APPLICATION OF SWARM INTELLIGENCE OPTIMIZATION ON BIO-PROCESS PROBLEMS

MOHAMAD ZIHIN BIN MOHD ZAIN

FACULTY OF ENGINEERING
UNIVERSITY OF MALAYA
KUALA LUMPUR

2018

APLLICATION OF SWARM INTELLIGENCE OPTIMIZATION ON BIO-PROCESS PROBLEMS

MOHAMAD ZIHIN BIN MOHD ZAIN

DISSERTATION SUBMITTED IN FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF ENGINEERING SCIENCE

FACULTY OF ENGINEERING
UNIVERSITY OF MALAYA
KUALA LUMPUR

2018

UNIVERSITY OF MALAYA

ORIGINAL LITERARY WORK DECLARATION

Name of Candidate: Mohamad Zihin bin Mohd Zain

Matric No: KGA140054

Name of Degree: Master of Engineering Science

Title of Dissertation: Application of Swarm Intelligence Optimization on Bio-process Problems.

Field of Study: Computer / Data networks

I do solemnly and sincerely declare that:

(1) I am the sole author/writer of this Work;
(2) This Work is original;
(3) Any use of any work in which copyright exists was done by way of fair dealing and for permitted purposes and any excerpt or extract from, or reference to or reproduction of any copyright work has been disclosed expressly and sufficiently and the title of the Work and its authorship have been acknowledged in this Work;
(4) I do not have any actual knowledge nor do I ought reasonably to know that the making of this work constitutes an infringement of any copyright work;
(5) I hereby assign all and every rights in the copyright to this Work to the University of Malaya ("UM"), who henceforth shall be owner of the copyright in this Work and that any reproduction or use in any form or by any means whatsoever is prohibited without the written consent of UM having been first had and obtained;
(6) I am fully aware that if in the course of making this Work I have infringed any copyright whether intentionally or otherwise, I may be subject to legal action or any other action as may be determined by UM.

Candidate's Signature Date:

Subscribed and solemnly declared before,

Witness's Signature Date:

Name:

Designation:

ABSTRACT

An improved version of Differential Evolution (DE) namely Backtracking Search Algorithm (BSA) is applied to several fed batch fermentation problems and its performance is compared with recent emerging metaheuristics such as Artificial Algae Algorithm (AAA), Artificial Bee Colony (ABC), Covariance Matrix Adaptation Evolution Strategy (CMAES) and DE. Also, fed batch fermentation problems in winery wastewater treatment and biogas generation from sewage sludge are developed for optimization. Though DE traditionally performs better than other evolutionary algorithms and swarm intelligence techniques in optimization of fed-batch fermentation, BSA edged DE and other recent metaheuristics to emerge as superior optimization method in this work. BSA gave the best overall performance by showing improved solutions and more robust convergence in comparison with various metaheuristics used in this work. Multi-objective optimization problems are also addressed by proposing a modified multi-criterion optimization algorithm based on a Pareto-based Particle Swarm Optimization (PSO) algorithm called Multi-Objective Particle Swarm Optimization (MOPSO). This modified algorithm called Modified Multi-Objective Particle Swarm Optimization (M-MOPSO) employs a fixed-sized external archive along with a dynamic boundary-based search mechanism to evolve the population. The proposed method is tested on 10 multi-objective benchmark problems of CEC 2009 and compared with four metaheuristics: Multi-Objective Grey Wolf Optimizer (MOGWO), Multi-Objective Evolutionary Algorithm Based on Decomposition (MOEA/D), Multi-Objective Differential Evolution (MODE) and MOPSO. Two multi-objective fed-batch models are also used as case studies to verify the performance of the proposed algorithm. Our method emerged highly competitive when compared with other algorithms based on their qualitative and quantitative results.

ABSTRAK

Versi penambahbaikan Differential Evolution (DE) yang dipanggil sebagai Backtracking Search Algorithm (BSA) diaplikasikan kepada beberapa masalah penapaian fed batch dan prestasinya dibandingkan dengan metaheuristic-metaheuristic terkini seperti Artificial Algae Algorithm (AAA), Artificial Bee Colony (ABC), Covariance Matrix Adaptation Evolution Strategy (CMAES) dan DE. Masalah-masalah penapaian fed batch di dalam rawatan sisa air wain dan penjanaan biogas daripada kumbahan enapcemar juga dibangunkan untuk pengoptimuman. Walaupun DE secara tradisinya mempunyai prestasi yang lebih baik daripada lain-lain algorithma evolusi dan teknik kecerdasan swarm dalam pengoptimuman penapaian fed batch, BSA telah mengatasi DE dan lain-lain metaheuristic untuk tampil sebagai kaedah pengoptimuman terbaik dalam kajian ini. BSA telah memberikan prestasi kesuluruhan terbaik dengan menunjukkan penyelesaian yang lebih baik dan penumpuan yang lebih teguh berbanding lain-lain metaheuristic yang digunakan dalam kajian ini. Masalah pengoptimuman pelbagai objektif juga telah ditumpukan dengan mencadangkan satu algoritma pengoptimuman pelbagai kriteria yang diubahsuai berdasarkan daripada Particle Swarm Optimization (PSO) algoritma yang berasaskan Pareto yang dipanggil sebagai Multi-Objective Particle Swarm Optimization (MOPSO). Algoritma yang diubahsuai ini yang dipanggil sebagai Modified Multi-Objective Particle Swarm Optimization (M-MOPSO) menggunakan arkib luaran bersaiz tetap disamping mekanisma carian berasaskan sempadan yang dinamik untuk mengevolusikan populasi. Kaedah yang dicadangkan diuji dengan 10 masalah penanda aras pelbagai objektif CEC 2009 dan dibandingkan dengan empat metaheuristic: Multi-Objective Grey Wolf Optimizer (MOGWO), Multi-Objective Evolutionary Algorithm Based on Decomposition (MOEA/D) Multi-Objective Differential Evolution (MODE) dan MOPSO. Dua model fed-batch pelbagai objektif juga digunakan sebagai kes pengajian

untuk mengesahkan prestasi algoritma yang dicadangkan. Kaedah kami tampil berdaya saing tinggi apabila dibandingkan dengan algoritma-algoritma lain berdasarkan kepada keputusan kualitatif dan kuantitatif.

ACKNOWLEDGEMENTS

The submission of this thesis was made possible due to the contributions of various people. I would like to take this opportunity to express my appreciation and gratitude to my supervisors Associate Prof. Dr. Jeevan A/L Kanesan and Ir. Dr. Chuah Joon Huang, for their support, inspiration and valuable suggestions throughout this project. I would like to extend special thanks to Prof. Graham Kendall and Associate Prof. Hernan Aguirre for their consultation and sharing of expertise in the field of evolutionary computation and optimization. I would also like to thank University of Malaya for the financial support through University of Malaya Research Grant (UMRG) RG 333-15AFR. I would also like to thank all members in Expert Systems and Optimization research group. I would like to express an utmost appreciation to my parents Mohd Zain bin Abd Jalil and Zarina bt Omar and also my family for their affection and encouragement.

Author

Mohamad Zihin bin Mohd Zain

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS AND ABBREVIATIONS

t_f	:	Final time
AAA	:	Artificial Algae Algorithm
ABC	:	Artificial Bee Colony Optimization
BSA	:	Backtracking Search Optimization Algorithm
CMA-ES	:	Covariance Matrix Adaptation Evolution Strategy
COD	:	Chemical oxygen demand
CS	:	Cuckoo Search
DE	:	Differential evolution
EA	:	Evolutionary algorithms
FA	:	Firefly Algorithm
FE	:	Function evaluation
GWO	:	Grey wolf optimizer
IGD	:	Inverted Generational Distance
M-MOPSO	:	Modified multi-objective particle swarm optimization
MODE	:	Multi-objective differential evolution
MOEA/D	:	Multi-objective evolutionary algorithm based on decomposition
MOGWO	:	Multi-objective grey wolf optimizer
MOM	:	Multi-objective metaheuristic
MOPSO	:	Multi-objective particle swarm optimization
MS	:	Maximum Spread
NSGA	:	Non-dominated Sorting Genetic Algorithm
ODE	:	Ordinary differential equation
PI	:	Performance index
PSO	:	Particle Swarm Optimization

SOM	:	Single-objective metaheuristic
SP	:	Spacing
SS	:	Sewage Sludge
WWTP	:	Waste Water Treatment Plant
CH_4	:	Methane
CO_2	:	Carbon dioxide
POP	:	Population
REP	:	Repository

LIST OF APPENDICES

CHAPTER 1: INTRODUCTIONS

Optimization is one of the most important research areas in applied mathematics. The diverse applications of optimization which range from manufacturing and engineering to business and medication have attracted many researchers to explore the field. The field of biotechnology contains many problems that can take advantage of the optimization process. One such problem is the fermentation problem. The crucial factors in the development and optimization of fermentation processes are the quality and quantity of the products, which can be improved at the cultivation levels. Traditionally, fermentation processes is done in batch mode, where an amount of substrate is fed only once at the beginning of the fermentation. This is in contrast to fed-batch mode, where the substrate is fed in a controlled amount during a set interval of time. In fed-batch fermentation, nutrient feeding along the process enhances higher product concentrations. Controlled nutrient feeding increases biomass in controlled manner and this improves product concentrations with less impact of product and/or nutrient inhibition of biomass. This complex nature of fed-batch fermentation encourages optimization method development that predicts optimal feeding profile to enhance the process performance. In order to obtain proper simulation of the process, usually differential equations that model the mass balances of various state variables are developed.

Metaheuristic is one of the means to solve optimization problems. It is a process of trial and error to discover the solution of a problem and consists of certain trade-off of randomization and local search. One of the most appealing characteristic of metaheuristic is that it uses derivation-free mechanisms and is stochastic in nature. This enables faster convergence and less expensive computation as compared to deterministic method. In optimization, a problem may consist of either one objective or

more than one. For the problem with one objective, we call it single-objective problem. For the problem with two or three objectives, we call it multi-objective problem. For the problem with more than three objectives, we call it many-objective problem. Consequently, a metaheuristic is developed to solve a particular type of problem. A metaheuristic that is used to solve a single-objective problem is called single-objective metaheuristic (SOM) while a metaheuristic that is used to solve a multi-objective problem is called multi-objective metaheuristic (MOM).

In fermentation or bioprocess problems, the input feeding profile or substrate feed rate is considered a key variable. Metaheuristic is considered as the most suitable optimization strategy to be used. This is because complexity involved in analytical approaches will increases with the increasing number of state and control variables. Deterministic algorithms also have a high computational overhead as well as have a tendency of premature convergence towards local optima.

One of the attributes of most real-world engineering is that they often have multiple conflicting goals. These multiple objectives may provide certain trade-offs which result in numerous solutions to be chosen from. From these solutions, it is up to the decision makers to choose one of the solutions to suit their needs. In contrast to a single-objective optimization problem where the optimal solution is clearly defined, there is no direct way to define the superiority of one solution compared to another in a multi-objective problem. One of the ways to solve this type of problem is by using the concepts of Pareto dominance and Pareto-optimality where there exists more than one 'optimal solutions'.

To understand the concept of Pareto dominance, consider a multi-objective optimization in a problem with two or three objective functions below:

$$Minimize: F(X) = f_1(X), f_2(X), \dots, f_G(X), \quad (1)$$

$$Subject\ to: R_i^{lower} \leq x_i \leq R_i^{upper}, i = 1, 2, \dots, d \quad (2)$$

where d is the number of variables, G is the number of objective functions, and $[R_i^{lower}, R_i^{upper}]$ are the boundaries of ith variables. In Pareto dominance, given that there are two candidate solutions: $Y = (y_1, y_2, \dots, y_d)$ and $Z = (z_1, z_2, \dots, z_d)$, vector Y dominates vector Z (denote as $Y \succ Z$) if and only if,

$$f_g(Y) \leq f_g(Z), \forall g \in \{1, \dots, G\} \quad (3)$$

$$f_g(Y) < f_g(Z), \forall g \exists \{1, \dots, G\} \quad (4)$$

If solution Y is not dominated by any other solutions, then Y is declared as a nondominated or Pareto optimal solution. There are no superior solutions to the problem than Y, although there may be other equally good solutions. On the other hand, a solution $Y \in X$ is called Pareto-optimal if and only if,

$$\nexists Z \in X | f(Z) \succ f(Y) \quad (5)$$

The set of solutions that satisfy (5) is known as the Pareto optimal set and the fitness values corresponding to these solutions form the Pareto front or trade-off surface in objective space.

1.1 Problem Statement

Stochastic algorithms or metaheuristics have been previously applied on various bioprocess optimization problems. A recent study shows differential evolution (DE) (Storn & Price, 1997) is a better solution for bio-process applications (Banga et al.,

2004). However, a new algorithm called the Backtracking Search Optimization Algorithm (BSA) was recently proposed by Civicioglu (2013). BSA was developed based on DE and has many elements similar to DE. However, it improved upon DE by incorporating new elements such as improved mutation and crossover operators and the utilization of a dual population. BSA also has only one control parameter compared to DE which requires two parameters for fine-tuning. With these improvements, it is expected that BSA will perform better than DE. Since DE is known to be efficient in solving fermentation problems (Banga et al., 2004; Da Ros et al., 2013; M. Rocha et al., 2014), BSA as a recent DE-based metaheuristic is proposed in this paper and we investigate various fermentation problems. Our hypothesis is that it will perform better compared to other stochastic algorithms. BSA, being a powerful evolutionary algorithm, is a suitable algorithm to be used in searching for optimal control profiles for the complex bioreactor chemical process.

In fermentation and bioprocess technology, the utilization of fed-batch operation is considered common. In biological wastewater treatment however, batch mode is still dominantly used and fed-batch is regarded as a relatively new technique (Montalvo et al., 2010). In a basic process of fed-batch wastewater treatment, the wastewater is fed slowly into the aerated bioreactor. During this process, the effluent is never removed until after the operating volume of the bioreactor is mostly filled. This enabled reduction of inhibitory or toxic effects through the dilution of highly concentrated toxic compounds in an aeration based large volume tank. This results in greater chemical oxygen demand (COD) removal rate and smaller required reactor volume. The aeration tank is emptied when it is almost full and the process is repeated.

The disposal of sludge is one of the major problems in municipal wastewater treatment, and constitutes up to half of the operating costs of a Waste Water Treatment

Plant (WWTP) (J. Baeyens et al., 1997). Though different methods for sludge disposal exist, anaerobic digestion is one of the preferred routes as it is not limited to the production of biogas from waste, but also lower the amount of final sludge solids for disposal and curtailing odour problems (Appels et al., 2008), resulting in cost reduction. This justifies the importance of anaerobic sludge digestion process in a modern WWTP. Nowadays, the potential of biogas as an energy source has gained plenty of recognition, with the majority of biogas is currently generated by the digestion of sewage treatment sludge while the minority of it is produced through the fermentation or gasification of solid waste or of lignocellulosic material (Chandra et al., 2012). The anaerobic digestion kinetics for methane fermentation of sewage sludge was proposed by Sosnowski et al. (2008). However, the proposed model was only designed for batch mode operation. Considering the advantages of fed-batch process in various fermentation problems, it is appropriate to convert this model into fed-batch mode. The utilization of fed-batch technique can increase the output of desirable products such as protein and biofuel in various fields of biotechnology and hence contribute to the development of renewable energy production and sustainable science.

In the past decade, several SOMs were converted to solve multi-objective problems. The conversions to MOMs were carried out by implementing some modification as well as introducing new concepts such as Pareto dominance and decomposition. However, due to increased search complexity in multi-objective optimization, premature convergence becomes a cumbersome problem (Marler & Arora 2004). Hence, the improvements in this research area remain open for developments.

1.2 Objectives

In order to investigate the effectiveness of metaheuristic in solving bioprocess problems, several goals need to be achieved:

1. Identify various single-objective and multi-objective real-world bioprocess problems.
2. Apply recent metaheuristics to solve the problems.
3. Propose modification of existing metaheuristic to improve its performance in solving multi-objective bioprocess problems.

1.3 Scope

This study applies the Backtracking Search Optimization Algorithm (BSA) (Civicioglu, 2013) to different bioprocess case studies and compares its performance with some well-known algorithms from the scientific literature. This is done by simulating the bioprocess through a set of differential equations that model the mass balances of various state variables. This study also introduces process optimization in the treatment of winery wastewater. Additionally, we also propose the modeling of fed-batch methane fermentation of sewage sludge. This model is converted from the existing batch model. The bioprocess problems considered in this study cover various aspects of human life, ranging from biofuel production of ethanol and pharmaceutical synthesis of protein and penicillin to treatment of wastewater and sewage sludge. Finally the multi-objective optimization of bioprocess application is addressed using MOMs.

- The bioprocess problems are selected from well-established bioprocess models drawn from the scientific literature which cover various aspects of human life. A problem with batch model will be converted into fed-batch model.
- Recent SOMs are applied to the optimization problems and their performances in solving single-objective bioprocess problems are compared. The SOMs are population-based algorithms.

- A pareto-based MOMs is modified and its performance is compared with other pareto and decomposition techniques. The comparisons are made through benchmark problems and real-world bioprocess problems.

CHAPTER 2: LITERATURE REVIEW

Biotechnology has been considered as one of the new knowledge-based economy and can provide advancements and growth for societies and economies while enabling better health care and sustainable transformation of raw materials and hazardous waste treatment in industries (Juma & Konde, 2001). Fermentation process is one of the fundamental elements in biotechnology. Stochastic algorithms or metaheuristics have been previously applied on various bioprocess optimization problems. Evolutionary algorithms (EA) have been utilized on the bioprocess of protein production with E. coli, and they have been compared with first order gradient algorithms and with dynamic programming by Roubos et al. (1999). According to I. Rocha (2003), health care is one of the most promising applications in biotechnology, with pharmaceutical recombinant DNA applications being the sector with the highest growth rate. Various valuable products such as antibiotics and recombinant protein have been produced using fermentation techniques. The optimization of feeding profile for ethanol and penicillin production was applied by Kookos (2004) using Simulated Annealing while the optimization of protein production in E. coli was applied using Ant Algorithms by Jayaraman et al. (2001). Chiou and Wang (1999) used Differential Evolution (DE) for the optimization of the Zymomous mobilis fed-batch fermentation while Wang and Cheng (1999) used the same algorithm for ethanol production in Saccharomyces cerevisiae. Sarkar and Modak (2004) used a genetic algorithm based technique to address fed-batch bioreactor application problems with single or multiple control variables.

A recent study shows DE is a better solution for bio-process applications (Banga et al., 2004). Da Ros et al. (2013) have even suggested DE hybrids for these applications after showing DE as the better method in the estimation of the kinetic parameters of an

alcoholic fermentation model. M. Rocha et al. (2014) compared the performance of EAs, DE and Particle Swarm Optimization (PSO) on four different bioprocess case studies taken from the scientific literature and found that DE had better performance when compared to other algorithms.

In recent years, many new nature-inspired algorithms have emerged such as Particle Swarm Optimization (PSO) (Kennedy & Eberhart, 1995), Artificial Bee Colony Optimization (ABC) (Basturk, 2006), Cuckoo Search (CS) (X. S. Yang & Suash, 2009), Firefly Algorithm (FA) (Xin She Yang, 2010) and Artificial Algae Algorithm (AAA) (Uymaz et al., 2015). A detailed discussion on the proliferation of search algorithms can be seen in Sörensen (2015) and an overview of some of the most widely used can be seen in Burke and Kendall (2014). These algorithms were applied to various problems and have shown improved performance compared to classical algorithms.

BSA was developed for solving real-valued numerical optimization problems based on the behaviour of living creatures in social groups revisiting at random intervals to preying areas enriched by food source. It has shown promising results in solving boundary-constrained benchmark problems. Due to its encouraging performance, several studies have been done to investigate BSA's capabilities in solving various engineering problems (Askarzadeh & Coelho, 2014; Das et al., 2014; El-Fergany, 2015; Guney et al., 2014; Song et al., 2015).

BSA uses a unique mechanism for generating trial individual by controlling the amplitude of the search direction through mutation parameter, F. This enables a balanced global and local search, thus enhances its problem solving ability. BSA also consults its historical population which is stored in its memory to generate more efficient trial population, resulting in improved searching ability. Other algorithms such as PSO, DE and Covariance Matrix Adaptation Evolution Strategy (CMAES) do not use

previous generation populations. BSA employs advanced crossover strategy, which has a non-uniform and complex structure that guarantees the generation of new trial population in each generation. This strategy, which enhances BSA's problem-solving capabilities, is different to those used in genetic algorithm and its variants. Also, its mutation strategy uses only one direction individual for each target individual as opposed to the strategy used in DE and its derivatives, where more than one individual can mutate in each generation. BSA also have only one control parameter in comparison to three used by DE for fine-tuning. Even though BSA is robust and less likely to be trapped in local optima, it has a weakness of poor convergence performance and accuracy.

The algorithms that we use in this thesis to be compared with BSA are CMAES, ABC, AAA and DE. We chose these algorithms in our work for various reasons. CMAES is used because it is recent swarm intelligence metaheuristic with good global convergence. It is a highly competitive, quasi parameter free global optimization algorithm for non-separable objective functions. However, it has poor performance for separable objective functions. Also, its very algorithmic features are undermined by the presence of constraints

ABC is chosen because it is a widely-used technique among swarm intelligence with promising performance on various problems. It has sufficiently strong local search ability for various types of problems. Its weakness is that it is sensitive to the control parameter used. It also has poor definition of search direction as it treats the signs of the fitness values equally.

AAA is the latest algorithm used in this work and represents the evolution of modern swarm intelligence method. It is a robust and high-performance global optimization

algorithm. However, it has three control parameters and is sensitive to the initial value of these control parameters.

Finally, DE is used as it is an established method in the field of fed-batch fermentation optimization and regarded as the best performing algorithm in the simulation of fed-batch fermentation problems. It is a very effective global search algorithm with a quite simple mathematical structure. The algorithm is also able to choose from up to ten different options for its combination of mutation and crossover schemes. However, it has three control parameters and the algorithm is sensitive to the initial value of these parameters. Also, the process of determining the optimum mutation and crossover strategies for the problem structure in the DE algorithm is time-consuming.

In the context of multi-objective optimization, several SOMs algorithms such as PSO, EA and grey wolf optimizer (GWO) (Mirjalili et al., 2014) have been converted into their multi-objective versions which are multi-objective particle swarm optimization (MOPSO) (Coello et al., 2004), multi-objective evolutionary algorithm based on decomposition (MOEA/D) (Q. Zhang & Li, 2007) and multi-objective grey wolf optimizer (MOGWO) (Mirjalili et al., 2016). There are also algorithms which are an improvements or modification of existing MOMs. One such example is the Non-dominated Sorting Genetic Algorithm (NSGA) (Goldberg, 1989; Srinivas & Deb, 1994), which was improved upon by NSGA-II (Deb et al., 2002).

Researches on the area of multi-objective bioprocess optimization problems are not new. Polymerization systems have been numerously studied (Cawthon & Knaebel, 1989; Silva & Biscaia Jr, 2003; Tsoukas et al., 1982). A detailed review on the application of multi-objective optimization in chemical engineering was presented by

Bhaskar et al. (2000). More recently, the multi-objective optimization of fed-batch bioreactors was addressed by Sarkar and Modak (2005) using NSGA-II.

2.1 Backtracking Search Algorithm (BSA)

BSA is an evolutionary algorithm based on DE (Civicioglu, 2013). It has advanced mutation and crossover operators for the generation of trial populations. It also has balanced exploration and exploitation abilities by generating parameter F. This parameter will control the range of the search direction by adjusting the size of the search amplitude (either large value for global search or low value for local search). The historical population, stored in its memory, promotes effective trial individuals generation and ensures high population diversity. BSA also has the advantage of having only one control parameter, the $mixrate$. This parameter determines the number of elements of individuals that will mutate in a trial, thus facilitating ease of application by reducing the number of parameters that require fine-tuning.

The procedures of BSA can be separated into five processes: initialization, selection-I, mutation, crossover and selection-II. A general BSA structure is presented in Figure 2.1. For details on the processes, refer to (Civicioglu, 2013). Overviews of the five processes are provided below:

```
1. Initialization
repeat
    2. Selection-I
    Generation of Trial-Population
        3. Mutation
        4. Crossover
    end
    5. Selection-II
until stopping conditions are met;
```

Figure 2.1: A general structure of BSA

2.1.1 Initialization

The procedures of BSA begin by initializing the population P as follows:

$$P_{i,j} = lower_j + \left(upper_j - lower_j\right) \times random, i = (1,2, \dots, NP), j = (1,2, \dots, DP) \quad (6)$$

where NP and DP are the size of the population and the number of dimension of the problem respectively. $random$ is a real value uniformly distributed between 0 and 1. $lower_j$ and $upper_j$ represent the lower and upper bound in the j-th element of the i-th individual respectively.

2.1.2 Selection-I

In the Selection-I procedure, the historical population $oldP$ is generated to calculate the search direction. Initially, it is calculated as follow:

$$oldP_{i,j} = lower_j + \left(upper_j - lower_j\right) \times random,\ i = (1,2, \dots, NP), j = (1,2, \dots, DP) \quad (7)$$

In each iteration, $oldP$ is defined as follow:

$$if\ a < b\ then\ oldP := P | a, b \in [0,1] \quad (8)$$

where $:=$ is the update operation. a and b are two random numbers with uniform distribution between 0 to 1. The above equation ensures that the population in BSA can be randomly selected from historical population. This historical population is memorized by the algorithm until it is changed through a random permutation.

2.1.3 Mutation

The initial trial population is generated through mutation operation as follows:

$$T = P + (oldP - P) \times F \quad (9)$$

where F is a scale factor which controls the amplitude of the search-direction matrix $(oldP - P)$. In the original paper, $F = 3 \cdot random$, where $random$ is a random real number with uniform distribution between 0 to 1. By involving the historical population in the calculation of the search-direction matrix, BSA learns from its memory of previous generations to obtain a trial population.

2.1.4 Crossover

The final trial population T is generated by crossover. The trial individuals with improved fitness values guide the search direction for the optimization problem. The crossover of the BSA works as follows. A binary integer-valued matrix (map) of size $NP \times DP$ is computed in the first step. The individuals of T are generated by using the relevant individuals of P. If $map_{i,j} = 1$, T is updated with $T_{i,j} := P_{i,j}$.

2.1.5 Selection-II

In the Selection-II phase, the T_i that outperforms the corresponding P_i in terms of fitness value is used to update the P_i. When the best solution $Pbest$ dominates the previous global optimal value found by the BSA, the global optimal solution is replaced by $Pbest$ and the global optimal value is also updated to be the fitness value of $Pbest$.

Eight fermentation models are as case studies in this work, six of which are single-objective while the other two are multi-objectives. These cases are chosen based on the different nature of the bioprocesses. The fed batch fermentation case studies considered in this study cover various aspects of human life, ranging from biofuel production of ethanol, pharmaceutical synthesis of protein and penicillin, to treatment of wastewater and sewage sludge. The idea is to compare the performance of the algorithms in different fed batch fermentation systems.

2.2 Case study I

The first case study in this paper is the fed-batch bioreactor process of ethanol by Saccharomyces cerevisiae. This problem was first proposed by Chen and Hwang (1990), with the goal of obtaining the substrate feed rate profile that maximizes the production of ethanol. The model equations are as follows:

$$\frac{dx_1}{dt} = g_1 x_1 - u\frac{x_1}{x_4} \tag{10}$$

$$\frac{dx_2}{dt} = -10 g_1 x_1 + u\frac{150 - x_2}{x_4} \tag{11}$$

$$\frac{dx_3}{dt} = g_1 x_1 - u\frac{x_3}{x_4} \tag{12}$$

$$\frac{dx_4}{dt} = u \tag{13}$$

The kinetic variables g_1 and g_2 (h^{-1}) are given by:

$$g_1 = \frac{0.408}{\left(1+\frac{x_3}{16}\right)}\frac{x_2}{(0.22+x_2)} \tag{14}$$

$$g_2 = \frac{1}{\left(1+\frac{x_3}{71.5}\right)}\frac{x_2}{(0.44+x_2)} \tag{15}$$

The performance index *(PI)* is defined as:

$$PI = x_3(t_f)x_4(t_f) \tag{16}$$

The variables for case study I are defined in Table 2.1. The variable constraints are: $0 \leq x_4(t) \leq 200$ and $0 \leq u(t) \leq 12$. The final time, t_f and the initial state conditions are given in Table 2.2.

Table 2.1: Variables definitions for case study I.

State variables	Definitions
x_1	Cell mass (g/L)
x_2	Substrate concentrations (g/L)
x_3	Ethanol concentrations (g/L)
x_4	Volume of the reactor (L)
u	Feeding rate (L/h)

Table 2.2: Parameter values for case study I.

