OPTIMIZATION OF EXPANDED-SOLVENT STEAM ASSISTED GRAVITY DRAINAGE USING DIFFERENTIAL EVOLUTION

BY

Tamer M. Moussa

A Thesis Presented to the
DEANSHIP OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

In

PETROLEUM ENGINEERING

May 2015

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN- 31261, SAUDI ARABIA

DEANSHIP OF GRADUATE STUDIES

This thesis, written by **Tamer M. Moussa** under the direction his thesis advisor and approved by his thesis committee, has been presented and accepted by the Dean of Graduate Studies, in partial fulfillment of the requirements for the degree of **MASTER OF SCIENCE IN PETROLEUM ENGINEERING.**

[illegible]

Dr. Abeeb Awotunde
(Advisor)

[illegible]

Dr. Abdullah Saad Sultan
Department Chairman

[illegible]

Dr. Abdullah Saad Sultan
(Member)

[illegible]

Dr. Salam A. Zummo
Dean of Graduate Studies

عمادة الدراسات العليا
DEANSHIP OF GRADUATE STUDIES
KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

[illegible]

Dr. Sidqi Ahmad Abu-Khamsin
(Member)

9/8/15
Date

Dedication

This thesis is dedicated to my wife, my sons and my parents for their love and encouragement.

ACKNOWLEDGMENTS

First of all, I want to thank my advisor Dr. Abeeb Awotunde for his support, contributions and encouragement. The simulation and modeling of oil reservoirs I learned from him guided me throughout my graduate studies.

I also would like to thank my committee members Dr. Abdullah Sultan and Dr. Sidqi Abu-Khamsin for their patience, continued guidance, and support went a long way in making this study and thesis possible. They have always been accessible and willing to help me with my research.

I thank the staff at the department of Petroleum Engineering for their cooperation. Also, thanks go to CMG STARS and CMOST groups who helped me through various problems during my study.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

EP	:	Evolutionary Programing
EA	:	Evolutionary Algorithm
DE	:	Differential Evolution
SaDE	:	Self-adaptive Differential Evolution
MSaDE	:	Modified Self-adaptive Differential Evolution
PSO	:	Particle Swarm Optimization
SAGD	:	Steam-Assisted Gravity Drainage
ES-SAGD	:	Expanding Solvent Steam-Assisted Gravity Drainage
VAPEX	:	Solvent Vapor Extraction
cSOR	:	Cumulative Steam Oil Ratio
RF	:	Recovery Factor
NPV	:	Net Present Value

ABSTRACT

Full Name	: Tamer Magdi Moussa
Thesis Title	: Optimization of Expanded-Solvent Steam Assisted Gravity Drainage Using Differential Evolution
Major Field	: Petroleum Engineering
Date of Degree	: June 2015

Differential Evolution (DE) optimization algorithm has shown good performance in many optimization problems. However, its control parameters greatly affect its performance and require many trials to determine the optimum values of control parameters for specific optimization problem. On the other hand, the Expanding Solvent Steam-Assisted Gravity Drainage (ES-SAGD) process is one of the most promising thermal techniques to recover heavy oil and extra heavy-oil reservoirs. However, because of high computation requirements, limited attention has been paid to integrate the ES-SAGD simulation with global optimization algorithms to handle more design elements. Without efficient and optimized recovery process design, the ultimate recovery from such unconventional resources will not be achieved, or it could be achieved with great cost and large environmental impact.

The objective of this work is introducing self-adaptive DE algorithm with a new adaptation technique to improve solution quality, speed convergence to optimum solution and reduce computational cost. As well as studying its performance on benchmark test functions before being applied to optimize the recovery performance of ES-SAGD process.

The proposed method is shown to enhance the convergence speed and balance the exploitation and global exploration capabilities of self-adaptive DE algorithms and it is superior over conventional DE algorithm. It has also be shown that cumulative Steam Oil Ratio (cSOR) is not the right performance indicator when optimizing ES-SAGD recovery process. Alternatively, Net Present Value (NPV) of the recovery process at the end of the project is better representative to project profitability and recovery capability.

ملخص الرسالة

الاسم الكامل	: تامر مجدي موسى
عنوان الرسالة	: تحسين أداء عملية ضخ بخار الماء والمذيبات العضوية لإنتاج الزيت عن طريق خوارزمية تطور التفاضلية
التخصص	: هندسة البترول
تاريخ الدرجة العلمية	: مايو 2015

لقد أظهرت خوارزمية تطور التفاضلية (Differential Evolution) تحسين أداء جيد في العديد من مشاكل التحسين العددية. ومع ذلك، عوامل التحكم في هذه الطريقة تؤثر بشكل كبير على أدائها وتتطلب العديد من التجارب لتحديد القيم المثلى لهذه العوامل لكل عملية تحسين عددية. من ناحية أخرى، عملية تحسن انتاج الزيت عن طريق ضخ بخار الماء والمذيبات (ES-SAGD) هي واحدة من التقنيات الحرارية الواعدة لاستخراج النفط الثقيل. ومع ذلك، بسبب متطلبات الحوسبة العالية، فقد تم إيلاء اهتمام محدود لدمج محاكاة عملية (ES-SAGD) مع خوارزميات التحسين العددية للتعامل مع أكثر من عنصر من عناصر التصميم. وبدون عملية متكاملة ومحسنة لاستخراج الزيت الثقيل، لن يتحقق الإنتاج المرجو من هذه الموارد غير التقليدية، او يمكن أن يتحقق ذلك ولكن بتكلفة كبيرة وأثر بيئي واسع.

ان الهدف من هذا العمل هو تحسين اداء خوارزمية تطور التفاضلية عن طريق إضافة خاصية التكيف الذاتي لتحسين نوعية الحل، وسرعة التقارب إلى الحل الأمثل وتقليل التكلفة الحسابية. وكذلك دراسة أدائها على مجموعة من مشاكل التحسين المعروفة لاختبار و قياس الأداء قبل تطبيقها لتحسين أداء عملية (ES-SAGD).

تظهر الطريقة المقترحة تعزيز سرعة التقارب وتحقيق التوازن بين الاستغلال والاستكشاف العام وانها متفوقة على خوارزمية تطور التفاضلية التقليدية. كما أن يتبين أن نسبة الزيت الى البخار المتراكم (cSOR) ليس مؤشر الأداء الصحيح عند لتحسين إنتاج الزيت في عملية (ES-SAGD) بدلا من ذلك، صافي القيمة الحالية (NPV) من العملية في نهاية المشروع هو الممثل الأفضل لربحية المشروع والقدرة على الإنتاج.

CHAPTER 1

INTRODUCTION

1.1 Introduction

The decline of conventional oil reserves and the ever-increasing energy demands make heavy oil exploration of primary interest to many oil companies. Due to the heavy oil recovery difficulties and low economic return, most of the heavy oil reserves remain undeveloped, despite the large volume of its original oil in place. Oil with API gravity less than 20^o is considered heavy oil and with less than 10^o is categorized as extra heavy-oil (IEA Conference, Calgary, 2002). Heavy-oil's high viscosity and density are the main challenges in heavy oil recovery, which sometimes make it immobile at initial reservoir conditions.

Recently, because of the development in directional drilling technologies, it has been possible to develop and produce heavy-oil reservoirs by very promising recovery methods such as Steam-Assisted Gravity Drainage (SAGD) and Solvent Vapor Extraction (VAPEX). These techniques have significantly improved oil recovery factor and reduce production costs due to the improved sweep efficiencies.

Thermal recovery techniques such as Steam- Assisted Gravity Drainage (SAGD) takes the advantage of the strong dependency of heavy oil viscosity on reservoir temperature. In other words, SAGD has been applied to reduce the viscosity of heavy oil by heating the reservoir. However, for reservoirs with high water content, thin pay zones or/and underlying aquifers, thermal recovery methods are usually not considered as an economical technique for heavy-oil recovery. In addition, despite the higher oil recovery of the SAGD method, there are several challenges associated with this technique including the requirement of large amount of fresh water, produced water treatment and handling, and high energy consumption to generate steam, resulting in high CO2 emissions. Therefore, another technique which uses the same well configuration as SAGD has shown up and might be considered as a good alternative to the SAGD. This technique is VAPEX. In this process hydrocarbon solvent is injected instead steam and the heavy oil viscosity is decreased by dilution. Although VAPEX is less energy intensive than SAGD, it produces at lower rates. In addition, there are also many challenges in VAPEX including it sensitivity to reservoir heterogeneity and the requirement of large amount of solvent.

Therefore, Nasr et al., in 2003, suggested to add solvent to the injected steam to improve the oil recovery or at least maintain the recovery and reduce steam injection with its associated water treatment cost and hazards (Nasr et al. 2003).This is the concept of Expanding Solvent Steam-Assisted Gravity Drainage (ES-SAGD) process. SAGD and ES-SAGD are the two most promising thermal techniques to recover heavy oil and extra heavy-oil reservoirs. The feasibility and efficiency of those processes have been studied in the literature. Generally, ES-SAGD has better performance based on the oil recovery factor, cumulative Steam Oil Ratio (cSOR) and oil production rate. However, there is no detailed analysis of their recovery performances from economical point of view and based on their Net Present Value (NPV) at the end of the project.

1.2 Problem Statement

ES-SAGD is a solvent/thermal recovery process with many operational parameters. Determining the optimal values of the most-effective control operational variables with the existing reservoir conditions is very challenging in practical life. These challenges are due to the significant co-effect of the operational parameters on the reservoir behavior and recovery performance. However, the ability to efficiently determine the appropriate values of those parameters improves the recovery performance and the project profitability.

In real life and due to the fact that optimization problems are restricted by the time and number of simulation runs required, especially, with the thermal/solvent and compositional effect in ES-SAGD, the parameters are often determined by running sensitivity studies on some or all the parameters where one parameter is varied while others are kept fixed. Then the best value of the first parameter is kept fixed and the next parameter is investigated, and so on. However, they are very limited in scope and cannot explore the entire domain of interest. Moreover, this is liable to miss optimal combination of parameters due to their likely interaction. This makes automatic optimization methods, especially the stochastic optimization algorithms, more efficient in determining the optimal operational parameters. Differential Evolution is one of the most successful optimization algorithms that have been applied in real-life and engineering problems. However, it has not been used to optimize the operational control parameters in ES-SAGD process. In addition, most of the literatures use the cumulative steam oil ratio, cSOR as the objective function to be minimized when optimizing ES-SAGD process, as they neglect the solvent loss in the formation when co-injected with steam, which leads to inaccurate representation of the project profitability. Therefore, this work will consider this solvent losses in the reservoir and use the Net

Present Value (NPV) of the project as the objective (cost) function to make our work economically representative to realistic processes.

1.3 Research Objectives

In this work, three main research objectives are proposed and are organized as following:

1- Recently, some self-adaptive strategies have been proposed to automatically adjust these parameters. Qui et al., in 2009, proposed a very promising Self-adaptive Differential Evolution algorithm (SaDE) used previous experiences of better solutions to design a self-adaptive parameter control mechanism (Qin et al. 2009). The first objective of our proposed work is to enhance SaDE algorithm in terms of solution quality, speed the convergence to optimum solution and reduce computational cost. Then study the performance of modified SaDE compared with the original SaDE on benchmark test functions.

2- This research shows how the most promising optimization techniques SaDE and proposed modified SaDE can be a robust tool for the design and performance evaluation of one of the most challenging recovery process of (ES-SAGD). In addition, this work, unlike most of the literatures, will consider solvent losses in the reservoir as well as all the economic parameters in the ES-SAGD process. Therefore, Net Present Value (NPV) at the end of the project is used as the objective (cost) function instead of cSOR to show the actual profitability, and make this study more, economically, representative to the real-life conditions.

3- SAGD and ES-SAGD are the two most promising techniques for production of heavy oil and extra-heavy oil reservoirs. The feasibility and efficiency of those processes have been studied in the literatures. Although ES-SAGD has, generally, better performance based on the recovery factor, cSOR and oil production rate, the real economic analysis of both methods has not been investigated. In this study, we will consider the NPV of both processes to be the performance indicator instead of cSOR to compare the profitability of each recovery method.

1.4 Methodology

1- Improve Self-adaptive Differential Evolution algorithm by completely modifying its adaption technique and changing both the size and type of its candidate pool of trial-vector generation strategies. This proposed modifications, will enhance solution quality and improve the convergence speed with keeping the exploration capabilities. As well as saving its computational costs.

2- Analyze the performance of proposed optimization algorithm compared with original SaDE algorithm on suite of 22 numerical optimization problems.

3- Construct a numerical flow simulation model of one of Athabasca heavy oil reservoirs by CMG STARS, a numerical flow simulation package for thermal recovery process.

4- Develop a framework that integrates previously mentioned optimization techniques, in addition to other two well-known optimization algorithms (DE and PSO), with CMG STARS, to optimize ES-SAGD recovery process.

5- A sensitivity study is done on ES-SAGD to determine a preliminary ranking of the control operational parameters according to their effect on the project's NPV. In addition, this study will figure out the realistic effective range (upper and lower limits) to be assigned to the selected operational parameters in optimization step.

6- A comparison study is done to figure out the most-effective optimization algorithm that maximized the NPV of the project while considering other performance indicators like Oil Recovery Factor (RF) and cumulative Steam Oil Ratio (cSOR).

7- Finally, for convenient, a performance analysis of optimized SAGD and ES-SAGD recovery processes is proposed.

CHAPTER 2

Evolutionary Programing

2.1 Differential Evolution Algorithm

Similar to all other Evolutionary Algorithms (EAs), the evolutionary process of DE uses mutation, crossover and selection operators at each generation to reach the global optimum. DE's performance highly depends on the mutation strategy and the crossover operator. In addition the control parameters; population size *NP* and scaling factor *F* play an important role in achieving the balance between the exploration and convergence speed of the algorithm. The original DE algorithm could be illustrated as following:

Initialization

The upper and lower bounds for each parameter must be defined before the initial population can be generated. Two D-dimensional initialization vectors, x^U and x^L, are used to store these values where *U* and *L* indicate the upper and lower bounds respectively. Once the initial bounds for each parameter have been specified, a random number generator assigns a value from within the prescribed range to each parameter of every vector. As shown, the initial value at generation ($g = 0$) of the j^{th} parameter of the i^{th} vector is:

$$x_{j,i,0} = rand_j\,(0,1).\left(x_j^u - x_j^l\right) + x_j^l \tag{2.1}$$

The random number generator, $rand_j\left(0,1\right)$ returns a uniformly distributed random number from within the range $\left[0,1\right]$. The current population, symbolized by P_x as determined in Eq.(2.2), is composed of $x_{i,g}$ vectors shown in Eq.(2.3) that have been already found to be acceptable as initial points or created randomly as mentioned above.

$$P_{x,g} = \left\{x_{i,g} : i = 1, 2, ..., NP\right\} \qquad g = 0, 1, ..., g_{\max} \tag{2.2}$$

$$X_{i,g} = (\mathrm{x}_{j,i,g}), \quad j = 1, 2, ..., D \tag{2.3}$$

The index, $g = 0, 1, ..., g_{\max}$, indicates the generation to which a vector belongs. In addition, each vector is assigned a population index, i, which runs from 1 to NP. Parameters within vectors are indexed with j, which runs from 0 to D.

Mutation

After initialization, DE mutates and recombines the population to produce a population of NP vectors $P_{v,g}$. In particular, differential mutation adds a scaled, randomly sampled, vector difference to a third vector. Eq.(2.4) shows how to combine three different, randomly chosen vectors to create a mutant vector $v_{i,g}$.

$$V_{i,g} = X_{r0,g} + F.\left(X_{r1,g} - X_{r2,g}\right) \tag{2.4}$$

The scale factor, $F \in \left(0,1\right]$, is a positive real number that controls the rate at which the population evolves. The base vector $r0$ and the other two vectors; $r1$ and $r2$ are chosen randomly and have distinct indexes from each other. Eqs.(2.5) and (2.6) describe the intermediary mutation population.

$$P_{v,g} = V_{i,g}, \quad i = 1, 2, ..., NP, \quad g = 0, 1, ..., g_{\max} \tag{2.5}$$

$$V_{i,g} = \{v_{j,i,g}\} \quad j = 1, 2, ..., D \tag{2.6}$$

Crossover

Each vector in the current population is then recombined with its mutant vector to produce the trial population, P_u , as described by Eqs.(2.7) and (2.8).

$$P_{u,g} = \{U_{i.g}\} \; i = 1, 2, ..., NP, \quad g = 0, 1, ..., g_{\max} \tag{2.7}$$

$$U_{i,g} = \left[u_{j,i,g}\right] \; j = 1, 2, ..., D \tag{2.8}$$

Differential evolution employs a uniform crossover to build trial vectors out of parameter values that have been copied from two different vectors. In particular, DE crosses each vector with a mutant vector as shown in Eq.(2.9).

$$u_{j,i,g} = \begin{cases} v_{j,i,g} & if \;\; \text{rand}_j(0,1) \le Cr \; or \; j = j_{rand} \\ x_{j,i,g} & otherwise. \end{cases} \tag{2.9}$$

Selection

If the trial vector, $U_{i,g}$, has an equal or more-optimum (lower in minimization problems, and higher in maximization problems) objective function value then that of its target vector, $X_{i,g}$, it replaces the target vector in the next generation; otherwise, the target retains its place in the population for at least one more generation as shown in Eq.(2.10).

$$X_{i,g+1} = \begin{cases} U_{i,g} & if \;\; f\,(U_{i,g}) \; is \; better \; than \; f\,(X_{i,g}) \\ X_{i,g} & otherwise. \end{cases} \tag{2.10}$$

2.1.1 Pseudo-Code for DE Algorithm

```
Step 1: Initialization
Set the generation number $g = 0$ and randomly initialize a population of $NP$ individuals
$P_{x,g} = \left\{\vec{X}_{1,g}, \vec{X}_{2,g}, ..., \vec{X}_{NP,g}\right\}$ with $\vec{X}_{1,g} = \left[x_{1,i,g}, x_{2,i,g}, ...x_{D,i,g}\right]$ and each individual
uniformly distributed in the range $\left[\vec{X}^{L}, \vec{X}^{U}\right]$, where $\vec{X}^{L} = \left\{x_1^l, x_2^l, ..., x_D^l\right\}$ and
$\vec{X}^{U} = \left\{x_1^u, x_2^u, ..., x_D^u\right\}$ with $i = \left[1, 2, ..., NP\right]$.
Step 2: WHILE the stop criteria is not satisfied
            DO
            FOR $i$ = 1 to $NP$
                Step 2.1: Mutation Step
                Generate a donor vector $\vec{V}_{i,g} = \left\{v_{1,i,g}, v_{2,i,g}, ..., v_{D,i,g}\right\}$ corresponding to
                the $i-th$ target vector $\vec{X}_{i,g}$ via one of the different mutation schemes
                of DE as per Eqs. (2.11) – (2.17).
                Step 2.2: Crossover Step
                Generate a trial vector $\vec{U}_{i,g} = \left\{u_{1,i,g}, u_{2,i,g}, ..., u_{D,i,g}\right\}$ for the $i-th$
                target vector $\vec{X}_{i,g}$ through Eqs.(2.9).
                Step 2.3: Selection Step
                Evaluate the trial vector $\vec{U}_{i,g}$
                IF $f\left(\vec{U}_{i,g}\right)$ is better than $f\left(\vec{X}_{i,g}\right)$, then $\vec{X}_{i,g+1} = \vec{U}_{i,g}$,
                $f\left(\vec{X}_{i,g+1}\right) = f\left(\vec{U}_{i,g}\right)$
                  IF $f\left(\vec{U}_{i,g}\right)$ is better than $f\left(\vec{X}_{Best,g}\right)$, then $\vec{X}_{Best,g} = \vec{U}_{i,g}$,
                  $f\left(\vec{X}_{Best,g}\right) = f\left(\vec{U}_{i,g}\right)$
                  END IF
                ELSE
                  $\vec{X}_{i,g+1} = \vec{X}_{i,g}$, $f\left(\vec{X}_{i,g+1}\right) = f\left(\vec{X}_{i,g}\right)$
                END IF
            END FOR
            Step 2.4: Increase the generation count
            $g = g + 1$
      END WHILE
```

2.1.2 Variants of Differential Evolution

Variants of DE are different in the way they perform mutation. In the literature usually different variant of DE are presented in the form of *DE / x / y / z*, where *x* is the vector that will be mutated, *y* specifies the number of difference vectors used and *z* is the crossover scheme (bin: binomial; exp: exponential).

