

**YAŞAR UNIVERSITY  
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES**

**MASTER THESIS**

**AN ENSEMBLE OF DIFFERENTIAL EVOLUTION  
ALGORITHMS FOR REAL-PARAMETER  
OPTIMIZATION AND ITS APPLICATION TO  
MULTIDIMENSIONAL KNAPSACK PROBLEM**

**Mert PALDRAK**

**Thesis Advisor: Prof. Dr. M. Fatih TAŞGETİREN**

**Department of Industrial Engineering**

**Presentation Date: 20.01.2016**

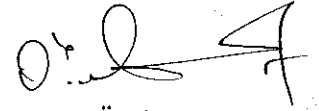
**Bornova-İZMİR  
2016**

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of master of science.



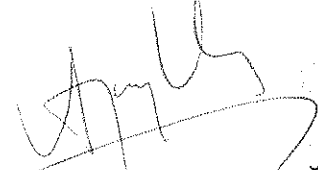
Prof. Dr. M. Fatih TAŞGETİREN (Supervisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of master of science.



Yard. Doç. Dr. Önder BULUT

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of master of science.



Prof. Dr. Adil BAYKASOĞLU



Prof. Dr. Cüneyt GÜZELİŞ  
Director of the Graduate School

*To my mother, Nalan*

## **ABSTRACT**

### **AN ENSEMBLE OF DIFFERENTIAL EVOLUTION ALGORITHM FOR REAL-PARAMETER OPTIMIZATION AND ITS APPLICATION TO MULTIDIMENSIONAL KNAPSACK PROBLEM**

PALDRAK, Mert

M.Sc in Industrial Engineering

Supervisor: Prof. Dr. M. Fatih TAŞGETİREN

January 2016, 86 pages

This thesis examines the recent real-parameter optimization methods through constrained single objective test functions. Inspired from this experience, it also presents the applicability of such methods to Multidimensional Knapsack Problem known as one of the most difficult discrete problems.

In the first part of this study, benchmark functions presented in CEC 2006 have been taken into consideration to solve. These benchmark problems are multi-dimensional and constrained real-parameter optimization problems with non-linear objective functions. Hence, it is quite difficult to solve them without using heuristic and metaheuristic approaches. In order to obtain optimal solutions, proposed algorithm (EDE-VNS) has been applied to these test functions and competitive results have been collected to compare with the best performing algorithms from the literature. The performance of DE algorithm depends on the mutation strategies, crossover operators and control parameters selected. As a result, an EDE-VNS algorithm that is possible to employ multiple mutation operators and control parameters in its VNS loops is proposed so as to be able further enhance the quality of the solution. By means of ensemble of variable mutation strategies in VNS loops, the performance of DE algorithm is affected so positively that most of benchmark functions could be optimally solved with zero standard deviations. In order to show the power of ensemble of mutation strategies, these test functions have also been solved by using all mutation strategies alone. It has been concluded that when using individual mutation strategies one by one, all of them fail to find the optimal solutions for test functions whereas when applying ensemble of these mutation strategies, algorithm could find optimal solutions easily by means of different properties of mutation strategies. Moreover, this algorithm was run for both 240,000

and 500,000 function evolutions. It is overly clear that EDE-VNS algorithm requires more function evolutions to find more optimal solutions with zero standard deviations. In addition, a diversification procedure which is based on the inversion of the target individual and the injection of some good dimensional values from promising areas in the population is also applied by using tournament selection with size 2. In order to take advantage of infeasible solutions in the evolved population, some constraint handling methods are also utilized to further improve the solution. The computational results show that the simple EDE-VNS algorithm was very competitive to the some of the best performing algorithms from the literature.

In the second part of this thesis, the 0-1 multidimensional knapsack problem which has a great range of applications in real-life problems is considered to be solved by proposed EDE-VNS algorithm. In the literature, most of the heuristic methods applied to multidimensional knapsack problem use and check and repair operator to improve solutions. Unlike the studies appearing in literature, some sophisticated constraint handling methods in order to enrich the population diversity are used. Differential evolution algorithm with variable neighbourhood search employing ensemble of mutation strategies to generate the trial population is proposed. Since the proposed DE-VNS algorithm in fact works on a continuous domain, the real-values of the chromosomes are converted to 0-1 binary values by using S-shaped and V-shaped transfer functions. The effects of these transfer functions are tested by using them one by one in each mutation strategies of ensemble. So as to qualify the solutions, a binary swap local search algorithm is combined with proposed EDE-VNS algorithm and the proposed algorithm is tested on a benchmark instances from the OR-library.

This thesis consists of 6 chapters which include all of these subjects

**Keywords:** Differential Evolution, Real Parameter Optimization, Variable Neighbourhood Search, Constraint Handling, Multidimensional Knapsack Problem,

## ÖZET

# GERÇEK PARAMETRE OPTİMİZASYONU İÇİN TOPLU DİFERANSİYEL EVRİM ALGORİTMASI VE ÇOK BUYUTLU SIRT ÇANTASI PROBLEMİNE UYGULANMASI

Mert PALDRAK

Yüksek Lisans, Endüstri Mühendisliği Bölümü

Tez Danışmanı: Prof. Dr. M. Fatih TAŞGETİREN

Ocak 2016, 86 sayfa

Bu tez, kısıtlanmış tek amaçlı test fonksiyonları aracılığı ile son dönemlerdeki gerçek parametre optimizasyon metotlarını incelenmiştir. Bu deneyimden esinlenerek, bu tür yöntemlerin aynı zamanda en zor ayrık problemlerden birisi olarak bilinen çok boyutlu sırt çantası problemine uygulanabilirliğini de ortaya koymuştur.

Bu çalışmanın ilk bölümünde, CEC 2006'da ortaya konulan kıyaslama problemleri çözülmek üzere ele alınmıştır. Bu kıyaslama problemleri doğrusal olmayan amaç fonksiyonlarına sahip, çok boyutlu ve kısıtlanmış gerçek parametrelili optimizasyon problemleridir. Bundan dolayı, sezgisel ve meta sezgisel yaklaşımları kullanmadan bu problemleri çözmek oldukça zordur. En iyi çözümler elde etmek için, önerilen algoritma (EDE-VNS) bu test fonksiyonlarına uygulanmıştır ve literatürdeki en iyi performansı gösteren algoritmalar ile karşılaştırılmış, rekabetçi sonuçlar elde edilmiştir. DE algoritmasının performansı çoğunlukla mutasyon stratejilerine, çaprazlama operatörlerine ve seçilmiş kontrol parametrelerine bağlıdır. Sonuç olarak, birden fazla mutasyon operatörleri ve kontrol parametrelerini kendi VNS döngüleri içerisinde bulundurabilen bir EDE-VNS algoritması çözümün kalitesini arttırabilmek amacıyla geliştirilmiştir. VNS döngüleri içindeki değişken mutasyon stratejilerinin toplu halde çalışmaları sayesinde, DE algoritmasının performansı o kadar olumlu etkilenmiştir ki çoğu kıyaslama problemleri sıfır standart sapma ile optimal olarak çözülmüştür. Mutasyon stratejilerinin toplu halde çalışmalarını etkisi göstermek için, bu test fonksiyonları bütün mutasyon stratejileri teker teker kullanılarak da çözülmüştür. Bireysel mutasyon stratejileri teker teker kullanıldığında, hepsi test fonksiyonlarında optimum çözümler bulma konusunda başarısız olduğu, oysaki bu mutasyon stratejileri toplu halde uygulandığında algoritma mutasyon stratejilerinin farklı özellikleri sayesinde optimal sonuçları kolaylıkla bulabildiği sonucuna

varılmıştır. Bunun üzerine, bu algoritma aynı zamanda 240,000 ve 500,000 fonksiyon değerlendirilmesi ile çalıştırılmıştır. . Bu apaçık ortadadır ki, EDE-VNS algoritması ile daha çok optimal çözümler bulmak, daha fazla fonksiyon değerlendirilmesine ihtiyaç duyulmaktadır. Buna ek olarak, hedef bireylerin evrimini ve popülasyon içinde umut vadeden alanlardan alınan bazı iyi boyutlu değerlerin enjeksiyonunu temel alan çeşitlendirme yöntemi de, iki boyutlu turnuva seçim yöntemi kullanılarak uygulanmıştır. Gelişmiş popülasyon içerisindeki uygun olmayan çözümlerden faydalanabilmek için, çözümü daha da geliştirmek amacıyla bazı kısıtlama işleme kuralları kullanılmıştır. Hesaplanan sonuçlar göstermektedir ki basit bir EDE-VNS algoritması literatürdeki bazı en iyi performansı gösteren algoritmalarla oldukça rekabetçidir.

Bu tezin ikinci bölümünde, gerçek hayat problemlerinde geniş ölçüde uygulamaları olan 0-1 çok boyutlu sırt çantası probleminin, önerilen EDE-VNS algoritması ile çözülebileceği öngörülmüştür. Literatürde, çok boyutlu sırt çantası probleminde uygulanan sezgisel yöntemlerin birçoğu, çözümleri geliştirmek için kontrol ve onarım operatörlerini kullanmıştır. Literatürde ortaya çıkan çalışmaların aksine, popülasyon çeşitliliğini zenginleştirmek için bazı gelişmiş kısıtlama işleme yöntemleri kullanılmıştır. Çeşitli toplu mutasyon stratejilerini kullanan değişken komşu aramalı diferansiyel evrim algoritması, deneme popülasyonunu oluşturmak için ortaya atılmıştır. Aslında önerilen bu EDE-VNS algoritması sürekli alanda çalıştığı için, gerçek değer kromozomları S-şeklindeki ve V-şeklindeki transfer fonksiyonlar kullanılarak 0-1 ikili değerlerine dönüştürülmüştür. Çözümleri geliştirmek için, EDE-VNS algoritmasıyla ikili takas yerel arama algoritması birleştirilmiş, önerilen algoritma OR-kütüphanesinden alınan karşılaştırma örnekleri üzerinde test edilmiştir.

Bu tez, yukarıda bahsedilen konuları içeren 5 üniteden oluşmaktadır.

**Keywords:** Diferansiyel Evrim Algoritması, Gerçek Parametre Optimizasyonu, Değişken Komşu Arama, Kısıtlama İşleme, Çok Boyutlu Sırt Çantası Problemi

## ACKNOWLEDGEMENTS

Immeasurable appreciation and deepest gratitude for the help and support are extended to the following persons who are in one way or another have contributed in making this thesis possible to complete.

First of all, I am thankful to my supervisor, **Prof. Dr. M. Fatih Taşgetiren**, for his support, advices, guidance, valuable and important comments, suggestions and provisions that helped me to complete this thesis successfully. He was more than a supervisor and adviser to me. He did his best to support and lead me at each step of this study in spite of all difficulties and challenges he faced during this thesis. Without his guidance and persistent help, it would have been impossible for me to complete this thesis.