Parameter	Value
t_f	54 hours
$x_1(0)$	1 g/L
$x_2(0)$	150 g/L
$x_3(0)$	0 g/L
$x_4(0)$	10 L

2.3 Case study II

The second case study involves induced foreign protein production by recombinant bacteria, firstly proposed by Lee and Ramirez (1994). The problem was later modified by Tholudur and Ramirez (1997). The model equations (Tholudur & Ramirez, 1997) are as follows:

$$\frac{dx_1}{dt} = u_1 - u_2 \tag{17}$$

$$\frac{dx_2}{dt} = g_1 x_2 - \frac{u_1+u_2}{x_1} x_2 \tag{18}$$

$$\frac{dx_3}{dt} = \frac{100u_1}{x_1} - \frac{u_1+u_2}{x_1} x_3 - \frac{g_1}{0.51} x_2 \tag{19}$$

$$\frac{dx_4}{dt} = R_{fp} x_2 - \frac{u_1+u_2}{x_1} x_4 \tag{20}$$

$$\frac{dx_5}{dt} = \frac{4u_2}{x_1} - \frac{u_1+u_2}{x_1} x_5 \tag{21}$$

$$\frac{dx_6}{dt} = -k_1 x_6 \tag{22}$$

$$\frac{dx_7}{dt} = k_2(1 - x_7) \tag{23}$$

The process kinetics are given by:

$$g_1 = \left(\frac{x_3}{14.35 + x_3\left(1 + \frac{x_3}{111.5}\right)}\right)\left(x_6 + \frac{0.22x_7}{0.22 + x_5}\right) \tag{24}$$

$$R_{fp} = \left(\frac{0.233x_3}{14.35 + x_3\left(1 + \frac{x_3}{111.5}\right)}\right)\left(\frac{0.005 + x_5}{0.022 + x_5}\right) \tag{25}$$

$$k_1 = k_2 = \frac{0.09x_5}{0.034 + x_5} \tag{26}$$

The PI is defined as:

$$PI = x_4(t_f)x_1(t_f) - Q\int_0^{t_f} u_2(t)dt \tag{27}$$

The variables for case study II are defined in Table 2.3. The variable constraints are: $0 \leq u_{1,2}(t) \leq 1$. The ratio of the cost of the inducer to the value of the protein product, Q, the final time, t_f and the initial state conditions are given in Table 2.4.

Table 2.3: Variables definitions for case study II.

State variables	Definitions
x_1	Reactor volume (L)
x_2	Cell concentrations (g/L)
x_3	Substrate concentrations (g/L)
x_4	Foreign protein concentrations (g/L)
x_5	Inducer concentrations (g/L)
x_6	Inducer shock factors on the cell growth rate
x_7	Recovery factors on the cell growth rate
u_1	Glucose feed rates (L/h)
u_2	Inducer feed rates (L/h)

Table 2.4: Parameter values for case study II.

Parameter	Value
Q	5
t_f	10 hours
$x_1(0)$	1 L
$x_2(0)$	0.1 g/L
$x_3(0)$	40 g/L
$x_4(0)$	0 g/L
$x_5(0)$	0 g/L
$x_6(0)$	1 g/L
$x_7(0)$	0 g/L

2.4 Case study III

The third case study is the fed-batch fermentation of penicillin which was presented by Banga et al. (2005).The model equations are as follows:

$$\frac{dx_1}{dt} = g_1 x_1 - u\left(\frac{x_1}{500x_4}\right) \tag{28}$$

$$\frac{dx_2}{dt} = g_1 x_1 - 0.01x_2 - u\left(\frac{x_2}{500x_4}\right) \tag{29}$$

$$\frac{dx_3}{dt} = -\left(\frac{g_1 x_1}{0.47}\right) - \left(\frac{g_2 x_2}{1.2}\right) - x_1\left(\frac{0.029x_3}{0.0001+x_3}\right) + \frac{u}{x_4}\left(1 - \frac{x_3}{500}\right) \tag{30}$$

$$\frac{dx_4}{dt} = \frac{u}{500} \tag{31}$$

The process kinetics are given by:

$$g_1 = 0.11\left(\frac{x_3}{0.006x_1+x_3}\right) \tag{32}$$

$$g_2 = 0.0055\left(\frac{x_3}{0.0001+x_3(1+10x_3)}\right) \tag{33}$$

The variable constraints are: $0 \leq x_1(t) \leq 40,\ 0 \leq x_3(t) \leq 25, 0 \leq x_4(t) \leq 10$ and $0 \leq u(t) \leq 50$. The PI is defined as:

$$PI = x_2(t_f)x_4(t_f) \tag{34}$$

The variables for case study III are defined in Table 2.5. The final time, t_f and the initial state conditions are given in Table 2.6.

Table 2.5: Variables definitions for case study III.

State variables	Definitions
x_1	Biomass concentrations (g/L)
x_2	penicillin concentrations (g/L)
x_3	substrate concentrations (g/L)
x_4	Volume of the reactor (L)
u	Feeding rate (L/h)

Table 2.6: Parameter values for case study III.

Parameter	Value
t_f	132 h
$x_1(0)$	1.5 g/L
$x_2(0)$	0 g/L
$x_3(0)$	0 g/L
$x_4(0)$	7 L

The above case studies are well-established bioprocess models drawn from the scientific literature. We use these models to verify the robustness of recent metaheuristics. Montalvo et al. (2010) used fed-batch operation in biological wastewater treatment though wastewater treatment rarely employs fed-batch operation. Thus, in the following sections, we propose the applications of fed-batch process optimization using the same metaheuristics on the field of biology wastewater treatment for the purpose of detoxification and methane production and investigate its effectiveness.

2.5 Case study IV & V: Pilot-scale fed-batch aerated lagoons treating winery wastewaters

Montalvo et al. (2010) proposed the treatment of winery wastewaters using two stage pilot-scale fed-batch aerated lagoons. The overall performance of this process can be evaluated by measuring the COD removal efficiency which is defined as the quotient between the difference of the initial COD and effluent COD concentrations and the initial COD concentration (Pelillo et al., 2006). The model equations (Montalvo et al., 2010) are as follows:

$$\frac{dV}{dt} = F \tag{35}$$

$$\frac{dS}{dt} = \left(\frac{F}{V}\right)(S_0 - S) - \left[\frac{\mu_m(S-S_{nb})}{K_S+(S-S_{nb})} - K_d\right]\left(\frac{X}{Y}\right) \tag{36}$$

$$\frac{dX}{dt} = \left[\left[\frac{\mu_m(S-S_{nb})}{K_S+(S-S_{nb})} - K_d\right] - \left(\frac{F}{V}\right)\right] X \tag{37}$$

The variables for case study IV and V are defined in Table 2.7. The values for the kinetic parameters are given in Table 2.8.

Table 2.7: Variables definitions for case study IV and V.

State variables	Definitions
V	Lagoon volume (L or m^3)
F	Volumetric flow-rate (L or m^3/day),
t	Operation time (days)
μ_m	Maximum specific microbial growth rate (1/days)
S_0	Influent substrate concentrations (mg or g COD/L)
S	Effluent substrate concentrations (mg or g COD/L)
S_{nb}	Non-biodegradable substrate concentration (mg or g COD/ L)
X	Cellular or biomass concentration (mg)
Y	Cellular yield coefficient (g VSS/g COD)
K_S	Saturation constant (mg or g COD/L)

Table 2.8: Kinetic parameters for case study IV and V.

Parameter	Value
μ_m	0.28 1/days
Y	0.26 g VSS/g COD
K_S	175 mg COD/L
K_d	0.12 1/days
S_{nb}	790 mg COD/L

The volume constraint is given as: $V \leq V_m$ where V_m is the maximum operational lagoon volume. The values for V_m and the final time, t_f along with the initial conditions for the two stages of operation is given in Table 2.9.

Table 2.9: Parameter values for case study IV and V.

Parameter	First stage	Second stage
V_m	27.20 m^3	10.80 m^3
t_f	30 days	24 days
$V(0)$	3.470 m^3	5.10 m^3
$S_0(0)$	8700 mg/L	1980.33 mg/L
$X(0)$	900 mg VSS/L	21373 mg VSS/L

The bounds on the decision variables are $F \in [0; 2]$ for the first stage and $F \in [0; 1]$ for the second stage. The PI is defined as:

$$PI = (S_0 - S)/S_0 \times 100 - (V_m - V) \times 100 \quad (38)$$

In this study, we consider the first stage and the second stage of this model as case study IV and case study V respectively.

2.6 Case study VI: Methane production from sewage sludge fermentation

The model for batch methane fermentation of Sewage Sludge (SS) was proposed by Sosnowski et al. (2008), where the carbon balance process was determined and the simple kinetic model of anaerobic digestion was developed. The batch experiment with the above mentioned feedstock was conducted in a large scale laboratory reactor of

working volume of 40.0 dm^{-3}. In our study, we convert this batch model into a fed-batch model which will be discussed in Section 3.1.1

2.7 Case study VII

In this case study, we will address the lysine fermentation model proposed by Ohno et al. (1976). The model equations are as follow:

$$\frac{dx_1}{dt} = \mu x_1 \tag{39}$$

$$\frac{dx_2}{dt} = FS_F - \sigma x_1 \tag{40}$$

$$\frac{dx_3}{dt} = \pi x_1 \tag{41}$$

$$\frac{dx_4}{dt} = F \tag{42}$$

where

$$\mu = 0.125x_2 \tag{43}$$

$$\sigma = \frac{\mu}{0.135} \tag{44}$$

$$\pi = -384\mu^2 + 134\mu \tag{45}$$

The variables for case study I are defined in Table 2.10. The variable constraints are: $x_4(t) \leq 20$ and $0 \leq F(t) \leq 2$. The initial state conditions and the value of S_F are given in Table 2.11.

Table 2.10: Variables definitions for case study VII.

State variables	Definitions
x_1	Cell mass (g/L)
x_2	Substrate concentrations (g/L)
x_3	Product (Lysine) concentrations (g/L)
x_4	Fermenter volume (L)
F	Substrate volumetric feeding rate (L/h)
S_F	Substrate feed concentration (g/L)
μ	Specific growth rates
σ	Substrate consumption
π	Product formation

Table 2.11: Parameter values for case study VII.

Parameter	Value
$x_1(0)$	0.1 g/L
$x_2(0)$	14 g/L
$x_3(0)$	0 g/L
$x_4(0)$	5 L
S_F	2.8 wt%

There are two performance index (PI) which are needed to be maximized. The first PI is the productivity (J_p)while the second PI is the yield (J_y). These are defined as follows:

$$J_p = \frac{x_3(t_f)}{t_f} \tag{46}$$

$$J_y = \frac{x_3(t_f)}{\int_0^{t_f} F(t)S_F dt} \tag{47}$$

where the final time, t_f is an additional variable to be found by the algorithm within the range of 30-40 h. The number of intervals for the feeding sequence is set as 20 intervals.

2.8 Case study VIII

This case study is the same as case study II except that in this case, there are two performance index (PI). In case study II, the PI is to maximize the amount of protein product while minimizing the amount of inducer by using the later term as a penalty. In case study VIII however, the two terms are separated into two different objectives as follows:

$$\underset{u_1(t),u_2(t)}{Maximize}\ J_1 = x_4(t_f)x_1(t_f). \tag{48}$$

$$\underset{u_1(t),u_2(t)}{Minimize}\ J_2 = \int_0^{t_f} u_2(t)dt. \tag{49}$$

CHAPTER 3: METHODOLOGY

The methodology of this study can be divided into two parts. The first part addresses the single-objective problems while the second part involves the multi-objective problems.

3.1 Single-objective optimization problems

Six case studies (case studies I-VI) which were discussed in Section 2 are used in our experiments.

3.1.1 Conversion of case study VI from batch mode into fed-batch mode.

The batch operation of methane fermentation can be converted into fed-batch by using the continuity equation:

$$m_{in} - m_{out} - m_{consumed} = \frac{dm}{dt} \tag{50}$$

Replace the formula with the rate of change of substrate:

$$S_{in} - S_{out} - S_{consumed} = \frac{dS}{dt} \tag{51}$$

In fed-batch, no substrate is taken out and the substrate is consumed at a constant rate:

$$S_{in} - kS = \frac{dS}{dt} \tag{52}$$

Where the substrate input is defined as follow:

$$S_{in} = \frac{u \cdot (S_0 - S)}{L} \tag{53}$$

where u is the feed flow rate, S_0 is the substrate concentration in the feed, S is the substrate concentration in the fermentor and L is the volume of the fermentor. When

converting a batch model into fed-batch, a diluting term is added into each element. The diluting term is added only to the elements which are either in solid or liquid state. Hence, the elements which are in gaseous state remain unchanged (del Rio-Chanona et al., 2016).

In this study, the methane fermentation of sewage sludge in fed-batch mode was investigated and is considered as case study VI. The fed-batch operation of sewage sludge fermentation, which was converted from the batch model by Sosnowski et al. (2008), was modeled as follows:

$$\frac{dS}{dt} = \frac{u}{L} * (S_0 - S) - k \cdot S \tag{54}$$

$$\frac{dV}{dt} = Y_{V/S} \cdot k \cdot S - v_V \cdot \frac{V}{K_S + V} \cdot X_0 - V * \frac{u}{L} \tag{55}$$

$$\frac{dCH_4}{dt} = Y_{CH_4/V} \cdot v_V \cdot \frac{V}{K_S + V} \cdot X_0 \tag{56}$$

$$\frac{dCO_2}{dt} = Y_{CO_2/S} \cdot k \cdot S + Y_{CO_2/V} \cdot v_V \cdot \frac{V}{K_S + V} \cdot X_0 \tag{57}$$

$$\frac{dL}{dt} = u \tag{58}$$

The variables for case study VI are defined in Table 3.1. The constant parameter values, the final time, t_f and the initial state conditions are given in Table 3.2.

Table 3.1: Variables definitions for case study VI.

State variables	Definitions
k	Constant of first-order reaction (d^{-1})
S	Carbon content in TSS ($g\ C\ dm^{-3}$)
V	Carbon content in VFA ($g\ C\ dm^{-3}$)
K_S	Saturation constant ($g\ C\ dm^{-3}$)
X_0	Biomass concentration ($g\ C\ dm^{-3}$)
v_V	Maximum specific utilization of VFA rate (d^{-1})
$Y_{V/S}$	Yield factor of VFA from substrate
$Y_{CH_4/V}$	Yield factor of CH_4 from VFA
$Y_{CO_2/S}$	Yield factor of CO_2 from S
$Y_{CO_2/V}$	Yield factor of CO_2 from VFA

Table 3.2: Parameter values for case study VI.

Parameter	Value
X_0	5 $g\ C\ dm^{-3}$
S_0	20 $g\ C\ dm^{-3}$
k	0.11 d^{-1}
$Y_{V/S}$	0.72 d^{-1}
K_S	11.24 $g\ C\ dm^{-3}$
v_V	2.08 d^{-1}
$Y_{CH_4/V}$	0.71 d^{-1}
$Y_{CO_2/S}$	0.17 d^{-1}
$Y_{CO_2/V}$	0.22 d^{-1}
t_f	23 d
$S(0)$	4.75 $g\ C\ dm^{-3}$
$V(0)$	0 $g\ C\ dm^{-3}$
$CH_4(0)$	0 $g\ C\ dm^{-3}$
$CO_2(0)$	0 $g\ C\ dm^{-3}$
$L(0)$	2.4 dm^3

The variable constraints are: $u \in [0;\ 1], S(t) \leq 5, L(t) \leq 40$. The total mass of carbon in the fermentor is constrained as follow:

$$[S(t) + V(t) + CH_4(t) + CO_2(t)] \cdot L(t) \leq 12 \tag{59}$$

The performance index (PI) is given by:

$$PI = CH_4(t_f) \tag{60}$$

3.1.2 Validation of batch results and improvement using fed batch for case study VI

To show the improvements of fed-batch operation over batch in the methane production from sewage sludge fermentation, we ran a preliminary test for this model. Figure 3.1 shows the comparison of batch and fed-batch for sludge fermentation where FB stands for fed-batch while B stands for batch. The result for fed-batch was obtained from our preliminary simulation using the methodology described above and BSA as the optimization algorithm. We found that fed-batch produced 8.95% more methane compared to the conventional batch process. This improvement comes from the controlled feeding for each day during the fermentation process. The amount of methane produced by fed-batch starts to increase over batch after the ninth day. It is worth noting that fed-batch was able to produce more methane even when the initial substrate is less than the amount used in batch (4.75 g dm^{-3} for fed-batch compared to 5 g dm^{-3} for batch). Figure 3.2 shows the best feeding rate obtained by BSA for case study VI.

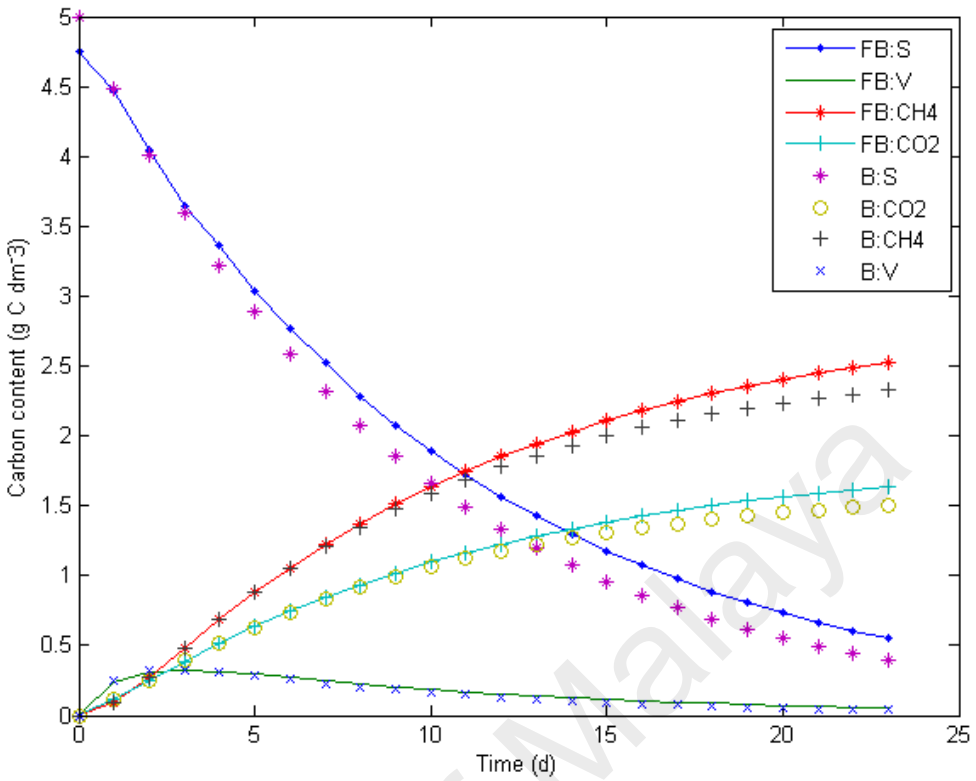

Figure 3.1: Comparison of batch and fed-batch for sludge fermentation

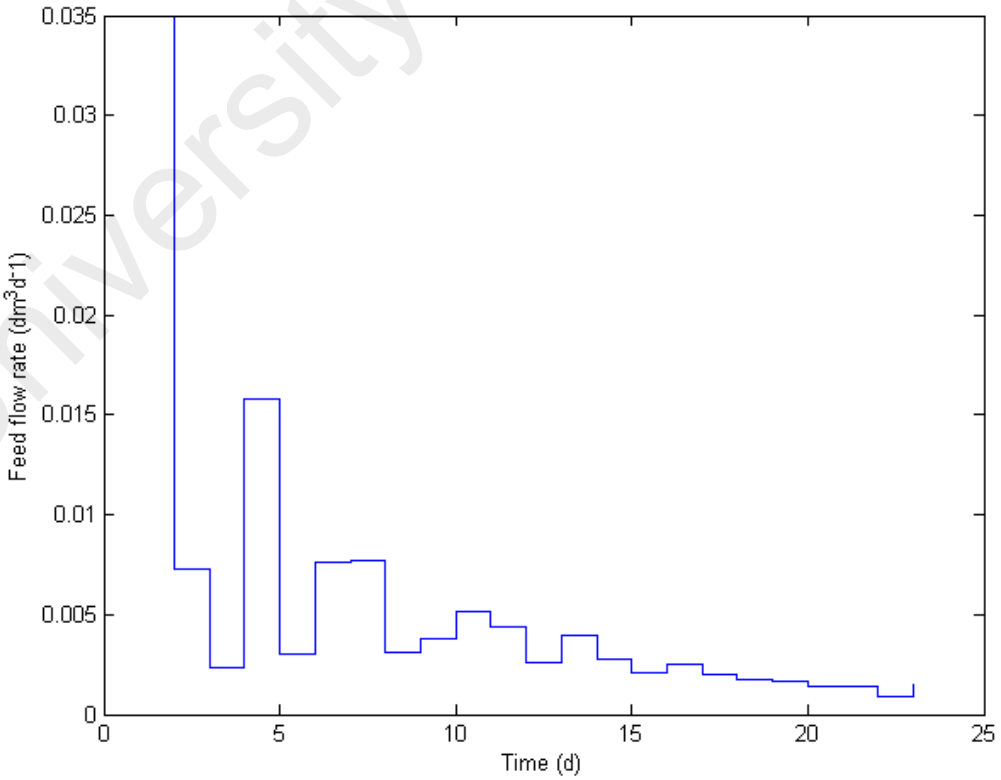

Figure 3.2: Control profile for the fed-batch sludge fermentation

3.1.3 Experimental setup

In this experiment, BSA is compared with four different metaheuristics: Covariance Matrix Adaptation Evolution Strategy (CMA-ES) (Hansen & Ostermeier, 1996), Differential Evolution (DE) (Storn & Price, 1997), Artificial Bee Colony (ABC) (Basturk, 2006) and Artificial Algae Algorithm (AAA) (Uymaz et al., 2015). All the algorithms are population-based algorithm. In the context of fed-batch fermentation processes optimization, the solutions found by the algorithms represent the trajectory of input variables. The solutions or input variables are represented by $M \times (N + 1)$ real valued vectors. M is the predetermined number of input variables. N is the predetermined size of input variables or the number of feeding intervals. Each vector encodes an input variable as a temporal sequence of values, defined as a piecewise linear function, with N equally spaced, linearly interpolated segments. For the cases where there are more than one input variables, all the M vectors are joined sequentially to create a solution. In this experiment, all the case studies have only one input variable except for case study II which has two input variables.

Each solution is evaluated by running a numerical simulation of the differential equation model defined in each case. This simulation is achieved using the Runge-Kutta method provided by Matlab ODE suite. After the simulation, the fitness value of the solution is calculated according to the PI of each case. Also, the relative and absolute error tolerances for integrations of the system dynamics were set to 10^{-8} in order to provide accurate and consistent results. The constraints for each case are handled by implementing constant penalty method. Figure 3.3 shows the flowchart of BSA implementation in the experiments.

The means of 30 runs along with its 95% confidence intervals are presented as results in this paper. T-test (Goulden, 1956) for two-sample comparisons is

implemented in this work. We also employed the Holm correction for the p-values (Holm, 1979) for the multiple pairwise comparisons. For ease of presentation, we used a symbolic encoding for the p-values obtained from t-tests results. Different symbols are employed that give straightforward comparison between the algorithms and report whether the mean of algorithm $A1$ is greater than the mean of $A2$ or vice versa, as shown in Table 3.3. In the experiments, some algorithms may show insignificant difference between each other based on their statistical evaluation. However, our goal is to determine the algorithm that can provide consistent good results by having high average and narrow confidence interval for all cases.

In our experiments, we use the standard parameters for each algorithm that were suggested by previous studies. The termination condition is set after 200,000 FEs (function evaluations) and the population size for all algorithms is 20. For DE in particular, the parameters are as follow: F = 0.5 and CR = 0.6. The value of N is equal to the value of t_f in all single-objective cases except for case studies II and III (25 and 10 respectively).

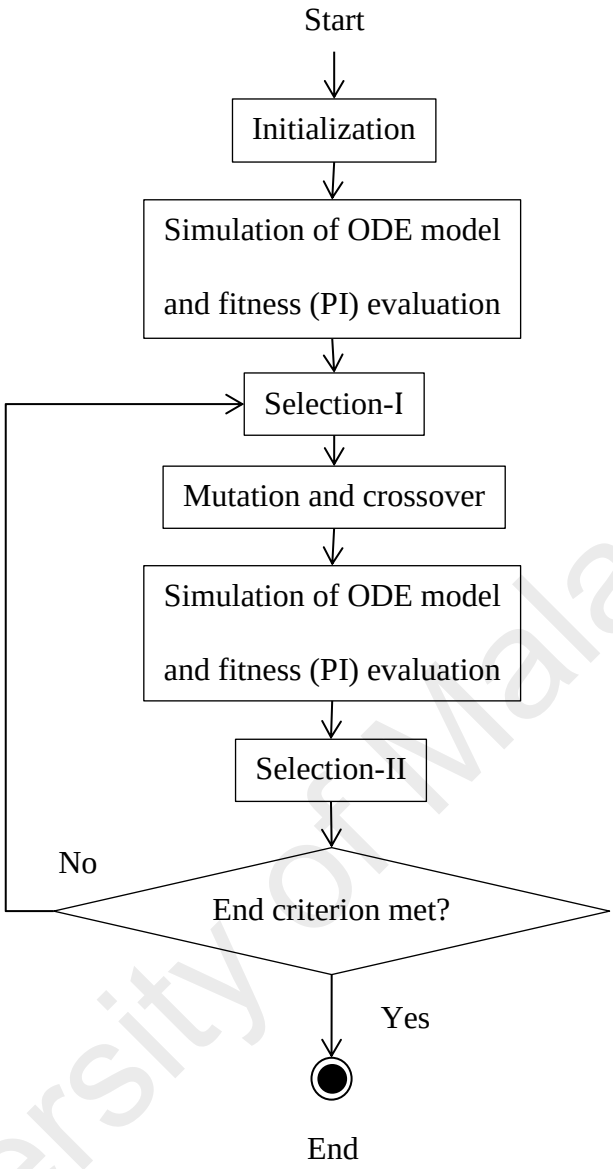

Figure 3.3: BSA flowchart.

Table 3.3: Symbolic encoding for comparing t-tests results.

p-Value	Condition	Symbol
$p \leqslant 0.001$	mean($A1$) > mean($A2$)	+++
$p \leqslant 0.001$	mean($A1$) < mean($A2$)	- - -
$0.001 < p \leqslant 0.01$	mean($A1$) > mean($A2$)	++
$0.001 < p \leqslant 0.01$	mean($A1$) < mean($A2$)	- -
$0.01 < p \leqslant 0.05$	mean($A1$) > mean($A2$)	+
$0.01 < p \leqslant 0.05$	mean($A1$) < mean($A2$)	-
$p \geqslant 0.05$		O

3.2 Multi-objective optimization problems

Two case studies (case studies VII and VIII), which was discussed in Section 2 will be used for our study in multi-objective bioprocess problems.

3.2.1 Modified Multi Objective Particle Swarm Optimization (M-MOPSO)

In the second part this study, we propose a modification of an existing MOM called multi-objective particle swarm optimization (MOPSO) (Coello et al., 2004). This modification, called modified MOPSO (M-MOPSO) retains some elements used in the original MOPSO but at the same time introduces new processes to either replace or combine with the original procedures.

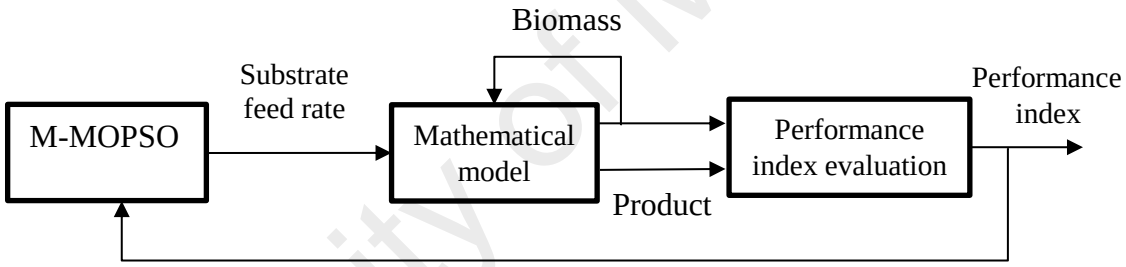

Figure 3.4: Fed-batch fermentation using M-MOPSO.

Figure 3.4 shows the overall flowchart of the fed-batch fermentation optimization system using M-MOPSO. M-MOPSO generates solutions which represent the substrate feed rate for the bioprocess. The unit of substrate feed rate is defined as the volume per unit time ($^{V}/_{t}$). This variable provides the feeding profile for the bioreactor to provide a certain amount of input at a certain time during the fermentation process. The bioprocess is simulated using mathematical model which is usually a set of ordinary differential equations (ODE). The ODE describe the relationship between operating parameters that includes inputs, intermediatory and outputs. The biomass is

continuously used by the substrate to produce yield. The output information from the bioprocess, such as the volume of the product and biomass are used to calculate the performance index (PI) of the solutions. The PI values are given back to M-MOPSO to find better solutions and the cycle repeats until the end criterion is met.

M-MOPSO shares many similarities with MOPSO. The biggest similarity is the utilization of external archive/repository (REP). In the original MOPSO, the repository is made up of two main elements: the archive controller and the grid. The archive controller governs the selection and removal of the repository members. The grid system used in MOPSO is in the form of adaptive hypercubes where the objective space is divided into several regions to store the solutions. This system is used to reduce the computational cost when the archive controller needs to add or remove the repository member. Though the same principle is used in M-MOPSO, the execution is different. While the M-MOPSO uses the same grid system, the procedure for its archive controller is modified in several ways. These modifications, along with the introduction of other new procedures are described in the following subsections.

3.2.1.1 Population initialization

M-MOPSO initialize by randomly generating n number of population POP within the problem's upper and lower boundary. The value of n is predetermined by the user.

$$POP = \begin{bmatrix} x_1^1 & \cdots & x_d^1 \\ \vdots & \ddots & \vdots \\ x_1^n & \cdots & x_d^n \end{bmatrix} \text{ or}$$

$$POP^j = \left[x_1^j, x_2^j, \dots, x_d^j\right] \tag{61}$$

where x_i^j is the variable in ith dimension of jth population.

