether the computed nondominated solutions correspond to a well-distributed set. All algorithms are run for 31 runs and the values of the quality indicators, i.e., HV, IGD, and GD are reported in Tables 6.1, 6.2, and 6.3, respectively.

The statistical significance of the results is assessed on the hypervolume values using the Mann-Whitney-Wilcoxon (MWM) nonparametric test developed in [87] and in [88]. The algorithm with the highest HV median value in each problem is compared with the other four algorithms using a one-tailed MWM test. The null hypothesis of equal median values is tested at the 5% significance level. If the null hypothesis is rejected in all four comparisons, the corresponding median value is highlighted using bold font style. The performance of the FAMOPSOkkt based on the HV results is very satisfactory as it produces the highest median value in six out of seven problems; a result that is statistically significant at the 5% level in all six problems.

Table 6.1: Hypervolume (HV) Results for Problems.

Problem	Algorithm	Mean	Median	Std. Dev.
ZDT2	GDE3	0.17163	0.17135	0.01391
	SMPSOHV	0.06446	0.00264	0.10116
	MOCellHV	0.04421	0.00338	0.05750
	NSGA-III	0.02134	0.00000	0.03269
	FAMOPSOkkt	0.32745	**0.32746**	0.00029
ZDT6	GDE3	0.39160	0.39698	0.01369
	SMPSOHV	0.38516	0.39549	0.03881
	MOCellHV	0.03141	0.03552	0.01969
	NSGA-III	0.00000	0.00000	0.00000
	FAMOPSOkkt	0.40037	**0.40037**	0.00015
TNK	GDE3	0.30612	0.30608	0.00049
	SMPSOHV	0.29844	0.29853	0.00168
	MOCellHV	0.30578	0.30587	0.00075
	NSGA-III	0.30500	0.30509	0.00103
	FAMOPSOkkt	0.30646	**0.30653**	0.00061
mTNK	GDE3	0.30397	0.30406	0.00114
	SMPSOHV	0.29294	0.29225	0.00328
	MOCellHV	0.30332	0.30355	0.00123
	NSGA-III	0.30336	0.30383	0.00143
	FAMOPSOkkt	0.30565	**0.30567**	0.00085
OSY	GDE3	0.73013	0.73133	0.01611
	SMPSOHV	0.64730	0.69893	0.10051
	MOCellHV	0.65910	0.74553	0.12252
	NSGA-III	0.66507	0.67258	0.15153
	FAMOPSOkkt	0.75307	**0.75308**	0.00026
WBD	GDE3	0.91261	0.91478	0.01041
	SMPSOHV	0.92228	0.92210	0.00294
	MOCellHV	0.92449	0.92457	0.00551
	NSGA-III	0.91109	0.91591	0.01759
	FAMOPSOkkt	0.92670	**0.92849**	0.00427
Viennet	GDE3	0.83388	0.83389	0.00055
	SMPSOHV	0.83592	0.83591	0.00029
	MOCellHV	0.83590	0.83589	0.00018
	NSGA-III	0.84118	**0.84118**	0.00001
	FAMOPSOkkt	0.83389	0.83396	0.00045

Table 6.2: Inverse Generational Distance (IGD) Results for Problems.

