Hybrid DE Algorithm for the Solution of Bound Constrained Single-Objective Computationally Expensive Numerical Optimization Problems

Mariam Abdulmoti Holoubi

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

Master of Science
in
Computer Engineering

Eastern Mediterranean University
January 2018
Gazimağusa, North Cyprus

Approval of the Institute of Graduate Studies and Research

Assoc. Prof. Dr. Ali Hakan Ulusoy
Acting Director

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

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

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

Asst. Prof. Dr. Ahmet Ünveren
Supervisor

Examining Committee

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

ABSTRACT

The Differential Evolution Algorithm is widely used for the purpose of optimization in many fields. This dissertation proposes a Hybrid Differential Evolution Algorithm and examines its feasibility based on the results of CEC'15 expensive benchmark problem optimization. A local search mechanism was used to develop three versions of Hybrid DE. All versions of the proposed method were used and compared according to the final feedback of their optimization results. Another comparison with five different methods proposed in the related literature was conducted. The final ranking of all the methods implied that Hybrid DE was always among the top best algorithms that were used for the same purpose.

Keywords: Differential Evolution, Evolutionary Algorithms, Local Search, Hybrid Algorithms.

ÖZ

Diferansiyel Evrim Algoritması (DE) bir çok alanda optimizasyon amacıyla yaygın olarak kullanılmaktadır. Bu tezde Hibrid Diferansiyel Evrim Algoritması önerilmektedir. Öneril enalgoritmanın başarımı CEC'15 pahalı en iyileme problemlerinin çözümleri üzerinden incelenmiştir. Bir yerel arama mekanizması kullanılarak üç farklı DE algorithması geliştirilmiştir. Önerilen yöntemin tüm versiyonları kullanılmış ve optimizasyon sonuçlarının son geri bildirimine göre karşılaştırılmıştır. İlgili literatürde önerilen beş farklı yöntemle karşılaştırma yapılmıştır. Tüm yöntemlerin son sıralaması yapıldığında önerilen metodun diğer en iyi algoritmalar ile karşılaştırılabileceği gözlenmiştir.

Anahtar Kelimeler: Diferansiyel Evrim, Evrim Algoritmaları, Yerel Arama, Hibrit Algoritmalar.

To those who turn a blind eye on our mistakes.

Instead of giving up, they aid us with a helping hand.

ACKNOWLEDGMENT

I would initially like to express my sincere gratitude to my supervisor, Asst. Prof. Dr. Ahmet Ünveren**,** for his invaluable guidance, encouragement and understanding. He has taught me more than I could ever give him credit for here. He has shown me, by his example, what a good scientist (and person) should be. I, also, would like to thank the members of the jury, Asst. Prof. Dr. Adnan Acan and Asst. Prof. Dr. Mehtap Köse Ulukök for their reviews and comments for the improvement of this thesis. Special gratitude to Asst. Prof. Dr. Adnan Acan for his support and positive energy that he provided me during all my time in the EMU. I should also express my deepest gratitude to the computer engineering department graduate committee chair, Assoc. Prof. Dr. Önsen Toygar, for all the time and effort she had given to carefully guide and help me throughout my journey.

I am especially grateful to Asst. Prof. Dr. Nilgün Hancioğlu. Although the time we spent together went by so fast, I appreciate and cherish every moment of it. For her warm assurance and guidance and for the effort and time she had given me, I will always be thankful.

Nobody has been more supportive to me than my parents, whose unconditional love is with me in whatever I pursue. I am indebted to them for everything in my life. They are my ultimate role models. And to God I am grateful for granting me with my loving family. My deepest love and gratitude for my brothers, Ibrahim and Ismail, and for my sisters, Batoul and Tasneem, who were there for me anytime I needed support.

How can I ever thank my dearest Esra'a, who always tells me that she believes in me. She could not be here by my side, but her profound love and support followed me overseas and comforted me the most.

My deepest gratitude to all the thoughtful wishes of my old friends, each message and call was deeply appreciated. I would also like to thank all those friends that accompanied and helped me in the pursuit of my Master's degree. Special thanks to my best friend in Famagusta, Basma Anber, who helped and encouraged me most of all to complete my thesis.

To my eldest sister, Batoul, whom together we went through thick and thin. I would have given up if it was not for you by my side. I am forever grateful.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

CR	Crossover Rate
DE	Differential Evolution
EA	Evolutionary Algorithm
GA	Genetic Algorithm
GRASP	Greedy Randomized Adaptive Search Procedure
LS	Local Search
MA	Memetic Algorithms
NP	Non Deterministic Polynomial Time Problem

Chapter 1

INTRODUCTION

1.1 Background to the Study

Since the term "Metaheuristics" was first incepted in the late half of the 80s, the researchers' understanding and working with metaheuristics is continuously progressing and shifting in different research areas. In a recently published research "A History of Metaheuristics" (K. Sorensen et al., 2017) the author suggests that people have been using heuristics and metaheuristics long before the term even existed. Also, he stated that the mentioned term has lacked a satisfying definition until recently, despite the fact that people have been using heuristics over the years [1]. The following statement was approved by the author to be the best definition:

> A Metaheuristics is a high-level problem-independent algorithmic framework that provides a set of guidelines or strategies to develop heuristic optimization algorithms." (Sorensen and Glover, 2013).

The research about history of metaheuristics brings to light the five different periods that shaped the evolution of the concept of "metaheuristics". The first phase was named the Pre-Theoretical period (until C.1940), which the author insisted that, during that phase, heuristics and even metaheuristics were used but were not formally studied. The second stage, the Early period (C.1940-1980), emphasized the beginning of formal studies on heuristics. "Artificial Intelligence" is the term that was used to recognize the work that was done during this period, because it tends to mimic human problem-solving behavior. Then another line of research of problem-solving behavior started in the 1960s that highlighted the use of evolution as a

problem-solving method. It started with the insight that the principle of natural evolution could be used to solve optimization problems in general. Box (1957) and several others had independently developed algorithms inspired by evolution, mainly aiming for function optimization and machine learning [43]. The first method that was recognized was called Evolution Strategy [44]. Due to the lack of using the concepts of population or crossover, it was not considered as an algorithm. One solution, called the parent, was mutated and the best of the two solutions became the next parent for the next round of mutation. Evolutionary programming was introduced few years later in 1960. However, it did not use both population and crossover method. In 1989, Goldberg published his book that was the spark of the evolutionary revolution. Evolutionary methods became extremely popular and a large number of variants were proposed [1]. Evolutionary strategies or Evolutionary Algorithms (EA) have become an important tool for performing a variety of search and optimization procedures. The recent method of any EA considers creating a finite group of correspondence structures to stand for the idea of a population. Each structure is exactly like the other and together they form a generation of individuals. An individual is presented typically by a string, imitating the biological genotype. Decoding the genotype will produce the phenotype data which is mathematically-based structure to present a solution. The referred solutions contain parameters which solve a correspondent fitness function to the problem we are trying to optimize. Each individual in the population is evaluated and then assigned a fitness value according to the fitness function of this particular problem. The corresponding fitness values will be the preference factor to decide which individual is more suitable of reaching optimality, or near-optimality status [2].

In the method-centric period (C.1980-C.2000) the field of metaheuristics truly took off and many different methods were proposed. The concept of annealing: controlling heating and cooling process used in metal and glass production (K. Patrick et al., 1983) was the first published paper of general problem-solving framework that was not based on natural evolution. The process of Simulated Annealing depends on an external parameter called the temperature. Random solution changes were used and accepted if they improved the solution. One of the most powerful ideas was that solutions could be gradually improved by iteratively making small changes, called moves [1]. This ignited the development of well-known heuristic algorithms that are now called Local Search mechanisms. By adapting the concept of small moves, the solution could be mutated by a single change for reaching another, yet very close, solution. By repeating these kinds of changes, the algorithms will be investigating all or some of the nearby solutions around the specific small space surrounding the first one. Such space is called the current solution's neighborhood.

Research of Metaheuristics had grown and several frameworks had been proposed around the early 90s. The innovation proposed GRASP (Greedy Randomized Adaptive Search Procedure). It followed a randomized greedy behavior by selecting through each iteration, not necessarily the best element, but one of the best elements randomly [45]. Similarly, Ant Colony Optimization [46] was proposed not only how to mix deterministic and stochastic information, but also proposed a way for solutions to exchange information [1]. The Differential Evolution Algorithm (DE) was officially introduced in a publication by R. Storn and K. Price, (1995). The article explained the steps of the algorithm thoroughly. Soon after that, R. Storn (1996) proposed an application of using DE for designing an IIR-filter[3]. Another

research used the DE algorithm (P. Thomas and D. Vernon, 1997) for image registration. The majority of research tended to apply DE in Image Processing Applications until 1998, a hybrid method of DE was introduced to start the recognition of the DE remarkable performance for solving some engineering optimization problems [4].

The Framework-centric period (C.2000-now) featured the worldwide knowledge growing that led to describing metaheuristics as frameworks, not only methods. A wide variety of EAs have been introduced and studied by assessing their performance and studies tended to develop them by introducing new, hybrid metaheuristic algorithms based on the merging of two or more procedures for the aim of optimizing results of problem-solving. Many systematic studies of the performance and behavior of heuristics such as evolutionary algorithms (Oliveto et al., 2007; Auger and Deorr, 2011; Neumann and Witt, 2010) discovered both easy problems where heuristics perform well and also easy problems where they fail and require more time [47,48,49]. Heuristics proved to be able to optimize several classical combinatorial problems efficiently and they could deliver good near optimal solutions for NP hard problems [1].

1.1.1 Evolutionary Algorithms

Evolutionary algorithm steps in general will start with ***initializing*** a population. After initializing two or more individuals, their fitness will be ***evaluated*** according to the objective function corresponding to the problem we are trying to optimize. After initialization, the evolution-loop starts processing its operators; ***recombination***, ***mutation***, ***evaluation*** and ***selection***. The selected parents are used to perform a hybridizing process in order to construct an ***offspring*** individual either using the recombination or mutation procedure. Recombination creates new individuals from

the parent population. Recombination is sometimes used, but mutation is generally the more preferred operator due to its factor of enhancing the variation in new generations in the evolutionary strategy. The newly created individuals are then evaluated, i.e., their fitness values are calculated. Based on the new fitness values, the selection stage identifies a subset of individuals which form the new population existing in the next iteration of the evolution loop. The loop is terminated based on a ***termination criterion*** set by the user; reaching a maximum number of evaluations or reaching a target fitness value for example [5,6].

Figure 1 shows the general outline of an evolutionary algorithm [5].

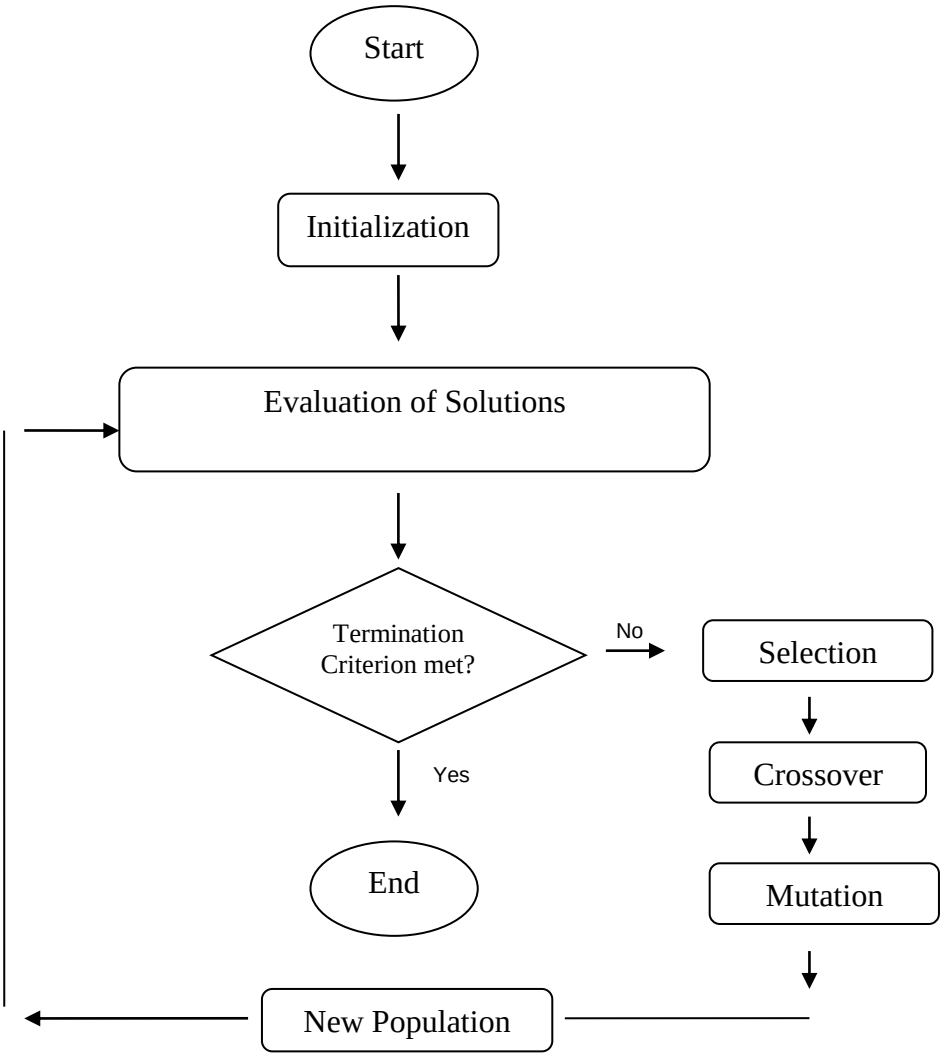

Figure 1: General outline of an evolutionary algorithm

Figure 2 demonstrates how one generation is broken down into a selection phase and a recombination phase. The strings are shown as being assigned into adjacent slots during selection. They can be assigned slots randomly in order to shuffle the intermediate generation [7].

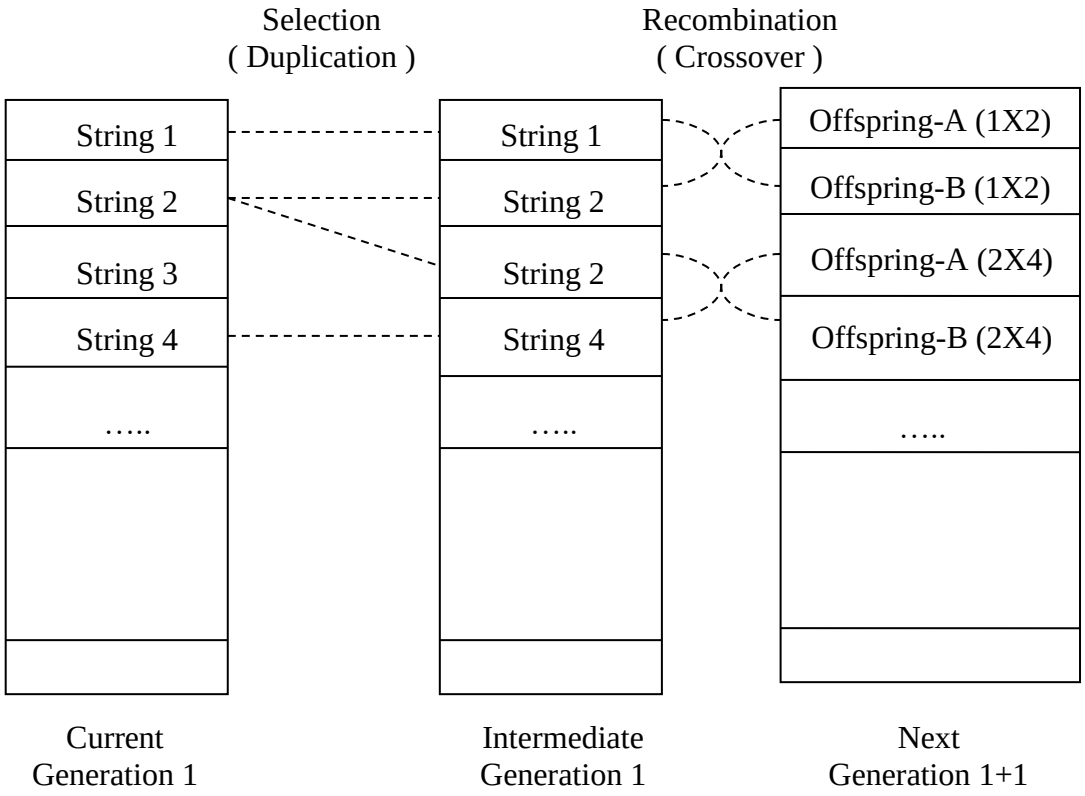

Figure 2: Selection and Recombination phases of standard evolutionary algorithm

1.1.2 Memetic Algorithms

Memetic Algorithms (MA) is a name of the set of metaheuristics specifically containing ***population-based*** evolutionary approaches that work cooperatively with agents concerned in periodic individual improvement of the solutions. The name of Memetic Algorithms (MA) was initially derived from the term "meme" that was defined by R. Daukins to emphasize the importance of small component improvement in the context of the big evolutionary process. An MA is a search

strategy in which a population of optimizing agents intrinsically cooperate and compete. They are well known for their success in solving many hard optimization problems. MAs exploit the search space by incorporating preexisting heuristics, processing data reduction rules, approximation or using local search techniques [50].

