Bee Colony Optimization for Single and Multi-Objective Numerical Optimization

Khaled Saady Ahmed Elhalawany

Submitted to the
Institute of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Master of Science
in
Computer Engineering

Eastern Mediterranean University
January 2019
Gazimağusa, North Cyprus

Approval of the Institute of Graduate Studies and Research

Assoc. Prof. Dr. Ali Hakan Ulusoy
Acting Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science in Computer Engineering.

Prof. Dr. Işık Aybay
Chair, Department of Computer Engineering

We certify that we have read this thesis and that in our opinion it is fully adequate in scope and quality as a thesis for the degree of Master of Science in Computer Engineering.

Assoc. Prof. Dr. Adnan Acan
Supervisor

Examining Committee

1. Assoc. Prof. Dr. Adnan Acan
2. Asst. Prof. Dr. Mehtap Köse Ulukök
3. Asst. Prof. Dr. Ahmet Ünveren

ABSTRACT

One common feature of natural systems is the ability for the dynamic interaction between the most basic individual organisms to produce systems capable of performing complex tasks. This thesis introduces a novel population based search algorithm known as Bees Algorithm (BA), that simulates the manner in which swarms of honey bees forage for food. This algorithm involves a collection of a neighborhood and stochastic search and is used in both functional and combinatorial optimization. After describing the algorithm in detail, this thesis attempts to elucidate the robustness and efficiency of the algorithm based on the outcomes for a library of complex numerical optimization problems.

The Artificial Bee Colony (ABC) is a swarm based on meta-heuristic algorithm used to optimize numerical optimization problems and provide accurate solutions. The use of the term ‘meta-heuristic’ here refers to the capacity of the algorithm to provide optimal solutions even in cases on imperfect or incomplete information. Bee colonies scour many sources of food to determine the best source based on a number of parameters, such as time, the amount and quality of nectar, etc. In a similar manner, models that use the ABC algorithm are composed of three components: Unemployed bees, Employed bees, and Food sources (Fitness). The employed bees are responsible for finding affluent sources of food close to the hive. In the algorithm, artificial forager bees acting as environmental agents search for rich food sources. The process of applying the algorithm begins with transforming the given optimization problem into one of examining the best parameter vectors, from a population of vectors, to minimize the objective function. Starting with population of preliminary solution vectors, potential solutions are enhanced using certain strategies.

This thesis work introduces a Bee Colony Optimization Algorithm and examines its feasibility based on the results of CEC'05 and CEC'17 expensive benchmark problems for single objective optimization problems , and used CEC'09 and CEC'18 expensive benchmark problem for Multi-objective optimization. The methods used in our studies are compared to different well-knows methods proposed in the related literature was conducted. The final ranking of all test problems indicate that BCO was always among the top best algorithms that were used for the same purpose.

Keywords: Multi-agent systems, Meta-heuristic algorithms, Multi-objective optimization, Swarm intelligence, Pareto optimality

ÖZ

Doğal sistemlerin ortak özelliklerinden biri temel bireysel organizmalar arasındaki dinamik etkileşim yeteneği ile karmaşık görevleri yerine getirebilmeleridir. Bu tez arı algoritması (BA) olarak bilinen bir popülasyonun tabanlı arama algoritmasını tanıtır ve bal arıları kolonisinin yem arama sürecindeki davranışlarını benzetimler. Arı algoritmasının daha basit sürümü olarak, bir komşuluk kümesi çerçevesinde ve işlevsel eniyileme problemlerinin çözümüne yönelik stokastik arama mekanizmaları içeririr. Algoritma ayrıntılarını detaylı olarak açıklayan bu tez çalışması, güvenilirlik ve verimlilik konularına bir dizi karmaşık problem üzrinden yapılan deneylerin sonuçlarına dayalı olarak algoritmanın etkinliğini aydınlatmaya çalışır.

Yapay arı kolonisi (ABC) algoritması, sayısal eniyileme problemlerine zaman, maliyet, hesaplama karmaşıklığı ve saklama ölçütleri gözetlenerek hassas çözümler üretmek üzere önerilmiş bir popülasyon tabanlı sezgisel yöntemdir. ‘Meta sezgisel' terimi önerilen bir algoritmanın problemin belirsizlik içerdiği durumlarda bile en uygun çözümleri sunmak için algoritma kapasitesini ifade eder. Arı kolonileri nektar kalitesini zaman ve miktar parameterlerine göre birden fazla kaynağı değerlendirerek belirler. Arı kolonilerine benzer şekilde, ABC algoritmasının da kullandığı eniyileme modelleri üç bileşenden oluşur: işsiz arılar, işçi arılar ve gıda kaynakları (Fitness). İşçi arılar kovana yakın zengin yiyecek kaynaklarını bulmaktan sorumludur. Algoritmada, yapay yiyecek-arayıcı arılar çevrede bulunan zengin yiyecek kaynaklarını aramak amacıyla hareket ederler.. Algoritma süreci ilk olarak, verilen eniyileme problemini vektörel gösterim temelinde modelleyerek bir amaç işlevine dnüştürmekle başlar ve bu amaç işlevinin parametrelerini değiştirerek işlev değerini en aza indirmeyi hedefler. Yapay arılar, eniyileme sürecinde kullandıkları arama stratejileri sonucunda başlangıçtaki popülasyonu içerisinde amaç işlevini eniyileyen vektörlerin de bulunduğu daha kaliteli bir popülasyona dönüştürür.

Bu tez çalışması arı kolonisi eniyileme algoritmasını sunar ve bu algoritmanın uygulanabilirliğini çok iyi bilinen ve yaygın kullanılan kıyaslama problemlerini kullanarak inceler. Bu anlamda tek amaçlı eniyime için CEC’05 ve CEC’07 kıyaslama problem kümeleri, çok amaçlı eniyileme için ise CEC’09 ve CEC’18 kıyaslama problem kümeleri kullanılmıştır. Bu kıyaslama problemeleri için elde edilen deneysel sonuçlar aynı problemler üzerinde sınanan diğer güçlü yöntemlerin sonuçlarıyla karşılaştırılarak detaylı analizler yapılmıştır. Bu analizlere göre, önerilen arı algoritması tüm kıyaslama problemleri için en iyi yöntemler arasında yer almıştır.

Anahtar kelimeler: Çoklu ajan sistemleri, Meta-sezgisel algoritmalar, Tek ve çok amaçlı işlevler, Eniyileme, Pareto eniyileme

ACKNOWLEDGMENT

This thesis would not have been possible without the inspiration and support of a number of wonderful individuals — my thanks and appreciation to all of them for being part of this journey and making this thesis possible. I owe my deepest gratitude to my supervisor Assoc.Prof.Dr.Adnan Acan. Without his enthusiasm, encouragement, support and continuous optimism this thesis would hardly have been completed. I, also, would like to thank the members of the jury, Asst. Prof. Dr. Ahmet Ünveren , Asst. Prof. Dr. Mehtap Köse Ulukök and Assoc.Prof.Dr.Muhammed Salamah for their reviews and comments for the improvement of this thesis. Special gratitude to Asst. Prof. Dr. Ahmet Ünveren for his support and positive energy that he provided me during all my time in the EMU.

My deep and sincere gratitude to my family for their continuous and unparalleled love, help and support. Thanks to my mom's prayers. I am forever indebted to my parents for giving me the opportunities and experiences that have made me who I am. They selflessly encouraged me to explore new directions in life and seek my own destiny. This journey would not have been possible if not for them, and I dedicate this milestone to them.

It is a pleasure to thank my friends Dr.Basmah Anber and Abdallah Alaraj for the wonderful times we shared. In addition, I would like to thank all my friends in Famagusta who gave me the necessary distractions from my research and made my stay in cyprus memorable.

Finally, My deepest gratitude to all the thoughtful wishes of my old friends, each message and call was deeply appreciated. I would also like to thank all those friends that accompanied and helped me in the pursuit of my Master's degree.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ABC	Artificial Bee Colony
ACO	Ant Colony Optimization
BA	Bees Algorithm
BCO	Bee Colony Optimization
DE	Deferential Evolution
DM	Decision Marker
EA	Evolutionary Algorithm
Fobj	Objective Function
GA	Genetic Algorithms
Gbest	Global Best
MOABC	Multi-Objective Artificial Bee Colony
MOEAs	Multi-Objective Evolutionary Algorithms
MOO	Multi-Objective Optimization
NSGAII	Non-Dominated Sorting Genetic Algorithm
Pbest	Personal Best
Pc	Crossover Probability
Pm	Mutation Probability
SA	Simulated Annealing
TS	Tabu Search

Chapter 1

INTRODUCTION

The classical optimization techniques are useful to find the optimum solution or unconstrained minima or maxima of continuous and differentiable functions. Such types of techniques are analytical in nature and often stuck of locally optimal solutions. Concerted research efforts have been made recently in order to invent novel optimization techniques for solving real life problems, which have the attributes of memory update and population-based search solutions. Presently, general-purpose optimization techniques such as Simulated Annealing, and Genetic Algorithms, have become standard optimization techniques. The popularity of such models is due to the capacity of biological systems to adjust themselves to their constantly changing environments in an efficient manner. Examples of such nature-inspired algorithms include: evolutionary computation, particle swarm optimization, neural networks, bacteria foraging algorithm, immune systems, bee colony and ant colony optimization.

Swarm behavior is a common feature of different colonies of social insects (termites, antes, wasps, bees). The main features of such behavior include division of labor, autonomy, and self-organization.

Swarm Intelligence [1, 2] is a branch of Artificial Intelligence, whose primary focus is the study of individual actions within different kinds of decentralized systems.

Researchers try to apply as many features of swarm intelligence in natural settings to the creation of Swarm Intelligence models and techniques.

1.1 Metaheuristics

The majority of engineering applications share the common challenge of how to solve optimization problems. These types of problems are solved using optimization algorithms like metaheuristics, which are a specific type of optimization algorithm inspired by nature. There are two kinds of metaheuristics: trajectory-based, which provide a single solution, and population-based, which provide a population of solutions. Metaheuristics utilize certain forms of stochastic optimization, which include algorithm sets that find the near-global or global optimal solution to a problem using random selection. As such, they are used in solving a many of optimization problems [3].

The more common types of trajectory-based Metaheuristics include Tabu Search [7] ,Great Deluge Algorithm [5] and Simulated annealing [4], and the types of population based Metaheuristics include the Genetic algorithm [8], Artificial bee colony [10] ant colony optimization [9] and Differential Evolution [11, 12].

1.2 Genetic Algorithms (GA)

Genetic algorithms (GAs) are search and optimization algorithms whose development was inspired by basic of natural development.” John Holland” (1975) is credited with developing the first algorithmic and computational description of GAs [8, 13, 14]. GAs function in relation to a population of potential solutions in which the individual solutions are known as chromosomes. Each chromosome’s content is taken to be the genotype of the relevant solution, while its phenotype or fitness indicates to the evaluation of the primary objective function. GAs begins, first, by randomly initializing a population of solutions, which is consecutively improved over a number of generations.

The individual chromosomes in each generation are modified using three kind of genetic operators: ‘natural selection’, ‘mutation’ and ‘crossover’. The natural selection operator is used in selecting the individual that is subjected to the crossover operator from the current population. It is a stochastic operator with a preference for individuals with high fitness levels whose genetic characteristics are then passed on to future generations. The crossover operator on the other hand, functions by mixing the genetic characteristics (also known as allelic values) of individuals to produce offspring based on the principle that their fitness value should at least be higher than that of their respective parents. As a type of intensification operator, the crossover does not change the gene content of population by adding any new genetic information, so this task is completed by the ‘mutation’ operator instead, which randomly specify allelic values to the relevant domain to the genetic location. A type of diversification operator, the application of mutation typically includes a small probability. Old populations are replaced through the generation of new offspring populations, which terminates upon the satisfaction of predetermined termination criteria. Algorithm 1.1 provides an algorithmic description of GAs, while their problem-specific representational issues and implementation details are contained in [15].

```
Algorithm 1.1. Genetic Algorithms (Pop, Pc, Pm),
  1. Iteration=1;
  2. Pop=Initial Population;
  3. Fitness=$F_{obj}$(Pop);
  4. Best_Solution=Best-fitness chromosome within the Pop;
  5. Termination_Cond=FALSE;
  6. While not(Termination_Cond),
       i.    Mating_Pool=Selection(Pop);
       ii.   Offspring=Crossover(Pc, Mating_Pool);
       iii.  New_Pop=Mutation(Pm, Offspring);
       iv.   New_Fitness=$F_{obj}$(New_Pop);
       v.    Update the Best_Solution;
       vi.   Pop=New_Pop;
       vii.  Fitness=new_Fitness;
       viii. Iteration= Iteration+1;
       ix.   Check(Termination_Cond);
  7. End While.
  8. Return Best_Solution found so far.
```

Figure 1: Genetic Algorithms [15]

1.3 Characteristics of the Proposed Bees Algorithm

In this section certain key characteristics of the proposed Bees Algorithm (BA) will be discussed in detail.

1.3.1 Neighborhood Search

Not least in the case of the BA, Neighborhood search is fundamental in all evolutionary algorithms. In BA, the process of searching a site is akin to that of the foraging field exploitation of honey bee colonies in nature.

As previously described; when a scouting bee discovers a 'fruitful' foraging field, it reports the location back to the hive to enlist more bees to that area. This practice is both useful and crucial in sustaining the colony. In the same vein, this productive process may well be an effective process for problems relating to optimization in engineering. The 'wriggle dance' back amongst the fellow bees is integral to the process of harvesting, where the recruit bees engage in a monitoring phase leading to decision making for the ultimate purpose of gathering the crop.

In the case of BA, this monitoring exercise can be related to and used as a neighborhood search. Basically, when the scout discovers a good field (good solution), it is advertised to more bees. In so doing, the recruited bees fly to the lode location, harvest the nectar and return to their hive.

Subject to the quality at that source and of the nectar quality, the location can be re-advertised by bees already aware of the location. In the proposed BA, this behavioral practice has been used as a neighborhood search.

As described above; from each foraging site (neighborhood site) only one bee is selected and must be equipped with the best solution information for that respective field. It is with this in mind, the algorithm may create certain solutions, which are related to the ones previous.

Neighborhood search is depend on a random dispersal of bees in a specific range (patch size). For each selected site, bees are distributed at random to seek a good solution (fitness).

Simultaneously, during the process of harvesting, additional elements should also be undertaken for the purposes of increased efficiency, ideally the number of recruited bees on that neighborhood patch and the patch size itself. Management of recruited bee numbers targeting selected sites should be properly defined.

Number of function evaluations will be increased or reduced respectively, depends on the neighborhood range. If the range can be organized sufficiently, then the number of bee recruits will be dependent upon the intricacy of a solution range.

1.3.2 Site Selection

There are two method have been implemented to use in site selection best site and probabilistic selection. ‘Roulette wheel’ method has been utilized in probabilistic selection and the site that have good fitness have more chance to selected, but in best site selection, the best site according to a good fitness will be selected.

Chapter 2

LITERATURE REVIEW

2.1 Artificial Bee Colony

The artificial bee colony (ABC) optimization technique is a member of the set of swarm based algorithm optimization techniques presently available. In simple terms, ABC is a meta heuristic technique modeled based on the lifestyle of bees. Bee colony optimization techniques include three type of bees: employed bees, scout bees and onlooker bees.

The Artificial Bee Colony is able to overcome limitations on the applicability of optimization techniques through the application of information share models. These approaches all have certain agents who simultaneously explore the solution space.

These agents (artificial bees) attempt to solve a given problem with incomplete information in all of the approaches under consideration, which are also not subject to any form of global control. Artificial bees are modeled based on cooperation, which serves to improve their efficiency and allow them satisfy objectives they otherwise could not achieve through individual action. The resulting algorithms represent algorithmic frameworks applicable to different type of optimization problems.

The food source in the ABC is used to represent a candidate solution for optimization i.e. the amount of nectar denotes the fineness ‘quality’ of the solution that food source represents. The fineness ‘quality’ of the prospective optimal solution raises parallel to the amount of nectar. Each search cycle in the ABC algorithm is split into three stages: first, deploying employed bees to food and to determine the amount of their nectar content; second, the selection of solution ‘food sources’ by onlookers bees predicted on information received from the employed bees about the amount of nectar in the foods; and third, mobilizing the scout bees and sending them out to possible food sources.

The ability of BCO to help solve non-standard combinatorial optimization problems, such as those with inaccurate data or including multiple-criteria optimization, is well documented. The utilization of BCO in such cases requires it to be hybridized using suitable methods.

The primary aim of this research is the development of swarm-based optimization algorithms motivated by the honey-bees behavior. These algorithms are intended to solve complex optimization problems more efficiently.

The main research objectives include:

• Developing an original intelligent optimization method premised on the swarm food-foraging behaviors of bees that is also applicable to industrial problems.

• Enhancing the search procedure used by the algorithm to improve its performance in combinatorial domains.

2.1.1 The ABC Algorithm Used for Unconstrained Optimization Problems

In ABC algorithm [16, 17], the colony of artificial bees consists of three groups of bees: employed bees, onlookers and scouts. First half of the colony consists of the employed artificial bees and the second half includes the onlookers. For every food source, there is only one employed bee. In other words, the number of employed bees is equal to the number of food sources around the hive. The employed bee whose the food source has been abandoned by the bees becomes a scout.

In ABC algorithm, the position of a food source represents a possible solution to the optimization problem and the nectar amount of a food source corresponds to the quality (fitness) of the associated solution. The number of the employed bees or the onlooker bees is equal to the number of solutions in the population. At the first step, the ABC generates a randomly distributed initial population P(G=0) of SN solutions (food source positions), where SN denotes the size of population. Each solution xi(i=1,2, ..., S N)isaD-dimensional vector. Here, D is the number of optimization parameters. After initialization, the population of the positions (solutions) is subjected to repeated cycles, C=1,2, ..., M CN ,of the search processes of the employed bees, the onlooker bees and scout bees. An employed bee produces a modification on the position (solution) in her memory depending on the local information (visual information) and tests the nectar amount (fitness value) of the new source (new solution). Provided that the nectar amount of the new one is higher than that of the previous one, the bee memorizes the new position and forgets the old one. Otherwise she keeps the position of the previous one in her memory. After all employed bees complete the search process, they share the nectar information of the food sources and their position information with the onlooker bees on the dance area.

An onlooker bee evaluates the nectar information taken from all employed bees and chooses a food source with a probability related to its nectar amount. As in the case of the employed bee, she produces a modification on the position in her memory and checks the nectar amount of the candidate source. Providing that its nectar is higher than that of the previous one, the bee memorizes the new position and forgets the old one.

An artificial onlooker bee chooses a food source depending on the probability value associated with that food source, pi , calculated by the following expression:

$$pi = \frac{fit_i}{\sum_{n=1}^{SN} fit_n}$$

Where fit_i is the fitness value of the solution i which is proportional to the nectar amount of the food source in the position i and SN is the number of food sources which is equal to the number of employed bees (BN).

In order to produce a candidate food position from the old one in memory, the ABC uses the following expression:

$$v_{ij} = x_{ij} + \varphi_{ij}(x_{ij} - x_{kj})$$

Where $k \in \{1,2,\dots,SN\}$ and $j \in \{1,2,\dots,D\}$ are randomly chosen indexes .Although *k* is determined randomly, it has to be different from i .φ_{ij} is a random number between [-1, 1]. It controls the production of neighbor food sources around $x_{i,j}$ and represents the comparison of two food positions visually by a bee. As the difference between the parameters of the $x_{i,j}$ and $x_{k,j}$ decreases, the perturbation on the position $x_{i,j}$ gets decrease, too.

Thus, as the search approaches to the optimum solution in the search space, the step length is adaptively reduced. If a parameter value produced by this operation exceeds its predetermined limit, the parameter can be set to an acceptable value. In this work, the value of the parameter exceeding its limit is set to its limit value. The food source of which the nectar is abandoned by the bees is replaced with a new food source by the scouts. In ABC, this is simulated by producing a position randomly and replacing it with the abandoned one. In ABC, providing that a position can not be improved further through a predetermined number of cycles, then that food source is assumed to be abandoned. The value of predetermined number of cycles is an important control parameter of the ABC algorithm, which is called “limit” for abandonment. Assume that the abandoned source is x_i and $j \in \{1, 2, ..., D\}$, then the scout discovers a new food source to be replaced with x_i.

$$x_i^j = x_{min}^j + rand(0,1)(x_{max}^j - x_{min}^j)$$

Pseudo-code of the ABC algorithm:

```
1: Initialize the population of solutions xi,j .i=1,…,SN,J=1,…D
2: Evaluate the population
3: cycle=1
4: repeat
5: Produce new solutions $\upsilon_{i,j}$ for the employed bees and evaluate them
6: Apply the greedy selection process
7: Calculate the probability values $P_{i,j}$ for the solutions $x_{i,j}$
8: Produce the new solutions $\upsilon_{i,j}$ for the onlookers from the solutions $x_{i,j}$
   selected depending on $P_{i,j}$ and evaluate them
9: Apply the greedy selection process
10: Determine the abandoned solution for the scout, if exists, and replace it
with a new randomly produced solution $x_{i,j}$ by (3)
11: Memorize the best solution achieved so far
12: cycle=cycle+1
13: until cycle=MCN
```

2.2 Single -Objective Optimization Problems

Optimization can be described as a process through which the most optimal outcome in terms of objective function can be determined. Single optimization problems are characterize by the being of a singular objective function with the goal of either minimizing or maximizing it using the relevant algorithms [18]. The general form of Single Objective Optimization Problems is the maximization or minimization of $f(x)$ subject to $gi(x)$ ≤ 0 i={1,2,3,….,m} and $hj(x)$=0,j={1,2,3,….,m}, $x \in \omega$ whereby $gi(x)$ and $hj(x)$ specify the constraints that must taken into consideration when optimizing $f(x)$. The solution to the problem is to either minimize or maximize $f(x)$ where x is the n-dimensional decision variable array is and ω is the universe for x . The global optimal is identified using the method known as global optimization.

2.3 Bee Colony Optimization for Single Objective Problems

BCO is a meta–heuristic method drawing from nature and developed as an efficient way to solve intricate combinatorial optimization problems. The underlying logic of BCO is to use a multi-agent system (Colony of artificial bees) to find solutions to many combinatorial optimization problems by exploiting the same rules honey bees use in the steps of collecting nectar.

While the artificial bee colony typically contains a lesser number of individual bees, the BCO principles it adheres to are replicated from natural systems. The autonomous artificial bees examine the search space to find the best possible solutions by sharing information and collaborative effort. This information sharing enables they develop a pool of collective knowledge that allows the artificial bees determine which areas are more likely to produce optimal solutions. The artificial bees then gradually and collectively produce or/and develop their solutions. The BCO search is continually repeated until it has satisfied a predefined stopping criterion.

BCO is conducted by a population of B individuals (artificial bees). Each of these individuals is expected to handle one possible solution to the given problem. forward pass and the backward pass the basic steps in BCO algorithm .All of the artificial bees examine the search space in the forward pass. By applying a previous amount of moves that improve the complete/partial solutions, they are able to generate new partial solutions. To illustrate, we assume Bee 1, Bee 2, . . . , Bee B partake in the process of decision-making in n entities. This entity could either be a complete singular solution expected to be enhanced by BCOi in the algorithm’s later phases, or a subset (one or more) of partial solution components according to the constructive version [19].