I would like to sincerely thank my teacher, **Dr. Efthimia Staiou**, for her time and effort in checking this manuscript, and for giving enough time to correct this manuscript and help me complete this thesis.

I gratefully acknowledge the contributions of my colleagues, **Bahar Taşar**, **Cemre Çubukçuoğlu** and **Sinem Özkan**. I would like to thank them for their sincere collaboration, encouraging and supporting me as I am conducting this research.

I want to express my gratitude and sincere thanks to my close friends, **Fulya Çabuk**, **Kibare Atik**, **Ozan Karakaya**, **Muhammet Değermenci**, **Ali Rıza Evren** and **Zeynep Bilgiç** for their being so continuously supportive.

I specially thank my friends, **Özkan İşleyen**, **Sinan Bilir** and **Ufuk Delihasan** in engineering department for encouraging me in carrying out this thesis and taking their times to help me complete this manuscript.

Lastly, I am sincerely thankful to my parents. I would like to thank to my mother, **Nalan Paldrak**, my father, **Mehmet Paldrak**, my brother, **Berk Paldrak** for their persistent supports, encouragements and helps.

Mert PALDRAK  
İzmir, 2015



## **TEXT OF OATH**

I declare and honestly confirm that my study, titled “An Ensemble of Differential Evolution Algorithm for Real-Parameter Optimization and Its Application to Multidimensional Knapsack Problem” and presented as a Master’s Thesis, has been written without applying to any assistance inconsistent with scientific ethics and traditions, that all sources from which I have benefited are listed in the bibliography, and that I have benefited from these sources by means of making references.

## TABLE OF CONTENTS

	<b>Page</b>
ABSTRACT	iv
ÖZET	vi
ACKNOWLEDGEMENTS	viii
TEXT OF OATH	ix
TABLE OF CONTENTS	x
INDEX OF FIGURES	xiii
INDEX OF TABLES	xiv
INDEX OF SYMBOLS AND ABBREVIATIONS	xv
1 INTRODUCTION	1
1.1 Subject of the Thesis	1
1.2 Aims of the Research	2
1.3 Context of the Thesis	3
1.4 Methodology	3
2 DIFFERENTIAL EVOLUTION ALGORITHM	5
2.1 Introduction to Differential Evolution Algorithm	5
2.2 Steps of DE Algorithm	8

2.2.1	Initialization of Target Population	9
2.2.2	Mutation with Difference Vector	10
2.2.3	Crossover	10
2.2.4	Selection	12
2.3	Self-Adaptive Differential Evolution	12
2.4	JADE	14
2.4.1	DE/current-to-pbest	15
2.4.2	Self-Adaptation of Parameters	15
2.4.3	Explaining JADE Algorithm Setting	17
2.5	Ensemble Differential Evolution	18
2.6	Opposition-Based Differential Evolution	20
2.7	Ensemble DE with VNS	22
2.7.1	Generating Initial Population	26
2.7.2	Generation of Trial Population	27
2.7.3	Selection	28
3	CONSTRAINED REAL PARAMETER OPTIMIZATION	30
3.1	Benchmark Problems	32
3.2	Constraint Handling Methods	33

3.2.1	Superiority of Feasible Solution	33
3.2.2	The Adaptive Penalty Function	34
3.2.3	$\epsilon$ – Constraint (EC)	35
3.2.4	Stochastic Ranking	36
3.3	Ensemble of Constraint Handling	36
3.4	Computational Results of Constrained RPO	38
4	APPLICATIONS ON MULTIDIMENSIONAL KNAPSACK PROBLEM	44
4.1	Solution Methodology for MKP	45
4.2	Solution Representation	46
4.3	Families of Transfer Functions	48
4.4	Binary Swap Local Search	50
4.5	Computational Results	52
5	CONCLUSIONS & FUTURE WORK	59
	REFERENCES	61
	CURRICULUM VITEA	71
	APPENDIX 1 CEC 2006 BENCHMARKS	72

## INDEX OF FIGURES

Figure 2.1 Steps of DE Algorithm	9
Figure 2.2 Self-Adapting: Encoding Aspect	14
Figure 2.3 Outline of EDE Algorithm	19
Figure 2.4 $VNS_1$ Algorithm	23
Figure 2.5 $VNS_2$ Algorithm	25
Figure 2.6 $VNS_3$ Algorithm	25
Figure 2.7 $VNS_4$ Algorithm	26
Figure 2.8 Injection Procedure	28
Figure 2.9 Outline of $EDE - VNS$ Algorithm	28
Figure 3.10 Stochastic Ranking	36
Figure 4.11 Solution Representation	47
Figure 4.12 Outline of BSWAP Local Search	50
Figure 4.13 Outline of $EDE - VNS$ Algorithm	51

## INDEX OF TABLES

Table 1 Details of CEC 2006 Benchmark Problems.....	32
Table 2 Computational Result of EDE-VNS, GA-MPC, APF-GA, MDE, ECHT-EP2 for CEC 2006 Test Problems.....	40
Table 3 Feasibility Rates of Benchmark Problems for EDE and Each Mutation Strategy.....	43
Table 4 Transfer Functions.....	49
Table 5 Details of Sento and Weing Instances.....	53
Table 6 Computational Results of Sento and Weing Instances.....	53
Table 7 Details of Weish Instances.....	54
Table 8 Computational Results of Weish.....	55
Table 9 Details of HP and PB Instances.....	57
Table 10 Computational Results of HP and PB Instances .....	58

## INDEX OF SYMBOLS AND ABBREVIATIONS

<u>Symbols</u>	<u>Explanations</u>
$\nu(\bar{x})$	Average violation of number of constraints
$\delta$	Tolerance value for equality constraints
$t_c$	Control generation
$x_{ij}^g$	$i^{th}$ target individual at generation $g$
$v_{ij}^g$	$i^{th}$ mutant individual at generation $g$
$u_{ij}^g$	$i^{th}$ trial individual at generation $g$
$f(x)$	Fitness value of solution (x)
$F_i$	Mutation scale factor
NP	Number of Population
$CR_i$	Crossover probability
$KF$	Value randomly chosen within the range [0,1]
$\mu_{CR}$	Mean Crossover
$\mu_F$	Mean Mutation Scale Factor
$S_{CR}$	Successful Crossover
$L(.)$	Lehmer Mean

$T(x)$	Transfer Function
$n$	Number of Items
$m$	Number of Constraints
$c_{ij}$	Units of Profit
$a_{ij}$	Units of Resource
$b_i$	Capacity

<u>Abbreviations</u>	<u>Explanations</u>
----------------------	---------------------

GA	Genetic Algorithm
ES	Evolution Strategy
DE	Differential Evolution
PSO	Particle Swarm Optimization
EP	Evolutionary Programming
QN	Quasi Newton
SA	Simulated Annealing
TS	Tabu Search
GP	Genetic Programming
CEC	Congress on Evolutionary Computation
10-D	10 Dimensional



30-D	30 Dimensional
jDE	Self-Adaptive Differential Evolution
JADE	Self-Adaptive Differential Evolution
SADE	Self-Adaptive Differential Evolution
FADE	Fuzzy Adaptive Differential Evolution
EDE	Ensemble Differential Evolution
OBL	Opposition Based Learning
OP	Opposite Population
GOBL	Global Opposition Based Learning
VNS	Variable Neighbourhood Search
CRS	Controlled Random Search
EDE-VNS	Ensemble Differential Evolution with Variable Neighbourhood Search
NP	Non Polynomial
CMA-ES	Covariance Matrix Adaptation Evolution Strategy
APF-GA	Adaptive Penalty Formulation with Genetic Algorithm
SF	Superiority of Feasible Solutions
LI	Linear Inequity
NI	Nonlinear Inequity

LE	Linear Equity
NE	Nonlinear Equity
NFT	Adaptive Penalty Function
SR	Stochastic Ranking
CH	Constraint Handling
ECHT	Ensemble of Constraint Handling
ISR	Improved Stochastic Ranking
RPO	Real Parameter Optimization
MDE	Modified Differential Evolution
ECHT-EP2	Ensemble of Constraint Handling Techniques
NFL	No Free Lunch
ICEO	International Contest on Evolutionary Optimization
MKP	Multidimensional Knapsack Problem
SACRO	Self-Adaptive Check and Repair
bFOA2	Binary Fruit Fly Algorithm
HEDA2	Hybrid EDA-based Algorithm
BPSO	Binary Particle Swarm Optimization
BSWAP	Operator-Based Swap Local Search

# 1 INTRODUCTION

## 1.1 Subject of the Thesis

The subject of thesis consists of the solution algorithms for optimization problems used for finding the best solution from all possible feasible solutions. An optimization problem is either minimizes or maximizes a function of decision variables under the hard and soft constraints. Optimization problems are divided into two categories according to types of their decision variables. An optimization problem with discrete variables is called combinatorial optimization problem, whereas an optimization problem with continuous variables is called continuous optimization problems. If these so-called decision variables contain real parameters, the problem is called real parameter optimization. One of the earliest application areas of evolutionary algorithms is real parameter optimization. The evolutionary algorithms applied so far to solve real parameter problems can be summarized as: real-parameter GAs, evolution strategies (ES), differential evolution (DE), particle swarm optimization (PSO), evolutionary programming (EP), classical methods such as quasi-Newton method (QN), hybrid evolutionary-classical methods, other non-evolutionary methods such as simulated annealing (SA), tabu search (TS) are some of most the well-known algorithms applied to solve real parameter problems. It is possible to find out the further improved versions of these algorithms mentioned above. Lately, different types of optimization problems used for solving real-parameter optimization problem have arisen among the evolutionary computation committees or conferences as well as journals. Many different algorithms, developed for solving problems in CEC 2006 Special Session on Constrained Real-Parameter Optimization yielded favourable solutions.

The biggest reason of developing heuristic and metaheuristic algorithms to solve these problems is that they are too hard and complicated problems for solving by using exact techniques. These algorithms also involve some important ones such as  $\epsilon$ -Constrained Differential Evolution with Gradient-Based Mutation and Feasible Elites (Takamaha T. & Sakai S., 2006), Dynamic Multi-Swarm Particle Swarm Optimizer with a Novel Constraint-Handling Mechanism (Liang J.J. & Suganthan P. N., 2006), Self-adaptive Differential Evolution Algorithm (Huang V. L., 2006), A Multi-Populated Differential Evolution Algorithm (Tasgetiren M. F. & Suganthan P. N., 2006). In those studies, optimization techniques are applied to the problems of

many field of science including standard test problems (Sphere function, Rosenbrock function or Schwefel function) and various engineering problems.