3.2.1.2 Dynamic boundary control mechanism

A new boundary control mechanism is introduced in this paper. This mechanism is the main evolutionary process of the population. In each iteration t, each individual in the population will produce a new population according to the current boundary of each individual. The boundary of each individual changes dynamically by taking into account the current position of the individual and the boundary factor, b_f which is defined in section 3.2.1.3. This new procedure is implemented to overcome the weakness of the swarm intelligence used in MOPSO in its exploitation aspect. With this new technique, balanced exploration and exploitation can be achieved by intelligently expand or shrink the search boundary of each individual based on some set of conditions. This boundary is determined by calculating the initial value of Q as follow:

$$Q_i = \frac{R_i^{lower} - R_i^{upper}}{2}, \quad i = 1, 2, \dots, d \tag{62}$$

where R_i^{upper} and R_i^{lower} is the problem's upper and lower boundary respectively. The value of Q will shrink in each iteration as follows:

$$Q_i^{t+1} = Q_i^t \times s_{f2}, \quad i = 1, 2, \dots, d \tag{63}$$

where s_{f2} is the predetermined parameter called shrink factor. The upper and lower individual boundary at iteration t is calculated as follow:

$$UB_i^j = x_i^j + Q_j, \quad i = 1, 2, \dots, d, \quad j = 1, 2, \dots, n \tag{64}$$

$$LB_i^j = x_i^j - Q_j, \quad i = 1, 2, \dots, d, \quad j = 1, 2, \dots, n \tag{65}$$

where x_i^j is the position in ith dimension of jth population. The values that exceed the specified problem boundary will be replaced with their respective boundary value. Each population will produce a new population, x'^j_i as follows:

$$x'^{j}_{i} = LB_i^j + rand \times \left(UB_i^j - LB_i^j\right), \quad i = 1, 2, \dots, d, \quad j = 1, 2, \dots, n \tag{66}$$

where $rand \sim \cup([0, 1])$. Each new population will be evaluated and their fitness is equal to their objective function value. For each population, some of the variables (the position in each dimension) have the probability to become the value of their respective boundaries as follows:

$$\text{IF } x'^{j}_{i} < 0.1 \times rand \times \left|R_i^{upper} - R_i^{lower}\right| + R_i^{lower}$$

$$x'^{j}_{i} = R_i^{lower}$$

$$\text{ELSE IF } x'^{j}_{i} > R_i^{upper} - 0.1 \times rand \times \left|R_i^{upper} - R_i^{lower}\right|$$

$$x'^{j}_{i} = R_i^{upper}$$

$$\text{END IF} \tag{67}$$

where $rand \sim \cup([0, 1])$. This is to ensure that the variables that are close to the value of their boundaries have the probability to go to their boundaries, hence improving exploitation. Occasionally, the value of Q may change to simulate abrupt boundary expansion or shrinking as follows:

$$\text{IF } Q_i < b_f \times \left|R_i^{upper} - R_i^{lower}\right| + R_i^{lower}$$

$$Q_i = OQ_i \times b_f$$

$$END\ IF \tag{68}$$

where OQ_i is the initial Q_i value obtained in equation (62). The determination of the boundary factor, b_f is explained in the next section.

3.2.1.3 Boundary factor determination

M-MOPSO employs a new dynamic boundary mechanism to search for new solutions. Each individual will evolve based on their respective boundaries. These boundaries may shrink or expand in each iteration depending on the boundary factor, b_f. Smaller b_f ensures greater exploitation while larger b_f encourages greater exploration. b_f is determined as follows:

1. Initially, b_f is calculated as follows:

$$b_f = rand \tag{69}$$

where $rand \sim \cup([0,1])$.

2. If the number of functional evaluations is more than half the maximum allowable, b_f is calculated as follow:

$$b_f = 0.1 \times rand \tag{70}$$

where $rand \sim \cup([0,1])$.

3. If the number of REP member is equal to the maximum allowable, $nRep$ for $nRep/_2$ iterations, b_f is determined using equation (69).
4. After S_{f1} iterations from previous step, b_f is determined using equation (70).
5. Repeat step 3.

S_{f1} is a predetermined parameter called saturation factor.

3.2.1.4 Repository member admittance method

In original MOPSO, the archive controller needs to determine the domination of each of repository member every time new members are admitted. This can lead to high

computational cost especially when the size of the repository grows larger in each iteration. Besides, identical solutions or populations (POP) may be admitted into the finite-sized repository, causing it to rapidly fill early in the iteration. In M-MOPSO, several modifications are made to lower the computational cost and ensure the uniqueness of the solutions in the repository. The pseudocode for repository member admittance is as follows:

1. Determine each POP domination.
2. Assign all REP member as nondominated.
3. FOR each nondominated POP

    FOR each REP member

        BREAK if current POP is identical to current REP

        Assign current REP as dominated if it is dominated by current POP

        ELSE assign current POP as dominated if it is dominated by current REP and BREAK

    END FOR

    END FOR

4. Insert nondominated REP and nondominated POP into REP.

3.2.1.5 Repository member deletion method

Each time new nondominated solutions are found, the archive controller will add them into the repository. However, if the size of the repository exceeds the maximum allowable, some members of the repository will be removed by the archive controller. The original MOPSO determines which member to be removed based on the density of

the grids. Members in the grid with higher density have higher chance to be removed. In M-MOPSO, the factor that determines the removal of a member depends on the Euclidean distance in the objective space between each repository member and the latest admitted member. The repository member deletion method in M-MOPSO is as follows:

1. Use roulette-wheel selection to select one member. The weight is defined as the Euclidean distance (in objective space) between each REP members and the latest admitted REP member (nearer member has higher weight).
2. Delete the selected member.
3. Repeat step 1 until the number of REP members does not exceeds the maximum allowable number.

3.2.1.6 Mutation operator and population update method

In M-MOPSO, the old population is replaced through the means of mutation. The same mutation operator used by MOPSO is used in M-MOPSO. Mutation happens with some probability (not every iteration). If the mutation process cannot improve the individual, its saturation counter is incremented by one. Once the saturation counter reached a predetermined value, that particular individual in the population may be replaced by one of the following method (randomly use one method with equal probability):

- Choose one REP member by roulette-wheel method. Members from less crowded area in the grids have higher probability to be selected.
- Randomly select one REP member.

3.2.1.7 M-MOPSO procedures

The whole procedures for M-MOPSO are as follows:

1. Randomly generate initial population, X^j for $j = 1, 2, \dots, n$ within the boundary.

   $x_i^j = R_i^{lower} + rand \times Q_i,$

   where $rand \sim \cup([0,1]), Q_i = \left|\frac{R_i^{upper} - R_i^{lower}}{2}\right|, i = 1, 2, \dots, d,$

   where d is the number of variables, n is the predetermined number of population, R_i^{upper} and R_i^{lower} are the upper and lower boundary respectively.

2. Evaluate $f(X^j)$, store nondominated solutions in archive/repository, REP, generate hypercubes, initialize saturation count, S^j to zero, store initial X^j positions as best found positions so far, BFP^j.

3. Generate new boundary.

   $$UB_i^j = x_i^j + Q_i$$

   $$LB_i^j = x_i^j - Q_i$$

4. Each particle in the population generates offspring, X'^j within new boundary.

   $x'^j_i = LB_i^j + rand \times \left|UB_i^j - LB_i^j\right|$

   where $rand \sim \cup([0,1])$

5. For each particle, some of the variables have the probability to become the value of their respective boundaries as follows:

   IF $x'^j_i < 0.1 \times rand \times \left|R_i^{upper} - R_i^{lower}\right| + R_i^{lower}$

   $$x'^j_i = R_i^{lower}$$

   ELSE IF $x'^j_i > R_i^{upper} - 0.1 \times rand \times \left|R_i^{upper} - R_i^{lower}\right|$

   $$x'^j_i = R_i^{upper}$$

   END IF

where $rand \sim \cup ([0, 1])$

6. Evaluate $f(X'^{j})$ and store nondominated X'^{j} in REP, update REP by removing dominated solutions, update hypercubes.
7. X^j performs mutation to generate X''^{j}. Evaluate $f(X''^{j})$ and replace X^j if X''^{j} is better.
8. Compare X^j with BFP^j and update BFP^j. Increment S^j if better BFP^j is not found, otherwise reset S^j to zero.
9. Replace all X^j with respective BFP^j.
10. Store nondominated X^jin REP, update REP by removing dominated solutions, update hypercubes.
11. If $S^j = S_{f1}$, replace X^j by population update method.
12. Shrink current boundary.

$$Q_i = Q_i \times S_{f2}$$

13. Determine boundary factor, b_f.
14. If $Q_i < b_f \times \left|R_i^{upper} - R_i^{lower}\right| + R_i^{lower}$, update the boundary.

$$Q_i = OQ_i \times b_f$$

where OQ_i is the original Q_i value obtained in Step 1.

15. If maximum iteration is achieved, terminate. Otherwise repeat step 3.

The flowchart for M-MOPSO is given in Figure 3.5.

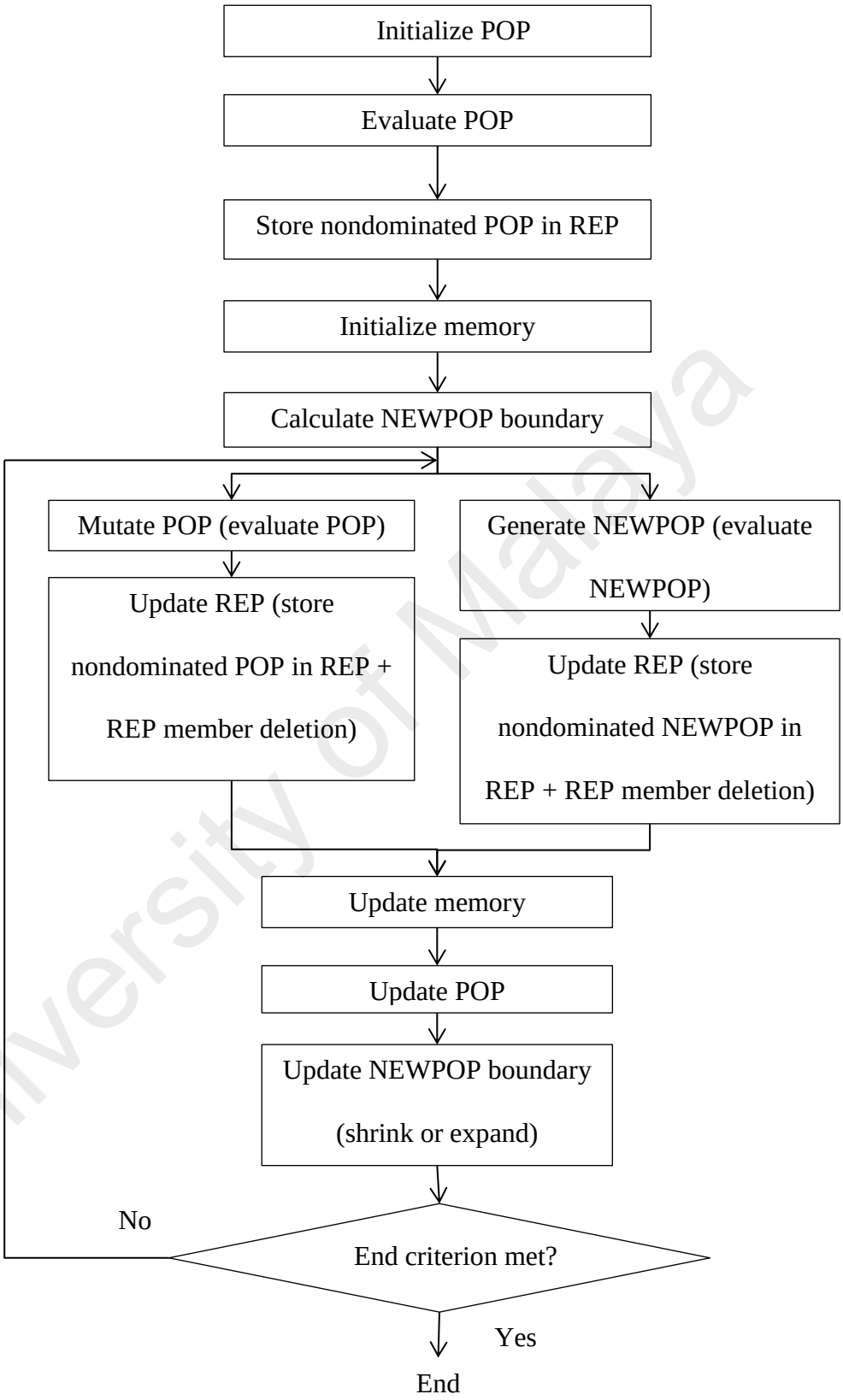

Figure 3.5: M-MOPSO's flowchart.

3.2.1.8 Similarities and differences between MOPSO and M-MOPSO

M-MOPSO have several similarities with the original MOPSO. Both are population-based evolutionary algorithm, which evolve the solution vector/population (POP) in each iteration. Like MOPSO, M-MOPSO also uses external population/repository (REP) to store the non-dominated POP and construct the Pareto front. M-MOPSO also employs identical adaptive grid mechanism used in MOPSO. Also, both utilize elitism to update individual's memory. Finally, M-MOPSO uses the same mutation operator utilized by MOPSO, though the subjects of mutation are different.

There are six major differences between M-MOPSO and MOPSO. The first difference is the population evolution procedure. MOPSO utilizes swarm intelligence technique where the whole population coordinates their movements by following a single leader. This is achieved by considering the current leader position, the memory of each individual (individual best position) and their velocity. M-MOPSO instead employs dynamic boundary control mechanism. This is achieved by deliberate shrinking and expansion of each individual boundary. New populations are randomly generated within these adaptive boundaries. M-MOPSO does not restrict its search direction by following a single leader but instead each individual evolves independently according to their own boundary.

Secondly, the mutation procedure in MOPSO is applied after the new position of the population has been found. The new population will replace the old one. In M-MOPSO, the mutation is applied directly on the old population (before evolution), not the new ones. The new population never replaces the old one.

The repository update procedure in both algorithms is also different. MOPSO updates the contents of its repository only after the new population has both been

generated and mutated while M-MOPSO updates the contents of its repository after the old population was mutated and also after the new population has been generated.

The procedure for repository member selection is also different between the two algorithms. MOPSO determines the domination of each of repository member every time new members are admitted. New members are admitted every time new nondominated solutions are found. M-MOPSO however only determines the domination of some necessary members and any redundant solutions are ignored and not admitted into the repository.

Both algorithms also use different repository member deletion procedure. MOPSO deletes its repository members based on the density of the grids, where members in the least populated grid have higher probability to be deleted. M-MOPSO deletes its repository members based on the Euclidean distance in the objective space between each repository member and the latest admitted member. The members nearer to the latest admitted member have higher probability to be deleted.

The final difference is in the aspects of leader selection and population update. In MOPSO, a single new leader is selected in each iteration and replaces the previous leader. The leader is selected from a repository member in the least populated grid. All other population remains the same as in previous iteration. In M-MOPSO, each individual population has their own saturation counter. If they cannot improve/found better solution after a predetermined number of iteration, they will be replaced either by randomly selecting a repository member or by selecting a repository member in the least populated grid.

CHAPTER 4: RESULTS AND DISCUSSIONS

4.1 Single-objective optimization problems

The results of our experiments for each case study will be shown in a pair of tables. The first table of each pair provides the mean and the 95% confidence intervals for the PI of each algorithm. We probe the PI at four different time-steps: when 25,000, 50,000, 100,000 and 200,000 FEs are performed by each algorithm. This decision is made to estimate the possibilities for terminating the optimization process earlier, immediately after good enough solutions are obtained. The second table of each pair provides the pairwise t-test results at 200,000 FEs. These results are intended to signify the statistical differences among the algorithms, where the algorithm on each row of the tables represents $A1$ on Table 3.3 while the algorithm on each column represents $A2$. The results for case studies I– III are provided in Tables 4.1 - 4.6. The results for case studies IV– V are provided in Tables 4.7 - 4.10 while the results for case study VI are provided in Tables 4.11 and 4.12.

Table 4.1: Mean and confidence intervals for case study I.

Algorithm	PI 25,000 FEs	PI 50,000 FEs	PI 100,000 FEs	PI 200,000 FEs
BSA	20285 ± 30.73	20341 ± 26.56	20392 ± 14.26	**20418 ± 4.71**
AAA	20348 ± 10.42	20357 ± 14.87	20369 ± 9.91	20382 ± 7.02
ABC	7875 ± 2576	11258 ± 4605	20299 ± 61.62	20317 ± 36.98
DE	**20384 ± 4.82**	20381 ± 24.62	20388 ± 18.93	20406 ± 2.27
CMAES	20211 ± 100.2	**20373 ± 46.09**	**20403 ± 29.87**	20412 ± 30.03

Table 4.2: T-test results for case study I.

	BSA	AAA	ABC	DE	CMAES
BSA		+++	+++	++	O
AAA	---		+	---	O
ABC	---	-		--	-
DE	--	+++	++		O
CMAES	O	O	+	O	

In case study I, during the early stages of optimization, namely at 25,000 FEs, DE obtains the highest PI as shown in Table 4.1. Later, CMAES edged other algorithms to obtain better PI at 50,000 and 100,000 FEs. However, at the saturation of optimization, BSA obtained the highest PI after 200,000 FEs. According to the t-test in Table 4.2, BSA performed better than DE, AAA and ABC while performing equally well in comparison to CMAES.

Table 4.3: Mean and confidence intervals for case study II.

Algorithm	PI 25,000 FEs	PI 50,000 FEs	PI 100,000 FEs	PI 200,000 FEs
BSA	5.5488 ± 0.0038	5.5668 ± 0.0002	5.5676 ± 0.0000	**5.5677 ± 0.0000**
AAA	5.5642 ± 0.0010	5.5659 ± 0.0004	5.5669 ± 0.0001	5.5673 ± 0.0000
ABC	3.1832 ± 1.1607	5.4637 ± 0.0749	5.5532 ± 0.0072	5.5652 ± 0.0005
DE	**5.5671 ± 0.0001**	5.5676 ± 0.0000	**5.5677 ± 0.0000**	**5.5677 ± 0.0000**
CMAES	0.0000 ± 0.0000	**5.5677 ± 0.0000**	**5.5677 ± 0.0000**	**5.5677 ± 0.0000**

Table 4.4: T-test results for case study II.

	BSA	AAA	ABC	DE	CMAES
BSA		+++	+++	O	O
AAA	---		+++	---	---
ABC	---	---		---	---
DE	O	+++	+++		O
CMAES	O	+++	+++	O	

In case study II, during the early stages of optimization namely at 25,000 FEs, DE obtains the highest PI as shown in Table 4.3. At 50,000 FEs, CMAES improved compared to other algorithms to obtain better PI though DE emerged to perform equally well as CMAES at 100,000 FEs to obtain the highest PI. At the saturation of optimization, BSA, DE and CMAES obtained the highest PI after 200, 000 FEs. According to the t-test in Table 4.4, BSA performed better than AAA and ABC while performing equally well in comparison to CMAES and DE.

Table 4.5: Mean and confidence intervals for case study III.

Algorithm	PI 25,000 FEs	PI 50,000 FEs	PI 100,000 FEs	PI 200,000 FEs
BSA	69.352 ± 22.656	87.487 ± 0.2997	87.876 ± 0.0699	**87.976 ± 0.0251**
AAA	32.433 ± 25.991	85.017 ± 1.0445	85.844 ± 0.6977	86.365 ± 0.7140
ABC	14.733 ± 19.259	78.110 ± 2.4286	78.612 ± 2.1388	78.612 ± 2.1387
DE	43.995 ± 28.743	43.974 ± 28.73	43.99 ± 28.74	43.996 ± 28.744
CMAES	**87.770 ± 0.2776**	**87.968 ± 0.0192**	**87.968 ± 0.0192**	87.968 ± 0.0192

Table 4.6: T-test results for case study III.

	BSA	AAA	ABC	DE	CMAES
BSA		++	+++	O	O
AAA	--		+++	O	--
ABC	---	---		O	---
DE	O	O	O		O
CMAES	O	++	+++	O	

In case study III, prior to convergence of optimization namely at 25,000, 50,000 and 100,000 FEs, CMAES obtains the highest PI as shown in Table 4.5. However, at the convergence of optimization, BSA obtained the highest PI after 200, 000 FEs. According to the t-test in Table 4.6, BSA performed better than AAA and ABC while performing equally well in comparison to CMAES and DE.

Table 4.7: Mean and confidence intervals for case study IV.

Algorithm	PI 25,000 FEs	PI 50,000 FEs	PI 100,000 FEs	PI 200,000 FEs
BSA	89.117 ± 0.1457	**89.404 ± 0.0027**	**89.406 ± 0.0015**	**89.408 ± 0.0012**
AAA	**89.402 ± 0.0049**	**89.404 ± 0.0057**	89.405 ± 0.0057	89.407 ± 0.0045
ABC	89.340 ± 0.0530	89.391 ± 0.0102	89.392 ± 0.0101	89.395 ± 0.0069
DE	89.364 ± 0.0272	89.347 ± 0.0290	89.376 ± 0.0141	89.391 ± 0.0134
CMAES	89.140 ± 0.2024	89.359 ± 0.0407	89.371 ± 0.0387	89.373 ± 0.0382

Table 4.8: T-test results for case study IV.

	BSA	AAA	ABC	DE	CMAES
BSA		O	O	O	O
AAA	O		O	O	O
ABC	O	O		O	O
DE	O	O	O		
CMAES	O	O	O	O	

In case study IV, during the early stages of optimization namely at 25,000 FEs, AAA obtains the highest PI as shown in Table 4.7. At 50,000 FEs, both BSA and AAA obtain the highest PI. However at the later stages of optimization namely at 100,000, and 200,000 FEs, BSA obtained the highest PI. According to the t-test in Table 4.8, all algorithms perform equally well.

Table 4.9: Mean and confidence intervals for case study V.

Algorithm	PI 25,000 FEs	PI 50,000 FEs	PI 100,000 FEs	PI 200,000 FEs
BSA	95.049 ± 0.0211	**95.071 ± 0.0015**	95.072 ± 0.0009	**95.073 ± 0.0001**
AAA	**95.065 ± 0.0083**	95.068 ± 0.0051	**95.073 ± 0.0001**	**95.073 ± 0.0000**
ABC	95.046 ± 0.0176	95.041 ± 0.0127	95.047 ± 0.0110	95.061 ± 0.0089
DE	75.907 ± 24.797	57.042 ± 30.428	57.043 ± 30.429	57.043 ± 30.429
CMAES	0.0000 ± 0.0000	0.0000 ± 0.0000	0.0000 ± 0.0000	0.0000 ± 0.0000

Table 4.10: T-test results for case study V.

	BSA	AAA	ABC	DE	CMAES
BSA		O	O	O	+++
AAA	O		O	O	+++
ABC	O	O		O	+++
DE	O	O	O		+
CMAES	---	---	---	-	

In case study V, during the early stages of optimization, namely at 25,000 FEs, AAA obtains the highest PI as shown in Table 4.9. Later, BSA edged other algorithms to obtain better PI at 50,000 FEs. At 100,000 FEs, AAA obtains the highest PI. At the

saturation of optimization, both BSA and AAA obtained the highest PI after 200,000 FEs. According to the t-test in Table 4.10, BSA performed better than CMAES while performing equally well in comparison to AAA, ABC and DE.

Table 4.11: Mean and confidence intervals for case study VI.

Algorithm	PI 25,000 FEs	PI 50,000 FEs	PI 100,000 FEs	PI 200,000 FEs
BSA	2.5044 ± 0.0028	2.5153 ± 0.0011	2.5186 ± 0.0010	**2.522 ± 0.0010**
AAA	2.5068 ± 0.0024	2.5112 ± 0.0011	2.5142 ± 0.0009	2.5165 ± 0.0007
ABC	2.4739 ± 0.0072	2.4739 ± 0.0072	2.4739 ± 0.0072	2.4739 ± 0.0072
DE	2.5176 ± 0.0004	2.5192 ± 0.0005	**2.5206 ± 0.0004**	2.5219 ± 0.0003
CMAES	**2.5196 ± 0.0012**	**2.5196 ± 0.0012**	2.5196 ± 0.0012	2.5196 ± 0.0012

Table 4.12: T-test results for case study VI.

	BSA	AAA	ABC	DE	CMAES
BSA		+++	+++	O	O
AAA	---		+++	---	--
ABC	---	---		---	---
DE	O	+++	+++		+
CMAES	O	++	+++	-	

In case study VI, during the early stages of optimization namely at 25,000 and 50,000 FEs, CMAES obtains the highest PI as shown in Table 4.11. Later, DE edged other algorithms to obtain better PI at 100,000 FEs. However at the saturation of optimization, BSA obtained the highest PI after 200,000 FEs. According to the t-test in Table 4.12, BSA performed better than AAA and ABC while performing equally well in comparison to DE and CMAES.

The results provide several insights on the capabilities of each algorithm in solving fermentation problems. The problems investigated in this paper can be divided into two categories: constrained and unconstrained. Case study II is unconstrained problem while the rest are constrained problems. For unconstrained problem, all algorithms performed almost equally well and saturated at almost the same PI value. This means that for

unconstrained problems, there is flexibility in choosing an algorithm to solve a given problem as most of them converged to the same solution. However, a different scenario exists for constrained problems. For constrained problems, different algorithms performed differently in each problem with the exception of BSA. In overall, BSA is able to obtain the best results in all case studies by providing the highest means and narrow confidence interval. BSA obtained the highest means at 200,000 FEs for all problems except for case II where DE and CMAES saturated at the same highest value as BSA. Case V is an exception for constrained problem where AAA managed to obtain equal means as BSA. Even though DE and CMAES obtained higher means than BSA at NFE lower than 200,000 for some cases, BSA manages to obtain higher means than both algorithms at the end of 200,000 FEs for all constrained problems. This shows that when given a sufficient amount of NFE, BSA is the best option for solving constrained fermentation problems and provides improved performance compared to DE and other metaheuristics studied in this work for solving bioreactor application problems in general.

AAA shows equal in performance as BSA for case IV and case V while it performs worse in other problems especially for case I and case III. ABC performs the worst in all the case studies except for case IV and case V where it performs relatively well. DE performs well for case I, II, IV and VI. However, it shows significantly worse results for case III and the V because of the difficulty of satisfying the constraints in these problems. Case III has three constraints to be satisfied, while case V has a single strict constraint as compared to other problems which either have more relaxed constraint or no constraints. CMAES performs well for most cases and even converged faster than BSA in case I, II, III and VI. However, it struggles to solve case V for the same reason as DE. Previously, M. Rocha et al. (2014) found that DE obtains the best overall performance for fed-batch fermentation problems. BSA, as an improved DE-based

algorithm is expected to perform better than DE. The results obtained from our experiments confirmed that BSA is a superior algorithm.

4.2 Multi-objective optimization problems

4.2.1 Benchmark problems

We compare our algorithm with four other multi-objective algorithms namely multi-objective grey wolf optimizer (MOGWO), multi-objective particle swarm optimizer (MOPSO), multi-objective evolutionary algorithm based on decompositions (MOEA/D) and multi-objective differential evolution (MODE). In this experiment, 20 standard multi-objective test problems proposed in CEC 2009 are used (Qingfu Zhang et al., 2008) to evaluate the contested algorithms. These test problems consist of 10 unconstrained functions (UF1-UF10) and 10 constrained functions (CF1-CF10). They are considered as one of the most challenging test problems in the literature that provide different multi-objective search spaces with different Pareto optimal fronts: convex, non-convex, discontinuous, and multi-modal. For all problems, 100 search agents are utilized and the algorithms are run for a maximum of 300,000 function evaluations. The number of parameters or variables for each of the unconstrained test functions is 30 while for the number of variables for constrained test functions is 10. The parameters for all algorithms are set at default values as recommended by their respective author. The two parameters used for M-MOPSO are as follows:

- $S_{f1} = 10$: saturation factor
- $S_{f2} = 0.97$: shrink factor

In M-MOPSO, these two parameters influence the converging behavior of the algorithm. These parameter values should be fine-tuned and in our experiment, we found the best parameter values after rigorous trial and error. As both MOPSO and M-

MOPSO use the same external repository system, they both use the same following parameters:

- $nGrid = 10$: number of grids per dimension
- $nRep = number\ of\ search\ agents$: repository size
- $\alpha = 0.1$: grid inflation rate
- $\beta = 2$: leader selection pressure
- $\gamma = 2$: deletion selection pressure

The performance metric used for comparison is the Inverted Generational Distance (IGD) proposed by Sierra and Coello Coello (2005) which is used for measuring convergence and spread. The mathematical formulation for IGD is as follow:

$$IGD = \frac{\sqrt{\sum_{i=1}^{n} d_i^2}}{n} \tag{71}$$

where n is the number of true Pareto optimal solutions and d_i indicates the Euclidean distance between the ith true Pareto optimal solution in the reference set and the closest obtained Pareto optimal solutions by the algorithm. The Euclidean distance is calculated for every true solution with respect to its nearest obtained Pareto optimal solutions in the objective space. In our experiment, we used 1,000 evenly distributed solutions as the reference set for test instances with two objectives except for CF1 which used 21 solutions while 10,000 solutions are used for test instances with three objectives.