DE / Rand / 1

In this strategy, the base vector is chosen randomly and one weighted difference vector is added to generate the mutant vector:

$$\vec{V}_{i,g+1} = \vec{X}_{r_1^i,g} + F.\left(\vec{X}_{r_2^i,g} - \vec{X}_{r_3^i,g}\right) \tag{2.11}$$

DE / Best / 1

This is similar to the *DE / Rand / 1*, but the base vector is the best vector resulting the optimum objective function (cost) instead of the random selection. As shown in Eq.(2.12) , the vectors difference is added to the best vector.

$$\vec{V}_{i,g+1} = \vec{X}_{r_1^i,g} + F.\left(\vec{X}_{r_2^i,g} - \vec{X}_{r_3^i,g}\right) \tag{2.12}$$

DE / Best / 2

This strategy is similar to *DE / Best / 1*. However, instead of one difference vector, two difference vectors, which selected randomly from the current population, are added to the base vector as shown in Eq. (2.13). The base vector is the best vector that gives the optimum value of the objection function

$$\vec{V}_{i,g+1} = \vec{X}_{Best,g} + F.\left(\vec{X}_{r_1^i,g} - \vec{X}_{r_2^i,g} + \vec{X}_{r_3^i,g} - \vec{X}_{r_4^i,g}\right) \tag{2.13}$$

DE / Rand / 2

This is similar to *DE / Rand / 1*, but two difference vectors are selected randomly and added to the base vector which is selected also randomly from the current population as shown in Eq. (2.14).

$$\vec{V}_{i,g+1} = \vec{X}_{r_1^i,g} + F.\left(\vec{X}_{r_2^i,g} - \vec{X}_{r_3^i,g} + \vec{X}_{r_4^i,g} - \vec{X}_{r_5^i,g}\right) \tag{2.14}$$

DE / Rand-to-Best / 1

In this strategy, some of the other strategies to create donor vector are mutated recombinants. It mutates a two-vector recombinant: $\vec{X}_{i,g} + F.\left(\vec{X}_{best,g} - \vec{X}_{i,g}\right)$ as shown in Eq. (2.15).

$$\vec{V}_{i,g+1} = \vec{X}_{i,g} + F.\left(\vec{X}_{best,g} - \vec{X}_{i,g}\right) + F.\left(\vec{X}_{r_1^i,g} - \vec{X}_{r_2^i,g}\right) \tag{2.15}$$

DE / Rand-to-Best / 2

This is similar to *DE / Rand-to-Best / 1*, but two difference vectors are selected randomly.

$$\vec{V}_{i,g+1} = \vec{X}_{i,g} + F.\left(\vec{X}_{best,g} - \vec{X}_{i,g}\right) + F.\left(\vec{X}_{r_1^i,g} - \vec{X}_{r_2^i,g}\right) + F.\left(\vec{X}_{r_3^i,g} - \vec{X}_{r_4^i,g}\right) \tag{2.16}$$

DE / Current-to-Rand / 1

Is a rotation-invariant strategy, its effectiveness has been verified when it was applied to solve multi-objective optimization problem.

$$\vec{V}_{i,g+1} = \vec{X}_{i,g} + F.\left(\vec{X}_{r_1,g} - \vec{X}_{i,g}\right) + F.\left(\vec{X}_{r_2^i,g} - \vec{X}_{r_3^i,g}\right) \tag{2.17}$$

Where $r_1^i, r_2^i, r_3^i, r_4^i$ and r_5^i are vectors selected randomly from the current population. $\vec{X}_{Best,g}$ is the best vector resulting the optimum value of the objective function.

2.2 Literature Review on DE

Researchers, over the past few years, have been investigating ways to improve the DE performance by tuning its control parameters. Storn and Price indicated that a reasonable value for *NP* cold be between 5*D* and 10*D* (D is the dimension of the problem), and a good initial choice of *F* could be 0.5 (Storn and Price 1995).

In 2002, Gamperle et al. tested different control parameters of DE on the Rosenbrock's, Sphere, and Rastrigin's functions (Gamperle et al. 2002). Their results showed that the global optimum searching capacity and the convergence speed are very sensitive to the choice of control parameters *NP, F* and *CR*. In addition, the best range of the population size *NP* is between 3*D* and 8*D*, with scaling factor *F* = 0.6 and *CR* in [0.3, 0.9].

In 2005, Ronkkonen et al. showed that typical F range is [0.4, 0.95] with F = 0.9 is a good first choice. CR usually lies in (0, 0.2) when function is separable, while in (0.9, 1) when the function's parameters are dependent (Ronkkonen et al. 2005).

2.2.1 Literature Review on Performance Comparison of DE and Other Optimization Algorithms

It is demonstrated that the Differential Evolution method converges faster and with more certainty than both Adaptive Simulated Annealing and the Annealed Nelder & Mead method. As the DE is robust, easy, requires few control parameters and lends itself very well to parallel computation (Storn and Price 1995). Storn et.al tested the DE method on function test-bed contains De Jong functions as presented in Ingber (1992) plus some additional functions which present further distinctive difficulties for global minimizer. The results showed that the Differential Evolution method was the only strategy to converge for all the functions in the test function suite and that could find all global minima of the test suite in the least number of function evaluations (Storn and Price 1995). Because the Differential Evolution technique is inherently parallel, a more significant speed up could be obtained if the algorithm is executed in a parallel machine or computer network, which is very useful in real-practical problems where optimizing the objective function requires extensive computational time.

Another study was done by Das et.al to compare the popular optimization method, Particle Swarm Optimization PSO with the Differential Evolution algorithm, where both algorithms do not require any gradient information of the function to be optimized, uses only primitive mathematical operators and are conceptually very simple. They also concluded that the DE performs better than PSO on the different test functions they used *(Das et al. 2008).*

In 2012. A performance comparison of GA, DE, PSO and SA methods was done by K. Chandrasekar et al., they also found that the Differential Evolution outperforms GA, PSO and SA methods both in enhancement of TTC and computational efficiency (Chandrasekar and Ramana 2012).

However, little is known about DE's scaling property and behavior in real-world applications and it is important for practical application to gain more knowledge on how to choose the control variables for DE for a particular type of problem.

2.2.2 Literature Review on Self-Adaptive DE Algorithms

In real-world optimization problems, this could be confusing for engineers, as several claims and counter claims were reported to choose the control parameters of DE. Therefore, researchers developed techniques to be self-adaptive in order to avoid manual parameters adjusting. Usually, self-adaption is applied to tune the control parameters *F* and *CR*.

Abbass self-adapted the crossover rate *CR* for multi-objective optimization problems, by encoding the value of *CR* into each individual and simultaneously evolving it with other search variables. The scaling factor *F* was generated for each variable from a Gaussian distribution *N* (0, 1) (Abbass 2002).

In 2003, Zaharie proposed a parameter adaption strategy for DE (ADE) based on the idea of controlling the population diversity, and implemented a multi-population approach (Zaharie 2003).

In 2005, Omran et al. proposed a self-adaptive scaling factor parameter *F* (Omran et al. 2005). They generated the value of CR for each individual from a normal distribution N (0.5, 0.15). This approach (SDE) was tested on four benchmark functions and performed better than other DE versions.

In addition to adapting the control parameters F or/and CR, some researchers also adapted the population size NP. Teo introduced DE with self-adaptive populations (DESAP)(Teo 2006), based on Abbass' self-adaptive Pareto DE (Abbass 2002).

Brest et al. encoded control parameters F and CR into the individual and evolved their values by using two new probabilities τ_1 and τ_2. In their algorithm (SADE), a set of F values was assigned to each individual in the population. With probability τ_1, F is reinitialized to a new random value in the range [0.1, 1.0], otherwise it is kept unchanged. The crossover CR assigned to each individual is adapted in an identical fashion, but with a different re-initialization range of [0, 1] and with the probability τ_2. With probability τ_2, CR takes a random value in [0, 1]. Otherwise it retains it earlier value in the next generation.

In 2008, Rahnamayan et al. introduced an Opposition-based DE (ODE) that is specially suited for noisy optimization problems. The conventional DE algorithm was enhanced by utilizing the opposition number-based optimization concept in three levels, namely population initialization, generation jumping, and local improvement of the population's best member (Rahnamayan et al. 2008).

Norman and Iba proposed the Fittest Individual Refinement (FIR); a crossover-based local search method for DE. The FIR scheme accelerates DE by enhancing its search capability through exploration of the neighborhood of the best solution in successive generation. (Noman and Iba 2008)

In 2009, Qin et al. proposed a Self-adaptive DE (SaDE) algorithm (Qin et al. 2009), in which both the trial vector generation strategies and their associated parameters are gradually self-adaptive by learning from their previous experience of generation promising solutions.

Inspired by SaDE algorithm and motivated by the recent success of diverse self-adaptive DE approaches, Mallipeddi et al. developed a self adaptive DE, called EPSDE, based on ensemble approach (Mallipeddi et al. 2011). In EPSDE, a pool of distinct mutation strategies along with a pool of values for each control parameter coexists throughout the evolution process and competes to produce offspring. The performance of EPSDE was evaluated on a set of bound constrained problems and compared with conventional DE and other state-of-the-art parameter adaptive DE variants. The comparative results showed that EPSDE algorithm outperformed conventional DE and other state-of-the-art parameter adaptive DE variants in terms of solution quality and robustness.

Gang et al. proposed a hybrid DE based on the one-step k-means clustering and 2 multi-parent crossovers, called clustering-based differential evolution with 2 multi-parent crossovers (2-MPCs-CDE) for the unconstrained global optimization problems (Liu et al. 2012). In 2-MPCs-CDE, k cluster centers and several new individuals generate two search spaces. These spaces are then searched in turn. This method utilized the information of the population effectively and improves search efficiency. Hence it can enhance the performance of DE. A comprehensive set of 35 benchmark functions was employed for experimental verification. Experimental results indicated that 2-MPCs-CDE is effective and efficient.

Piotrowski et al. presented an algorithm to improve optimization performance, namely DE with Separated Groups (DE-SG) (Piotrowski et al. 2012), which distributed population into small groups, defined rules of exchange of information and individuals between the groups and used two different strategies to keep balance between exploration and exploitation capabilities. The performance of DE-SG is compared to that of eight algorithms belonging to the class of Evolutionary Strategies (Covariance Matrix Adaptation ES), Particle Swarm Optimization (Comprehensive Learning PSO and Efficient Population Utilization Strategy PSO), Differential Evolution (Distributed DE with explorative-exploitative population families, Self-adaptive DE, DE with global and local neighbors and Grouping Differential Evolution) and multi-algorithms (AMALGAM). Although slow for simple functions, the DE-SG algorithm achieved a good success rate for more difficult 30- and 50-dimensional problems.

In 2013, Mohamed et al. introduced an Effective Differential Evolution (EDE) algorithm for solving real parameter optimization problems over continuous domain (Mohamed et al. 2013). The proposed algorithm proposed a new mutation rule based on the best and the worst individuals among the entire population of a particular generation. The mutation rule is combined with the basic mutation strategy through a linear decreasing probability rule. The proposed mutation rule is shown to promote local search capability of the basic DE and to make it faster. Furthermore, a random mutation scheme and a modified Breeder Genetic Algorithm (BGA) mutation scheme are merged to avoid stagnation and/or premature convergence. Additionally, the scaling factor and crossover of DE are introduced as uniform random numbers to enrich the search behavior and to enhance the diversity of the population. The EDE algorithm is shown to be competitive with other algorithms in terms of final solution quality, efficiency, convergence rate, and robustness.

2.3 Self-adaptive Differential Evolution

The SaDE algorithm can be described in two major steps:

Adaption of Trial Vector Generation Strategy

When solving different optimization problems, DE realizations using diverse trial vector generation strategies typically perform differently. Unlike using the computationally expensive trial-and-error search for the most suitable strategy and its associated control parameters, Qin et al., in 2009, kept a strategy candidate pool including several effective trial vector generation strategies with effective yet various characteristics (Qin et al. 2009). With respect to each target vector in the existing population, during evolution, one strategy will be chosen from the candidate pool according to a probability learned from its previous experience of generating promising solutions and applied to execute the mutation task. The more successfully one strategy behaved in previous generations to generate favorable solutions, the more probably it will be chosen in the current generation to generate solutions.

In SaDE algorithm, four trial vector generation strategies are used; "*DE/rand/1/bin*" Eq. (2.11), "*DE/rand/2/bin*" Eq. (2.14), "*DE/rand-to-best/2/bin*" Eq.(2.16), and "DE/current-to-rand/1" Eq. (2.17), as candidate pool.

In the SaDE algorithm, one trial vector generation strategy is selected from the candidate pool, with respect to each target vector in the current population, according to the probability learned from its success rate in generating improved solutions within a certain number of previous generations. The selected strategy is subsequently applied to the corresponding target vector to generate a trial vector. More specifically, at each generation, the probabilities of choosing each strategy in the candidate pool are summed to 1. These probabilities are gradually adapted during evolution in the following manner.

Assume that the probability of applying the k^{th} strategy in the candidate pool to a target vector in the current population is $p_k, k = 1, 2, ..., K$, where K is the total number of strategies contained in the pool. The probabilities with respect to each strategy are initialized as $1/k$, i.e., all strategies have the equal probability to be chosen. SaDE used the stochastic universal selection method to select one trial vector generation strategy for each target vector in the current population. At the generation g, after evaluating all the generated trial vectors, the number of trial vectors generated by the k^{th} strategy that can successfully enter the next generation is recorded as $ns_{k,g}$, while the number of trial vectors generated by the k^{th} strategy that are discarded in the next generation is recorded as $nf_{k,g}$. SaDE has success and failure memories to store these numbers within a fixed number of previous generations hereby named learning period (LP). As illustrated in Table 2.1 and Table 2.2, at the generation g, the number of trial vectors generated by different strategies that can enter or fail to enter the next generation over the previous LP generations are stored in different columns of the success and failure memories. Once the memories overflow after LP generations, the earliest records stored in the memories, i.e. ns_{g-LP} or nf_{g-LP} will be removed so that those numbers calculated in the current generation can be stored in the memories, as shown in Table 2.3.

After the initial LP generations, the probability of choosing different strategies will be updated at each subsequent generation based on the success and failure memories as following:

$$p_{k,g} = \frac{S_{k,g}}{\sum_{k=1}^{K} S_{k,g}} \tag{2.18}$$

where:

$$S_{k,g} = \frac{\sum_{g=g_{\max}-LP}^{g_{\max}-1} ns_{k,g}}{\sum_{g=g_{\max}-LP}^{g_{\max}-1} ns_{k,g} + \sum_{g=g_{\max}-LP}^{g_{\max}-1} nf_{k,g}} + \varepsilon, \quad k = 1, 2, ..., K;\ g_{\max} > \text{LP} \tag{2.19}$$

where $S_{k,g}$ represents the success rate of the trial vectors generated by the k^{th} strategy and successfully entering the next generation with the previous LP generation with respect to generation g. The small constant value $\varepsilon = 0.01$ is used to avoid the possible null success rates. To ensure that the probabilities of choosing strategies are always summed to 1, we further divide $S_{k,g}$ by $\sum_{k=1}^{K} S_k$ to calculate $p_{k,g}$. Obviously, the larger the success rate for the k^{th} strategy within the previous LP generation is, the larger the probability of applying it to generate the trial vectors at the current generation.

Table 2.1: Success memory

Index	Strategy 1	Strategy 2	. . .	Strategy K
1	$ns_{1,g-LP}$	$ns_{2,g-LP}$	. . .	$ns_{k,g-LP}$
2	$ns_{1,g-LP+1}$	$ns_{2,g-LP+1}$	. . .	$ns_{k,g-LP+1}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
LP	$ns_{1,g-1}$	$ns_{2,g-1}$	. . .	$ns_{k,g-1}$

Table 2.2: Failure memory

Index	Strategy 1	Strategy 2	. . .	Strategy K
1	$nf_{1,g-LP}$	$nf_{2,g-LP}$	. . .	$nf_{k,g-LP}$
2	$nf_{1,g-LP+1}$	$nf_{2,g-LP+1}$	. . .	$nf_{k,g-LP+1}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
LP	$nf_{1,g-1}$	$nf_{2,g-1}$	. . .	$nf_{k,g-1}$

Table 2.3: Progress of Success memory

$ns_{1,g-LP}$	. . .	$ns_{k,g-LP}$
$\vdots$	. . .	$\vdots$
$ns_{1,g-1}$	. . .	$ns_{k,g-1}$

Generation g

$\rightarrow$

$ns_{1,g-LP+1}$	. . .	$ns_{k,g-LP+1}$
$\vdots$		$\vdots$
$ns_{1,g}$	. . .	$ns_{k,g}$

Generation $g+1$

$\rightarrow$

$ns_{1,g-LP+2}$	. . .	$ns_{k,g-LP+2}$
$\vdots$	. . .	
$ns_{1,g+1}$	. . .	$ns_{k,g+1}$

Generation $g+2$

$\rightarrow \ldots$

Adaption of Control Parameters

In the conventional DE, the choice of numerical values for the three control parameters F, CR, and NP highly depends on the problem under consideration. In the SaDE algorithm, NP is left as a user-specified parameter because it highly replies on the complexity of a given problem. In fact, the population size NP does not need to be fine-tuned and just a few typical values can be tried according to the pre-estimated complexity of the given problem. Between other two parameters, CR is usually more sensitive to problems with different characteristics, e.g., the uni-modality and multimodality, while F is closely related to the convergence speed (Qin et al. 2009). In SaDE algorithm, the parameter F is approximated by a normal distribution with mean value 0.5 and standard deviation 0.3, denoted by $N\left(0.5, 0.3\right)$. A set of F values are randomly sampled from such normal distribution and applied to each target vector in the current population. It is easy to verify that values of F must fall into the range $\left[-0.4, 1.4\right]$ with the probability of 0.997. By doing so, we attempt to maintain both exploitation (with small F values) and exploration (with large F values) capabilities throughout the entire evolution process.