Moreover, the main focus of the study revolves around the application of an evolutionary algorithm to one of the most popular optimization problem called multidimensional knapsack problem with very important applications in financial and industrial areas such as investment decision, budget control, project choice, resources assignments and goods loading. This problem is a generalization of the standardize 0-1 knapsack problem and known as one of the most difficult discrete optimization problems in the literature. It is probable to find the family of the knapsack problem in different areas of study. Members of the knapsack problem family are 0-1 knapsack problem, bounded knapsack problem, multiple choice knapsack problem and multiple or multidimensional knapsack problem. These problems require a subset of some given items to be chosen such that corresponding profit sum is maximized without exceeding the capacity of the knapsack or knapsacks. All of these members belong to the family of NP-hard problems. In spite of being NP-hard problems, many large instances of knapsack problems can be solved in seconds. This is because of several years of research having proposed many solution methodologies including exact as well as heuristic and metaheuristic algorithms. Heuristic algorithms involve simulated annealing ( Liu et al, 2006), genetic algorithm ( Thiel & Voss, 1994), ant colony optimization ( Zhao & Zhang, 2006), differential evolution ( Peng et al., 2008), immune algorithm (Lei et al., 2000) and particle swarm optimization (Ye et al., 2006). In recent, some of powerful heuristic algorithms such as fruit fly optimization (Wang et al., 2013) and differential evolution with variable neighbourhood search (Taşgetiren et al., 20015) could further improve the solutions for knapsack problem in literature. The proposed heuristic algorithm (EDE-VNS) employing different mutation strategies in their VNS loops is tested on benchmark instances from the OR-library and results are compared with other heuristic algorithms in the literature.

## **1.2 Aims of the Research**

Based on the previously given statements, the main goal of this thesis is to make use of the power of real-parameter optimization methods in constrained single objective test functions taken from literature. Based on this experience, the further goal of the thesis is to introduce its application to multidimensional knapsack problem by using differential evolution algorithm.

### **1.3 Context of the Thesis**

This thesis content is based on real-parameter optimization techniques and its applicability to the various real life problems. Lately, since most of the optimization problems are too complicated to solve by means of traditional methods, different evolutionary algorithms have been being developed. In addition to this, differential evolution algorithm has become one of the most powerful evolutionary algorithms to solve optimization problems. Therefore, in this thesis, a variant of differential evaluation algorithm named EDE-VNS is considered to solve constrained-real parameter optimization problems. In order to test how much well the proposed algorithm is working, CEC 2006 benchmark instances are taken into consideration. Since these benchmark instances include various real-life problems in such disciplines as engineering, logistics, energy systems, scheduling, finance and so on, dealing with these problems by proposed algorithm also shows how suitable to apply EDE-VNS algorithm to real life-problems. The optimal solutions of the benchmark instances obtained by using proposed algorithm and comparisons with other best performing algorithms in the literature are provided in tables.

To extend the application of the proposed algorithm to real-life problems, this thesis also covers multidimensional knapsack problem in detail. Instead of traditional methods in the literature, it is aimed to solve this NP-hard problem by using proposed EDE-VNS algorithm with the help of constraint handling techniques. As the real-parameter optimization problems run the algorithms in a continuous domain, some types of transfer functions are utilized to convert real values to binary 0-1 variables. In order to demonstrate the effect of algorithm on knapsack problem, benchmark instances varying in size on OR-library are used. The optimal solutions obtained and necessary comparisons with previously developed algorithms are provided in tables. According to these comparisons, it is concluded that this proposed algorithm is competitive to best performing algorithms from literature to solve a real-life problem.

### **1.4 Methodology**

The method of this thesis is based firstly on the problem definitions of CEC'2006 competition for constrained single objective real-parameter numerical optimization. In "CEC' 2006 Constrained Single Objective Real-Parameter Optimization Special Session" some heuristic algorithms are competing through the

test functions including engineering problems presented in advance. In literature, it is possible to find the articles of researchers who have been trying to find the best solutions with zero standard deviations. So as to achieve the optimal solutions, the results are obtained from Ensemble Differential Evolution Algorithm with Variable Neighborhood Search (EDE-VNS) was compared with best performing algorithms in the literature. With the proposed EDE-VNS algorithm, such constraint handling techniques as  $\epsilon$ -constraints, self-adaptive penalty function and superiority of feasible solution are used in order that possible important information carried by infeasible solutions can be used. Moreover, to demonstrate the effect of number of function evaluations on ensemble mutation strategies in differential evolution algorithm, the proposed algorithm was run for both 240,000 and 500,000 function evaluations. In order to further improve the solutions, Opposition Based Learning (OBL) and diversification methods are utilized for diversifying the initial and target populations respectively. After algorithm comparison for all benchmarks, this proposed algorithm is applied to multidimensional knapsack problem. Since it is a NP-hard problem aiming to maximize the profit under certain capacity constraints with 0-1 binary values, the dimensions of each chromosome of proposed algorithm is changed to binary numbers by using transfer functions. For better qualification of solutions, binary swap local search is applied to the best solution at each generation. Instead of using check and repair operators in literature, constraint handling methods are involved in the problem within a predetermined threshold. Finally, the computational results of the instances from OR-library demonstrate the efficiency of the algorithm in solving benchmark instances and its superiority to the best performing algorithm from the literature.

All in all, constrained Real-Parameter Optimization part of this study aims to test the performance of EDE-VNS Algorithm through CEC 2006 benchmarks by comparing to the best performing algorithms from the literature. According to results obtained, proposed EDE-VNS algorithm is competitive with the best performing algorithms. As an application of proposed algorithm, one of the most important real life problems, multidimensional knapsack problem is dealt with. This problem is handled in order to show how applicable and usable the evolutionary algorithms are to solve real life problems.

## **2 DIFFERENTIAL EVOLUTION ALGORITHM**

### **2.1 Introduction to Differential Evolution Algorithm**

Differential Evolution (DE) Algorithm was firstly introduced by Storn & Price (1995) as an efficient and powerful population-based heuristic search technique in order to minimize the objective function of optimization problems over nonlinear and non-differentiable continuous space. The optimization problems solved by differential evolution algorithm arise from many scientific and engineering fields. Moreover, DE has been successfully applied to optimization problems in such fields as mechanical engineering, communication and pattern recognition.

One of the recent studies conducted by Das& Suganthan(2011) has clearly explained the history of DE and its success in details. It is known that DE is one of the most powerful stochastic real-parameter optimization algorithms used currently. DE operates through similar computational steps like a standard evolutionary algorithm (EA). Nonetheless, unlike traditional evolutionary algorithms, DE variants perturb the current population members with the differences of randomly selected and distinct population members ( Das&Suganthan 2011).

The DE algorithm (Price & Storn, 1995-1996-1997) has become an important and competitive algorithm to evolutionary algorithms more than a decade ago. After the publication of first article written on DE by R. Storn and K. Price, DE algorithm was demonstrated as the best evolutionary algorithm in order to solve real-valued test function in the 1<sup>st</sup> ICEO (International Contest on Evolutionary Optimization) and then turned to be the one of the best among the competing algorithms at 2<sup>nd</sup> ICEO in 2007. In two different journal articles, Price (1997), Storn and Price (1997) have introduced the algorithm in details which is followed by immediately in quick succession. In 2005, CEC competition on real parameter optimization, on 10-D problems classical DE secured 2<sup>nd</sup> rank and a self-adaptive DE variant called SaDe (Quin, Suganthan, 2005) secured 3<sup>rd</sup> rank although they performed poorly over 30-D problems. Even though some variants of ES gave much better results than classical and self-adaptive DE, later on many improved such variants of DE as opposition-based DE (ODE) (Rahnamayan et al, 2008), DE with global and local neighbourhoods (Das et al., 2009) and (Zhang & Sanderson, 2009) were being proposed between the years 2006 and 2009. On the other hands, it became necessary

to determine how well these variants of DE can compete against the restart CMA-ES and other real parameter optimizers over the standard numerical benchmarks. It is also interesting to notify that the some variants of DE algorithm continued securing front ranks in the subsequent CEC competitions (Suganthan, 2012) like CEC 2006 competition on constrained real parameter optimization (first rank), CEC 2007 competition on multi objective optimization (second rank), CEC 2008 competition on large scale optimization (third rank). A very recent study conducted by Neri and Tirronen focuses on the variants of DE for single-objective optimization problems, as well as compared them on some set of benchmark problems. Based on the review studies, it is conducted that DE-variants are as effective as original DE to solve the complex optimization problems.

In DE community, the individual trial solutions are called *parameter vectors* or *genomes*. In DE, there exist many trial vector generation strategies in which some of them might be powerful and suitable to solve a particular problem. However, DE employs difference of parameter vectors to find the objective function landscape. Because of this, DE algorithm owes a lot to its ancestors namely- the Nelder-Mead algorithm (Nelder & Mead, 1965) and controlled random search (CRS) algorithm (Price, 1977) which is based on the difference vectors to perturb the current trial solutions.

Compared to the other evolutionary algorithms, DE algorithm has following advantages:

- DE algorithm has much more simple and straightforward code structure to implement. It enables users to practically solve optimization by means of its simple implementation. Main body of the algorithm takes from four to five lines to code in any programming language. Even though such algorithms as PSO is also quite easy to code, the performance of DE and its variants is largely better than the PSO variants over a wide range of optimization problems. (Das et al, 2009), (Rahnamayan et al, 2008), (Vesterstrom & Thomson, 2004)
- As indicated by recent studies conducted by (Das et al, 2009), (Rahnamayan et al, 2008) and (Zhang & Sanderson), DE demonstrates



much better performance in comparison with several others like G3 with PCX, MA-S2, ALEP, CPSO-H, and so on of current interest on a wide variety of problems that include unimodal, bimodal, separable, non-separable and so on. In spite of the fact that strong EAs like the restart CMA-ES was able to beat DE at CEC 2005 competition, on non-separable objective functions, the total performance of DE in terms of accuracy robustness and convergence speed makes DE more suitable to apply to various real-world optimization problems in which finding an appropriate solution takes too much time.

- Another advantage of DE algorithm is the fact that it requires few number of control parameters. Three crucial control parameters involved in DE, i.e., population size  $NP$ , scaling factor  $F$ , and crossover rate  $CR$ , may significantly affect the optimization performance of DE. Liu & Lampien (2002) has reported that the effectiveness, efficiency and robustness of the DE algorithm are very sensitive to the setting of its control parameters. The best settings for the control parameters depend on the function and requirements for consumption of time and accuracy. The effects of these parameters on the performance of algorithm have been being studied well. When only a simple rule of scaling factor  $F$  and cross over  $CR$  is altered, the performance and robustness of algorithm is significantly improved without imposing any important computational burden as presented in Brest et al. (2006), Qin et al. (2009), Zhang and Sanderson (2009).
- Furthermore, DE is better to handle the large scale and expensive optimization problems owing to its feature that the space complexity of DE is less than the other competitive real parameter optimizers as mentioned in the article of Hansen & Ostermeier (2001).