4.2.1.1 Unconstrained problems

The algorithms are run 30 times for each test problem and the statistics results are provided in Table 4.13 and Figure 4.3. Aside from the above quantitative evaluation, we also provide the means for qualitative evaluations by illustrating the best set of Pareto

optimal solutions obtained by each algorithm on the objective space. These qualitative results are provided in Figure 4.4 - 4.13.

The first three variables of the Pareto sets after 50,000, 100,000 and 300,000 number of function evaluations (NFE) are illustrated in Figure 4.1 and 4.2. The rate of convergence can be observed from Figure 4.1 and 4.2. It shows that during initial search process at NFE=10,000, only a small number of nondominated solutions are stored in the external archive and the solutions are distributed randomly in the search space. After 100,000 NFE, more solutions were approaching to the true Pareto set. At the final stage of search (NFE=300,000), the maximum number of 100 solutions are stored in the archive and they mostly cover the true Pareto set. At this stage, the convergence and coverage of M-MOPSO can be clearly seen.

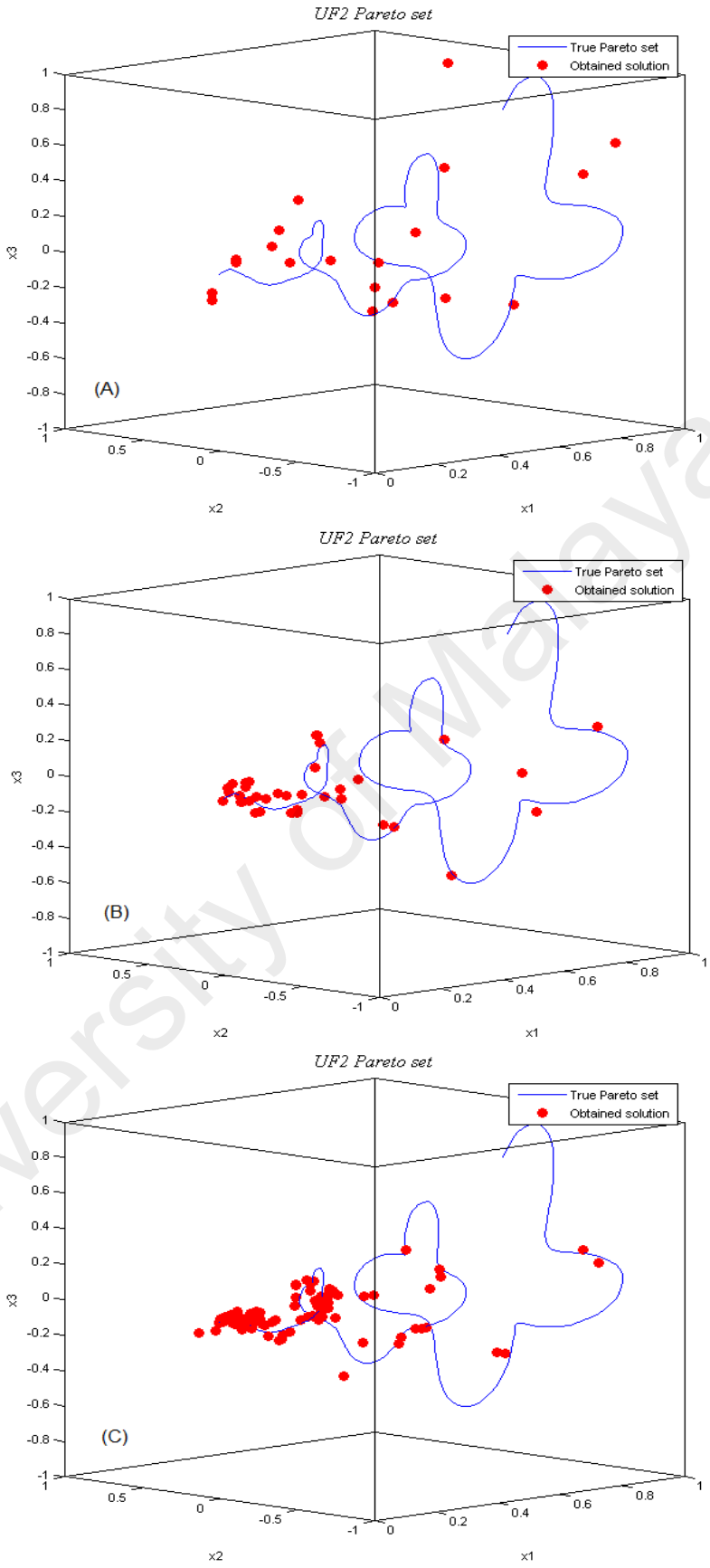

Figure 4.1: True and obtained Pareto sets of M-MOPSO for UF2: (A) at 10,000 function evaluations, (B) at 100,000 function evaluations, (C) at 300,000 function evaluations.

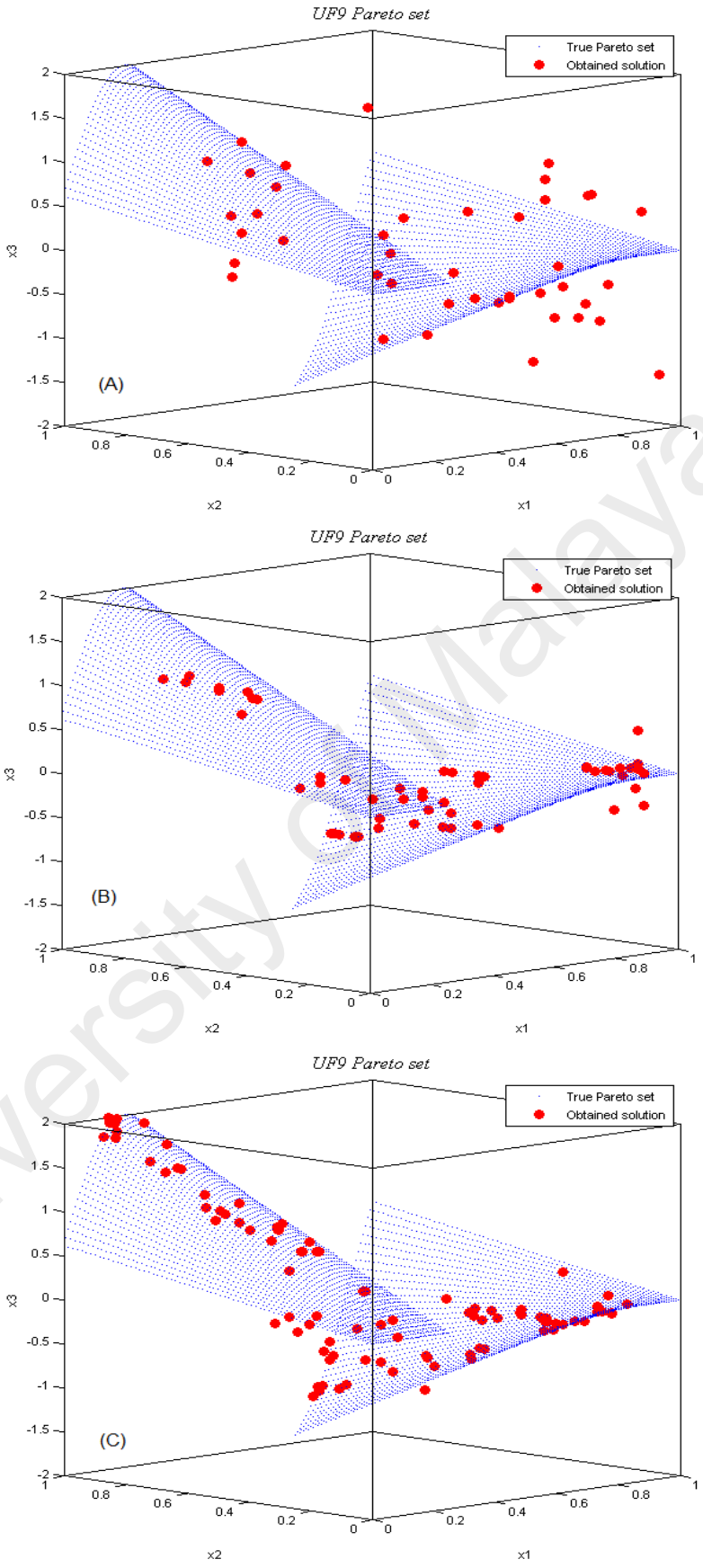

Figure 4.2: True and obtained Pareto sets of M-MOPSO for UF9: (A) at 10,000 function evaluations, (B) at 100,000 function evaluations, (C) at 300,000 function evaluations.

Table 4.13 describes the IGD result based on average, standard deviation, median and worst outcome from 30 test simulations on 10 unconstrained benchmark multi-objective problems of CEC 2009. The tested algorithms are MOGWO, MOPSO, MOEA/D, MODE and M-MOPSO. Based on Table 4.13, M-MOPSO showed improved average result in comparison to all other algorithms in all benchmark problems except for problem UF2 and UF8. For UF2, MODE obtained the best average while for UF8, the best average was obtained by MOEA/D. It is worth noting that for UF2, M-MOPSO was the third best algorithm after MODE and MOEA/D in term of average result while for UF8, M-MOPSO was the second best after MOEA/D. This shows that in overall, M-MOPSO have good convergence for multi-objective problems and rivals other algorithms used in this experiment.

Table 4.13: IGD results for unconstrained CEC 2009 benchmark problems.

	UF1 (bi-objective)					UF2 (bi-objective)				
IGD	MOGWO	MOPSO	MOEA/D	MODE	M-MOPSO	MOGWO	MOPSO	MOEA/D	MODE	M-MOPSO
Average	0.10955	0.10566	0.15906	0.03365	**0.02642**	0.06042	0.05577	0.02024	**0.01380**	0.02050
Median	0.10950	0.10840	0.12142	0.03345	**0.02669**	0.06084	0.05584	**0.01215**	0.01345	0.01987
STD. Dev.	0.00553	0.00968	0.09000	**0.00179**	0.00395	0.00956	0.00526	0.01696	**0.00105**	0.00278
Worst	0.11842	0.12226	0.31089	0.03719	**0.03286**	0.08232	0.06902	0.06325	**0.01699**	0.02750
	UF3 (bi-objective)					UF4 (bi-objective)				
IGD	MOGWO	MOPSO	MOEA/D	MODE	M-MOPSO	MOGWO	MOPSO	MOEA/D	MODE	M-MOPSO
Average	0.25349	0.39195	0.28254	0.18019	**0.08247**	0.05921	0.07847	0.06363	0.04595	**0.03319**
Median	0.25625	0.40898	0.28788	0.17864	**0.07990**	0.05880	0.08004	0.06322	0.04641	**0.03282**
STD. Dev.	0.05642	0.05864	0.03000	**0.01058**	0.01291	0.00226	0.00561	0.00453	**0.00148**	0.00218
Worst	0.34665	0.46685	0.32290	0.20554	**0.11507**	0.07031	0.08640	0.07296	0.04875	**0.04240**
	UF5 (bi-objective)					UF6 (bi-objective)				
IGD	MOGWO	MOPSO	MOEA/D	MODE	M-MOPSO	MOGWO	MOPSO	MOEA/D	MODE	M-MOPSO
Average	0.69931	0.43357	0.94569	0.77167	**0.16981**	0.28241	0.38170	0.46731	0.49191	**0.15018**
Median	0.62907	0.40824	0.92679	0.77098	**0.17529**	0.28452	0.32407	0.46227	0.48846	**0.15447**
STD. Dev.	0.32321	0.20739	0.17414	0.05703	**0.02186**	0.04076	0.20857	0.10427	**0.01555**	0.06454
Worst	2.19045	1.18900	1.28565	0.87944	**0.21685**	0.38275	0.85390	0.81389	0.52104	**0.31263**
	UF7 (bi-objective)					UF8 (tri-objective)				
IGD	MOGWO	MOPSO	MOEA/D	MODE	M-MOPSO	MOGWO	MOPSO	MOEA/D	MODE	M-MOPSO
Average	0.09982	0.15624	0.32143	0.11995	**0.02825**	2.73042	0.29777	**0.12909**	0.21322	0.13808
Median	0.06949	0.06208	0.40413	0.10905	**0.02802**	3.25743	0.29766	**0.11940**	0.20849	0.13505
STD. Dev.	0.08411	0.14338	0.21034	0.03745	**0.00486**	1.24334	0.04143	0.02528	0.02469	**0.01655**
Worst	0.35172	0.40408	0.58568	0.20833	**0.03742**	4.15865	0.36733	0.18874	0.26685	**0.17958**

Table 4.13, continued.

IGD	UF9 (tri-objective)					UF10 (tri-objective)				
	MOGWO	MOPSO	MOEA/D	MODE	M-MOPSO	MOGWO	MOPSO	MOEA/D	MODE	M-MOPSO
Average	0.32416	0.37910	0.16876	0.22429	**0.07254**	2.55824	1.14397	0.45272	1.03144	**0.31347**
Median	0.23950	0.36722	0.17164	0.22455	**0.07146**	1.04314	1.07498	0.49515	1.01960	**0.30887**
STD. Dev.	0.19293	0.05252	0.01787	0.02450	**0.00828**	4.05456	0.57334	0.19059	0.07885	**0.03822**
Worst	0.79260	0.51674	0.17960	0.28344	**0.09267**	15.33101	2.47642	0.73603	1.22866	**0.40404**

Figure 4.3 shows the boxplot for IGD results based on 30 test simulations on 10 unconstrained benchmark multi-objective problems of CEC 2009. Based on Figure 4.3, M-MOPSO showed improved average result in comparison to all other algorithms in all benchmark problems except for UF2 and UF8. Except for UF2 and UF8, the boxplots for M-MOPSO are lower and narrower compared to others. This shows that M-MOPSO has better convergence, spread and stability compared to other algorithms.

Figure 4.4 - 4.13 illustrate both the coverage and convergence of the best solution found by each algorithm. The higher proximity of solutions to the true Pareto front reveals the improved convergence of the solutions obtained. M-MOPSO managed to construct well-defined Pareto fronts which closely resemble the true optimal Pareto fronts of each test functions.

For UF1, M-MOPSO obtained more distributed solutions and closer to the true Pareto front compared to the others, as shown in Figure 4.4. For UF2, M-MOPSO, MOEA/D and MODE obtained almost similar results, which are well distributed and have good convergence toward the true Pareto front, as shown in Figure 4.5. In Figure 4.6, M-MOPSO clearly shown better convergence and spread throughout the true Pareto front of UF3 when compared to other algorithms. For UF4, even though all algorithms have good Pareto front distribution, M-MOPSO has the advantage of better convergence towards the true Pareto front, as shown in Figure 4.7. Figure 4.8 shows that M-MOPSO obtained solutions which are closest to the true solutions compared to the solutions

obtained by other algorithms for UF5. From Figure 4.9, we can see that the true Pareto front for UF6 consists of three separate parts. M-MOPSO managed to find solutions which are close to all three parts. In Figure 4.10, MOEA/D shows good convergence and distribution of its solutions except at the lower part of its Pareto front. For M-MOPSO, even though its solutions are not as well distributed as MOEA/D's, the Pareto front covers almost all areas of the true Pareto front. Figures 4.11, 4.12 and 4.13 show the Pareto fronts for the three objectives problems of UF8, UF9 and UF10 respectively. For UF8 and UF9, M-MOPSO shows better convergence and distribution compared to the others while for UF10, MOEA/D has the best result.

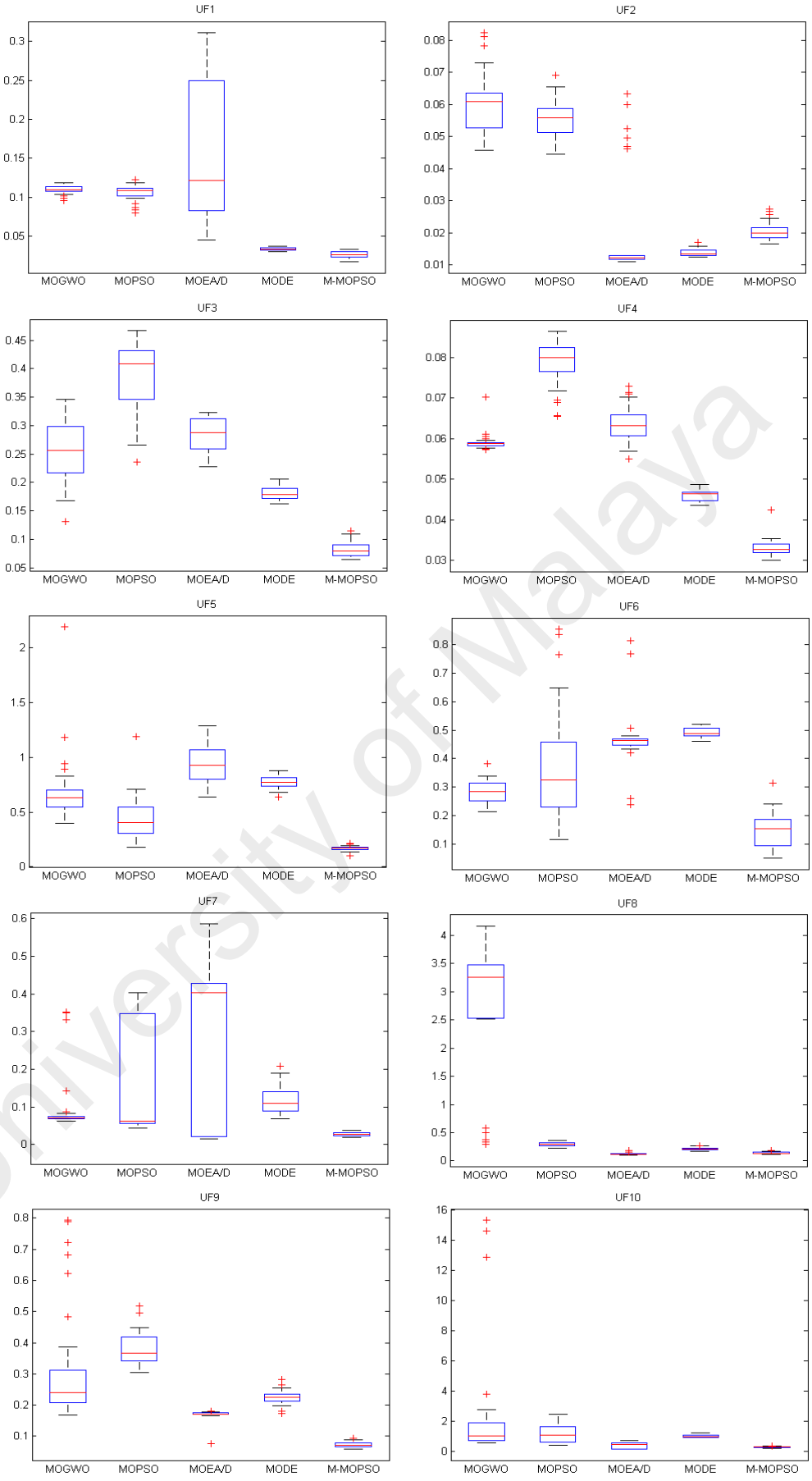

Figure 4.3: Boxplot for the statistical results for IGD on UF1 to UF10.

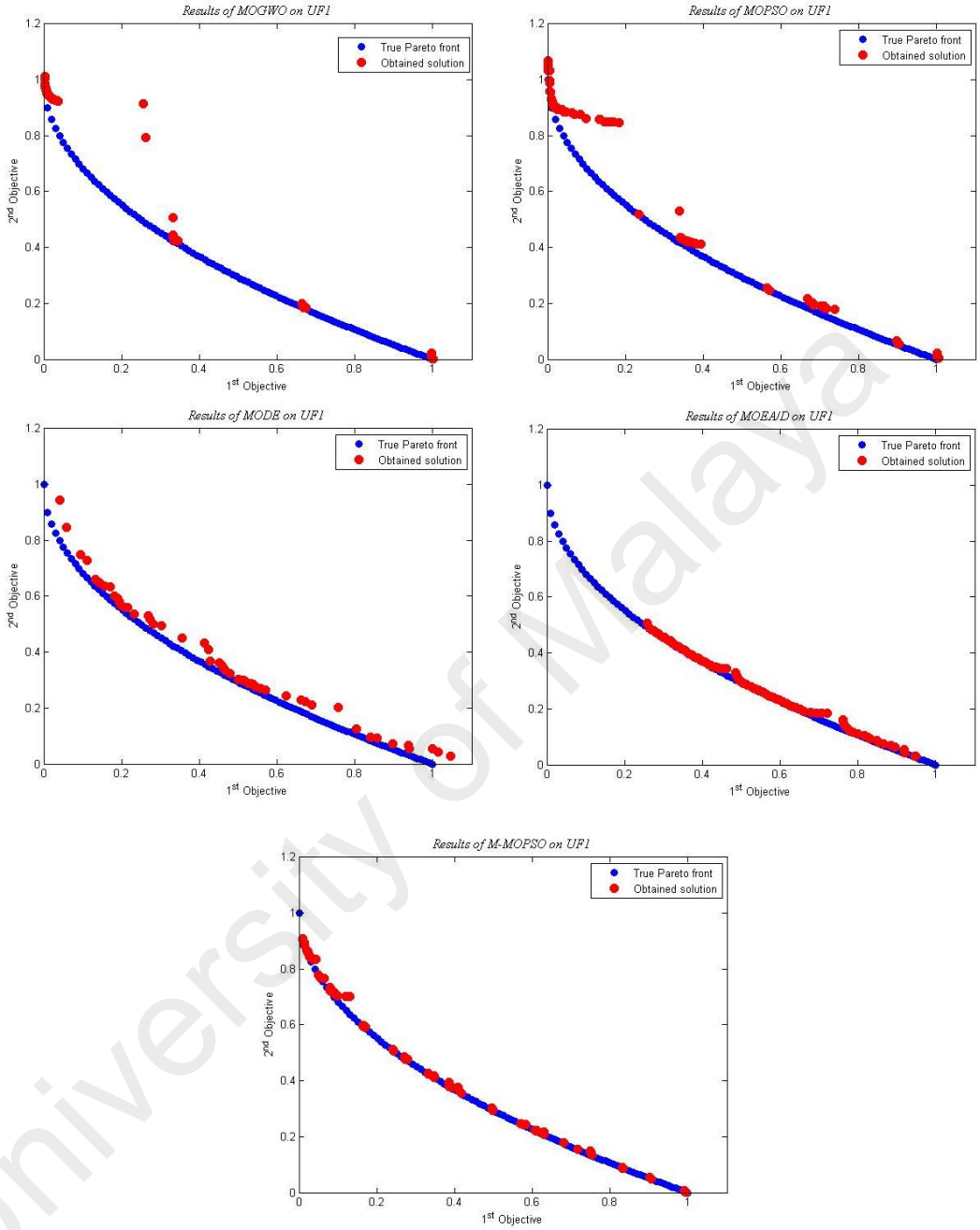

Figure 4.4: Obtained Pareto solutions for UF1.

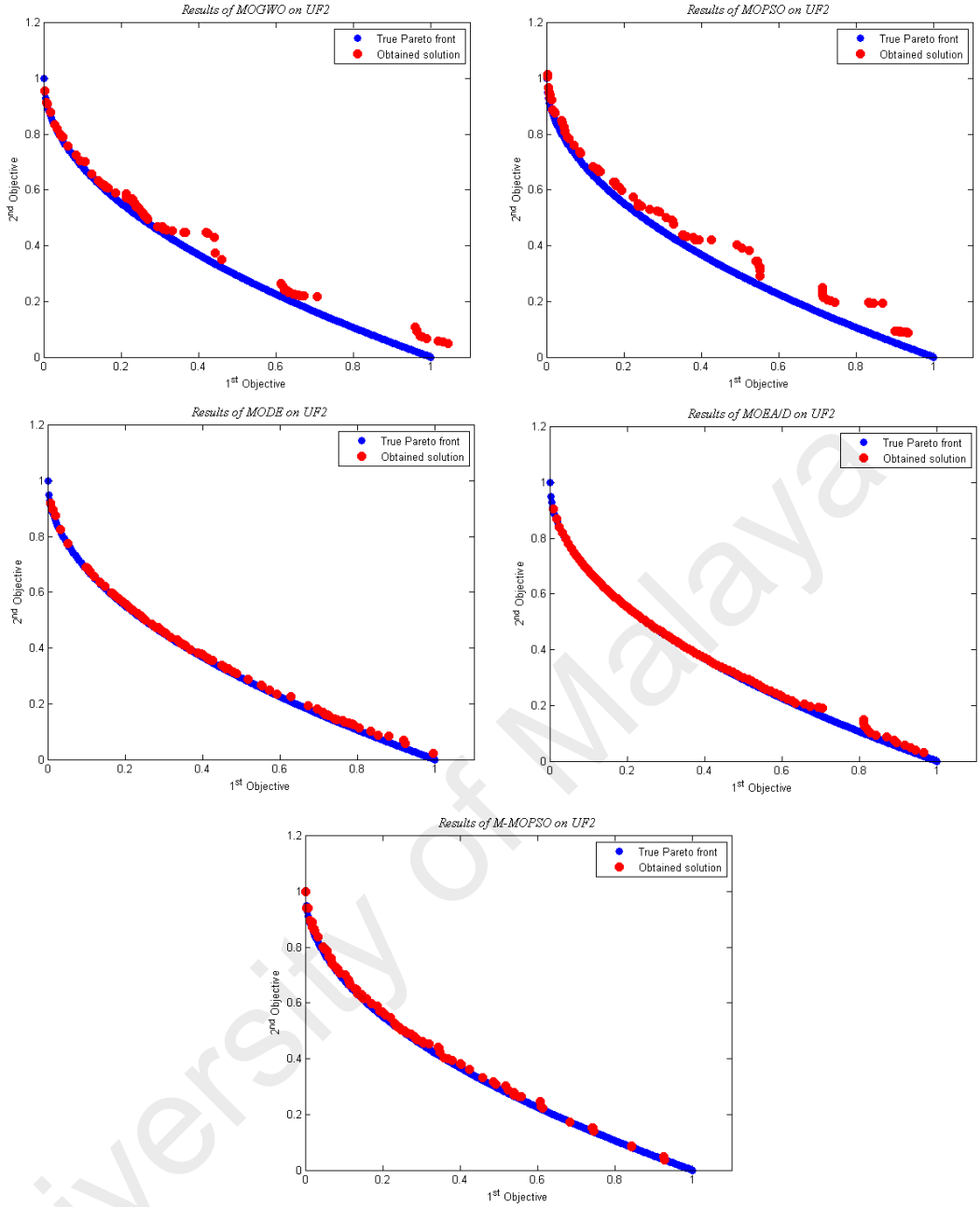

Figure 4.5: Obtained Pareto solutions for UF2.

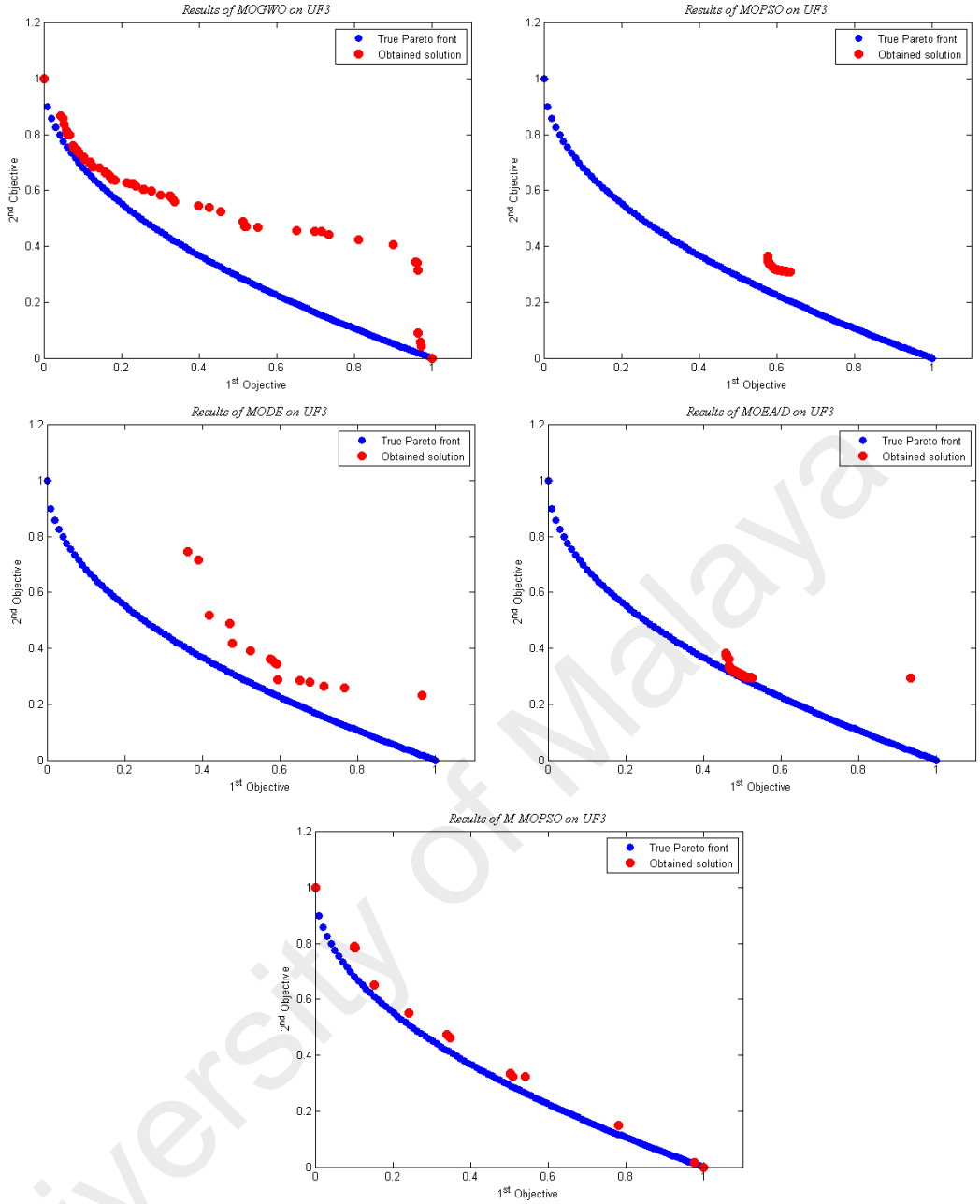

Figure 4.6: Obtained Pareto solutions for UF3.