The proper choice of CR can lead to successful optimization performance while a wrong choice may deteriorate the performance. In fact, good values of CR generally fall into a small range for a given problem, with which the algorithm can perform consistently well. Therefore, SaDE considered gradually adjusting the range of CR values for a given problem according to previous CR values that have generated trial vectors successfully entering the next generation. Specifically, Qin et al., in 2009, assumed that CR obeys a normal distribution with mean value CR_m and standard deviation $Std = 0.1$, denoted by

$N\left(CR_m, Std\right)$ where CR_m is initialized as 0.5. The Std should be set as a small value to guarantee that most CR values generated full within $[0,1]$, even when CR_m is near 0 or 1. Hence, the value of Std is set as 0.1. Experiments showed that minor changes to the Std of the Gaussian distribution do not influence the performance of SaDE significantly (Qin et al. 2009).

In SaDE, the value of CR_m is adapted with respect to each trial vector generation strategy as following:

With respect to the k^{th} strategy, the value of $CR_{m,k}$ is initialized to 0.5. A set of CR values are randomly generated according to $N\left(CR_{m,k}, 0.1\right)$ and then applied to those target vectors to which the k^{th} strategy is assigned. To adapt the crossover rate CR, memories named CR_{Memory_k} are established to store those CR values with respect to the k^{th} strategy that have generated trial vectors successfully entering the next generation within the previous LP generations. Specifically, during the first LP generations, CR values with respect to k^{th} strategy are generated by $N\left(CR_{m,k}, 0.1\right)$. At each generation after LP generations, the median value stored in CR_{Memory_k} will be calculated to overwrite $CR_{m,k}$. Then, CR values can be generated according to $N\left(CR_{m,k}, 0.1\right)$ when applying the k^{th} strategy. After evaluating the newly generated trial vectors, CR values in CR_{Memory_k} that correspond to earlier generations will be replaced by promising CR values obtained at the current generation with respect to the k^{th} strategy.

In the SaDE algorithm, both trial vector generation strategies and their associated control parameters are gradually self-adapted by learning their previous experiences of generating promising solutions. Consequently, a more suitable strategy along with its parameter setting can be determined adaptively to suit different phases of the search process. Extensive experiments described and done by Qin et al., in 2009, verified the promising performance of the SaDE to handle problems with distinct properties such as uni-modality and multi-modality (Qin et al. 2009).

2.3.1 Pseudo-Code for SaDE Algorithm

```
Step 1: Set the generation number $g = 0$ and randomly initialize a population of $NP$
        individuals $P_{x,g} = \{\vec{X}_{1,g}, \vec{X}_{2,g}, ..., \vec{X}_{NP,g}\}$ with $\vec{X}_{1,g} = [x_{1,i,g}, x_{2,i,g}, ...x_{D,i,g}]$ and
        each individual uniformly distributed in the range $[\vec{X}^L, \vec{X}^U]$, where
        $\vec{X}^L = \{x_1^l, x_2^l, ..., x_D^l\}$ and $\vec{X}^U = \{x_1^u, x_2^u, ..., x_D^u\}$ with $i = [1, 2, ..., NP]$.
        Initialize the median value of $CR\,(CR_{m,k})$, strategy probability $p_{k,g}$, $K$ is the
        number of available strategies, and learning period $LP$ .
Step 2: Evaluate the population
Step 3: WHILE stopping criterion is not satisfied
            DO
            Step 3.1: Calculate strategy probability $p_{k,g}$ and update the success and
                      failure memory
                      IF $g \geq LP$
                         FOR $k = 1\ to\ k$
                             Update the $p_{k,g}$ by Eq. (2.18)
                             Remove $ns_{k,G-LP}$ and $nf_{k,G-LP}$ out of the Success and
                             Failure Memory respectively.
                         END FOR
                      END IF
            Step 3.2: Assign trial vector generation strategy and parameter to each
                      target vector $X_{i,g}$
                      Assign trial vector generation strategy
                      Using stochastic universal sampling to select one strategy $k$ for
                      each target vector $X_{i,g}$
```

```
**Assign control parameter** $F$
FOR $i = 1 \text{ to } NP$
        $F_i = Normrnd\left(0.5, 0.3\right)$
END FOR
**Assign control parameter** $CR$
IF $g \geq LP$
    FOR $k = 1 \text{ to } K$
        $CR_{m,k} = median\left(CR_{Memory_k}\right)$
    END FOR
END IF
FOR $k = 1 \text{ to } K$
      FOR $i = 1 \text{ to } NP$
            $CR_{k,i} = Normrnd\left(CR_{m,k}, 0.1\right)$
            WHILE $CR_{m,i} < 0$ or $CR_{m,\mathrm{i}} > 1$
                    $CR_{k,i} = Normrnd\left(CR_{m,k}, 0.1\right)$
            END WHILE
      END FOR
END FOR
```

Step 3.3: Generate a new population where each trial vector $U_{i,g}^{k}$ is generated according to associated trial vector generation strategy k and parameters F_i and $CR_{k,i}$ in Step 3.2.

Step 3.4: Randomly reinitialize the trial vector $U_{i,g}^{k}$ within the search space if any variable is outside it boundaries.

Step 3.5: Selection

```
FOR $i = 1 \text{ to } NP$
      Evaluate the trial vector $U_{i,g}^{k}$
      IF $f\left(U_{i,g}^{k}\right)$ is better than $f\left(X_{i,g}\right)$
          $X_{i,g+1} = U_{i,g}^{k}, \quad f\left(X_{i,g+1}\right) = f\left(U_{i,g}^{k}\right)$
          $ns_{k,g} = ns_{k,g} + 1$
          Store $CR_{k,i}$ into $CR_{Memory_k}$
          IF $f\left(U_{i,g}^{k}\right)$ is better than $f\left(X_{best,g}\right)$
              $X_{best,g} = U_{i,g}^{k}, \quad f\left(X_{best,g}\right) = f\left(U_{i,g}^{k}\right)$
          END IF
```

```
                ELSE
                    $nf_{k,G} = nf_{k,G} + 1$
                END IF
            END FOR
            Store $ns_{k,g}$ and $nf_{k,g}$, $(k = 1, 2, ..., K)$ into the Success and
            Failure Memory respectively.
        Step 3.6: Increment the generation count
            $g = g + 1$
END WHILE
```

2.4 Proposed Work: Modified Self-adaptive Differential Evolution

Qin et al, when proposing self-adaptive DE algorithm, overcame the dilemma of selecting appropriate trial vector generation strategy along with its associated parameter values. Therefore, SaDE algorithm avoided the expensive computational costs spent on searching for the optimum strategy for each optimization problem. However, a good candidate pool should be restrictive so that the unfavorable influences of less effective strategies can be suppressed. Moreover, a set of effective strategies contained in a good candidate pool should have diverse characteristics. That is, the used strategies should demonstrate distinct capabilities when dealing with optimization problem. Generally speaking, having more strategies in candidate pool means an obligation of more number of function evaluations to achieve the success of self-adaptive algorithm. In other words, SaDE depends on spreading out number of population NP, equally, on the strategies in candidate pool at the initialization step. Having insufficient population vectors for each strategy, results in bad performance in the learning-period stage, and would give non-representative success and failure rates for each strategy, which is the main indicator to self-adapt DE algorithm to a successful trial-vector generation strategy in the optimization problem.

The main advantage of SaDE is avoiding the expensive cost of computation and this would not be valid if large number of strategies are selected in the candidate pool. On the other hand, having less number of strategies could result in bad optimization performance on same problem due to the lack of variety in strategy characteristics. Therefore, the main challenges that are investigated and solved by the proposed algorithm can be classified as: firstly, form a good candidate pool with the minimum number of effective strategies that possess exploration and exploitation capabilities. Secondly, improve the convergence speed without allowing the algorithm to be stagnant in local optima.

These two challenges become very critical when dealing with real-life complex optimization problems which are very expensive in terms of computational cost.

In this work, we modify and improve the Self-adaptive Differential Evolution SaDE algorithm proposed by Qin et al. (Qin et al. 2009) in terms of higher solution quality, convergence speed enhancement and maintaining the exploration capabilities required for successful DE algorithm. The proposed algorithm, which is been referred as Modified Self-adaptive Differential Evolution (MSaDE), introduces new technique to determine the success rate of each trial-vector generation strategy based on the quality of improvement in solution toward optimum solution that each strategy achieves, not on the successful number of entries to following generations as applied in original SaDE. In addition, to downsize the candidate pool to include only two effective trial-vectors generation strategies instead of four as described in the original SaDE. Therefore, the modification proposed to SaDE can be divided to two major steps:

A New Technique to Adapt Trial-Vector Generation Strategy

In SaDE, Qin et al. (Qin et al. 2009) used the success and failure rates in learning period to determine which strategy of generating trial vector is more effective as shown in Eqs. (2.18) and (2.19). then, in the following generations, more population vectors are assigned to the more successful strategy. In other words, the original SaDE algorithm determined the success rate $S_{k,g}$ of k^{th} strategy by calculating how many times (ns_k) that strategy, successfully, enter to the next generation as shown in Eq. (2.19), neglecting the “quality” of improvement that k^{th} strategy has achieved toward the optimum solution. Therefore, if a strategy A enters next generations in learning period more than strategy B, even if strategy B has a bigger effect on the solution improvement toward optimum solution, strategy A will still be considered more successful than strategy B. Consequently, in next generation, original SaDE algorithm will assign more population vectors to strategy A than B, which will result in misrepresentation in both strategies performance and weakening the real-effective strategy that leads to , relatively, bad results.

In order to overcome this dilemma, we introduce a different technique to determine the successful rate $S^*_{k,g}$ of a strategy k. The proposed technique depends on the “quality” of solution improvement that has been achieved by strategy k within the learning period toward optimum solution as following:

After the initial learning period LP, at generation g, the probability of choosing k^{th} strategy ($k = 1, 2, ..., K$) is update by Eq. (2.20).

$$p^*_{k,g} = \frac{S^*_{k,g}}{\sum_{k=1}^{K} S^*_{k,g}} \tag{2.20}$$

where $S^*_{k,g}$ is determined by Eq. (2.21).

$$S^*_{k,g} = \frac{\sum_{g=g_{\max}-LP}^{g_{\max}-1} WF_{k,g}}{\sum_{k=1}^{K}\sum_{g=g_{\max}-LP}^{g_{\max}-1} WF_{k,g}} + \varepsilon, \ \left(k = 1, 2, ..., K; g_{\max} > LP\right) \tag{2.21}$$

The Weight Factor, $WF_{k,g}$, is determined by Eqs. (2.22) and (2.23). in maximization and minimization optimization problems respectively.

$$WF_{k,g} = f\left(U_g\right)_k - f\left(X_g\right)_k \tag{2.22}$$

$$WF_{k,g} = f\left(X_g\right)_k - f\left(U_g\right)_k \tag{2.23}$$

where:

X_g	: is the target vector
U_g	: is the trial vector
ε	: is small constant (0.01) used to avoid null success rate

As shown, the success rate in the proposed Modified Self-adaptive Differential Evolution (MSaDE) depends on the solution improvement that achieved by each strategy which, truly, represents the actual-success of a strategy and enables the algorithm to be self-adapted to the real-effective strategy. This novel technique enhances the solution quality and speed the convergence rate.

Then, a combination technique between the old and new methods is also tested. In which the probability $p_{k,g}$ from Eq. (2.18) and probability $p^*_{k,g}$ from Eq. (2.20), are combined as :

$$p^{**}_{k,g} = \left(p^{*}_{k,g}\right).\left(p_{k,g}\right) \tag{2.24}$$

$p^{**}_{k,g}$ combines two success indications for strategy k ; the successful number of entries for k^{th} strategy represented by $p_{k,g}$, as well as the solution improvement represented by $p^{*}_{k,g}$. Therefore, in this combination technique, it can be shown that if a strategy k has successfully entered the next generation and, at the same time, improve the solution better than remaining strategies in the candidate pool, the success probability is significantly increased. Subsequently, the convergence speed will be increased. This excess acceleration in convergence is preferable in solving simple optimization methods with low dimensions where the algorithm is able to find out, quickly, the successful strategy to dominate the generation of trial-vectors. However, in complex optimization problems, this combination technique could be ineffective and cause premature convergence and stagnation problems at local optima.

Therefore, the effectiveness of the proposed technique and combination technique is compared with the original technique to find out the best utilized technique with self-adapted differential evolution algorithm. The numerical experiments and results in this study show that the proposed new-adaption technique and combination technique are more effective, in terms of solution quality and convergence speed, than the old-adaption technique of SaDE.

Downsize Candidate Pool of Strategies

The successful candidate pool of trial-vector generation strategies should have the capabilities of exploration and exploitation. The importance of exploration in DE is to overcome the problem of premature convergence, where the population converges to some local optima, losing its diversity and resulting in stagnation. On the other hand, the exploitation features enable the algorithm to have fast convergence speed toward the optimal solution.

Original SaDE has four strategies; "*DE/Rand/1/bin*", "*DE/Rand/2/bin*", "*DE/Rand-to-Best/2/bin*", and "*DE/Current-to-Rand/1*" as illustrated in Eqs. (2.11), (2.14), (2.16) and (2.17) respectively. Qin et al., in 2009, tried to make the candidate pool having all the effective strategies on the literature and typically possess the exploration and exploitation features. However, and as mentioned before, it is very difficult to be applied on the real-life and complex engineering problems as it requires large number of population vectors to be initiated which means huge number of function evaluations. In addition, Qin et al, in 2009, admitted that the optimal pool size and strategies selection deserve further investigation (Qin et al. 2009).

The proposed MSaDE optimization algorithm has only two strategies to generate trial-vectors. These two strategies have been selected to preserve the exploration and exploitation capabilities as well as improve convergence speed by distributing population vectors NP on only two strategies instead of four as proposed in the original SaDE. Downsizing the candidate pool and having effective strategies saves a lot of computational time and results in higher solution quality. The two strategies selected in MSaDE are: "*DE/Best/1/bin*" [Eq. (2.12)]; which relies on the best solution found so far and has fast convergence speed, and “*DE/Rand/1/bin*” [Eq. (2.11)]; which usually demonstrates slow convergence but bears stronger exploration capability. Having these two strategies in the same candidate pool, achieves the required balance between exploration and exploitation capabilities of the MSaDE algorithm, in addition to reducing the function evaluations required to find the optimum solution. The numerical experiments in the study shows that downsizing the candidate pool, with effective selected strategies to generate trial-vectors in mutation step, improves the solution quality and increase convergence speed.

The remaining of this work refers to self-adaptive DE algorithms as following:

SaDE:	The original Self-adaptive Differential Evolution with four trial-vector generation strategies in the candidate pool, and original adaption technique [Eq. (2.18)].
SaDE-2:	In this algorithm, the original adaption method is utilized with changing and reducing number of trial-vector generation strategies to “*DE/Rand/1/bin*” and “*DE/Best/1/bin*”.
MSaDE-1:	Refers to the proposed Modified Self-adaptive DE algorithm in which the novel adaption technique shown in Eq. (2.20) is used and downsizing candidate pool of trial vector generation strategies to two as mentioned.
MSaDE-2:	Is similar to MSaDE-1, but with combination adaption technique shown in Eq. (2.24).

2.5 Numerical Experiments

In this section, a comparison study is done to analyze the performance of proposed algorithms (SaDE-2, MSaDE-1 and MSaDE-2) versus the original SaDE algorithm.

2.5.1 Benchmark Functions

In order to evaluate the performance of the proposed algorithms, twenty two well-known benchmark test functions mentioned by (Yao et al. 1999) ; (Hedar 2007), presented inTable 2.4, are used.

Functions $f_1 - f_{11}$ are high-dimensional problems. Functions $f_1 - f_5$ are unimodal. Functions $f_6 - f_{11}$ are multi-modal functions where the number of local minima increases exponentially with the problem dimension. They seem to be the most difficult class of problems for many optimization problems (Yao et al. 1999). Functions $f_{12} - f_{22}$ are low-dimensional functions, which have only a few local minima.

Table 2.4 shows the benchmark functions used in this study. n is the dimension of the function, $f_{\min}$ is the minimum value of the function, and $S \subseteq R^n$.