Problem	Algorithm	Mean	Median	Std. Dev.
ZDT2	GDE3	4.26E-003	4.20E-003	4.81E-004
	SMPSOHV	1.74E-002	1.81E-002	1.07E-002
	MOCellHV	1.63E-002	1.85E-002	6.69E-003
	NSGA-III	2.31E-002	2.93E-002	9.53E-003
	FAMOPSOkkt	2.69E-004	2.71E-004	1.71E-005
ZDT6	GDE3	5.04E-004	3.09E-004	3.09E-004
	SMPSOHV	8.56E-004	4.10E-004	1.75E-003
	MOCellHV	1.41E-002	1.35E-002	2.35E-003
	NSGA-III	5.70E-002	5.71E-002	6.86E-003
	FAMOPSOkkt	2.40E-004	2.41E-004	3.66E-005
TNK	GDE3	5.38E-004	5.27E-004	5.66E-005
	SMPSOHV	1.01E-003	1.01E-003	1.44E-004
	MOCellHV	6.19E-004	6.11E-004	7.16E-005
	NSGA-III	7.43E-004	7.19E-004	1.39E-004
	FAMOPSOkkt	3.50E-004	3.49E-004	4.46E-005
mTNK	GDE3	5.11E-004	4.86E-004	9.77E-005
	SMPSOHV	1.18E-003	1.17E-003	2.28E-004
	MOCellHV	4.32E-004	4.25E-004	5.33E-005
	NSGA-III	5.15E-004	5.05E-004	7.17E-005
	FAMOPSOkkt	3.29E-004	3.17E-004	7.45E-005
OSY	GDE3	4.20E-003	3.82E-003	3.30E-003
	SMPSOHV	8.36E-003	9.06E-003	2.39E-003
	MOCellHV	5.73E-003	2.93E-003	6.02E-003
	NSGA-III	6.15E-003	6.04E-003	2.04E-003
	FAMOPSOkkt	2.86E-004	2.87E-004	1.47E-005
WBD	GDE3	2.04E-003	1.49E-002	3.94E-003
	SMPSOHV	9.05E-003	9.53E-003	2.65E-003
	MOCellHV	6.92E-003	6.52E-003	4.14E-003
	NSGA-III	1.24E-002	1.16E-002	4.96E-003
	FAMOPSOkkt	7.26E-003	8.16E-003	2.56E-003
Viennet	GDE3	4.96E-003	1.60E-004	1.47E-005
	SMPSOHV	8.72E-004	8.92E-004	6.46E-005
	MOCellHV	8.77E-004	8.81E-004	4.02E-005
	NSGA-III	3.04E-005	2.95E-005	3.28E-006
	FAMOPSOkkt	2.79E-004	2.71E-004	9.51E-005

Table 6.3: Generational Distance (GD) Results for Problems.

Problem	Algorithm	Mean	Median	Std . Dev
ZDT2	GDE3	2.57E-002	2.62E-002	3.64E-003
	SMPSOHV	1.22E-001	6.33E-002	1.60E-001
	MOCellHV	3.32E-002	2.89E-002	2.18E-002
	NSGA-III	1.41E-001	1.49E-001	7.58E-002
	FAMOPSOkkt	1.08E-004	1.021E-004	1.765E-005
ZDT6	GDE3	7.85E-003	4.45E-005	1.98E-002
	SMPSOHV	6.27E-002	6.13E-002	3.88E-002
	MOCellHV	1.04E-001	9.65E-002	3.18E-002
	NSGA-III	6.03E-001	5.75E-001	1.32E-001
	FAMOPSOkkt	7.20E-004	7.20E-004	3.48E-005
TNK	GDE3	6.83E-004	6.95E-004	7.95E-005
	SMPSOHV	1.44E-003	1.42E-003	2.47E-004
	MOCellHV	6.31E-004	6.12E-004	9.27E-005
	NSGA-III	6.94E-004	6.71E-004	1.38E-004
	FAMOPSOkkt	5.19E-004	4.95E-004	1.04E-004
mTNK	GDE3	5.44E-004	5.45E-004	1.21E-004
	SMPSOHV	2.40E-003	2.36E-003	4.96E-004
	MOCellHV	4.80E-004	4.13E-004	1.83E-004
	NSGA-III	3.84E-004	3.74E-004	1.26E-004
	FAMOPSOkkt	3.89E-004	3.86E-004	8.11E-005
OSY	GDE3	1.72E-003	1.57E-003	7.62E-004
	SMPSOHV	8.47E-003	1.94E-003	1.04E-002
	MOCellHV	5.86E-003	1.91E-003	7.98E-003
	NSGA-III	8.13E-003	8.23E-003	1.54E-003
	FAMOPSOkkt	2.33E-004	2.21E-004	6.91E-005
WBD	GDE3	3.50E-004	3.32E-004	6.08E-005
	SMPSOHV	4.44E-004	4.02E-004	2.65E-004
	MOCellHV	5.63E-003	1.17E-003	8.19E-003
	NSGA-III	8.43E-004	5.61E-004	6.84E-004
	FAMOPSOkkt	7.26E-004	4.79E-004	5.52E-004
Viennet	GDE3	2.42E-004	1.97E-004	1.23E-004
	SMPSOHV	8.19E-005	5.80E-005	6.63E-005
	MOCellHV	5.80E-005	5.65E-005	6.67E-006
	NSGA-III	8.42E-006	8.41E-006	1.17E-007
	FAMOPSOkkt	4.11E-004	2.81E-004	3.38E-004