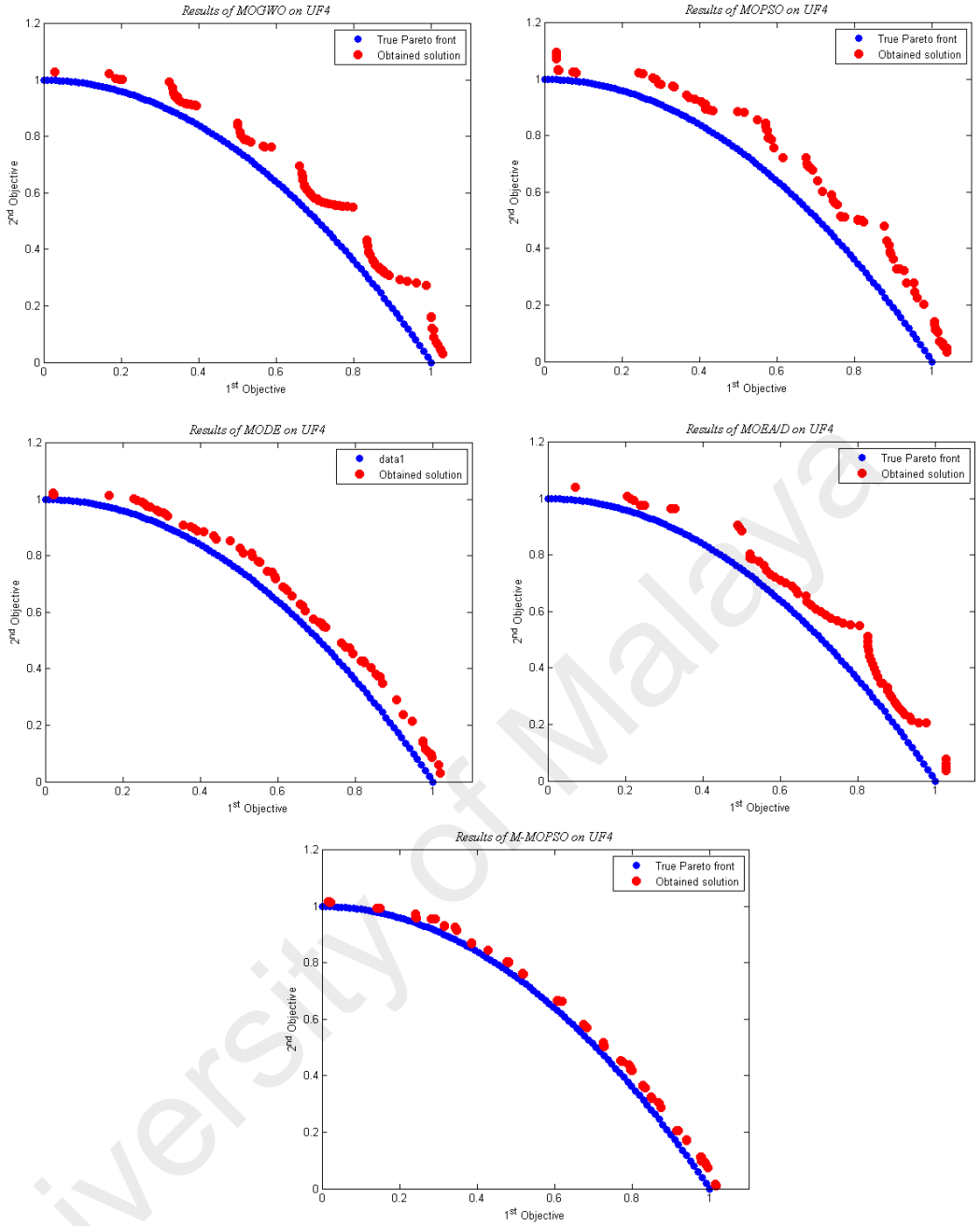

Figure 4.7: Obtained Pareto solutions for UF4.

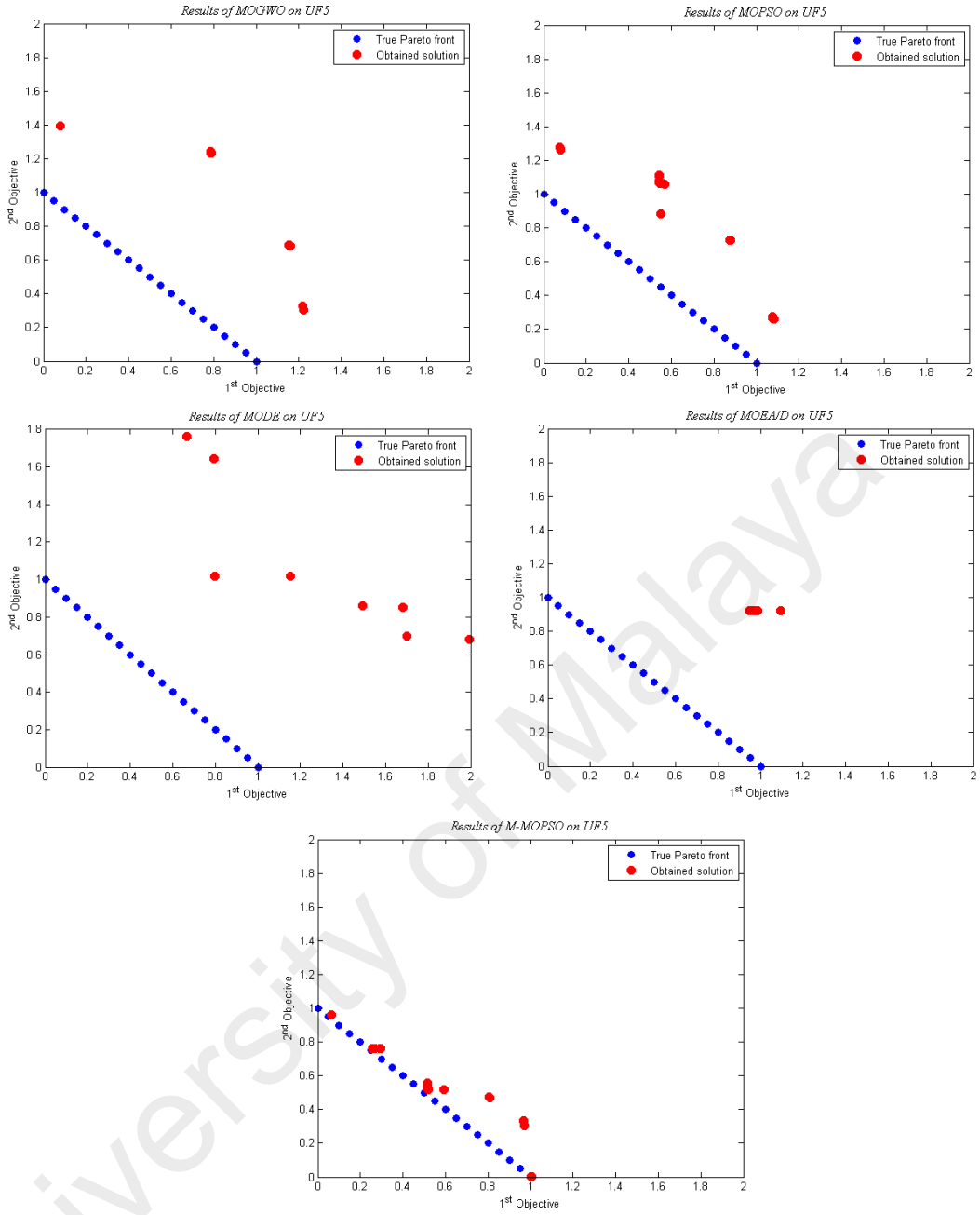

Figure 4.8: Obtained Pareto solutions for UF5.

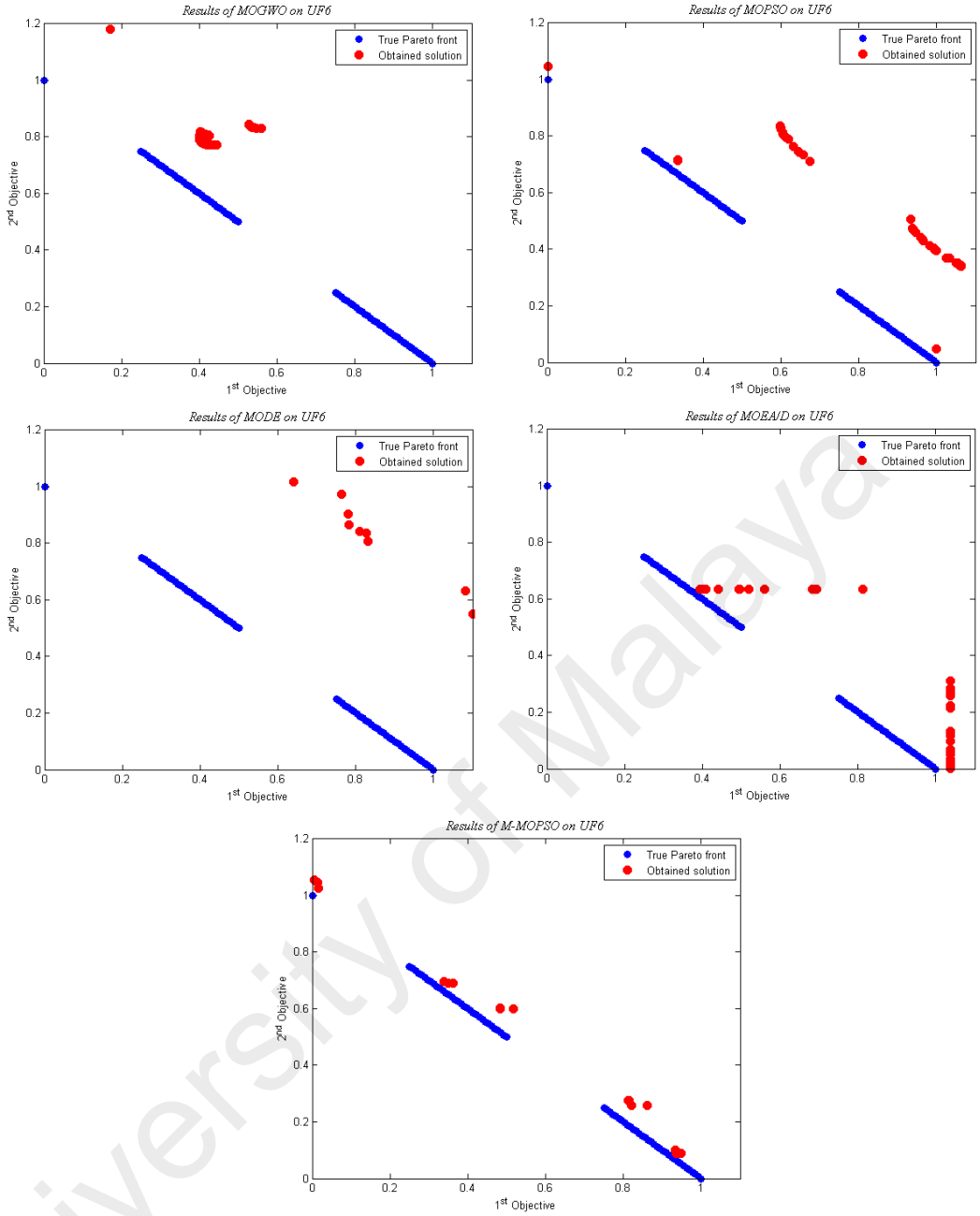

Figure 4.9: Obtained Pareto solutions for UF6.

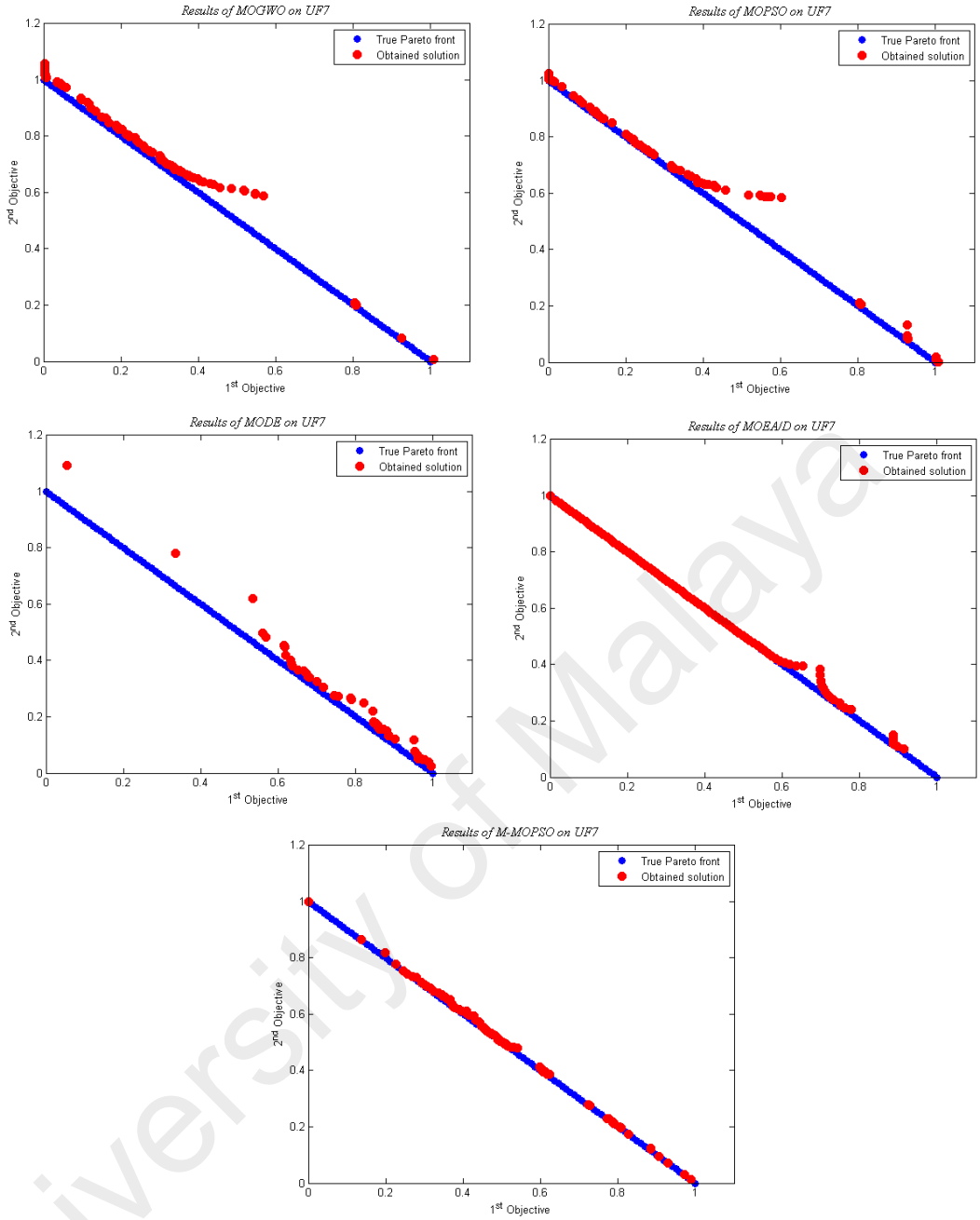

Figure 4.10: Obtained Pareto solutions for UF7.

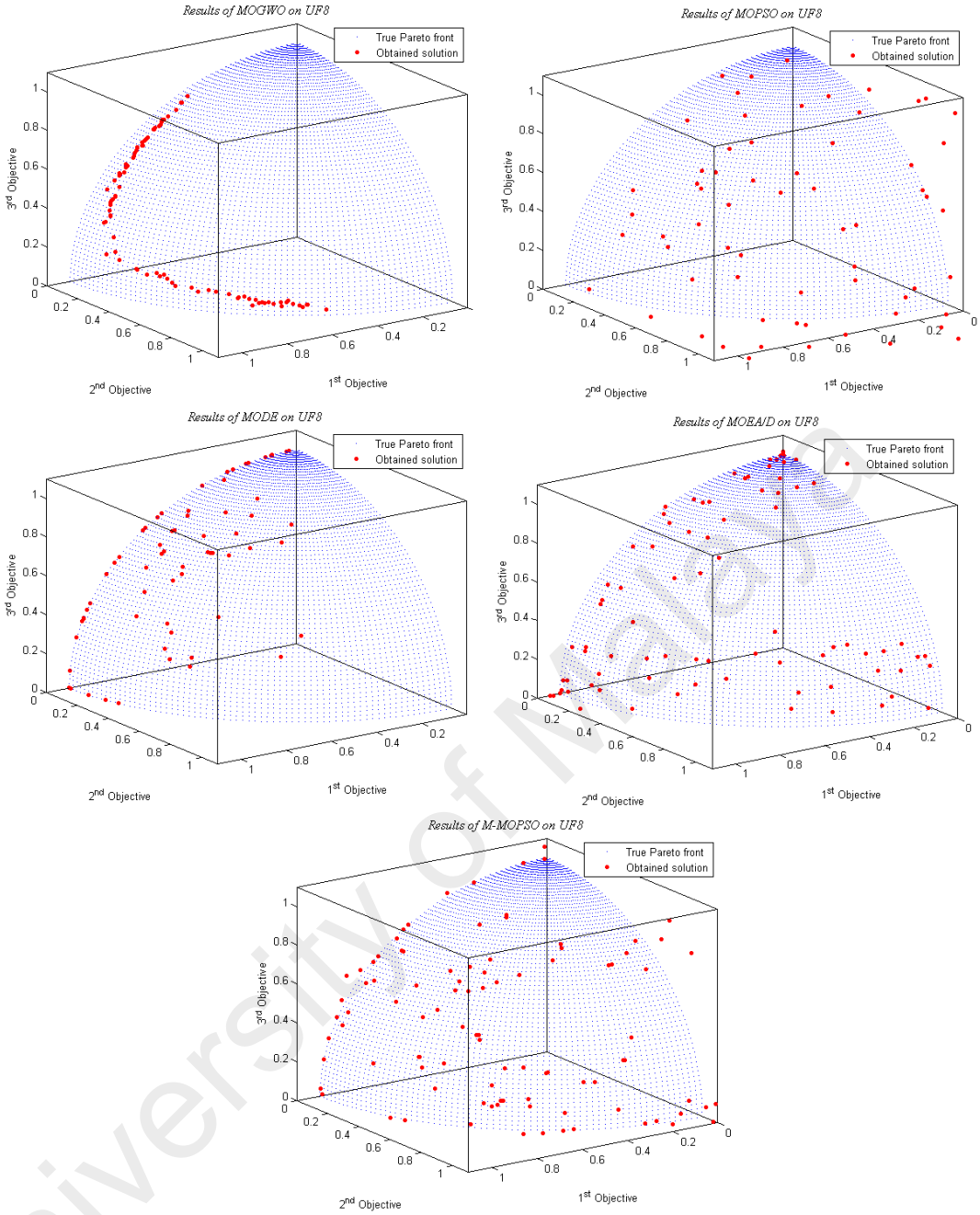

Figure 4.11: Obtained Pareto solutions for UF8.

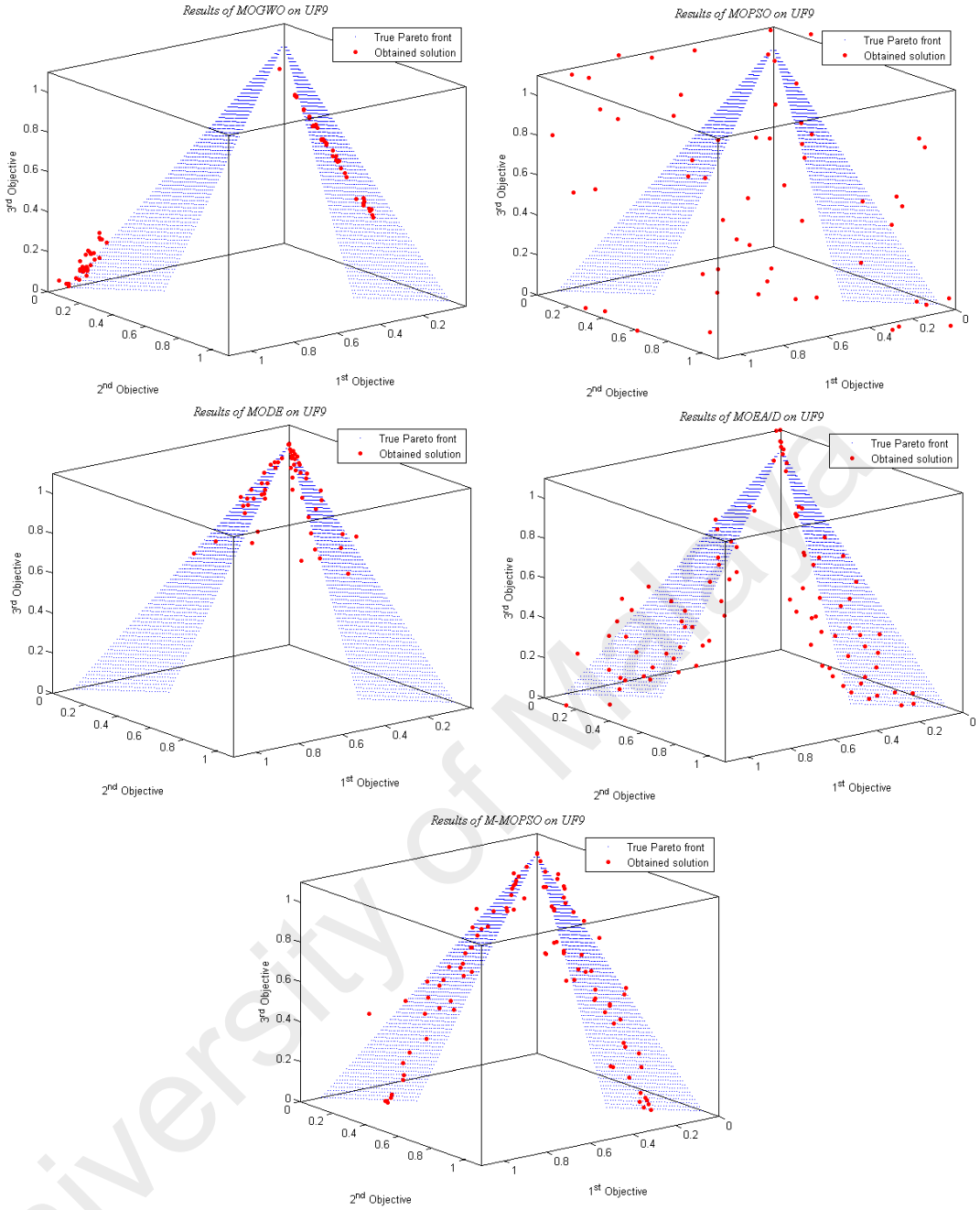

Figure 4.12: Obtained Pareto solutions for UF9.

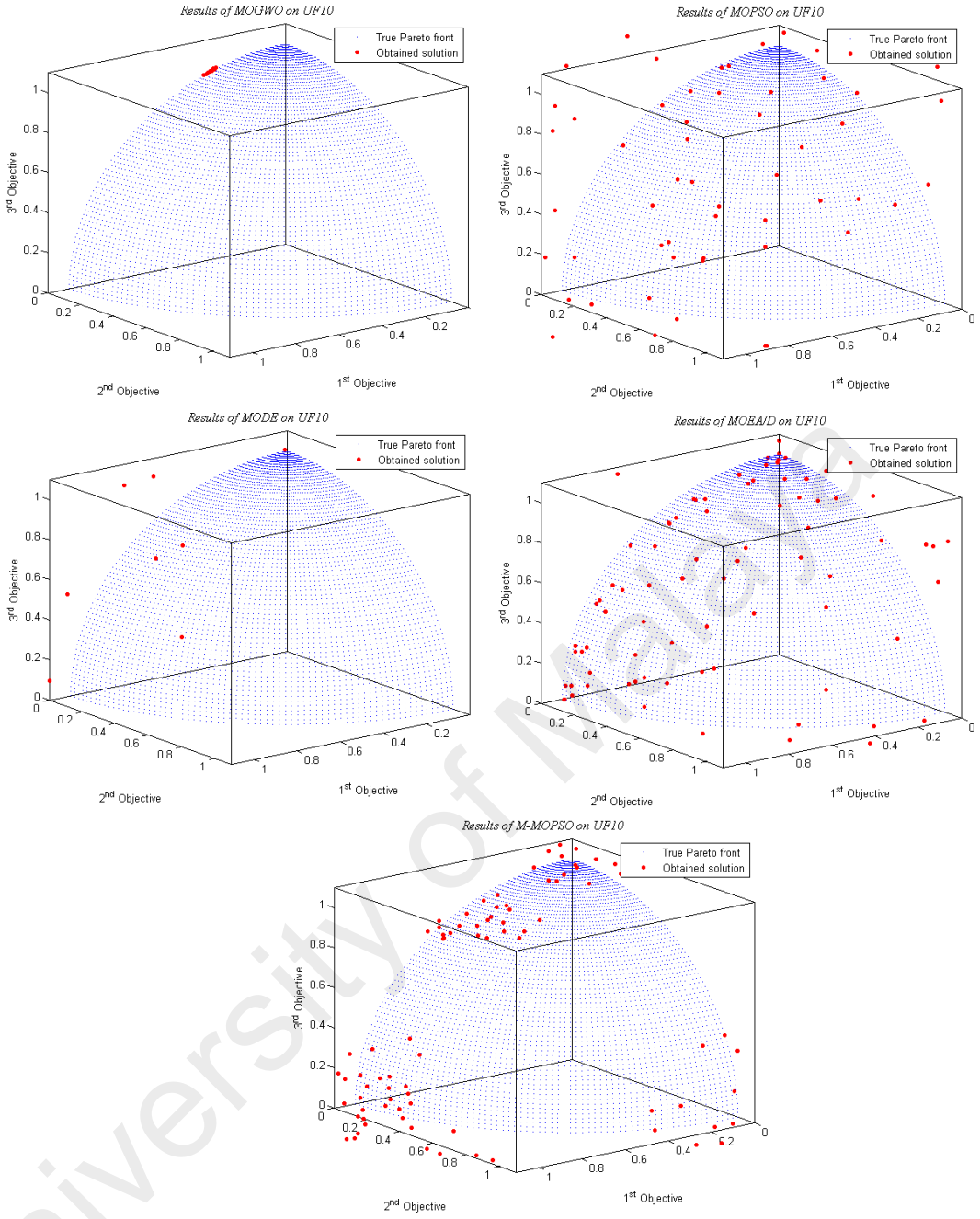

Figure 4.13: Obtained Pareto solutions for UF10.

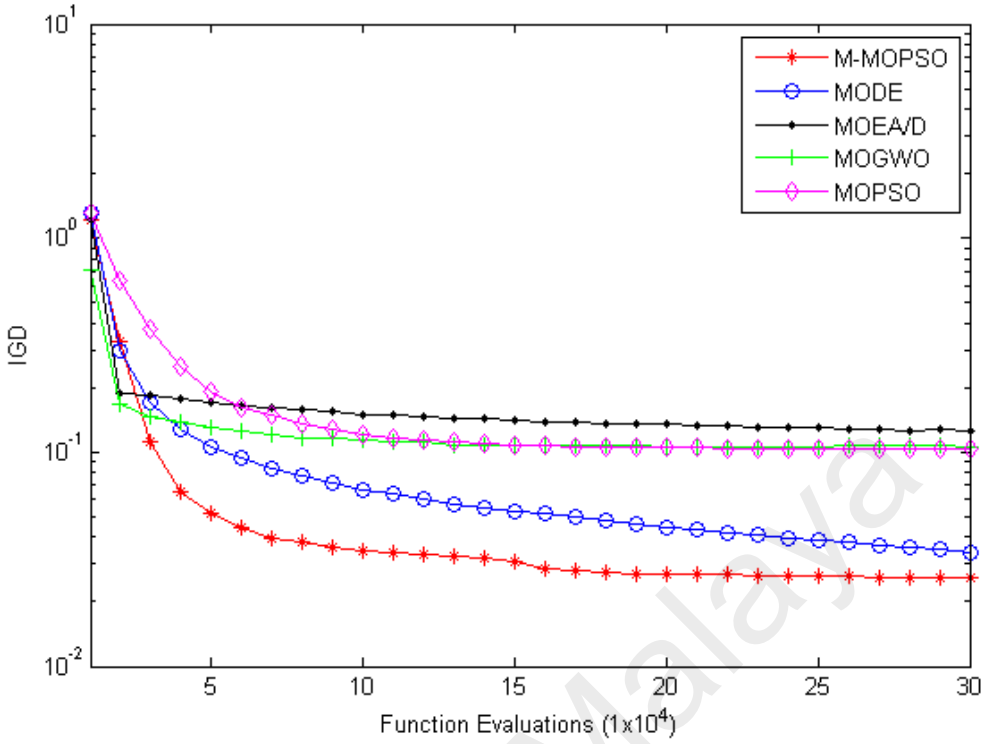

Figure 4.14: Convergence graph for UF1.