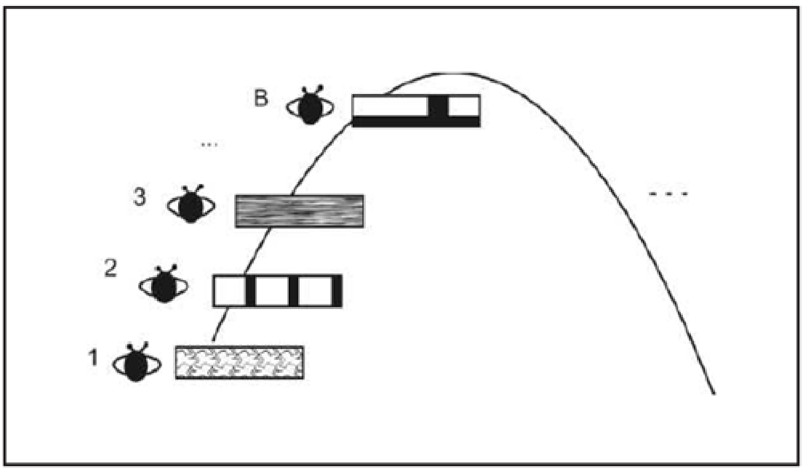


Figure 2.1: Partial /Complete solutions after (n) pass [19]

Figure.2.1 illustrates the possible solutions after (n) forward pass. The rectangles in Figure show the partial/complete solutions related to each bee with different style denoting the different solutions associated to each bee.

The second backward pass phase begins after the new partial/complete solutions have been obtained. This phase involves information sharing between all the artificial bees regarding the fitness (quality) of their individual solutions. In natural settings, honey bees turn back to the hive and use the 'waggle dance' to signal to other bees the number of food they found, as well as its distance from the hive; that is, the overall quality of their discovery. In the search algorithm, this 'waggle dance' announcement takes the form of calculating the objective function values of every partial or complete solution. Following the evaluation of each solution, each artificial bee determines the likelihood of its continued loyalty to its solution.

Bees with relatively superior solutions are in a better position to both retain and promote their respective solutions. In contrast to natural bees, however, artificial bees that remain loyal to their solutions simultaneously function as 'recruiters' – that is, other bees will also consider their solutions. One a bee abandons its solution; it will need to detect one from the advertised solutions. This selection process is based on probability in that the more advertised a solution, the better its chances of being selected for further exploration. As such, all the bees are categorized into one of two

groups during each backward pass: R recruiters and B – R uncommitted bees (see Figure. 2.2). The values for each group change for each backward pass.

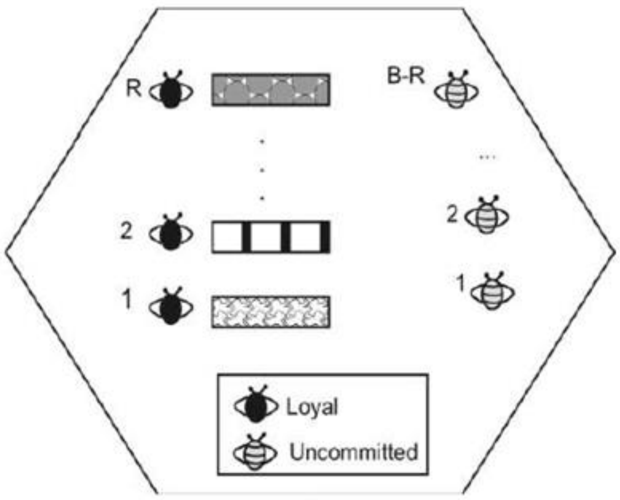


Figure 2.2: Recruiting of uncommitted followers [19]

If we assume that Bee-1 from Figure 2.1 decides to put away its solution after all the generated partial/complete solutions have been compared and joins ‘Bee B’, both ‘Bee-1’ and Bee-B “fly with each other” through the route that produced by the ‘Bee B’ (see Figure.2.3). Simply put, the solution produced by ‘Bee B’ is adopted by ‘Bee 1’ as illustrated by their similar solution-rectangles in Figure.2.3. After this stage, however, both bees are free to individually decide their next steps. Bees 2, 3 from the earlier example retain their generated solutions.

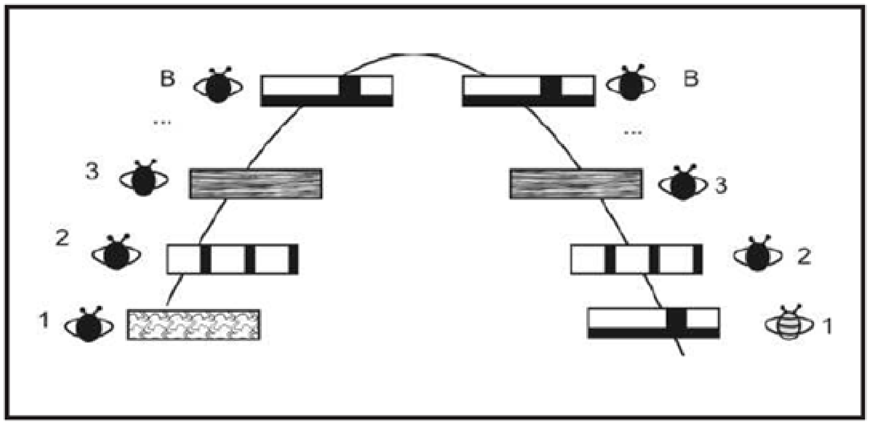


Figure 2.3: Recruiting process in (n) backward pass [19]

Based on the deductive BCO, each bee give a various set of components to the partial solution it generated before in the next forward pass, while the bees in BCOi attempts to improve the quality of their complete solutions by altering some of their components. Figure. 4.2 illustrate this condition after the next forward pass when the bees' solutions have changed as symbolized by new patterns in their associated rectangles.

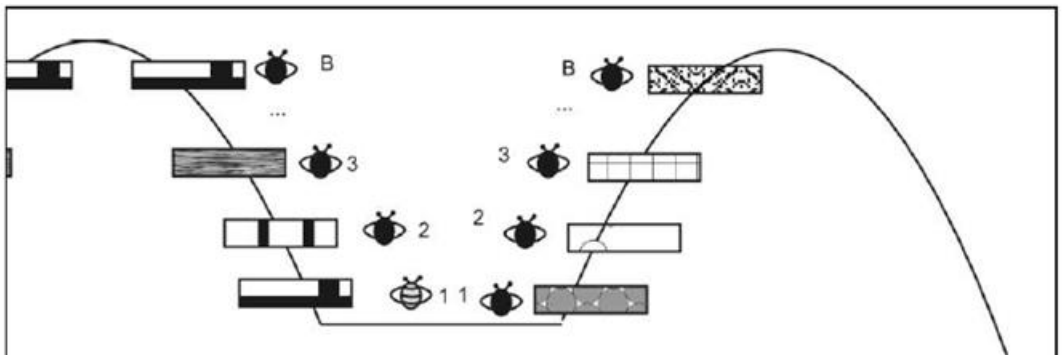


Figure 2.4: Partial / Complete solutions after (n) forward pass [19]

The forward and backward pass stages of the algorithm also have alternating NC times; that is, they cannot begin until each bee is finished generating its solution or preforming NC solution modifications. Earlier versions of the BCO algorithm used the NC parameter to determine the amount of factors that needed to be assigned to the partial solutions in each forward pass. The meaning of "NC" was changed after the development of BCOi in an effort to unify the algorithm description. Recent literature also contains other amendments to BCO parameters [20, 21].

Regardless, the parameter NC denotes how frequently the bees exchange information. The most suitable of all B solutions is selected after the completion of the NC steps. This solution is utilized to enhance the global best solution, thereby completing one BCO iteration. The B solutions are then deleted at this juncture and subsequently followed by the start if a new BCO iteration. The BCO algorithm continues running iterations until the termination condition is satisfied. Examples of such criterion include: maximum allowed CPU time, maximum number of iterations without any advancement in the objective function score, maximum number of iterations, among others. Once the criterion has been the current global best solution is considered final.

The values of the following factor need to be declared before executing the BCO algorithm:

B—Size bee colony (Number of bees)

NC —Number of passes (forward and backward) in each iteration.

2.3.1 Loyalty Decision

Each bee has to decide whether or not to remain loyal to its previous solution at the end of every forward pass. This decision is made by comparing the fitness of its solution to that of the other solutions. The following formula calculates the probability that the bee will remain loyal to its previous solution [18]:

$$P_b^{U+1} = e^{-\frac{Omax-Ob}{u}}, b = 1,2, \ldots\ldots, B \qquad (1)$$

Where:

Omax: The maximum of all solutions to be compared.

Ob: objective functions of solution created by the bee by, and the number of forward passes by u (taking values 1, 2. . . NC).

Based on whether the objective function needs to be maximized or minimized, the normalization can be performed one of two ways. If the objective function value of bee solution is represented by Cb (b =1, 2. . . B), its normalized value in the case of minimization is computed using the following formula [19]:

$$Ob = \frac{Cmax - Cb}{Cmax - Cmin}, b = 1,2, \dots, B \qquad (2)$$

Where the values of solutions associated to minimum and maximum objective function values gotten by all the relevant bees are represented by Cmin and Cmax respectively. Equation (2) illustrates that Ob is larger when the bee's solution is further from the maximal value of all solutions (Cmax) than its normalized value, and vice versa.

Equation (3) is used to evaluate the normalized value of Cb in the case of maximization criterion [19],

$$Ob = \frac{Cb - Cmax}{Cmax - Cmin}, b = 1,2, \dots, B \qquad (3)$$

It is evident from equation (3) that the normalized value Ob is larger when the value of the solution Cb is higher and vice versa. In Equation (1) with a randomly-generated number, the individual artificial bees determine even to continue examining their own solutions or turn into uncommitted follower. If the selected random number is less than the calculated probability then the bee remains loyal to its solution. Conversely, if the calculated probability is less than the random number p_b^{u+1}, then the bee becomes uncommitted.

2.3.2 Recruiting Process

For each 'Uncommitted bee' has to decide which recruiter to monitor based on the fitness of all advertised solutions. The chance that bee solution is chosen by an uncommitted bee is mathematically computed as [19]:

$$Pb = \frac{Ob}{\sum_{k=1}^{R} Ok}, b = 1,2,\dots,R \tag{4}$$

Where

Ok value for the objective function of the advertised solution.

R represents the total number of recruiter’s bees; roulette wheel is used to pair each uncommitted follower to one recruiter.

2.3.3 External Archive

The proposed algorithm works with an internal population and an external archive. It uses a decomposition-based strategy for evolving its working population and uses a domination-based sorting for maintaining the external archive. Information extracted from the external archive is used to generate new solution using genetic operator ,first solution selected from the archive using roulette wheel method and by select one solution from archive and second solution will generate randomly then modified both solution by crossover or mutation .

2.4 Independent Run of BCO Algorithms

Parallelization of BCO in its simplest form presents the independent procedure of important computations on various processors. Speed up the search performed is the aim of this strategy in BCO by divide total of work between different processors .In [22], it was recognize by a reduction of the stopping condition on each processor, the BCO could work in parallel on q processors for runtime=q seconds. The BCO parameters (number of bees B and number of forward/backward passes NC) were the same for all BCO processes executing on various processors in order to ensure a load balance between all processors. The BCO algorithms running on different processors were different in the seeds values. This variant of parallelized BCO was named Distributed BCO (DBCO) [23]. Another way to implement the coarse grained parallelization strategy proposed in [22] was the following: Instead of the stopping criterion, the number of bees could be divided. Namely, if the sequential execution uses B bees for the search, parallel variant executing on q processors would be using B=q bees only. Actually, on each processor, a sequential BCO is running with the reduced number of bees. This variant was referred to as BBCO [19].

2.5 Synchronous Cooperation of BCO Algorithms

More advance way to perceive parallelization is cooperative work of several BCO processes. At certain execution points, all processes share the relevant information that are used to guide further search. This synchronous strategy named Cooperative BCO (CBCO) and proposed in [22]. The communication points were specified in two different ways: fixed and processor dependent. In the first case, the best solution was shared 10 times during the parallel BCO execution. In such a way processors were given more freedom to execute independent part of the search [19].

2.6 Asynchronous Cooperation of BCO Algorithms

To decrease the communication and synchronization overhead during the cooperative execution of different BCO algorithms, in [22] the authors proposed the use of the asynchronous execution method. They implemented this method in two different ways, but under the common name General BCO (GBCO). The first implementation concerned a centrally coordinated knowledge exchange, while the second utilized non-centralized parallelism [19].

The first asynchronous approach proposed in [22] supposed the presence of a central blackboard - to which each processor has access. Improvement of the current best solution is the aim of communication condition. The stopping condition was not reduced and was set to maximum allowed CPU time in order to ensure better load balancing. Non-centralized asynchronous parallel BCO execution supposed the existence of several blackboards so that only a subset of (neighboring) processors may post and access information on the corresponding blackboard.

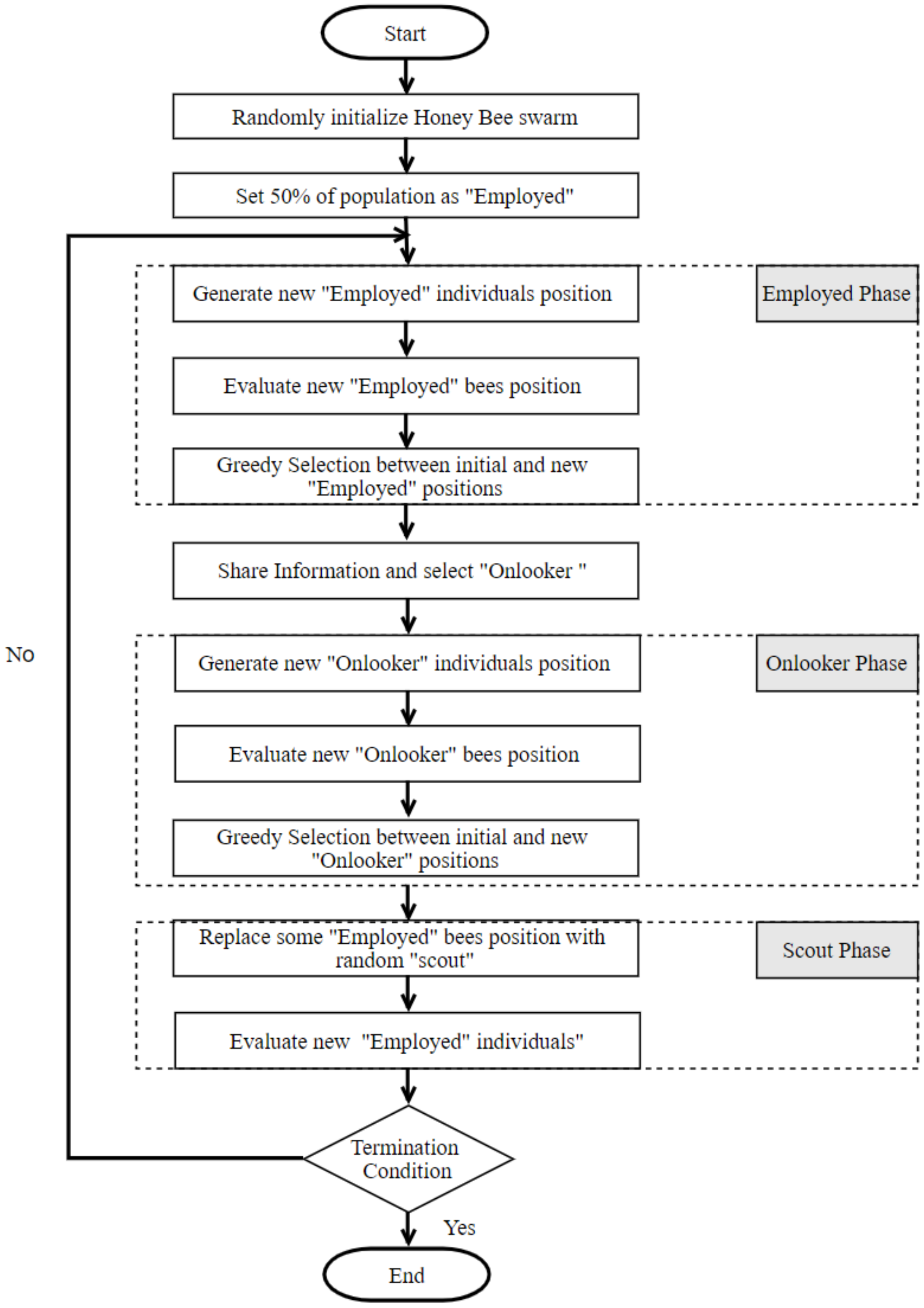

Start
Randomly initialize Honey Bee swarm
Set 50% of population as "Employed"
Generate new "Employed" individuals position
Employed Phase
Evaluate new "Employed" bees position
Greedy Selection between initial and new "Employed" positions
Share Information and select "Onlooker "
No
Generate new "Onlooker" individuals position
Onlooker Phase
Evaluate new "Onlooker" bees position
Greedy Selection between initial and new "Onlooker" positions
Replace some "Employed" bees position with random "scout"
Scout Phase
Evaluate new "Employed" individuals"
Termination Condition
Yes
End

Figure 2.5: Flowchart of BCO Algorithm [24]

2.7 Multi-Objective Optimization Problems

The essential notion behind multi-objective optimization is the existence of a multi-objective problem that has more than one functions that need to be improved (optimized “minimized or maximized”) using the solution x, as well as different constraints to satisfy as seen in Equation 5 [25].

$$\begin{aligned} &\text{Minimize / Maximize } fm(x), m = 1,2,\dots,M; \\ &\quad\text{Subject to } gj(x) \geq 0, j = 1,2,\dots,J; \\ &\qquad hk(x) = 0, k = 1,2,\dots,K; \\ &\qquad X_i^L \leq X_i \leq X_i^U i = 1,2,\dots,N; \end{aligned} \tag{5}$$

X: vector of decision variables: $x = (x1, x2, \dots, xn)\,T$, each of which $xi \in R$ is constrained by the lower and upper bounds x_{Li} and x_{Ui} respectively [26] . These bounds establish the decision range *D* and the *M*: number of objective functions *fm(x)* define a converting from *D* to the objective range Z. This mapping is subjective and occurs between the n-dimensional solution vectors x ∈ D and the m dimensional objective vectors *fm(x)* ∈ *Z* so that each *x* ∈ *D* is linked to a point *y* ∈ *Z* (See Figure 2.6).

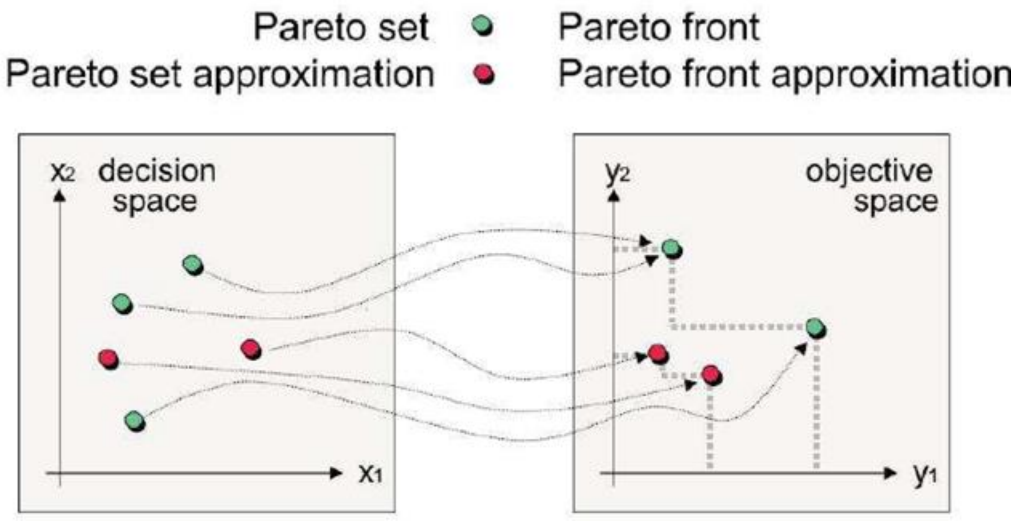


Figure 2.6: Mapping from Decision Space to Objective Space [25]

Equation 5 can also be made to show J inequality and K equality constraints by constraining the problem.

The switch to single-objective optimization problems from multi-objective problems presents a new challenge for how solutions are compared as performance becomes a vector of objective values rather than a single scalar. This issue is addressed by the idea of Pareto dominance, which allows solutions to be compared. . The solution x can be dominating solution when the following conditions have been met:

1) Solution y is not better than solution x in all objectives function.

2) Solution x is better than solution y on at least in one objective function.

Considering all objectives are subject to minimization, this is mathematically written as [25]:

$$fm(x) \leq fm(y) \forall m \wedge \exists i : fi(x) < fi(y) \tag{6}$$

This binary dominance relation is asymmetric, non-reflexive and transitive. Several relations between solutions, however, can still be observed. Table 2.1 outlines some of the more common relations between solutions, their corresponding notations and formal interpretations [26]. The list is arranged based on the level of strictness enforced.

Table 2.1: Solution Relation [25]

Relation	Notation	Interpretation
Strictly dominates	$x \prec\prec y$	$f_m(x) < f_m(y) \forall_m$
Dominates	$x \prec y$	$f_m(x) < f_m(y) \forall_m \wedge \exists_i : f_i(x) < f_i(y)$
Weakly dominates	$x \preccurlyeq y$	$f_m(x) \leq f_m(y) \forall_m$
Incomparable	$x \parallel y$	$\neg(x \leq y) \wedge \neg(y \leq x)$
Indifferent	$x \sim y$	$f_m(x) = f_m(y) \forall_m$

A number of other important definitions can also be derived from the definition of dominance. One important concern during optimization is to locate the non-dominated set of solutions. Non-dominated set of solutions ‘ P` ’ in a set of solutions ‘P’ are the solutions not dominated by any other members of P. Accordingly, the globally Pareto-optimal set is defined as the non-dominated set of the whole usable search range $S \in D$. Denoted by Pareto optimal set, the approximation of this set is the expected goal of multi-objective optimizers. The mapping from the Pareto-optimal set in an objective space denotes the true Pareto-optimal front/true Pareto front as illustrated in Figure 2.6 [25].

2.7.1 Multi-Objective Optimization Using Evolutionary Algorithms

An evolutionary algorithm (EA) can be utilized to carry out a multi-objective optimization. EAs are optimizers that draw inspiration from Darwinian evolution. The solutions to a particular problem in an EA are assumed to be individuals in a population with each individual’s fitness determined by its efficacy at solving said problem. Mating between individuals in a population produces offspring who then compete with their parents for the chance to be added in the next generation. Since only the fittest individuals survive, the general population is improved with each iteration [26]. In a more formal sense, the advantage of EA is that is uses a set of solutions, as opposed to merely enhancing a single solution. This allows for the combination of useful solutions to create new ones. Based on probabilistic operators, an EA is in fact a stochastic meta-heuristic, that is, a method of optimization. Consequently, different executions of the EA can produce different outcomes, in contrast to deterministic algorithms.