CHAPTER 7

CONCLUSIONS

The multifaceted goal of solving optimization problems with conflicting objectives is to generate a a well-distributed set of near-Pareto-optimal solutions. Stochastic algorithms based on evolutionary principles or swarm intelligence attempt to solve multi-objective problems without requiring knowledge of the derivatives of the problem objectives and constraints. That approach, although successfull when applied to various problems, typically requires a trial-and-error process in order to adjust the algorithmic parameters. The KKT proximity measure was derived by evolutionary algorithm optimization researchers in order to quantify the proximity of the computed nondominated solutions to the global Pareto-optimal front. It has also been shown to converge monotonically towards the Pareto-optimal front.

In this work, a Fuzzy-Adaptive Multi-Objective Optimization Algorithm with the KKT proximity measure (FAMOPSOkkt) is proposed with two main goals. The first goal is to develop an algorithm capable of obtaining feedback from the search process performed by the swarm particles regarding the convergence towards the Pareto-optimal front and use that feedback to adapt its algorithmic parameters in an interactive manner. The second goal is to enhance its local search capability by utilizing a set of reference points to cluster the computed nondominated solutions. These clusters interact with their corresponding sub-swarms to provide their leaders, which are utilized as the attraction basin of the swarm particles. The reference points and their associated clusters are further utilized to manage the external archive of nondominated solutions.

The proposed algorithm is evaluated in a number of benchmark problems that provide various challenges to a multi-objective optimizer. A comparison of the performance of the proposed multi-objective optimization algorithm in the benchmark problems with the performance of the optimization algorithm that was used as the blueprint for the development of the proposed algorithm

reveals that both targets hane been met, i.e., successful adaptation of the algorithmic parameters and more effective local search in the vicinity of the Pareto-optimal front. Further benchmarking of FAMOPSOkkt against state-of-the-art evolutionary optimization algorithms with similar features is performed on seven benchmark problems using the hypervolume quality indicator. The statistical significance of the results, with respect to the median values, is assessed using the Mann-Whitney-Wilcoxon nonparametric test. The FAMOPSOkkt algorithm performs better than all the other algorithms in six problems; the results are statistically significant at the 5% level in all six problems.

The performance of the FAMOPSOkkt algorithm in synthetic benchmark problems and a real-world design problem is very promising. However, the performance of the algorithm in a problem with three objectives was not as successful as in the two-objective problems. A direction for future research is to investigate methods that would allow FAMOPSOkkt to scale its performance to many-objective optimization problems without a significant increase in the computational cost.

REFERENCES

[1] Merriam-Webster Online, "Merriam-Webster Online Dictionary," 2009. [Online]. Available: http://www.merriam-webster.com

[2] O. L. De Weck, "Multiobjective optimization: History and promise," in *Invited Keynote Paper, GL2-2, The Third China-Japan-Korea Joint Symposium on Optimization of Structural and Mechanical Systems, Kanazawa, Japan*, vol. 2, 2004, p. 34.