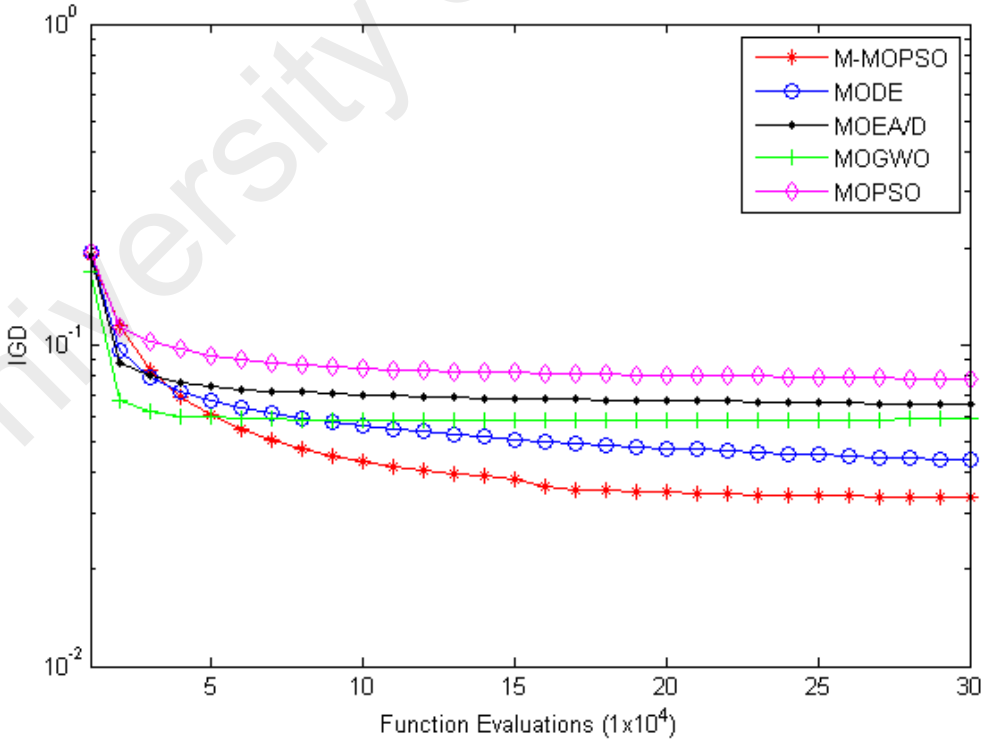

Figure 4.15: Convergence graph for UF4.

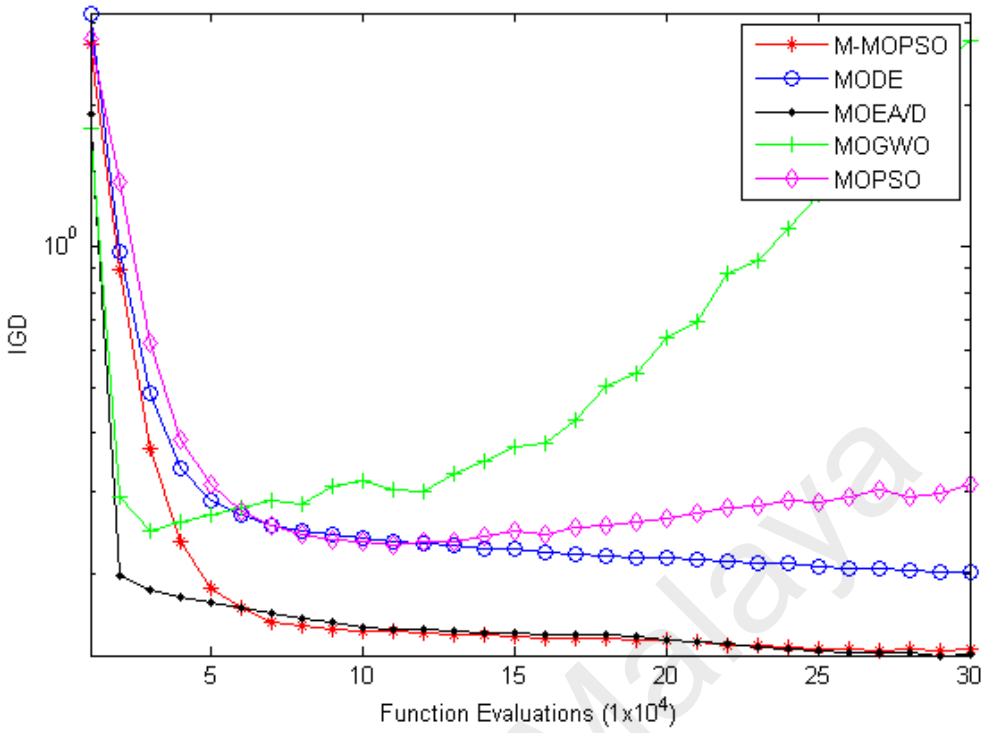

Figure 4.16: Convergence graph for UF8.

Figure 4.14 shows the convergence graph for UF1. The graph shows the average IGD of 30 runs recorded at 30 intervals of function evaluations (FE) obtained by each algorithm. MOGWO and MOEA/D converge faster by obtaining lower average IGD compared to M-MOPSO and other algorithms at 20,000 FE. However, they become saturated quicker than other algorithms and trapped at local optima after around 30,000 FE. Meanwhile, M-MOPSO and MODE manage to avoid premature convergence and continue to converge gradually until termination, with M-MOPSO showing more noticeable improvements after every FE compared to others.

The same scenario occurs in Figure 4.15 which shows the convergence graph for UF4. MOGWO and MOEA/D converge quicker but also saturates faster compared to M-MOPSO and MODE. M-MOPSO also obtains the best convergence curve after 300,000 FE compared to others.

In Figure 4.16 which shows the convergence graph for UF8, MOGWO and MOEA/D converge faster than others by obtaining lower average IGD until at around 30,000 FE. However, after 30,000 FE, MOGWO's performance becomes worse. The same problem occurs for the original MOPSO, where its performance worsens after around 140,000 FE. This negative behaviour can be attributed to repository member deletion procedure implemented by both algorithms. This procedure is executed after the maximum number of repository members is obtained and the algorithm proceeds to delete one of their members based on the density of the grids. Depending on the member selected to be deleted, the IGD value can either improve or worsen. M-MOPSO eliminates the possibility to delete the 'wrong' member by implementing distance-based member deletion as opposed to density-based member deletion. Hence, it is able to maintain the quality of its solutions throughout the run as shown in the convergence graph. MODE and MOEA/D also maintain the smooth descend of their convergence curve as both does not employ external archive to store the solutions.

4.2.1.2 Constrained problems

In order to verify the their performance in solving constraint problems, the algorithms are run 30 times for each test problem and the statistics results are provided in Table 4.14 and Figure 4.17. To represent the qualitative results, the best set of Pareto optimal solutions obtained by each algorithm on the objective space are provided in Figure 4.18-4.27.

Based on Table 4.14, M-MOPSO showed improved average result in comparison to all other algorithms in all benchmark problems except for problem CF1 and CF10. MOEA/D obtained the best average for CF1 and CF10. It is worth noting that for CF1 and CF10, M-MOPSO was the second best algorithm after MOEA/D in term of average

result. This shows that in overall, M-MOPSO have good convergence for constrained multi-objective problems and rivals other algorithms used in this experiment.

Table 4.14: IGD results for constrained CEC 2009 benchmark problems.

	CF1 (bi-objective)					CF2 (bi-objective)				
IGD	MOGWO	MOPSO	MOEA/D	MODE	M-MOPSO	MOGWO	MOPSO	MOEA/D	MODE	M-MOPSO
Average	0.01284	0.05009	**0.00342**	8.56482	0.00555	0.11466	0.08856	0.10155	0.19962	**0.01319**
Median	0.01337	0.04838	**0.00326**	8.36916	0.00562	0.12056	0.07102	0.09299	0.19901	**0.00718**
STD. Dev.	0.00276	0.00755	0.00132	0.51661	**0.00060**	0.02889	0.03743	0.05864	0.02332	**0.02158**
Worst	0.01651	0.06521	0.00717	9.35663	**0.00700**	0.17114	0.17985	0.27604	0.24734	**0.09347**
	CF3 (bi-objective)					CF4 (bi-objective)				
IGD	MOGWO	MOPSO	MOEA/D	MODE	M-MOPSO	MOGWO	MOPSO	MOEA/D	MODE	M-MOPSO
Average	0.89932	0.52639	0.64998	0.23541	**0.16287**	0.15811	0.11103	0.69385	0.91112	**0.03307**
Median	0.78064	0.51161	0.66162	0.23434	**0.15667**	0.11689	0.10641	0.67518	0.92238	**0.03156**
STD. Dev.	0.43240	0.16151	0.04619	0.02519	**0.04166**	0.13796	0.01516	0.03798	0.03850	**0.00659**
Worst	2.24078	0.89686	0.67927	0.28927	**0.24743**	0.64263	0.13524	0.76853	0.95548	**0.05787**
	CF5 (bi-objective)					CF6 (bi-objective)				
IGD	MOGWO	MOPSO	MOEA/D	MODE	M-MOPSO	MOGWO	MOPSO	MOEA/D	MODE	M-MOPSO
Average	0.55472	0.55721	0.68451	3.54554	**0.20899**	0.08492	0.13105	0.81639	4.30615	**0.06520**
Median	0.56186	0.57480	0.67518	3.69688	**0.19251**	0.08118	0.12431	0.81639	4.02548	**0.05716**
STD. Dev.	0.08743	0.11383	0.02848	1.35938	**0.07403**	0.02194	0.03990	**0.00000**	3.60753	0.02877
Worst	0.72049	0.67521	0.76853	5.73554	**0.40945**	0.16408	0.21753	0.81639	13.2606	**0.12662**
	CF7 (bi-objective)					CF8 (tri-objective)				
IGD	MOGWO	MOPSO	MOEA/D	MODE	M-MOPSO	MOGWO	MOPSO	MOEA/D	MODE	M-MOPSO
Average	1.39173	0.37812	0.81639	45.8691	**0.16155**	4.82411	4.29267	0.77133	283.612	**0.19009**
Median	1.07876	0.33577	0.81639	46.3603	**0.14967**	0.61483	0.50137	**0.14983**	272.578	0.18861
STD. Dev.	0.89057	0.15463	**0.00000**	5.5944	0.06373	8.00306	9.97090	3.08474	65.151	**0.01965**
Worst	3.41388	0.95016	0.81639	54.1519	**0.29748**	23.64818	47.3373	17.07868	423.237	**0.24888**
	CF9 (tri-objective)					CF10 (tri-objective)				
IGD	MOGWO	MOPSO	MOEA/D	MODE	M-MOPSO	MOGWO	MOPSO	MOEA/D	MODE	M-MOPSO
Average	2.48865	0.26847	0.13579	268.447	**0.11914**	3.14974	4.36486	**0.59738**	231.203	0.98001
Median	2.55997	0.27029	0.13789	266.811	**0.11640**	1.01020	5.42038	**0.45683**	240.133	0.47228
STD. Dev.	1.95641	0.03127	0.02274	51.116	**0.01724**	4.71217	2.96877	**0.94537**	43.123	1.39953
Worst	5.78166	0.32490	0.17516	378.677	**0.16263**	19.36988	8.57767	5.56766	321.288	**5.03013**

Figure 4.17 shows the boxplot for IGD results. Based on Fig. 18, M-MOPSO showed noticeable improvement compared to other algorithms for CF2, CF3, CF4 and CF5. For other benchmark problems, the distinction between each algorithm is difficult to observe due to the noticeably worse performance of MODE compared to others.

Figure 4.18-4.27 illustrate both the coverage and convergence of the best solution found by each algorithm. The higher proximity of solutions to the true Pareto front reveals the improved convergence of the solutions obtained. M-MOPSO managed to construct well-defined Pareto fronts which closely resemble the true optimal Pareto fronts of each test functions.

For CF1, M-MOPSO, MOEA/D and MOGWO obtained almost perfect solutions, as shown in Figure 4.18. For CF2, both M-MOPSO and MOEA/D obtained good solutions which cover almost all the areas of the true Pareto front. For CF3, as shown in Figure 4.20, there are three separate parts of true Pareto front. M-MOPSO found four solutions which are close to each part. Figure 4.21 shows a peculiarity where only three out of the five algorithms found solutions close to the true Pareto front. The three algorithms are M-MOPSO, MOGWO and MOPSO. All these three algorithms have one thing in common: they all employ external repository/archive to store non-dominated solutions. This shows the importance of archiving system in solving CF4. Also, M-MOPSO have better convergence compared to MOGWO and MOPSO. For CF5 and CF6 as shown in Figure 4.22 and 4.23 respectively, M-MOPSO converged better than other algorithms. It covers a larger portion of the Pareto front compared to others. For CF7, M-MOPSO obtained a more distributed solution which all converged on Pareto front compared to others, as shown in Figure 4.24. In Figure 4.25, there are five disconnected parts which represent the Pareto front of CF8 in three dimensional objective space. M-MOPSO obtained solutions that cover almost all parts. In Figure 4.26, M-MOPSO have good solutions for CF9 which are close to the true Pareto front and are well distributed. For CF10, MOEA/D has the best solutions as seen in Figure 4.27.

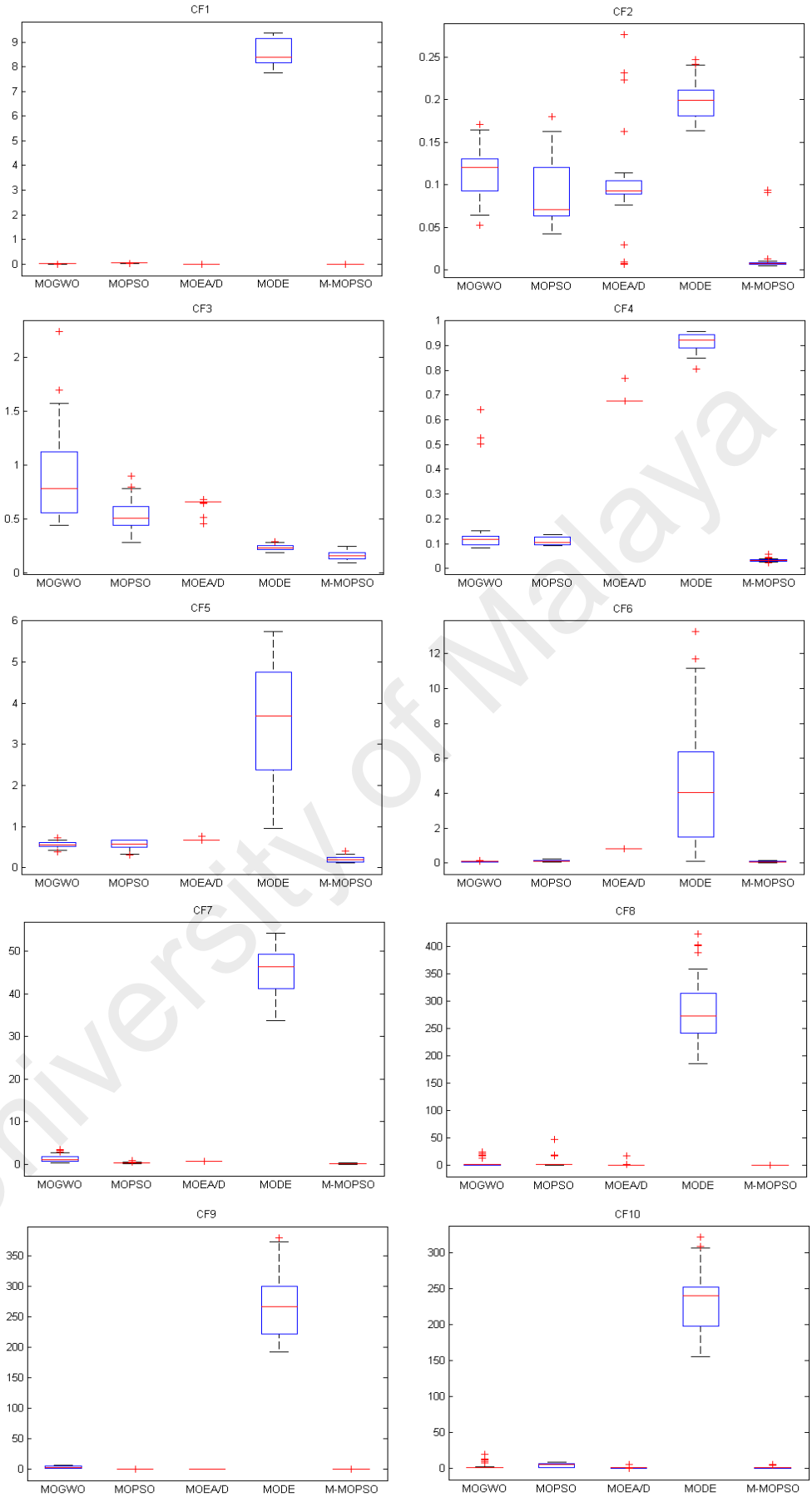

Figure 4.17: Boxplot for the statistical results for IGD on CF1 to CF10.

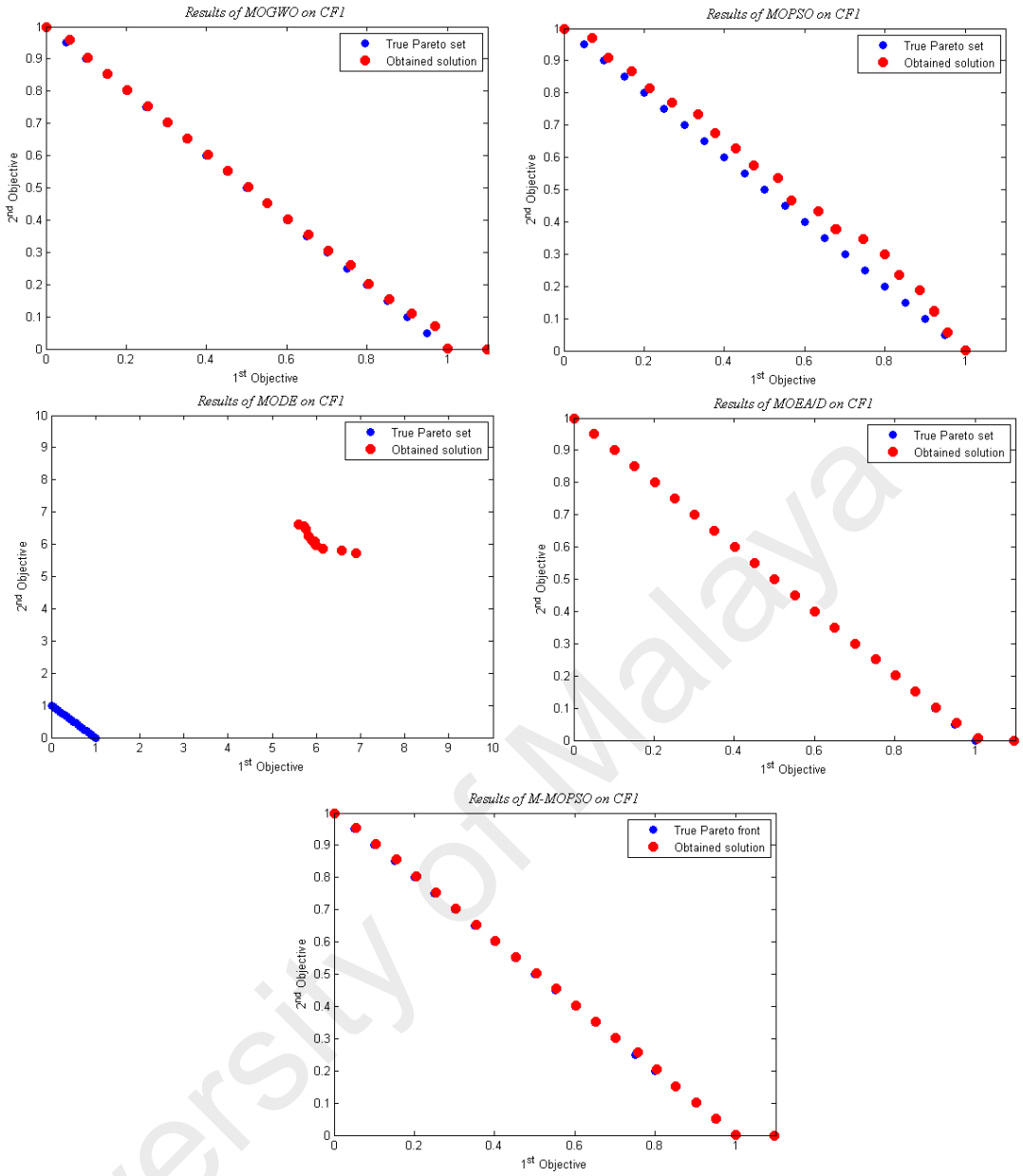

Figure 4.18: Obtained Pareto solutions for CF1.

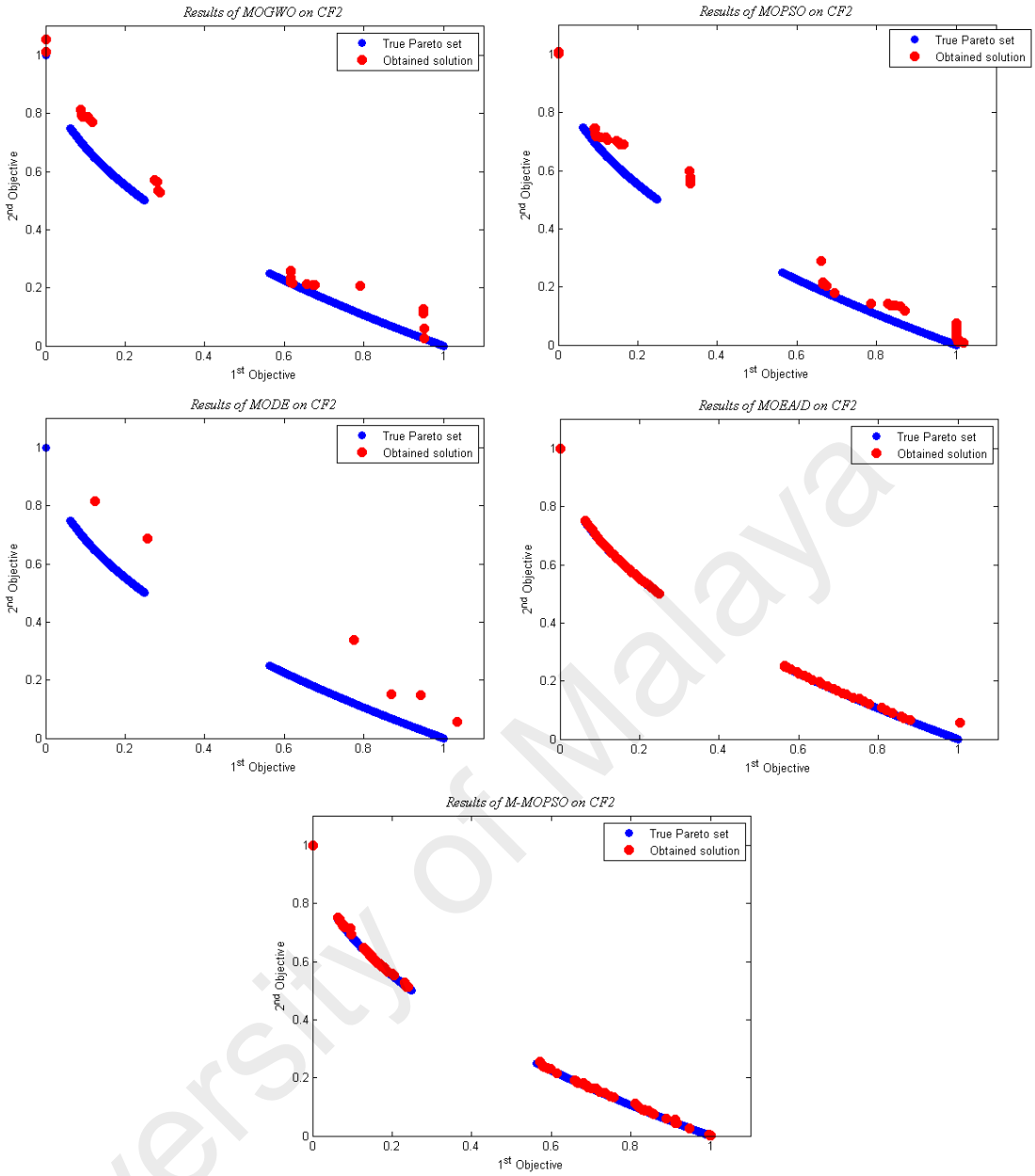

Figure 4.19: Obtained Pareto solutions for CF2.

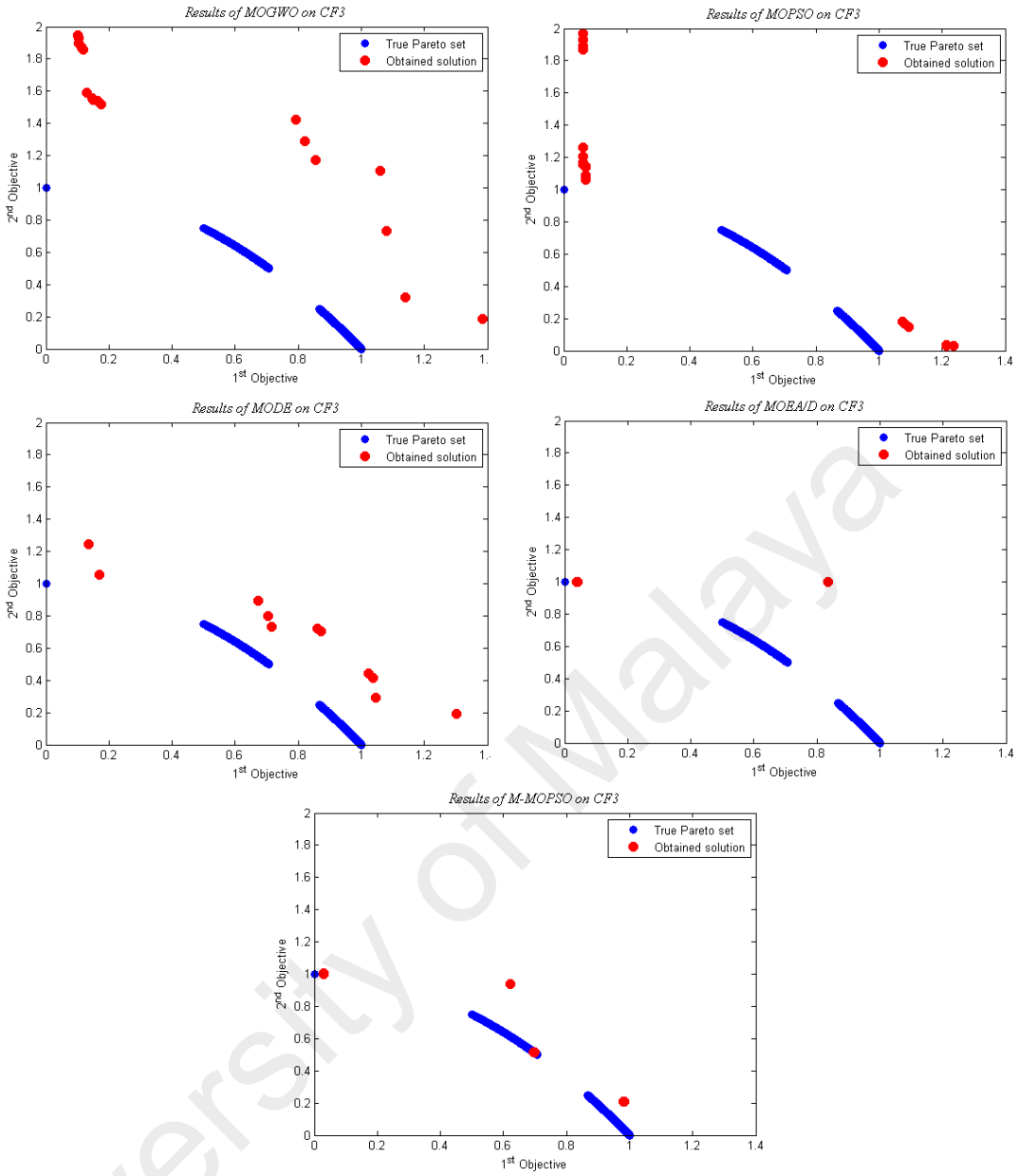

Figure 4.20: Obtained Pareto solutions for CF3.