The most significant distinction between single- objective and multi-objective EAs (MOEAs) is that while returning to the most optimal solution for a population is relatively simple in single objective optimization due to the implication of a specific order among solutions, MOEAs present an entirely different situation [25]. The greater dimensionality inherent to the objective space makes it impossible to compare all of the individuals in a population to one another since they each represent an optimal compromise between objectives. Put differently, this simply means that the set of solutions produced by a MOEA will most likely be non-dominated. It is therefore the decision-maker's responsibility to determine which solution(s) to realize. A description of the entire process is provided in Figure.2.7.

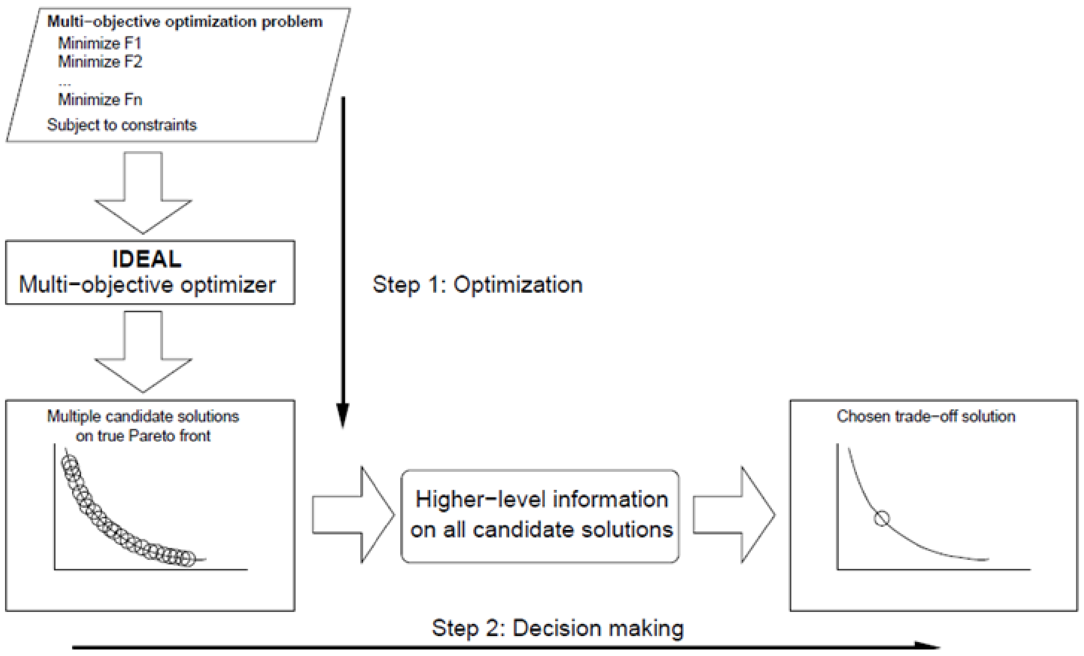

Multi−objective optimization problem
Minimize F1
Minimize F2
...
Minimize Fn
Subject to constraints
IDEAL
Multi−objective optimizer
Step 1: Optimization
Multiple candidate solutions on true Pareto front
Higher−level information on all candidate solutions
Chosen trade−off solution
Step 2: Decision making

Figure 2.7:Multi-Objective Optimization Process [25]

2.7.2 Goals Of Multi-Objective Evolutionary Algorithms

As was noted earlier, the purpose of a MOEA is the estimation of the "Non-dominated solution" of solutions [25]. This goal, however, is often separated into three objectives:

1. Proximity to the real Pareto front set.
2. Evenly-distributed solutions.
3. Well-dispersed solutions.

First, by getting all the solutions as closeness to the Pareto optimal front, we can guarantee they are as optimal as possible. This closeness is calculated as the Euclidian distance in an objective space. Because NP complete combinatorial problems constitute the majority of problems solved using MOEAs, it is impossible to 'guess' the mapping of the decision vector to a suitable point in the objective range. The most dominated or non-dominated solution are selected for survival in the MOEA to help the population reach its Pareto optimal front since such individuals are the most similar to it. Under ideal condition, all of the solutions returned from a MOEA are on the Pareto front [27].

The MOEAs second objective is to occupy as large an area of the Pareto front as can be managed an is rather unique to multi-objective optimization. When the individuals on the Pareto front are evenly distributed, a variable set of exchange between objectives is guaranteed. When the solutions in a set are equidistant from their neighbors, the DM is provided with an survey of the Pareto front that allows for the final selection, which then occurs on the basis of the exchange between objectives described by the population, as shown in Figure 2.6 [25].

The third objective is very much related to the second. A high expansion means a similarly high space between the two extremes for solutions in an objective range, which ensures that the Pareto front is covered. The diversity of the population is usually guaranteed by the application of a crowding or density measure, which punished individuals in close proximity to one another within the objective space [28].

In terms of application, the first objective is undoubtedly the most important as it relates directly to how optimal the returned solutions are. The second objective, although also important, is less so since only a few solutions in the final population are typically investigated further. Lastly, the third objective is virtually unimportant as very few extreme solutions are every applied in reality.

Traditional MOEAs utilize two mechanisms in pursuance of these three objectives. These mechanisms are intended to directly promote the convergence of the Pareto front true as well as a suitable distribution of solutions. The first, elitism, is used to guarantee that solutions in nearness to the true Pareto optimal front will remain in the population all through its evolution, i.e., there can be no reduction in the amount of Pareto optimal set ('non-dominated solutions') in the population.

2.8 Basic Operators of Multi-Objective Evolutionary Algorithms

The operators are iteratively applied until they satisfy some predefined termination criterion, which is typically determined by the amount of function evaluations carried out since this constitutes the bulk of the entire computational effort. This, however, rely on the dimensionality of the test problem M, amount of Bees (size of the population) N, and the number of production(generation) performed T. It is not unusual for a new offspring to be produced per parent so that the M × N × T function evaluations are performed in T generations. The primary operators used in the MOEA are [25]:

• Evaluation.

• Selection.

• Variation.

Evaluation is usually grounded in the dominance relation described above. By assigning each individual a Pareto-rank depend on the number of other individuals they dominate, its intended purpose is to indicate the level of dominance of each individual. This ranking out to be Pareto-compliant and more or less graded, contingent on the method used for the ranking itself. The assignment of the final fitness is incorporated with a second fitness criterion I traditional MOEAs so as to enforce and order on individual quality prior to selection. Evaluation occurs in an objective space and is dependent on the objective functions, which are themselves problem-dependent.

Selection comes in one of two forms ad is based on the fitness attributed to the each individual in the evaluation stage described above. The first, sexual/mating selection is used to define which individuals in a generation will be allowed to mate and produce offspring. This typically random selection can also involve all individuals in a population so as to ensure an equal mating chance. The second, environmental selection occurs after variation and includes the application of the famous concept of survival of the fittest, whereby the next generation exclusively comprises of the best pairs of parents and their offspring. The effect of this is to cut the population's size down to its original value. Both selection types are applied in objective space.

Variation is used to individuals in a population that have been selected as candidates for mating. These individuals are given the opportunity to produce offspring that are essentially variations of themselves. These variations are achieved using either recombination or mutation. While the recombination operator allows the offspring to retain the best parts of multiple parents, mutation causes only relatively minor changes in the offspring. Generally, recombination improves exploration of the search space, whereas mutation improves exploitation. A shared feature of all variation operators is their common application in the decision space.

2.9 Bee Colony Optimization in Multi-Objective Problem

One problem common to scientists and engineers is that of multi-objective optimization, which involves the optimization of problems that have various and often inconsistent objectives. In theory, a multi-objective optimization problem (MOP) cannot have a multi-solution but instead has a group of non-dominated solution (Pareto-optimal solutions). This following continuous MOP is taken into consideration in this paper [29]:

$$\text{Minimize } fm(x) = (f1(x), f2(x), \dots, fm(x)) \qquad (7)$$
$$\text{Subject to } X \in \prod_{i=1}^{N}[Lbi, Ubi]$$

Where the decision space is $\prod_{i=1}^{n}[\text{Lbi}, \text{Ubi}]$, m real valued continuous objective functions $f1, f2, \dots, fm$ are contained in f: $\prod_{i=1}^{n}[\text{Lbi}, \text{Ubi}]$ →Rm, and the objective space is R^m. In contrast to single-objective optimization, MOP solutions exist such that the efficacy of one objective can only be enhanced by sacrificing that of at least one other objective. Consequently, an MOP's solution takes the form of a replacement trade off described as a 'Pareto optimal set'.

The Pareto optimal Set is expressed on the basis of Pareto dominance [30]. If u (u1, u2... um) and v (v1, v2,..., vm) are two vectors in an objective space, v can be said to be dominated by u only if vi ≥ ui for every individual i and at least one instance where ui < vi. If no x is found in the decision range for example" F(x∗)" is dominated by "F(x)", 'x∗' is known a (globally) Pareto optimal solution [31].

Artificial Bee Colony (ABC) algorithm is one of the most recent additions to swarm based search approach. There are three types of bees found in the ABC performing different kinds of functions to ensure the algorithm remains beneficial. Employed bees are deployed to sources of food as part of an attempt at improving them based on neighbor information. Using a greedy method on the information regarding the fitness of solution provided by 'employed bees', each of the onlooker bees decides on a specific food source and attempts to enhance it. Lastly, scout bees search for solution that yet not optimized using a restricted number of iteration in an effort to re-initialize it and eliminate the poor solution. The ABC is especially suited to multi-objective optimization primarily since it discovers good solutions and has a relatively speedier convergence for single-objective optimization.

Main feature of our MOABC are [29]:

1. Provided elitism strategy. In its simple form, employed bees and onlooker bees in ABC algorithm produce solution exclusively using neighbor information. This mechanism, however, can easily trap the entire colony in a poor region since neighbors are concentrated around an optimal point. In the employed bees phase, the elite is considered to be the intermediate solution with the highest value for its crowding distance and is used in producing new food sources. The values for

crowing distance are updated when the whole bee colony has been updated. A new elite is then selected and utilized in the subsequent phase of onlooker bees. Each of this process is repeated until the algorithm is terminated [10, 20]. The exploitation ability of the eMOABC can be enhances using such an elitism strategy since employed and onlooker bees exploit the regions containing elites FN times. The whole colony is consequently pulled towards the least populated zone. The benefits of this strategy include:

(1) The exploitation of more potentially “non-dominated solutions”.

(2) Keep the spread of solutions in the approximated set.

2. Two control parameters that need to be manually configured in the MOABC algorithm can be said to be of particular significance: the *limit* (the Stopping criteria) and *CS* (the number of bees).

2.9.1 External Archive

In contrast to single objective optimization, MOEAs are more likely to keep a group of non-dominated solutions. Due to the lack of preference information in multi-objective optimization, no solution be able to superior to others. Consequently, Bee algorithm utilize an external archive as way to accurately document the pareto optimal set non dominated vectors encountered during the explore method [31,32].

2.9.2 Diversity

The success of MOEA can be attributed to its capability to uncover a group of non-dominated solution (“Pareto optimal solutions”) from one iteration. Evolutionary algorithms need to conduct a multimodal search, which includes a variety of unique potential solutions, as a way to determine a reliable approximation of the Pareto optimal set from a single optimization run. As such, the efficiency of MOEA considerably relies on the availability of a diverse population [33].

2.9.3 Update External Archive

New solutions are consistently assigned to the external archive over the course of the evolution. The decision of new solution remains in the external archive or not is based on a comparison between it and every other pareto optimal set (non-dominated 0solution in the archive, the size of which is limited.

Each individual in our algorithm search for a new solution in each generation. The new solution is allowed into the external archive if it is found to dominate the original individual. Conversely, if the original individual dominates the new solution, it is not permitted into the external archive. If neither of the two solutions dominates the other, one of them is chosen at random to add into external archive, which is updated after each generation. If the number of Pareto optimal set is greater than the pre-determined archive size, crowding distance [35] is then used to delete any extra members.

Chapter 3

METHODOLOGY

3.1 Pseudo-code for BCO in Single Objective Optimization Problems

Artificial bee colonies collectively explore for the new optimal solution to a specific function. The individual bees in the colony each generate a solution to the function using two steps: the ‘forward pass’ and the ‘backward pass’ (both of which collectively constitute one iteration in our proposed algorithm). The ‘forward pass’ is reminiscent of a searcher bee who leaves the hive in search of a solution (food source), while the backward pass resembles said bee’s turn back to the beehive to share its information about the solution(food source) with other searcher (forager) bees (role change).

```
Initialization: Read problem data, Parameter values (B and NC),
Do
    1. Assign a (n) (empty) solution to each bee.
    2. For (i=0 ; i < NC ; i++)
       //Forward pass
          a) For (b= 0 ; b < B ; b++)
                For (s=0; s < f (NC); s++) //Count moves
                   I.  Evaluate Possible Moves;
                   II. Choose one move using the roulette wheel;
       //Backward pass
          b) For (b= 0 ; b < B ; b++)
                Evaluate the (partial/complete) solution of bee b;
          c) For (b= 0 ; b < B ; b++)
                Loyalty decision for bee b;
          d) For (b= 0 ; b < B ; b++)
                If (b is uncommitted), choose a recruiter by the roulette wheel.
    3. Evaluate all solutions and find the best one .Update $X_{best}$ and $f(X_{best})$
       While stopping criterion is not satisfied.
       Return ($X_{best}$, f ($X_{best}$))
```

Figure 3.1: Pseudo-Code for BCO [36]

The duration of each forward pass in the algorithm is regulated by *NC*, itself a representation of the number of solution components each bee visits in every forward pass. The forager bee assesses the usefulness of all partial routes from a trip in each forward movement, as well as delivers the food (scheduling solution) and shares information on quantity of solution with the other bees during its turn back to the beehive on the backward pass. Upon receiving information about the new partial or complete solutions, the forager bees initiate the backward pass and turn back to the beehive to meet their nest mate. The fitness of each solution is then evaluated and each of the bees has to decide whether to remain loyal to its food source and continue foraging it or recruit nest mates using the waggle dance, or to abandon the food source entirely in favor of one chosen by its nest mate. In the BCO algorithm, the operation of this rule functionally divides the bees into two types: scouts and followers. If a forager bee finds a solution with a greater profitability than the expectation of the colony, it assumes the role of a scout for that round and can advertise its solution to the other bees. Conversely, if the bee's solution in the forward pass is less profitable than the colony's expectations, it assumes the role of a follower bee.

The random function below is used to produce initial food sources based on the acceptable ranges of the parameters:

$$X_{i,j} = X_j^{Lower\ bound} + rand(0,1)\left(X_j^{Upper\ bound} - X_j^{lower\ bound}\right) \qquad (8)$$

Where $i = 1, \ldots, B$ and $j = 1, \ldots, D$, with B and D respectively denoting the total number of bees and optimization parameters. The initial solutions of the bees were produced using the roulette wheel method, while each forward pass has a specific number of constructive moves determined by *NC*.

Each bee corresponds to a single food source and modifies the solution in its memory using a combination of local information and neighboring food sources:

$$X'_{i,j} = X_{i,j} + \propto_{i,j} \left(X_{i,j} - X_{kj}\right) \qquad (9)$$

Where $k = 1, \ldots, D$. In the function above,xij, xkj, and $x'ij$ are a component of the solution, the neighbor of a component and a modified value of xij, respectively. The component can be reset to a more suitable value if the value of $x'ij$ exceeds its boundaries; in this study, the component was set as the boundary itself. The better value between xij and $x'ij$ is chosen based on the fitness values of the overall solution.

3.2 Pseudo code for BCO in Multi Objective Optimization Problems

An enhancement of the ABC algorithm, the design of the eMOABC stores Pareto optimal set (“non-dominated solutions”) discovered at the time the explore process in an external archive. The following are some of control parameters that need to be set prior to its initiation:

•*CS*, number of bees colony (Size).

•*limit* a solution which not is able to improve through “*limit*” of trials and will be abandoned.

This parameters have effect on performance of essential ABC and eMOABC algorithm .There are two other factor in eMOABC that are mostly used in swarm-based algorithms or multi objective evolutionary:

- *AS*, capacity of external archive (“Non-dominated solution”).
- *MaxCycle,* the stopping criteria. This value can be the greatest number of function evaluations.

The MOABC approaches have a ‘six’ main phases: initialization, send employed bees, crowding-distance assignment, send onlooker bees, Crowding Distance function, Conservation of the crowding distance archive, and send scout bees.

```
Algorithm: eMOABC

    Input:
        o  CS, the size of the bee Colony;
        o  AS, the size of the crowding archive;
        o  Limit, the abandonment criteria;
        o  MaxCycle, the termination criteria;
    Output:
        o  Archive.
1.  Initialization:
    1.1 Initialize the colony with the parameter CS;
    1.2 Initialize the crowding –distance archive with the parameter AS.
    1.3 Add non-dominated solutions within the initial colony into the
        archive.
        //Compute crowding-Distance for the members in the initial archive.
    1.4 CrowdingDistanceAssignment (Archive);
2.  For(t=0 ;t<maxCycle-1;t++){
    2.1 SendEmployedBees(Colony, archive);
    2.2 CrowdingDistanceAssignment (Archive);
    2.3 SendOnlookerBees(Colony, archive);
    2.4 CrowdingDistanceAssignment (Archive);
    2.5 SendScoutBees(Colony, archive);
        }
3.  Return archive
```

Figure 3.2: Pseudo-Code for eMOABC Algorithm [37]

3.2.1 Initialization

Every food source in the eMOABC algorithm symbolizes one solution to the given problem, and *FN* (the total number of individual 'solution') represents half the size of the colony (*CS*). Particularly, "*FN*=*CS*/2" fitness representing the entire bee colony in an '*N*-dimensional' and '*M*-objective' MOP are randomly produced in the decision range $\prod_{i=1}^{n}[lbi, Ubi]$ In the initialization phase. As such, a randomly-generated *n*-dimensional vector *Xi= (Xi1, Xi2...Xin)* is allocated to the solution using the following equation:

$$X_{id} = Lb_d + rand(0,1).(Ub_d - Lb_d). \qquad (10)$$

Where$i = 1,2,\dots,FN$; $d = 1,2,\dots,n$, and *rand (0, 1)* is a random number between [0,1]; *L* and *U* represent the lower bound and upper bound of the *dth* dimension.

Subsequently, variable $tril_i$ are attributed to every food source in an effort to determine which sources need to be abandoned in following runs. . Parameters $tril_i$ is a number of failer trials for finding solutions, and in the initialization phase each $tril_i$, *i=1, 2,..., FN* is set to be 0. The employed bee for a solution that cannot be enhanced in a given a amount of "*trils*" is transformed into a scout bee and back into an employed bee after performing a random search.

3.2.2 Crowding-Distance Assignment

The following process is utilized to calculate the crowding distance: First, the population is arranged in an ascending order based on the magnitude of each objective function value. Second, infinite distance value is assigned for each objective function. Third, distance value is the absolute difference in fitness function of two neighbor solution is added to the other intermediate solutions. The aggregate 'Crowding-Distance' is then computed as the addition of the distance score for each individual objective.

```
Function 1: CrowdingDistanceAssignment (archive)

1. n=|archive| // number of solutions in the archive
   // Initialize distance as 0.0 for each solution i
2. For each i ,archive [i].setCrowdingDistance(0.0)
3. For each objective m
   //Sort archive using each objective value
   3.1. archive =sort(archive ,m)
   3.2. archive [1].setCrowdingDistance(∞)
   3.3. archive [n].setCrowdingDistance(∞)
   3.4. For i=2 to n-1
        Distance= archive [i+1].m- archive [i-1].m
        Distance= Distance/ (objMax.m-obj.Min.m)
        // (objMax.m-obj.Min.m), range of the m_th objective
         archive [i].setCrowdingDistance (Distance)
   End For i
   End For m
```

Figure 3.3: Algorithm of Crowding Distance Assignment [37]

3.2.3 Conservation of the Crowding Distance In Archive

In the archive, each solution i is allocated a crowding distance i-dist representing an evaluate of the density of food source (solution) i in the search range. Solutions that form part of this set in the archive are Pareto optimal set ("non-dominated solution") and are more preferable when their location is a less-crowded region as opposed to a very-crowded region and a consequently smaller crowding-distance value.

When a new food source *I* is added to the archive, *I* is abandoned if it turns out to either be dominated by any member(solution) in the archive or equal to any solution in the archive. All 'dominated solutions' are discarded from the 'archive' if several members are dominated by *i*, which is then added to the archive. *I* solution can be inserted directly to the archive directly if the size of archive is empty or not full and solution *I* is not dominated by any member. If *I* is to be added to a full archive, however, it is necessary to first insert *i* into the ‘archive’ and then utilize the crowding Distance Assignment method in order to determine the new score of crowding distances for each member in the archive’s, as well as the new solution *i*, Solution that have minimum value crowding-distance is subsequently discarded from the archive.