On the other hands, like all other metaheuristic methods, DE has got some drawbacks that should be considered for the discussion of future research directions with DE. Some of the current publications made by Rahnamayan et al. (2008)

indicates that DE faces up with some important difficulties in solving such functions that are not linearly separable and can be outperformed by CMA-ES. As mentioned by Sutton et al.(2007), on some functions, DE is more dependent on its differential mutation procedure, which, unlike its recombination strategy with ( $CR < 1$ ), is rationally invariant. Sutton et al. (2007) also surmises that this mutation strategy lacks enough selection pressure as appointing target and donor vectors to have satisfying power on non-separable functions. The authors of the article also present a rank-based parent selection scheme so as to impose bias on the selection step, in order that DE can learn distribution information from elite individuals selected from population and can thus sample the local topology of the fitness landscape better. Nonetheless, they concluded that much more research is necessary in this realm to deduce that DE is sufficiently robust against the strong interdependency of the search variables. In another article written by Langdon & Poli (2007) made an attempt in order to evolve certain fitness landscapes with GP to show the advantages and disadvantages of a few population-based metaheuristics like PSO, DE, and CMA-ES. Authors highlighted that some problem landscapes might deceive DE such that it will get stuck in local optima most of the time. The same authors also pointed out that DE may sometimes show limited ability to move its population large distances across the search space when a limited portion of the population is clustered together. The work done by Langdon & Poli (2007) indicated that some landscape in which DE is outperformed by CMA-ES and a non-random gradient search based on Newton-Raphson's method. The most effective DE-variants improved so far should be investigated with the problem evolution methodology of Langdon & Poli in order to identify some specific weak points over different function surfaces.

## **2.2 Steps of DE Algorithm**

DE is a simple real parameter optimization algorithm. In a real parameter optimization problem, all of decision numbers must be real numbers. To describe it well, the DE/rand/1/bin scheme of Storn and Price (1995) was used. This scheme has been applied to a variety of problems that can be found in Corne et al. (1999), Lampinen (2011), Babu and Onwubolu (2004), Price et al. (2005) and Chakraborty (2008) and Das and Suganthan (2011). A simple cycle of stages worked through by DE is presented in Figure 2.1.

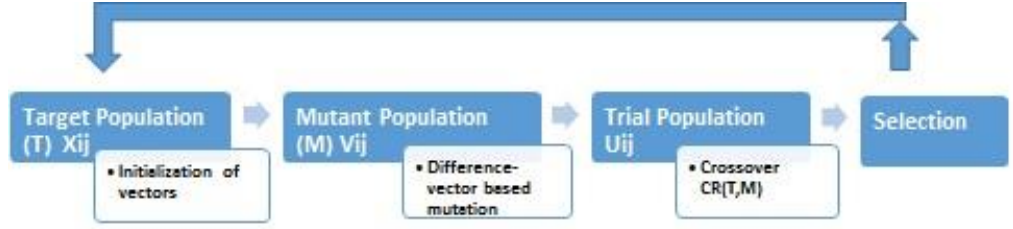


Figure 2.1. Steps of DE algorithm

### 2.2.1 Initialization of Target Population

In DE algorithm, global optimal point is searched over a  $D$ -dimensional real parameter space  $\mathfrak{R}^D$ . It starts with a randomly initiated population with size  $NP$  (number of parents) having a  $D$ -dimensional real-values parameter vectors. Each vector, known as *genome/chromosome* carries an alternative solution to the multidimensional optimization problem. Subsequent generations in DE are denoted by  $G = 0, 1, \dots, G_{max}$ . As the parameter vectors are likely to alter over different generations, it may be adapted to following notations in order to represent  $i$ th vector of the population at any generation:

$$\overline{X}_i^G = [x_{i1}^G, x_{i2}^G, x_{i3}^G, \dots, \dots, x_{iD}^G] \quad (1)$$

For each parameter of the problem, there might be a certain range within which the value of the parameter should be restricted, because parameters are related to physical components or measurements that own natural bounds. Each vector is obtained randomly and uniformly within the search space constrained by the predefined minimum and maximum bounds:  $[x_{ij}^{min}, x_{ij}^{max}]$ . Therefore, the initialization of  $j^{th}$  component of  $i^{th}$  vector can be defined as:

$$x_{ij}^0 = x_{ij}^{min} + r \times (x_{ij}^{max} - x_{ij}^{min}) \quad (2)$$

where  $x_{ij}^0$  is the  $i^{th}$  target individual at generation  $g = 0$ ; and  $r$  is a uniform random number in the range  $[0, 1]$ .

### 2.2.2 Mutation with Difference Vector

As a biological term, mutation is defined as a sudden change in the gene characteristics of a chromosome. In the context of the evolutionary paradigm, mutation is seen as a change or perturbation with a random element selected. Mutation is a way to create new solutions. However, it consists in random changing value of parameters in the context of GAs and EAs. In DE literature, base vectors are mutated with scaled population-derived difference vectors and this method is believed to be one of the main strength of DE (Storn & Price, 1997). These differences tend to adapt to the natural scaling of the problem as generations continue passing. Therefore, DE differs from the other evolutionary based algorithms because it requires only the specification of a single relative scale factor  $F$  for all variables.

By the definition of Das & Sughantan (2011), a parent vector from the current generation is called target vector, a mutant vector gained through the differential mutation operation is known as donor vector and finally an offspring generated by recombination of the donor with the target vector is called trial vector.

So as to obtain mutant individuals, the weighted difference of two individuals from target population is added to a third individual randomly chosen from population.

$$v_{ij}^g = x_{aj}^{g-1} + F \times (x_{bj}^{g-1} - x_{cj}^{g-1}) \quad (3)$$

where  $a, b, c$  are three randomly chosen individuals from the target population such that  $(a \neq b \neq c \neq i \in (1, \dots, NP))$  and  $j = 1, \dots, D$ .  $F > 0$  is a mutation scale factor influencing the differential variation between two individuals.

### 2.2.3 Crossover

Whereas Genetic Algorithm always recombine two vectors to generate two separate trial vectors with one-point crossover, DE algorithm is managed to crossover to produce one single trial vector. Crossover is used to enhance the potential variety of the population, and comes into play after creating the donor vector through vector mutation. The donor vector exchanges its components with the target vector  $\vec{X}_i^G$  under crossover operation in order to generate  $\vec{U}_i^G = [u_{i1}^G, u_{i2}^G, u_{i3}^G, \dots, u_{iD}^G]$ .  $N$ -point

crossover is one of the most well-known crossover techniques for real coded GAs. In this technique, the offspring vector is divided into  $(n + 1)$  parts such that parameters in contiguous parts are obtained by different parent vectors.

Price et al. (2005) notifies that the DE family of algorithms may use two different crossover methods—*exponential* (or two-point modulo) and *binomial* (or uniform). By the definition of Das & Sughantan (2011), in exponential crossover, an integer number  $n$  is selected randomly among the numbers  $[1, D]$ . This integer number is considered as a starting point in target vector, from where the crossover of components with the donor vector begins. Another integer  $L$  denoting the number of components the donor vector which contributes to the target vector is selected from the interval  $[1, D]$ . After choosing  $n$  and  $L$ , the trial vector is acquired as:

$$\begin{aligned} u_{ij}^g &= v_{ij}^g & \text{for } j = \langle n \rangle_D \quad \langle n + 1 \rangle_D, \dots, \dots, \langle n + L - 1 \rangle_D \\ x_{ij}^g & \text{for all other } j \in [1, D] \end{aligned} \quad (4)$$

where the angular brackets  $\langle . \rangle_D$  denote the modulo function with modulus  $D$ . The integer  $L$  is taken from  $[1, D]$  according to following pseudo-code:

```

L = 0; DO
{
L = L + 1
} WHILE ((rand(0,1) ≤ CR )AND (L ≤ D))

```

where “ $CR$ ” is known as crossover rate and appears as a control parameter of DE just like  $F$ .  $CR$  is defined by users in the range  $[0,1]$ , and  $r_{ij}^g$  is a uniform random number in the range  $[0,1]$ .

On the other hand, in this thesis, binomial crossover is applied to each variable when a randomly generated number between 0 and 1 is less than or equal to  $CR$  value. In this case, the number of parameters obtained from the donor has approximately binomial distribution. Trial individuals are gained by recombination of mutant individuals with its corresponding target individuals. The scheme can be outlined as:

$$u_{ij}^g = \begin{cases} v_{ij}^g & \text{if } r_{ij}^g \leq CR \text{ or } j = D_j \\ x_{ij}^{g-1} & \text{otherwise} \end{cases} \quad (5)$$

where the index  $D_j$  is a randomly chosen dimension ( $j = 1, \dots, D$ ). It assures that at least one parameter of the trial individual  $u_{ij}^g$  will be different from the target individual  $x_{ij}^{g-1}$ .

As generating trial individuals, parameter values might violate search ranger. In order to avoid this, parameter values that violate the search range are randomly and uniformly re-generated by using following formula:

$$x_{ij}^g = x_{ij}^{min} + r \times (x_{ij}^{max} - x_{ij}^{min}) \quad (6)$$

#### 2.2.4 Selection

In order to have constant number of population size as generations pass, the next step of the algorithm is selection. *Selection* is applied to determine whether the target of the trial vector survives to the next generation, i.e., at  $g = g + 1$ . For the next generation, selection is based on the survival of the fittest among trial and target individuals such that:

$$x_i^g = \begin{cases} u_i^g & \text{if } f(u_i^g) \leq f(x_i^{g-1}) \\ x_i^{g-1} & \text{otherwise} \end{cases} \quad (7)$$

The objective function is supposed to be minimized. Based on the equation above, if the fitness value of new trial vector yields an equal or lower value of the function, it replaces the corresponding target individual in the next generation, other else the target is held in the population. As a conclusion, the population either gets better or remains the same in fitness status, but never gets worse.

### 2.3 Self-Adaptive Differential Evolution

Selection of the control parameters of DE is an important issue because it is quite possible to reach at different conclusions even if only one of them is changed.

The existence of different variants of DE has already been mentioned in previous chapter. In this study, the DE scheme presented by Storn et al. (1995) and Das et al. (2005) was applied to problems which can be grouped by using notation as *DE/rand/1/bin* strategy.

In the article of Janez, the version of self-adaptive DE is compared with the classical DE algorithm and the FADE algorithm conducted by Liu & Lampinen (2005). For comparison, some benchmark optimization problems from literature were tested by all algorithms. As a result of this comparison, it is deduced that “DE algorithm with self-adaptive control parameters setting is quite better or at least comparable to the standard DE algorithm and evolutionary algorithms from literature considering the quality of the solutions found with”. Their proposed algorithm yielded better results than the FADE algorithm.

The fuzzy adaptive differential evaluation algorithm (FADE) is a new variant of DE using fuzzy logic controllers in order to adapt the control parameters, scaling factor  $F_i$  and crossover rate  $CR_i$  for mutation and crossover operations. Like other proposed adaptive DE algorithms, the population size is assumed to be constant in advance and kept fixed through whole evolution process of FADE. When fuzzy logic controlled approach is tested with a set of 10 benchmark problems, it is concluded that FADE yields better results than the classical DE in high dimensional problems.