Table 2.4: Benchmark of test functions

Test Functions	n	S	$f_{\min}$
$f_1(x)=\sum_{i=1}^{n} x_i^2$	$10,30$	$[-100,100]^n$	0
$f_2(x)=\sum_{i=1}^{n}(x_i+0.5)^2$	$10,30$	$[-100,100]^n$	0
$f_3(x)=\sum_{i=1}^{n} i x_i^2$	$10,30$	$[-10,10]^n$	0
$f_4(x)=\sum_{i=1}^{n}\lvert x_i\rvert+\prod_{i=1}^{n}\lvert x_i\rvert$	$10,30$	$[-10,10]^n$	0
$f_5(x)=\sum_{i=1}^{n-1}\left[100\left(x_{i+1}-x_i^2\right)^2+\left(x_i-1\right)^2\right]$	$10,30$	$[-30,30]^n$	0
$f_6(x)=-20\exp\left(-0.2\sqrt{\frac{1}{30}\sum_{i=1}^{n} x_i^2}\right) -\exp\left(\frac{1}{30}\sum_{i=1}^{n}\cos 2\pi x_i\right)+20+e$	$10,30$	$[-32,32]^n$	0
$f_7(x)=\frac{1}{400}\sum_{i=1}^{n} x_i^2-\prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$	$10,30$	$[-600,600]^n$	0
$f_8(x)=\frac{\pi}{n}\left\{10\sin^2(\pi y_1)+\sum_{i=1}^{n-1}(y_i-1)^2.\left[1+10\sin^2(\pi y_{i+1})\right]+(y_n-1)^2\right\} +\sum_{i=1}^{n}u(x_i,10,100,4)$	$10,30$	$[-50,50]^n$	0
$f_9(x)=0.1\left\{\sin^2(\pi 3x_1)+\sum_{i=1}^{n-1}(x_i-1)^2.\left[1+\sin^2(3\pi x_{i+1})\right]+(x_n-1)^2\left[1+\sin^2(2\pi x_n)\right]\right\} +\sum_{i=1}^{n}u(x_i,5,100,4)$	$10,30$	$[-50,50]^n$	0
$f_{10}(x)=\sum_{i=1}^{n}\left[x_i^2-10\cos(2\pi x_i)+10\right]$	$10,30$	$[-5.12,5.12]^n$	0
$f_{11}(x)=\sum_{i=1}^{n/4}\left[\left(x_{4i-3}+10x_{4i-2}\right)^2+5\left(x_{4i-1}-x_{4i}\right)^2 +\left(x_{4i-2}-2x_{4i-1}\right)^4+10\left(x_{4i-3}-x_{4i}\right)^4\right]$	$8,28$	$[-4,5]^n$	0

Cont.../

Test Functions	n	S	$f_{\min}$
$f_{12}(x)=\left[\frac{1}{500}+\sum_{j=1}^{25}\frac{1}{j+\sum_{i=1}^{2}\left(x_i-a_{ij}\right)^6}\right]^{-1}$	2	$[-65.536,65.536]^n$	1
$f_{13}(x)=\sum_{i=1}^{11}\left[a_i-\frac{x_1\left(b_i^2+b_ix_2\right)}{b_i^2+b_ix_3+x_4}\right]^2$	4	$[-5,5]^n$	0.0003075
$f_{14}(x)=4x_1^2-2.1x_1^4+\frac{1}{3}x_1^6+x_1x_2-4x_2^2+4x_2^4$	2	$[-5,5]^n$	−1.0316285
$f_{15}(x)=\left(x_2-\frac{5.1}{4\pi^2}x_1^2+\frac{5}{\pi}x_1-6\right)^2 +10\left(1-\frac{1}{8\pi}\right)\cos x_1+10$	2	$[-5,10]x\,[0,15]$	0.398
$f_{16}(x)=\left[1+\left(x_1+x_2+1\right)^2\begin{pmatrix}19-14x_1+3x_1^2-14x_2\\+6x_1x_2+3x_2^2\end{pmatrix}\right] \times\left[30+\left(2x_1-3x_2\right)^2\times\begin{pmatrix}18-32x_1+12x_1^2+48x_2\\-36x_1x_2+27x_2^2\end{pmatrix}\right]$	2	$[-2,2]^n$	3
$f_{17}(x)=-\sum_{i=1}^{4}c_i\exp\left[-\sum_{j=1}^{n}a_{ij}\left(x_j-p_{ij}\right)^2\right]$	3	$[0,1]^n$	−3.86
$f_{18}(x)=\sum_{i=1}^{n}\left(x_i-1\right)^2-\sum_{i=2}^{n}x_ix_{i-1}$	6	$\left[-n^2,n^2\right]^n$	−50
$f_{19}(x)=-\sum_{i=1}^{n}\sin\left(x_i\right)\sin^{2m}\left(\frac{ix_i^2}{\pi}\right)$	5	$[0,\pi]^n$	−4.687658
$f_{20}(x)=\left(\sum_{i=1}^{5}i\cos\left((i+1)x_1+i\right)\right) .\left(\sum_{i=1}^{5}i\cos\left((i+1)x_2+i\right)\right)$	2	$[-10,10]^n$	−186.7309
$f_{21}(x)=\sum_{i=1}^{n}\left[\left(\sum_{j=1}^{n}x_j^i\right)-b_i\right]^2$	4	$[0,n]^n$	0
$f_{22}(x)=\sum_{i=1}^{n}\left[\sum_{j=1}^{n}\left(j^i+\beta\right)\left(\left(\frac{x_i}{j}\right)^i-1\right)\right]^2$	4	$[-n,n]^n$	0

2.5.2 Experimental Setup

Experiments were conducted on the twenty-two functions to evaluate the performance of four algorithms SaDE, SaDE-2, MSaDE-1 and MSaDE-2. For functions $f_1 - f_{10}$ both 10-dimensional (10-D) and 30-dimensional (30-D) functions were tested. For function f_{11}, both 8-D and 28-D functions were tested. For the remaining functions $f_{12} - f_{22}$, the function dimension is shown in Table 2.4.

All experiments were run 25 times independently and statistical results are provided including the best, median and worst obtained results versus number of function evaluation numbers. The population sizes are set to be between 30, 50 and 100 based on the function dimension. All other control parameters are self-adaptive for the four algorithms.

In MATLAB 2014, the minimum and maximum double-precision values are from 2.22507E-308 to 1.79769E+308. Therefore, we put a tolerance of 1E-15. Beyond this value, the function's fitness is considered to be 0.0. In other words, the optimization run will be stopped, if the error, $\left| f^* - f_{\min} \right|$, becomes less than 1E-15; where f^* is function fitness and $f_{\min}$ is the global minimum of a function f .

CHAPTER 3

RESULTS AND CONCLUSION

3.1 Experimental Results

The results (best run, median run and worst run) of the comparison are provided in Table 3.1 for 10-dimensions functions, Table 3.2 for 30-dimensional functions and Table 3.3 for the remaining functions. The best "median" results are typed in bold and it means the minimum objection function error, $\left| f^* - f_{\min} \right|$, obtained within the 25 runs. While the "worst" result means the maximum error obtained. Note that for functions $f_1 - f_{11}$, f_{21} and f_{22} , the values listed in Table 3.1,Table 3.2 and Table 3.3 are the absolute difference between the obtained results (f_{best}, f_{median} or f_{worst}) and tolerance (1E-15). For the functions, $f_{12} - f_{20}$, the optimum values are not zeros. Therefore, the listed results (best, median and worst) for those functions in Table 3.3 are the absolute difference between the obtained results (f_{best}, f_{median} and f_{worst}) and actual optimum $f_{\min}$. Furthermore, in order to analyze the performance behavior of each algorithm, the convergence characteristics in terms of how fast the "median" of each algorithm reaches to the minimum value ($f_{\min}$) for 10-D unimodal functions $f_1 - f_5$, 10-D multimodal functions $f_6 - f_{11}$, 30-D unimodal functions $f_1 - f_5$, 30-D multimodal functions $f_6 - f_{11}$ and the remaining functions $f_{12} - f_{22}$ are shown in Figure 3.1 to Figure 3.5 respectively.

Table 3.1: Comparison between SaDE, SaDE-2, MSaDE-1 and MSaDE-2 on $f_1 - f_{11}$ over 25 runs for 10-D functions.

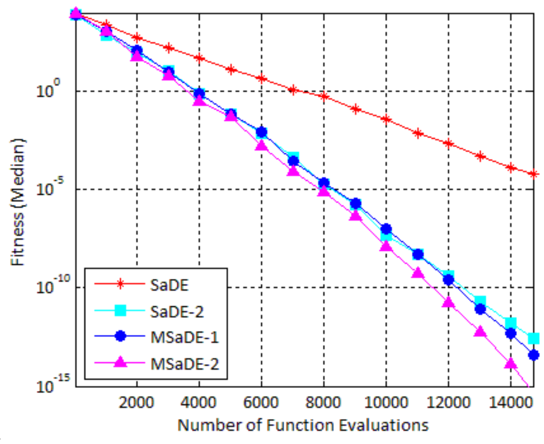
Fcn.	SaDE			SaDE-2			MSaDE-1			MSaDE-2		
	Best	Median	Worst	Best	Median	Worst	Best	Median	Worst	Best	Median	Worst
f_1	1.22E-5	5.96E-5	8.71E-5	3.87E-14	2.56E-13	1.68E-12	2.56E-15	3.69E-14	9.52E-14	0.00E0	**0.00E0**	8.52E-15
f_2	2.63E-7	5.42E-7	2.21E-6	0.00E0	6.02E-14	1.47E-13	0.00E0	**0.00E0**	9.40E-14	0.00E0	**0.00E0**	2.92E-15
f_3	3.79E-5	1.51E-4	5.05E-4	1.20E-12	3.81E-11	9.17E-11	0.00E0	5.42E-15	3.38E-14	0.00E0	**0.00E0**	7.76E-15
f_4	2.48E-2	5.44E-2	1.40E-1	1.10E-5	5.34E-5	1.69E-4	1.34E-6	5.56E-6	3.64E-5	6.31E-7	**3.19E-6**	1.79E-5
f_5	2.31E-1	7.18E-1	1.57E0	8.06E-15	1.56E-9	3.99E0	0.00E0	**0.00E0**	3.99E0	0.00E0	6.80E-13	4.04E0
f_6	9.66E-15	6.65E-14	2.12E-13	2.55E-15	**2.55E-15**	6.11E-15	2.55E-15	**2.55E-15**	6.11E-15	2.55E-15	**2.55E-15**	6.11E-15
f_7	1.25E-6	7.49E-4	8.19E-2	4.22E-15	2.22E-13	2.21E-2	1.22E-15	2.44E-15	1.72E-2	0.00E0	**0.00E0**	3.94E-2
f_8	3.89E-7	1.66E-6	7.35E-6	2.55E-15	1.01E-12	3.11E-1	0.00E0	**0.00E0**	2.98E-15	0.00E0	**0.00E0**	6.18E-14
f_9	2.63E-6	1.02E-5	4.04E-5	8.98E-14	1.49E-12	1.20E-11	0.00E0	**0.00E0**	3.23E-14	0.00E0	1.19E-15	1.37E-14
f_{10}	2.91E-7	1.17E-6	4.47E-5	0.00E0	**0.00E0**	6.11E-15	0.00E0	1.99E0	4.97E0	0.00E0	1.99E0	4.97E0
f_{11}	3.16E-10	2.82E-9	6.34E-8	1.28E-14	4.85E-11	5.30E-9	2.42E-15	**5.40E-12**	4.81E-7	0.00E0	7.17E-12	1.63E-8

Table 3.2: Comparison between SaDE, MSaDE, MSaDE.a and MSaDE.b on $f_1 - f_{11}$ over 25 runs for 30-D functions

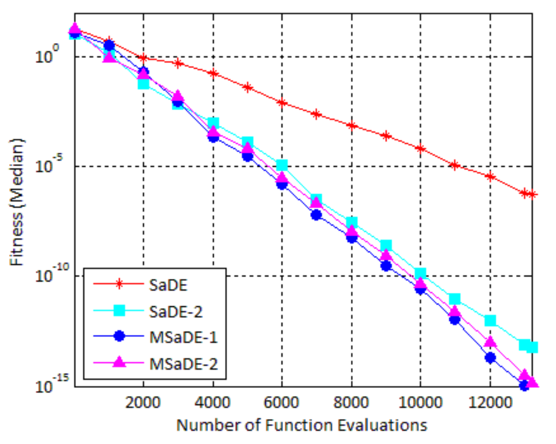
Fcn.	SaDE			SaDE-2			MSaDE-1			MSaDE-2		
	Best	Median	Worst	Best	Median	Worst	Best	Median	Worst	Best	Median	Worst
f_1	4.65E-8	3.02E-7	5.44E-7	0.00E0	2.15E-15	7.44E-14	0.00E0	1.30E-15	2.17E-14	0.00E0	**0.00E0**	7.12E-14
f_2	1.83E-8	3.73E-8	8.78E-8	2.28E-15	3.13E-14	5.19E-13	0.00E0	**0.00E0**	1.78E-13	0.00E0	1.08E-14	6.80E-14
f_3	8.25E-8	3.40E-7	6.52E-7	2.22E-15	1.54E-14	3.56E-13	0.00E0	**0.00E0**	1.40E-13	0.00E0	4.23E-15	3.71E-11
f_4	7.15E-2	1.15E-1	1.91E-1	1.26E-4	3.44E-4	3.29E-3	9.60E-5	**2.31E-4**	2.71E-3	7.87E-5	3.33E-4	6.71E-3
f_5	4.54E-7	1.43E1	1.65E1	2.53E-5	**7.10E0**	2.47E1	1.44E1	2.85E1	8.26E1	1.67E1	2.59E1	8.40E1
f_6	2.10E-7	4.79E-7	8.00E-7	2.06E-12	6.30E-12	1.16E0	3.76E-13	3.02E-12	9.31E-1	1.77E-13	**1.37E-12**	9.31E-1
f_7	5.96E-10	3.75E-9	1.55E-8	0.00E0	2.00E-15	1.48E-2	0.00E0	**0.00E0**	1.48E-2	0.00E0	**0.00E0**	1.48E-2
f_8	3.35E-9	1.23E-8	2.39E-8	0.00E0	2.53E-14	7.17E-14	0.00E0	**0.00E0**	1.99E-14	0.00E0	4.64E-14	1.36E-13
f_9	1.77E-9	3.20E-9	1.90E-8	9.08E-16	7.76E-15	1.04E-1	0.00E0	**0.00E0**	1.04E-1	0.00E0	**0.00E0**	1.04E-1
f_{10}	2.21E-12	**3.61E-11**	1.05E-8	0.00E0	9.95E-1	1.39E1	3.98E0	1.39E1	4.08E1	4.97E0	1.99E1	3.78E1
f_{11}	3.23E-4	1.49E-3	3.77E-3	1.31E-4	1.03E-3	3.56E-3	1.70E-4	**8.73E-4**	1.05E-2	1.21E-4	1.49E-3	2.22E-2

Table 3.3: Comparison between SaDE, MSaDE, MSaDE.a and MSaDE.b on $f_{12} - f_{22}$ over 25 runs for small Dimensions functions

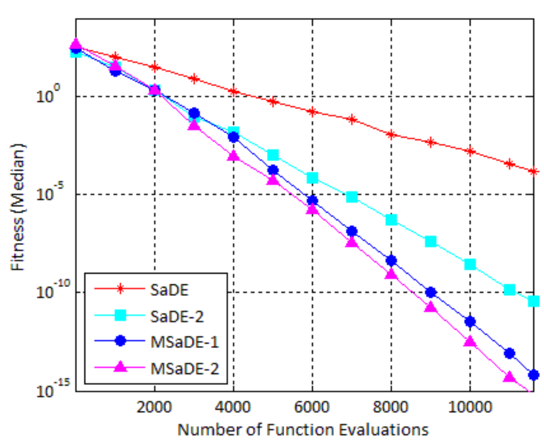
Fcn.	**SaDE**			**SaDE-2**			**MSaDE-1**			**MSaDE-2**		
	Best	Median	Worst	Best	Median	Worst	Best	Median	Worst	Best	Median	Worst
f_{12}	9.83E-11	2.66E-8	3.11E-4	0.00E0	**0.00E0**	0.00E0	0.00E0	**0.00E0**	1.33E-15	0.00E0	**0.00E0**	0.00E0
f_{13}	0.00E0	**0.00E0**	0.00E0	0.00E0	**0.00E0**	0.00E0	0.00E0	**0.00E0**	0.00E0	0.00E0	**0.00E0**	0.00E0
f_{14}	1.72E-8	2.26E-6	5.65E-5	0.00E0	**0.00E0**	0.00E0	0.00E0	**0.00E0**	0.00E0	0.00E0	**0.00E0**	0.00E0
f_{15}	2.60E-7	1.41E-4	1.01E-2	0.00E0	3.55E-15	5.28E-13	0.00E0	**0.00E0**	9.70E-13	0.00E0	**0.00E0**	1.42E-13
f_{16}	2.92E-10	6.53E-8	1.23E-6	4.00E-15	1.51E-14	3.46E-14	0.00E0	2.13E-14	2.82E-13	0.00E0	**7.11E-15**	2.80E-14
f_{17}	3.07E-6	1.03E-4	4.55E-4	3.43E-9	6.72E-8	1.35E-6	9.27E-11	**2.07E-8**	6.75E-7	0.00E0	3.70E-8	2.48E-6
f_{18}	4.61E-2	9.29E-2	3.46E-1	8.89E-5	6.85E-3	3.48E-1	6.98E-5	3.30E-3	3.91E-2	0.00E0	**1.22E-3**	8.86E-2
f_{19}	2.09E-5	2.49E-4	3.48E-3	0.00E0	1.41E-10	1.92E-1	0.00E0	**0.00E0**	1.92E-1	0.00E0	**0.00E0**	1.50E-1
f_{20}	3.74E-3	1.61E-1	1.90E0	0.00E0	**0.00E0**	0.00E0	0.00E0	**0.00E0**	0.00E0	0.00E0	**0.00E0**	0.00E0
f_{21}	4.61E-4	5.32E-3	1.95E-2	6.47E-11	4.48E-6	4.78E-5	4.01E-10	**3.46E-6**	5.50E-5	1.92E-8	4.48E-6	1.87E-4
f_{22}	4.80E-3	4.94E-3	1.54E-2	0.00E0	1.85E-3	4.72E-1	0.00E0	**1.15E-3**	1.31E-1	0.00E0	**1.15E-3**	4.94E-3