1.1.3 Previous Work

Paperwork of the previously conducted experiments on the same group of problems had little interest in the scope of hybridizing DE with LS mechanisms for superior optimization results. Noor Awad [9] et al. introduced a new technique to adapt the control parameters using a memory-based structure of the past successful settings and employing the population resizing factor for differential evolution algorithm. Another paper, Shu-Mei Gou [10] et al. (2015), proposed L-SHADE. A variant of DE algorithm based on a linear population size reduction concept. The method was tested for real parameter single objective optimization of CEC2015 problems. The mechanism was incorporated with a binomial crossover operator and successful parent selecting framework to avoid stagnation. Moreover, Neurodynamic Differential Evolution is a recent approach that showed remarkable results for a variant of dimensions regarding a group of problems. The proposed algorithm is a linear population size reduction DE dependent on modification of success history based parameter embedded with the concept of neurodynamic[11]. Another study on CEC2015 problems tested problem optimization using Self-adaptive Dynamic Multi-Swarm Particle Optimizer (sDMS-PSO). The factor of difference between sDMS-PSO and the original PSO algorithm is demonstrated in the employment of ***self-adaptive strategy*** of parameters, while in original PSO, a specific number of three parameters will be given either according to experimental or empirical behavior. At the end, a local search method of the quasi-Newton is included to enhance the ability

of exploitation [12]. The final study that was introduced was the Hybrid Cooperative Co-evolution for CEC2015 Benchmarks (hCC). The experiment tested the performance of hCC. The method's concept is to separate the variables into groups of separable and non-separable in its early stage. During the second stage, it continues in adopting different algorithms within the cooperative co-evolution (CC) framework [13].

Where previous research has often focused on variant ways to conduct single objective problem optimization, they showed little interest in the idea of hybridizing evolutionary algorithms.

1.2 Aim of the Study

In this study, hybridizing the Differential Evolution Algorithm with local search (LS) mechanism, which will be explained in detail later, is the main experimental concept. The results of hybrid DE assessed on solving CEC2015 Benchmark Problems will be discussed [14] . This research targeted emphasizing the empowerment of using LS with DE; the well-known global optimization metaheuristic. The aim of the experiment is to reach optimality, or near-optimality solutions for single objective problems.

A general single objective optimization problem is defined as minimizing, or maximizing, $f(x)$ subject to $g(x)$ and $h_j(x)$ in (eq. no 1),

$$g(x) \leq 0,\ i = \{ 1,\ldots,m \} \qquad (1)$$

$$h_j(x) = 0,\ j = \{ 1,\ldots,p \}\ x \in \Omega.$$

x is a n-dimensional decision variable vector. $x = (x_1,\ldots,x_n)$ which belongs to the search space ranged by the constrains of the problem. $g(x)$ and $h(j)$ represent the

constraints that must be fulfilled while optimizing *f(x)*. **Ω** is the set of all possible real values that satisfy the evaluation of *f(x)* [8].

The significance of using objective function is presented in providing the capability of approaching the global minimum between all the possible values of *x* by evaluating an objective function *f(x)* to find the fitness values of *x*. x^* is called a global minimum if and only if the condition in (eq. no(3)) is fulfilled.

$$\forall x \in \Omega: f(x^*) \leq f(x) \tag{2}$$

Where **Ω** is the set of all possible real values that satisfy the evaluation of *f(x).*

1.3 Significance of the Study

The aim of optimization is to determine the best-suited solution to a problem under a given set of constraints [38]. In the process of single objective problem optimization, local search is considered to be an excellent tool for exploitation of a limited area of the search space, but using only LS for optimization risks reaching stagnation when stuck in the local optimum. On the other hand, the DE algorithm will provide the feature of global exploration during its mutation stage. The combination of this ***local*** and ***global*** heuristic methods will very probably result in excellent solutions to reach our aim of optimizing single objective problems.

1.4 Structure of the Thesis

This thesis is organized so that first chapter is the introduction and background to the study. The second chapter will discuss the Differential Evolution Algorithm method in detail listing its development stages since the inception. Third chapter will present our proposed method of Hybrid DE with Fmincon LS for optimizing Single Objective Benchmark Problems of CEC2015. Next is the fourth chapter that will

investigate experimental part and discuss results of the experiment. The final chapter is the conclusion of this study.

Chapter 2

THE DIFFERENTIAL EVOLUTION ALGORITHM

2.1 Taxonomy

Differential Evolution (DE) is arguably one of the most powerful ***stochastic*** real-parameter optimization algorithms in current use [15]. DE uses a few control parameters for reaching the true global minimum, regardless of the initial parameters values. Being a stochastic method, it mainly uses random mechanisms to ***initiate*** population and then proceed in the same operators originally from Genetic Algorithm (GA); ***crossover***, ***mutation*** and ***selection*** [2]. The algorithm operates through similar computational steps as employed by a standard EA. However, unlike traditional EAs, the DE-variants perturb the current-generation population members with the scaled difference of randomly selected and distinct population members [15].

2.2 Procedure

In DE, a population of NP number of individuals is randomly initialized using (eq. no 3) with the ***bounds*** on decision variables [17]

$$x_{i,j}(0) = x_j^L + rand(0,1) \,.\, (x_j^U - x_j^L) \qquad (3)$$

Where, i = 1,…..,N (N: population size), j = 1,…..,D (length of an individual)[] and *rand(0,1)* is a random number from uniform distribution between 0 and 1. $(x_j^U - x_j^L)$ are the limitations of ***upper bound*** and ***lower bound*** on the j^{th} decision variable [17]. The basic mechanism the used variant of DE works upon is ***subtraction***, demonstrated in equation (4), by randomly selecting mutually different ***vectors*** r_1, r_2 ***and*** r_3, subtracting two of them and the differences are applied weight given to them

by a factor ***F*** called the ***differential weight***. Finally, by adding the difference to the third vector, the result will be obtaining the ***perturbation*** vector u_i, (eq. no 4), as follows [16]:

$$u_i = r_{3i} + F(r_{1i} - r_{2i}), i = 1,2, \dots, D \qquad (4)$$

where D is the dimensionality of the individuals. Perturbation vector *u* is also called a ***donor*** because it is produced only for donating its parts to the new offspring. This perturbation technique follows the basic rule of ***DE/rand/1*** variant of DE. The second step is to find the ***trial*** vector *y* by applying binary crossover shown in fig. 3 on the ***target*** vector *x* and the ***donor*** vector *u*. This step relies mainly upon the ***crossover rate*** factor (CR) which is the key to decision whether the new individual takes its component from vector *x* or vector *u* [8].

```
j = rand[1,D]
for i = 1 to D
   if(rand[0,1] < CR or i == j) y_i = u_i;
   else y_i = x_i;
end
```

Figure 3: Selection Procedure in DE using a stochastic binary Crossover rate

Binary crossover mainly depends on the strategy of single-point crossover that is used in many applications of binary coded EAs. In single-point crossover, a random cross site is identified along the length of the solution string and the bits of one side are swapped between the two parent strings. In single-variable optimization problem, the action of the crossover is to used to create two new offspring strings from two parent strings, while in multi-parent optimization problem, each variable is usually coded in a certain number of bits and these bits are then combined to form the string of the solution [51].

The basic steps of DE are demonstrated in Figure 4 [18]:

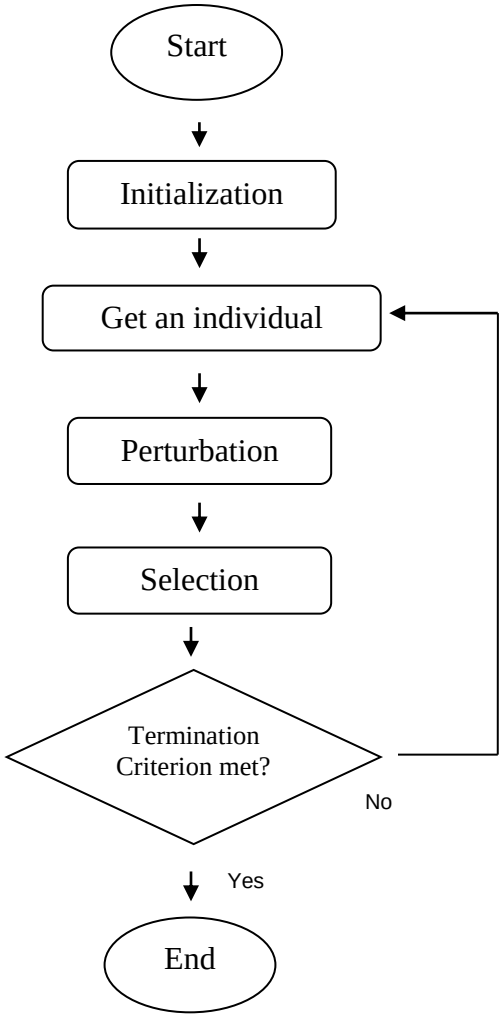

Figure 4: Flowchart of the basic Steps of DE[18]

One of the most important features of DE is contour matching, which means that the generation population works in such way that promising regions of the objective function surface are investigated automatically once they are detected. An important ingredient besides ***selection*** is the promotion of basin to basin transfer; search points may move from one basin of attraction (local minimum) to another. This suggests that DE only accepts better solutions as the searching process advances [16].

```
Input: Population_size, Problem_size, Weighting_factor,
       Crossover_rate
Output: S_best
1 Population ← InitializePopulation(Population_size,
  Problem_size );
2 EvaluatePopulation ( Population );
3 S_best ← GetBestSolution(Population);
4 while ¬ StopCondition() do
5       NewPopulation ← Ø;
6       foreach P_i ∈ Population do
7             S_i ← NewSample (P_i, Population, Problem_size,
               Weighting_factor, Crossover_rate );
8             if Cost(S_i) ≤ Cost (P_i) then
9                   NewPopulation ← S_i;
10            else
11                  NewPopulation ← P_i;
12            end
13      end
14      Population ← NewPopulation;
15      EvaluatePopulation(Population);
16      S_best ← GetBestSolution(Population);
17 end
18 return S_best;
```

Figure 5: Pseudo Code of DE[18]

Where:

- $Population_{size}$: No. of individuals in one population.
- $Problem_{size}$: No. of decision variables in one vector.
- $Weighting_{factor}$: Differential weight F.
- $Crossover_{rate}$: CR factor.
- Population: Current generation of individuals.
- NewPopulation: The next generation of individuals.
- S_{best}: The best solution found so far.
- P_i: An individual in the current population.
- S_i: New individual vector found after applying DE process.
- InitializePopulation(): Returns randomly-generated population.
- EvaluatePopulation(): Returns fitness values of all the population individuals.

- GetBestSolution(): Returns the individual with minimum fitness value.
- StopCondition(): Stopping Criteria.
- NewSample(): Returns the trial vector y_i.
- Cost(): Returns the fitness value of one vector.

2.3 Chronological Evolution of Hybrid DE

Since its ***inception*** in ***1995***, DE has drawn the attention of many researchers all over the world resulting in a lot of variants of the basic algorithm with improved performance [15]. The article that was published by R. Storn and K. Price officially introduced DE algorithm with thorough explanations of the steps which DE is based upon. That first publication of DE was proposed two years before R. Storn wrote two different articles about using "Differential Evolution design of an IIR-filter" and the "Usage of differential evolution for function optimization". P. Thomas and D. Vernon (***1997***) together proposed a method for "Image registration by Differential Evolution". Most of the studies that were interested in the usage of DE focused on image processing until J. P. Chiou and F. Sh. Wang (***1998***) realized the fact that some engineering optimization problems are being solved with the aid of all different EAs including DE. They proposed a hybrid method of DE for the purpose of engineering optimization problems. In ***1999***, the DE was described as a simple problem optimization procedure for constraint based problems with the aim of simplifying system design [19].

Studies during the first decade of 2000s about DE were more detailed with experiments and even showed the problems that DE could face. J. Lampinen and I. Zelinka wrote about ***stagnation*** of the DE algorithm in ***2000***. They stated that stagnation is different from ***premature convergence*** because of the consistent

diversity remaining in the population even after reaching stagnation, but the optimization process does not progress anymore. They concluded that the reason for stagnation remained unknown so far. The first introduced DE variant was Pareto-Frontier Differential Evolution (PDE) [20] in ***2001***. PDE was targeted for solving multi-objective optimization problems. The same author published a paper the following year, describing a self-adaptive Pareto Differential Evolution (SPDE) [21]. Self adaptive Differential Evolution (SADE) was proposed by A. K. Qin and P. N. Suganthan in ***2005***. The algorithm used ***learning strategy***. The F parameters and CR parameters were not required to be pre-specified; rather they will be self adapted during evolution using a suitable learning strategy. In ***2008***, For the enhancement of effective EAs, a crossover-based adaptive LS was used with the standard DE featuring the adjusting of the length of the search accordingly using a hill-climbing heuristic [22]. Another hybrid DE method (HDE) was proposed for solving the permutation flow-shop scheduling which is a combinatorial NP hard-single-objective optimization problem [23]. First, they changed the continuous nature of DE individuals to job permutation using largest-order-value, then applied a simple LS designed corresponding to be suitable with the problem's ***scope***, ***nature***, ***range*** and ***features***. Finally, HDE was extended to Multi-objective HDE (MHDE) to solve muti-objective version of the same problem. In ***2009***, J. Zhang and A. Sanderson introduced the ***JADE***; Adaptive DE with optional External Archive. The variant used a new mutation strategy with optional memory-usage and adaptive updating for control parameters. Moreover, a novel hybridization of two well-known EAs; ***DE*** and ***PSO***, was proposed also in the same year for the purpose of unconstrained optimization. The algorithm was called DE-PSO which included basic mechanisms from both EAs [24].

The concept of hybridization of DE became more popular in ***2010*** when two noticeable studies were published. The first was hybrid DE with biogeography-based optimization [25]. It was designed for global numerical optimization. It depended on the biogeography-based ***migration*** operator for exchanging information between DE individuals, which combined the exploration feature of DE with the exploitation of BBO effectively. The second publication on hybridizing DE during the same year proposed two hybrid DE algorithms for engineering design optimization [26]. After that, in ***2011***, Young Wang et al. published an article about DE with composite trial vector ***generation strategies*** and ***control parameters***. Results of the study have shown that employing generation strategies and control parameters have significant influence on the performance. The proposed method was tested on all the CEC2005 contest test instances [27].

The previously mentioned study in ***2010*** that proposed two hybrid algorithms [26] led to another experiment in ***2012*** for hybridizing DE with another EA. An article about Co-evolutionary DE with Harmony search (DEHS) for reliability-redundancy optimization was published [28]. The method of the algorithm was to divide the problem into a ***continuous*** part and an ***integer*** part. Eventually, two ***populations*** evolve simultaneously and co-operatively. Hybrid Robust Differential Evolution (HEDE) was proposed in the same year [29], adding positive properties of the Taguchi's method to the DE for minimizing the production cost associated with multi-pass tuning problems.

Success History based DE (SHADE) variant of DE, was proposed in ***2013*** by Ryoji Tanabe; an enhancement of JADE [30] which uses a history-based parameter adaptation scheme, instead of generating new control parameters. SHADE method

uses a historical memory (M_{CR}, M_F) which stores a set of combination of these parameters that have performed well before. Then, it generates new (CR) and (F) parameters close to ones of the pairs stored in the memory. Another variant of DE was proposed in the same year, SapsDE [31]. Population resizing mechanism was used in this method to enhance performance of DE by dynamically choosing one of two ***mutation*** strategies and tuning control parameters in a self-adaptive manner. The method was tested on 17 benchmark functions.

Fireworks algorithm (FA) is relatively a new swarm-based metaheuristic for global optimization. An improved version of FA was developed in ***2015*** [32] using the combination with DE operators; ***mutation***, ***crossover*** and ***selection***. At each iteration, the newly generated solutions are updated under the control of randomly selected vectors out of the best-so-far solutions. Another hybrid method in ***2015*** was proposed to merge the Genetic algorithm (GA) with DE, termed (hGADE) [33], to solve one of the most important power system optimization problems known as the unit commitment (UC) scheduling. The binary UC variables were evolved using GA while the continuous dispatch variables were evolved using DE. That is due to the GA capability of handling binary variables efficiently and the DE remarkable performance in real parameter optimization.