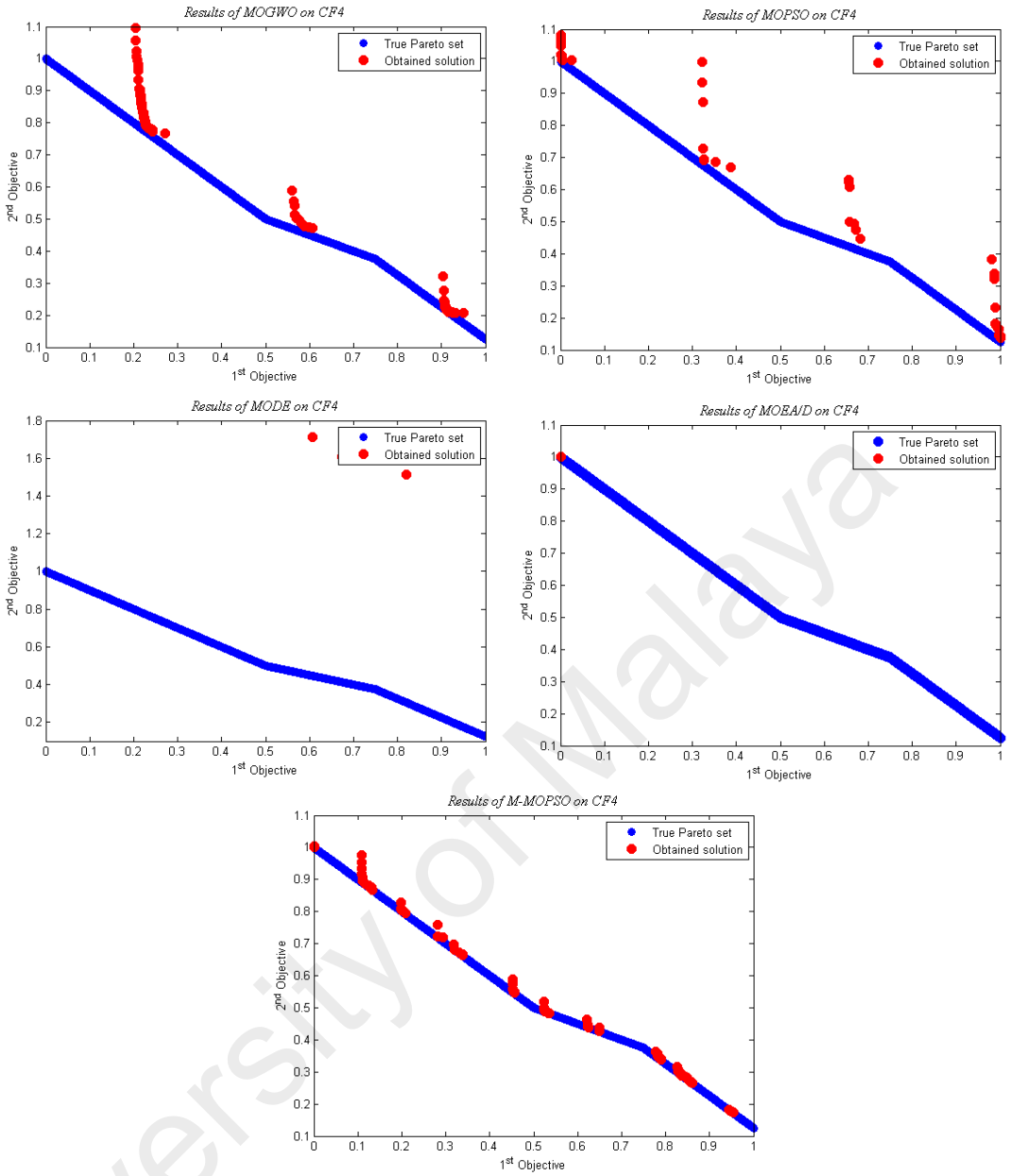

Figure 4.21: Obtained Pareto solutions for CF4.

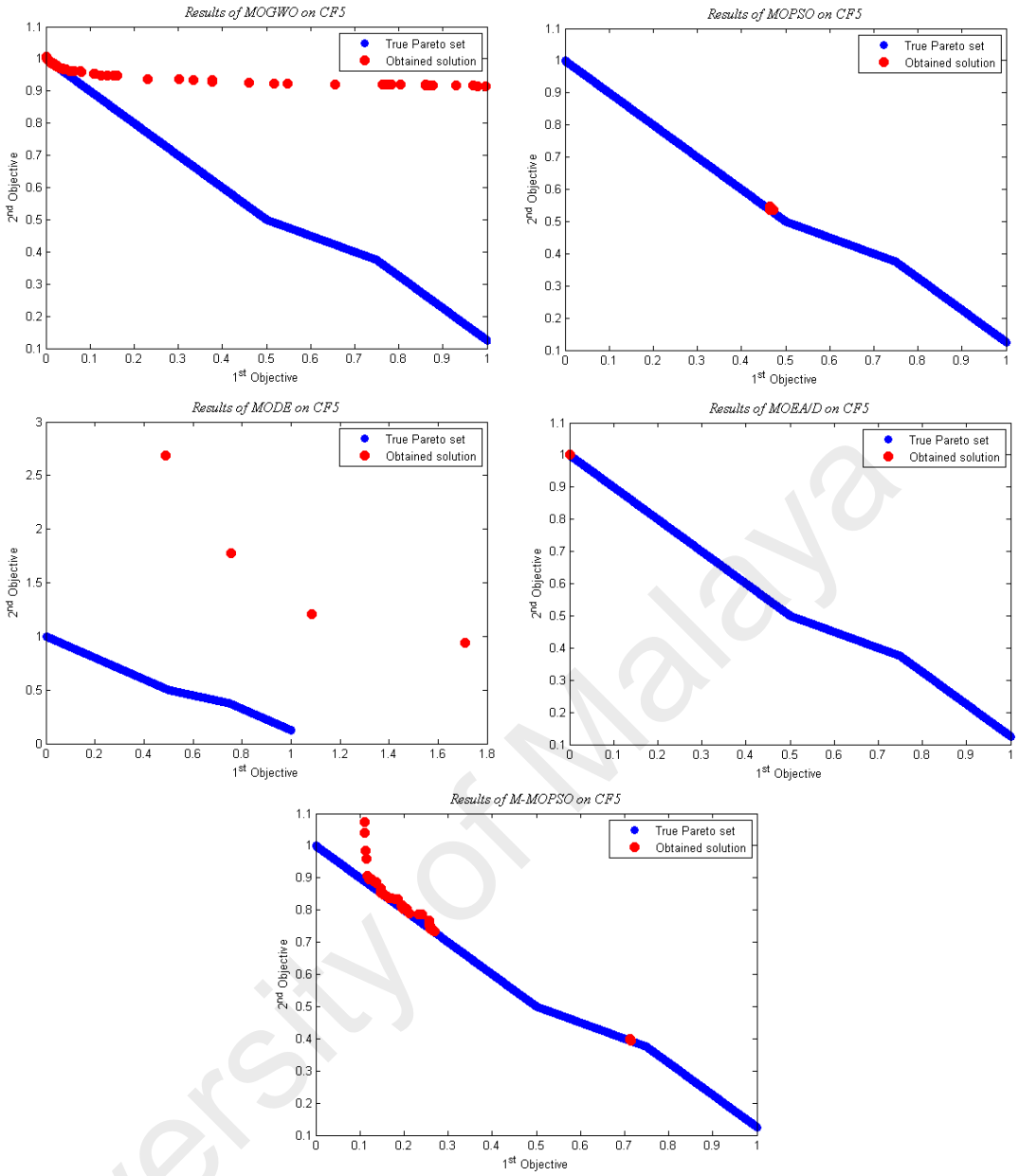

Figure 4.22: Obtained Pareto solutions for CF5.

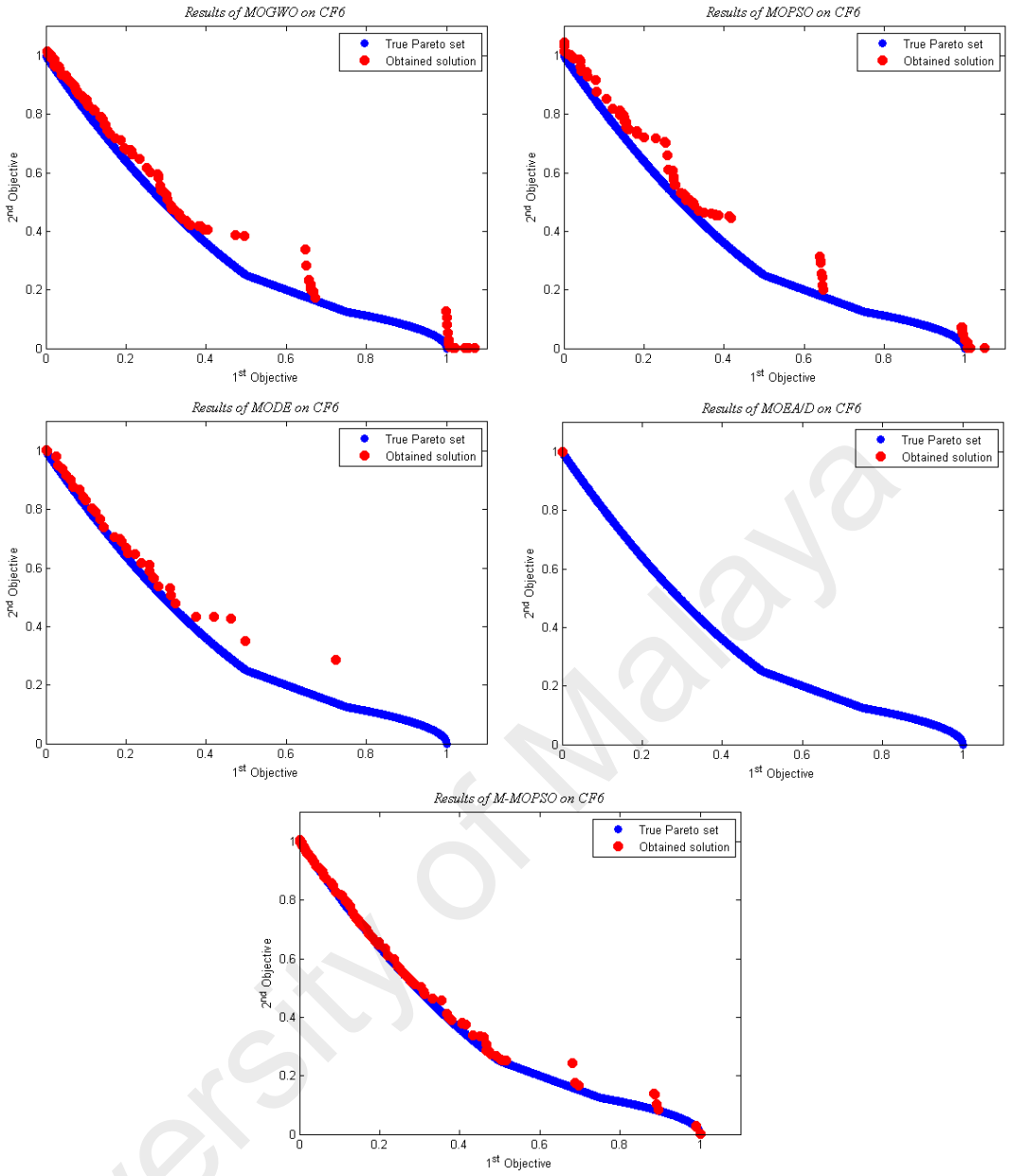

Figure 4.23: Obtained Pareto solutions for CF6.

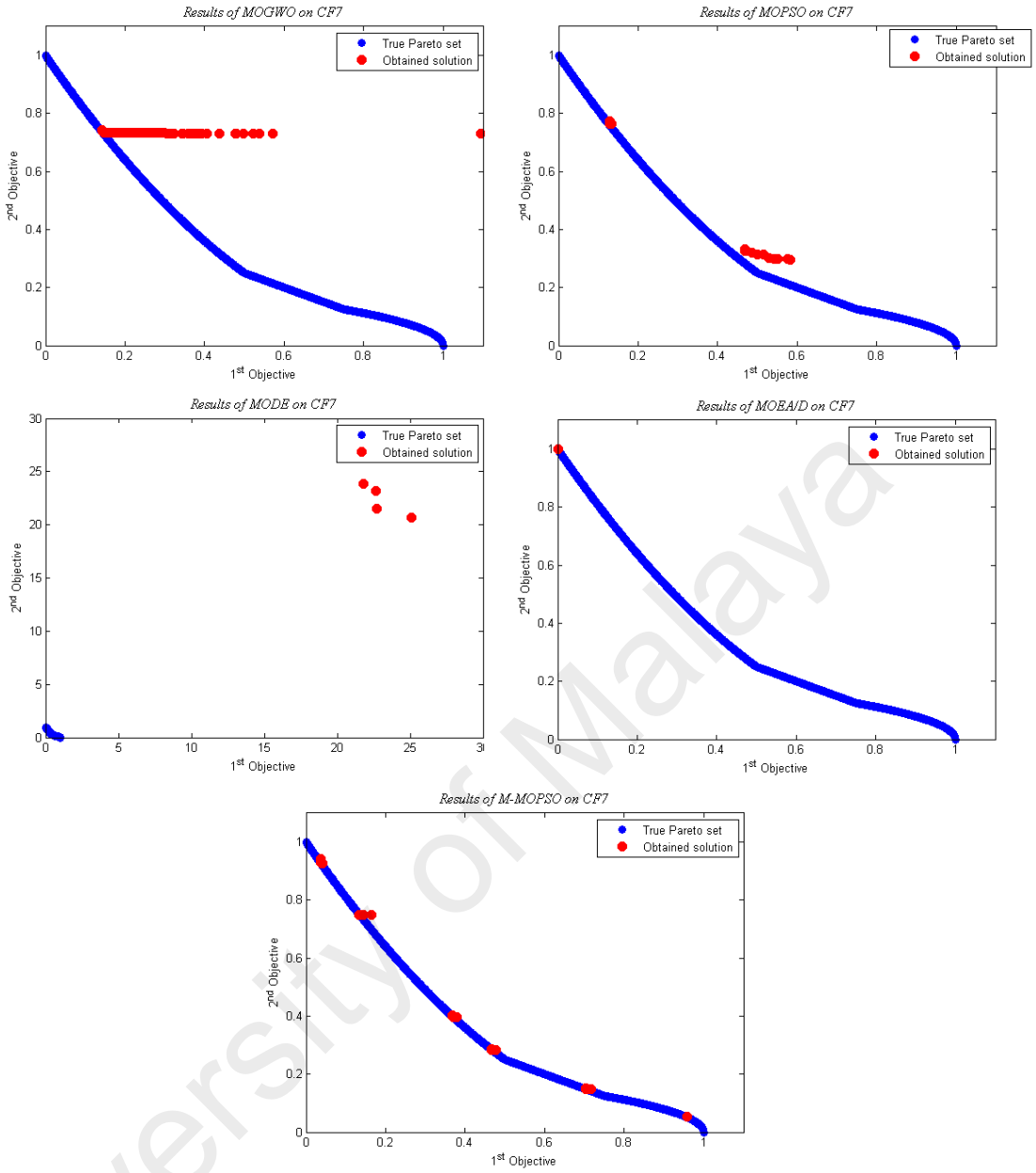

Figure 4.24: Obtained Pareto solutions for CF7.

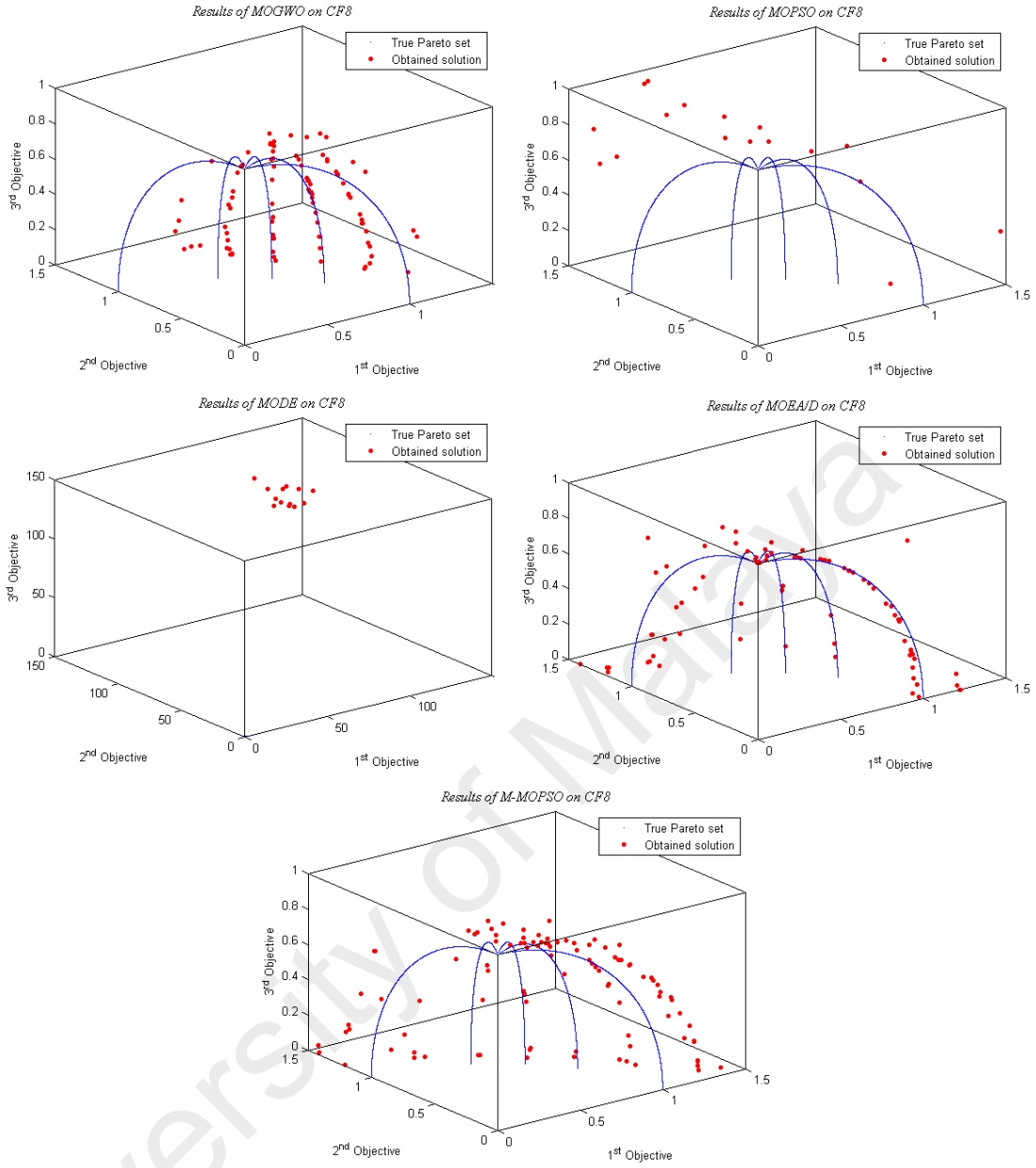

Figure 4.25: Obtained Pareto solutions for CF8.

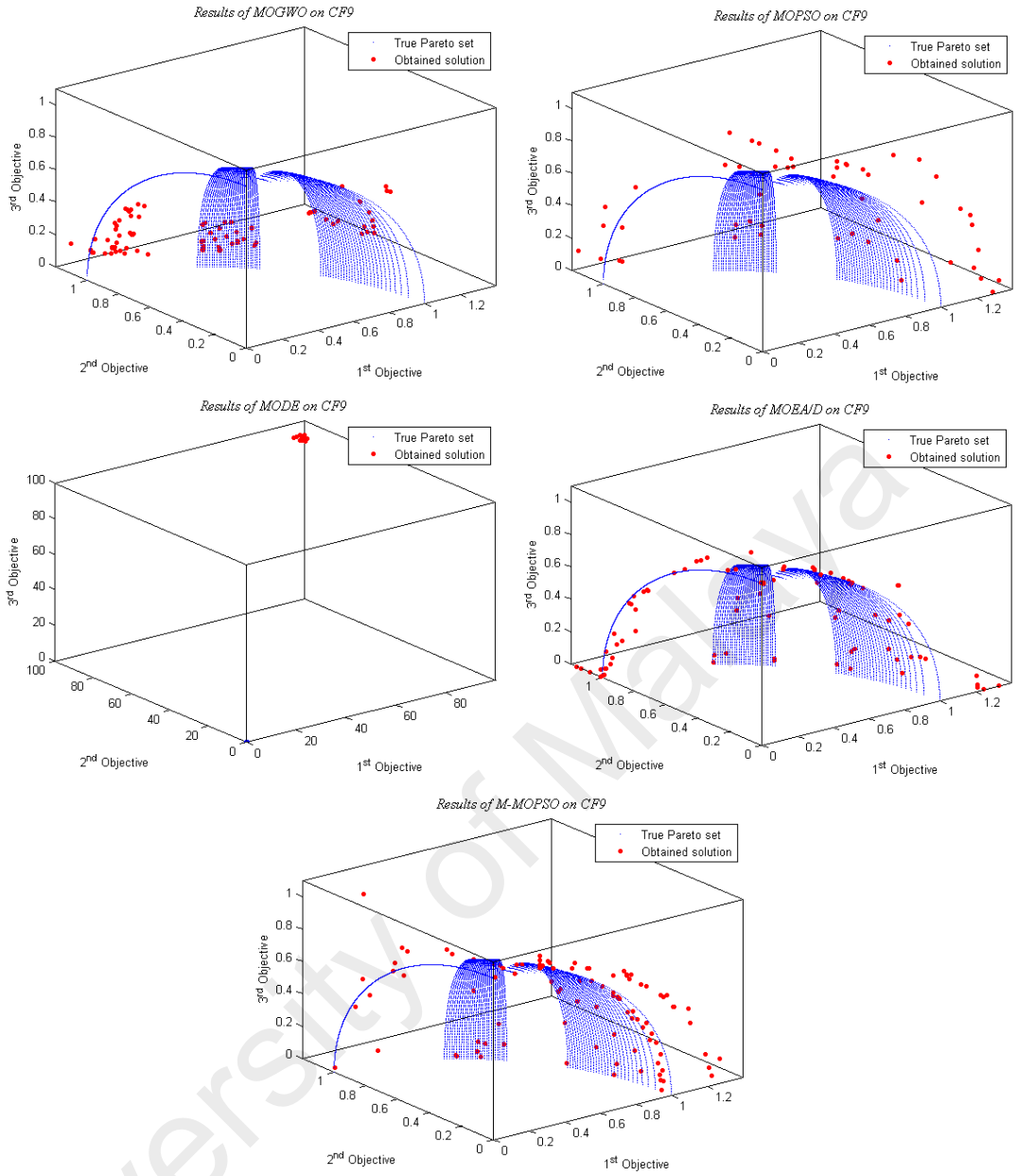

Figure 4.26: Obtained Pareto solutions for CF9.

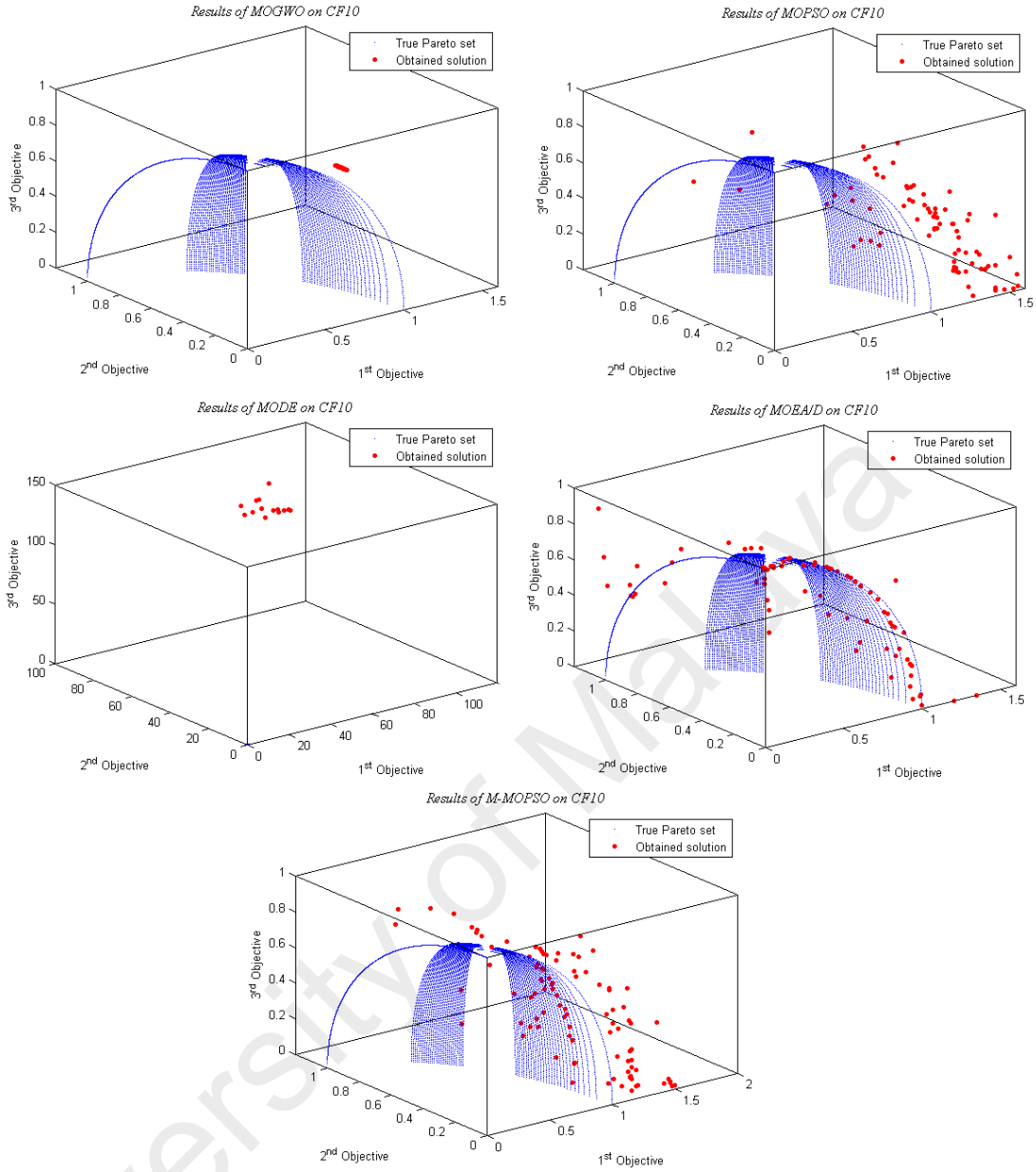

Figure 4.27: Obtained Pareto solutions for CF10.

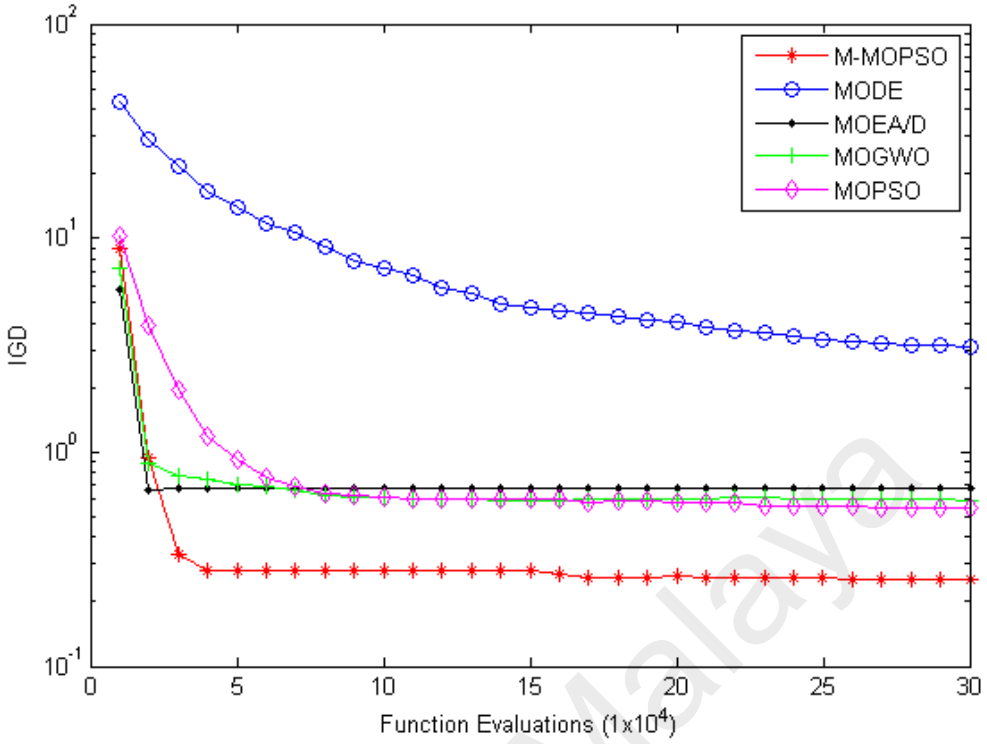

Figure 4.28: Convergence graph for CF5.