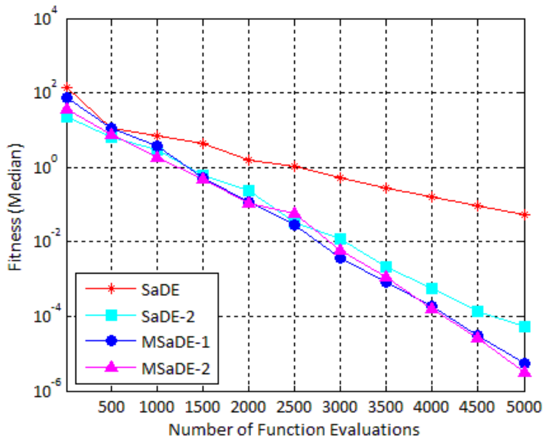
(a) f_1



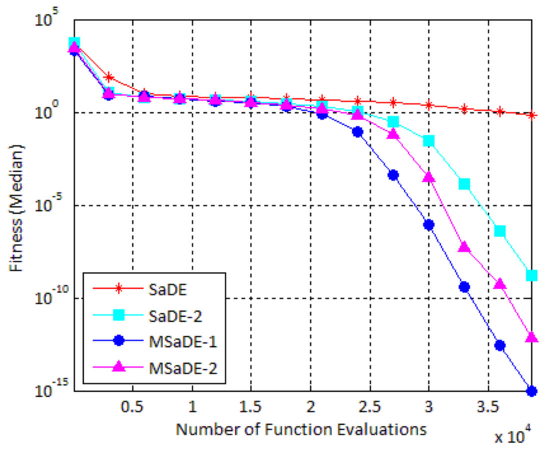
(b) f_2



(c) f_3

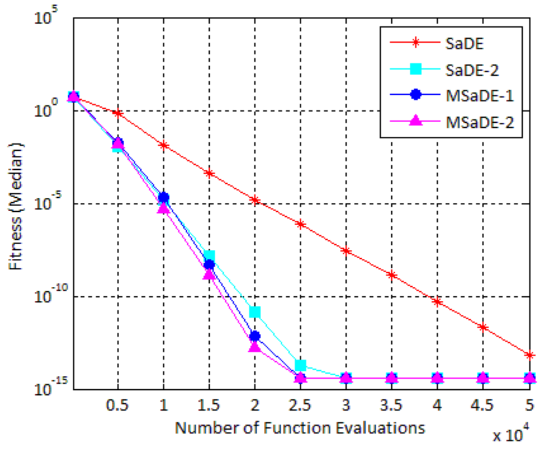


(d) f_4

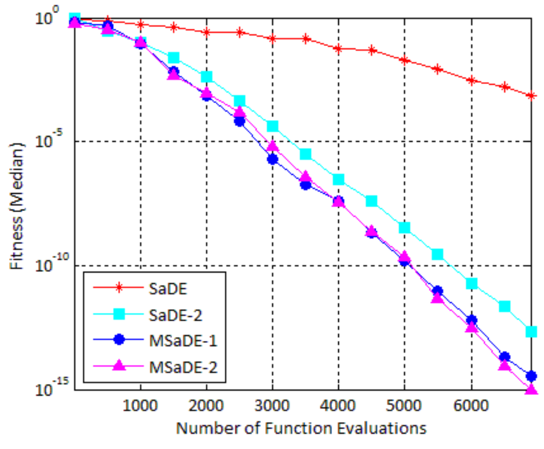


(e) f_5

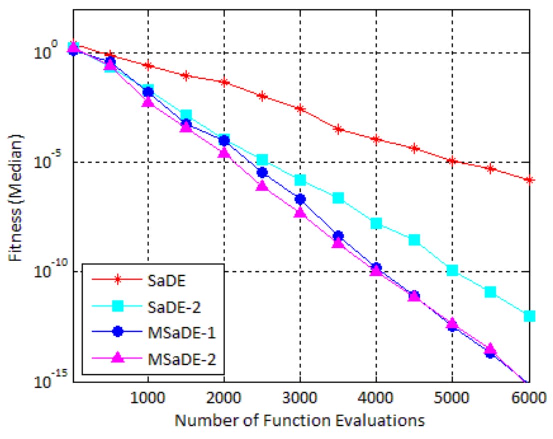
Figure 3.1: Optimization performance (median curves) of SaDE, SaDE-2, MSaDE-1 and MSaDE-2 on uni-modal 10-D functions



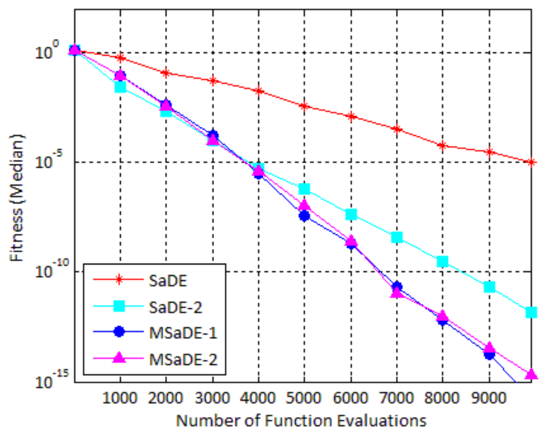
(a) f_6



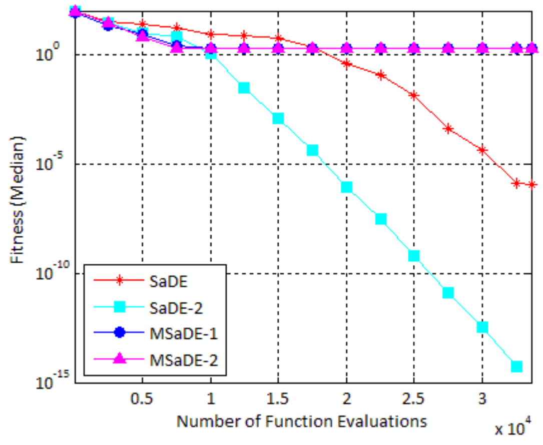
(b) f_7



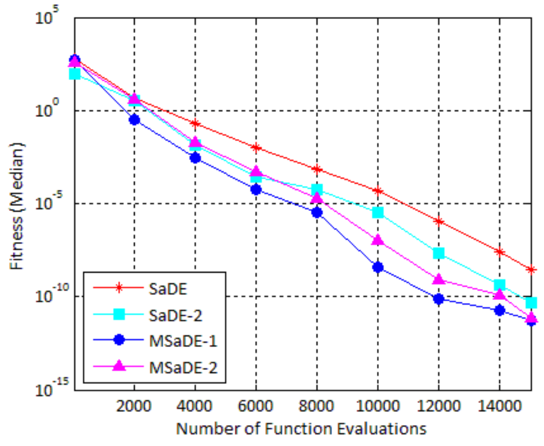
(c) f_8



(d) f_9

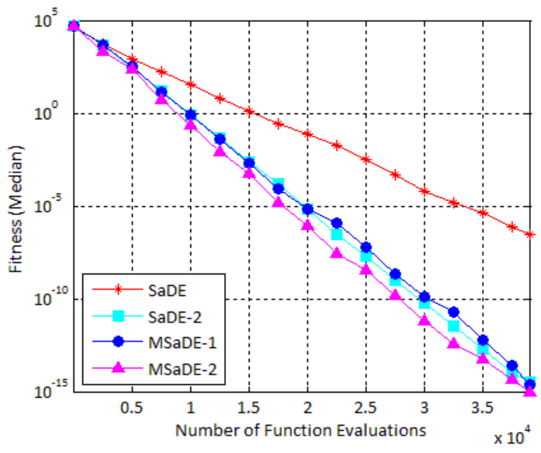


(e) f_{10}

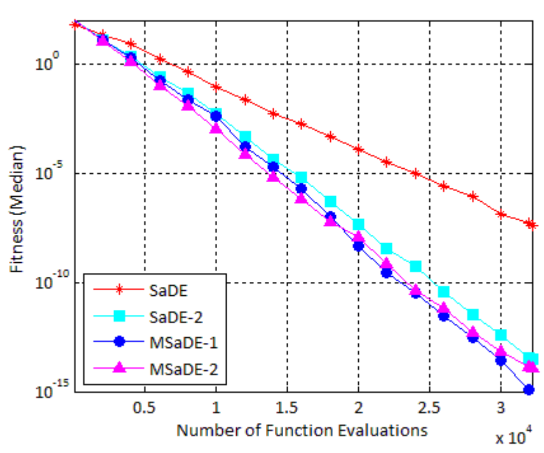


(f) f_{11}

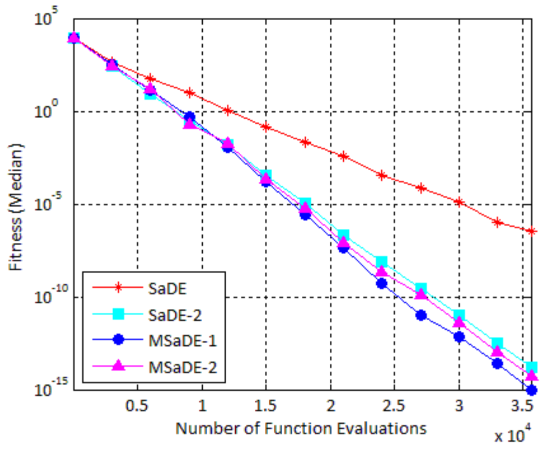
Figure 3.2: Optimization performance (median curves) of SaDE, SaDE-2, MSaDE-1 and MSaDE-2 on 10-D multi-modal functions



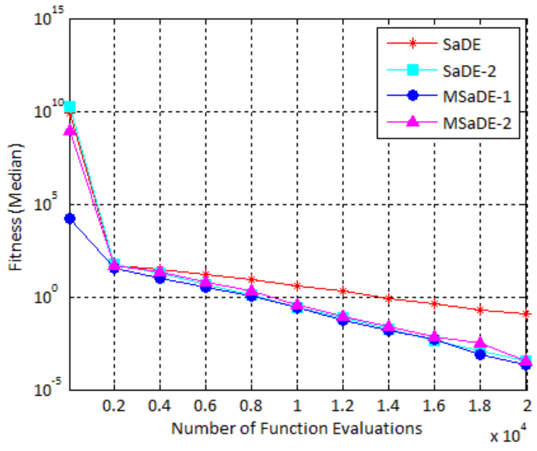
(a) f_1



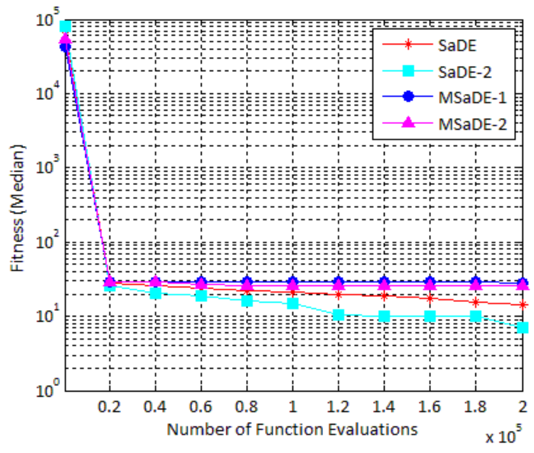
(b) f_2



(c) f_3

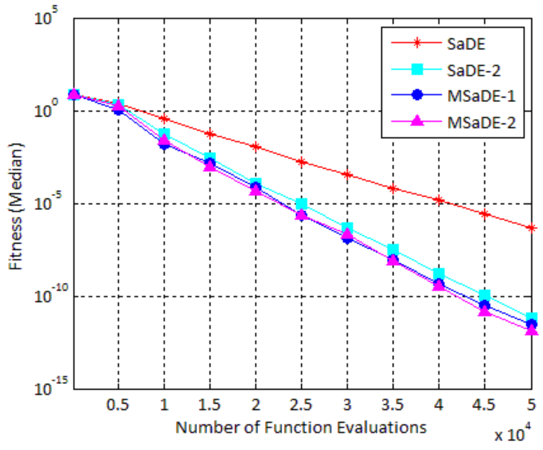


(d) f_4

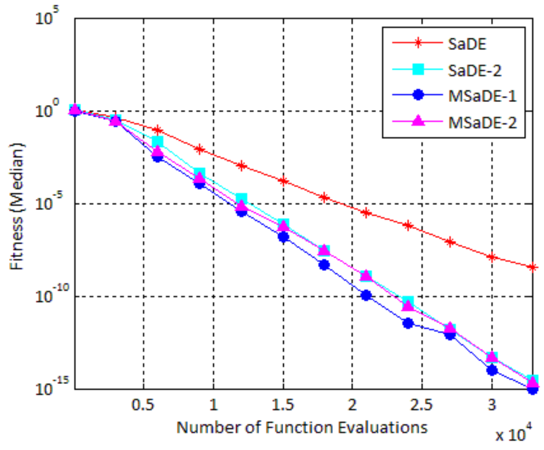


(e) f_5

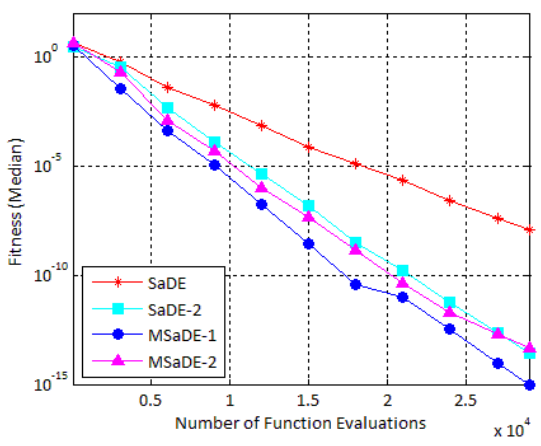
Figure 3.3: Optimization performance (median curves) of SaDE, SaDE-2, MSaDE-1 and MSaDE-2 on 30-D unimodal functions



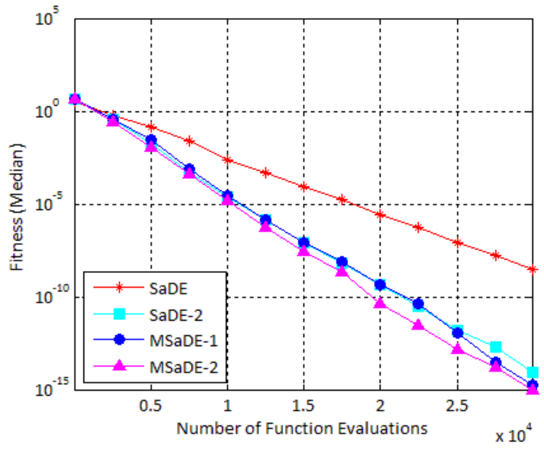
(a) f_6



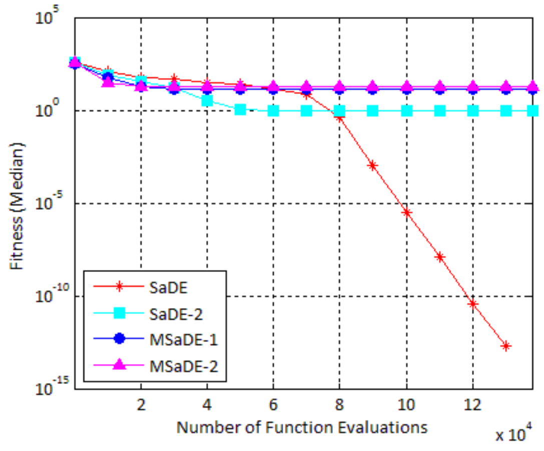
(b) f_7



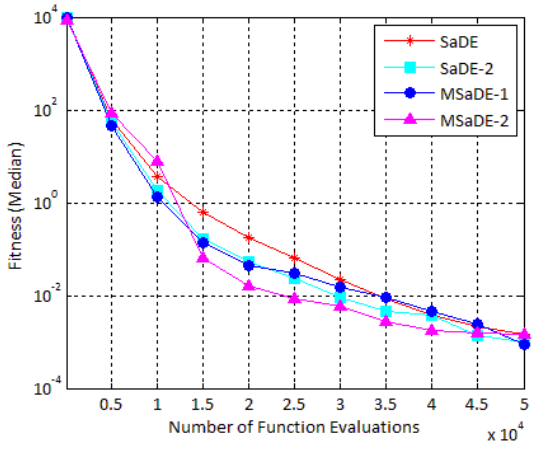
(c) f_8



(d) f_9

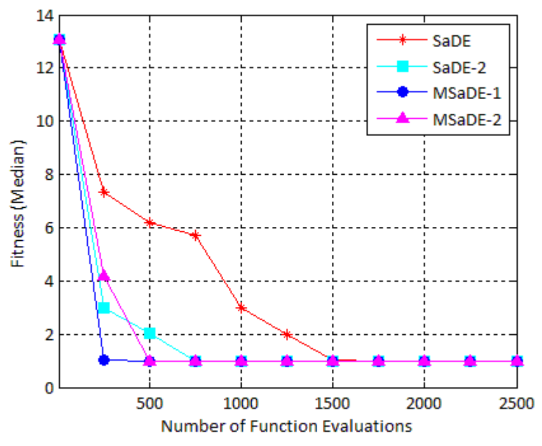


(e) f_{10}

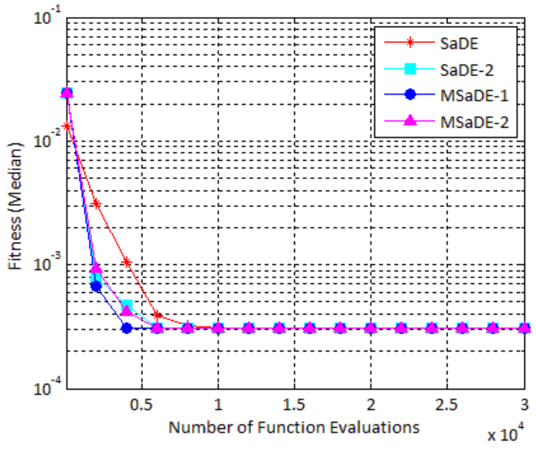


(f) f_{11}

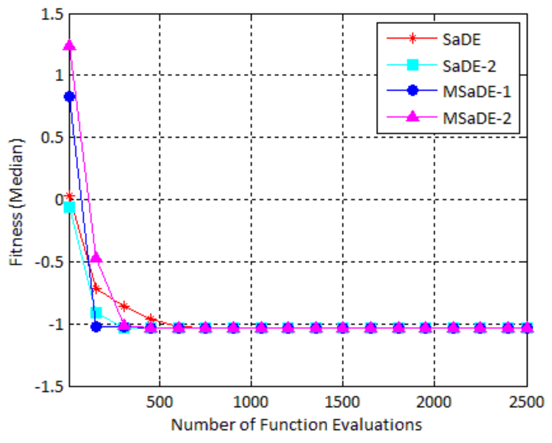
Figure 3.4: Optimization performance (median curves) of SaDE, SaDE-2, MSaDE-1 and MSaDE-2 on 30-D multimodal functions