An article published in ***2016*** proposed Memory-based DE (MBDE)[34]. The method had two swarm operators introduced which were based on the ***personal best*** (pBest) and ***global best*** (Gbest) mechanisms of PSO. The method was tested on 12 basic, 25 CEC2005 and 30 CEC2014 unconstrained benchmark functions. Another variant was proposed in the same year, based on modified JADE (MJADE) and modified CoDE (MCode), named (HMJCDE) [35]. Both of the hybrid algorithms were operated

alternatively according to the improvement rate of the fitness value. The proposed method performance was assessed on 30 benchmark problems taken from CEC2014. Generalized Differential Evolution (GDE) is the most recent hybrid DE, proposed in ***2017***, for solving numerical and evolutionary optimization [36]. GDE is a general purpose optimizer for global non-linear optimization. The basic DE was extended to handle multiple constraints and objectives just by modifying the ***selection*** rule. Another newly published article introduced the idea of continuous adaptive population reduction (CAPR)for DE. The improvements upon this method are in terms of efficiency and convergence over the original DE and constant population reduction DE. It continuously adjusts the reduction of population size accordingly during exploitation stage [37].

The concept of hybridization has been the centre of attention of research in the optimization area during the last twenty years. Various hybrid DE algorithms have been introduced and conducted mainly for problem optimization. Overall, further optimization research on improving performance of DE and all other EAs that are known has a promising future.

Chapter 3

METHODOLOGY

This study employs a hybridization technique of metaheuristic evolutionary algorithm with a local search mechanism to examine the results of single-objective problem optimization. Hybridizing EAs have been used by different researchers during the past twenty years. EAs have proven their ability to explore large search spaces, but they are comparatively inefficient in fine tuning the solution. This drawback is usually avoided by means of local optimization algorithms that are applied to the individuals of the population. The algorithms that use local optimization procedures are usually called hybrid algorithms [39].

In the process of merging the Differential Evolution algorithm together with a Local Search technique, ***Fmincon*** LS which is a non-linear programming method was used as an optimization tool applied to the individuals of the DE population. Using a non-linear programming function gives a significant aid to our aim to minimize single-objective problem. The experiment was conducted by the implementation of three different variants of hybrid DE with Fmincon LS. The first variant applied the local optimization on the area around the ***new individual*** that was found after going through DE ***mutation*** then ***selection*** steps. In the experiment of the second variant of hybrid DE, we applied LS to the ***best solution*** found in the ***current*** population before starting the DE main loop. The final experiment was intended to fuse the past two

variants of hybridization by executing LS to the ***best individual*** in the ***current*** DE population first, and then applying the LS again after finding the ***new solution*** of DE.

3.1 Fmincon LS

The ***Fmincon*** method finds a constrained minimum of a scalar function of several variables starting at an initial estimate. This is generally referred to as constrained ***nonlinear optimization*** or ***nonlinear programming***. The minimum of constrained nonlinear multivariable function (eq. no 5)

$$\min_{x} f(x) \tag{5}$$

subject to

$$c(x) \leq 0$$

$$ceq(x) = 0$$

$$A \cdot x \leq b$$

$$Aeq \cdot x = beq$$

$$lb \leq x \leq ub$$

Where

- *x*, *b*, *beq*, *lb* and *ub* are vectors.
- *A* and *Aeq* are metrics.
- *c(x)* and *ceq(x)* are functions that return vectors.
- *f(x)* is a function that returns a scalar.

f(x), *c(x)*, and *ceq(x)* can be nonlinear functions [40].

3.1.1 Fmincon Function Description

$$x = fmincon\ (fun, x0, A, b, Aeq, beq, lb, ub\ ) \tag{6}$$

Starts at *x0* and finds a minimum *x* to the function described in *fun*() subject to the linear inequalities

$$A \cdot x \leq b$$

x0 can be a scalar, vector or matrix. It also minimizes *fun*() subject to the linear equalities

$$Aeq \cdot x = beq$$

as well as $A \cdot x \leq b$

Also defines a set of lower and upper bounds on the design variables, *x* , so that the solution is always in the range

$$lb \leq x \leq ub$$

Sets *Aeq* = [ ] and *beq* = [ ] if no equalities exist.

$$[x, fval] = fmincon(\dots) \quad (7)$$

Returns the value *fval* of the objective function *fun*() at the solution *x* [40].

3.2 Hybrid DE Version 1: LS around New Solution

Starting with the randomly created Differential Evolution population and reaching the ***selection*** stage of the DE method means that the algorithm has created the ***donor*** vector u_i and the decision of selecting each ***decision variable*** for the new individual depends mainly on the random shuffle of ***Crossover-rate*** (CR) value. S_i will be the newly created vector by the selection step in DE. While P_i is the original individual from current DE iteration population, the algorithm will decide whether S_i or P_i is going to be accepted in the new population after finishing ***selection*** step by comparing both of their ***cost values***, and choosing the better (lower) one. We can say that applying the Fmincon LS method around the area of the ***New Solution*** selected by the DE makes a good move due to the fact that if either P_i or S_i was selected to be the new individual in the next population, each of these two is supposed to have a

low cost value overall. This experimental point was taken to confirm that starting the local search method with a ***good solution*** could lead to ***better solutions*** around the area of it to fulfill the aim of reaching optimal, or near-optimal solutions.

```
Input: Population_size, Problem_size, Weighting_factor,
       Crossover_rate
Output: S_best
1 Population ← InitializePopulation(Population_size,
  Problem_size );
2 EvaluatePopulation ( Population );
3 S_best ← GetBestSolution(Population);
4 while ¬ StopCondition() do
5       NewPopulation ← Ø;
6       foreach P_i ∈ Population do
7             S_i ← NewSample (P_i, Population, Problem_size,
               Weighting_factor, Crossover_rate );
8             if Cost(S_i) ≤ Cost (P_i) then
9                   Fmincon (S_i);
10                  NewPopulation ← S_i;
11            else
12                  NewPopulation ← P_i;
13            end
14      Fmincon(NewPopulation_i);
15      end
16      Population ← NewPopulation_i;
17      EvaluatePopulation(Population);
18      S_best ← GetBestSolution(Population);
19 end
20 return S_best;
```

Figure 6: Pseudo Code of first variant of Hybrid DE
(Fmincon LS applied to New individual)

Where:

- $Population_{size}$: No. of individuals in one population.
- $Problem_{size}$: No. of decision variables in one vector.
- $Weighting_{factor}$: Differential weight F.
- $Crossover_{rate}$: CR factor.
- Population: Current generation of individuals.

- NewPopulation$_i$: The next generation of individuals.
- S_{best}: The best solution found so far.
- P_i: An individual in the current population.
- S_i: New individual vector found after applying DE process.
- InitializePopulation(): Returns randomly-generated population.
- EvaluatePopulation(): Returns fitness values of all the population individuals.
- GetBestSolution(): Returns the individual with minimum fitness value.
- StopCondition(): Stopping Criteria.
- NewSample(): Returns the trial vector y_i.
- Cost(): Returns the fitness value of one vector.
- Fmincon (): Returns the local optimum found after applying local search.

3.3 Hybrid DE Version 2: LS around Best individual in Current population

Initializing a population in the first iteration of the DE algorithm is carried out through a stochastic behavior, which may lead to very large differences in the cost values among individuals. The algorithm will start by ***evaluating*** the initialized individuals in order to find the one with the best cost value; S_{best}, which may not necessarily have a very good fitness value, but overall it is the best-so-far in the current population. Performing the Fmincon LS mechanism around the area of S_{best} can be considered in our experiment as a Hill-climbing [18] strategy. The LS is granted a starting point with an individual that may not be a good solution overall, but may lead to better solutions after reaching the local optimum. After LS is finished, the ***local optimum*** vector found in the area around S_{best} will be assigned to it, then it will proceed to perform the DE method.

```
Input: Population_size, Problem_size, Weighting_factor,
       Crossover_rate
Output: S_best
1 Population ← InitializePopulation(Population_size,
  Problem_size );
2 EvaluatePopulation ( Population );
3 S_best ← GetBestSolution(Population);
5 while ¬ StopCondition() do
6 Fmincon (S_best);
7       NewPopulation ← Ø;
8       foreach P_i ∈ Population do
9             S_i ← NewSample (P_i, Population, Problem_size,
               Weighting_factor, Crossover_rate );
10            if Cost(S_i) ≤ Cost (P_i) then
11                  NewPopulation ← S_i;
12            else
13                  NewPopulation ← P_i;
14            end
15      end
16      Population ← NewPopulation;
17      EvaluatePopulation(Population);
18      S_best ← GetBestSolution(Population);
19 end
20 return S_best;
```

Figure 7: Pseudo Code of second variant of Hybrid DE
(Fmincon LS applied to Best individual in Current Population)

Where:

- $Population_{size}$: No. of individuals in one population.
- $Problem_{size}$: No. of decision variables in one vector.
- $Weighting_{factor}$: Differential weight F.
- $Crossover_{rate}$: CR factor.
- Population: Current generation of individuals.
- NewPopulation$_i$: The next generation of individuals.
- S_{best}: The best solution found so far.
- P_i: An individual in the current population.
- S_i: New individual vector found after applying DE process.
- InitializePopulation(): Returns randomly-generated population.

- EvaluatePopulation(): Returns fitness values of all the population individuals.
- GetBestSolution(): Returns the individual with minimum fitness value.
- StopCondition(): Stopping Criteria.
- NewSample(): Returns the trial vector y_i.
- Cost(): Returns the fitness value of one vector.
- Fmincon (): Returns the local optimum found after applying local search.

3.4 Hybrid DE Version 3: LS around Best individual in Current population & around the New solution

The previous two hybrid DE methods proposed active usage of ***Hill-climbing*** strategy and employed the point of starting the LS with a good solution in order to reach better near-optimal fitness valued individuals. The third proposed method creation depended on ***fusing*** the two past strategies together. The algorithm will start by initializing the population and selecting the best individual S_{best}, then Fmincon LS is performed around S_{best} aiming to reach the local optimum. After first LS procedure is finished, the DE method proceeds until reaching the ***selection*** stage. LS will be performed with the starting solution P_i or S_i upon the decision of which one will be in the new population after comparing their fitness values. The concept of merging the two previous strategies and performing LS before and after the DE method is executed ensures performing the search in both exploration and exploiting behavior and may have better performance and result in serving the aim of enhancing DE performance overall.

```
Input: Population_size, Problem_size, Weighting_factor,
       Crossover_rate
Output: S_best
1 Population ← InitializePopulation(Population_size,
  Problem_size );
2 EvaluatePopulation ( Population );
3 S_best ← GetBestSolution(Population);
4 while ¬ StopCondition() do
5 Fmincon (S_best);
6       NewPopulation ← Ø;
7       foreach P_i ∈ Population do
8             S_i ← NewSample (P_i, Population, Problem_size,
9              Weighting_factor, Crossover_rate );
10            if Cost(S_i) ≤ Cost (P_i) then
11                  NewPopulation ← S_i;
12            else
13                  NewPopulation ← P_i;
14            end
15      Fmincon (NewPopulation_i);
16      end
17      Population ← NewPopulation;
18      EvaluatePopulation(Population);
19      S_best ← GetBestSolution(Population);
20 end
21 return S_best;
```

Figure 8: Pseudo Code of third variant of Hybrid DE
(Fmincon LS applied to Best Individual in Current Population and around the New individual)

Where

- $Population_{size}$: No. of individuals in one population.
- $Problem_{size}$: No. of decision variables in one vector.
- $Weighting_{factor}$: Differential weight F.
- $Crossover_{rate}$: CR factor.
- Population: Current generation of individuals.
- $NewPopulation_i$: The next generation of individuals.
- S_{best}: The best solution found so far.
- P_i: An individual in the current population.

- S_i: New individual vector found after applying DE process.
- InitializePopulation(): Returns randomly-generated population.
- EvaluatePopulation(): Returns fitness values of all the population individuals.
- GetBestSolution(): Returns the individual with minimum fitness value.
- StopCondition(): Stopping Criteria.
- NewSample(): Returns the trial vector y_i.
- Cost(): Returns the fitness value of one vector.
- Fmincon (): Returns the local optimum found after applying local search.

3.5 Summary

Three different variants of ***Hybrid DE*** were proposed. Each variant featured combining the DE algorithm with the ***Fmincon*** LS tool. the ***first*** version of Hybrid DE was conducted based on the concept of starting the LS with a good fitness valued solution with the aim of reaching better solutions around its neighborhood. The ***second*** version of the proposed Hybrid DE was based on ***a Hill-climbing*** idea by applying the LS in the area of the best fitness valued solution in the randomly initialized ***current*** population. Finally, the ***last*** proposed version of Hybrid DE featured the ***fusion*** of both first and second versions by applying LS around the best solution in current population area, then also applying the LS after DE execution was finished and the new individual of the population was found.

Chapter 4

EXPERIMENTAL RESULTS

For the purpose of demonstrating the differences between results, the first step of the experiment was to execute optimization of 15 black-box benchmark functions [14] using the original Differential Evolution Algorithm. Then, we experimented with all of the three proposed variants of *Hybrid DE* on the benchmark functions with 10 and 30 dimensions. The empirical results, supported with comprehensive secondary data obtained from the single-objective problem optimization experiment revealed that optimization process using an EA was influenced by the support of local optimization method. The difference between results obtained from implementing the original DE algorithm and results of the three variants of *Hybrid DE* in both 10 and 30 dimensions was huge.

4.1 CEC'15 Expensive Optimization Test Problems

By downloading the Matlab Codes for CEC'15 test suite [41], all the problems were installed and treated as ***black-box optimization problems*** and without any prior knowledge. Neither the analytical equations nor the problem characteristics extracted from analytical equations were allowed to be seen or studied [14].

4.1.1 Common Definitions

All test functions are minimization problems defined as follows in (eq. no 8):

$$\min f(x), x = [x_1, x_2, \ldots, x_D]^T \quad (8)$$

Where D is the ***dimension*** of the problem. all search ranges are pre-defined for all test functions as $[-100, 100]^D$. The termination criterion is based on reaching the maximum number of ***function evaluations*** according to each dimension [14].

4.1.2 Experimental Settings

- Number of independent runs: 20
- Maximum number of exact function evaluations:
    - 10-dimension: 500
    - 30- dimension: 1500
- *Initialization:* using a problem-independent initialization method.
- *Termination:* Terminate when reaching the maximum number of exact function evaluations or the error value ($F_i^* - F_i(x^*)$) is smaller than 10^{-3} [14].

Practically, modern stochastic optimization methods such, as EAs, are considered computationally expensive because they require many thousands of objective function calls to the simulation codes in order to locate a near-optimal solution. EAs are also time consuming since their search cycle time is directly proportional to the number of calls of the expensive fitness function [52].

Table 1: Summary of CEC'15 expensive optimization test problems [14]

Categories	No	Functions	Related Basic Functions	F_i*
Unimodal Functions	**1**	Rotated Bent Cigar Function	Bent Cigar Function	100
	2	Rotated Discus Function	Discus Function	200
Simple Multimodal Functions	**3**	Shifted and Rotated Weierstrass Function	Weierstrass Function	300
	4	Shifted and Rotated Schwefel's Function	Schwefel's Function	400
	5	Shifted and Rotated Katsuura Function	Katsuura Function	500
	6	Shifted and Rotated HappyCat Function	HappyCat Function	600
	7	Shifted and Rotated HGBat Function	HGBat Function	700
	8	Shifted and Rotated Expanded Griewank's puls Rosenbrock's Function	Griewank's Function Rosenbrock's Function	800
	9	Shifted and Rotated Expanded Scaffer's F6 Function	Expanded Scaffer's F6 Function	900
Hybrid Functions	**10**	Hybrid Function 1 (N=3)	Schwefel's Function Rastrigin's Function High Conditioned Elliptic Function	1000
	11	Hybrid Function 2 (N=4)	Griewank's Function Weierstrass Function Rosenbrock's Function Scaffer's F6 Function	1100
	12	Hybrid Function 3 (N=5)	Katsuura Function HappyCat Function Griewank's Function Rosenbrock's Function Schwefel's Function Ackley's Function	1200
Composition Functions	**13**	Composite Function 1 (N=5)	Rosenbrock's Function High Conditioned Elliptic Function Bent Cigar Function Discus Function	1300
	14	Composite Function 2 (N=3)	Schwefel's Function Rastrigin's Function High Conditioned Elliptic Function	1400
	15	Composite Function 3 (N=5)	HGBat Function Rastrigin's Function Schwefel's Function Weierstrass Function High Conditioned Elliptic Function	1500

4.2 Results

The three proposed variants of ***Hybrid DE*** were tested distinctly for optimizing CEC2015 single objective problems in ***Dimension 10*** featuring only 500 function evaluations and in ***Dimension 30*** featuring a larger number of function evaluations up to 1500 times. The results of both dimensions intended to demonstrate a large improvement from the primary original DE solutions with high adjacency to the optimal F_i* results.