3.2.4. Send Employed Bees

The simulated algorithm used to ‘Send Employed Bees’ (colony, archive) is shown in Figure.3.4. The employed bee attributed to each food source xi explores the temporary position $vi. vi$ is identical to the food source with the exception of the added change to a randomly-selected dimension *d*. As such, the equation for each food source can be updated to:

$$V_{id} = X_{id} + \delta_{id}.\,(X_{id} - X_{kd}) + \delta_{id}.\,(X_{id} - elite_d) \qquad (11)$$

Where: δ_{id} and δ_{id} two random number selected between *[−1, 1]*, and the solution xk is a neighbor to xi. The *elite* is the most suitable solution in the external *archive*, either because it is an intermediate solution in the least-crowded region or with the highest crowding distance value in the archive.

```
Function2: SendEmployedBees (Colony, archive)
------------------------------------------------
Select the elite from the archive
For i=1 to FN
    Determine one dimension d to be modified randomly
    Select a neighbor $X_k$ from the colony stochastically
    For food source $X_i$, Calculate its new Position $V_i$
        $V_{id} = X_{id} + \varphi_{id}.(X_{id} - X_{kd}) + \varphi_{id}.(X_{id} - elite_d)$
    Evaluate $V_i$
    If $V_i$ dominated $X_i$
        $X_i \leftarrow V_i$
        $Trial_i = 0$
        Add $V_i$ into the archive.
    Else If $X_i$ and $V_i$ are non-dominated with each other
        If $V_i$ is successfully added into the archive
            $X_i \leftarrow V_i$
            $Trial_i = 0$
        Else
            $Trial_i = Trial_i + 1$
        End If
    Else
        $Trial_i = Trial_i + 1$
    End If
End For
Use the function CalculateFitness (Colony, archive) to compute fitness
value of each food source.
------------------------------------------------
End of SendEmployedBees(Colony, archive)
```

Figure 3.4: Algorithms of Send Employed Bees [37]

3.2.5. Send Onlooker Bees

The simulated code used to 'Send Onlooker Bees' (colony, archive) is described in Figure.3.5. Once they are done optimizing heir solution, the employed bees back to the beehive and share their info regarding the quality of their respective solution with the onlooker bees, which then determine which food sources to exploit. To that end, the following formula is used to determine the probability of solution k advertised by employed bee:

$$Prob_k = 1 - \frac{F(Xk)}{\sum_{m=1}^{FN} F(Xm)} \tag{12}$$

Where the fitness value of Xm is reinstated using F (Xm). It is evident from Eq. (12) that a high selection probability is attributed to the food source with a lower fitness value due to the necessity of minimizing the fitness value used in the eMOABC.

```
Function 3: SendOnlookerBees (Colony, Archive)
Calculate the selection probility prob by Eq. (12) for each food source
Select the elite from the archive

i=1, t-1;

While t $\leq FN$

If rand $< Prob$

    t=t+1

    Determine one dimension d to be modified randomly
     Select a neighbor $X_k$ from the colony stochastically
      For food source Xi, Calculate its new position Vi

    $V_{id} = X_{id} + \varphi_{id}.(X_{id} - X_{kd}) + \varphi_{id}.(X_{id} - elite_d)$
 Evaluate Vi
 If  Vi dominated Xi
        $X_i \leftarrow V_i$
        $Trial_i = 0$

Add Vi into the archive.

          $X_i \leftarrow V_i$
          $Trial_i = 0$
    Else

          $Trial_i = Trial_i + 1$

    End If
 Else

          $Trial_i = Trial_i + 1$
 End If

 i=i+1
 If i== FN+1, then i=1

   End If
 End While

End of SendOnlookerBees (Colony, archive)
```

Figure 3.5: Algorithm of Send Onlooker Bees [37]

3.2.6. Send Scout Bees

The simulated algorithm used to ‘Send Scout Bees’ (colony, limit) is shown in Figure.3.6. Here, abandoned food sources are identified in the algorithm and subsequently replaced with new solution. Food sources that could not be enhanced by their onlooker or employed bee for limit cycles are abandoned and substituted for a vector, which is generated in a manner similar to the initialization phase.

```
Function4: SendScoutBees (Colony, Limit)
  Determine one food Source $X_k$ with maximum trial in the colony
      If $trial_k \geq limit$
          $X_{kd}=L_{bd}+rand (0, 1) . (U_{bd}-L_{bd}), d=1, 2,\ldots, n.$
          Evaluate the new food source $X_k$
          $Trial_k=0$
      End If
End of SendScoutBees (Limit)
```

Figure 3.6: Algorithm of Send Scout Bees [37]

Chapter 4

EXPERIMENTAL RESULT AND EVALUATIONS

Execution evaluation of the algorithm suggested, and the display of the comparable success set apart from the standard meta heuristics is to be undertaken within the difficulties of CEC2005 [25], CEC2009 [26,27,28,29] , CEC2017 [30] and CEC2018. [31].

Although the definitions, categorizations and characteristics (fitness landscape) are not described here, the functional benchmarks are clearly explained in the references. To ensure an equitable and comparative evaluation, the independent runs, and the stopping criteria of the function evaluations will be identical to those of the corresponding references. Likewise, the proposed algorithmic parameter methodology will remain the same in all test functions; throughout the program executions, there will be no interactive intervention. Test function integrity also dictates that the number of variables, in respect of the test functions, also obtain to that of the corresponding references.

4.1 CEC'05 Expensive Optimization Test Problems [25]

4.1.1 Common Definitions

All test functions are minimization problems defined as follows in (equation 1):

$$\min f(x), x = [x_1, x_2, \ldots., x_D]^T \tag{1}$$

Where D is the number of decision variable. All search ranges and Dimension are clearly explained in the references.

4.1.2 Results

Our Proposed algorithm was tested distinctly for optimizing CEC2005 single objective problems [25] .The results intended to demonstrate a large improvement from the BCO solutions. Each problem have been Averaged over 30 runs.

Table 4.1: Comparison between BCO, FEP and CEP on $f1 - f7$

Function	Number of Generation	BCO		FEP		CEP	
		Mean Best	Std Dev	Mean Best	Std Dev	Mean Best	Std Dev
$F1$	1500	**2.25×10^{-6}**	**1.85×10^{-6}**	5.7×10^{-4}	1.3×10^{-4}	2.2×10^{-4}	5.9×10^{-4}
$F2$	2000	**2.91×10^{-6}**	**8.42×10^{-7}**	8.1×10^{-3}	7.7×10^{-4}	2.6×10^{-3}	1.7×10^{-4}
$F3$	5000	**6.96×10^{-8}**	**9.52×10^{-8}**	1.6×10^{-2}	1.4×10^{-2}	5.0×10^{-2}	6.6×10^{-2}
$F4$	5000	**0.28**	**0.7**	0.3	0.5	2.0	1.2
$F5$	20000	**5.03**	**4.48E-01**	5.06	5.87	6.17	13.61
$F6$	1500	**0**	**0**	**0**	**0**	577.76	1125.76
$F7$	3000	**9.14×10^{-5}**	**1.00×10^{-5}**	7.6×10^{-3}	2.6×10^{-3}	1.8×10^{-2}	6.4×10^{-3}

The results from $f1 - f7$ test problems generated by competing algorithms are listed in Table 4.1. BCO shows a better performance on all tests than all other peer competitors, except $f6$ it obtains similar statistical results compared to FEP Algorithm.

Table 4.2: Comparison between BCO, FEP and CEP on $f8-f13$

Function	Number of Generation	BCO		FEP		CEP	
		Mean Best	Std Dev	Mean Best	Std Dev	Mean Best	Std Dev
$F8$	9000	-11823	2163	**-12554.5**	**52.6**	-7917.1	634.5
$F9$	5000	**8×10^{-8}**	**4.40×10^{-8}**	4.6×10^{-2}	1.2×10^{-2}	89.0	23.1
$F10$	1500	**5.7×10^{-10}**	**2.9×10^{-10}**	1.8×10^{-2}	2.1×10^{-3}	9.2	2.8
$F11$	2000	**1.61×10^{-3}**	**2.75×10^{-3}**	1.6×10^{-2}	2.2×10^{-2}	8.6×10^{-2}	0.12
$F12$	1500	5.77×10^{-3}	2.63×10^{-2}	**9.2×10^{-6}**	**3.6×10^{-6}**	1.76	2.4
$F13$	1500	3.66×10^{-4}	2.01×10^{-3}	**1.6×10^{-4}**	**7.3×10^{-5}**	1.4	3.7

From the results measured by BCO on $f8-f13$ test problems (Table 4.2), it is clearly show that BCO achieves the best performance among other competitors on $f9, f10$ and $f11$ test function, while performs slightly worse on $f8$, $f9$ and $f13$ by FEP.

Table 4.3: Comparison between BCO, FEP and CEP on $f14-f23$

Functio n	Number of Generation	BCO		FEP		CEP	
		Mean Best	Std Dev	Mean Best	Std Dev	Mean Best	Std Dev
$F14$	100	**9.98×10^{-1}**	**3.39×10^{-16}**	1.22	0.56	1.66	1.19
$F15$	4000	7.94×10^{-4}	3.15×10^{-4}	**5.0×10^{-4}**	**3.2×10^{-4}**	4.7×10^{-4}	3.0×10^{-4}
$F16$	100	**-1.03**	**1.55×10^{-8}**	-1.03	4.7×10^{-7}	-1.03	4.9×10^{-7}
$F17$	100	**3.98×10^{-4}**	**1.13×10^{-16}**	0.398	1.5×10^{-7}	0.398	1.5×10^{-7}
$F18$	100	**3**	**0**	3.02	0.11	3.0	0
$F19$	100	**-3.86**	**1.18×10^{-11}**	-3.86	1.4×10^{-5}	-3.86	1.4×10^{-2}
$F20$	200	**-3.25**	**3.66×10^{-5}**	-3.27	5.9×10^{-2}	-3.28	5.8×10^{-2}
$F21$	100	**-9.53**	**1.007**	-5.52	1.59	-6.86	2.67
$F22$	100	**-9.39**	**1.51**	-5.52	2.12	-8.27	2.95
$F23$	100	**-6.49**	**1.58**	-6.57	3.14	-9.10	2.92
$F14$	100	**9.98×10^{-1}**	**3.39×10^{-16}**	1.22	0.56	1.66	1.19

Table 4.3 shows that BCO wins over FEB and CEP on $f14 - f23$ and shows a better performance on all tests than all other peer competitors, but underperforms on $f15$ in which FEP performs better.

Table 4.1, 4.2, 4.3 shows comparison results for the Bees Algorithm, FEP and CEP in expression of average and standard deviations. The experiment results gained by the Bees approaches were compared with the solutions from [25].Note that the best results so far in the literature are reported in bold in all tables given in this section.

The average relative deviation of the Bees Algorithm was compared to the FEP and CEP. As can be seen the Bees Algorithm outperforms these two algorithms. And the total averages in Table 4.1, 4.2, 4.3, the Bees Algorithm is better than the FEP and CEP.

The standard deviation for the Bees Algorithm very small (nearly zero), which means that it is more robust than CEP and FEP. All the tables show that the execution of the Bees Algorithm is supreme to all other Algorithm.

4.2 CEC'17 Expensive Optimization Test Problems [30]

After downloading the Codes for ‘CEC'17’ test suite [30], all the test function were installed and handle as black-box optimization.

4.2.1 Common Definitions

All test problem are minimization function as following in (equation 2):

$$\min f(x), x = [x_1, x_2, \ldots., x_d]^T \tag{2}$$

Where:

D is the amount of decision variable of the test problem. All search space are declared for all function as [-100, 100] ^{D}.

4.2.2 Results

Our Proposed algorithm was tested clearly for optimizing CEC2017 single objective problems in 10 Dimension. The results designed to show a large enhancement from the BCO solutions.

Table 4.4: Summary of CEC’17 Optimization Test Problem

	No.	Functions	F_i^*=$F_i(x^*)$
Unimodel Functions	1	Shifted and Rotated Bent Cigar Function	100
	2	Shifted and Rotated Sum of Different Power Function	200
	3	Shifted and Rotated Zakharov Function	300
Simple Multimodal Functions	4	Shifted and Rotated Rosenbrock’s Function	400
	5	Shifted and Rotated Rastrigin’s Function	500
	6	Shifted and Rotated Expanded Scaffer’s F6 Function	600
	7	Shifted and Rotated Lunacek Bi_Rastrigin’s Function	700
	8	Shifted and Rotated Non-Continuous Rastrigin’s Function	800
	9	Shifted and Rotated Levy Function	900
	10	Shifted and Rotated Schwefel’s Function	1000
Hybrid Functions	11	Hybrid Function 1(N=3)	1100
	12	Hybrid Function 2(N=3)	1200
	13	Hybrid Function 3(N=3)	1300
	14	Hybrid Function 4(N=4)	1400
	15	Hybrid Function 5(N=4)	1500
	16	Hybrid Function 6(N=4)	1600
	17	Hybrid Function 6(N=5)	1700
	18	Hybrid Function 6(N=5)	1800
	19	Hybrid Function 6(N=5)	1900
	20	Hybrid Function 6(N=6)	2000
Composition Function	21	Composition Function 1 (N=3)	2100
	22	Composition Function 1 (N=3)	2200
	23	Composition Function 1 (N=4)	2300
	24	Composition Function 1 (N=4)	2400
	25	Composition Function 1 (N=5)	2500
	26	Composition Function 1 (N=5)	2600
	27	Composition Function 1 (N=6)	2700
	28	Composition Function 1 (N=6)	2800
	29	Composition Function 1 (N=3)	2900
	30	Composition Function 1 (N=3)	3000
Search R:$[-100,100]^D$			

Table 4.5: Best Result of BCO Algorithm in Dimension 10 Over 30 Runs.

Function Number	Optimal Solution	BCO	Error
1	100	100	0.00E+00
2	200	200	0.00E+00
3	300	300	0.00E+00
4	400	400	0.00E+00
5	500	500	0.00E+00
6	600	600	0.00E+00
7	700	710	1.04E+01
8	800	800	0.00E+00
9	900	900	0.00E+00
10	1000	1000.25	2.50E-01
11	1100	1100	0.00E+00
12	1200	1200.187	1.87E-01
13	1300	1300.144	1.44E-01
14	1400	1400.022	2.18E-02
15	1500	1500.005	5.37E-03
16	1600	1600.205	2.05E-01
17	1700	1700.81	8.10E-01
18	1800	1800.13	1.30E-01
19	1900	1900.174	1.74E-01
20	2000	2000	0.00E+00
21	2100	2100.011	1.09E-02
22	2200	2200	0.00E+00
23	2300	2300.247	2.47E-01
24	2400	2400.008	8.25E-03
25	2500	2500.233	2.33E-01
26	2600	2600.03	2.97E-02
27	2700	2700.19	1.90E-01
28	2800	2800.65	6.50E-01
29	2900	2900	0.00E+00
30	3000	3000.002	1.53E-03

The results of the analyses of Table 4.5 revealed an apparent improvement in the quality of solutions, which obviously tend to get closeness to the optimal values.

The findings of our experiment with BCO are consistent to some extent with the past studies on CEC 2017 problem optimization .Both of the Unimodal functions results in BCO algorithm in Dimension 10 reached optimal solutions without any small differences from optimality. Multimodal functions were mixed between problems which had very high differences from optimal solutions; problem no. 7, and Composite functions that included problem no. 10, had very small differences from optimal solutions.

While the rest of the problems' results in the same category reached to the optimal solutions. Finally, Hybrid functions and Composition function which included problems from no.11 to no.30, reached near-optimal solutions with relatively small differences from optimality.

Table 4.6: IGD Values Obtained by BCO and it are 3 Competitors for CEC’17 Test Function

Function Number	BCO	LSHADE	SPA	SPACMA
1	**0.00E+00**	0.00E+00	0.00E+00	0.00E+00
2	**0.00E+00**	0.00E+00	0.00E+00	0.00E+00
3	**0.00E+00**	0.00E+00	0.00E+00	0.00E+00
4	**0.00E+00**	0.00E+00	0.00E+00	0.00E+00
5	**0.00E+00**	3.0E+00	1.8E+00	1.8E+00
6	**0.00E+00**	0.00E+00	0.00E+00	0.00E+00
7	**1.04E+01**	1.2E+01	1.2E+01	1.1E+01
8	**0.00E+00**	2.4E+00	1.9E+00	8.4E-01
9	**0.00E+00**	0.00E+00	0.00E+00	0.00E+00
10	**2.50E-01**	2.2E+01	2.2E+01	2.2E+01
11	**0.00E+00**	4.1E-01	0.00E+00	0.00E+00
12	**1.87E-01**	7.7E+01	1.2E+02	1.2E+02
13	**1.44E-01**	3.2E+00	3.6E+00	4.4E+00
14	2.18E-02	1.7E-01	**2.0E-02**	1.6E-01
15	**5.37E-03**	1.7E-01	2.7E-01	4.1E-01
16	**2.05E-01**	4.1E-01	5.2E-01	7.4E-01
17	8.10E-01	1.7E-01	**1.2E-01**	1.6E-01
18	**1.30E-01**	2.8E-01	2.4E+00	4.4E+00
19	1.74E-01	**1.1E-02**	5.5E-02	2.3E-01
20	**0.00E+00**	1.5E-02	1.8E-01	3.1E-01
21	**1.09E-02**	1.6E+02	1.6E+02	1.0E+02
22	**0.00E+00**	1.0E+02	1.0E+02	1.0E+02
23	**2.47E-01**	3.0E+02	3.0E+02	3.0E+02
24	**8.25E-03**	3.2E+02	2.9E+02	2.7E+02
25	**2.33E-01**	4.1E+02	4.2E+02	4.3E+02
26	**2.97E-02**	3.0E+02	3.0E+02	3.0E+02
27	**1.90E-01**	3.9E+02	3.2E+02	3.2E+02
28	**6.50E-01**	3.6E+02	4.0E+02	3.2E+02
29	**0.00E+00**	2.3E+02	2.3E+02	2.3E+02
30	**1.53E-03**	7.8E+04	4.1E+04	4.1E+02

Table 4.6 shows the best IGD values for CEC’17 Test Problem. The results gained Bee Colony algorithm as against with the solution from [30]. IGD score of obtained solutions found by Bees approach are very small. That means Bees Algorithm can discover a well spread sets and high quality solution in objective range for each problems.

Comparisons between BCO with the other algorithms in the competition denote that the Bees approach is more efficacious than other algorithms in all test problems.

By examining the comparison between error rates demonstrated in Table 4.6, it can be concluded that the highest number of ***best*** problem optimization results belong to the BCO Algorithm. The table showed superior performance of BCO from optimizing results of 27 out of 30 problems, which is the highest between all the methods from literature. In problems number (1,2,3,4,5,6,8,9,11,20,22 and 29), the error rates of *BCO* appeared to be very close to optimality. The rest of the problems' results varied between generally small differences and extreme differences from the optimal values.

4.3 CEC'09 Test Problems for Multi-objective problem [26]

Set of benchmarks through ‘hybrid composition operations’, ‘random shifting’ and ‘random shifting and rotation’. More information for each problem are presented in [26, 27, 28, 29].

Table 4.7: Min, Max, Average, Standard Deviation of IGD Values and Number of Function Evaluation of BCO in 30 Runs.

Function	Average	Min	Max	Std	Function Evaluation
ZDT1	4.77E-04	3.91E-04	9.84E-04	1.88E-04	60300
ZDT2	4.97E-04	4.43E-04	6.86E-04	7.86E-05	60300
ZDT3	6.63E-04	4.36E-04	9.97E-04	1.54E-04	146400
ZDT4	3.76E-04	3.69E-04	4.53E-04	2.94E-05	60300
ZDT6	5.32E-04	3.38E-04	1.04E-03	2.16E-04	24120
DTLZ1	1.14E-03	1.57E-04	1.61E-03	6.27E-04	91000
DTLZ2	5.09E-03	6.92E-04	7.48E-03	2.82E-03	91000
DTLZ3	5.79E-02	7.82E-04	8.42E-02	3.60E-02	91000
DTLZ4	3.39E-03	4.63E-04	4.91E-03	1.87E-03	91000
DTLZ5	5.53E-04	5.40E-04	6.87E-04	4.57E-05	91000
DTLZ6	1.26E-04	1.32E-04	1.66E-04	1.09E-05	91000
DTLZ7	5.21E-03	2.89E-03	7.01E-04	7.80E-03	91000
UF1	5.27E-03	7.25E-04	7.39E-03	2.91E-03	24120
UF2	2.21E-03	3.07E-04	3.14E-03	1.22E-03	12060
UF3	4.23E-03	5.65E-04	6.03E-03	2.33E-03	24120
UF4	1.59E-03	2.22E-04	2.30E-03	8.76E-04	12060
UF5	6.31E-02	9.20E-04	9.39E-02	3.93E-02	60300
UF6	3.36E-02	2.18E-04	6.57E-02	3.33E-02	24600
UF7	7.74E-03	1.85E-04	6.54E-02	2.13E-02	60300
UF8	1.86E-02	1.79E-04	6.51E-02	1.88E-02	60300
UF9	1.86E-02	1.85E-04	6.52E-02	1.88E-02	18090
UF10	1.81E-02	1.91E-04	6.57E-02	1.90E-02	18090
WFG1	2.15E-03	2.20E-03	2.12E-03	2.66E-05	1026000
WFG 2	7.14E-03	7.55E-03	6.96E-03	2.45E-04	1026000
WFG 3	8.36E-02	8.86E-02	7.96E-02	3.64E-03	542700
WFG 4	4.71E-03	4.88E-03	4.55E-03	1.59E-04	92250
WFG 5	4.90E-03	6.54E-03	6.57E-04	2.40E-03	93600
WFG 6	1.42E-03	1.90E-03	1.13E-03	3.32E-04	1026000
WFG 7	4.97E-03	4.25E-04	5.33E-03	4.25E-03	542700
WFG 8	3.84E-03	6.74E-04	4.99E-03	3.27E-03	93600
WFG 9	1.78E-03	1.25E-04	1.93E-03	1.63E-03	92250

It can be seen from Table 4.7 that BCO is a robust and successful approach explain with small 'IGD' score and their standard deviation.

Tables 4.8, 4.9, 4.10, 4.11, 4.12, 4.13, 4.14, 4.15 and 4.16 clarify the rating of all problems in "CEC2009" and BCO with respect to the IGD values. From[26], MOEA/D, SMPSO, GDE3, MOCell and SPEA2 are the best five approaches in the contest in order .The defender of this contest was "MOEA/D". From all our result can see that BCO acted more better than "MOEA/D" in all test problem .The proposed BCO takes the first rank in all test problems.

Table 4.8: IGD Values Obtained by BCO and it are 11 Competitors for UF1, UF2 and UF3.

Rank	UF1	IGD	UF2	IGD	UF3	IGD
1	**MOBCO**	**5.27E-03**	**MOBCO**	**2.21E-03**	**MOBCO**	**4.23E-03**
2	GDE3	$8.60e-02$	IBEA	$7.51e-02$	MOEA/D	$7.85e-02$
3	CellDE	$6.24e-02$	MOCell	$6.82e-02$	GDE3	$3.52e-01$
4	MOEA/D	$3.11e-02$	AbYSS	$6.51e-02$	PAES	$3.45e-01$
5	PAES	$3.64e-01$	SMPSO	$4.66e-02$	IBEA	$2.90e-01$
6	MOCell	$1.64e-01$	SPEA2	$4.52e-02$	MOCell	$2.83e-01$
7	IBEA	$1.46e-01$	GDE3	$4.30e-02$	AbYSS	$2.80e-01$
8	AbYSS	$1.34e-01$	OMOPSO	$4.05e-02$	CellDE	$2.65e-01$
9	SMPSO	$1.26e-01$	NSGAII	$3.85e-02$	SMPSO	$2.26e-01$
10	SPEA2	$1.24e-01$	CellDE	$3.85e-02$	SPEA2	$1.94e-01$
11	OMOPSO	$1.08e-01$	MOEA/D	$2.47e-02$	OMOPSO	$1.78e-01$
12	NSGAII	$1.03e-01$	PAES	$1.79e-01$	NSGAII	$1.58e-01$

Table 4.9: IGD Values Obtained by BCO and it are 11 Competitors for UF4, UF5 and UF6.

Rank	UF4	IGD	UF5	IGD	UF6	IGD
1	**MOBCO**	**1.59E-03**	**MOBCO**	**6.31E-02**	**MOBCO**	**3.36E-02**
2	MOEA/D	$7.96e-02$	MOEA/D	$7.61e-01$	PAES	$7.26e-01$
3	IBEA	$6.83e-02$	PAES	$5.29e-01$	SMPSO	$6.85e-01$
4	OMOPSO	$6.43e-02$	AbYSS	$5.07e-01$	OMOPSO	$5.34e-01$
5	AbYSS	$6.26e-02$	MOCell	$4.59e-01$	MOCell	$4.30e-01$
6	MOCell	$5.69e-02$	CellDE	$4.16e-01$	AbYSS	$4.16e-01$
7	CellDE	$5.51e-02$	IBEA	$3.99e-01$	IBEA	$3.77e-01$
8	SMPSO	$5.49e-02$	SPEA2	$4.59e-01$	CellDE	$3.16e-01$
9	NSGAII	$5.20e-02$	NSGAII	$4.16e-0$	SPEA2	$2.74e-01$
10	SPEA2	$5.12e-02$	GDE3	$3.99e-01$	NSGAII	$2.58e-01$
11	GDE3	$4.70e-02$	OMOPSO	$1.19e-00$	MOEA/D	$2.56e-01$
12	PAES	$2.18e-01$	SMPSO	$1.86e-00$	GDE3	$1.56e-01$

It is apparent from Table 4.8 and Table 4.9 that the MOBCO is the most competitive algorithm obtaining the best values on 6 problems (UF1, UF2 ,UF3,UF4,UF5 and UF6), and then it is the GDE3 algorithm which has computed the second best fronts regarding to this indicator on this evaluated problems .MOEA/D, SMPSO,AbYSS,MOCell and OMOPSO perform similarly, while CellDE and PAES give poor results regarding this indicator.