In the DE algorithm above, a novel self-adapting parameter scheme improved by Brest et al. (2006) was used, known as *j*DE. It uses self-adapting mechanism on the control parameters  $F$  and  $CR$ . Brest et al. used the self-adaptive control mechanism of “rand/1/bin”. This strategy is mostly used in practice such as Storn et al. (1997), Gamperle et al.(2002), Liu and Lampien (2002), Sun et al. (2004).

In the article of Brest et. al. (2006), a self-adaptive control mechanism was used for changing the control parameters  $F$  and  $CR$  when the program is run. The third control parameter  $NP$  did not alter during the run. Each individual in population was extended with parameter values. The control parameters having been adjusted by means of evaluations are  $F$  and  $CR$ . Both of them were applied in individual levels. The better values of these encoded control parameters direct to better individuals that, in turn, are more probable to survive and produce offspring, hence propagate these better parameter values.

$\vec{X}_1^g$	$F_1^g$	$CR_1^g$
$\vec{X}_2^g$	$F_2^g$	$CR_2^g$
.....	.....	.....
$\vec{X}_{NP}^g$	$F_{NP}^g$	$CR_{NP}^g$

**Figure 2.2. Self-Adapting: Encoding Aspect**

It is very effective and converges much faster than the traditional DE, especially when the dimensionality of the problem is very high and important and so-called problem is complicated. In *j*DE, each individual is given its own  $F_i$  and  $CR_i$  values. Initially, they are assigned to  $CR_i = 0,5$  and  $F_i = 0,9$  and new control parameters are calculated as follows:

$$F_i^g = \begin{cases} F_l + r_1 \cdot F_u & \text{if } r_2 < \tau_1 \\ F_i^{g-1} & \text{otherwise} \end{cases} \quad (8)$$

$$CR_i^g = \begin{cases} r_3 & \text{if } r_4 < \tau_2 \\ CR_i^{g-1} & \text{otherwise} \end{cases} \quad (9)$$

where  $r_j \in \{1,2,3,4\}$  are uniform random numbers in the range  $[0,1]$ .  $\tau_1$  and  $\tau_2$  denote the probabilities to adjust the  $F$  and  $CR$ . They are taken as  $\tau_1 = \tau_2 = 0,1$  and  $F_l = 0,1$  and  $F_u = 0,9$ .

## 2.4 JADE

In this section, a new DE algorithm called JADE that implements a mutation strategy “DE/current-to-*p*-best” including optional archive and controls  $F$  and  $CR$  in and adaptive manner with self-adaptive parameters  $\mu_F$  and  $\mu_{CR}$ . JADE adopts the



same binary crossover and one-to-one selection as the classic DE. The algorithm JADE will be introduced in three sections:

#### 2.4.1 DE/current-to-pbest

DE/rand/1 is the first mutation strategy developed for DE by Storn & Price (1997) and it said by Babu & Jehan (2003) that it is known as the most successful and widely used DE scheme in the literature. Nevertheless Gamperle et al. (2002) in his article, claims that DE/best/2 might have some advantages on DE/rand/1 and Pahner & Hameyer (2000) favours DE/rand/1 for the most technical problems investigated. Also Mezure-Montes et al. (2006) argue that the incorporation of best-solution information is beneficial and use DE/current-to-best/1 in their algorithm. When compared to DE/rand/ $k$ , other greedy strategies such as DE/current-to-best/ $k$  and DE/best/ $k$  generally have faster convergence rate. On the other hands, their utilization of best-solution is likely to cause such problems as premature convergence due to the resultant decreased population diversity.

Because of the fast but less reliable convergence performance of greedy strategies, a new mutation strategy called as DE/current-to-p-best is introduced in order to be able to serve as the basis of self-adaptive DE algorithm. In DE/current-to-p-best/1, a mutation vector is generated in the following manner:

$$v_{ij}^g = x_{ij}^{g-1} + F_i \times (x_{i,pbest}^{g-1} - x_{ij}^{g-1}) + F_i \times (x_{aj}^{g-1} - x_{bj}^{g-1}) \quad (10)$$

where  $x_{i,pbest}^{g-1}$  is uniformly chosen as one of the top  $100p\%$  individuals of the current population with  $p \in (0,1]$ , and  $F_i$  is a mutation factor which has been associated with  $x_{ij}^{g-1}$  and it is again created by the adaptation process at each iteration as algorithm is being run. DE/current-to- $p$ -best is indeed a generalization of DE/current-to-best. Any of the top  $100p\%$  solutions can be randomly selected in order to play the role of the single best solution in DE/current-to-best.

#### 2.4.2 Self-Adaptation of Parameters

In JADE, the self-adaptation is applied to update the parameters  $\mu_{CR}$  and  $\mu_F$  used for generating mutation factor  $F_i$  and crossover probability  $CR_i$  associated with

each individual vector  $x_i$ , respectively. The  $F_i$  and  $CR_i$  are then used for creating the trial vector  $u_i$ . At each generation  $g$ , crossover probabilities  $CR_i$  are generated based on an independent normal distribution with  $randn_i(\mu_{CR}, 0, 1)$  of mean  $\mu_{CR}$ , standard deviation 0,1 and truncated to the interval (0,1]:

$$CR_i = randn_i(\mu_{CR}, 0, 1) \quad (11)$$

Denoting that  $S_{CR}$  as the set of all successful crossover probabilities  $CR_i$ 's at generation  $g$ . The mean  $\mu_{CR}$  is then updated by using following formula:

$$\mu_{CR} = (1 + c) * \mu_{CR} + c * mean(S_{CR}) \quad (12)$$

where  $c$  is a positive constant number between 0 and 1 and  $mean(.)$  is the usual arithmetic mean operation.

At each generation  $g$ , the mutation factor  $F_i$  of each individual  $x_i$  of the population is created independently based on the mixture of a uniform distribution  $rand_i(0, 1.2)$  and a normal distribution  $randn_i(\mu_F, 0.1)$ , and truncated to (0, 1.2]. That means:

$$F_i^g = \begin{cases} rand_i(0, 1.2) & \text{if } i < I_{1/3} \\ randn_i(\mu_F, 0.1) & \text{otherwise} \end{cases} \quad (13)$$

where  $I_{1/3}$  denotes a random collection of one-third indicates of the set  $\{1, 2, \dots, NP\}$ . Denoting that  $S_F$  is the set of all successful mutation factors  $F_i$ 's at a generation  $g$ . The mean  $\mu_F$  of the normal distribution is updated by using following formula:

$$\mu_F = (1 - c) * \mu_F + c * L(S_F) \quad (14)$$

where  $L(.)$  is the Lehmer mean:

$$L(S_F) = \frac{\sum_{F \in S_F} F^2}{\sum_{F \in S_F} F} \quad (15)$$

### 2.4.3 Explaining JADE Algorithm Setting

There are some principles followed in order to provide the adaptation of  $\mu_{CR}$ . Better values of control parameters have more tendencies to create individuals which are more likely to survive and hence these values should be spread. The basic operation is therefore to keep successful crossover probabilities and make use of them when it is necessary to guide the generation process of future  $CR_i$ 's. The standard deviation of the process is aimed to set to be small because other else the self-adaptation would not work properly. For instance, as an extreme case of an infinite standard deviation, the truncated normal distribution becomes a uniform distribution and thus independent of the value of  $\mu_{CR}$ . In JADE, the standard deviation of both mutation and crossover parameters is set to be 0.1.

First, compared to  $CR$ , there are two different operations in the adaptation of  $F$ . At each generation, only two thirds of all  $F_i$ 's are generated based on a normal distribution while others are generated according to a uniform distribution. The component including normal distribution has small variance so it is useful for searching a suitable mutation factor in a manner similar to  $CR$  adaptation. On the other hands, the uniform distribution component helps diversify the mutation factors and therefore block from getting premature convergence which are quite possible to occur in greedy mutation strategies when the mutation factors are highly around a fixed value.

Secondly, using the Lehmer mean given in equation (15) is much better than the arithmetic mean used in  $\mu_{CR}$  adaptation because the adaptation of  $\mu_F$  places more weights on larger successful mutation factors. Arithmetic means of  $S_F$  tend to be smaller than the optimal value of the mutation factor hence causes to have a smaller  $\mu_F$  and premature convergence at the end. The decrease in trend mainly is because of inconsistency between success probability and progress rate of an evolution search. On the other hands, Zang & Sanderson (2007) states that the DE/current-to- $p$ -best with small  $F_i$  is similar to an  $(1 + 1)$  ES scheme in the sense that both generating an offspring in the small neighbourhood of the base vector. For  $(1 + 1)$  ES, it is known that it is better to keep the mutation variance as small as possible in order to have a higher successful probability. However, when a mutation variance is close to 0, it obviously causes to get a trivial evolution progress. A simple and effective way is to

place more weight on larger successful mutation factors in order to achieve a rapid rate of progress.

## **2.5 Ensemble Differential Evolution**

In this study, an ensemble approach for DE algorithm was used. Mallipedi and Suganthan (2011) stated that the performance of conventional DE as solving real world optimisation problem relies on the selected mutation and crossover strategy and its associated parameter values. Nevertheless, different type of optimization problems may require different mutation strategies with different parameter values regarding to the nature of the problem and necessary computation resources. It can also be said that different mutation strategies with different parameter settings could be better during stages of the evolution than a single mutation strategy using unique parameter settings as traditional DE. By means of the motivation by these observations, Mallipedi & Suganthan (2011) have proposed an ensemble of mutation and crossover strategies and parameter values for DE where a pool of mutation strategies, along with a pool of values to each associated parameter tries to generate a better offspring population. Qin et al (2009) has mentioned that the candidate pool of mutation and crossover must be restrictive in order to avoid the undesirable influences of less effective mutation strategies and parameters. The mutation strategies or the parameters in the pool should own diverse characteristics, in order that they can demonstrate different performance characteristics during different levels of the evolution, as focusing on a particular problem.

Ensemble DE contains a pool of mutation and crossover strategies with a pool of values for each of the control parameters associated. A different mutation strategy is randomly assigned to each member of initial population with the associated values obtained from the perspective pools. Therefore, trial vectors are produced by the population members with the assigned mutation strategy and parameter values. When the generated trial vector is better than the target vector, the mutation strategy and corresponding parameter values are held with the trial vector which becomes a parent vector of next generation. The combination of the mutation strategy and the parameter values creating a better trial vector than the parent are kept. When the target vector is better than the trial vector, then the target vector is initialized again with a mutation strategy and the associated parameter values from either the pool or

the successful combinations of mutation strategy and the associated control parameter for the following generations.

The ensemble idea was presented in Tasgetiren et al. (2010) and Mallipeddi et al. (2011). In both studies, ensemble of mutation strategies is considered to improve EDE algorithm. Inspiring from these studies, following mutation strategies ( $M_i$ ) have been taken into consideration in this thesis.