4.2.1 Hybrid DE variants in Dimension 10

Table 2 data are obtained from ***dimension 10*** implementation of original DE, followed by the results of versions 1, 2 and 3 of ***Hybrid DE***. The best results out of

Table 2: Best results of Hybrid DE versions in Dimension 10 (20 runs)

#	F_i*	DE	Hybrid DE V.1	Hybrid DE V. 2	Hybrid DE V. 3
1	100	3E+09	**100.051678**	100.1647	100.0613
2	200	31156.62115	200.0142	200.0143	**200.0112**
3	300	309.9202	308.5307	308.2675	**307.9077**
4	400	1684.209	1022.401	846.0667	**833.142**
5	500	501.4494256	500.1929	500.2548	**500.1859**
6	600	602.7712	**600.0904**	600.2087	600.096
7	700	724.67707	700.3239	**700.2243**	700.2266
8	800	1861.4487	**801.5499**	807.4774	802.66
9	900	903.98633	903.3542	903.1379	**902.3923**
10	1000	143000.37	1323.485	**1005.393**	1229.117
11	1100	1111.3384	**1105.355**	1105.896	1105.974
12	1200	1260.531	1261.581	**1243.613**	1266.532
13	1300	1691.2347	**1612.527**	**1612.527**	**1612.527**
14	1400	1614.3457	**1595.872**	1595.915	1602.9
15	1500	1941.8559	1591.424	1655.731	**1526.254**

F_i*: Optimal solution of the i^{th} problem

20 distinct ***runs*** for 15 single objective problems' optimization are demonstrated.

The results of the analyses of Table 2 revealed significant differences between the original DE solutions and the ***Hybrid DE*** solutions which clearly tend to get close to the optimal values in some of the problems, but do not in the others. Overall, both of version 1 and version 2 of **Hybrid DE** results tend to show an apparent improvement in the quality of solutions, while fusing both of their concepts in version 3 of the algorithm demonstrates ***best*** experiment solutions in most of the 15 problems.

Table 3: Best results of Hybrid DE versions in Dimension 30 (20 runs)

#	F_i*	DE	Hybrid DE V.1	Hybrid DE V. 2	Hybrid DE V. 3
1	100	8.9E+08	100.1575171	**100.0616349**	100.1114339
2	200	21757.5	200.015307	200.0131796	**200.0099**
3	300	308.1646	308.7714	307.6581	**307.4994**
4	400	1459.511	**653.1932**	932.2627	764.1311
5	500	501.617	**500.0504**	500.2256	500.0967
6	600	601.8938	600.3841	600.2546	**600.1087**
7	700	708.4069	700.1476	700.192	**700.1407**
8	800	822.8918	**804.109**	813.425952	807.28347
9	900	903.8918	**902.5808**	903.1077	903.0219
10	1000	68398.03	**1021.345084**	1166.67	1147.98849
11	1100	1109.029	1106.377	1108.743	**1106.031**
12	1200	1299.266	1238.05	1229.16	**1224.18**
13	1300	1630.788	**1612.527**	**1612.527**	**1612.527**
14	1400	1609.633	**1588.785**	1599.846	1597.523
15	1500	1771.877	**1573.638**	1695.892	1584.807

F_i*: Optimal solution of the i^{th} problem

Both of the ***Unimodal functions*** results in all three Hybrid DE variants in ***Dimension 10*** reached near-optimal solutions with relatively small differences from optimality. ***Multimodal functions*** were mixed between problems which had very small differences from optimal solutions; problems no. ***5, 6, 7 and 8*** while the rest of the problems' results in the same category showed big figured numbers. Finally, ***Hybrid functions*** which included problems ***10, 11 and 12***, and Composite functions that included problems ***13, 14 and 15***, was able to improve the primary result of original DE algorithms, but not reaching any near optimal solutions in any of these problems.

4.2.2 Hybrid DE variants in Dimension 30

Table 3 demonstrates the ***dimension 30*** implementation of original DE results, and the solutions of version 1, 2 and 3 of ***Hybrid DE***. The best results out of **20 runs** which were executed separately for 15 single objective problems' optimization are demonstrated. F_i* are the optimal solutions for problems.

The results of the analyses in Table 3 revealed significant differences between the original DE solutions and the ***Hybrid DE*** solutions. Reaching near-optimal solutions overall seem to be dependent on starting with a good solution in a way. Hybrid DE ***version 1*** here owns the highest number of **best** experiment solutions with ***version 2*** of Hybrid DE only resulting the best in **problem No. 1**. This may indicate that most of the time, when starting the local exploitation with a good solution, the procedure could lead to better optimization results.

Both of the ***Unimodal functions*** results in all three Hybrid DE variants in ***Dimension 30*** reached near-optimal solutions with relatively small differences from optimality. ***Multimodal functions*** were mixed between problems which had very small differences from optimal solutions; problems no. ***5, 6, 7 and 9*** while the rest of the

problems' results in the same category showed big figured numbers. Finally, ***Hybrid functions*** which included problems ***10, 11 and 12***, and Composite functions that included problems ***13, 14 and 15***, was able to improve the primary result of original DE algorithms, but not reaching any near optimal solutions in any of these problems.

Table 4: CPU Time for Best results of Hybrid DE versions in D30 (sec/20 runs)

#	DE	Hybrid DE V.1	Hybrid DE V. 2	Hybrid DE V. 3
1	16.25	232.5	9122.5	4575
2	14.688	107.812	7068.75	3160
3	18.438	283.75	11489.376	6575.626
4	13.126	137.5	9054.688	4479.376
5	14.062	313.75	30628	19136.562
6	13.438	154.688	10578.75	4131.876
7	26.25	159.688	8951.876	29144
8	13.438	234.688	15270.312	5054.688
9	11.876	175.626	5880.626	10126.562
10	13.988	163.126	6792.812	3552.5
11	18.75	164.376	22848	6493.75
12	26.25	329.376	6464.688	6888.75
13	3.576	173.126	7666.25	6291.562
14	14.688	110	3362.188	4926.25
15	17.5	151.25	14462.188	6115.938

CPU time, demonstrated in Table 4, was calculated distinctively from Hybrid DE versions implementation in Dimension 30 for each problem per single run. Then, the total time for 20 runs for each Hybrid DE version optimization of a single problem was calculated. ***Version 2*** of the proposed method appeared to be the most time-consuming compared with the two other versions, yet did not reach good solutions.

4.3 Comparison with Literature

The findings of our experiment with Hybrid DE are consistent to some extent with the past studies on CEC 2105 problem optimization. A number of the previously proposed methods for the solutions of the same group of problems show clear relation to the results of ***Hybrid DE***. DEsPA [9] is a technique proposed by Noor Awad et al. which featured using a memory-based structure to adapt control parameters. L-SHADE [10] is another method proposed by Shu-Mei Gou et al. it depended on the population resizing concept. Neurodynamic Differential Evolution [11] proposed a linear population size reduction DE dependent on modification of success history parameter within the concept of neurodynamic. Moreover, Self-adaptive Dynamic Multi-Swarm Particle [12] which differs primarily from original PSO in the employment of of ***self-adaptive strategy*** of parameters. Finally, the Hybrid Cooperative Co-evolution (hCC), which consists the concept of separating the variables into groups and continue in adopting different algorithms within the cooperative co-evolution (CC) framework [13].

Tables 5, 6 and 7 demonstrated below compare the ***error rates*** of the versions 1, 2 and 3 of ***Hybrid DE*** with the ***error rates*** from literature in ***dimension 30.***

4.3.1 Hybrid DE with LS around New solution

The data demonstrated in Table 5 are obtained from literature representing error rates of the previously proposed methods results for CEC2015 expensive problems optimization. The comparison conducted between Hybrid DE version 1 error rates of application to the same group of problems.

Table 5: Comparison of dimension 30 error rates of Hybrid DE V. 1 results with literature

#	F_i*	Hybrid DE V. 1	DEsPA	SPS-L-SHADE-EI	LSHADE-ND	sDMS-PSO	hCC
1	100	1.58E-01	**0.00E+00**	**0.00E+00**	**0.00E+00**	0.000513	1.56E-13
2	200	1.53E-02	**0.00E+00**	**0.00E+00**	**0.00E+00**	0.000807	2.84E-14
3	300	**8.77E+00**	2.00E+01	2.00E+01	2.000E+01	19.9998	2.01E+01
4	400	2.53E+02	3.98E+00	**1.05E-02**	4.9750E+00	24.87397	5.22E+00
5	500	**5.04E-02**	9.48E+02	6.58E+02	7.5217E+02	1587.52	2.59E+02
6	600	**3.84E-01**	2.72E+01	2.68E+01	4.4798E+01	564.0676	4.50E+01
7	700	**1.48E-01**	1.07E+00	6.23E-01	3.6485E+00	5.829585	2.25E+00
8	800	4.11E+00	3.40E+00	**2.07E+00**	2.3365E+00	538.468	1.15E+01
9	900	**2.58E+00**	1.16E+02	1.02E+02	1.022E+02	102.5592	1.06E+02
10	1000	**2.13E+01**	**3.50E+01**	1.48E+02	3.3222E+02	2613.849	4.15E+02
11	1100	**6.38E+00**	2.01E+02	3.00E+02	4.000E+02	306.3833	3.18E+02
12	1200	**3.81E+01**	1.08E+02	1.02E+02	1.0295E+02	103.4556	1.04E+02
13	1300	3.13E+02	6.93E+01	2.56E-02	2.5584E-02	89.6766	**2.51E-02**
14	1400	**1.89E+02**	2.73E+04	3.11E+04	3.1070E+04	17469.59	3.11E+04
15	1500	**7.36E+01**	2.73E+02	1.00E+02	1.000E+02	100	1.00E+02

By examining the comparison between error rates demonstrated in Table 5, it can be concluded that the highest number of ***best*** problem optimization results belong to the ***first*** version of ***Hybrid DE*** method. The table showed superior performance of Hybrid DE from optimizing results of **10** out of **15** problems, which is the highest between all the methods from literature. In problems number **1 and 2**, the error rates of ***Hybrid DE version 1*** appeared to be very close to optimality. The rest of the problems' results varied between generally small differences and extreme differences from the optimal values.

4.3.2 Hybrid DE with LS around Best individual in Current Population

The data demonstrated in Table 6 are obtained from literature representing error rates of the previously proposed methods results for CEC2015 expensive problems optimization. The comparison conducted between Hybrid DE version 2 error rates of application to the same group of problems.

Table 6: Comparison of dimension 30 Hybrid DE V. 2 results with literature

#	F_i*	Hybrid DE V. 2	DEsPA	SPS-L-SHADE-EIG	LSHADE-ND	sDMS-PSO	hCC
1	100	6.16E-02	**0.00E+00**	**0.00E+00**	**0.00E+00**	0.000513	1.56E-13
2	200	1.32E-02	**0.00E+00**	**0.00E+00**	**0.00E+00**	0.000807	2.84E-14
3	300	**7.66E+00**	2.00E+01	2.00E+01	2.000E+01	19.9998	2.01E+01
4	400	5.32E+02	3.98E+00	**1.05E-02**	4.9750E+00	24.87397	5.22E+00
5	500	**2.26E-01**	9.48E+02	6.58E+02	7.5217E+02	1587.52	2.59E+02
6	600	**2.55E-01**	2.72E+01	2.68E+01	4.4798E+01	564.0676	4.50E+01
7	700	**1.92E-01**	1.07E+00	6.23E-01	3.6485E+00	5.829585	2.25E+00
8	800	1.34E+01	3.40E+00	**2.07E+00**	2.3365E+00	538.468	1.15E+01
9	900	**3.11E+00**	1.16E+02	1.02E+02	1.022E+02	102.5592	1.06E+02
10	1000	1.67E+02	**3.50E+01**	1.48E+02	3.3222E+02	2613.849	4.15E+02
11	1100	**8.74E+00**	2.01E+02	3.00E+02	4.000E+02	306.3833	3.18E+02
12	1200	**2.92E+01**	1.08E+02	1.02E+02	1.0295E+02	103.4556	1.04E+02
13	1300	3.13E+02	6.93E+01	2.56E-02	2.5584E-02	89.6766	**2.51E-02**
14	1400	**2.00E+02**	2.73E+04	3.11E+04	3.1070E+04	17469.59	3.11E+04
15	1500	1.96E+02	2.73E+02	**1.00E+02**	1.000E+02	100	1.00E+02

According to Table 6, the ***second*** proposed version of ***Hybrid DE*** had the largest number of best optimization results in **8** out of **15** CEC expensive problems. The error rates of both problems number **1 and 2** tend to be very close to the optimal value. Problems number ***4*****, 8, 10, 13** and **15** results show a considerably big difference from the optimal values of problem solutions.

4.3.3 Hybrid DE with LS around Best individual in Current Population & around the New solution

The data demonstrated in Table 7 are obtained from literature representing error rates of the previously proposed methods results for CEC2015 expensive problems optimization. The comparison conducted between Hybrid DE version 3 error rates of application to the same group of problems.

Table 7: Comparison of dimension 30 Hybrid DE V. 3 results with literature

#	F_i*	Hybrid DE V. 3	DEsPA	SPS-L-SHADE-EIG	LSHADE-ND	sDMS-PSO	hCC
1	100	1.11E-01	**0.00E+00**	**0.00E+00**	**0.00E+00**	0.000513	1.56E-13
2	200	9.90E-03	**0.00E+00**	**0.00E+00**	**0.00E+00**	0.000807	2.84E-14
3	300	**7.50E+00**	2.00E+01	2.00E+01	2.000E+01	19.9998	2.01E+01
4	400	3.64E+02	3.98E+00	**1.05E-02**	4.9750E+00	24.87397	5.22E+00
5	500	**9.67E-02**	9.48E+02	6.58E+02	7.5217E+02	1587.52	2.59E+02
6	600	**1.09E-01**	2.72E+01	2.68E+01	4.4798E+01	564.0676	4.50E+01
7	700	**1.41E-01**	1.07E+00	6.23E-01	3.6485E+00	5.829585	2.25E+00
8	800	7.28E+00	3.40E+00	**2.07E+00**	2.3365E+00	538.468	1.15E+01
9	900	**3.02E+00**	1.16E+02	1.02E+02	1.022E+02	102.5592	1.06E+02
10	1000	1.48E+02	**3.50E+01**	1.48E+02	3.3222E+02	2613.849	4.15E+02
11	1100	**6.03E+00**	2.01E+02	3.00E+02	4.000E+02	306.3833	3.18E+02
12	1200	**2.42E+01**	1.08E+02	1.02E+02	1.0295E+02	103.4556	1.04E+02
13	1300	3.13E+02	6.93E+01	**2.56E-02**	2.5584E-02	89.6766	**2.51E-02**
14	1400	**1.98E+02**	2.73E+04	3.11E+04	3.1070E+04	17469.59	3.11E+04
15	1500	**8.48E+01**	2.73E+02	1.00E+02	1.000E+02	100	1.00E+02

Looking at Table 7, the comparison between the ***third*** proposed version of ***Hybrid DE*** show that it could reach the ***best*** results in optimizing **9** out of **15** problems of the CEC expensive problems, which is the largest between all the other methods in the literature. The results of problem **1 and 2** are very close to the optimal values of the problems. The rest of the problems error rates are between considerably small and large differences from optimal values.

4.4 Friedman Ranking Test

The ***Friedman Test*** is a non-parametric statistical test developed by Milton Friedman. It is used to check the statistical similarities in treatments across multiple test attempts. The procedure involves ranking each row together, then considering the values of ranks by columns [42]. The P-value indicator represents the difference between the ranked functions statistically. The smaller the p-value is, the bigger the statistical differences between the ranked methods are [54].

The ranking procedure was used in order to assess the quality of the proposed Hybrid DE. A comparison among the three proposed variants of Hybrid DE in ***dimension 10*** and ***dimension 30*** opposed to the original DE results, and between each Hybrid DE proposed variant in ***dimension 30*** with literature studies was conducted using ***Friedman test***.

Table 8: Friedman Ranking between Hybrid DE versions in D10

Rank	Function
1	Hybrid DE with LS around Best individual in Current Population & around the New solution (V.3)
2	Hybrid DE with LS around New solution (V.1)
3	Hybrid DE with LS around Best individual in Current Population (V.2)
4	DE

p-value = 3.1041e-05

according to the Ranking between the three proposed Hybrid DE variants including the original DE results in ***dimension 10*** demonstrated in Table 8, ***version 3*** of the Hybrid DE method; applying LS around the best individual in current population and around the new solution, was ahead of the other two proposed hybrid methods. The P-value is very close to zero which indicates obvious difference between their performance statistically.