Table 4.10: IGD Values Obtained by BCO and it are 11 competitors for UF7, UF8 and UF9.

Rank	UF7	IGD	UF8	IGD	UF9	IGD
1	**MOBCO**	**7.74E-03**	**MOBCO**	**1.86E-02**	**MOBCO**	**1.86E-02**
2	OMOPSO	$6.48e-02$	IBEA	$4.43e-01$	CellDE	$6.57e-01$
3	SMPSO	$6.19e-02$	PAES	$4.13e-01$	OMOPSO	$5.59e-01$
4	CellDE	$5.67e-02$	CellDE	$4.054e-01$	SMPSO	$3.976e-01$
5	GDE3	$4.11e-02$	OMOPSO	$3.66e-01$	AbYSS	$3.81e-01$
6	MOEA/D	$1.52e-02$	GDE3	$3.32e-01$	NSGAII	$3.15e-01$
7	PAES	$5.05e-01$	AbYSS	$2.80e-01$	MOCell	$2.87e-01$
8	MOCell	$3.45e-01$	MOCell	$2.57e-01$	IBEA	$2.83e-01$
9	AbYSS	$3.25e-01$	SMPSO	$2.26e-01$	PAES	$2.71e-01$
10	NSGAII	$1.85e-01$	NSGAII	$2.21e-01$	SPEA2	$2.04e-01$
11	IBEA	$2.84e-01$	SPEA2	$1.99e-01$	GDE3	$2.01e-01$
12	SPEA2	$1.69e-01$	MOEA/D	$1.15e-01$	MOEA/D	$1.57e-01$

Table 4.11: IGD Values Obtained by BCO and it are 11 Competitors for UF10, ZDT1 and ZDT2.

Rank	UF10	IGD	ZDT1	IGD	ZDT2	IGD
1	**MOBCO**	**1.81E-02**	**MOBCO**	**4.77E-04**	**MOBCO**	**4.97E-04**
2	MOEA/D	$8.73e-01$	NSGAII	$4.83e-03$	IBEA	$9.41e-03$
3	AbYSS	$6.69e-01$	CellDE	$4.83e-03$	MOEA/D	$9.13e-03$
4	IBEA	$6.01e-01$	IBEA	$4.10e-03$	NSGAII	$4.89e-03$
5	PAES	$5.10e-01$	SPEA2	$3.92e-03$	CellDE	$4.36e-03$
6	MOCell	$4.43e-01$	GDE3	$3.77e-03$	GDE3	$3.91e-03$
7	NSGAII	$3.85e-01$	AbYSS	$3.72e-03$	SPEA2	$3.89e-03$
8	SPEA2	$3.24e-01$	OMOPSO	$3.71e-03$	OMOPSO	$3.83e-03$
9	SMPSO	$2.92e-01$	MOCell	$3.68e-03$	AbYSS	$3.82e-03$
10	GDE3	$1.55e+00$	SMPSO	$3.67e-03$	MOCell	$3.79e-03$
11	OMOPSO	$2.20e+00$	MOEA/D	$1.25e-02$	SMPSO	$3.79e-03$
12	CellDE	$2.58e+00$	PAES	$1.17e-02$	PAES	$1.46e-02$

Table 4.10 and Table 4.11 indicate that the MOBCO is the most competitive algorithm obtaining the best values on 4 problems (UF7, UF8, UF9, UF10). Roughly, SMPSO, AbYSS, CellDE, MOCell,SPEA2 and GDE3 give competitive results regarding this indicator. The rest of the algorithms (MOEA/D,PAES,IBEA and NSGAII) obtain worse fronts compared to other algorithms.

Tables 4.8, 4.9, 4.10 and 4.11 shows the best IGD values for all two objective UF1 to UF7 and for three objective UF8 to UF10 .The solutions gained by the proposed approach (Bee Colony Algorithm) were compared with the solutions from [26]. IGD score of gained solutions computed by Bees Approach are very small. This indicates that Bee Colony algorithm able detect a well distributed sets and high quality solution in objective range for all objective function in each problem.

Tables 4.8, 4.9, 4.10, 4.11 illustrate the ranking of all two objective UF1 to UF7 and for three objective UF8 to UF10. Tables show that BCO performance much better than other algorithms in all problems .The proposed BCO takes the first position in all test problems.

Table 4.12: IGD Values Obtained by BCO and it are 11 Competitors for ZDT3, ZDT4 and ZDT5.

Rank	ZDT3	IGD	ZDT4	IGD	ZDT6	IGD
1	**MOBCO**	**6.63E-04**	**MOBCO**	**3.76E-04**	**MOBCO**	**5.32E-04**
2	MOCell	$6.17e-03$	PAES	$7.34e-03$	PAES	$7.07e-03$
3	NSGAII	$5.38e-03$	NSGAII	$4.93e-03$	IBEA	$5.16e-03$
4	SPEA2	$4.84e-03$	MOCell	$3.84e-03$	NSGAII	$4.76e-03$
5	GDE3	$4.36e-03$	SMPSO	$3.71e-03$	MOEA/D	$4.16e-03$
6	OMOPSO	$4.35e-03$	AbYSS	$4.41e-03$	CellDE	$3.43e-03$
7	SMPSO	$4.28e-03$	SPEA2	$4.07e-03$	SPEA2	$3.17e-03$
8	PAES	$5.61e-02$	IBEA	$6.26e-01$	GDE3	$3.12e-03$
9	IBEA	$2.97e-02$	GDE3	$4.72e-01$	AbYSS	$3.05e-03$
10	MOEA/D	$1.72e-02$	MOEA/D	$1.43e-01$	OMOPSO	$3.01e-03$
11	AbYSS	$1.50e-02$	OMOPSO	$4.92e+00$	SMPSO	$3.03e-03$
12	CellDE	$1.02e-02$	CellDE	$4.24e+00$	MOCell	$3.00e-03$

Table 4.11 and Table 4.12 indicate that the MOBCO has been also the best algorithm obtaining the values in 5 Problems (ZDT1, ZDT2, ZDT3, ZDT4 and ZDT6) . GDE3 may be the second best algorithm which out performs other algorithms on considerable number of problems. CellDE, IBEA and PAES are three worst algorithms in terms of this indicator.

Tables 4.11 and 4.12 illustration that the best IGD values for ZDT1 – ZDT6 of gained solution sets found by the proposed algorithm are also very small.

This means the BCO performs most efficient than the other algorithms on all test cases over. Tables illustrate the ranking for ZDT1– ZDT6 compared with other algorithms, and it can be seen over the tables that BCO performed better than other algorithms in all test problems and takes the first position in all test problems.

Table 4.13: IGD Values Obtained by BCO and it are 11 Competitors for WFG1, WFG2 and WFG3.

Rank	WFG1	IGD	WFG2	IGD	WFG3	IGD
1	**MOBCO**	**2.15E-03**	**MOBCO**	**7.14E-03**	**MOBCO**	**8.36E-02**
2	CellDE	$8.73e-02$	IBEA	$9.84e-02$	PAES	$1.67e-01$
3	GDE3	$5.07e-02$	AbYSS	$6.21e-02$	MOEA/D	$1.43e-01$
4	OMOPSO	$8.36e-01$	MOEA/D	$4.97e-02$	CellDE	$1.42e-01$
5	AbYSS	$7.32e-01$	MOCell	$4.93e-02$	NSGAII	$1.41e-01$
6	SPEA2	$3.71e-01$	NSGAII	$3.75e-02$	IBEA	$1.39e-01$
7	MOCell	$3.46e-01$	SPEA2	$3.58e-02$	AbYSS	$1.39e-01$
8	MOEA/D	$3.21e-01$	CellDE	$1.14e-02$	SPEA2	$1.39e-01$
9	IBEA	$2.89e-01$	SMPSO	$1.071e-02$	GDE3	$1.386e-01$
10	NSGAII	$1.96e-01$	OMOPSO	$1.03e-02$	SMPSO	$1.385e-01$
11	PAES	$1.25e+00$	GDE3	$1.00e-02$	MOCell	$1.38e-01$
12	SMPSO	$1.10e+00$	PAES	$3.06e-01$	OMOPSO	$1.38e-01$

Table 4.14: IGD Values Obtained by BCO and it are 11 Competitors for WFG4, WFG5 and WFG6.

Rank	WFG4	IGD	WFG5	IGD	WFG6	IGD
1	**MOBCO**	**4.71E-03**	**MOBCO**	**4.90E-03**	**MOBCO**	**1.42E-03**
2	SMPSO	$2.69e-02$	IBEA	$7.28e-02$	PAES	$9.74e-02$
3	OMOPSO	$2.30e-02$	PAES	$6.97e-02$	AbYSS	$9.32e-02$
4	MOEA/D	$2.22e-02$	MOEA/D	$6.82e-02$	MOCell	$6.32e-02$
5	IBEA	$2.02e-02$	NSGAII	$6.81e-02$	IBEA	$5.39e-02$
6	CellDE	$1.61e-02$	SPEA2	$6.67e-02$	NSGAII	$3.49e-02$
7	PAES	$1.55e-02$	CellDE	$6.64e-02$	SPEA2	$2.31e-02$
8	NSGAII	$1.36e-02$	GDE3	$6.64e-02$	MOEA/D	$1.90e-02$
9	SPEA2	$1.27e-02$	SMPSO	$6.63e-02$	CellDE	$1.45e-02$
10	GDE3	$1.08e-02$	MOCell	$6.62e-02$	GDE3	$1.30e-02$
11	MOCell	$1.04e-02$	OMOPSO	$6.62e-02$	SMPSO	$1.28e-02$
12	AbYSS	$1.04e-02$	AbYSS	$6.59e-02$	OMOPSO	$1.26e-02$

Table 4.15: IGD Values Obtained by BCO and it are 11 Competitors for WFG7, WFG8 and WFG9.

Rank	WFG7	IGD	WFG8	IGD	WFG9	IGD
1	**MOBCO**	4.97E-03	**MOBCO**	3.84E-03	**MOBCO**	1.78E-03
2	MOEA/D	$2.02e-02$	NSGAII	$9.90e-02$	PAES	$2.06e-02$
3	PAES	$1.95e-02$	MOEA/D	$8.30e-02$	AbYSS	$2.03e-02$
4	NSGAII	$1.62e-02$	PAES	$6.95e-02$	MOCell	$2.01e-02$
5	IBEA	$1.55e-02$	GDE3	$1.36e-02$	IBEA	$1.74e-02$
6	CellDE	$1.43e-02$	IBEA	$1.32e-02$	NSGAII	$1.55e-02$
7	SPEA2	$1.29e-02$	AbYSS	$1.30e-02$	SPEA2	$1.45\ e-02$
8	GDE3	$1.24e-02$	OMOPSO	$1.26e-02$	MOEA/D	$1.45e-02$
9	SMPSO	$1.19e-02$	CellDE	$1.21e-02$	CellDE	$1.45e-02$
10	AbYSS	$1.19e-02$	MOCell	$1.17e-02$	GDE3	$1.35e-02$
11	OMOPSO	$1.17e-02$	SPEA2	$1.05e-02$	SMPSO	$1.35e-02$
12	MOCell	$1.17e-02$	SMPSO	$1.03e-02$	OMOPSO	$3.04e-02$

By observing Table 4.13, 4.14 and 4.15 carefully, we find that the MOBCO is the most competitive algorithm obtaining the best values on all problems, and then it is the PAES algorithm which has computed the second best fronts regarding to this indicator on evaluated problems. MOEA/D, SMPSO, AbYSS, MOCell and OMOPSO perform similarly, while CellDE and GDE3give poor results regarding this indicator.

The IGD results from WFG1-WFG9 test problems generated by BCO algorithms are listed in Tables 4.13, 4.14 and 4.15, it is clearly shown that BCO achieves the best performance among its all other competitors in three objective function problems.

In addition, Tables illustrate the ranking of all three objective function problems WFG1 to WFG9.From tables can be see that the BCO performed more better than other algorithms in all problems and takes the first position in all test problems.

Table 4.16: IGD Values Obtained by BCO and it are 11 Competitors for DTLZ1, DTLZ2 and DTLZ3.

Rank	DTLZ1	IGD	DTLZ2	IGD	DTLZ3	IGD
1	**MOBCO**	**1.14E-03**	**MOBCO**	**5.09E-03**	**MOBCO**	**5.79E-02**
2	PAES	$5.86e-02$	SMPSO	$7.17e-02$	MOCell	$7.55e-01$
3	MOCell	$2.86e-02$	AbYSS	$6.88e-02$	IBEA	$5.11e-01$
4	SMPSO	$2.82e-02$	NSGAII	$6.88e-02$	AbYSS	$3.94e-01$
5	AbYSS	$2.73e-02$	OMOPSO	$6.88e-02$	SPEA2	$3.38e-01$
6	NSGAII	$2.61e-02$	MOEA/D	$6.71e-02$	NSGAII	$2.93e-01$
7	MOEA/D	$2.54e-02$	MOCell	$6.68e-02$	PAES	$1.91e-01$
8	GDE3	$2.33e-02$	CellDE	$6.61e-02$	SMPSO	$1.15e-01$
9	SPEA2	$2.02e-02$	GDE3	$6.28e-02$	MOEA/D	$1.17e+00$
10	CellDE	$1.60e-01$	SPEA2	$5.42e-02$	GDE3	$2.25e+00$
11	IBEA	$1.81e-01$	PAES	$3.15e-01$	CellDE	$8.51e+00$
12	OMOPSO	$1.18e+01$	IBEA	$1.22e-01$	OMOPSO	$1.15e+02$

Table 4.17: IGD Values Obtained by BCO and it are 11 Competitors for DTLZ4, DTLZ5 and DTLZ6.

Rank	DTLZ4	IGD	DTLZ5	IGD	DTLZ6	IGD
1	**MOBCO**	**3.39E-03**	**MOBCO**	**5.53E-04**	**MOBCO**	**1.26E-04**
2	CellDE	$7.71e-02$	CellDE	$8.56e-03$	MOEA/D	$9.36e-03$
3	SMPSO	$6.80e-02$	PAES	$6.83e-03$	PAES	$7.13e-03$
4	GDE3	$6.57e-02$	NSGAII	$5.42e-03$	CellDE	$4.54e-03$
5	OMOPSO	$6.48e-02$	SPEA2	$4.33e-03$	GDE3	$4.15e-03$
6	NSGAII	$6.39e-02$	GDE3	$4.19e-03$	SMPSO	$3.93e-03$
7	AbYSS	$6.05e-02$	OMOPSO	$4.13e-03$	OMOPSO	$3.89e-03$
8	MOEA/D	$5.49e-02$	AbYSS	$4.08e-03$	AbYSS	$7.89e-02$
9	PAES	$3.99e-01$	MOCell	$4.05e-03$	IBEA	$5.75e-02$
10	IBEA	$2.10e-01$	SMPSO	$4.09e-03$	NSGAII	$1.35e-02$
11	SPEA2	$1.37e-01$	IBEA	$1.93e-02$	SPEA2	$1.25e-02$
12	MOCell	$1.35e-01$	MOEA/D	$1.04e-02$	MOCell	$7.55e-01$

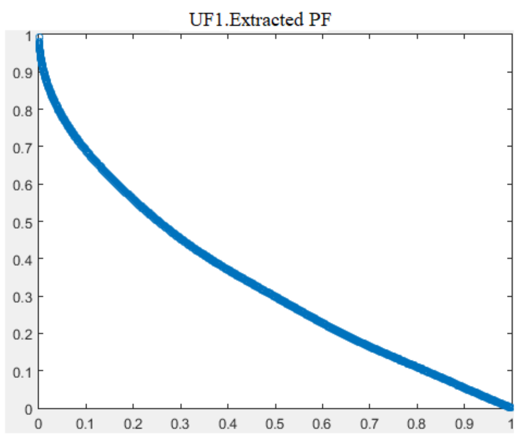
Table 4.18: IGD Values Obtained by BCO and it are 11 Competitors for DTLZ7.

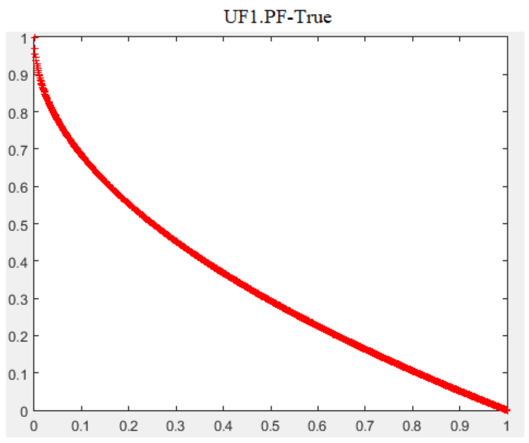
Rank	DTLZ7	IGD
1	**MOBCO**	**5.21E-03**
2	OMOPSO	$8.68e-02$
3	SMPSO	$8.52e-02$
4	NSGAII	$7.64e-02$
5	GDE3	$7.47e-02$
6	SPEA2	$6.96e-02$
7	PAES	$8.87e-01$
8	IBEA	$3.99e-01$
9	AbYSS	$3.94e-01$
10	MOCell	$2.45e-01$
11	MOEA/D	$1.90e-01$
12	CellDE	$1.23e-01$

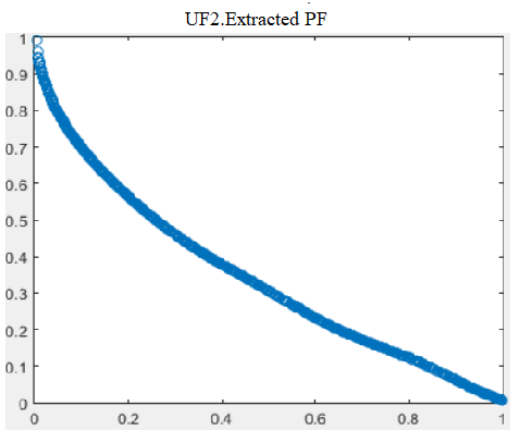
It is apparent from Table 4.16, 4.17 and Table 4.9 that the MOBCO is the most competitive algorithm obtaining the best values on 7 problems (DTLZ1, DTLZ2, DTLZ3 , DTLZ4 , DTLZ5 , DTLZ6 and DTLZ7), and then it is the NSGAII algorithm which has computed the second best fronts regarding to this indicator on this evaluated problems .MOEA/D, SMPSO, AbYSS and OMOPSO perform similarly, while CellDE and MOCell give poor results regarding this indicator.

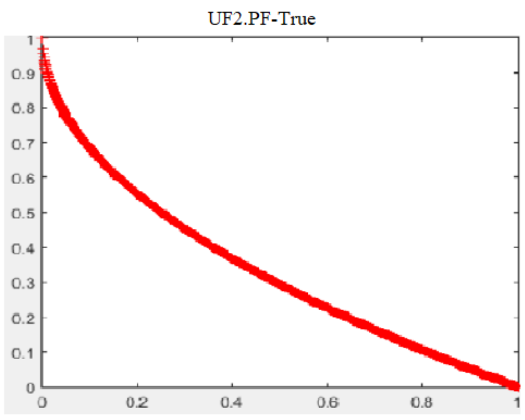
Tables 4.16, 4.17 and 4.18 shows the best IGD values for all three objective function problems DTLZ1 to DTLZ7.The results gained Bee Colony algorithm as against with the solution from [26]. IGD score of obtained solutions found by Bees approach are very small. That means Bees Algorithm can discover a well spread sets and high quality solution in objective range for all objective function for each problems. Tables illustrate the ranking of all three objectives DTLZ1 to DTLZ7 .From tables can be see that BCO acted better than other approaches in all problems. BCO takes the first position in all test problems.

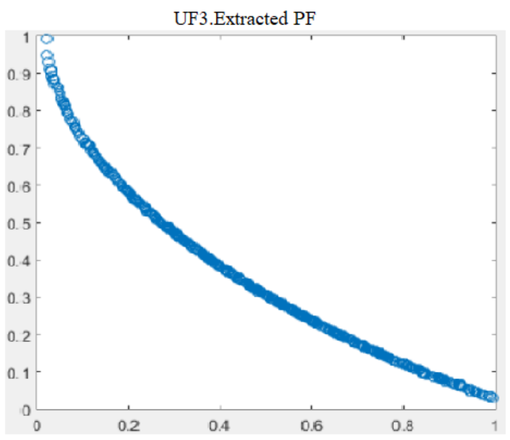
Comparisons between BCO with the second best acting algorithm in the competition called SMPSO denote that the Bees approach is more efficacious than SMPSO in all test problems. BCO executed substantially better rank than GDE3, MOCell and SPEA2 approach in all test problems.