[3] F. Y. Edgeworth, *Mathematical psychics: An essay on the application of mathematics to the moral sciences*. Kegan Paul, 1881, vol. 10.

[4] V. Pareto and M. di Economia Politica, "Societa editrice libraria, milano, italy, 1906. translated into english by as schwier as manual of political economy," 1971.

[5] R. S. Rosenberg, "Stimulation of genetic populations with biochemical properties: I. the model," *Mathematical Biosciences*, vol. 7, no. 3, pp. 223 – 257, 1970.

[6] ——, "Simulation of genetic populations with biochemical properties: Ii. selection of crossover probabilities," *Mathematical Biosciences*, vol. 8, no. 1, pp. 1 – 37, 1970.

[7] J. D. Schaffer, "Some experiments in machine learning using vector evaluated genetic algorithms," Vanderbilt Univ., Nashville, TN (USA), Tech. Rep., 1985.

[8] I. Alaya, C. Solnon, and K. Ghedira, "Ant colony optimization for multi-objective optimization problems," in *Tools with Artificial Intelligence, 2007. ICTAI 2007. 19th IEEE International Conference on*, vol. 1. IEEE, 2007, pp. 450–457.

[9] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," *IEEE computational intelligence magazine*, vol. 1, no. 4, pp. 28–39, 2006.

[10] C. A. C. Coello, G. B. Lamont, D. A. Van Veldhuizen *et al.*, *Evolutionary algorithms for solving multi-objective problems*. Springer, 2007, vol. 5.

[11] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach," *IEEE transactions on Evolutionary Computation*, vol. 3, no. 4, pp. 257–271, 1999.

[12] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: Empirical results," *Evolutionary computation*, vol. 8, no. 2, pp. 173–195, 2000.

[13] A. Zhou, B.-Y. Qu, H. Li, S.-Z. Zhao, P. N. Suganthan, and Q. Zhang, "Multiobjective evolutionary algorithms: A survey of the state of the art," *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 32–49, 2011.

[14] U. Chakraborty, *Advances in differential evolution*. Springer, 2008, vol. 143.

[15] S. Kukkonen and J. Lampinen, "Gde3: The third evolution step of generalized differential evolution," in *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, vol. 1. IEEE, 2005, pp. 443–450.

[16] T. Robič and B. Filipič, "Differential evolution for multiobjective optimization," in *Evolutionary multi-criterion optimization*. Springer, 2005, pp. 520–533.

[17] J. Moore, R. Chapman, and G. Dozier, "Multiobjective particle swarm optimization," in *Proceedings of the 38th annual on Southeast regional conference*. ACM, 2000, pp. 56–57.

[18] M. Reyes-Sierra and C. C. Coello, "Multi-objective particle swarm optimizers: A survey of the state-of-the-art," *International journal of computational intelligence research*, vol. 2, no. 3, pp. 287–308, 2006.

[19] K. Miettinen, "Nonlinear multiobjective optimization, volume 12 of international series in operations research and management science," 1999.

[20] K. Deb and J. Sundar, "Reference point based multi-objective optimization using evolutionary algorithms," in *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO '06. New York, NY, USA: ACM, 2006, pp. 635–642. [Online]. Available: http://doi.acm.org/10.1145/1143997.1144112

[21] K. Sindhya, K. Miettinen, and K. Deb, "A hybrid framework for evolutionary multi-objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 4, pp. 495–511, Aug 2013.

[22] Q. Zhang and H. Li, "Moea/d: A multiobjective evolutionary algorithm based on decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, pp. 712–731, Dec 2007.

[23] Q. Zhang, H. Li, D. Maringer, and E. Tsang, "Moea/d with nbi-style tchebycheff approach for portfolio management," in *IEEE Congress on Evolutionary Computation*, July 2010, pp. 1–8.