$M_1$ : *DE/rand/1/bin*:

$$v_{ij}^g = x_{aj}^{g-1} + F \times (x_{bj}^{g-1} - x_{cj}^{g-1}) \quad (16)$$

$M_2$ : *DE/rand/2/bin*:

$$v_{ij}^g = x_{aj}^{g-1} + F \times (x_{bj}^{g-1} - x_{cj}^{g-1}) + F \times (x_{dj}^{g-1} - x_{ej}^{g-1}) \quad (17)$$

$M_3$ : *DE/best/1/bin*:

$$v_{ij}^g = x_{best,j}^{g-1} + F \times (x_{aj}^{g-1} - x_{bj}^{g-1}) \quad (18)$$

$M_4$ : *DE/best/2/bin*:

$$v_{ij}^g = x_{best,j}^{g-1} + F \times (x_{aj}^{g-1} - x_{bj}^{g-1}) + F \times (x_{cj}^{g-1} - x_{dj}^{g-1}) \quad (19)$$

where  $a, b, c, d, e$  are five randomly chosen individuals from the target population such that  $(a \neq b \neq c \neq d \neq e \neq i \in (1, \dots, NP))$  and  $j = 1, \dots, D$ .  $F > 0$  is a mutation scale factor affecting the differential variation between two individuals, and  $x_{best,j}^{g-1}$  is the best vector in generation  $g - 1$ .

*Procedure EDE()*

- Step 1.*      *Set parameters  $g = 0, NP = 100, M_{max} = 4$*
- Step 2.*      *Establish initial population randomly*  
 $P^g = \{x_1^g, \dots, x_{NP}^g\}$  with  $x_i^g = \{x_{i1}^g, \dots, x_{iD}^g\}$
- Step 3.*      *Assign a mutation strategy to each individual randomly*

- $M_i = \text{rand}() \% M_{max}$  for  $i = 1, \dots, NP$
- Step 4. Evaluate population and find  $x_{best}^g$   
 $f(P^g) = \{f(x_1^g), \dots, f(x_{NP}^g)\}$
- Step 5. Assigning  $CR[i] = 0.5$  and  $F[i] = 0.9$  to each individual
- Step 6. Repeat the following for each individual  $x_i^g$
- obtain  $v_i^g = M_i(x_i^g)$
  - obtain  $u_i^g = CR_i(x_i^g, v_i^g)$
  - obtain  $x_i^g = \begin{cases} u_i^g & \text{if } f(u_i^g) \leq f(x_i^{g-1}) \\ x_i^{g-1} & \text{otherwise} \end{cases}$
  - if  $f(u_i^g) > f(x_i^{g-1})$ ,  $M_i = \text{rand}() \% M_{max}$
  - if  $(f(x_i^g) \leq f(x_{best}^g))$ ,  $x_{best}^g = x_i^g$
  - Update  $F_i^g$  and  $CR_i^g$
- Step 7. If the stopping criterion is not met, go to Step 6,  
else stop and return  $\pi_{best}$

**Figure 2.3. Outline of EDE Algorithm**

## 2.6 Opposition-Based Differential Evolution

The concept of opposition-based learning was firstly presented by Tizhoosh (2005) and its applications can be found in Tizhoosh (2005) and Tizhoosh (2006). Rahnamayan et al. has lately introduced an ODE for faster global search and optimization. This algorithm also provides important applications to the noisy optimization problems. The traditional DE is changed by taking advantage of *opposition number* based optimization concept in three different levels, namely, initialization of population, generation jumping, and local improvement in the population's best member. When a *priori* information about the actual optima is not provided, an EA begins with *random guesses*. It is possible to increase the possibility of starting with a better solution by instantaneously checking fitness of the *opposite solution*. By means of this way, the fitter one which is either guess or opposite guess can be selected as an initial solution. As mentioned in article of Tizhoosh (2005), according to the probability theory, %50 of the time a guess have may have lower fitness value than its opposite guess. Hence, to start with the fitter of the two guesses is more probable to converge faster than opposite guess. The same approach is likely to be applied continuously to each solution in the current population. When population starts to converge into a smaller neighbourhood which surrounds and optimum point, taking opposition moves may be able to increase the variability of the

population. Moreover, as the population converges, the magnitude of the difference vector will be smaller. Nevertheless, difference vectors obtained by using parents that just underwent an opposite move will be large thereby resulting larger perturbation in the mutant vector. Thus, ODE has superior capability to get rid of local optima basins.

Das & Suganthan (2011) have defined opposite numbers as given below:

*Definition 1: Let  $x$  be a real number defined in closed interval  $[a, b]$ , i. e.,*

*$x \in [a, b]$ . Then the opposite number  $x^U$  of  $x$  may be defined as:*

$$x^U = a + b - x \quad (20)$$

The ODE changes the classical DE by using the concept of opposite numbers at the following different stages:

- *Opposition based population initialization:* Firstly, a population is generated according to uniform distribution randomly  $P(NP)$  and then the opposite population  $OP(NP)$  is calculated. The  $i$ th opposite individual corresponding to  $i$ th parameter vector of  $P(NP)$  is as given in the article of Feoktistov and Janaqi (2004):

$OP_{ij} = a_{ij} + b_{ij} - P_{ij}$ , where  $i = 1, 2, \dots, NP$  and  $j = 1, 2, \dots, D$   
 $a_{ij}$  and  $b_{ij}$  denote the interval boundaries of  $j$ th and  $k$ th vector i. e.

$$x_{ij} \in [a_{ij}, b_{ij}]$$

As a final, NP fittest individuals are selected from the  $\{P(NP), OP\{NP)\}$  as the initial population.

- *Opposition based generation jumping:* in this stage, after each iteration, instead of creating new population by evolutionary process, the opposite population is found by using a known probability  $Jr(\in (0, 0.04))$  and the  $NP$  fittest individuals can be selected both from the current population and its corresponding opposite population.

- *Opposition based best individual jumping*: in this stage, in the current population, difference-offspring of the best individual is generated by the following:

$$x_{newbest,j}^g = x_{best,j}^{g-1} + F'(x_{aj}^{g-1} - x_{bj}^{g-1}) \quad (21)$$

where  $a$  and  $b$  are mutually different integers indices selected from population and  $F'$  is a real constant between 0 and 1. In the following step, the opposite of offspring created as  $x_{opp-newbest}^g$  and finally the current best member is changed with the fittest member of the set including best, new best and opposition of new best.

## 2.7 Ensemble DE with VNS

In order to make an ensemble of *DE* algorithms, it is inspired from the *VNS* algorithm and the idea of neighbourhood change in the *VNS* algorithm is used. Gamperle et al. (2002) and Brest et al. (2006) state that the core idea is to apply different mutation strategies in the ensemble because it is known that the performance of *DE* algorithms is very sensitive to mutation strategies selected. In this thesis, the first four mutation strategies for the ensemble purposes are chosen in order to be employed in *VNS* loops and the fifth one is not employed in *VNS* algorithm. Mutation strategies  $M_3$  and  $M_4$  are taken from the article of Elsayed et al. (2014) in which genetic algorithm is used for optimization problems:

$M_1 = DE/rand /1/bin$ :

$$v_{ij}^g = x_{aj}^{g-1} + F(x_{bj}^{g-1} - x_{cj}^{g-1}) \quad (22)$$

$M_2 = DE/p - best /1/bin$ :

$$v_{ij}^g = x_{p,j}^{g-1} + F(x_{bj}^{g-1} - x_{cj}^{g-1}) \quad (23)$$

$M_3 = GA Operator in [Elsayed et al.]$ :



$$v_{ij}^g = x_{aj}^{g-1} + F(x_{bj}^{g-1} - x_{cj}^{g-1}) \text{ such that } f(x_a) \leq f(x_b) \quad (24)$$

$M_4 = GA \text{ Operator in [Elsayed et al.]}$ :

$$v_{ij}^g = x_{cj}^{g-1} + F(x_{aj}^{g-1} - x_{bj}^{g-1}) \text{ such that } f(x_a) \leq f(x_b) \quad (25)$$

$M_5 = DE/current \text{ to rand } /1/bin$ :

$$u_{ij}^g = x_{ij}^{g-1} + K(x_{aj}^{g-1} - x_{ij}^{g-1}) + F(x_{bj}^{g-1} - x_{cj}^{g-1}) \quad (26)$$

In mutation strategies given above,  $K$  is randomly chosen within the range  $[0,1]$ .  $x_{p,j}$  is the individual selected by using the tournament selection with size of 2. In other words, two individuals are randomly selected from the population and the better one is picked up.

So as to improve the  $EDE\_VNS$  algorithm, four different  $VNS$  local searches are devised in order to generate trail individuals in an ensemble framework. Moreover, a neighbourhood  $N_k$  by using a mutation strategy and a crossover operator together is generated to clarify the  $VNS$  local searches as follows:

$$N_k(x) = M_k(y), CR(x, y) \quad (27)$$

Equation (27) indicates that mutation strategy  $M_k$  is used for generating a mutant individual  $y$  first so as to find a neighborhood of an individual  $x$ , and then mutant individual  $y$  with individual  $x$  by means of crossover operator is recombined in equation (5). With this definition and a temporary individual  $\tau$ , first of all  $VNS_1$  algorithm is developed as follows:

*Procedure*  $VNS_1(x_i^g)$

$k_{max} = 2$

$k = 1$

$\tau = x_i^g$

do{

$x^* = N_k(\tau)$

if  $f(x^*) < f(\tau)$

$N_1(\tau) = M_1(y), CR(\tau, y)$

$N_2(\tau) = M_2(y), CR(\tau, y)$

```

         $\tau = x^*$ 
         $k = 1$ 
    else
         $k = k + 1$ 
    }while( $k \leq k_{max}$ )
return  $\tau$ 
endprocedure

```

Figure 2.4.  $VNS_1$  Algorithm.

The performance of  $VNS$  algorithms depends on the choice of the first neighborhood strategy since the algorithm is quite sensitive. It should be noted that as long as the first neighbourhood keeps improving the current solution, the neighbourhood counter  $k$  will become 1 meaning that the first neighbourhood will be used. Other else, the neighbourhood counter will be increased to 2 because first neighbourhood has stopped improving the current solution. It indicates that second neighbourhood will be employed. If the second neighbourhood manages to improve the solution, the algorithm gets back to the first neighbourhood again until the second neighbourhood fails to improve the current solution.

As seen in Fig. 2.4., the following neighbourhood structures are used in the  $VNS_1$  algorithm:

$$N_1(\tau) = M_1(y), CR(\tau, y), N_2(\tau) = M_2(y), CR(\tau, y) \quad (28)$$

The second  $VNS_2$  algorithm can be gained by changing the sequence of the neighborhoods as follows:

$$N_1(\tau) = M_2(y), CR(\tau, y), N_2(\tau) = M_1(y), CR(\tau, y) \quad (29)$$

Similar  $VNS$  algorithms can be obtained by neighbourhood structures through the use of mutation strategies  $M_3$  and  $M_4$ .