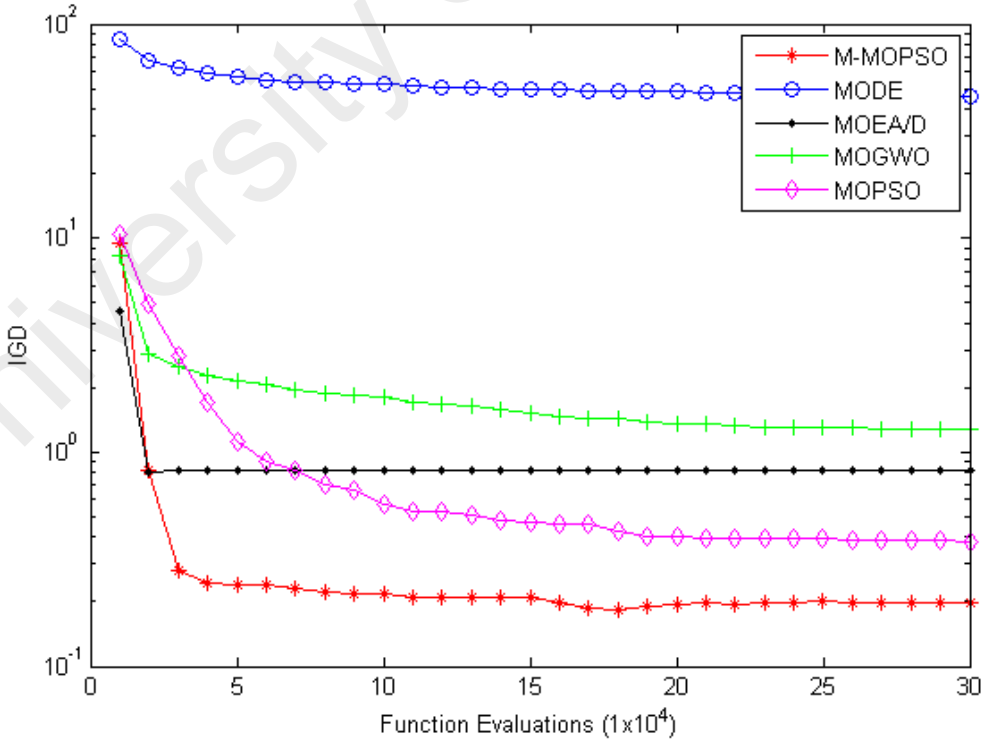

Figure 4.29: Convergence graph for CF7.

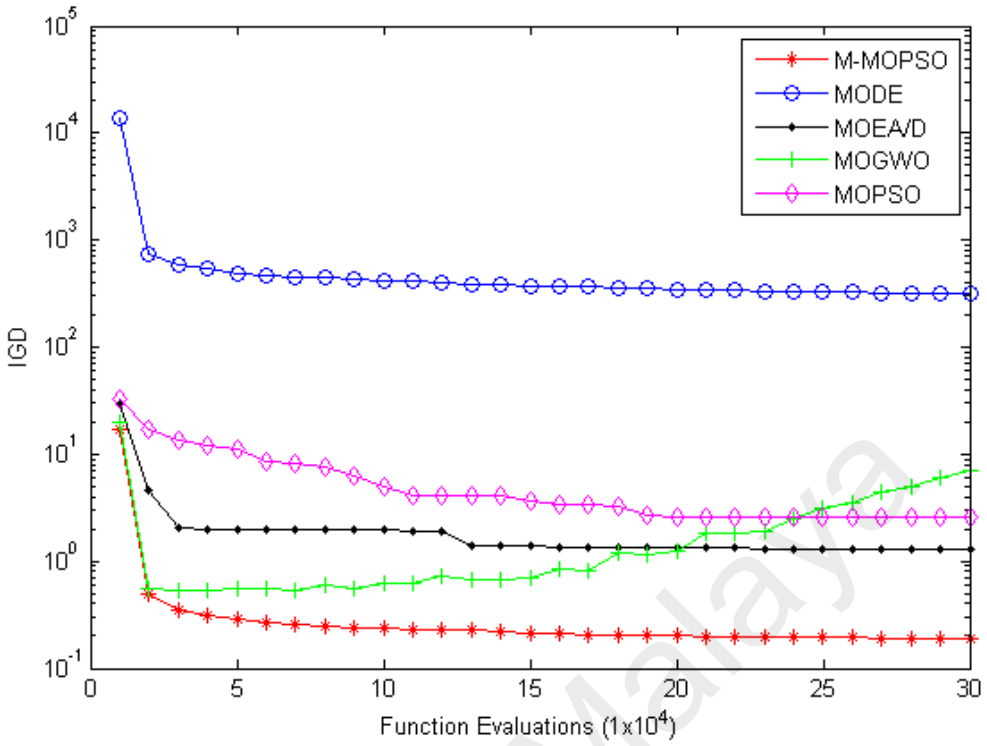

Figure 4.30: Convergence graph for CF8.

Figure 4.28 shows the convergence graph for CF5. The graph shows the average IGD of 30 runs recorded at 30 intervals of function evaluations (FE) obtained by each algorithm. MOEA/D converges slightly faster by obtaining lower average IGD compared to M-MOPSO and other algorithms at 20,000 FE. However, MOEA/D become saturated quicker than others and trapped at local optima after 30,000 FE. Meanwhile, M-MOPSO converges better than all other algorithms as early as 30,000 FE and stays as the algorithm with the best convergence until termination.

The convergence graph for CF7 is shown in Figure 4.29. MOEA/D has the best IGD at 10,000 FE but quickly become saturated at 20,000 FE. M-MOPSO however matches the IGD of MOEA/D at 20,000 FE and has the best IGD among all the algorithms at 30,000 FE. It continues to have the best IGD until termination.

In Figure 4.30, the convergence graph for CF8 is shown. At 10,000 and 20,000 FE both M-MOPSO and MOGWO have almost the same IGD. However, at 30,000 FE the IGD of MOGWO starts to become worse while the IGD of M-MOPSO continues to improve. This scenario occurs until termination.

4.2.2 Fed-batch bioprocess problems

For the application problems in case study VII and VIII, 200 search agents are utilized. The algorithms are run for a maximum of 200,000 and 400,000 function evaluations for case study VII and VIII respectively. The Pareto-optimal front between the yield and the productivity for case study VII is shown in Figure 4.31 while Figure 4.32 shows the Pareto-optimal front for Case study VIII.

In this experiment, we use two performance metrics. The first metric is Spacing (SP) (Schott, 1995), which is used to quantify the coverage by measuring the distance between consecutive solutions obtained in the Pareto front. The mathematical formulation for SP is as follow:

$$SP = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(\bar{d} - d_i)^2} \tag{72}$$

where $\bar{d}$ is the average of all d_i, n is the number of Pareto optimal solutions obtained, and $d_i = \min_j(\left|f_1^i(\vec{x}) - f_1^j(\vec{x})\right| + \left|f_2^i(\vec{x}) - f_2^j(\vec{x})\right|)$ for all $i, j = 1, 2, 3, \ldots, n$.

The second metric is Maximum Spread (MS) (Zitzler, 1999), which measures the extent of spread in the obtained solutions in the Pareto front.

$$MS = \sqrt{\sum_{i=1}^{o}\max(d(a_i, b_i))} \tag{73}$$

where d is a function to calculate the Euclidean distance, a_i is the maximum value in the ith objective, b_i is the minimum value in the ith objective and o is the number of objectives.

Figure 4.31 showed that the original MOPSO is not effective in solving the constraint problem and could find only one feasible solution. M-MOPSO however, obtained a very good Pareto front and rivaled the performance of MOEA/D. Qualitatively speaking, MOGWO is the third best algorithm followed by MODE.

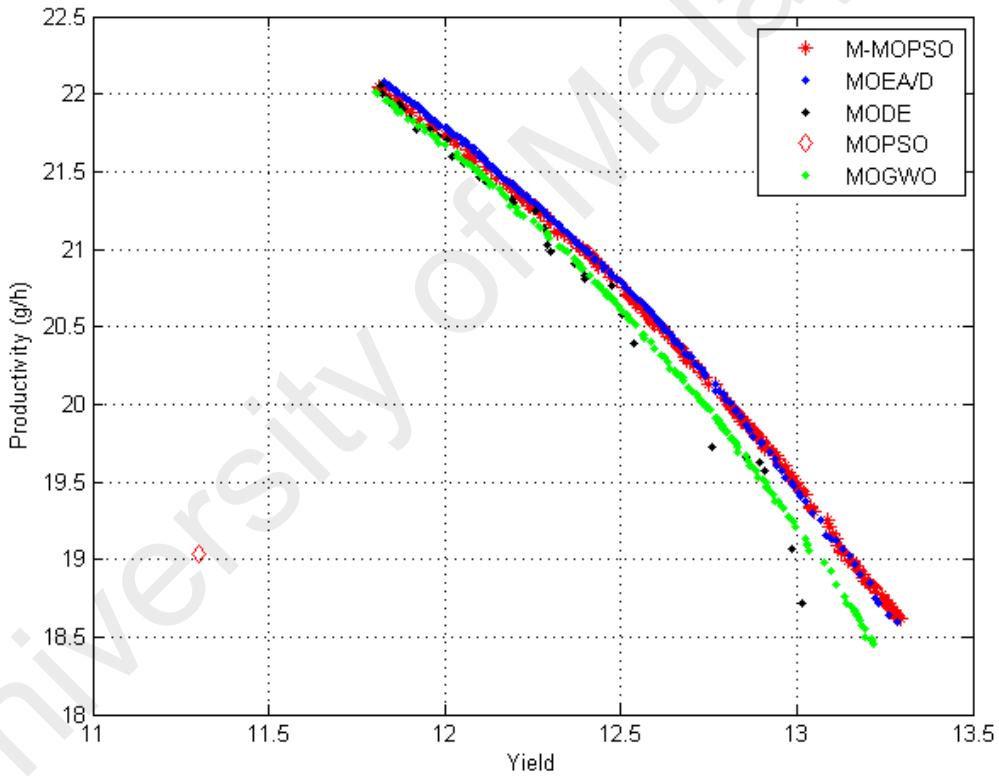

Figure 4.31: Productivity-yield pareto-optimal front for case study VII.

In Figure 4.32, M-MOPSO obtained a good Pareto front for case study VIII. All algorithms have almost equal convergence for this problem. The difference that set them apart is the spread of the Pareto front. M-MOPSO have the best coverage of the

whole objective space in which the distribution of the solutions found by M-MOPSO extends towards a greater area compared to other algorithms. In term of coverage, the second best algorithm is MOPSO, followed by MOEA/D, MOGWO and MODE.

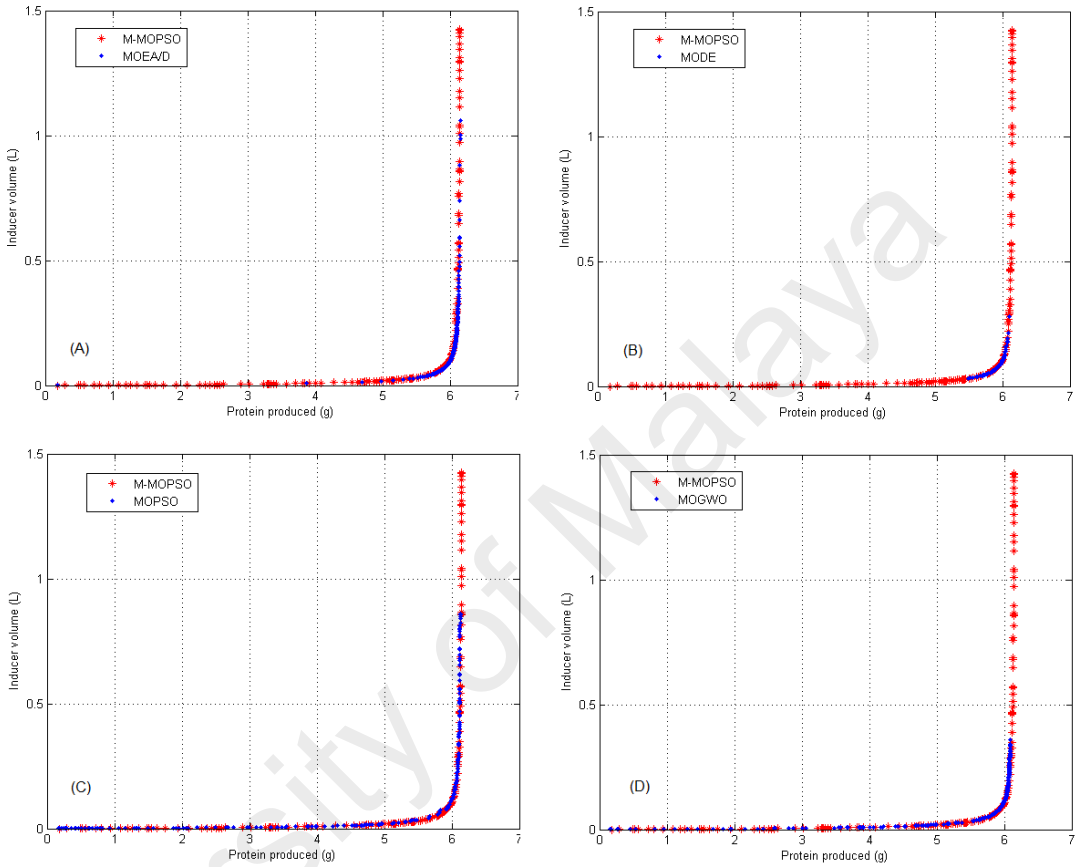

Figure 4.32: Pareto-optimal front of M-MOPSO against others for case study VIII: (A) M-MOPSO against MOEA/D, (B) M-MOPSO against MODE, (C) M-MOPSO against MOPSO, (D) M-MOPSO against MOGWO.

Table 4.15: SP and MS results for chemical problems.

	Case 1		Case 2	
	SP	MS	SP	MS
M-MOPSO	0.01298	2.21730	0.03412	2.71857
MODE	0.09649	2.13055	0.01060	0.92574
MOEA/D	0.01578	2.22270	0.06437	2.65010
MOGWO	0.01379	2.22884	0.03134	2.50830
MOPSO	0	0	0.02751	2.60926

Table 4.14 shows the results of SP and MS for all algorithms in case VII and case VIII. In case VII, aside of MOPSO which only found one unique solution, M-MOPSO obtained the lowest SP compared to others. This shows that M-MOPSO has the most uniform distribution of its Pareto front. For MS, MOGWO obtained the highest value which means that it has the largest spread compared to others, though marginally. In case VIII, MODE obtained the lowest SP compared to others, unfortunately it also obtained a significantly lower MS. M-MOPSO however obtained the better balance between SP and MS by not only obtaining the largest MS but also comparatively low SP.

In overall, M-MOPSO was able to solve multi-objective bioprocess application problems effectively. This can be seen by the results obtained in both case study VII and VIII, where M-MOPSO edged over most algorithms tested in this study.

CHAPTER 5: CONCLUSION

This study proposed the application of Backtracking Search Algorithm (BSA) on fed-batch fermentation processes. In fed-batch fermentation, nutrient feeding during fermentation process enhances higher product yield. Optimized nutrient feeding stimulates biomass growth and this increases product concentrations while curtailing biomass inhibition due to product and/or nutrient accumulation. Hence, the substrate feed rate plays crucial role in fed-batch process optimization.

We also demonstrated the application of metaheuristics on fed-batch aerated lagoon wastewater treatment. This process involves the intermittent feeding of concentrated wastewater into an aerated lagoon. The amount of wastewater to be fed into the lagoon at each day is treated as the variables to be optimized by the metaheuristic. Another contribution of this study is the formulation of fed-batch model for methane production from sewage sludge fermentation. Apart from the proper and cost-effective disposal of sewage sludge from the Waste Water Treatment Plant (WWTP), anaerobic digestion of sewage sludge plays a key role in the production of biogas namely methane. Usually batch mode fermentation is used to generate biogas. In the current work, biogas production was shown to be further enhanced by using fed-batch operation as feed rate becomes key optimization variable for metaheuristics.

Based on past literature, Differential Evolution (DE) is considered as a more appropriate solution for bio-process applications. Since DE is known to be efficient in solving fermentation problems, BSA as a recent DE-based metaheuristic is deemed to be superior to the former. Four recent metaheuristics that included DE were applied on three bioprocess engineering problems widely used in literature alongside with the problems mentioned above and the results were compared with BSA. From the results, BSA showed consistency of obtaining highest fitness value in comparison to other four

metaheuristics for all the cases at convergence point. Therefore, BSA is suggested as the first choice metaheuristic to use when solving bioprocess engineering problems.

The performances of metaheuristcs in solving multi-objectives fed-batch fermentation problems were also evaluated. In multi-objectives problems, the objectives to be optimized can extend beyond the production rate and include substrate utilization, environmental impact and economic benefits. Therefore, we presented a modification of multi-objective particle swarm optimization (MOPSO) to tackle multi-objective optimization problems. Our algorithm, called modified multi-objective particle swarm optimization (M-MOPSO) employs a new dynamic search boundary mechanism to properly balance exploration and exploitation during the search procedure. The archiving procedure used in MOPSO was also modified to maintain diversity in the Pareto front while reducing the computational cost of the archive controller.

Our experiment used the CEC2009 multi-objective benchmark problems to verify the performance of our algorithm. Comparisons were made with four other recent algorithms, namely multi-objective grey wolf optimizer (MOGWO), multi-objective particle swarm optimizer (MOPSO), and multi-objective evolutionary algorithm based on decompositions (MOEA/D) and multi-objective differential evolution (MODE). Based on the results, M-MOPSO emerged as the better algorithm by obtaining better Inverted Generational Distance (IGD) average for eight out of the ten test instances.

We also ran some simulations of multi-objective bioprocess application problems to investigate the capability of M-MOPSO in solving real-world engineering problems. M-MOPSO showed promising results by rivaling other state-of-the art techniques used in this study. It also displayed better capability in handling constraint compared to MOPSO. For the unconstraint problem, M-MOPSO obtained superior coverage of the Pareto front while maintaining good convergence compared to other algorithms.

In overall, M-MOPSO was able to solve multi-objective problems with good convergence and it is interesting to extend the capability of this algorithm in solving many-objective problems and other more complex application problems in the future.

REFERENCES

Appels, L., Baeyens, J., Degrève, J., & Dewil, R. (2008). Principles and potential of the anaerobic digestion of waste-activated sludge. *Progress in Energy and Combustion Science, 34*(6), 755-781.

Askarzadeh, A., & Coelho, L. d. S. (2014). A backtracking search algorithm combined with Burger's chaotic map for parameter estimation of PEMFC electrochemical model. *International Journal of Hydrogen Energy, 39*(21), 11165-11174.

Banga, J. R., Balsa-Canto, E., Moles, C. G., & Alonso, A. A. (2005). Dynamic optimization of bioprocesses: Efficient and robust numerical strategies. *Journal of Biotechnology, 117*(4), 407-419.

Banga, J. R., Moles, C. G., & Alonso, A. A. (2004). Global Optimization of Bioprocesses using Stochastic and Hybrid Methods. In C. A. Floudas & P. Pardalos (Eds.), *Frontiers in Global Optimization* (pp. 45-70). Boston, MA: Springer US.

Basturk, B., Karaboga, D. (2006). *An artificial bee colony (ABC) algorithm for numeric function optimization.* Paper presented at the Proceedings of the IEEE Swarm Intelligence Symposium 2006, Indianapolis, Indiana, USA.

Bhaskar, V., Gupta, S. K., & Ray, A. K. (2000). Applications of multiobjective optimization in chemical engineering. *Reviews in chemical engineering, 16*(1), 1-54.

Burke, E. K., & Kendall, G. (2014). *Search Methodologies - Introductory Tutorials in Optimization and Decision Support Techniques* (2 ed.). New York: Springer US.

Cawthon, G. D., & Knaebel, K. S. (1989). Optimization of semibatch polymerization reactions. *Computers & Chemical Engineering, 13*(1), 63-72.

Chandra, R., Takeuchi, H., & Hasegawa, T. (2012). Methane production from lignocellulosic agricultural crop wastes: A review in context to second generation of biofuel production. *Renewable and Sustainable Energy Reviews, 16*(3), 1462-1476.

Chen, C.-T., & Hwang, C. (1990). Optimal control computation for differential-algebraic process systems with general constraints. *Chemical Engineering Communications, 97*(1), 9-26.

Chiou, J.-P., & Wang, F.-S. (1999). Hybrid method of evolutionary algorithms for static and dynamic optimization problems with application to a fed-batch fermentation process. *Computers & Chemical Engineering, 23*(9), 1277-1291.

Civicioglu, P. (2013). Backtracking search optimization algorithm for numerical optimization problems. *Applied Mathematics and Computation, 219*(15), 8121-8144.

Coello, C. A. C., Pulido, G. T., & Lechuga, M. S. (2004). Handling multiple objectives with particle swarm optimization. *Ieee Transactions on Evolutionary Computation, 8*(3), 256-279.

Da Ros, S., Colusso, G., Weschenfelder, T. A., de Marsillac Terra, L., de Castilhos, F., Corazza, M. L., & Schwaab, M. (2013). A comparison among stochastic optimization algorithms for parameter estimation of biochemical kinetic models. *Applied Soft Computing, 13*(5), 2205-2214.

Das, S., Mandal, D., Kar, R., & Ghoshal, S. P. (2014). *Interference suppression of linear antenna arrays with combined Backtracking Search Algorithm and Differential Evolution.* Paper presented at the International Conference on Communications and Signal Processing (ICCSP).

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *Ieee Transactions on Evolutionary Computation, 6*(2), 182-197.

del Rio-Chanona, E. A., Zhang, D., & Vassiliadis, V. S. (2016). Model-based real-time optimisation of a fed-batch cyanobacterial hydrogen production process using economic model predictive control strategy. *Chemical Engineering Science, 142*, 289-298.

El-Fergany, A. (2015). Optimal allocation of multi-type distributed generators using backtracking search optimization algorithm. *International Journal of Electrical Power & Energy Systems, 64*, 1197-1205.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*: Addison-Wesley Longman Publishing Co., Inc.

Goulden, C. H. (1956). *Methods of statistical analysis* (2nd ed.): John Wiley & Sons Ltd.

Guney, K., Durmus, A., & Basbug, S. (2014). Backtracking search optimization algorithm for synthesis of concentric circular antenna arrays. *International Journal of Antennas and Propagation, 2014*.

Hansen, N., & Ostermeier, A. (1996). *Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation.* Paper presented at the Proceedings of IEEE International Conference on Evolutionary Computation, 1996

Holm, S. (1979). A simple sequentially rejective multiple test procedure. *Scandinavian journal of statistics*, 65-70.

J. Baeyens, L. Hosten, & E. Van Vaerenbergh. (1997). *Afvalwaterzuivering (Wastewater treatment)* (2nd ed.). The Netherlands: Kluwer Academic Publishers.

Jayaraman, V. K., Kulkarni, B. D., Gupta, K., Rajesh, J., & Kusumaker, H. S. (2001). Dynamic Optimization of Fed-Batch Bioreactors Using the Ant Algorithm. *Biotechnology Progress, 17*(1), 81-88.

Juma, C., & Konde, V. (2001). *The new bioeconomy: industrial and environmental biotechnology in developing countries.* Paper presented at the United Nations Conference on Trade and Development.

Kennedy, J., & Eberhart, R. (1995). *Particle swarm optimization*. Paper presented at the IEEE Int. Conf. on Neural Networks, Piscataway, NJ.

Kookos, I. K. (2004). Optimization of Batch and Fed-Batch Bioreactors using Simulated Annealing. *Biotechnology Progress, 20*(4), 1285-1288.

Lee, J., & Ramirez, W. F. (1994). Optimal fed-batch control of induced foreign protein production by recombinant bacteria. *AIChE Journal, 40*(5), 899-907.

Marler, R. T., & Arora , J. S. (2004). Survey of multi-objective optimization methods for engineering. *Structural and Multidisciplinary Optimization, 26*(6), 369-395.

Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Grey wolf optimizer. *Advances in Engineering Software, 69*, 46-61.

Mirjalili, S., Saremi, S., Mirjalili, S. M., & Coelho, L. d. S. (2016). Multi-objective grey wolf optimizer: A novel algorithm for multi-criterion optimization. *Expert Systems with Applications, 47*, 106-119.

Montalvo, S., Guerrero, L., Rivera, E., Borja, R., Chica, A., & Martín, A. (2010). Kinetic evaluation and performance of pilot-scale fed-batch aerated lagoons treating winery wastewaters. *Bioresource Technology, 101*(10), 3452-3456.

Ohno, H., Nakanishi, E., & Takamatsu, T. (1976). Optimal control of a semibatch fermentation. *Biotechnology and Bioengineering, 18*(6), 847-864.

Pelillo, M., Rincón, B., Raposo, F., Martín, A., & Borja, R. (2006). Mathematical modelling of the aerobic degradation of two-phase olive mill effluents in a batch reactor. *Biochemical Engineering Journal, 30*(3), 308-315.

Rocha, I. (2003). *Model-based strategies for computer-aided operation of recombinant E. coli fermentation.* (Ph.D. thesis), Universidade do Minho. Retrieved from <http://hdl.handle.net/1822/1269>

Rocha, M., Mendes, R., Rocha, O., Rocha, I., & Ferreira, E. C. (2014). Optimization of fed-batch fermentation processes with bio-inspired algorithms. *Expert Systems with Applications, 41*(5), 2186-2195.

Roubos, J. A., van Straten, G., & van Boxtel, A. J. B. (1999). An evolutionary strategy for fed-batch bioreactor optimization; concepts and performance. *Journal of Biotechnology, 67*(2–3), 173-187.

Sarkar, D., & Modak, J. M. (2004). Optimization of fed-batch bioreactors using genetic algorithm: multiple control variables. *Computers & Chemical Engineering, 28*(5), 789-798.

Sarkar, D., & Modak, J. M. (2005). Pareto-optimal solutions for multi-objective optimization of fed-batch bioreactors using nondominated sorting genetic algorithm. *Chemical Engineering Science, 60*(2), 481-492.

Schott, J. R. (1995). *Fault Tolerant Design Using Single and Multicriteria Genetic Algorithm Optimization: DTIC Document.* Massachusetts Institute of Technology. Department of Aeronautics and Astronautics.

Sierra, M. R., & Coello Coello, C. A. (2005). Improving PSO-Based Multi-objective Optimization Using Crowding, Mutation and ∈-Dominance. *Proceedings of the Evolutionary Multi-Criterion Optimization: Third International Conference.* (pp. 505-519). Berlin, Heidelberg: Springer Berlin Heidelberg.

Silva, C. M., & Biscaia Jr, E. C. (2003). Genetic algorithm development for multi-objective optimization of batch free-radical polymerization reactors. *Computers & Chemical Engineering, 27*(8–9), 1329-1344.

Song, X., Zhang, X., Zhao, S., & Li, L. (2015). Backtracking search algorithm for effective and efficient surface wave analysis. *Journal of Applied Geophysics, 114*, 19-31.

Sörensen, K. (2015). Metaheuristics—the metaphor exposed. *International Transactions in Operational Research, 22*(1), 3-18.

Sosnowski, P., Klepacz-Smolka, A., Kaczorek, K., & Ledakowicz, S. (2008). Kinetic investigations of methane co-fermentation of sewage sludge and organic fraction of municipal solid wastes. *Bioresource Technology, 99*(13), 5731-5737.

Srinivas, N., & Deb, K. (1994). Muiltiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. *Evolutionary Computation, 2*(3), 221-248.

Storn, R., & Price, K. (1997). Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces. *Journal of Global Optimization, 11*(4), 341-359.

Tholudur, A., & Ramirez, W. F. (1997). Obtaining smoother singular arc policies using a modified iterative dynamic programming algorithm. *International Journal of Control, 68*(5), 1115-1128.

Tsoukas, A., Tirrell, M., & Stephanopoulos, G. (1982). Multiobjective dynamic optimization of semibatch copolymerization reactors. *Chemical Engineering Science, 37*(12), 1785-1795.

Uymaz, S. A., Tezel, G., & Yel, E. (2015). Artificial algae algorithm (AAA) for nonlinear global optimization. *Applied Soft Computing, 31*, 153-171.

Wang, F.-S., & Cheng, W.-M. (1999). Simultaneous Optimization of Feeding Rate and Operation Parameters for Fed-Batch Fermentation Processes. *Biotechnology Progress, 15*(5), 949-952.

Yang, X. S. (2010). *Nature-inspired metaheuristic algorithms*. Beckington, UK: Luniver press.

Yang, X. S., & Suash, D. (2009). *Cuckoo Search via Levy flights.* Paper presented at the World Congress on Nature & Biologically Inspired Computing, 2009. NaBIC 2009.

Zhang, Q., & Li, H. (2007). MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition. *Ieee Transactions on Evolutionary Computation, 11*(6), 712-731.

Zhang, Q., Zhou, A., Zhao, S., Suganthan, P. N., Liu, W., & Tiwari, S. (2008). *Multiobjective optimization test instances for the CEC 2009 special session and competition.* Paper presented at the University of Essex, Colchester, UK and Nanyang technological University, Singapore. Special session on performance assessment of multi-objective optimization algorithms, technical report.

Zitzler, E. (1999). Evolutionary algorithms for multiobjective optimization: Methods and applications.

LIST OF PUBLICATIONS AND PAPERS PRESENTED

bin Mohd Zain, M. Z., Kanesan, J., Kendall, G., & Chuah, J. H. (2018). Optimization of fed-batch fermentation processes using the Backtracking Search Algorithm, *Expert Systems with Applications*, 91, 286-297.

Mohamad Zihin, M. Z., Kanesan, J., Chuah, J. H., Dhanapal, S., Kendall, G. (2018). Modified Multi-Objective Particle Swarm Optimization for constrained optimization problems, *Applied Soft Computing* (Submitted).