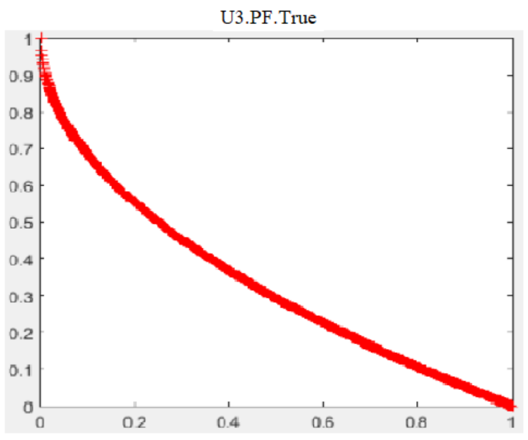

UF1.Extracted PF

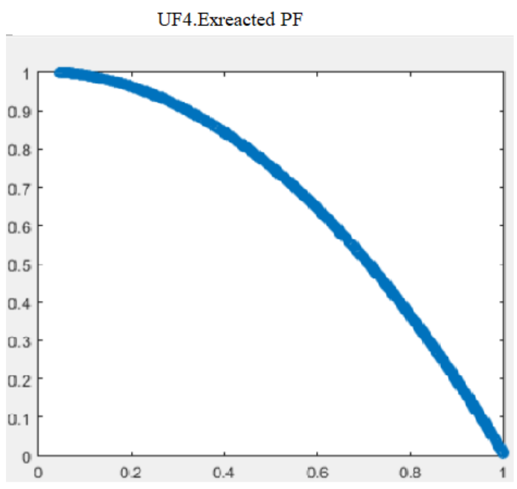

UF1.PF-True

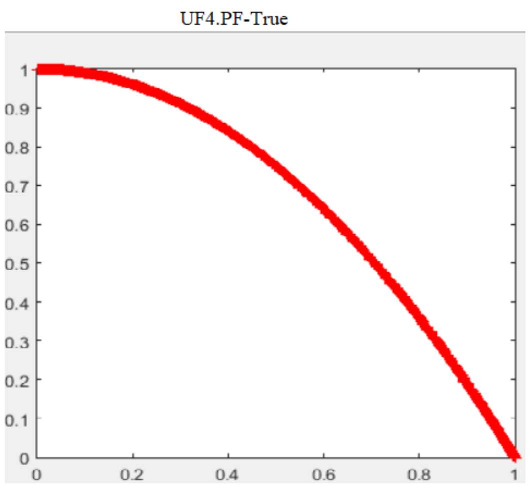

UF2.Extracted PF


UF2.PF-True

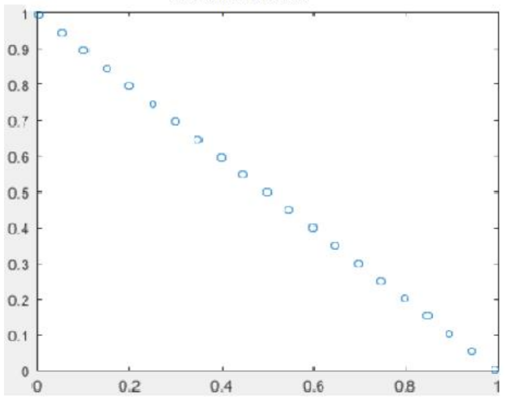

UF3.Extracted PF


U3.PF.True

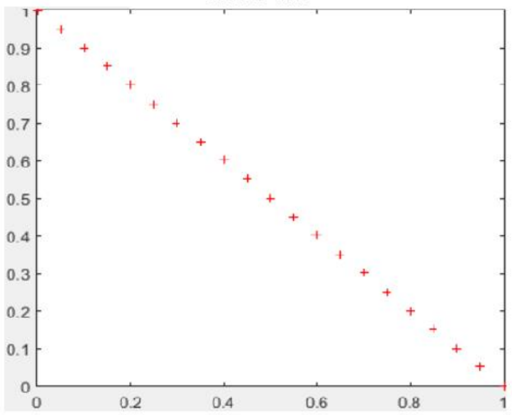

UF4.Exreacted PF


UF4.PF-True

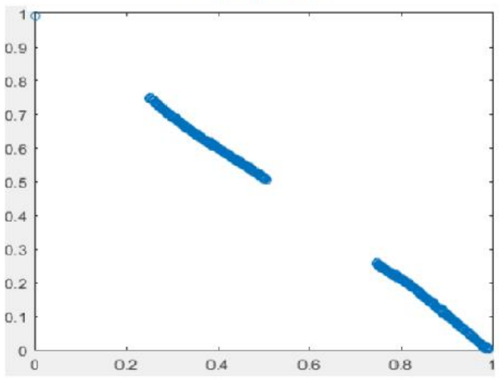
UF5.Extracted PF



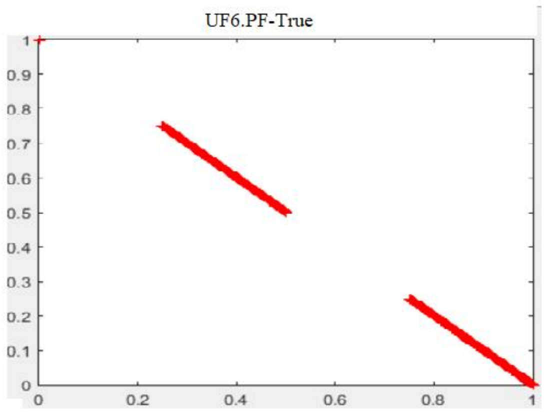
UF5.PF-True



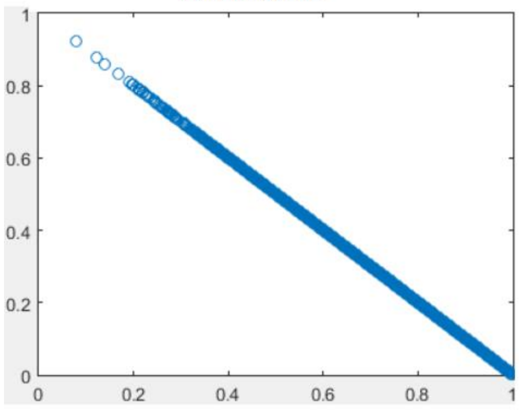
UF6.Extracted PF



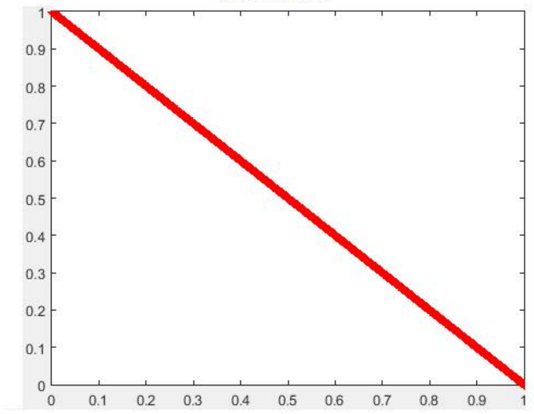
UF6.PF-True



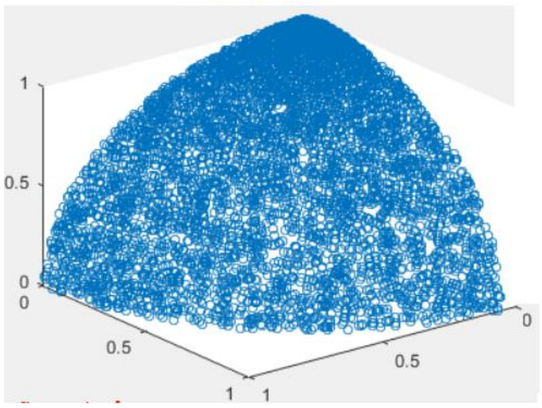
UF7.Extracted PF



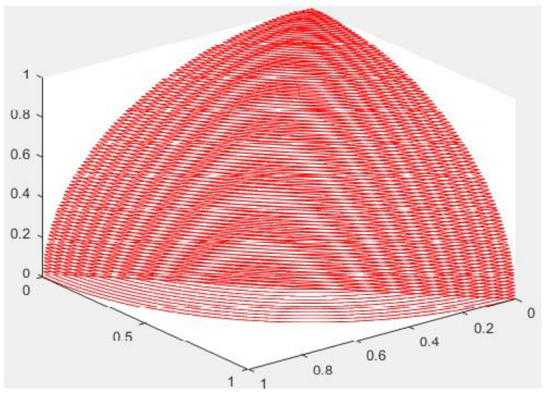
UF7.PF-True

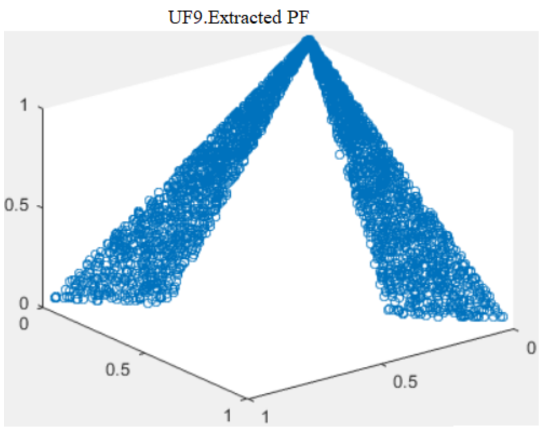


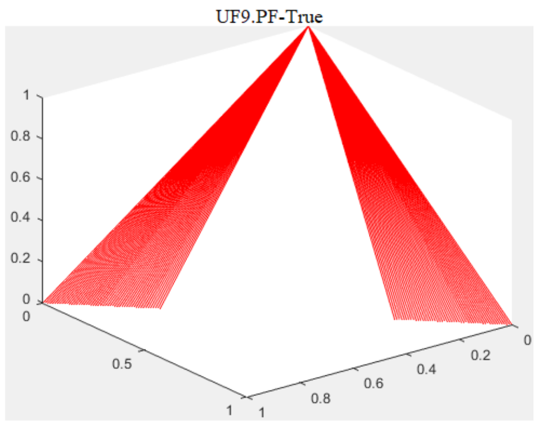
UF8.Extracted PF

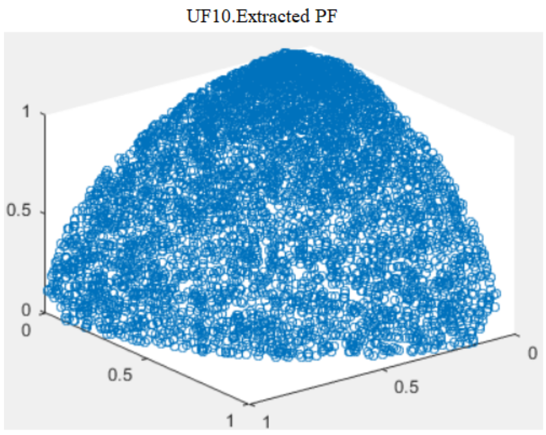


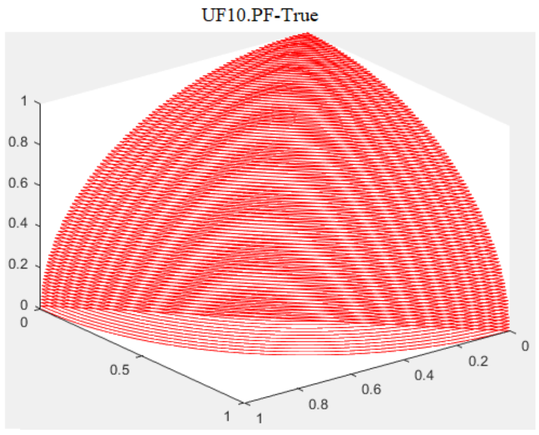
UF8.PF-True

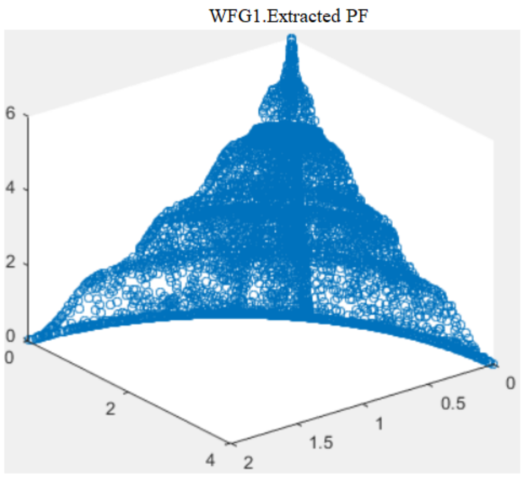


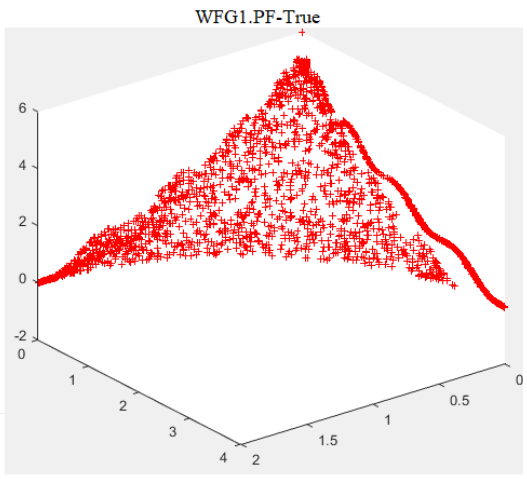

UF9.Extracted PF

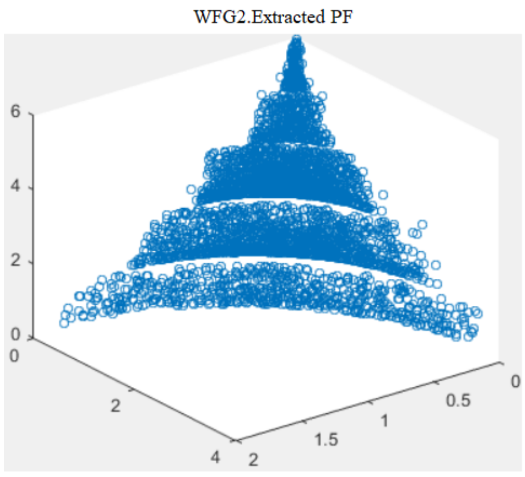

UF9.PF-True

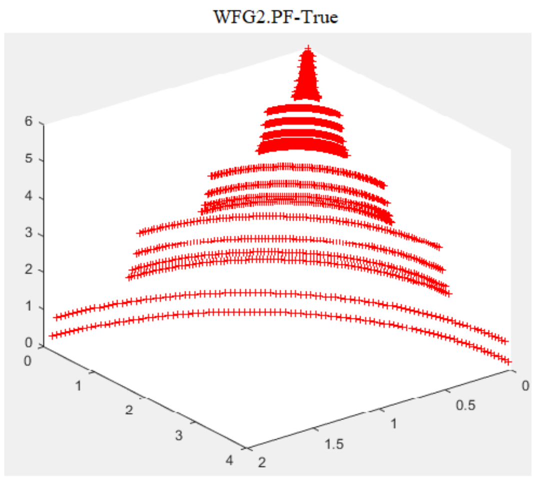

UF10.Extracted PF


UF10.PF-True

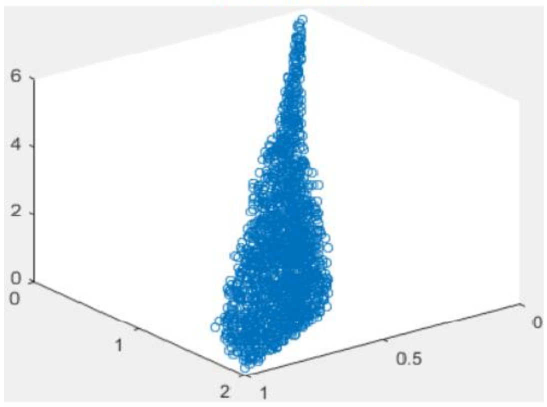

WFG1.Extracted PF


WFG1.PF-True

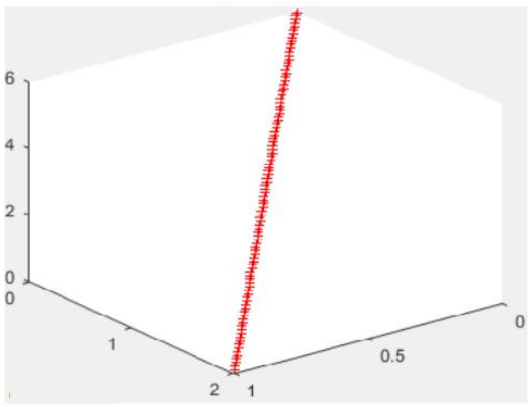

WFG2.Extracted PF


WFG2.PF-True

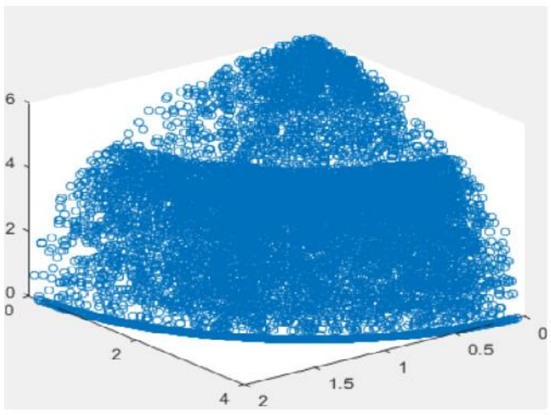
WFG3.Extracted PF



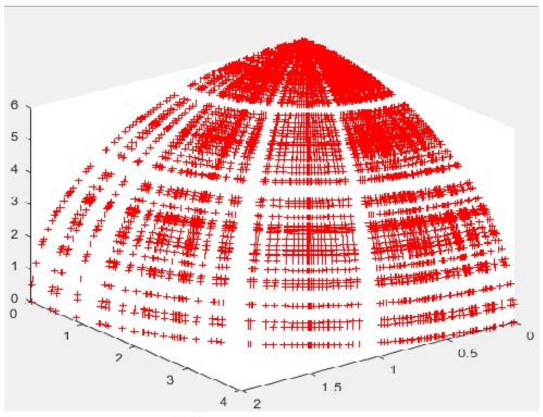
WFG3.PF-True



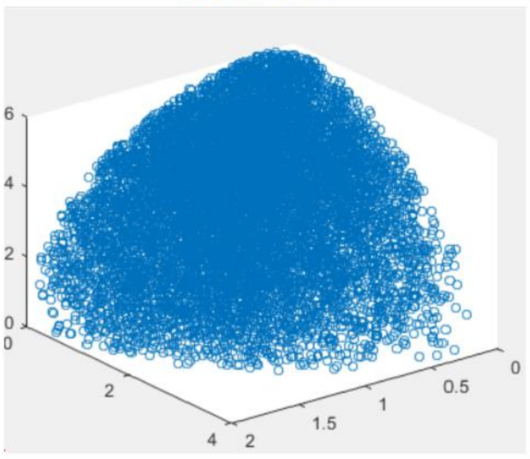
WFG4.Extracted PF



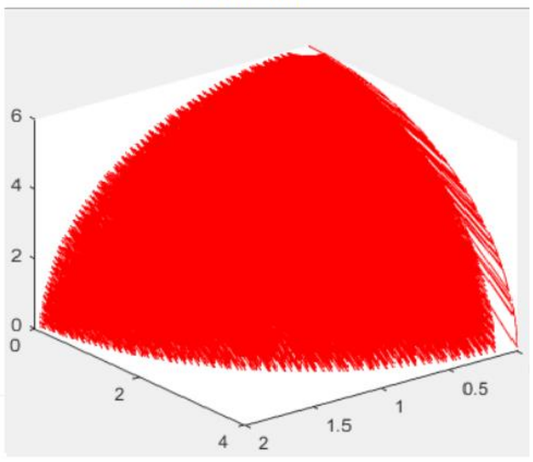
WFG4.PF-True



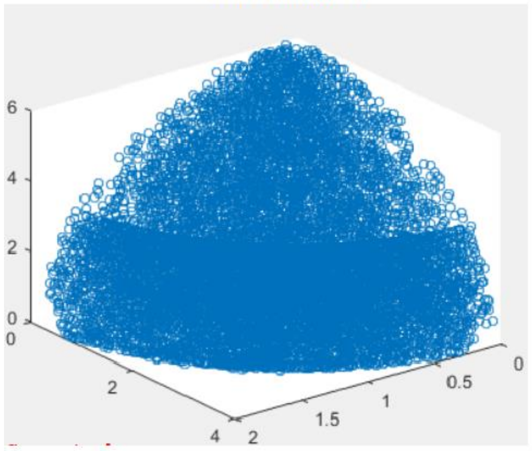
WFG5.Extracted PF



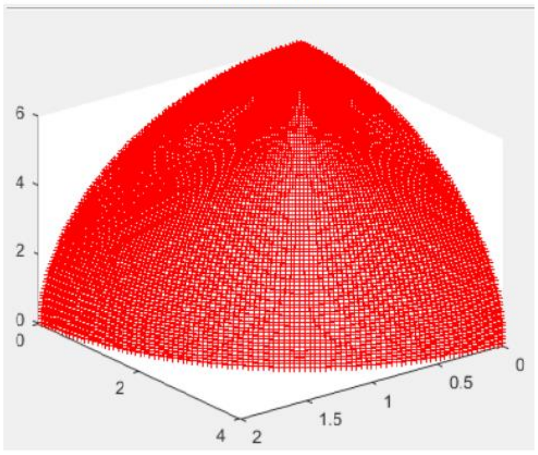
WFG5.PF-True



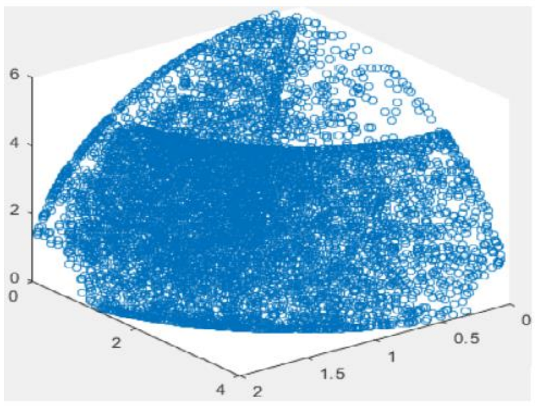
WFG6.Extracted PF



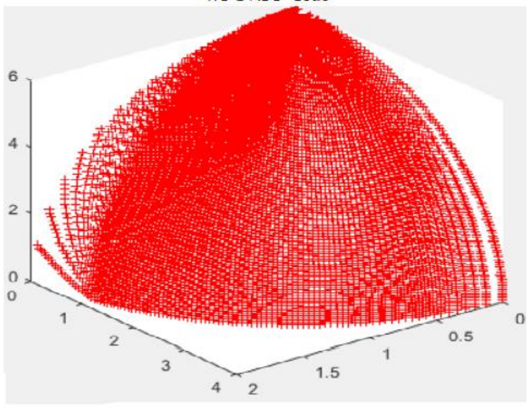
WFG6.PF-True



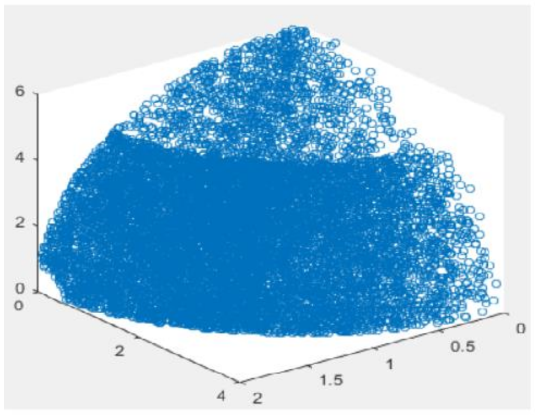
WFG7.Extracted PF



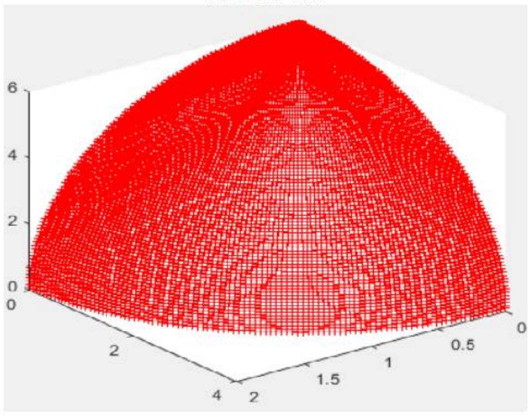
WFG7.PF-True



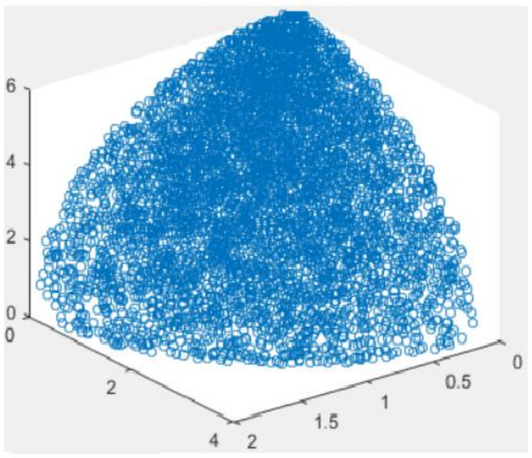
WFG8.Extracted PF



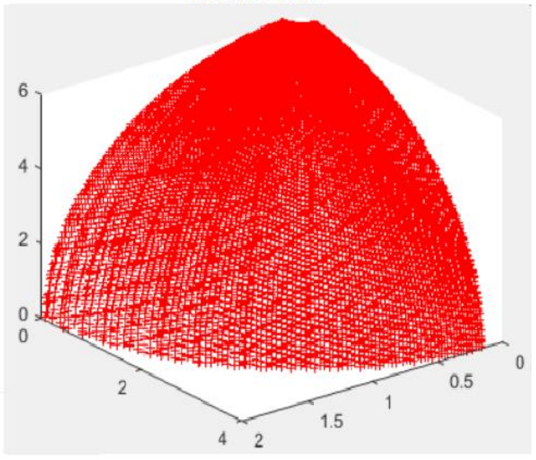
WFG8.PF-True



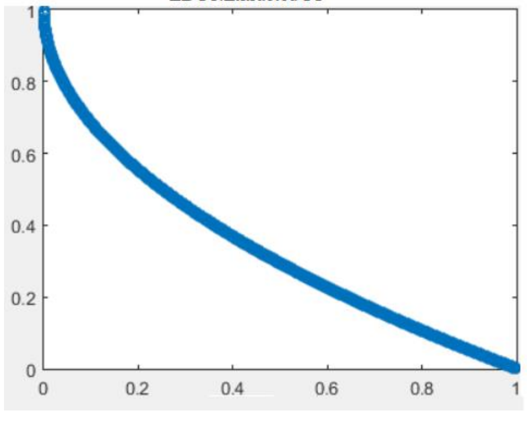
WFG9.Extracted PF



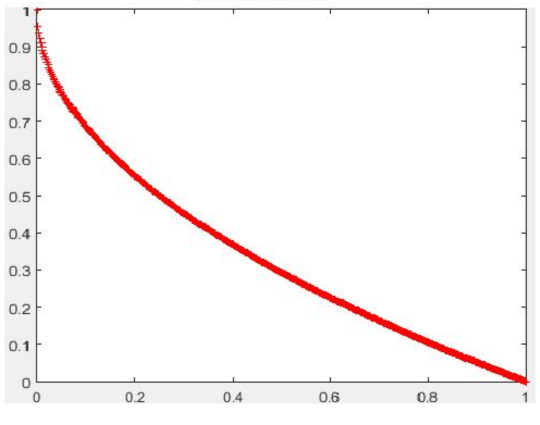
WFG9.PF-True

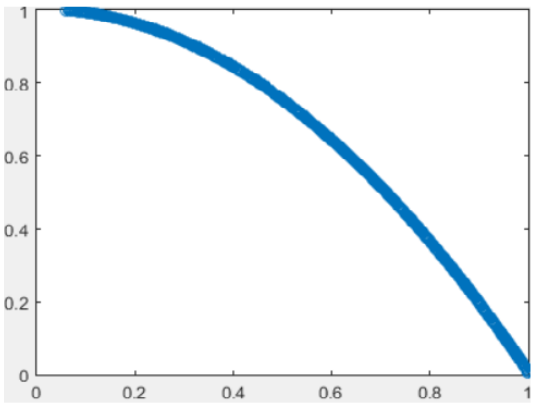


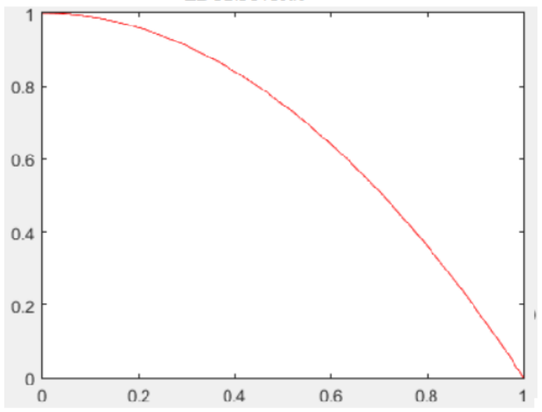
ZDT1.Extracted PF

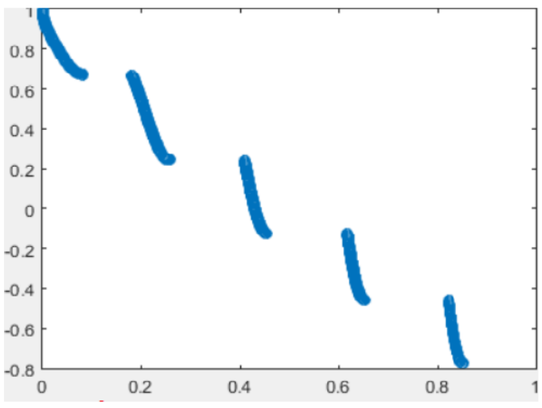


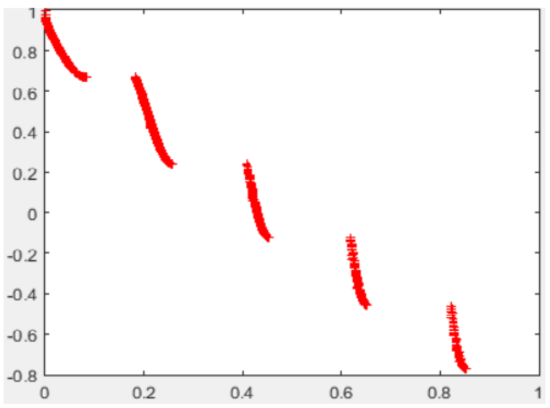
ZDT1.PF.True

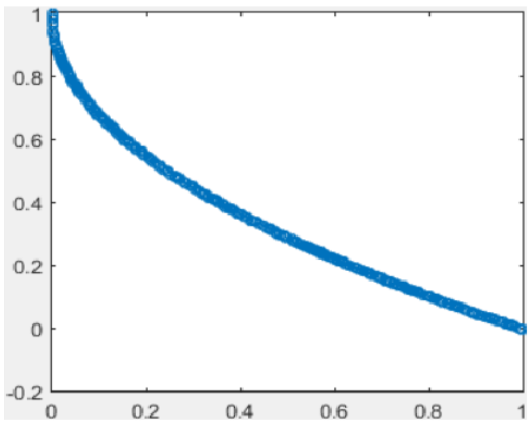


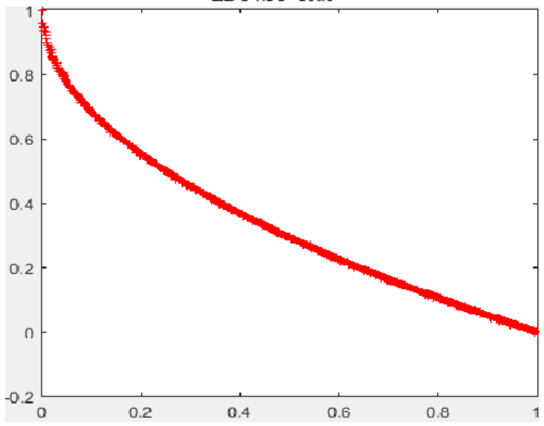

ZDT2.Extraxted PF

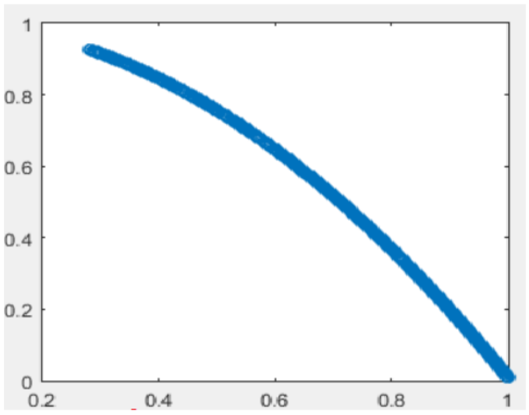

ZDT2.PF.True

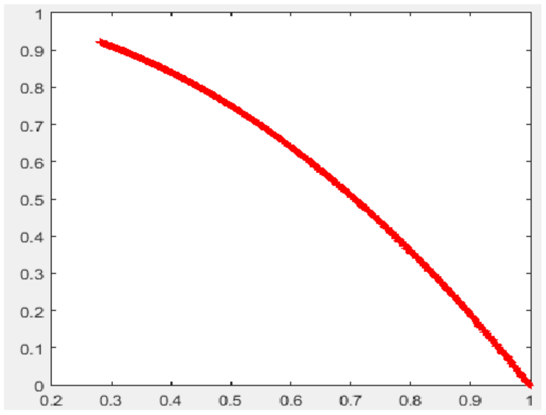

ZDT3.Extracted PF

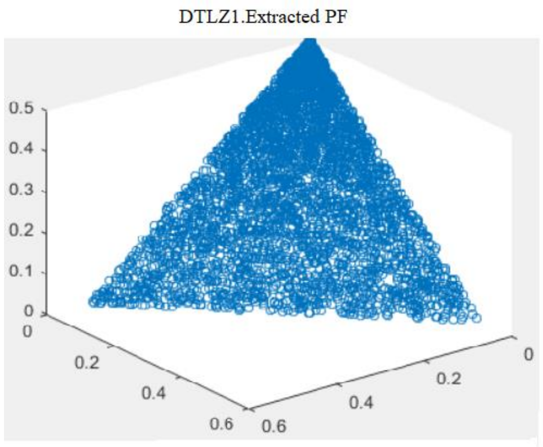

ZDT3.PF.True

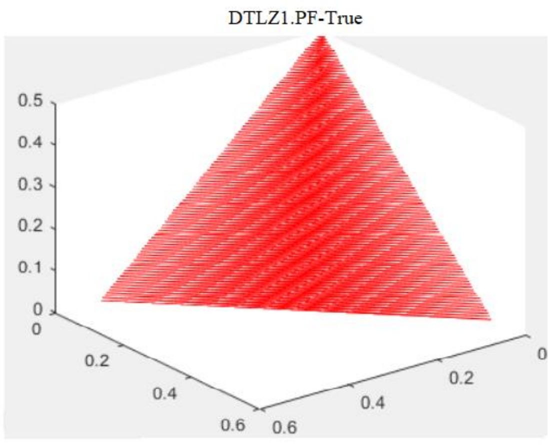

ZDT4.Extracted PF

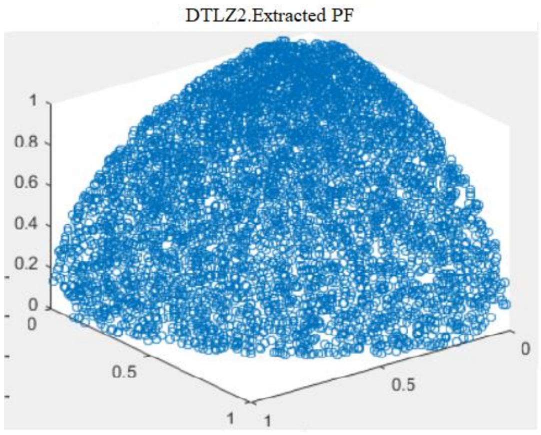

ZDT4.PF-True

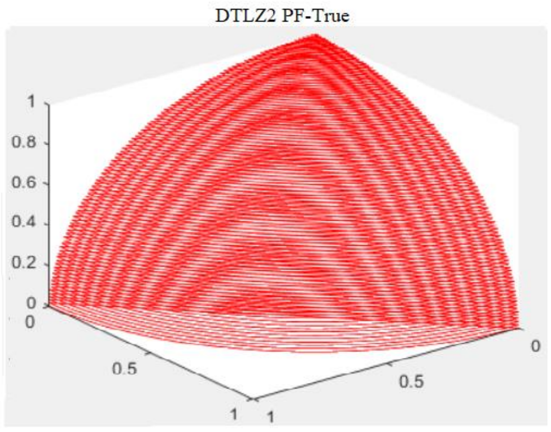

ZDT6.Extracted PF

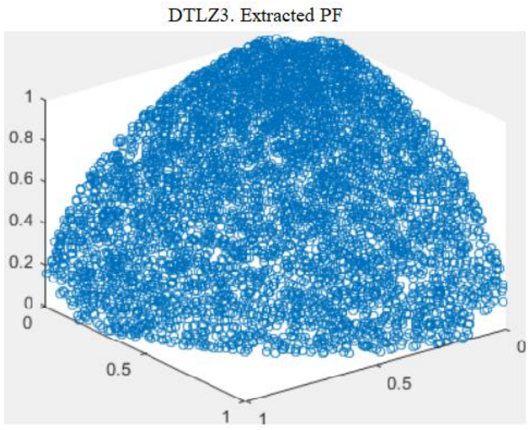

ZDT6.PF-True

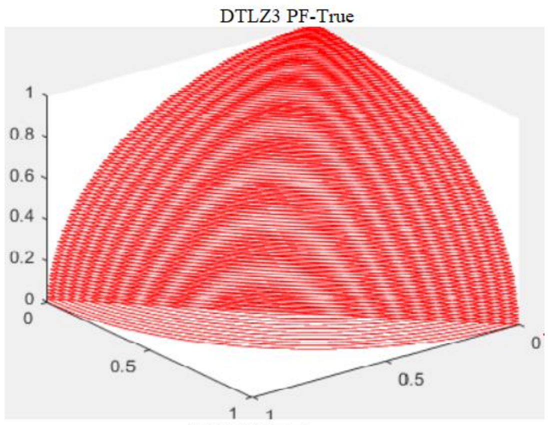

DTLZ1.Extracted PF

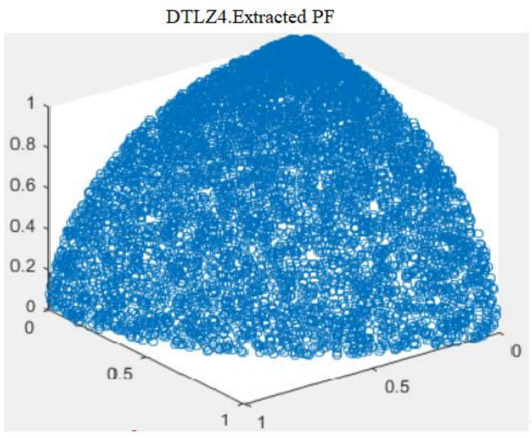

DTLZ1.PF-True