Table 9: Friedman Ranking between Hybrid DE versions in D30

Rank	Function
1	Hybrid DE with LS around New solution (V.1)
1	Hybrid DE with LS around Best individual in Current Population & around the New solution (V.3)
2	Hybrid DE with LS around Best individual in Current Population (V.2)
3	DE

p-value = 1.19198e-06

In ***dimension 30***, ***version 1*** and **3** of Hybrid DE had the same level of performance according to Friedman Test ranking in Table 9. Both of version ***1 and 3*** had the best rank before the second version of proposed Hybrid DE followed by the original DE.

Table 10: Friedman Ranking between Hybrid DE V.1 and Literature in D30

Rank	Function
1	Hybrid DE V.1
2	SPS-L-SHADE-EIG
3	DEsPA
4	LSHADE-ND
5	hCC
6	sDMS-PSO

p-value = 0.0057213

Table 10 ranking results showed that between all literature results in ***dimension 30***, compared with the ***version 1*** of Hybrid DE. The proposed Hybrid DE method showed the best performance overall.

Version 2 and ***version 3*** of Hybrid DE had the ***second best*** rank in performance compared with literature results in ***dimension 30*** in Table 11 and Table 12. The best overall results of optimizing the majority of CEC'15 problems was ***SPS-L-SHADE-EIG*** method compared to both of the last two versions of Hybrid DE.

Table 11: Friedman Ranking between Hybrid DE V.2 and Literature in D30

Rank	Function
1	SPS-L-SHADE-EIG
2	Hybrid DE V. 2
3	LSHADE-ND
4	DEsPA
5	hCC
6	sDMS-PSO

p-value = 0.022563

Table 12: Friedman Ranking between Hybrid DE V.3 and Literature in D30

Rank	Function
1	SPS-L-SHADE-EIG
2	Hybrid DE V. 3
3	DEsPA
4	LSHADE-ND
5	hCC
6	sDMS-PSO

p-value = 0.0070841

Chapter 5

CONCLUSION

5.1 Summary of the Study

The experiment proposed ***Hybrid Differential Evolution*** method for the purpose of optimizing the ***CEC2015*** of 15 Benchmark of single objective problems [14] . The merging consisted of the DE global optimization that served as an exploration factor with the employment of a local search technique as an exploitation factor. The base of the idea focused on fusing both ***diversification-based*** and ***intensification-based*** algorithms that may lead to better optimized problems' solutions. Three different versions of ***Hybrid DE*** were proposed and the experiment was conducted for all in both ***dimension 10*** and ***dimension 30***. Finally, we compared the findings of our proposed ***Hybrid DE*** algorithm with the previous research with the aim of optimizing CEC2015 problems.

5.2 Conclusions

The patterns of results from the experiment appear to fit criteria supporting the hypothesis of using the local search methods for the aim of single objective problem optimization. The results of the ***Hybrid DE*** tended to show little differences with the optimal solutions in some of the findings. Comparison with the previously proposed techniques suggests that the proposed method owns the upper hand in the number of best solutions. The primary conclusion of this experiment is that using ***local search*** techniques with the aim of optimization has a powerful impact resulting in better solutions.

5.3 Implications of the Study

The main contribution of this study is the support of the concept that using ***local search*** methods for the aim of optimization could result in better problem solutions. Our findings contribute practical implications and insights into ***hybridizing global optimization*** methods with local search techniques. The study established the strategy of fusing ***DE algorithm*** with ***Fmincon*** local search tool that was found to contribute effectively to our aim of the study.

5.4 Implications for Further Research

Further research can be based on the concept of ***hybridizing DE*** with another local search method that may be more efficient in the competing procedure. Another suggestion could be using the different ***variants*** of the DE in the experiment which may have a stronger impact on the findings in the future.

REFERENCES

[1] Sorensen, K.; Sevaux, M.; Glover, F. (2017). *A History of Metaheuristics*. January, 2017 from the World Wide Web: https://www.researchgate.net/publication/315811561_A_History_of_Metaheuristics

[2] Karaboga, D.; Okdem, S. (2004). *A Simple and Global Optimization Algorithm For Engineering Problems: Differential Evolution Algorithm*. Turk J Elec Engin, Volume.12, No.1.

[3] Storn, R. (1996). *Differential Evolution Design of an IIR-Filter*. Proceedings of IEEE International Conference on Evolutionary Computation. DOI: 10.1109/ICEC.1996.542373

[4] Chiou, J.P.; Wang, F.Sh. (1998). *A hybrid method of differential evolution with application to optimal control problems of a bioprocess system*. IEEE World Congress on Computational Intelligence, Proceedings of IEEE International Conference on Evolutionary Computation. DOI: 10.1109/ICEC.1998.700101

[5] Back, TH.; Foussette, C.; Krause, P. (2013). *Contemporary Evolution Strategies*. Springer.

[6] Thomas Back, (1996). *Evolutionary Algorithms in Theory and Practice* (pp. 1). Oxford University Press, ISBN 0-19-509971-0 (hard cover).

[7] Darrell Whitley, (2001). An *overview of evolutionary algorithms: practical issues and common pitfalls*. Elsevier Science, PII: S 0950-5849(01)001-884. DOI: 10.1016/S0950-5849(01)00188-4

[8] Coello, C.; Lamont, G.; Veldhuizen, D. (2007). *Evolutionary Algorithms For Solving Multi-Objective Problems* (pp. 4). Springer Scienc, ISBN 978-0-387-33254-3

[9] Awad, N.; Ali, M.; Reynods, R. (2015). *A Differential Evolution Algorithm with Success-based Parameter Adaptation for CEC2015 Learning-based Optimization.* IEEE Congress on Evolutionary Computation (CEC), PII: 978-1-4799-7492-4. DOI: 10.1109/CEC.2015.7257012

[10] Gou, Sh.; Yang, Ch.; Tsai, J.; Hsu, P. (2015). *A Self-Optimization Approach of L-SHADE Incorporated with Eigenvector-Based Crossover and Successful-Parent-Selecting Framework on CEC 2015 Benchmark Set*. IEEE Congress on Evolutionary Computation (CEC), PII: 978-1-4799-7492-4. DOI: 10.1109/CEC.2015.7256999

[11] Sallam, K.; Sarker, R.; Essam, D.; Elsayed, S. (2015). *Neurodynamic Differential Evolution Algorithm and Solving CEC2015 Competition Problems.* IEEE Congress on Evolutionary Computation (CEC), PII: 978-1-4799-7492-4. DOI: 10.1109/CEC.2015.7257003

[12] Liang, J.; Gou, L.; Liu, R.; Qu, B. (2015). *A Self-adaptive Dynamic Particle Swarm Optimizer*. IEEE Congress on Evolutionary Computation (CEC), PII: 978-1-4799-7492-4. DOI: 10.1109/CEC.2015.7257290

[13] Mohammed El-Abd. (2015). *Hybrid Cooperative Co-evolution For The CEC15 Benchmarks*. IEEE Congress on Evolutionary Computation (CEC), PII: 978-1-4799-7492-4. DOI: 10.1109/CEC.2015.7257006

[14] Chen, Q.; Liu. B.; Zhang, Q.; Liang, J. J.; Suganthan, P. N.; Qu, B. Y. (2015). *Problem Definitions and Evaluations Criteria for CEC 2015 Special Session on Bound Constrained Single-Objective Computationally Expensive Numerical Optimization.*

[15] Das, S.; Suganthan, P. N. (2010). *Differential Evolution: A Survey of the State-of-the-Art*. IEEE transactions on Evolutionary Computation, vol 15, issue 1. DOI: 10.1109/TEVC.2010.2059031

[16] Storn R. (2008). *Differential Evolution Research – Trends and Open Questions*. In: Chakraborty U.K. (eds) Advances in Differential Evolution. Studies in Computational Intelligence, vol 143. Springer, Berlin, Heidelberg. ISBN: 978-3-540-68830-3. DOI: 10.1007/978-3-540-68830-3_1

[17] Rangaiah, G. P.; Sharma, Sh. (eds) *Differential Evolution In Chemical Engineering: Developments And Applications*. (2017). Advances in Process

System Engineering, vol 6. World Scientific Publishing. ISBN 97898/3207516

[18] Brownlee J. (2011). *Clever Algorithms: Nature-Inspired Programming Recipes*. ISBN:978-1-4467-8506-5. Link: http://blog.shuo1.com/zms/books/program/Clever%20Algorithms.pdf

[19] Storn, R. (1999). *System Design by Constraint Adaptation and Differential Evolution*. IEEE transactions on Evolutionary Computation, vol 3, No 1. DOI: 10.1109/4235.752918

[20] Abbass, H. A.; Sarker, R.; Newton, Ch. (2001). *PDE: A Pareto-frontier Differential Evolution Approach for Multi-objective Optimization Problems*. IEEE Proceedings of the 2001 Congress on Evolutionary Computation. DOI: 10.1109/CEC.2001.934295

[21] Abbass, H. A. (2002). *The Self-Adaptive Pareto Differential Evolution Algorithm*. IEEE Proceedings of the 2002 Congress on Evolutionary Computation. DOI: 10.1109/CEC.2002.1007033

[22] Noman, N.; Iba, H. (2008). *Accelerating Differential Evolution Using an Adaptive Local Search*. IEEE Transactions on Evolutionary Computation, vol 12, issue 1. DOI: 10.1109/TEVC.2007.895272

[23] Qian, B.; Wang, L.; Hu, R. et al. (2008). *A hybrid Differential Evolution method for Permutation Flow-shop Scheduling*. Int J Adv Manuf Technol (2008) 38: 757. Springer. DOI: 10.1007/s00170-007-1115-8

[24] Zhang, Ch.; Ning, J.; Lu, Sh. et al. (2009). *A Novel hybrid Differential Evolution and Particle Swarm Optimization Algorithm for Unconstrained Optimization*. Operations Research Letters, vol 37, issue 2. Elsevier. DOI: 10.1016/j.orl.2008.12.008

[25] Gong, W.; Cai, Z.; Ling, Ch. X. (2010). *DE/BBO: a hybrid Differential Evolution with Biogeography-based Optimization for Global Numerical Optimization*. Soft Computing (2010) 15: 645. Springer. DOI: 10.1007/s00500-010-0591-1

[26] Liao, T. W. (2010). Two *hybrid Differential Evolution Algorithms for Engineering Design Optimization*. Applied Soft Computing, vol 10, issue 4 (pp. 1188-1199). Elsevier. DOI: 10.1016/j.asoc.2010.05.007

[27] Wang, Y.; Cai Z.; Zhang, Q. (2011). *Differential Evolution With Composite Trial Vector Generation Strategies and Control Parameters*. IEEE, DOI: 10.1109/TEVC.2010.2087271

[28] Wang, L.; Li, L. P. (2012). *A coevolutionary Differential Evolution with Harmony Search for Reliability-Redundancy Optimization*. Expert Systems Optimization, vol 39, issue 5 (pp. 5271-5278). Elsevier. DOI: 10.1016/j.eswa.2011.11.012

[29] Yildiz, A. R. (2012). *Hybrid Taguchi-Differential Evolution Algorithm for Optimization of Multi-pass Turning Operations*. Applied Soft Computing, vol 13, issue 3 (pp. 1433-1439). Elsevier. DOI: 10.1016/j.asoc.2012.01.012

[30] Zhang, J.; Sanderson, A. C. (2009). *JADE: Adaptive Differential Evolution with Optional External Archive*. IEEE transactions on Evolutionary Computation, vol 13, issue 5. DOI: 10.1109/TEVC.2009.2014613

[31] Wang, X.; Zhao, Sh. (2013). *Differential Evolution Algorithm with Self-Adaptive Population Resizing Mechanism*. Mathematical Problems in Engineering, vol 2013. DOI: http://dx.doi.org/10.1155/2013/419372

[32] Zheng, Y. J.; Xu, X. L.; Ling, H. F. et al. (2015). *A hybrid Fireworks Optimization Method with Differential Evolution Operators*. Neurocomputing, vol 148 (pp. 75-82). DOI: 10.1016/j.neucom.2012.08.075

[33] Trivedi, A.; Srinivasan, D.; Biswas, S.; Riendl, T. (2015). *Hybridizing Genetic Algorithm with Differential Evolution for solving the Unit Commitment Scheduling problem*. Swarm and Evolutionary Computation, vol 23 (pp. 50-64). Elsevier. DOI: 10.1016/j.swevo.2015.04.001

[34] Parouha, R. P.; Das, K. N. (2016). *A Memory based Differential Evolution Algorithm for Unconstrained Optimization*. Applied Soft Computing, vol 38 (pp. 501-517). Elsevier. DOI: 10.1016/j.asoc.2015.10.022

[35] Li, G.; Lin, Q.; Cui, L. et al. (2016). *A novel hybrid Differential Evolution Algorithm with Modified CoDE and JADE*. Applied Soft Computing, vol 47 (pp. 577-599). Elsevier. DOI: 10.1016/j.asoc.2016.06.011

[36] Kukkonen S., Coello Coello C.A. (2017). *Generalized Differential Evolution for Numerical and Evolutionary Optimization*. In: Schütze O., Trujillo L., Legrand P., Maldonado Y. (eds) NEO 2015. Studies in Computational Intelligence, vol 663. Springer, Cham. DOI: 10.1007/978-3-319-44003-3_11

[37] Wong, L.; Liu, W.; Ho, Ch. M.; Ding, X. (2017). *Continuous Adaptive Population Reduction (CARP) for Differential Evolution Optimization*. SLAS TECHNOLOGY: Translating Life Sciences Innovation, vol 22, issue 3 (pp. 289 – 305). DOI: 10.1177/2472630317690318

[38] Luke, S. (2009). *Essentials in Metaheurisitics*. Available at http://cs.gmu.edu/~sean/book/metaheuristics/

[39] Martinez-Estudillo, A. C.; Hervas-Martinez, C.; Martinez-Estudillo, F. J.; Garcia-Pedrajas, N. (2005). *Hybridization of Evolutionary Algorithms and Local Search by means of a Clustering method*. IEEE transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), vol 36, issue 3. DOI:10.1109/TSMCB.2005.860138

[40] *Fmincon , Optimization Toolbox PDF*. Available at http://bwrcs.eecs.berkeley.edu/Classes/icdesign/ee141_s03/Project/Project1_solutions/fmincon.pdf

[41] *Codes for CEC'15 test suite*. Available at http://www.nut.edu.sg/home/EPNSugan/index_files/CEC2015

[42] *Friedman Ranking Test* website: http://vassarstats.net/textbook/ch15a.html

[43] Box, G. E. P. (1957). *Journal of the Royal Statistical Society*. Series C (Applied Statistics) Vol. 6, No. 2, pp. 81-10. DOI: 10.2307/2985505

[44] Rechenberg, I. (1989). *Evolution Strategy: Nature's Way of Optimization*. In: Bergmann H.W. (eds) Optimization: Methods and Applications, Possibilities and Limitations. Lecture Notes in Engineering, vol 47. Springer, Berlin, Heidelberg

[45] Feo, T. A.; Resende, M. G. C. (1995). *J Glob Optim* 6: 109. DOI: https://doi.org/10.1007/BF01096763

[46] Colorni, A.; Dorigo, M.; Maniezzo, V. (1992). *Distributed optimization by ant colonies*, In Toward a Practice of Autonomous Systems, The MIT Press, pp. 134-142.

[47] Oliveto, P.S., He, J. & Yao, X. (2007). *Int J Automat Comput* 4: 281. DOI: https://doi.org/10.1007/s11633-007-0281-3

[48] Auger, A; Doerr, B. (2011). *Theory of Randomized Search Heuristics Foundations and recent Developments*. World Scientific Publishing Co. ISBN-13 978-981-4282-66-6. ISBN-10 981 4282-66-9

[49] Friedrich, T.; He J.; Hebbinghaus, N.; Neumann, F.; Witt, C. (2010). *Approximating Covering Problems by Randomized Search Heuristics Using Multi-Objective Models*. Massachusetts Institute of Technology. Evolutionary Computation, vol 18, issue 4, pp 617-633.

[50] Moscato, P.; Cotta, C. (2002). *Memetic Algorithms*. Handbook of Applied Optimization. DOI: http://www.lcc.uma.es/~ccottap/papers/memetic_HAAM.pdf

[51] Deb, K.; Agrawal, R. B. (1994). *Simulated Binary Crossover For Continuous Search Space*. Complex Systems. Convenor, Technical Reports. Department of Mechanical Engineering, Indian Institute of Technology.