[24] R. Cheng, Y. Jin, M. Olhofer, and B. Sendhoff, "A reference vector guided evolutionary algorithm for many-objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 5, pp. 773–791, Oct 2016.

[25] K. Deb and H. Jain, "An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 4, pp. 577–601, Aug 2014.

[26] H. Jain and K. Deb, "An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting approach, part ii: Handling constraints and extending to an adaptive approach," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 4, pp. 602–622, Aug 2014.

[27] S. Greco, M. Ehrgott, and J. R. Figueira, *Multiple criteria decision analysis : state of the art surveys*. SpringerLink, 2005.

[28] H. P. Benson, "Vector maximization with two objective functions," *Journal of Optimization Theory and Applications*, vol. 28, no. 2, pp. 253–257, Jun 1979. [Online]. Available: https://doi.org/10.1007/BF00933245

[29] J. Kennedy and R. Eberhart, "Particle swarm optimization," 1995.

[30] R. C. Eberhart, Y. Shi, and J. Kennedy, *Swarm intelligence*. Elsevier, 2001.

[31] G. Beni and J. Wang, "Swarm intelligence in cellular robotic systems," in *Robots and Biological Systems: Towards a New Bionics?* Springer, 1993, pp. 703–712.

[32] K. E. Parsopoulos and M. N. Vrahatis, "Particle swarm optimization method in multiobjective problems," in *Proceedings of the 2002 ACM Symposium on Applied Computing*, ser. SAC '02. New York, NY, USA: ACM, 2002, pp. 603–607. [Online]. Available: http://doi.acm.org/10.1145/508791.508907

[33] C. A. C. Coello, G. T. Pulido, and M. S. Lechuga, "Handling multiple objectives with particle swarm optimization," *IEEE Transactions on evolutionary computation*, vol. 8, no. 3, pp. 256–279, 2004.

[34] J. Alvarez-Benitez, R. Everson, and J. Fieldsend, "A mopso algorithm based exclusively on pareto dominance concepts," in *Evolutionary Multi-Criterion Optimization*. Springer, 2005, pp. 459–473.

[35] C. K. Goh, K. C. Tan, D. Liu, and S. C. Chiam, "A competitive and cooperative co-evolutionary approach to multi-objective particle swarm optimization algorithm design," *European Journal of Operational Research*, vol. 202, no. 1, pp. 42–54, 2010.

[36] M. Kotinis, "Implementing co-evolution and parallelization in a multi-objective particle swarm optimizer," *Engineering Optimization*, vol. 43, no. 6, pp. 635–656, 2011. [Online]. Available: http://dx.doi.org/10.1080/0305215X.2010.508522

[37] K. Khalili-Damghani, A.-R. Abtahi, and M. Tavana, "A new multi-objective particle swarm optimization method for solving reliability redundancy allocation problems," *Reliability Engineering & System Safety*, vol. 111, pp. 58–75, 2013.

[38] Y. Zhang, S. Wang, and G. Ji, "A comprehensive survey on particle swarm optimization algorithm and its applications," *Mathematical Problems in Engineering*, vol. 2015, 2015.

[39] X. Li, "A non-dominated sorting particle swarm optimizer for multiobjective optimization," in *Genetic and Evolutionary Computation—GECCO 2003*. Springer, 2003, pp. 198–198.

[40] R. C. Eberhart and Y. Shi, "Comparing inertia weights and constriction factors in particle swarm optimization," in *Evolutionary Computation, 2000. Proceedings of the 2000 Congress on*, vol. 1. IEEE, 2000, pp. 84–88.

[41] S. Mostaghim and J. Teich, "Covering pareto-optimal fronts by subswarms in multi-objective particle swarm optimization," in *Evolutionary Computation, 2004. CEC2004. Congress on*, vol. 2. IEEE, 2004, pp. 1404–1411.