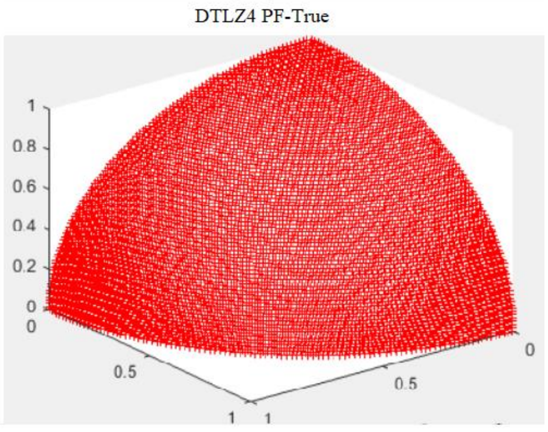

DTLZ2.Extracted PF


DTLZ2 PF-True


DTLZ3. Extracted PF


DTLZ3 PF-True


DTLZ4.Extracted PF


DTLZ4 PF-True

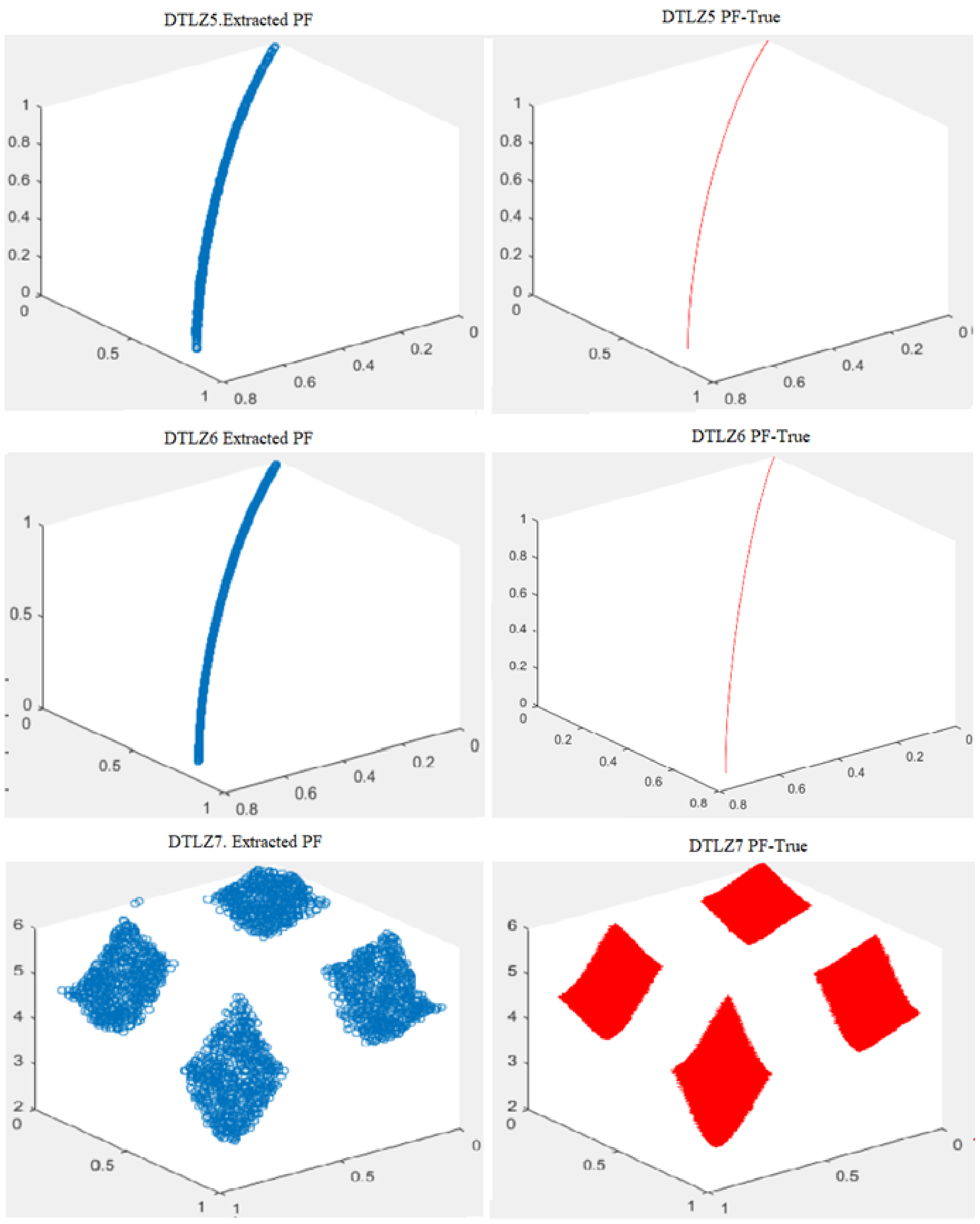


Figure 4.12: The Plots of Best Computed Pareto-Fronts and PF-True

Figure 4.12 illustrate the plots of computed Pareto-optimal set gained by BCO and Pareto Front True shared as a result of the competition .Plots present that the Pareto-optimal set found by Bee Algorithm quite close to PF-True and has a good spread.

4.4 CEC'18 Test Problems for Multi-objective problem

4.4.1 Definition

Multi-objective unconstrained test problems within this group. All test functions are minimization problems defined as follows:

$$\begin{aligned}
&Minimize \quad f_1 &&= r_1 \times a_1(x) \times \left(1 + g_1(X)\right)\\
&Minimize \quad f_2 &&= r_2 \times a_2(x) \times \left(1 + g_2(X)\right)\\
&\ldots\ \ldots\ \ldots. && \\
&Minimize \quad f_{M-1} &&= r_{M-1} \times a_{M-1}(x) \times \left(1 + g_{M-1}(X)\right)\\
&Minimize \quad f_M &&= r_M \times a_M(x) \times \left(1 + g_M(X)\right)
\end{aligned}$$

Where

- x= $(x_1,\ldots, x_N)$ is the decision vector in $[0,1]^N$, and $(F_1,\ldots, F_M)$ is the objective vector in R^M;
- r_i,i = 1,…, M, is the scaling factor for the $i-th$ objective function ;
- a_i,i = 1,…, M , is the shape function for the $i-th$ objective function ;
- $g_i \geq 0$,i = 1,…, M , is the distance function for the $i-th$ objective function ;

4.4.2. Result

Table 4.19: Min, Max, Average, Standard Deviation of IGD Values and Number of Function Evaluation of BCO in 30 Runs.

Function	Average	Min	Max	Std Dev	Function Evaluation
MaOP1	7.79E-03	8.28E-03	9.09E-03	2.35E-04	36180
MaOP2	7.42E-03	7.78E-03	8.63E-03	2.45E-04	36180
MaOP3	7.97E-03	8.27E-03	9.21E-03	2.90E-04	36180
MaOP4	7.45E-03	8.09E-03	8.55E-03	1.57E-04	36180
MaOP5	6.57E-03	7.00E-03	7.48E-03	1.66E-04	36180
MaOP6	6.58E-03	7.05E-03	7.67E-03	2.21E-04	36180
MaOP7	7.91E-03	8.27E-03	9.01E-03	2.26E-04	36180
MaOP8	7.87E-03	8.35E-03	9.05E-03	2.11E-04	36180
MaOP9	7.96E-03	8.61E-03	9.18E-03	1.98E-04	36180
MaOP10	7.79E-04	8.24E-04	8.99E-04	2.87E-05	36180

Table 4.19 describe that BCO is a robust and successful approach illustrated with small IGD scores and the standard deviation with small number of function Evaluation.

Table 4.20: IGD Values Obtained by BCO and it are 2 Competitors for MaOP1, MaOP2 and MaOP3.

Rank	MaOP1	IGD	MaOP2	IGD	MaOP3	IGD
1	**MOBCO**	7.79E-04	**MOBCO**	7.42E-04	**MOBCO**	7.97E-04
2	MOPSO	23.3509	MOPSO	14.446	MOPSO	22.669
3	MONSGA-II	6.66e+05	MONSGA-II	1.79e+03	MONSGA-II	28.855

Table 4.21: IGD Values Obtained by BCO and it are 2 Competitors for MaOP4, MaOP5 and MaOP6.

Rank	MaOP4	IGD	MaOP5	IGD	MaOP6	IGD
1	**MOBCO**	7.45E-04	**MOBCO**	6.57E-04	**MOBCO**	6.58E-04
2	MOPSO	247.93	MOPSO	2.16e+04	MOPSO	75.213
3	MONSGA-II	2.45e+05	MONSGA-II	2.28e+05	MONSGA-II	2.56e+04

Table 4.22: IGD Values Obtained by BCO and it are 2 Competitors for MaOP7, MaOP8 and MaOP9.