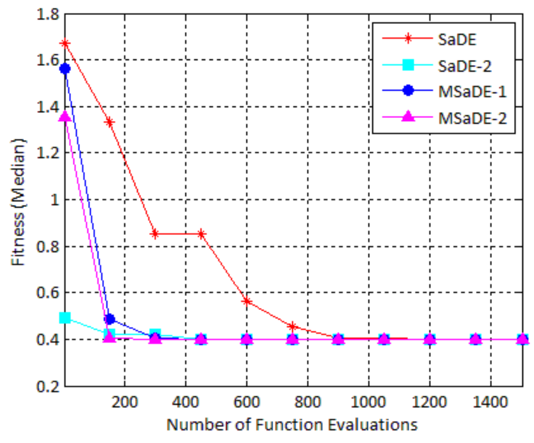
(a) f_{12}



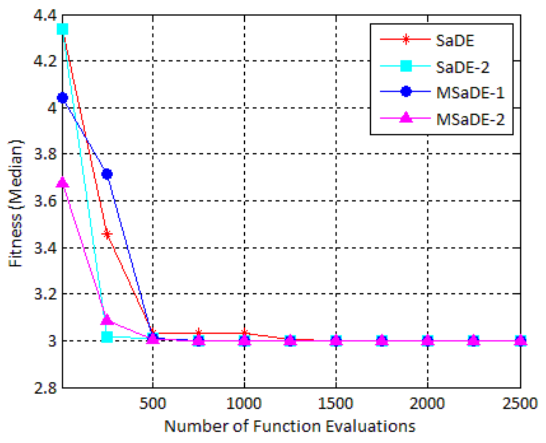
(b) f_{13}



(c) f_{14}

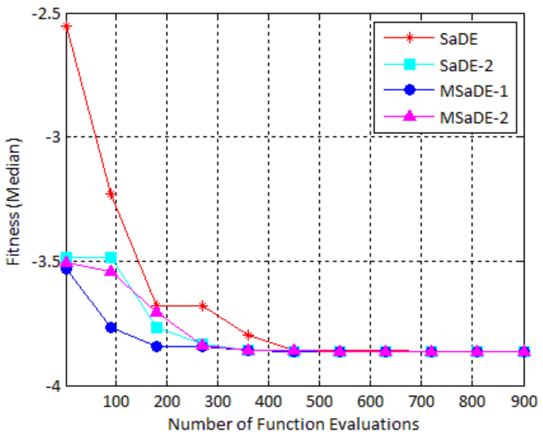


(d) f_{15}

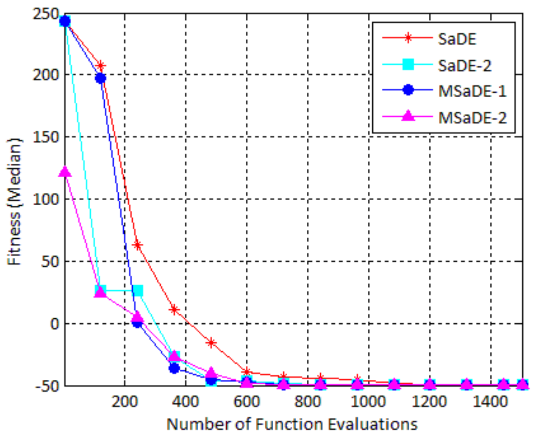


(e) f_{16}

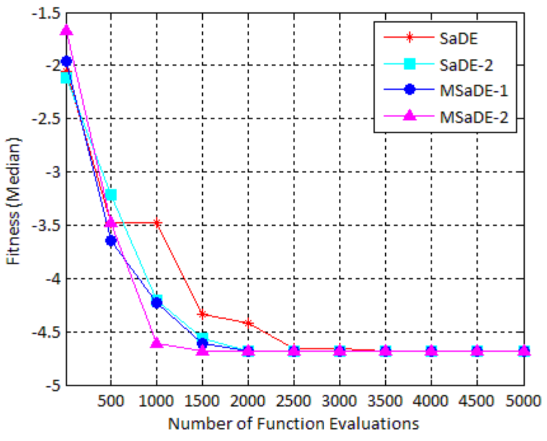
Figure 3.5: Optimization performance (median curves) of SaDE, SaDE-2, MSaDE-1 and MSaDE-2 on functions $f_{12} - f_{22}$



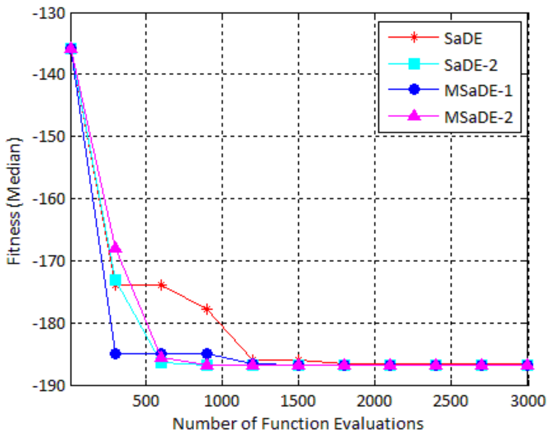
(f) f_{17}



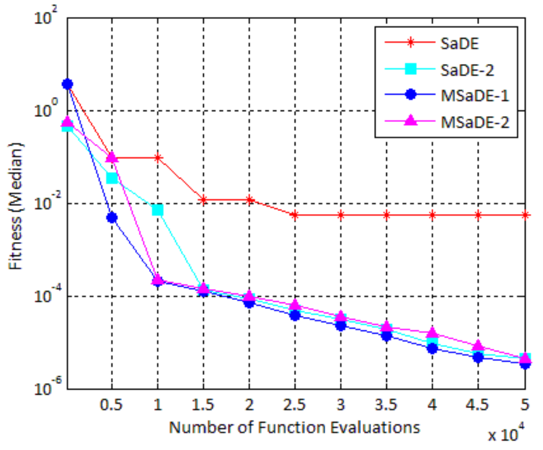
(g) f_{18}



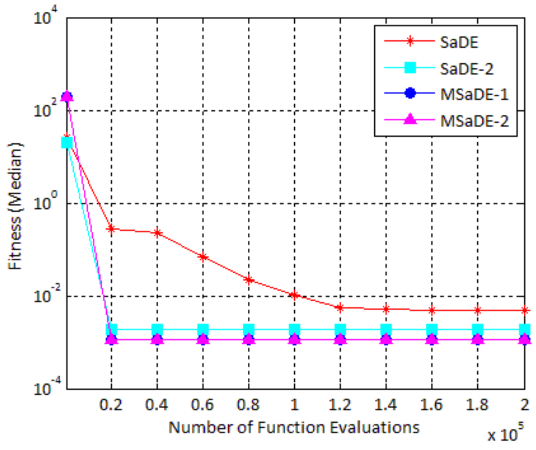
(h) f_{19}



(i) f_{20}



(j) f_{21}



(k) f_{22}

Figure 3.5 **(*continued*)**

3.1.1 10-D Unimodal Functions $(f_1 - f_5)$

From the results shown in Table 3.1, it can be seen that, generally, MSaDE outperforms SaDE for all 10-D unimodal functions, $f_1 - f_5$. Additionally, it can be observed that MSaDE-1 and MSaDE-2 algorithms are almost the same and they approximately achieved the same results. However, MSaDE-2 exhibit better results in this case.

As shown in Figure 3.1.e, changing and downsizing the candidate pool of trial-vector generation strategies (as in SaDE-2) result in significant improvement in solution quality and convergence speed. Furthermore, MSaDE algorithm exhibit better optimization performance than SaDE and SaDE-2 due to the implementation of new adaption technique.

3.1.2 10-D Multimodal Functions $(f_6 - f_{11})$

Table 3.1 shows that MSaDE is better than SaDE for all functions, $f_6 - f_{11}$, except for f_{10} where SaDE-2 shows the best "median". However, as shown in Figure 3.6, it can be observed that the "best" results for MSaDE-1 and MSaDE-2 converge faster than SaDE.

It can also be shown in Figure 3.2 that big improvement is achieved by adding the strategy "*DE/Best/1/bin*" to the candidate pool and reducing the number of strategies to two instead of four, which is represented by SaDE-2. Then, additional improvement is clearly achieved when applying MSaDE-1 and MSaDE-2 algorithms.

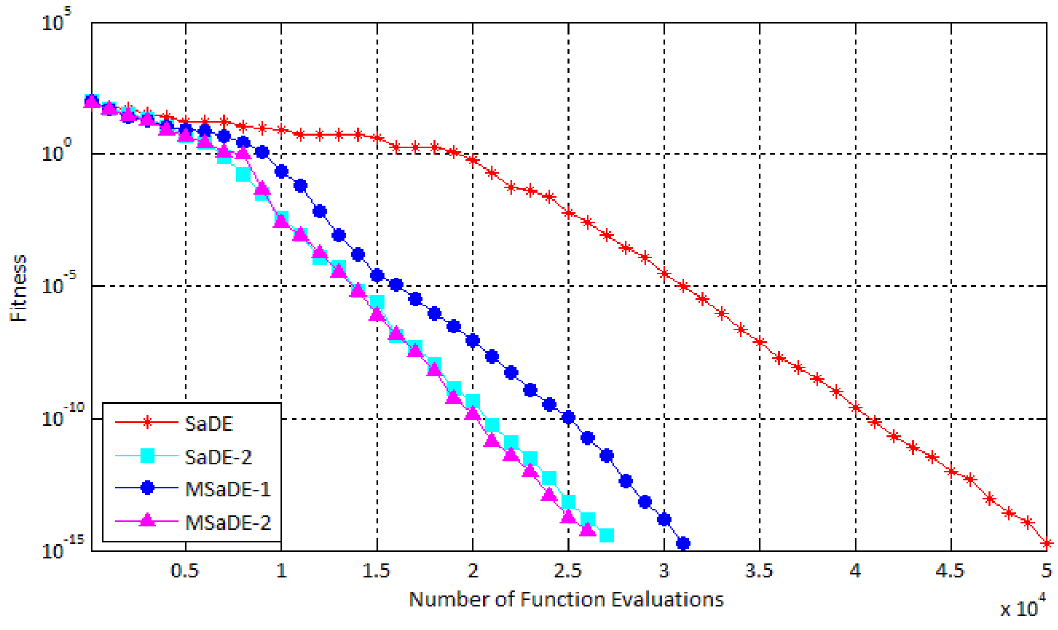


Figure 3.6: Optimization performance (best curves) of SaDE, SaDE-2, MSaDE-1 and MSaDE-2 on 10-D multimodal function, f_{10}.

3.1.3 30-D Unimodal Functions $(f_1 - f_5)$

As shown in Table 3.2, MSaDE exhibits better optimization performance than SaDE algorithm for all functions ($f_1 - f_5$) except for function f_5 where SaDE-2 shows the best "median" result. However, when applying the new adaption technique from Eq. (2.20) and combination technique Eq. (2.24) with keeping the candidate pool of trial-vector generation strategies as same as original SaDE, performance improvement in MSaDE is observed. MSaDE-1B and MSaDE-2B refer to the Modified Self-adaptive DE algorithms in which candidate pool of trial-vector generation strategies are the same as in the original SaDE, while applying the new adaption and combination techniques, respectively. As shown in Figure 3.7, the proposed methods are, significantly more effective than the original SaDE in terms of solution quality.

In contrary to what has been shown in 10-D unimodal functions $\left(f_1 - f_5\right)$, MSaDE-1 performs better than MSaDE-2 in 30-D unimodal functions.

3.1.4 30-D Multimodal Functions $(f_6 - f_{11})$

With reference to Table 3.2, the results show that MSaDE is surpassed by SaDE algorithm on function f_{10} only. However, MSaDE algorithm optimization performance is superior in all other functions $f_6 - f_{11}$. In addition, as shown in Figure 3.8, when applying MSaDE-1B and MSaDE-2B to that function f_{10}, they perform better and reach to the optimum converge faster than the original SaDE. In additions, MSaDE-1 performs better than MSaDE-2 in this case.

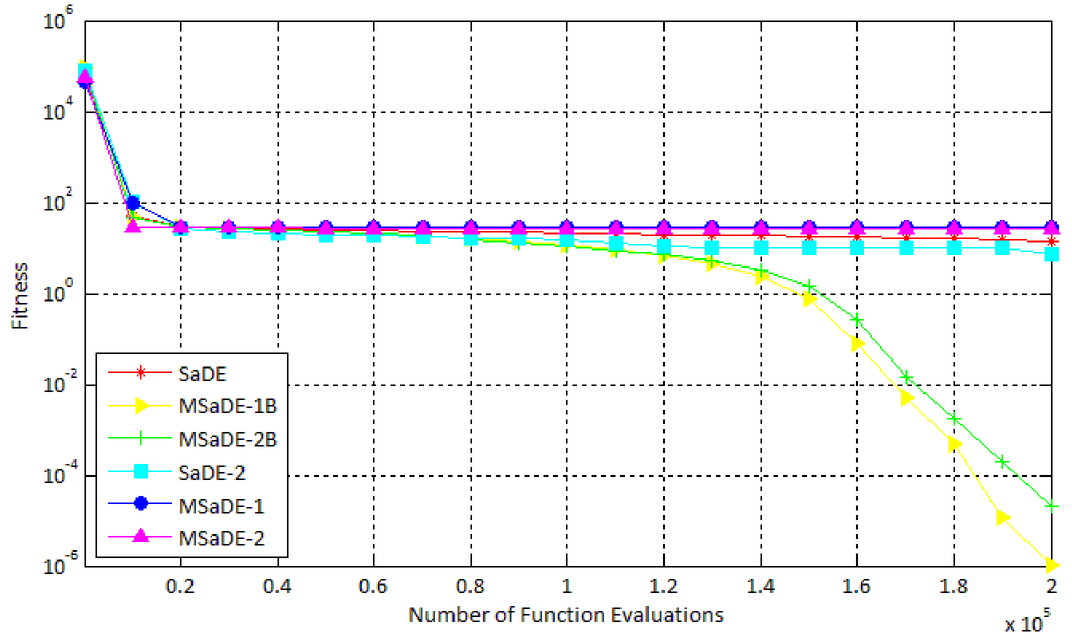


Figure 3.7: Optimization performance (median curves) of SaDE, MSaDE-1B, MSaDE-2B, SaDE-2, MSaDE-1 and MSaDE-2 on 30-D unimodal function f_5

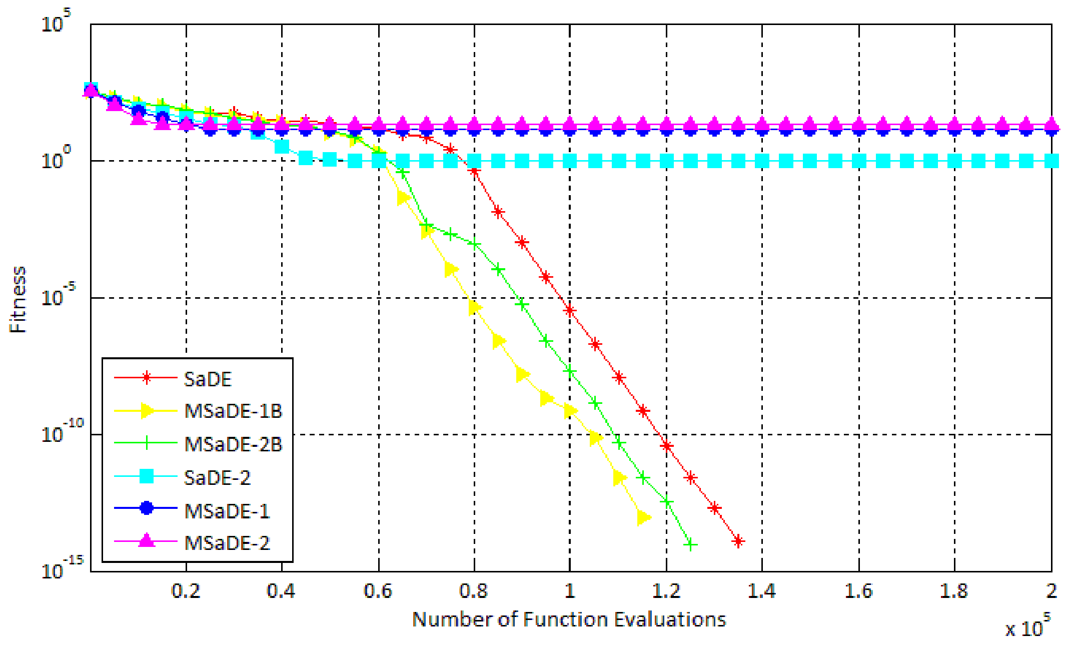


Figure 3.8: Optimization performance (median curves) of SaDE, MSaDE-1B, MSaDE-2B, SaDE-2, MSaDE-1 and MSaDE-2 on 30-D multimodal function f_{10}

3.1.5 Remaining Functions $(f_{12} - f_{22})$

Table 3.3 shows the comparison between SaDE, SaDE-2, MSaDE-1 and MSaDE-2 on $f_{11} - f_{22}$ over 25 runs for small dimensions functions. It is obviously shown that MSaDE is superior to the SaDE algorithm in all functions in terms of best “median” results.
In function f_{13}, all algorithms converged to the optimum solution. In functions f_{12}, f_{14} and f_{20} SaDE-2, MSaDE-1 and MSaDE-2 reach to the optimum solution, while original SaDE did not, even the best results obtained from SaDE did not reach the optimum solution.

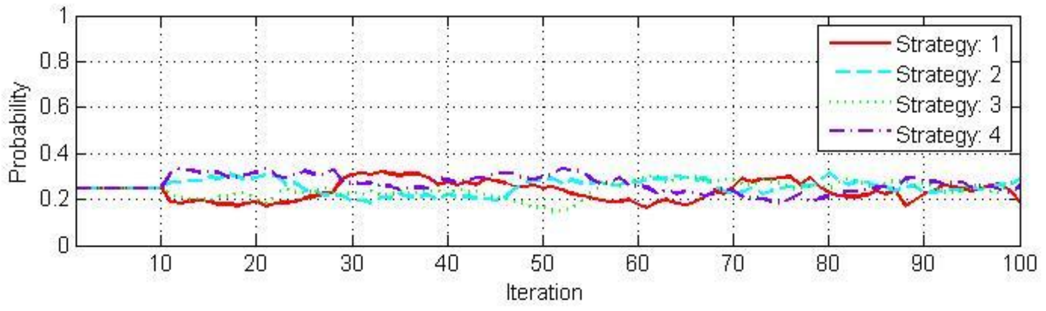
Generally, it is clear that MSaDE outperformed SaDE algorithms; MSaDE performed better than SaDE in 82% of all the 22 functions considered. As shown in Table 3.4, MSaDE-1 is the best in optimizing 30-D functions while MSaDE-2 performs a little better than MSaDE-1 in optimizing 10-D functions. However, MSaDE-1 is the second best in this case. For functions with small dimensions (<10-D), both MSaDE-1 and MSaDE-2 have the same strength to find the optimal solution for the functions in this case.

Table 3.4: Performance comparison summary of SaDE, SaDE-2, MSaDE-1 and MSaDE-2 over 25 independent runs for all functions

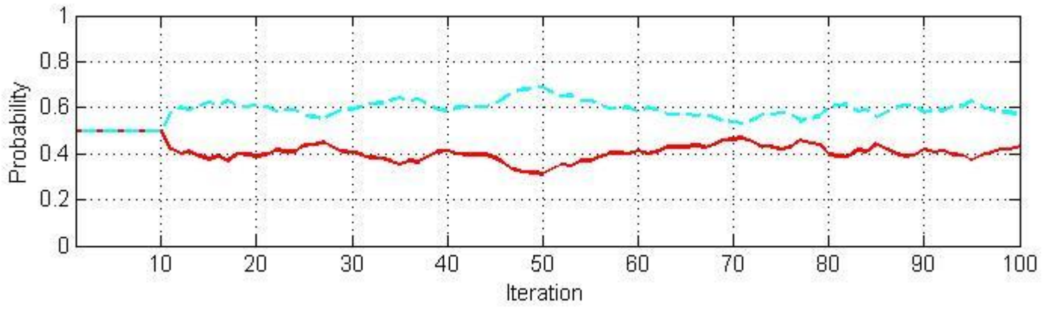
Functions	Success Percentage Over all Benchmark Test Functions, $f_1 - f_{22}$, (%)			
	SaDE	SaDE-2	MSaDE-1	MSaDE-2
10-D Functions	0%	13%	40%	47%
30-D Functions	8%	8%	58%	25%
<10-D Function	4%	16%	40%	40%
Total	4%	14%	**44%**	38%

Therefore, we conclude that the proposed algorithm, Modified Self-adaptive Differential Evolution MSaDE, with both versions (MSaDE-1 and MSaDE-2) exhibits better performance than the original SaDE algorithm. In addition, the completely new adaption technique, based on Eq. (2.20), is more effective than the combination technique, Eq. (2.24) . The reason for this improvement can be seen in Figure 3.9, which shows the success probability progression vs algorithm generations. It is shown in Figure 3.9a that in SaDE algorithm, no single strategy has dominated the trial-vector generation techniques, which means that all strategies in the candidate pool almost has the same chance to be selected in all generations. Besides, almost the same number of population vectors is assigned to each strategy. Therefore, the algorithm loses its ability to find out the best technique to generate new trial vectors and still use all strategies available in candidate pool with almost the same probabilities. This means that the optimization algorithm needs relatively more computational time to reach optimum solution.