[52] Zhou, Z.; Ong, Y.S.; Lim, M.H. et al. Soft Comput (2007) 11: 957. DOI: https://doi.org/10.1007/s00500-006-0145-8

[53] Droste S., Jansen T., Wegener I. (1998). *On the optimization of Unimodal Functions with the (1+1) Evolutionary Algorithm*. In: Eiben A.E., Bäck T., Schoenauer M., Schwefel HP. (eds) Parallel Problem Solving from Nature — PPSN V. PPSN 1998. Lecture Notes in Computer Science, vol 1498. Springer, Berlin, Heidelberg

[54] Sherinov, Z.; Ünveren, A. (2017). *Multi-objective Imperialistic Competitive Algorithm with Multiple Non-Dominated Sets for the Solution of Global Optimization Problems*. Soft Comput. DOI: https://doi.org/10.1007

APPENDIX

Appendix A: Introduction of the CEC'15 expensive optimization test problems [14]

This section defines the set of basic f_i functions which were used to construct the set of CEC 2015 expensive optimization problems. Then, it is followed by the detailed definitions of 15 expensive functions F(x) which are categorized into four groups: ***Unimodal Functions***, ***Simple Multimodal Functions***, ***Hybrid Functions*** and ***Composite Functions***.

f_i (x): ith basic function used to construct the expensive function.

F(x) : expensive function.

n : number of basic functions. The bigger n is, the more complex F(x).

D : dimension.

1. Definitions of basic functions

1. Bent Cigar Function

$$f_1(\boldsymbol{x}) = \boldsymbol{x}_1^2 + 10^6 \sum_{i=2}^{D} \boldsymbol{x}_i^2 \quad (1)$$

2. Discus Function

$$f_2(\boldsymbol{x}) = \mathbf{10^6}\, \boldsymbol{x}_{\mathbf{1}}^{\mathbf{2}} + \sum_{\boldsymbol{i}=\mathbf{2}}^{\boldsymbol{D}} \boldsymbol{x}_{\boldsymbol{i}}^{\mathbf{2}} \qquad \textbf{(2)}$$

3. Weierstrass Function

$$f_3(\boldsymbol{x}) = \sum_{\boldsymbol{i}=\mathbf{1}}^{\boldsymbol{D}} \left(\sum_{\boldsymbol{k}=\mathbf{0}}^{\boldsymbol{k}\,\boldsymbol{max}} \left[\boldsymbol{a}^{\boldsymbol{k}} \cos\left(\mathbf{2\pi}\boldsymbol{b}^{\boldsymbol{k}}(\boldsymbol{x}_{\boldsymbol{i}} + \mathbf{0.5})\right)\right]\right) - \boldsymbol{D} \sum_{\boldsymbol{k}=\boldsymbol{D}}^{\boldsymbol{k}\,\boldsymbol{max}} \left[\boldsymbol{a}^{\boldsymbol{k}} \cos\left(\mathbf{2\pi}\boldsymbol{b}^{\boldsymbol{k}} \cdot \mathbf{0.5}\right)\right] \quad \textbf{(3)}$$

where a=0.5, b=3, and kmax=20.

4. Modified Schwefel's Function

$$f_4(\boldsymbol{x}) = \mathbf{418.9829} \times \boldsymbol{D} - \sum_{\boldsymbol{i}=\mathbf{1}}^{\boldsymbol{D}} \boldsymbol{g}(\boldsymbol{z}_{\boldsymbol{i}}), \qquad z_i = x_i + 4.209687462275036e+002$$

$$g(z_i) = \begin{cases} \boldsymbol{z}_{\boldsymbol{i}} \sin\left(|\boldsymbol{z}_{\boldsymbol{i}}|^{\mathbf{1\backslash 2}}\right) & \text{if } |z_i| \le 500 \\ \left(\mathbf{500} - \boldsymbol{mod}(\boldsymbol{z}_{\boldsymbol{i}}, \mathbf{500})\right) \sin\left(\sqrt{|\mathbf{500} - \boldsymbol{mod}(\boldsymbol{z}_{\boldsymbol{i}}, \mathbf{500})|}\right) - \frac{(z_i - 500)^2}{10000D} & \text{if } z_i > 500 \\ \left(\boldsymbol{mod}(|\boldsymbol{z}_{\boldsymbol{i}}|, \mathbf{500}) - \mathbf{500}\right) \sin\left(\sqrt{|\boldsymbol{mod}(|\boldsymbol{z}_{\boldsymbol{i}}|, \mathbf{500}) - \mathbf{500}|}\right) - \frac{(z_i + 500)^2}{10000D} & \text{if } z_i < -500 \end{cases} \qquad \textbf{(4)}$$

5. Katsuura Function

$$f_5(\boldsymbol{x}) = \frac{\mathbf{10}}{\boldsymbol{D}^{\mathbf{2}}} \prod_{\boldsymbol{i}=\mathbf{1}}^{\boldsymbol{D}} \left(\mathbf{1} + \boldsymbol{i} \sum_{\boldsymbol{j}=\mathbf{1}}^{\mathbf{32}} \frac{\left|\mathbf{2}^{\boldsymbol{j}}\boldsymbol{x}_{\boldsymbol{i}} - \boldsymbol{round}(\mathbf{2}^{\boldsymbol{j}}\boldsymbol{x}_{\boldsymbol{i}})\right|}{\mathbf{2}^{\boldsymbol{j}}}\right)^{\frac{\mathbf{10}}{\boldsymbol{D}^{\mathbf{1.2}}}} - \frac{\mathbf{10}}{\boldsymbol{D}^{\mathbf{2}}} \qquad \textbf{(5)}$$

6. HappyCat Function

$$f_6(\boldsymbol{x}) = \left|\sum_{i=1}^{D} x_i^2 - D\right|^{1/4} + (0.5\sum_{i=1}^{D} x_i^2 + \sum_{i=1}^{D} x_i) / D + 0.5 \quad \textbf{(6)}$$

7. HGBat Function

$$f_7(\boldsymbol{x}) = \left|(\sum_{i=1}^{D} x_i^2)^2 - (\sum_{i=1}^{D} x_i)^2\right|^{1/2} + (0.5\sum_{i=1}^{D} x_i^2 + \sum_{i=1}^{D} x_i) / D + 0.5 \quad \textbf{(7)}$$

8. Expanded Griewank's plus Rosenbrock's Function

$$f_8(\boldsymbol{x}) = f_{11}(f_{10}(x_1,x_2)) + f_{11}(f_{10}(x_2,x_3)) + \ldots + f_{11}(f_{10}(x_{D-1},x_D)) + f_{11}(f_{10}(x_D,x_1)) \quad \textbf{(8)}$$

9. Expanded Scaffer's F6 Function

$$g(\boldsymbol{x},\boldsymbol{y}) = 0.5 + \frac{(sin^2(\sqrt{x^2+y^2})-0.5)}{(1+0.001(x^2+y^2))^2}$$

$$f_9(\boldsymbol{x}) = g(x_1,x_2) + g(x_2,x_3) + \ldots + g(x_{D-1},x_D) + g(x_D,x_i)) \quad \textbf{(9)}$$

10. Rosenbrock's Function

$$f_{10}(\boldsymbol{x}) = \sum_{i=1}^{D-1}(100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2) \quad \textbf{(10)}$$

11. Griewank's Function

$$f_{11}(\boldsymbol{x}) = \sum_{i=1}^{D} \frac{x_i^2}{4000} - \prod_{i=1}^{D} \cos(\frac{x_i}{\sqrt{i}} + 1 \quad \textbf{(11)}$$

12. Rastrigin's Function

$$f_{12}(\boldsymbol{x}) = \sum_{i=1}^{D}(\boldsymbol{x}_i^2 - \mathbf{10}\cos(\mathbf{2}\pi\boldsymbol{x}_i) + \mathbf{10}) \quad \textbf{(12)}$$

13. High Conditioned Elliptic Function

$$f_{13}(\boldsymbol{x}) = \sum_{i=1}^{D}(\mathbf{10}^{\mathbf{6}})^{\frac{i-1}{D-1}}\boldsymbol{x}_i^2 \quad \textbf{(13)}$$

14. Ackley's Function

$$f_{14}(\boldsymbol{x}) = -\mathbf{20}\exp\left(-\mathbf{0.2}\sqrt{\frac{1}{D}\sum_{i=1}^{D}\boldsymbol{x}_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(\mathbf{2}\pi\boldsymbol{x}_i)\right) + \mathbf{20} + \boldsymbol{e} \quad \textbf{(14)}$$

2. Definitions of the CEC'15 Expensive Test Suite

2.1 Unimodal Functions

all search ranges are pre-defined for all test functions as $[-100, 100]^D$ and F_i^* is the set of optimal values. Where D is the *dimension* of the problem.

1) Rotated Bent Cigar Function

D = 10, D = 30, F_1^* = **100**

$$F_1(\boldsymbol{x}) = f_1(\mathbf{M}(\boldsymbol{x}\text{-}\circ_1)) + F_1^* \quad \textbf{(15)}$$

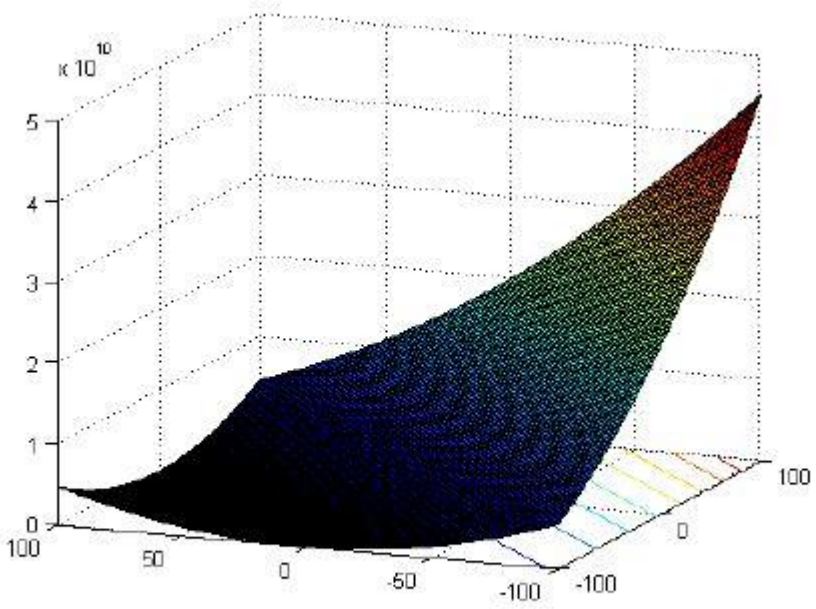

Figure 1. 3-*D* map for 2-*D* function

Properties:

- Unimodal
- Non-separable
- Smooth but narrow ridge

2) Rotated Discus Function

D = 10, D = 30, F_2* = **200**

$$F_2(\boldsymbol{x}) = f_2(\mathbf{M}(\boldsymbol{x}\text{-}\circ_2)) + F_2^* \quad \textbf{(16)}$$

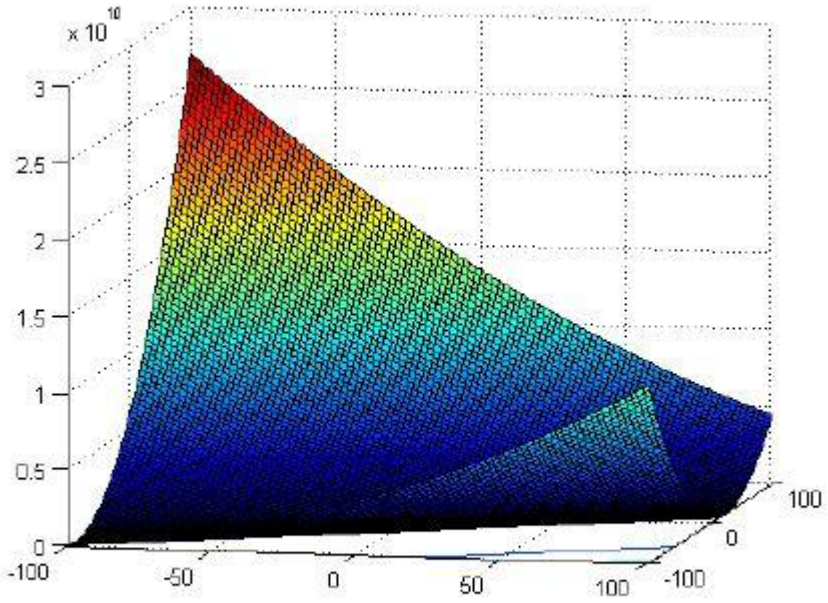

Figure 2. *3-D map for 2-D function*

Properties:

- Unimodal
- Non-seperable
- With one sensitive direction

2.2 Simple Multimodal Functions

3) Shifted and Rotated Weirestrass Function

D = 10, D = 30, F_3^* = **300**

$$F_3(\boldsymbol{x}) = f_3(\mathbf{M}(\frac{\mathbf{0.5}(\boldsymbol{x}-\mathbf{3})}{\mathbf{100}})) + F_3^* \quad \textbf{(17)}$$

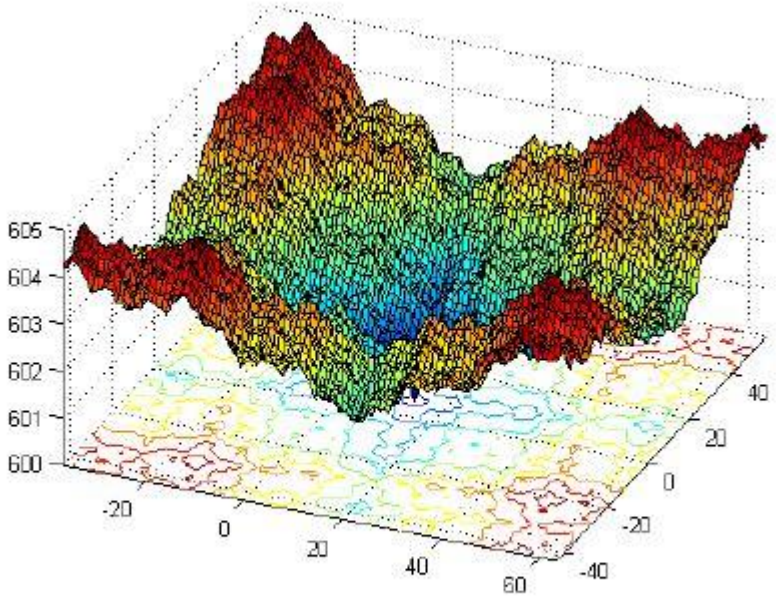

Figure 3. *3-D map for 2-D function*

Properties:

- Multi-modal
- Non-separable
- Continuous but differentiable only on a set of points

4) Shifted and Rotated Schwefel's Function

D = 10, D = 30, F_4* = **400**

$$F_4(\boldsymbol{x}) = f_4(\mathbf{M}(\frac{\mathbf{1000}(\boldsymbol{x}-\mathbf{°4})}{\mathbf{100}})) + F_4^* \qquad \textbf{(18)}$$

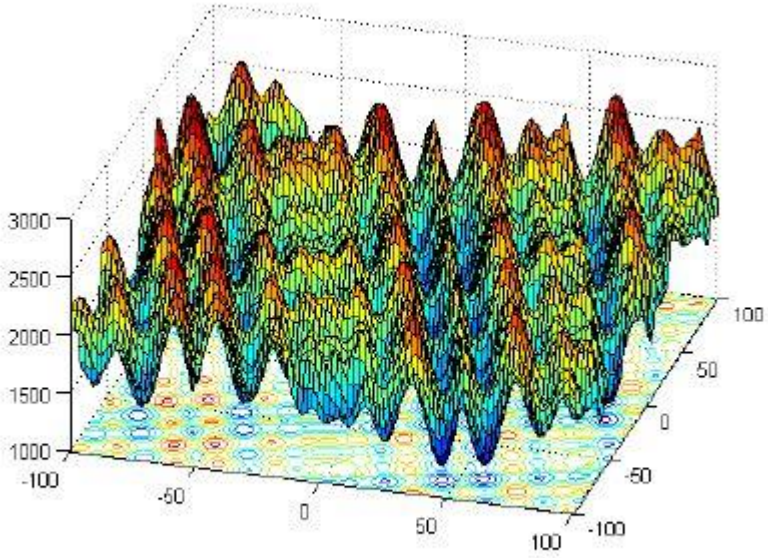

Figure 4(a). *3-D map for 2-D function*

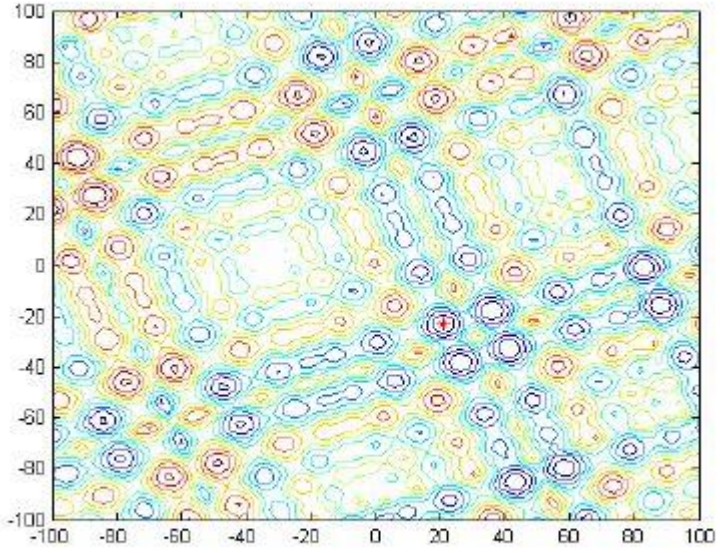

Figure 4(b).*Contour map for 2-D function*

Properties:

- Multi-modal
- Non-separable

- Local optima's number is huge and second better local optimum is far from the global optimum

5) Shifted and Rotated Katsuura Function

D = 10, D = 30, F_5^* = **500**

$$F_5(\boldsymbol{x}) = f_5(\mathbf{M}(\frac{\mathbf{5}(\boldsymbol{x}-\mathbf{°5})}{\mathbf{100}})) + F_5^* \quad \textbf{(19)}$$

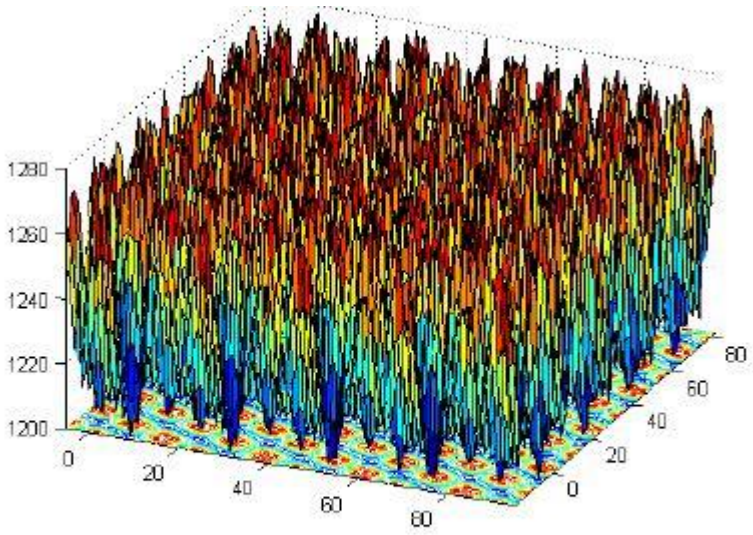

Figure 5(a). *3-D map for 2-D function*

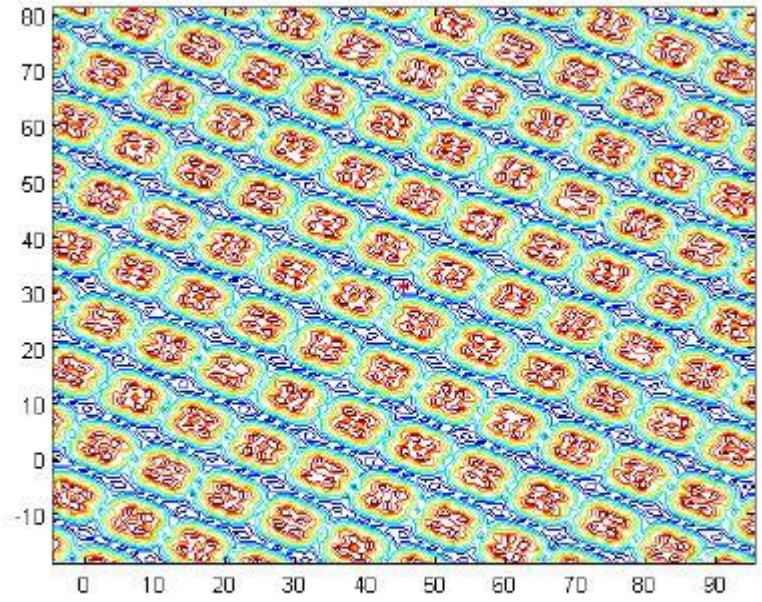

Figure 5(b). *Contour map for 2-D function*

Properties:

- Multi-modal
- Non-separable
- Continuous everywhere yet differentiable nowhere

6) Shifted and Rotated HappyCat Function

D = 10, D = 30, F_6^* = **600**

$$F_6(\boldsymbol{x}) = f_6(\mathbf{M}(\frac{\mathbf{5(}\boldsymbol{x}\mathbf{-°6)}}{\mathbf{100}})) + F_6^* \qquad \textbf{(20)}$$

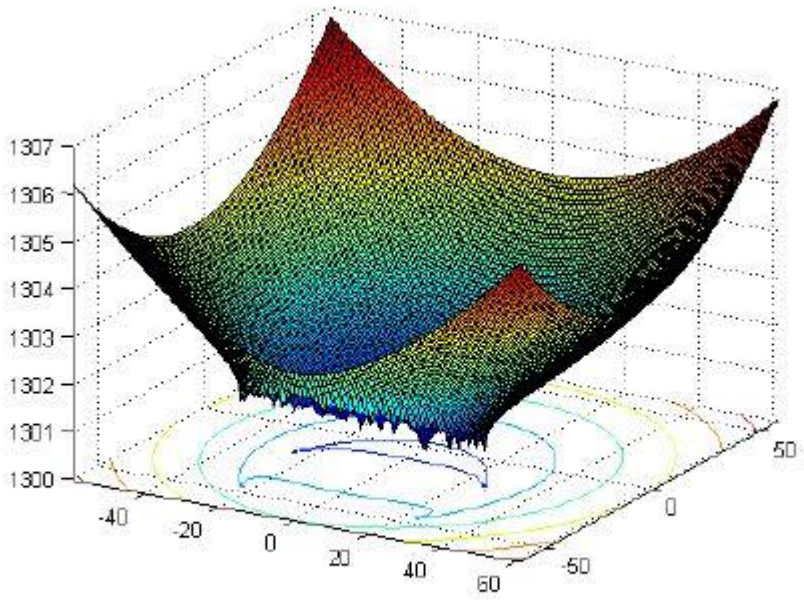

Figure 6(a). *3-D map for 2-D function*

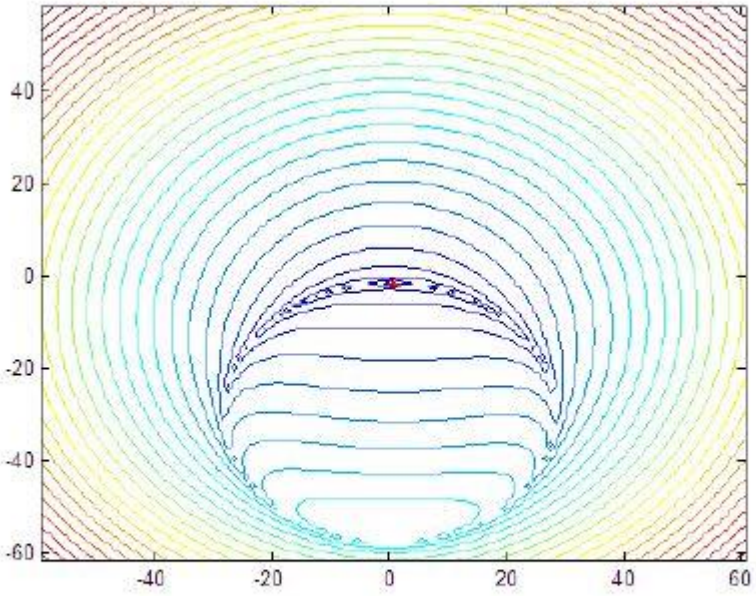

Figure 6(b).*Contour map for 2-D function*

Properties:

- Multi-modal

- Non-separable

7) Shifted and Rotated HGBat Function

D = 10, D = 30, F_7^* = **700**

$$F_7(\boldsymbol{x}) = f_7(\mathbf{M}(\frac{\mathbf{5}(\boldsymbol{x}-\mathbf{°7})}{\mathbf{100}})) + F_7^* \quad \textbf{(21)}$$

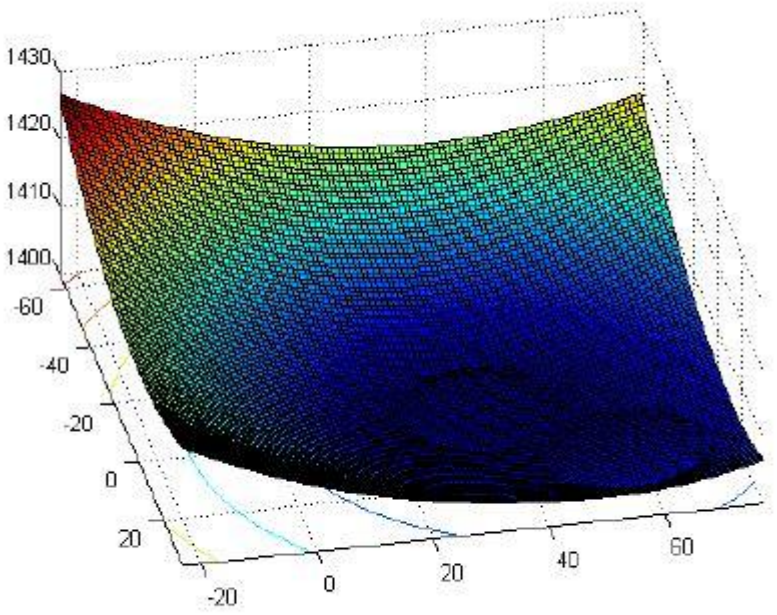

Figure 7(a). *3-D map for 2-D function*

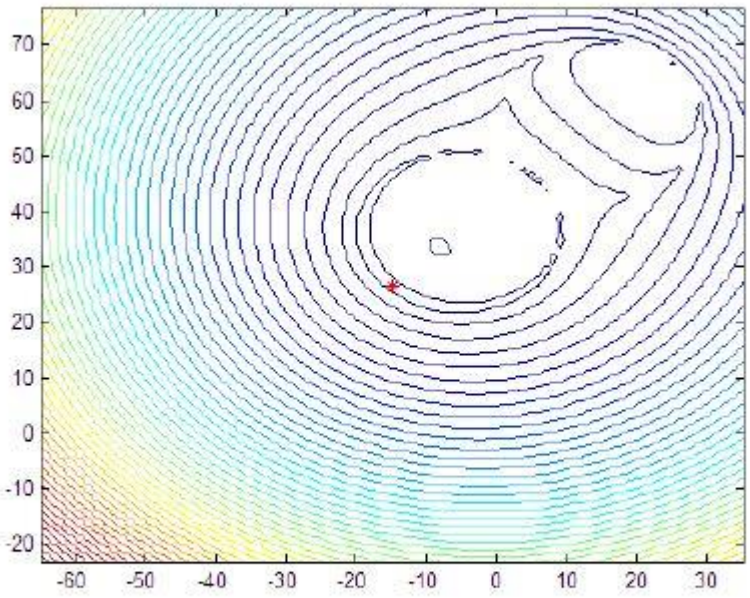

Figure 7(b). *Contour map for 2-D function*

Properties:

- Multi-modal
- Non-separable

8) Shifted and Rotated Expanded Griewank's plus Rosenbrock's Function

D = 10, D = 30, F_8^* = **800**

$$F_8(\boldsymbol{x}) = f_8(\mathbf{M}(\frac{\mathbf{5}(\boldsymbol{x}-\mathbf{°8})}{\mathbf{100}}) + 1) + F_8^* \qquad \textbf{(22)}$$

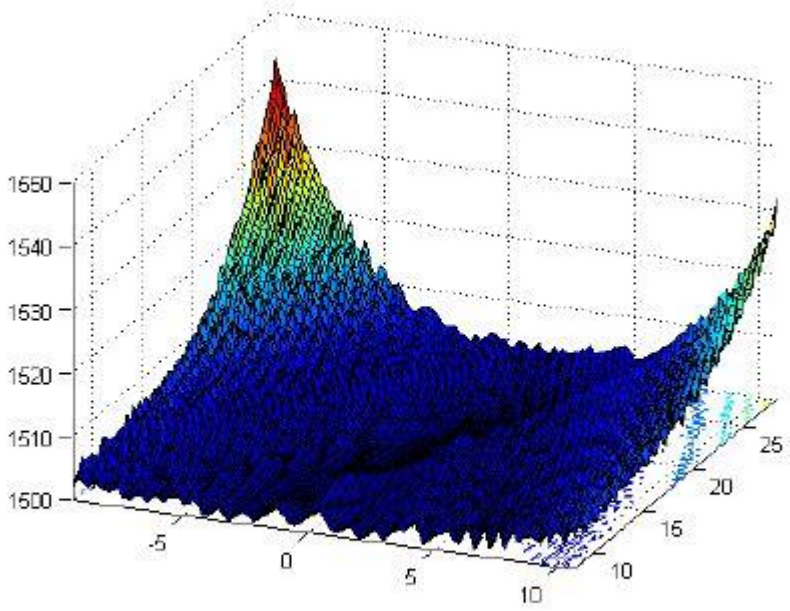

Figure 8(a). *3-D map for 2-D function*

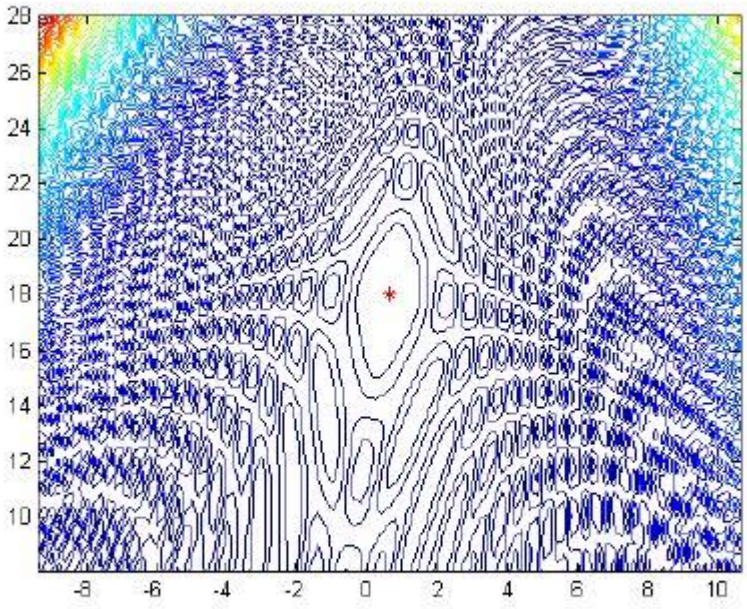

Figure 8(b). *Contour map for 2-D function*

Properties:

- Multi-modal

- Non-separable

9) Shifted and Rotated Expanded Scaffer's F6 Function

D = 10, D = 30, $F_9^* = \mathbf{900}$

$$F_9(\boldsymbol{x}) = f_9(\mathbf{M}(\boldsymbol{x} - \circ_9) + 1) + F_9^* \quad \textbf{(23)}$$

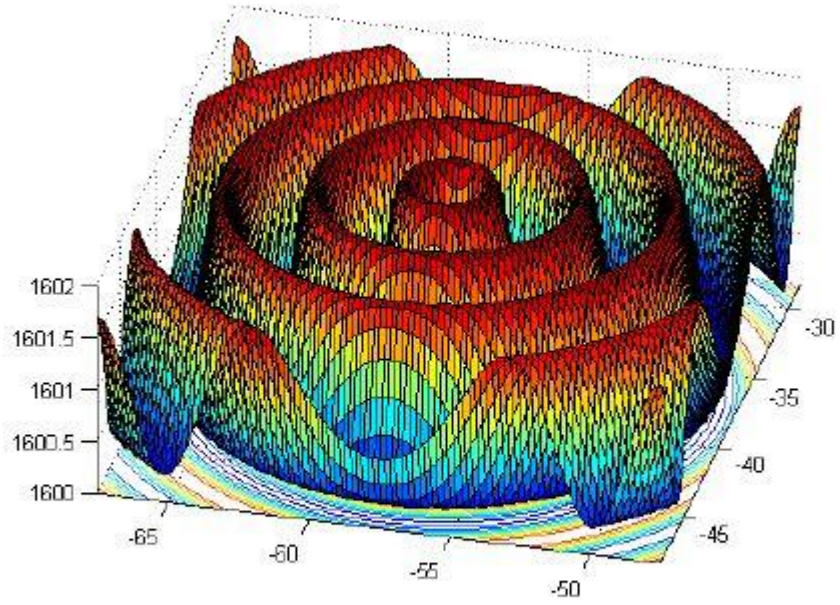

Figure 9. *3-D map for 2-D function*

Properties:

- Multi-modal
- Non-separable

2.3 Hybrid Functions

In real-world optimization problems, different subsets of the variables may have different properties. In this set of hybrid functions, the variables are randomly divided into some subsets and then different basic functions are used for different subsets.

$$F(\boldsymbol{x}) = g_1(\mathbf{M}_1\boldsymbol{z}_1) + g_2(\mathbf{M}_1\boldsymbol{z}_1) + \ldots + g_N(\mathbf{M}_N\boldsymbol{z}_N) + F^*(\boldsymbol{x}) \quad \textbf{(24)}$$

$F(\boldsymbol{x})$: hybrid function

$g_i(\boldsymbol{x})$: i[th] basic function used to construct the hybrid function

N: number of basic functions
$\boldsymbol{z} = [\, \boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_N \,]$
$\boldsymbol{z}_1 = [\, \boldsymbol{y}_{S1}, \boldsymbol{y}_{S2}, \ldots, \boldsymbol{y}_{Sm} \,], \boldsymbol{z}_2 = [\, \boldsymbol{y}_{Sm+1}, \boldsymbol{y}_{Sm+2}, \ldots, \boldsymbol{y}_{Sm+n2} \,], \ldots,$

$$\boldsymbol{z}_N = [\, \boldsymbol{y}_{S\sum_{i=1}^{N-1} n_i + 1}, \boldsymbol{y}_{S\sum_{i=1}^{N-1} n_i + 2}, \ldots, \boldsymbol{y}_{S_D} \,] \quad \textbf{(25)}$$

where, $\boldsymbol{y} = \boldsymbol{x} - \circ_i$ and $S = randperm(1: D)$

p_i: used to control the percentage of $g_i(\boldsymbol{x})$

n_i: dimension for each basic function $\sum_{i=1}^{N} ni = D$

$$\boldsymbol{n}_1 = \lceil \boldsymbol{p}_1 \boldsymbol{D} \rceil, \boldsymbol{n}_2 = \lceil \boldsymbol{p}_2 \boldsymbol{D} \rceil, \ldots, \boldsymbol{n}_{N-1} = \lceil \boldsymbol{p}_{N-1} \boldsymbol{D} \rceil, \boldsymbol{n}_N = D - \sum_{i=1}^{N-1} \boldsymbol{n}_i \quad \textbf{(26)}$$

10) Hybrid Function 1 (N=3)

D = 10, D = 30, F_{10}* = **1000**

p = [0.3,03.0.4]
g_1: Modified Schwefel's Function f_4
g_2: Rastrigin's Function f_{12}
g_3: High Conditioned Elliptic Function f_{13}

11) Hybrid Function 2 (N=4)

D = 10, D = 30, F_{11}* = **1100**

p = [0.2, 0.2, 0.3, 0.3]
g_1: Griewank's Function f_{11}
g_2: Weierstrass Function f_3
g_3: Rosenbrock's Function f_{10}
g_4: Scaffer's F6 Function f_9

12) Hybrid Function 3 (N=5)

D = 10, D = 30, F_{12}* = **1200**

p = [0.1, 0.2, 0.2, 0.2, 0.3]
g_1: Katsuura Function f_5
g_2: HappyCat Function f_6
g_3: Expanded Griewank's plus Rosenbrock's Function f_8
g_4: Modified Schwefel's Function f_4
g_5: Ackley's Function f_{14}

2.4 Composite Functions

$$F(\boldsymbol{x}) = \sum_{i=1}^{N}\{\boldsymbol{\omega_i} * [\boldsymbol{\lambda_i g_i}(\boldsymbol{x}) + \boldsymbol{bias_i}]\} + \boldsymbol{f} * \qquad \textbf{(27)}$$

$F(x)$: composition function
$g_i(x)$: i^{th} basic function used to construct the composition function
N: number of basic functions
$\circ_i$: new shifted optimum position for each $g_i(x)$, define the global and local optima's position.
$bias_i$: defines which optimum is global optimum
σ_i: used to control each $g_i(x)$'s coverage range, a small σ_i give a narrow range for that $g_i(x)$
λ_i: used to control each $g_i(x)$'s height
w_i: weight value for each $g_i(x)$, calculated as below

$$w_i = \frac{1}{\sqrt{\sum_{j=1}^{D}(x-o_{ij})^2}} \exp\left(-\frac{\sum_{j=1}^{D}(x_j-o_{ij})^2}{2D\sigma_i^2}\right) \quad \textbf{(28)}$$

Then normalize the weight $\varpi_i = \begin{cases} 1 & j = i \\ 0 & j \neq i \end{cases}$ $for\ j = 1, 2, \ldots, N$, $f(x) = bias_i + f*$

The optimum which has the smallest *bias* value is the global optimum. The composition function merges the properties of the sub-functions better and maintains continuity around the global/local optima.

13) Composition Function 1 (N=5)

D = 10, D = 30, F_{13}* = **1300**

N=5, σ = [10, 20, 30, 40, 50]
λ = [1, le-6, le-26, le-6, le-6]
bias = [0,100,200, 300, 400]
g_1: Rotated Rosenbrock's Function f_{10}
g_2: High Conditioned Elliptic Function f_{13}
g_3: Rotated Bent Cigar Function f_1
g_4: Rotated Discus Function f_2
g_5: High Conditioned Elliptic Function f_{13}

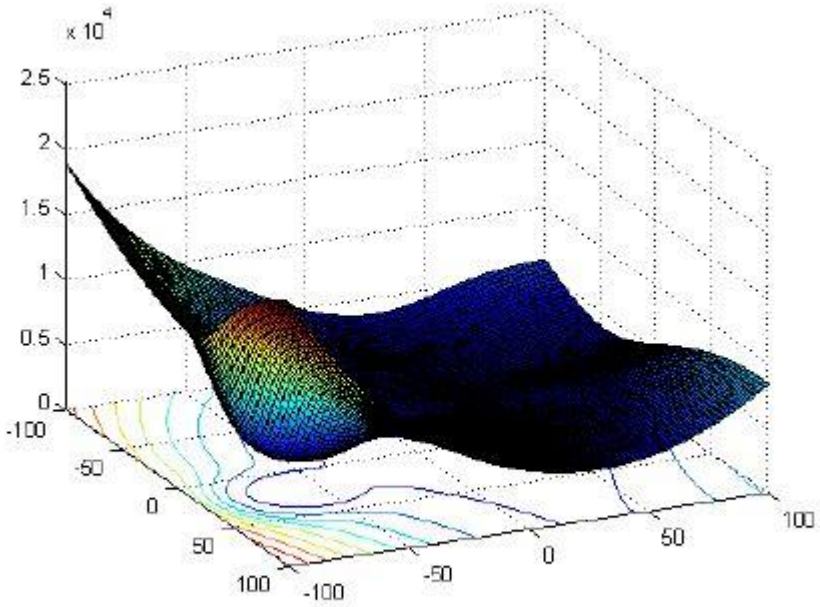

Figure 10(a). *3-D map for 2-D function*

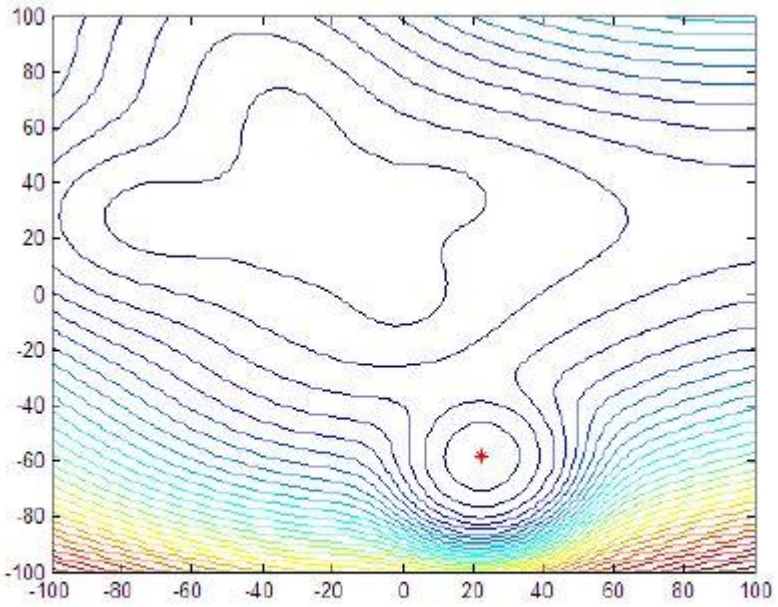

Figure 10 (b).*Contour map for 2-D function*

Properties:

- Multi-modal

- Non-separable

- Asymmetrical

- Different properties around different local optima

14) Composition Function 2 (N=3)

D = 10, D = 30, F_{14}* = **1400**

N = 3
$\boldsymbol{\sigma}$ = [10, 30, 50]
$\boldsymbol{\lambda}$ = [0.25, 1, le-7]
bias = [0, 100, 200]
g_1: Rotated Schwefel's Function f_4
g_2: Rotated Rastrigin's Function f_{12}
g_3: Rotated High Conditioned Elliptic Function f_{13}

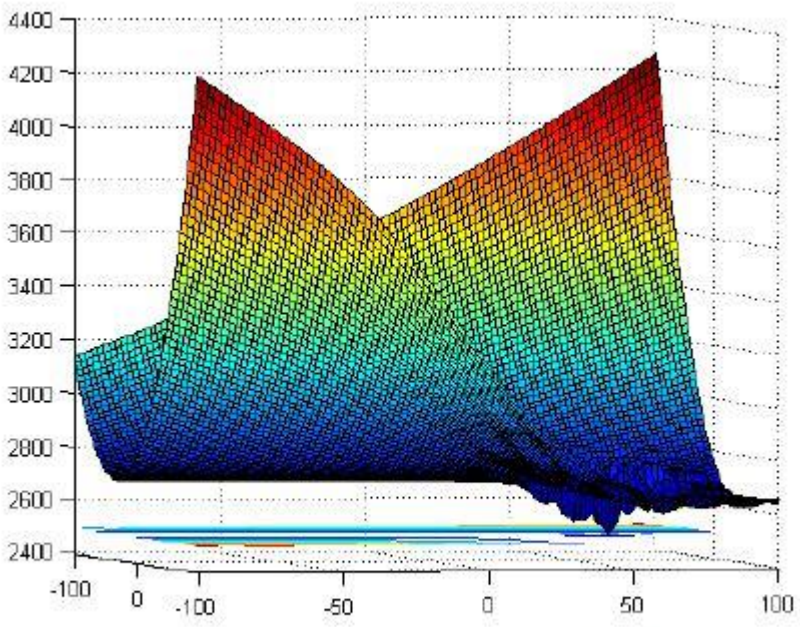

Figure 11(a). *3-D map for 2-D function*

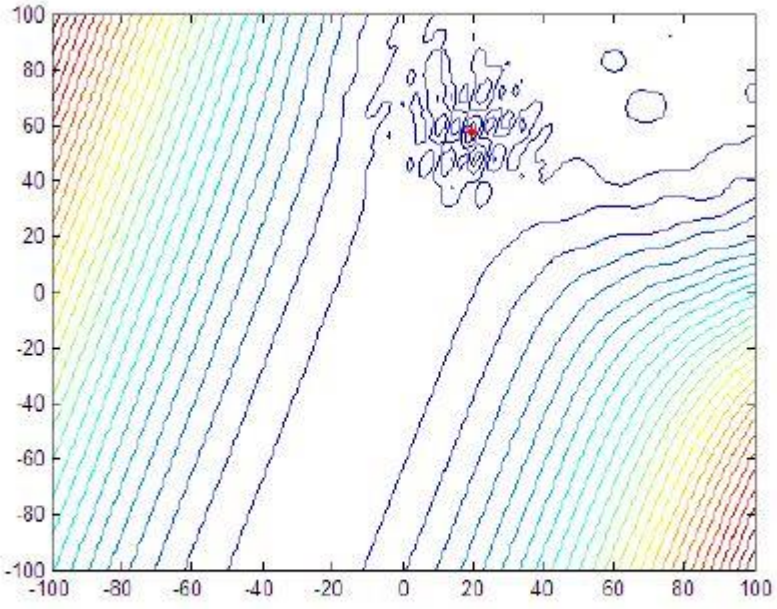

Figure 11(b).*Contour map for 2-D function*

Properties:

- Multi-modal
- Non-separable
- Asymmetrical
- Different properties around different local optima

15) Composition Function 3 (N=5)

D = 10, D = 30, F_{15}* = **1500**

N = 5
$\boldsymbol{\sigma}$ = [10, 10, 10, 20, 20]
$\boldsymbol{\lambda}$ = [10, 10, 2.5, 25, le-6]
bias = [0, 100, 200, 300, 400]
g_1: Rotated HGBat Function f_7

g_2: Rotated Rastring's Function f_{12}
g_3: Rotated Schwefel's Function f_4
g_4: Rotated Weierstrass Function f_3
g_5: Rotated High Conditoined Elliptic Function f_{13}

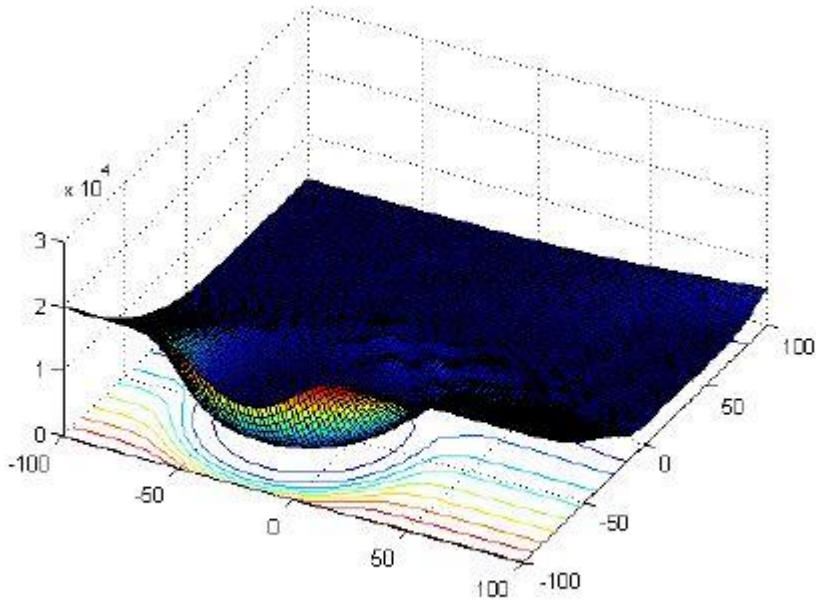

Figure 12(a). *3-D map for 2-D function*

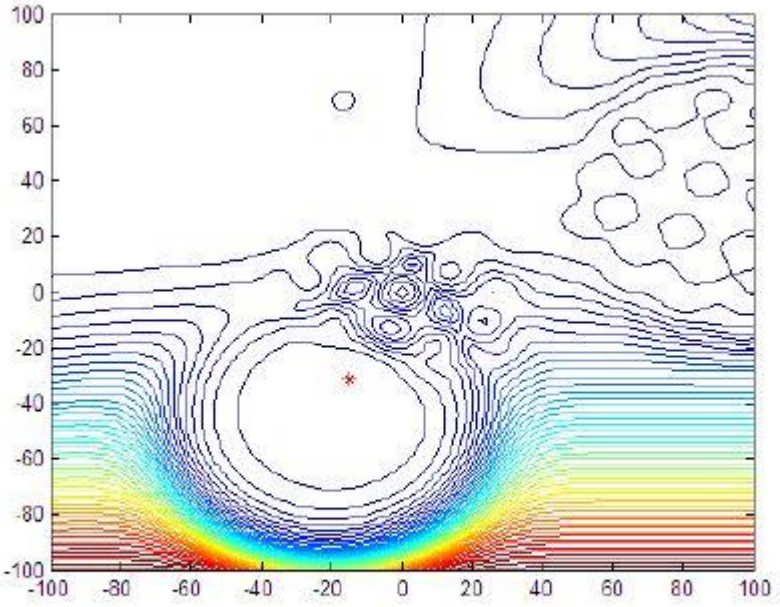

Figure 12(b). *Contour map for 2-D function*

Properties:

- Multi-modal
- Non-separable
- Asymmetrical
- Different properties around different local optima