The third  $VNS_3$  can be obtained by the following neighbourhoods as follows:

$$N_1(\tau) = M_3(y), CR(\tau, y), N_2(\tau) = M_4(y), CR(\tau, y) \quad (30)$$

The fourth  $VNS_4$  algorithm can be obtained by changing the sequence of the neighborhoods as follows:

$$N_1(\tau) = M_4(y), CR(\tau, y), N_2(\tau) = M_3(y), CR(\tau, y) \quad (31)$$

The fifth strategy generates directly applying the mutation strategy to the individuals.

$$u_{ij}^g = x_{ij}^{g-1} + K(x_{aj}^{g-1} - x_{ij}^{g-1}) + F(x_{bj}^{g-1} - x_{cj}^{g-1}) \quad (32)$$

The pseudo codes of  $VNS$  algorithms are given in Fig. 2.5. to 2.7.

```

Procedure  $VNS_2(x_i^g)$ 
 $k_{max} = 2$ 
 $k = 1$ 
 $\tau = x_i^g$ 
do{
     $x^* = N_k(\tau)$ 
    if  $f(x^*) < f(\tau)$ 
         $\tau = x^*$ 
         $k = 1$ 
    else
         $k = k + 1$ 
}while( $k \leq k_{max}$ )
 $u_i^g = \tau$ 
return  $\tau$ 
endprocedure

```

Figure 2.5.  $VNS_2$  Algorithm.

```

Procedure  $VNS_3(x_i^g)$ 
 $k_{max} = 2$ 
 $k = 1$ 
 $\tau = x_i^g$ 
do{
     $x^* = N_k(\tau)$ 
    if  $f(x^*) < f(\tau)$ 
         $\tau = x^*$ 
         $N_1(\tau) = M_3(y), CR(\tau, y)$ 
         $N_2(\tau) = M_4(y), CR(\tau, y)$ 

```

```

                                k = 1
else
                                k = k + 1
}while(k ≤ kmax)
return τ
endprocedure

```

Figure 2.6. *VNS<sub>3</sub> Algorithm.*

```

Procedure VNS4(xig)
kmax = 2
k = 1
τ = xig
do{
                                x* = Nk(τ)                                N1(τ) = M4(y), CR(τ, y)
                                if f(x*) < f(τ)                                N2(τ) = M3(y), CR(τ, y)
                                    τ = x*
                                    k = 1
                                else
                                    k = k + 1
                                }while(k ≤ kmax)
return τ
endprocedure

```

Figure 2.7. *VNS<sub>4</sub> Algorithm.*

In *VNS* algorithms given above, a novel self-adapting parameter scheme developed by Brest et al. is employed, so called *jDE*. It is very simple and effective. In *jDE*, each individual has its own *F* and *CR* values, they are updated according to the equations given in (8) and (9) respectively.

### 2.7.1 Generating Initial Population

In this thesis, target population is randomly established by using the equation (2). Nevertheless, the opposition-based learning algorithm in order to enrich the initial population and improve the solution is used. *OBL* is firstly proposed by Tizhoosh (2005) as a new method in computational intelligence and has been applied to more improve various heuristic optimization algorithms (Rahnamayan & Tizhoosh). *OBL* highly relies on the idea that as evaluating the solution of a given problem, its opposite solution is also probable to find a candidate solution which

might be closed to the global optimum. Inspired from *OBL*, generalized *OBL* (*GOBL*) is presented in Wang et al. (2011). It is assumed that  $x$  is the current solution with  $x \in [a, b]$ . Then its opposition solution is given by:

$$x^* = k * (a + b) - x \quad (33)$$

In *GOBL*, opposite solutions are obtained by dynamically updated interval boundaries in the population as follows:

$$x_{ij}^* = k * [a_j^g + b_j^g] - x_{ij}^g \quad (34)$$

$$a_j^g = \min(x_{ij}^g), b_j^g = \max(x_{ij}^g) \quad (35)$$

$$x_{ij}^* = \text{rand}(a_j^g, b_j^g) \text{ if } x_{ij}^* < x_{\min} \text{ or } x_{ij}^* > x_{\max} \\ i = 1, \dots, NP, j = 1, \dots, D, k = \text{rand}[0,1] \quad (36)$$

After establishing and evaluating the target population, the *GOBL* algorithm given above is also used for obtaining the opposite target individual. The better one is kept in the target population.

### 2.7.2 Generation of Trial Population

In order to generate trial individuals of population, VNS algorithm is applied through the solution process. After obtaining each individual from the VNS algorithm, an injection procedure is applied to trial individuals to diversify it escape from the local minima. By injection procedure, an individual from the target population is selected by using tournament selection with size of 2. Therefore, depending on the injection probability, some good dimensional values are injected to the trial individuals in such a way that a uniform random number  $r$  is less than the injection probability  $iP$ , that dimension is obtained from individual  $x_a$ , that is determined by using tournament selection procedure. Other else, the dimension of trial individual is retained. The injection procedure is given in Fig. 2.8. below:

```

for i = 1 to NP
  for j = 1 to D
    if (r < iP) then
      xaj = TournamentSelect()
      uij = xaj
    else
      uij = uij
    endifor
  endfor
endfor

```

Figure 2.8. Injection Procedure

### 2.7.3 Selection

When the selection of the next population is performed, EC and SF constraint handling methods that will be summarized in following sections are employed. For each individual in the trial population,  $\varepsilon(t)$  level is checked. The individual is treated as a feasible one, when its constraint violation is less than  $\varepsilon(t)$  level. Later on, the SF method is used in order to determine if trial individual will be able to survive to be in the next generation. Moreover, the SF method is simply used so as to update the best so far solution in the population. The pseudo code for EDE-VNS algorithm is provided below in Figure 2.9.

#### Procedure EDE\_VNS()

Set parameters  $g = 0, NP = 50, S_{max} = 5$

Step 1. Establish initial population randomly

$$P^g = \{x_1^g, \dots, x_{NP}^g\} \text{ with } x_i^g = \{x_{i1}^g, \dots, x_{iD}^g\}$$

Step 2. Apply OBL to population and choose the best ones

$$P^g = \{x_1^g, \dots, x_{NP}^g\} \text{ with } x_i^g = \{x_{i1}^g, \dots, x_{iD}^g\}$$

Step 3. Assign a strategy to each individual

$$M_i = \text{rand()} \% M_{max} \text{ for } i = 1, \dots, NP$$

Step 4. Evaluate population and find  $x_{best}^g$

$$f(P^g) = \{f(x_1^g), \dots, f(x_{NP}^g)\}$$

*Step 5. Determine  $e(0)$*

*Step 6. Repeat the following for each individual  $x_i^g$*

- *obtain  $u_i^g = VNS_i(x_i^g)$  through NFT*
- *apply injection to trial individual*
- *obtain  $x_i^{g+1}$  through  $e - \text{constraint method}$*
- *if  $(f(x_i^g) \leq f(x_{best}^g))$ ,  $x_{best}^g = x_i^g$  through SF*
- *Update  $e(g) = e(0) \left(1 - \frac{g}{gc}\right)$*

*Step 7. If the stopping criterion is not met, go to Step 6,  
else stop and return  $\pi_{best}$*

**Figure 2.9. Outline of EDE-VNS Algorithm**

### 3 CONSTRAINED REAL PARAMETER OPTIMIZATION

Most of the optimization problems included in science and engineering involve constraints. When constraints are present in the optimization problem, feasible region reduces and the search process of the problem gets more complicated. Evolutionary Algorithms (EAs) generally perform unconstrained searches. In order that the evolutionary algorithm can solve constrained optimization problems, it is required to put additional mechanisms to handle given constraints. In the literature, Coello (2002) gives important information about several constraint handling techniques in order to be used for EAs.

During solving constrained optimization problems, solution candidates that can satisfy all constraints are feasible individuals whereas individuals that cannot satisfy all constraints are infeasible individuals. One of the most important issues of constrained handling optimization is to determine how to deal with the infeasible individuals through the search process. One way to handle it is to totally ignore all infeasible individuals found and go on the search process with feasible individuals only. This kind of approach is probable to be ineffective because EAs are probabilistic search methods and important potential information in infeasible individuals may be lost because of completely disregarding them. When the search space is discontinuous, EAs are also possible to be trapped in one of the local minima. Hence, different techniques have been developed to get the information held by infeasible individuals. Michalewicz & Schoenauer (1996) has grouped the methods to handle the constraints with EAs into four categories: preserving feasibility of functions, penalty functions, make a separation between feasible and infeasible solutions and hybrid methods. Wang et al. (2007) stated that a constrained optimization problem can be formulated as a multi-objective problem, nevertheless it is computationally intensive due to non-domination sorting.

According to *the no free lunch* (NFL) theorem stated in the article of Wolpert and Macready (1997), no single state-of-the-art constraint handling technique can be said better than all others on every problem. Therefore, solving a particular constrained problem needs many trial-and-error runs in order to be able to choose an appropriate constraint handling technique and to fine tune the associated parameters. This approach clearly has the disadvantage of suffering from unrealistic computational requirements in particular when the objective function of the problem



is computationally expensive as in the article of Jin (2005) or the solutions are needed in real-time. In the article of Mallipeddi & Suganthan (2010), an ensemble of constraint handling techniques was proposed as an effective alternative to the trial-and-error-based search for the best constrained handling technique with its best parameters for any given problem. Each constraint handling technique has its own population and each function is efficiently utilized by each of these populations.

As mentioned above, According to J.J. Liang et al. (2006), because evolutionary algorithms and other meta-heuristics behave as if they are unconstrained search technique because of their nature during optimization process, additional mechanism is required. Lately, the mostly used method is to generate the penalty functions to incorporate constraints. On the other hand, to solve a problem the optimum solution lies in the boundary between the feasible and the infeasible regions. In addition to this, penalty functions require an effective fine-tuning to decide the most suitable penalty factors in order to be used with meta-heuristics. Therefore, same ranking methods are proposed in the literature to handle the constraints.

In this thesis, 22 benchmark problems which were presented at CEC'2006 are taken into consideration to solve by using EDE-VNS algorithms. The list of benchmark functions are provided in Appendix 1. (<http://www.ntu.edu.sg/home/epnsugan/>).

All of the 22 test functions were defined with minimization problem as following:

$$\text{Minimize } f(x), x = [x_1, x_2, \dots, x_n] \text{ and } X \in S \quad (37)$$

subject to that constraints:

$$g_i(x) \leq 0, i = 1, \dots, q \quad (38)$$

$$h_j(x) = 0, j = q + 1, \dots, m \quad (39)$$

For the converting of equality constraints into inequality form, following strategy is used:

$$|h_j(x)| - \varepsilon \leq 0, \text{ for } j = q + 1, \dots, m \quad (40)$$

when equations (5) and (6) are satisfied, a solution  $x$  is defined as feasible. In accordance with the value of  $\varepsilon$  taken as  $\varepsilon = 0,0001$  in the special session, so it is taken as the same value in this study as well.