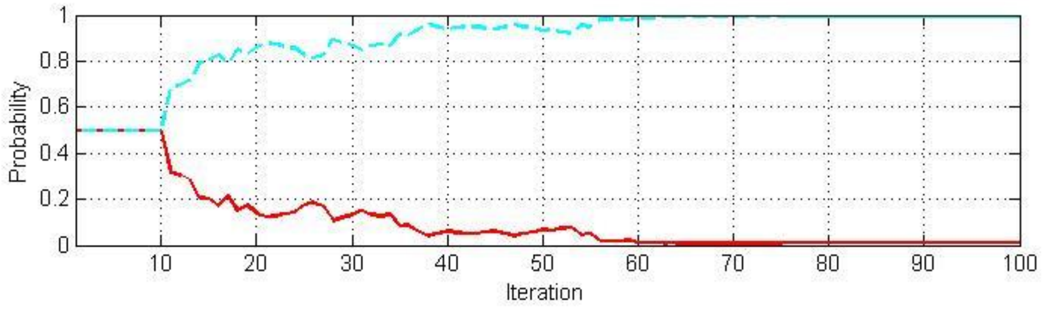
In SaDE-2 algorithm; as shown in Figure 3.9b, after the learning period which is 10 generations in this case, one of the strategies has a success probability of more than 60% which means that it possesses more population vectors to be mutated. In other words, by downsizing the candidate pool size from four strategies to only two, and having the “good” strategies that have the capabilities of exploration and exploitation, the algorithm is able to self-adapt to one trial-vector generation strategy than the other and an improvement is obtained.



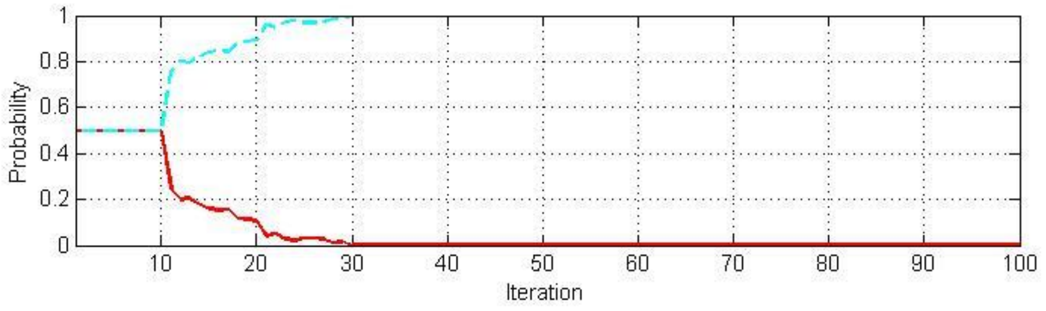
(a) SaDE Algorithm



(b) SaDE-2 Algorithm



(c) MSaDE-1 Algorithm



(d) MSaDE-2 Algorithm

Figure 3.9: Success probabilities progression of strategies with algorithm generation

As shown in Figure 3.9c, after 60 generations, MSaDE-1 is able to find out the most compatible strategy with optimization problem and be 100% self-adapted. This exhibits higher solution quality in less computational time. With reference to Eq. (2.20), it can be seen that depending on the quality of improvement (WF), that is taken place by k^{th} strategy, is more efficient than calculating the success rate based on the number of vectors entering to the next generations in learning period as expressed in Eq. (2.18). Therefore, after 60 generations, all population vectors were assigned to the most successful trial-vector generation strategy which enabled the algorithm to rapidly be self-adapted and reach the optimal solution.

In MSaDE-2 algorithm, as shown in Figure 3.9d, after only 30 generations, 100% of population vectors were assigned to the successful strategy. In this algorithm the success probability of a strategy is calculated by Eq. (2.24), which is a combination between the old and new adaption techniques. As shown this combination technique accelerates the process of adaption which is very effective in optimizing relatively less complex functions such as the 10-D functions. However, in 30-D functions which are more complex, MSaDE-1 performs better. Because in complex optimization problems, in order to avoid stagnation and/or premature convergence to local optima, a convergence speed-balance is required to find out the most successful trial-vector generation strategy within candidate pool before assigning all population vectors to it. This required-balance is achieved by MSaDE-1 algorithm which is able to be 100% self-adapted to the successful strategy after satisfactory number of generations to preserve the exploration and exploitation capabilities.

3.2 Conclusion

In order to enhance the convergence speed and exploitation capability with maintaining a satisfactory exploration level of SaDE algorithm, a Modified Self-adaptive Differential Evolution MSaDE algorithm with novel technique in determining the success rate of trial-vector generation strategy to solve global numerical optimization problems over continues space is proposed. The proposed algorithm introduces a new adaptation technique to determine the success rate of each trial-vector generation strategy based on the quality of improvement in solution toward optimum solution that each strategy achieves, not on the successful number of entries to following generations as applied in original SaDE. The proposed method is shown to enhance the convergence speed and balance the exploitation and global exploration capabilities of self-adaptive DE algorithms. The proposed MSaDE algorithm has been compared with original SaDE over a suite of 22 numerical optimization problems. The experimental results and comparison showed that the MSaDE algorithm performs better than original SaDE algorithm in 82% of optimization problems with different types, complexity and dimensionality. It performs better in terms of final solution quality, convergence speed and robustness. Finally, it would be very interesting to investigate the performance of proposed MSaDE to solve practical engineering optimization problems and real-world applications.

CHAPTER 4

Optimization of ES-SAGD Application: Comparative Analysis of Optimization Techniques

4.1 Literature Review on ES-SAGD Optimization

Butler et al., in 1981, proposed the concept of the Steam-Assisted Gravity Drainage (SAGD) recovery method; where two horizontal wells with vertical distance are placed near to the bottom of the formation. The steam is continuously injected from the upper well and heavy oil produced from the lower well (Butler and Stephens 1981).

The concept of Expanded Solvent Steam-Assisted Gravity Drainage (ES-SAGD) was introduced by Nasr et al., in 2003, with detailed laboratory test results which showed that the highest recovery performance was achieved when the vaporization temperature of the added solvent (Hydrocarbon) is closer to the temperature of the injected steam (Nasr et al. 2003). Then, Das studied the dispersion and diffusion of solvent in VAPEX method. He introduced the results of simulation study to investigate the effect of solvent components inside vapor chamber (Das 2005).

Generally, it is expected that adding solvent components improves the process.

Boak and Palmgren presented a numerical analysis for a naphtha co-injection test during SAGD for the MacKay River McMurray formation (Boak and Palmgren 2007). The effects of co-injecting a multi-component solvent, naphtha, and a single component solvent, propane or pentane were investigated. Co-injection of any of the solvents studied (propane, pentane and naphtha) resulted in an improved Steam Oil Ratio (SOR). Only naphtha co-injection resulted in an improved oil production rate because the components of naphtha travelled freely in the vapor chamber and accumulated along the vapor chamber front in both the vapor and oil phases.

In 2008, Ivory et al. studied the low pressure ES-SAGD performance through lab experiments and numerical simulations. They found that the effects of minimum production pressure, sub-cool and solvent concentration must be considered simultaneously as they impact each other. Sensitivity runs on minimum BHP resulted that a lower producer BHP yielded a higher oil rate with less SOR (Ivory et al. 2008).

Govind et al. performed detailed simulation studies on ES-SAGD. They stated that the effective variables that control the performance of the ES-SAGD process are the solvent type, concentration, operating pressure and the injection strategy. The results of sensitivity studies performed on the solvent selection, dilation effect and operating condition were presented with conclusions and recommendations for an operating strategy. They also indicated the dilation is an important factor in the high pressure injection ES-SAGD process (Govind et al. 2008).

In 2010, Ayodele et al. implemented experiment and history-matched simulation results of 2D scaled laboratory tests of ES-SAGD with hexane as the co-injected solvent. The comparison of ES-SAGD and SAGD experiments shows that ES-SAGD using hexane performed better than an equivalent SAGD experiment (Ayodele et al. 2010).

Kumar et al. investigated the impact of geological heterogeneity on SAGD wellbore design and the optimization of the length and positioning of multiple tubular strings as well as the allocation of injected steam among multiple tubing strings (Kumar et al. 2010).

In 2012, Gates and Chakrabarty used the Simulated Annealing, SA method to find the optimal solvent concentration that minimize the cost function which is cSOR. The results showed that ES-SAGD can yield lower cSOR than SAGD and that the optimized ES_SAGD operating strategy used ½ of the steam per unit volume of produced oil when compared with SAGD method. In their study, they assumed that the solvent recovery predicted from the simulations was about 90% which is in reasonable agreement with existing thermal-solvent field pilots, therefore, they assumed that there is no losses in the injected solvent (Gates and Chakrabarty 2012).

4.1.1 DE Application for Optimizing Oilfield-related Problems

Despite the successful applications of differential evolution in many engineering fields, there is a limited number of publications related to the applications of DE algorithm for tackling petroleum engineering problems.

Jahangiri (Jahangiri 2007) applied differential evolution to optimize smart well operations to maximize oil production. Hajizadeh (Hajizadeh et al. 2009) used DE method to history match production data in a black oil reservoir model. Other works include estimation of oil and water relative permeabilities to match core flood data (Wang and Buckley 2006), Estimation of geostatistics variogram parameters frameworks (Zhang et al. 2009) waveform inversion of cross-well data using differential evolution (Wang et al. 2011).

Recently in 2013, Nghiem et al. applied the DE technique to a SAGD case study to history match saturation and temperature profiles in addition to cumulative oil, water production and cSOR. The results showed good history matching, which allowed the assessment of uncertainty for the forecast stage. The match-quality was compared with the Particle Swarm Optimization method (PSO). The comparison showed that DE offers much better solutions with much lower numbers of simulation runs (Nghiem et al. 2013).

4.2 Research Optimization Framework

In this research, we have constructed a numerical flow simulation model of one of Athabasca heavy oil reservoirs using CMG STARS, a numerical flow simulation package for thermal recovery process. Then we have developed a framework that integrates previously mentioned optimization techniques (SaDE, SaDE-2, MSaDE-1 and MSaDE-2), in addition to other two well-known optimization algorithms (DE and PSO) with CMG STARS, to optimize ES-SAGD recovery process. Before that, a sensitivity study is done on ES-SAGD to determine a preliminary ranking of the control operational parameters according to their effect on the project’s NPV. In addition, this study will figure out the realistic effective range (upper and lower limits) that would be assigned to the selected operational parameters in the optimization step.

Subsequently, a comparison study is done to figure out the most-effective optimization algorithm that maximized the NPV of the project while considering other performance indicators like Oil Recovery Factor (RF) and cumulative Steam Oil Ratio (cSOR).

Finally, for convenience, a performance analysis of optimized SAGD and ES-SAGD recovery processes is performed.

4.3 Net Present Value

NPV was used as the performance indicator in this study. The NPV (Khan 1993) of a project is defined as the sum of the present values of individual cash flows of that project; where the cash flow is positive for revenue and negative for expenditure. In ES-SAGD project, the capital cost at the project's beginning consists of the exploration cost, the drilling and well completion cost, steam generators capital cost, water treatment capital cost, and solvent injection capital cost. The recurrent expenditure includes the cost of steam generation, steam injection, produced water treatment, solvent handling and recompression, solvent cost and operating costs including well remediation and human resources. Such costs are discounted to the beginning time of the project. Revenue are in the form of heavy-oil sales, which is also discounted to the present time. All these cash flows are combined to give the NPV of the project and defined mathematically (Onwunalu and Durlofsky. 2010), as shown in Eq. (4.1):

$$NPV = \sum_{n=1}^{N} \frac{CF_n}{(1+r)^n} - C_{cap}, \tag{4.1}$$

Where N is the total number of discounting periods (total number of years in this study), n is the year index, r is the annual discount rate, C_{cap} is the capital cost and is given by:

$$C_{cap} = C_f + C_{ex} + N_w C_{w_{therm}} + C_{SG} + C_{So} \tag{4.2}$$

and CF_n is the cash flow rate in year is given by:

$$CF_n = R_n - E_n, \tag{4.3}$$

where R_n is the total revenue for year (n), given by:

$$R_n = P_n^o Q_n^{o_{pro}} \tag{4.4}$$

and E_n is the expenditure for year (n), given by:

$$E_n = C_n^{W_{pro}} Q_n^{W_{pro}} + C_n^{S_{inj}} Q_n^{S_{inj}} + N_w C_n^{So_h} + C_n^{So_{rec}} Q_n^{So_{pro}} + C_n^{op} Q_n^{O_{pro}} \tag{4.5}$$

All the parameters used in Eqs. (4.2) to (4.5) are explained, and their estimations are shown in Table 4.1; the costs of facility installation, steam injection, treating produced water and other operating costs are estimated with reference to (Azad et al. 2013). The costs of exploration, well drilling and completion, steam generation facility, water treatment facility, solving injection facility, and solvent's cost, handling and recompression are based on (Frauenfeld et al. 2009). In this study, the heavy oil price is estimated to be less than the light oil price by 25% as per BAYTEX – Alberta, 2013 Heavy Oil Pricing Reports. The abbreviations listed in Eqs. (4.1) to (4.5) are listed in Table 4.1.

Table 4.1: Abbreviation list for Eqs. (4.1) to (4.5)

C_{ex}	: Exploration cost	$Q_n^{O_{pro}}$	: Total oil production in year n
$C_{w_{therm}}$	: Cost of thermal well	$Q_n^{S_{inj}}$	: Total water injected in year n
$C_{w_{cold}}$	: Cost of non-thermal well	$Q_n^{So_{inj}}$	: Total solvent injected in year n
C_n^{op}	: Operating cost per bbl oil	$Q_n^{W_{pro}}$	: Total water produced in year n
$C_n^{So_{inj}}$	: Cost of solvent injection per ft3	$Q_n^{So_{pro}}$	: Total solvent produced in year n
$C_n^{So_h}$	: Cost of solvent handling per well		
$C_n^{S_{inj}}$	: Cost of steam injection per bbl water	P_n^o	: Oil price, $ ¥¥
$C_n^{W_{pro}}$	: Cost of treating produced water per bbl	N_w	: Number of wells
$C_n^{So_{rec}}$	: Cost of vapor solvent recompression per ft3	n	: Year index
C_{SG}	: Capital cost of Steam Generators	r	: Annual discount rate
C_f	: Capital cost of facility installation cost	N	: Total number of years
C_{So}	: Capital cost of Solvent injection facility		

4.4 Model Description

Steam, Thermal and Advanced Processes Reservoir Simulator (STARS) from CMG is used in this work to simulate reservoir model of ES-SAGD. A homogeneous 2D Cartesian model with one pair of horizontal injection and production wells, with 51 grid blocks along X-axis and 30 grid blocks along Z-axis is used, with a gas cap layer with a thickness of 10 ft, gas saturation of 85%, initial water saturation of 15% and gas cap pressure of 145 psi. The common reservoir and fluid parameters (Gates and Chakrabarty 2005) are shown in Table 4.2. Three phase relative permeability model with end-points values are used to generate the oil-water and gas-oil relative permeability curves as shown in Figure 10. ES-SAGD model is simulated for 10 years where a sensitivity study was done on the ES-SAGD recovery processes to determine the optimum operating parameters which result in the maximum NPV at the end of the project.

Table 4.2: Simulator input parameters

Input Parameter	Value
Reference Depth, ft	900
Reservoir Initial Pressure @ 900 ft, psi	145
Reservoir Temperature, °F	52
Porosity	38%
Average Horizontal Permeability, mD	7000
Average Vertical Permeability, mD	3000
Rock Heat Capacity, Btu/ft^3 F	417
Rock Thermal Conductivity, Btu/(ft day F)	106
Over/Underburden Heat Capacity, Btu/ft^3 F	417
Over/Underburden Thermal Conductivity, Btu/(ft day F)	106
Bitumen Thermal Conductivity, Btu/(ft day F)	1.85
Bitumen Viscosity Correlation	A = 22.8515
$\ln\ln\mu(cp) = \mathrm{A} + \mathrm{B}\ln\mathrm{T}(k)$	B = -3.5784