[42] G. T. Pulido and C. A. C. Coello, "Using clustering techniques to improve the performance of a multi-objective particle swarm optimizer," in *Genetic and Evolutionary Computation Conference*. Springer, 2004, pp. 225–237.

[43] D. Saha, S. Banerjee, and N. D. Jana, "Multi-objective particle swarm optimization based on adaptive mutation," in *Proceedings of the 2015 Third International Conference on Computer, Communication, Control and Information Technology (C3IT)*, Feb 2015, pp. 1–5.

[44] X. Zheng and H. Liu, "A hybrid vertical mutation and self-adaptation based mopso," *Computers and Mathematics with Applications*, vol. 57, no. 11, pp. 2030 – 2038, 2009, proceedings of the International Conference. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0898122108005154

[45] K. Deb, M. Abouhawwash, and J. Dutta, *An Optimality Theory Based Proximity Measure for Evolutionary Multi-Objective and Many-Objective Optimization*. Cham: Springer International Publishing, 2015, pp. 18–33. [Online]. Available: https://doi.org/10.1007/978-3-319-15892-1_2

[46] K. Deb, M. Abouhawwash, and H. Seada, "A computationally fast convergence measure and implementation for single-, multiple-, and many-objective optimization," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 1, no. 4, pp. 280–293, Aug 2017.

[47] K. Deb, *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley & Sons, 2001.

[48] V. Chankong and Y. Y. Haimes, *Multiobjective Decision Making: Theory and Methodology*. New York: Dover Publications, 1983.

[49] K. V. Price, R. M. Storn, and J. A. Lampinen, *Differential Evolution A Practical Approach to Global Optimization*, ser. Natural Computing Series, G. Rozenberg, T. Bäck, A. E. Eiben, J. N. Kok, and H. P. Spaink, Eds. Berlin, Germany: Springer-Verlag, 2005. [Online]. Available: http://www.springer.com/west/home/computer/foundations?SGWID=4-156-22-32104365-0&teaserId=68063&CENTER_ID=69103

[50] K. V. Price, *Eliminating Drift Bias from the Differential Evolution Algorithm*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 33–88. [Online]. Available: https://doi.org/10.1007/978-3-540-68830-3_2

[51] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," *ACM SIGGRAPH computer graphics*, vol. 21, no. 4, pp. 25–34, 1987.

[52] J. Bishop, "Stochastic searching networks," in *Artificial Neural Networks, 1989., First IEE International Conference on (Conf. Publ. No. 313)*. IET, 1989, pp. 329–331.

[53] J. Kennedy and R. E. P. S. Optimization, "Ieee int," in *Conf. on Neural Networks*, vol. 4, 1995.

[54] M. Lechuga and E. Coello, "Mopso: A proposal for multiple objective particle swarm optimization," in *Proceedings of the 2002 Congress on Evolutionary Computation, Part of the 2002 IEEE World Congress on Computational Intelligence*, 2002, pp. 2051–11 056.

[55] B. Li, J. Li, K. Tang, and X. Yao, "Many-objective evolutionary algorithms: A survey," *ACM Comput. Surv.*, vol. 48, no. 1, pp. 13:1–13:35, Sep. 2015. [Online]. Available: http://doi.acm.org/10.1145/2792984

[56] H. Sato, H. E. Aguirre, and K. Tanaka, *Controlling Dominance Area of Solutions and Its Impact on the Performance of MOEAs*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 5–20. [Online]. Available: https://doi.org/10.1007/978-3-540-70928-2_5

[57] A. B. de Carvalho and A. Pozo, "Measuring the convergence and diversity of cdas multi-objective particle swarm optimization algorithms: A study of many-objective problems," *Neurocomput.*, vol. 75, no. 1, pp. 43–51, Jan. 2012. [Online]. Available: http://dx.doi.org/10.1016/j.neucom.2011.03.053

[58] A. Nebro, J. Durillo, J. García-Nieto, C. Coello Coello, F. Luna, and E. Alba, "Smpso: A new pso-based metaheuristic for multi-objective optimization," in *2009 IEEE Symposium on Computational Intelligence in Multicriteria Decision-Making (MCDM 2009)*. IEEE Press, 2009, pp. 66–73.