Rank	MaOP7	IGD	MaOP8	IGD	MaOP9	IGD
1	**MOBCO**	7.91E-04	**MOBCO**	7.87E-04	**MOBCO**	7.96E-04
2	MOPSO	1.91e+03	MOPSO	7.20e+03	MOPSO	1.44e+03
3	MONSGA-II	1.14e+04	MONSGA-II	6.35e+04	MONSGA-II	1.50e+03

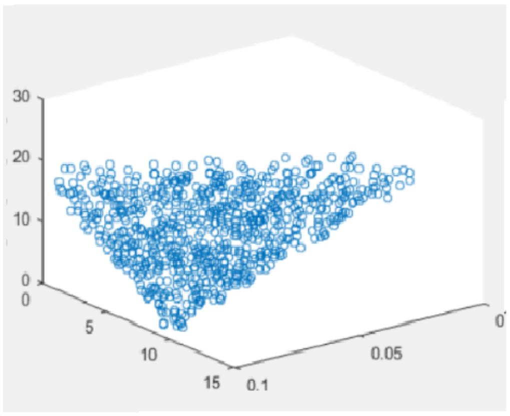
Table 4.23: IGD Values Obtained by BCO and it are 2 Competitors for MaOP10.

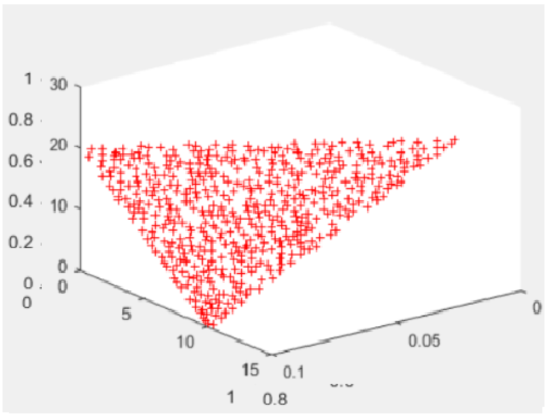
Rank	MaOP10	IGD
1	**MOBCO**	7.79E-04
2	MOPSO	1.20e+03
3	MONSGA-II	5.23e+06

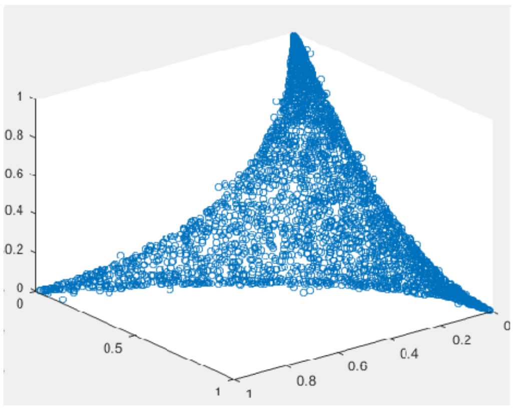
IGD results from MaOP1- MaOP10 test problems generated by BCO algorithms are listed in Tables 4.20, 4.21, 4.22 and 4.23, it is clearly shown that BCO achieves the best performance among its all other competitors in three objective function problems.

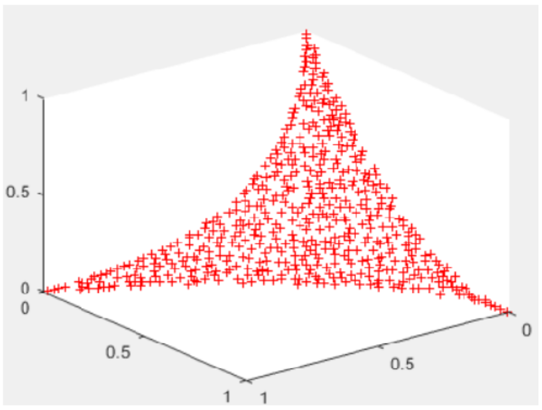
Table 4.11 and Table 4.12 show that the MOBCO has been also the best algorithm obtaining the values in 10 Problems (MaOP1-MaOP10) .MOPSO may be the second best algorithm which out performs other algorithms on considerable number of problems. MONSGA-II the worst algorithm in terms of this indicator.

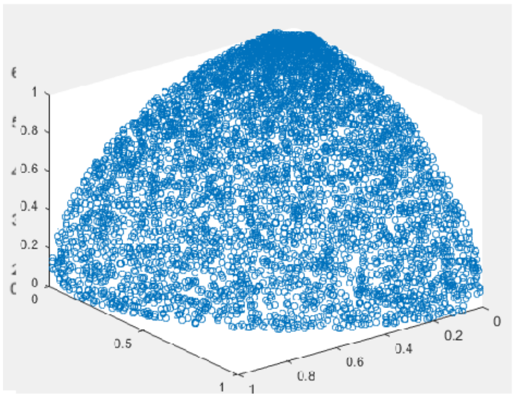
In addition, Tables illustrate the ranking of all three objective function problems MaOP1- MaOP10.From the tables can indicate that BCO performed more efficient than other algorithms in all problems and takes the first position in all test problems.

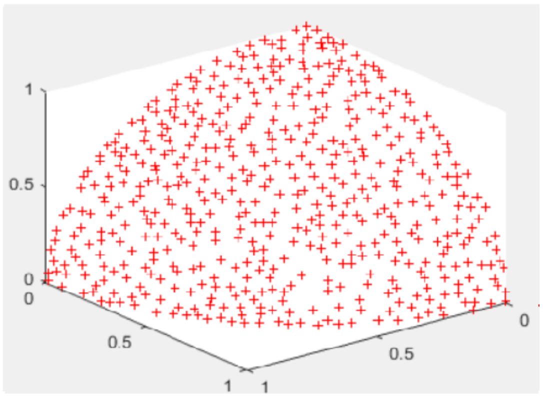

MaOP1 Extracted PF


MaOP1 PF-True

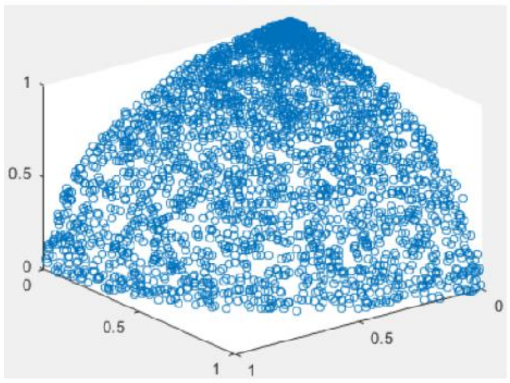

MaOP2 Extracted PF


MaOP2 PF-True

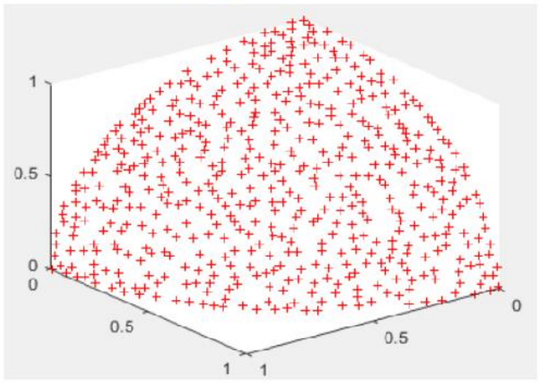

MaOP3 Extracted PF


MaOP3 PF-True

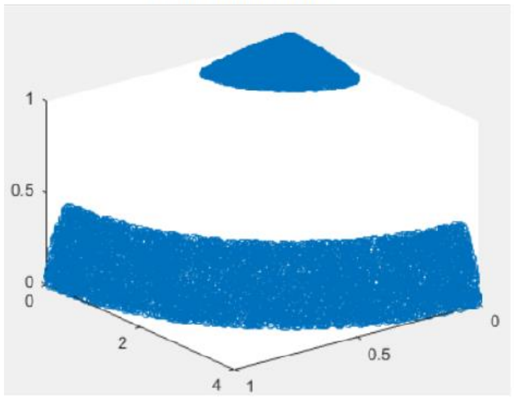
MaOP4 Extracted PF



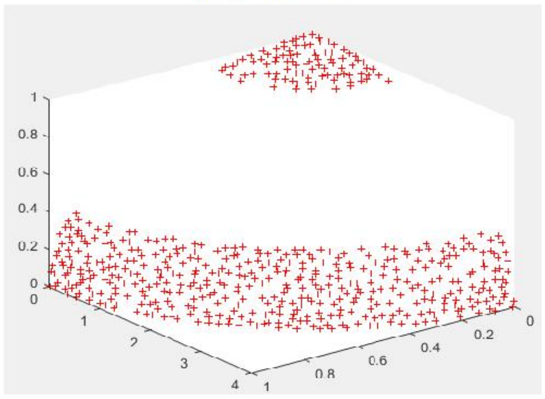
MaOP4 PF-True



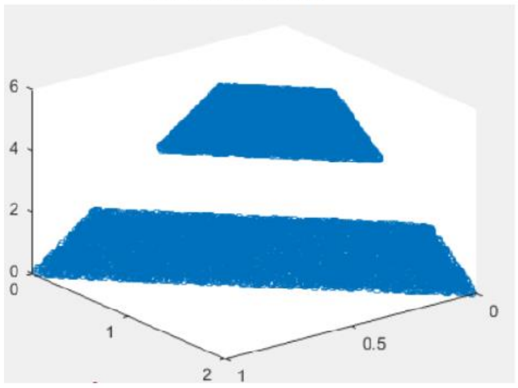
MaOP5 Extracted PF



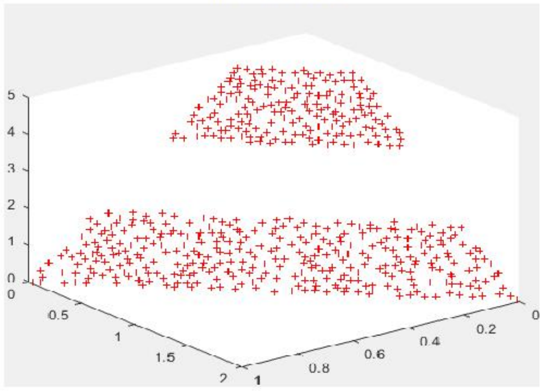
MaOP5 PF-True



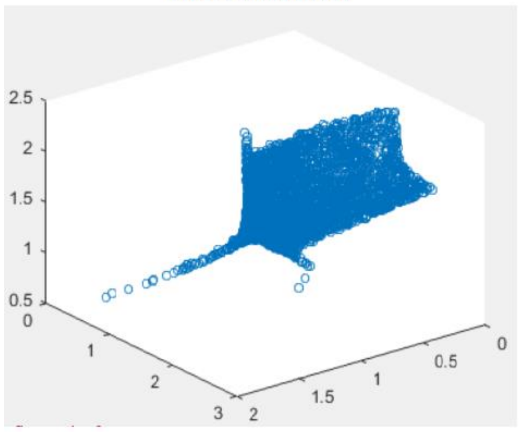
MaOP6 Extracted PF



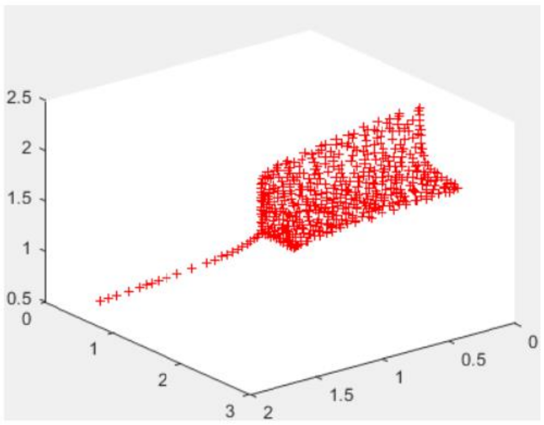
MaOP6 PF-True



MaOP7 Extracted PF



MaOP7 PF-true



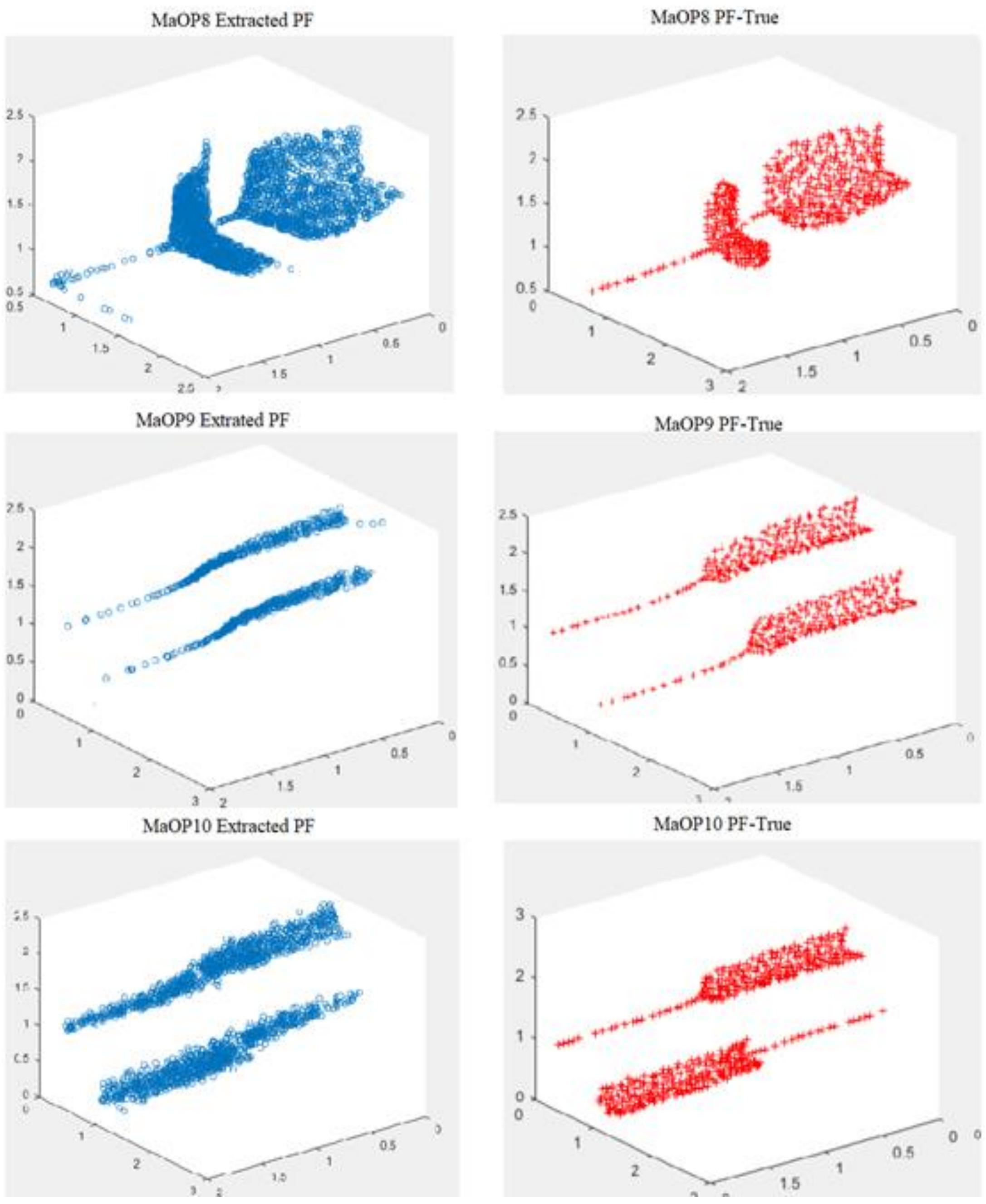


Figure 4.2: The Plots of Best Computed Pareto-Fronts and PF-True

Chapter 5

CONCLUSION

The Bee Colony Optimization (BCO) algorithm, one of several Swarm Intelligence techniques, is a meta heuristic approach, driven by the action of foraging honeybees.

It presents a general algorithmic framework pertinent to a variety of optimization difficulties in the areas of management, engineering and control, to name a few, and should always be tailor-made for any particular function. BCO is founded on the conceptual model of collaboration; increasing the adequacy of artificial bees.

BCO is capable of intensifying searches within the favorable areas of the 'solution-space' by way of information reciprocate and the recruiting operation. The procedure of diversification is achieved through limiting the search within deferent runs.

An archive founded on ‘crowding distance’ is utilized in the proposed algorithm as a way to record all non-dominated solutions that are found. When there is no space in archive, a new individual (solutions) replaces the individual(solution) in the archive with the lowest crowding distance at the time of its addition in an effort maintain diversity. An enhanced ABC algorithm utilizing an elitism method is used to stop early convergence by selecting the elite (member in the least crowded region) and using it to generate new solution(food sources) in the ‘employed bee’ stage and the ‘onlooker bee’ stage.

Experiments conducted on n-dimension uni-modal and multi-modal functions have discovered that the Bee Colony Algorithm is significantly robust and agile with a 100% success rate. The algorithm has also proven minimum and maximum convergence without getting stuck at local optima. The bee algorithm has been found to be faster and more accurate compared to results gotten from other techniques.

REFERENCES

[1] V. Tereshko and A. Loengarov, *Collective Decision-Making in Honey Bee Foraging Dynamics*. Computing and Information Systems Journal, ISSN 1352-9404, 2005.

[2] G. Beni and J. Wang, *Swarm intelligence*. Tokyo: Proc. Seventh Annual Meeting of the Robotics Society of Japan, ISSN 425–428, 1989.

[3] S.Luku, *Essentials of metaheuristics*, 2nd ed. http://cs.gmu.edu/sean/book /metaheuristics, 2014.

[4] D. Bertsimas and j. Tsitsiklis, *Simulated Annealing*. ISSN 10-15, 1993.

[5] G. Dueck, *New optimization heuristics: the great deluge algorithm and the record to record travel*. Journal of Computational physics. ISSN 86-92, 1993.

[6] M. Naeem, S. Xue and D. Lee, *Cross-entropy optimization for sensor selection problems*. Communications and information technology. ISCIT 396-401, 2009.

[7] R. Chelouah and P. Siarry, *Tabu search applied to global optimization*. European Journal of Operational Research. ISSN 256-270, 2000.

[8] D. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Boston.USA: Addison-Wesley Longman Publishing Co, 1989.

[9] M. Dorigo and D. Caro, *The ant colony optimization-heuristics*. New York. McGraw: New Ideas in Optimization .ISSN 11-32, 1999.

[10] I. Kennedy and R. Eberhart, *Particle Swarm optimization*. IEEE international conference on neural networks.1942-1948, 1995.

[11] K. Price, *An Introduction to Differential Evolution*. McGraw-Hill. London.: New Ideas in Optimization, 1997.

[12] R. Storn and K. Price, *Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Space*. Journal of Global Optimization .341-359, 1997.

[13] S. Sivanandam and S. Deepa, *Introduction to genetic algorithms*. Berlin.Germany: Springer verlog berlin Heidelberg, 2008.

[14] R. Haupt and S. Haupt, *Practical genetic algorithms*. New Jersey: John wiley and sons, 2004.

[15] R. Caruana and L. Eshelman, *Representation and Hidden Bias II: Eliminating Defining Length Bias in Genetic Search via Shuffle Crossover*. IJCAI. 750-755, 1995.

[16] D. Karaboga, An Idea Based On Honey Bee Swarm For Numerical Optimization, Technical Report-TR06, Erciyes University, Engineering Faculty, Computer Engineering Department, 2005.

[17] B. Basturk, D.Karaboga, An Artificial Bee Colony (ABC) Algorithm for Numeric function Optimization, IEEE Swarm Intelligence Symposium 2006, May 12-14, 2006.

[18] C. Lamont and G. Van Veldhuizen, *Evolutionary Algorithms For Solving Multi-objective Problems*, 2nd ed. Springer, 2007.

[19] D. Tatjana, T. Duˇsan and S. Milica, *BEE COLONY OPTIMIZATION PART I: THE ALGORITHM OVERVIEW*. 2014.

[20] M. Nikoli´c and D. Teodorovi´c, *Empirical study of the bee colony optimization (BCO) algorithm,Expert Systems with Applications*. 4609–4620, 2013.

[21] M. Nikoli´c and D. Teodorovi´c, *Transit network design by bee colony optimization,Expert Systems with Applications*. 5945–5955, 2013.

[22] Davidovi´c, T., Jakˇsi´c, T., Ramljak, D., ˇSelmi´c, M., and Teodorovi´c, D., ”MPI parallelization strategies for bee colony optimization”, *Optimization, Special Issue entitled “Advances in Discrete Optimization”*, 62(8) (2013) 1113–1142,2011.

[23] P. Lu, J. Zhou, H. Zhang, R. Zhang and C. Wang, *Chaotic differential bee colony optimization algorithm for dynamic economic dispatch problem with valve-point effects*. International Journal of Electrical Power & Energy Systems, 2014.

[24] M. Kefayat, A. Lashkar Ara and S. Nabavi Niaki, *A hybrid of ant colony optimization and artificial bee colony algorithm for probabilistic optimal placement and sizing of distributed energy resources*. Energy Conversion and Management, 2015.

[25] P. Justesen, C. Pedersen and R. Ursem, *Multi-objective Optimization using Evolutionary Algorithms*. Denmark, 2009.

[26] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms: An Introduction*. India: Indian Institute of Technology Kanpur, 2011.

[27] J. Knowles, L. Thiele, and E. Zitzler. A Tutorial on the Performance Assessment of Stochastic Multiobjective Optimizers. In: TIK Report 214, Computer Engineering and Networks Laboratory (TIK), ETH Zurich, 2006.

[28] J. Handl, J. Knowles, An Evolutionary Approach to Multiobjective Clustering. In: IEEE Transactions on Evolutionary Computation, volume 11, 2007.

[29] Y. Xianga, Y. Zhoua and H. Liuc, *An elitism based multi-objective artificial bee colony algorithm*. China: European Journal of Operational Research, 2015.

[30] R. Akbari, R. Hedayatzadeh, K. Ziarati and B. Hassanizadeh, *A multi-objective artificial bee colony algorithm*. Swarm and Evolutionary Computation, 2012.

[31] H. Liu, L. Gu and Q. Zhang, *Decomposition of a multi objective optimization problem into a number of simple multi objective sub problems*. IEEE Transactions on Evolutionary Computation, 2014.

[32] E. Zitzler, M. Laumanns, and L. Thiele, “SPEA2: improving the strength Pareto evolutionary algorithm,” Tech. Rep., Computer Engineering and Networks Laboratory _TIK_, Swiss Federal Institute of Technology _ETH_, Zurich, Switzerland, May 2001.

[33] J. D. Knowles and D. W. Corne, “Approximating the nondominated front using the Pareto Archived Evolution Strategy,” *Evolutionary computation*, vol. 8, no. 2, pp. 149–172, 2000.

[34] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar, “Comprehensive learning particle swarm optimizer for global optimization of multimodal functions,” *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 3, pp. 281–295, 2006.

[35] K. Deb, A. Pratap, S. Agarwal and T. Meyarivan, *A fast and elitist multiobjective genetic algorithm: NSGA-II*. IEEE Transactions on Evolutionary Computation,182–197, 2002.

[36] D. Teodorović, T. Davidović and M. Šelmić, *Bee Colony Optimization Overview*. submitted for publication, 2010.

[37] Y. Xiang, Y. Zhou and H. Liu, *An elitism based multi-objective artificial bee colony algorithm*. European Journal of Operational Research.168-193, 2015.