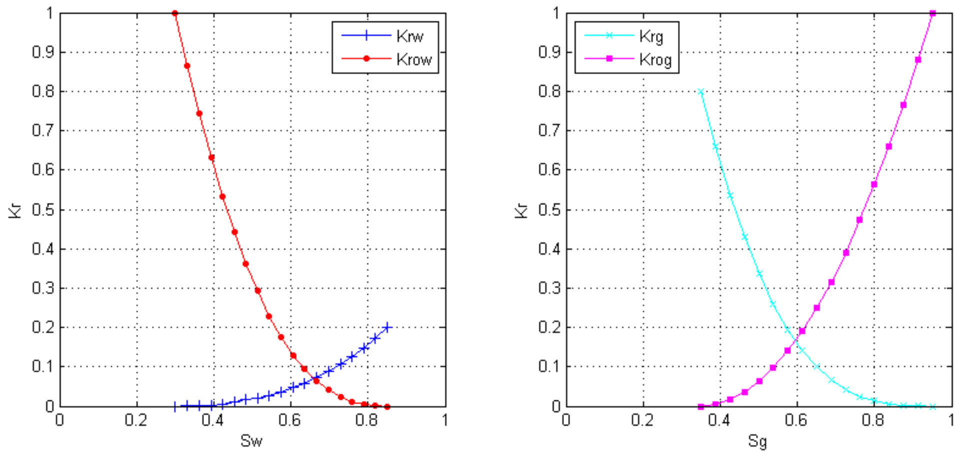


Figure 4.1: Relative Permeability Data

CHAPTER 5

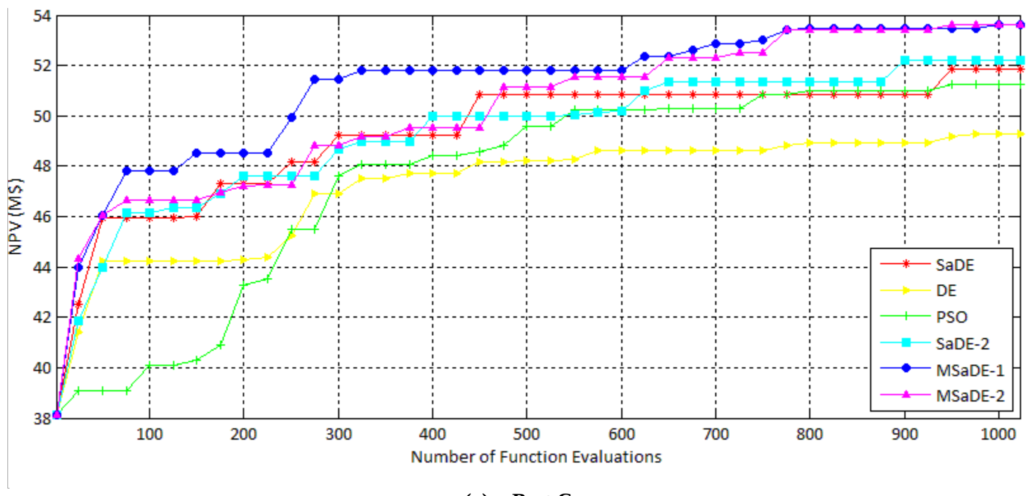
RESULTS AND CONCLUSION

5.1 ES-SAGD Optimization Results

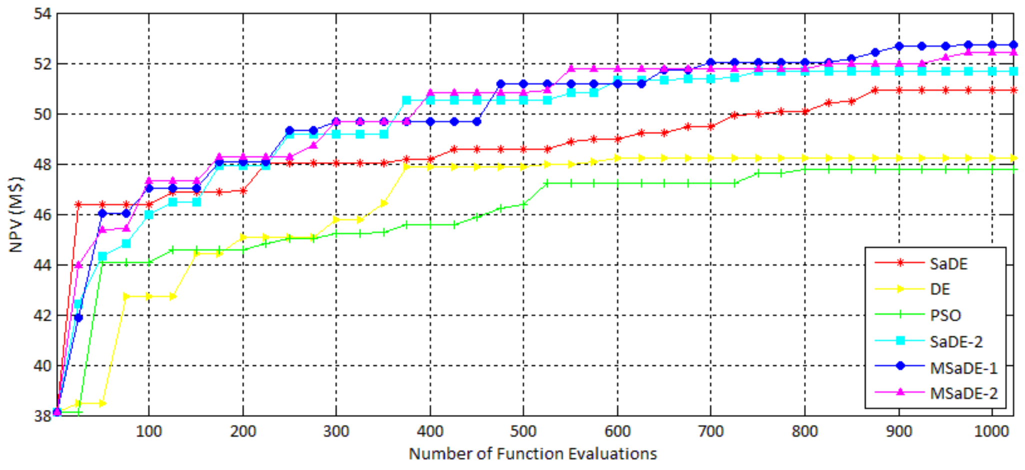
The optimization results of ES-SAGD (best run, median and worst run) of the comparison between SaDE, DE, PSO, SaDE-2, MSaDE-1 and MSaDE-2 algorithms are provided in Table 9. The best “median” result is typed in bold. The “best” result of an algorithm means the maximum NPV obtained within the five independent runs initiated with different populations. While the “worst” result means the minimum NPV obtained. Furthermore, in order to analyze the performance behavior of each algorithm, the convergence characteristics in terms of how fast the “Best”, “Median” and “Worst” of each algorithm reaches to the maximum NPV are shown in Figure 5.1a-c, respectively.

Table 5.1: Comparison between SaDE, DE. PSO, SaDE-2, MSaDE-1 and MSaDE-2 algorithms on optimizing ES-SAGD recovery method

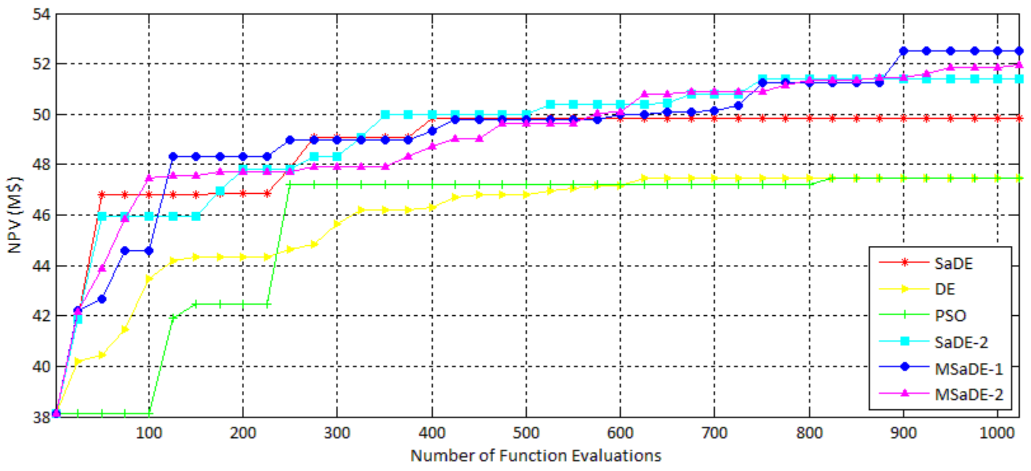
Algorithm	**NPV (x 10^6 $)**		
	Best	Median	Worst
SaDE	51.9	50.9	49.8
DE	49.3	48.3	47.5
PSO	51.3	47.9	47.5
SaDE-2	52.2	51.7	51.4
MSaDE-1	53.6	**52.7**	52.5
MSaDE-2	53.6	52.4	51.9



(a) Best Curves



(b) Median Curves



(c) Worst Curves

Figure 5.1: Optimization performance of SaDE, DE, PSO, SaDE-2, MSaDE-1 and MSaDE-2 on ES-SAGD recovery method.

5.1.1 Best Results Analysis

Figure 5.1.a shows the “best” runs which result in maximum NPV of ES-SAGD process at the end of the project. It is clearly shown that MSaDE algorithm performs better than other optimization algorithms including SaDE with two strategies in candidate pool. Although MSaDE-1 and MSaDE-2 achieved the same maximum NPV at the end of the project, MSaDE-1, with completely new adaptation technique, shows faster convergence speed toward optimum solution. MSaDE-2 reaches an NPV of 52 Million $ after 325 function evaluations, while MSaDE-1 and SaDE-2 reach to the same NPV after 650 and 900 function evaluations respectively.

5.1.2 Median Results Analysis

As shown in Figure 5.1.b, it is obvious that self-adaptive algorithms outperform normal methods without self-adaption techniques, although all algorithms have started with the same initial population. In addition, the proposed self-adaptive algorithms in this study (SaDE-2, MSaDE-1 and MSaDE-2) show better optimization performance than original SaDE algorithm. In particular, MSaDE-1, with completely new adaptation technique, performs better than MSaDE-2 and SaDE-2 and shows the maximum NPV at the end of the project.

5.1.3 Worst Results Analysis

The “Worst” results mean the minimum NPV value obtained by each optimization algorithm in five independent optimization runs. As shown in Figure 5.1.c, generally, self-adaptive algorithms exhibit better solutions than others do. It can be seen that starting from 750 function evaluations up to 875; MSaDE-1, MSaDE-2 and SaDE-2 almost have the same NPV value. However, MSaDE-1 is able to improve the solution and achieved the maximum NPV at the end of optimization process.

5.2 Performance Analysis of SAGD and ES-SAGD Recovery Methods

In this section, the performances of optimized ES-SAGD and SAGD recovery processes are studied and compared. Table 10 shows the maximum NPV values that have been achieved by ES-SAGD and SAGD recovery methods when applying MSaDE-1 optimization algorithm. It also shows the NPV values that have been obtained when optimizing the cumulative Steam Oil Ratio cSOR to the minimum (as most of the literatures do) instead of optimizing NPV.

As shown in Table 10, in Case #1, when NPV is the objective function to be optimized, the NPV values at the end of the project of ES-SAGD and SAGD are 52.5 and 31.8 M$ respectively. In other words, the ES-SAGD is 65% more profitable than SAGD in this example cse. In additions, cSOR in ES-SAGD and SAGD are 1.1 and 2.1 bbl/bbl respectively. Therefore, the ES-SAGD has cSOR value, almost, 50% less than SAGD. The oil recovery factors in this case are 66% and 47% for ES-SAGD and SAGD respectively; it also shows that ES-SAGD is better than SAGD it terms of oil recovery.

Table 5.2: Performance indicators of ES-SAGD and SAGD

Recovery Method	**Case #1: Optimizing NPV**			**Case #2: Optimizing cSOR**		
	NPV (M$)	cSOR (bbl/bbl)	RF %	NPV (M$)	cSOR (bbl/bbl)	RF %
ES-SAGD	52.5	1.1	66	31.9	0.8	47
SAGD	31.8	2.1	47	30.0	2.0	46

On the other hand, in Case #2, when cSOR is the objective function to be minimized, there is no significant difference between ES-SAGD and SAGD in terms of NPV and RF, although cSOR has reduced in ES-SAGD by 60% than SAGD. Therefore, this case does not show significant advantage of ES-SAGD in terms of economical profitability or oil recover capability.

This is one of the main objectives in this study; to show that having cSOR as objective function to compare the performance of ES-SAGD and SAGD recovery processes, is not the optimum selection. Because although the optimized cSOR of ES-SAGD is 60% less than SAGD (which agrees with literatures), the project profitability was not improved. Moreover, in real-life the most important considerations are the project's profitability and how much oil the process can recover. On the other hand, when selecting NPV as the objective function to be optimized, it is clearly shown that maximizing NPV results in improving cSOR and RF, as well as the project's profitability at the first place.

When optimizing only SAGD recovery process (not to be compared with ES-SAGD), it can be shown that selecting NPV or cSOR has no significant effect on representing recovery performance. In Case #1, it is shown that the NPV, cSOR and RF of SAGD are 31.8 M$, 2.1 bbl/bbl and 47% respectively. While in Case #2, the values are 30 M$, 2 bbl/bbl and 46% respectively. Therefore, there is no much difference in any of the performance indicators. On the other hand, in ES-SAGD, because of injecting solvent in addition to steam, cSOR is not representative, as it does not consider solvent cost or solvent losses in the reservoir. Therefore, NPV is better indicator to the performance of recovery process and project's profitability.

5.3 Conclusion

It was shown that the proposed self-adaptive DE algorithm with a new adaptation technique (MSaDE-1) exhibited better solution quality and convergence speed than other self-adaptive DE algorithms. Actually, the worst solution obtained by MSaDE-1 is better than the best solutions achieved by SaDE, DE, PSO and SaDE-2. It is also shown that the new proposed adaptation technique of trial-vector generation strategies with the aid of new candidate pool, achieve the required balance between exploration and exploitation capabilities of differential evolution optimization algorithms. Therefore, MSaDE-1 shows faster convergence speed, better solution quality and has never been stagnant in local optima.

It has also be shown that cumulative Steam Oil Ratio (cSOR) is not the right performance indicator when optimizing ES-SAGD recovery process, or comparing its recovery performance with SAGD. Because cSOR does not consider solvent cost or losses in the reservoir. On the other hand, NPV is better representative to project profitability and recovery capability. In addition, it has be shown that optimizing NPV, also, improving recovery factor (RF) and reduces cSOR proportionally.

References

Abbass, H. (2002). "The self-adaptive pareto differential evolution algorithm." in Proc. of the 2002 Congress on Evolutionary Computation: 831-836.

Ayodele, O. R., T. N. Nasr, J. Ivory, G. Beaulieu and G. Heck (2010). Testing and History Matching ES-SAGD (Using Hexane). SPE Western Regional Meeting, Anaheim. California.

Azad, M., S. Alnuaim and A. Awotunde (2013). Stochastic Optimization of Cyclic Steam Stimulation in Heavy Oil Reservoirs. 2013 SPE Kuwait Oil and Gas Show and Conference. Kuwait International Fair, Kuwait City, Kuwait, 2013, Society of Petroleum Engineers.

Boak, J. J. and C. Palmgren (2007). "Preliminary Numerical Analysis for a Naphtha Co-injection Test During SAGD." Journal of Canadian Petroleum Technology.

Butler, R. M. and D. J. Stephens (1981). "The Gravity Drainage of Steam-heated Heavy Oil to Parallel Horizontal Wells." Journal of Canadian Petroleum Technology.

Chandrasekar, K. and N. V. Ramana (2012). "Performance Comparison of GA, DE, PSO and SA Approaches in Enhancement of Total Transfer Capability using FACTS Devices." Journal of Electrical Engineering & Technology **7**.

Das, S. (2005). Diffusion and Dispersion in the Simulation of Vapex Process. SPE/PS-CIM/CHOA International Thermal Operations and Heavy Oil Symposium. Calgary.

Das, S., A. Abraham and A. Konar (2008). "Particle Swarm Optimization and Differential Evolution Algorithms: Technical Analysis, Applications and Hybridization Perspectives." Springer-Verlag Berlin Heidelberg.

Frauenfeld, T. W., C. Jossy, J. Bleile, D. Krispin and J. Ivory (2009). "Experimental and Economic Analysis of the Thermal Solvent and Hybrid Solvent Processes." Journal of Canadian Petroleum Technology **48**(11): 55-62.

Gamperle, R., S. D. Muller and A. Koumoutsakos (2002). Parameter study for differential evolution. WSEAS NNA-FSFS-EC 2002.

Gates, I. D. and N. Chakrabarty (2005). Optimization of Steam-Assisted Gravity Drainage in McMurray Reservoir. Canadian International Petroleum Conference. Calgary, Alberta, Petroleum Society of Canada.

Gates, I. D. and N. Chakrabarty (2012). "Design of the Steam and Solvent Injection Strategy in Expanding Solvent Steam-Assisted Gravity Drainage." Journal of Canadian Petroleum Technology.

Govind, P. A., S. Das, S. Srinivasab and T. Wheeler (2008). Expanding Solvent SAGD in Heavy Oil Reservoirs. SPE/PS/CHOA International Thermal Operations and Heavy Oil Symposium. Calgary.

Hajizadeh, Y., M. Christie and V. Demyanov (2009). Application of Differential Evolution as a New Method for Automatic History Matching. Kuwait International Petroleum conference and Exhibition. Kuwait.

Hedar, A. (2007). GLOBAL OPTIMIZATION TEST PROBLEMS. Kyoto University JAPAN.

Ivory, J., R. Zheng, T. Nasr, X. Deng, G. Beaulieu and G. Heck (2008). Investigation of Low Pressure ES-SAGD. SPE/PS/CHOA International Thermal Operations and Heavy Oil Symposium. Calgary.

Jahangiri, H. R. (2007). Production Optimization using Smart Well Technology with Differential Evolution Algorithm. Graduate Student Symposium. University of Southern California.

Khan, M. Y. (1993). Theory & Problems in Financial Management. Boston, Boston: McGraw Hill Higher Education.

Kumar, A., V. Oballa and C. C. Card (2010). Fully-Coupled Wellbore Design and Optimization for Thermal Operations. presented the Canadian Unconventional Resources & International Petroleum Conference. Calgary, Canada.

Liu, G., Y. Li, X. Nie and H. Zheng (2012). A novel clustering-based differential evolution with 2 multi-parent crossovers for global optimization. Appl Soft Comput.

Mallipeddi, R., P. Suganthan, Q. Pan and M. Tasgetiren (2011). Differential evolution algorithm with ensemble of parameters and mutation strategies. Appl Soft Computer.

Mohamed, A., H. Sabry and T. Abd-Elaziz (2013). "Real parameter optimization by an effective differential evolution algorithm." Egypt Inform J.

Nasr, G. Beaulieu, H. Golbeck and G. Heck (2003). Novel Expanding Solvent-SAGD Process ES-SAGD. presented at Canadian International Petroleum conference (CIPC). Calgary.

Nghiem, L., A. Mirzabozorg, C. Yang and Z. Chen (2013). Differential Evolution for Assisted History Matching Process: SAGD Case Study, Society of Petroleum Engineers.

Noman, N. and H. Iba (2008). "Accelerating Differential Evolution Using an Adaptive Local Search." IEEE Transactions on Evolutionary Computation **12**(1): 107-125.

Omran, M., A. Salman and A. P. Engelbrecht (2005). "Self-adaptive differential evolution, computational intelligence and security." PT 1, Proceedings Lecture Notes In Artificial Intelligence **3801**: 192-199.

Onwunalu, J. E. and L. J. Durlofsky. (2010). "Application of a particle swarm optimization algorithm for determining optimum well location and type." Computational Geosciences.

Piotrowski, A., J. Napiorkowski and A. Kiczko (2012). Differential evolution algorithm with separated groups for multi-dimensional optimization problems. Eur J Oper Res.

Qin, A. K., V. L. Huang and P. N. Suganthan (2009). "Differential Evolution Algorithm With Strategy Adaptation for Global Numerical Optimization." EEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION **13**(2).

Rahnamayan, S., H. R. Tizhoosh and M. M. A. Salama (2008). "Opposition-Based Differential Evolution." IEEE Transactions on Evolutionary Computation **12**(1): 64-79.

Ronkkonen, J., S. Kukkonen and K. V. Price (2005). "Real parameter optimization with differential evolution." IEEE Congress on Evolutionary Computation (CEC-2005) **1**: 506 – 513.

Storn, R. and K. Price (1995). "Differential Evolution - A simple and efficient adaptive scheme for global optimization over continuous spaces."

Teo, J. (2006). "Exploring dynamic self-adaptive populations in differential evolution." Soft Computing (A Fusion of Foundations, Methodologies and Applications).

Wang, C., J. Gao, H. Yang, X. Jiantong, W. Zhao and Z. Zhu (2011). Waveform Inversion of Cross-well Data with Cooperative Coevolutionary Differential Evolution Algorithm. SEG Annual Meeting. San Antonio.

Wang, J. and J. S. Buckley (2006). Automatic History Matching using Differential Evolution Algorithm. International Symposium of the Society of Core Analysts. Trondheim, Norway.

Yao, X., Y. Liu and G. Lin (1999). "Evolutionary programming made faster." IEEE Transactions on Evolutionary Computation **3**(2): 82-102.

Zaharie, D. (2003). Control of population diversity and adaptation in differential evolution algorithms. 9th International Conference on Soft Computing. Czech Republic.

Zhang, D., X. Gong and L. Peng (2009). Estimating Geostatistics Variogram Parameters Based on Hybrid Orthogonal Differential Evolution Algorithm. 4th International Symposium on Intelligence Computation and Applications, ISICAConference. Huangshi, China.

Vitae

Name	: Tamer Magdi Moussa
Nationality	: Egyptian
Date of Birth	: 05/21/1985
Email	: engineertamer.2007@gmail.com
Address	: P.O. Box 2421, Dammam 31451, Saudi Arabia
Academic Background	: Petroleum Engineering
	Submitted journal paper “Optimization of Expanded-Solvent Steam Assisted Gravity Drainage Using Differential Evolution”