[59] S. Mostaghim and J. Teich, "Strategies for finding good local guides in multi-objective particle swarm optimization (mopso)," in *Swarm Intelligence Symposium, 2003. SIS '03. Proceedings of the 2003 IEEE*, April 2003, pp. 26–33.

[60] E. Figueiredo, T. Ludermir, and C. Bastos-Filho, "Many objective particle swarm optimization," *Inf. Sci.*, vol. 374, no. C, pp. 115–134, Dec. 2016. [Online]. Available: https://doi.org/10.1016/j.ins.2016.09.026

[61] A. Britto and A. Pozo, "Using reference points to update the archive of mopso algorithms in many-objective optimization," *Neurocomputing*, vol. 127, no. Complete, pp. 78–87, 2014.

[62] J. Dutta, K. Deb, R. Tulshyan, and R. Arora, "Approximate kkt points and a proximity measure for termination," *Journal of Global Optimization*, vol. 56, no. 4, pp. 1463–1499, Aug 2013. [Online]. Available: https://doi.org/10.1007/s10898-012-9920-5

[63] M. Abouhawwash, H. Seada, and K. Deb, "Karush-kuhn-tucker optimality based local search for enhanced convergence of evolutionary multi-criterion optimization methods," Tech. Rep., 2015.

[64] A. P. Wierzbicki, "Basic properties of scalarizing functionals for multiobjective optimization," *Optimization*, vol. 8, no. 1, pp. 55–60, 1977.

[65] ——, "A methodological guide to multiobjective optimization," in *Optimization Techniques*. Springer, 1980, pp. 99–123.

[66] ——, "The use of reference objectives in multiobjective optimization," in *Multiple criteria decision making theory and application*. Springer, 1980, pp. 468–486.

[67] ——, "A mathematical basis for satisficing decision making," in *Organizations: Multiple agents with multiple criteria*. Springer, 1981, pp. 465–486.

[68] ——, "A mathematical basis for satisficing decision making," *Mathematical modelling*, vol. 3, no. 5, pp. 391–405, 1982.

[69] ——, "On the completeness and constructiveness of parametric characterizations to vector optimization problems," *OR spectrum*, vol. 8, no. 2, pp. 73–87, 1986.

[70] ——, "A methodological approach to comparing parametric characterizations of efficient solutions," in *Large-scale modelling and interactive decision analysis*. Springer, 1986, pp. 27–45.

[71] S. Huband, P. Hingston, L. Barone, and L. While, "A review of multiobjective test problems and a scalable test problem toolkit," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 5, pp. 477–506, 2006.

[72] D. A. Van Veldhuizen, "Multiobjective evolutionary algorithms: Classifications, analyses, and new innovations," Ph.D. dissertation, Wright Patterson AFB, OH, USA, 1999, aAI9928483.

[73] K. Deb and A. Srinivasan, "Innovization: Innovating design principles through optimization," in *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO '06. New York, NY, USA: ACM, 2006, pp. 1629–1636. [Online]. Available: http://doi.acm.org/10.1145/1143997.1144266

[74] E. H. Mamdani and S. Assilian, "An experiment in linguistic synthesis with a fuzzy logic controller," *International journal of man-machine studies*, vol. 7, no. 1, pp. 1–13, 1975.

[75] E. H. Mamdani, "Application of fuzzy algorithms for control of simple dynamic plant," in *Proceedings of the institution of electrical engineers*, vol. 121, no. 12. IET, 1974, pp. 1585–1588.

[76] M. Sugeno, "An introductory survey of fuzzy control," *Information sciences*, vol. 36, no. 1-2, pp. 59–83, 1985.

[77] S. Kukkonen and J. Lampinen, "Gde3: the third evolution step of generalized differential evolution," in *2005 IEEE Congress on Evolutionary Computation*, vol. 1, Sept 2005, pp. 443–450 Vol.1.

[78] A. Nebro, J. Durillo, and C. A. C. C. C.A. Coello Coello, "Analysis of leader selection strategies in a multi-objective particle swarm optimizer," in *CEC 2013*, 2013.

[79] A. J. Nebro, J. J. Durillo, F. Luna, B. Dorronsoro, and E. Alba, "A cellular genetic algorithm for multiobjective optimization," in *Proceedings of the Workshop on Nature Inspired Cooperative Strategies for Optimization (NICSO 2006)*, D. A. Pelta and N. Krasnogor, Eds., Granada, Spain, 2006, pp. 25–36.

[80] ——, "Design issues in a multiobjective cellular genetic algorithm," in *Evolutionary Multi-Criterion Optimization. 4th International Conference, EMO 2007*, ser. Lecture Notes in

Computer Science, S. Obayashi, K. Deb, C. Poloni, T. Hiroyasu, and T. Murata, Eds., vol. 4403. Springer, 2007, pp. 126–140.

[81] Y. Yuan, H. Xu, and B. Wang, "An improved nsga-iii procedure for evolutionary many-objective optimization," in *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO '14. New York, NY, USA: ACM, 2014, pp. 661–668. [Online]. Available: http://doi.acm.org/10.1145/2576768.2598342

[82] J. Durillo, A. Nebro, and E. Alba, "The jmetal framework for multi-objective optimization: Design and architecture," in *CEC 2010*, Barcelona, Spain, July 2010, pp. 4138–4325.

[83] J. J. Durillo and A. J. Nebro, "jmetal: A java framework for multi-objective optimization," *Advances in Engineering Software*, vol. 42, pp. 760–771, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0965997811001219

[84] A. J. Nebro, J. J. Durillo, and M. Vergne, "Redesigning the jmetal multi-objective optimization framework," in *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO Companion '15. New York, NY, USA: ACM, 2015, pp. 1093–1100. [Online]. Available: http://doi.acm.org/10.1145/2739482.2768462

[85] S. Bleuler, M. Laumanns, L. Thiele, and E. Zitzler, "PISA — a platform and programming language independent interface for search algorithms," in *Evolutionary Multi-Criterion Optimization (EMO 2003)*, ser. Lecture Notes in Computer Science, C. M. Fonseca, P. J. Fleming, E. Zitzler, K. Deb, and L. Thiele, Eds. Berlin: Springer, 2003, pp. 494 – 508.

[86] L. While, L. Bradstreet, and L. Barone, "A fast way of calculating exact hypervolumes," *IEEE Transactions on Evolutionary Computation*, vol. 16, no. 1, pp. 86–95, Feb 2012.

[87] H. B. Mann and D. R. Whitney, "On a test of whether one of two random variables is stochastically larger than the other," *The annals of mathematical statistics*, pp. 50–60, 1947.

[88] F. Wilcoxon, “Individual comparisons by ranking methods,” *Biometrics bulletin*, vol. 1, no. 6, pp. 80–83, 1945.

VITA

Amit A. Kulkarni

Old Dominion University

Department of Mechanical and Aerospace Engineering

238 Kaufman Hall

Norfolk, VA 23508

Phone: (757) 683-6363

Fax: (757) 683-5344

Email: kulkarniamit2@gmail.com

Education

- M.S. Mechanical Engineering, Old Dominion University, 2011.
- B.E. Mechanical Engineering, Shivaji University, India, 2006.

Publications

- Kotinis, M. and Kulkarni, A., 2012. Multi-objective shape optimization of transonic airfoil sections using swarm intelligence and surrogate models. *Structural and Multidisciplinary Optimization*, 45(5), pp.747-758.