### 3.1 Benchmark Problems

The benchmark problems used for solving by *EDE – VNS* algorithm are taken from 2006 IEE Congress Evolutionary Computation (CEC'2006) competition of single objective constrained problems.

A variety of engineering optimization problems provided by (CEC' 2006) is taken into consideration in this thesis. These problems vary in terms of mathematical properties, presence and absence of function constraints, number of variables included, static and dynamic in nature, and modality. The number of constraints in each constrained problems also changes. The further information about these problems is given in table below provided in the article of Elsayed et al. (2014).

**Table 1 Details of the 24 test problems**

Pr	D	Objective Function	$ F / S (\%)$	LI	NI	LE	NE	a	Optimal
g1	13	Quadratic	0.00000	9	0	0	0	6	-15.000000000
g2	20	Nonlinear	99.9971	0	2	0	0	1	-0.8036191042
g3	10	Polynomial	0.0000	0	0	0	1	1	-1.0005001000
g4	5	Quadratic	52.1230	0	6	0	0	2	-30665.53867178
g5	4	Cubic	0.0000	2	0	0	3	3	5126.4967140071
g6	2	Cubic	0.0066	0	2	0	0	2	-6961.813875580
g7	10	Quadratic	0.0003	3	5	0	0	6	24.3062090681
g8	2	Nonlinear	0.8560	0	2	0	0	0	-0.0958250415
g9	7	Polynomial	0.5121	0	4	0	0	2	680.6300573745
g10	8	Linear	0.0010	3	3	0	0	6	7049.2480205286
g11	2	Quadratic	0.0000	0	0	0	1	1	0.7499000000
g12	3	Quadratic	4.7713	0	1	0	0	0	-1.0000000000
g13	5	Nonlinear	0.0000	0	0	0	3	3	0.0539415140
g14	10	Nonlinear	0.0000	0	0	3	0	3	-47.7648884595
g15	3	Quadratic	0.0000	0	0	1	1	2	961.7150222899

g16	5	Nonlinear	0.0204	4	34	0	0	4	-1.9051552586
g17	6	Nonlinear	0.0000	0	0	0	4	4	8853.5396748064
g18	9	Quadratic	33.4761	0	13	0	0	6	-0.8660254038
g19	15	Nonlinear	0.0000	0	5	0	0	0	32.6555929502
g21	7	Linear	0.0000	0	1	0	15	6	193.7245100700
g23	9	Linear	0.0000	0	2	3	1	6	-400.0551000000
g24	2	Linear	79.6556	0	2	0	0	2	-5.5080132716

According to the table,  $D$  is the number of decision variables,  $|F|/|S|$  is the estimated ratio between the feasible region and the search space.  $LI$  is the number of linear inequality constraints,  $NI$  is the nonlinear inequality constraints,  $LE$  is the number of linear equality constraints,  $NE$  is the number of nonlinear equality constraints, and  $\alpha$  is the number of active constraints. Optimal solutions for each problem are also given in the last column of the table.

### 3.2 Constraint Handling Methods

There are different types of constraint handling techniques provided in articles in the literature. In this thesis, the methods used for constraint handling are described the following sections:

#### 3.2.1 Superiority of Feasible Solution

SF (Superiority of Feasible Solutions) is one of the constraint handling methods proposed by Deb (2000). It is developed for constrained optimization. If two solutions  $x$  and  $y$  are compared, it can be said that  $x$  is considered as superior to  $y$  under the satisfaction of following conditions:

- $x$  is feasible but  $y$  is not feasible.
- $x$  and  $y$  are both feasible but  $x$  has a smaller objective value for minimization and greater objective value for maximization problem than  $y$ .
- $x$  and  $y$  are both feasible, but  $x$  has a smaller overall constraint violation  $v(x)$  as computed by the following formula:

$$G_i(x) = \max\{g_i(x), 0\} \quad i = 1, \dots, p$$

$$H_i(x) = \max\{|h_i(x)| - \delta, 0\} \quad i = p + 1, \dots, m$$

$$v(x) = \frac{\sum_{i=1}^p G_i(x) + \sum_{i=p+1}^m H_i(x)}{m} \quad (41)$$

where  $v(x)$  is the average violation of  $m$  number of constraints. Furthermore,  $\delta$  is the tolerance value for equality constraints and it is generally taken as 0,0001 in the literature.

In SF, feasible individuals are always taken into consideration as better than the infeasible ones. Two infeasible solutions are compared according to their overall constraint violations only, whereas two feasible solutions are compared according to their objective functions only. The aim of comparing infeasible solutions based on the overall constraint violation is to push the infeasible solutions to feasible solutions as much as possible, while comparison of two feasible solutions only aims to improve the value of the objective function. Hence, in Phase 1, infeasible solutions with low overall constraint violation are to be selected. In Phase 2, all the feasible ones are selected first and then infeasible ones with low overall constraint violation are selected. In Phase 3, only feasible ones with better objective function values are selected.

### 3.2.2 The Adaptive Penalty Function

In the article of Smith & Tate (2003), an adaptive penalty approach is introduced. In adaptive penalty function, the idea of *near feasibility threshold* also called NFT is used, in which both solutions with-in feasible region and the NFT-neighbourhood of the in the infeasible region are favoured. Moreover, to be able to differentiate the gap between the best feasible value and best infeasible value found so far, an adaptive part is also included in the penalty method. The adaptive penalty function is given as follows:

$$f_p(x) = f(x) + (f_{feas} - f_{all}) \sum_{i=1}^m \left( \frac{v_i(x)}{NFT_i} \right)^{\alpha_i} \quad (42)$$

where  $f_{feas}$  is the value of the best feasible solution yet obtained while  $f_{all}$  is the nonpenalized value of the best solution obtained so far. As Coello (2002) has mentioned in his article, the adaptive term in the formula above might result in zero- or over-penalty. Because of this reason, it is thought to be better to take only dynamic of the function with NFT threshold into account as following:

$$f_p(x) = f(x) + \sum_{i=1}^p \left( \frac{G_i(x)}{NFT_i} \right)^\alpha + \sum_{j=p+1}^m \left( \frac{H_j(x)}{NFT_j} \right)^\alpha \quad (43)$$

The basic form of the *NFT* method is introduced as  $NFT = \frac{NFT_0}{1+\lambda*t}$  where  $NFT_0$  is the initial value of *NFT* method;  $\lambda$  and  $t$  are user-defined positive value and generation counter, respectively.  $\alpha$  is severity parameter. In view of the conversion process of the equality constraints to the inequality constraints by subtracting  $\delta$  from the absolute value of the constraint value and  $\delta$  is determined in advance, the  $NFT_0$  is selected as 1e-4.

### 3.2.3 $\epsilon$ – Constraint (EC)

Takahama & Sakai (2006) has proposed another constraint handling method called  $\epsilon$ -constraint. In the basis of their research, an appropriate control for the epsilon parameter is required as the good feasible solutions for problems with equality constraints are attained. According to control generation notated by  $g_c$ , the  $\epsilon$  level is updated. After generation counter  $g$  becomes higher than control generation  $g_c$ , the  $\epsilon$  level is set to zero to finalize with feasible solutions. The solutions which have violations less than  $\epsilon(g)$  are taken to become feasible solutions for selection process in the next generation. The main notion can be explained with equations:

$$\epsilon(0) = v(x_\theta) \quad (44)$$

$$\epsilon(g) = \begin{cases} \epsilon(0) \times \left(1 - \frac{g}{g_c}\right)^{cp} & \text{if } (g < g_c). \\ 0 & \text{otherwise} \end{cases} \quad (45)$$

where  $x_\theta$  is the top  $\theta$ -th individual.

### 3.2.4 Stochastic Ranking

Runarsson & Yao (2000) has presented *SR* method in order to achieve a balance between objective and the overall constraint violation in a stochastic way. A probability factor  $p_f$  is required in order to determine if the objective function value or the constraint violation value determines the rank of each individual in population. Basic form of *SR* is presented in figure below:

```
if (no constraint violation or rand < pf)
    Rank based on the objective value only
else
    Rank based on constraint violation only
end
```

Figure 3.10. Stochastic Ranking

Using evolution strategies and differential variation, Runarsson & Yao (2005) has proposed the improved version of the (*ISR*). In *SR*, comparison between two individuals might be performed based on only either objective value or constraint violation as randomly determined. Hence, infeasible solutions yielding better objective value are possible to be selected in all three phases of the evolution. Mallipeddi & Suganthan (2010) used the modified version of *SR* presented by Runarsson & Yao (2005). They have maintained the value of  $p_f$  a linearly decreasing function from  $p_f = 0,475$  in the initial generation to  $p_f = 0,025$  in the final generation instead of a constant value.

### 3.3 Ensemble of Constraint Handling

Each constrained optimization problem would differ from one another according to the ratio between feasible search space and the whole search space, multimodality and the nature of constraint functions. Mallipeddi & Suganthan (2010) that since evolutionary algorithms uses stochastic nature to solve optimization problems, the evaluation could possibly follow the different paths in every run even when the same problem is solved by using the same algorithm. Hence, depending upon such factors as the ratio between feasible search space and the whole search space, multimodality of the problem, nature of equality/inequality constraints, the chosen EA, global exploration, local exploitation stages of the search algorithm, different constraint handling methods might be useful in different stages of the search

process. In view of the strong relationship between these diverse factors and the randomness of the evolutionary algorithms, it is not practical to determine which constraint handling method outperforms the others or which one is the best during a particular stage of the evolution in order to solve a given problem when using a given EA. By means of these observations, Mallipeddi & Suganthan (2010) developed the *ECHT* to implicitly take advantage of the match between constraint handling methods, characteristics of the so-called problem, chosen EA and the exploration-exploitation stages of the search process.

Jin (2005) states that it might take several minutes to several hours in order to compute the objective value of a real-world problem. Thus, it is thought to be difficult to find a better constraint handling method for any problem by using trial-and-error method. The computation time spent on finding searching for a better constraint handling method can be saved by using the proposed ensemble method.

In the article of Mallipeddi & Suganthan (2010), *ECHT* with four different constrained handling methods given above sections, each of them has its own population and parameters. Each population produces its own offspring according to a constraint handling method. The parent population corresponding to a particular constraint handling method competes with both its own spring population and the offspring population of the other three constraint handling methods. Because of this fact, it can be said that an offspring produced by a specific constraint handling method could be refused by its own population, however might be accepted by the population of other constraint handling methods. When the evolution of the objective or constraint functions is rather expensive, it is better to include more constraint handling methods in the ensemble in order to take more advantages of each function call. And if a particular constraint handling technique is most suitable to the search method and the problem during a point in the search process, the offspring population produced by using that constraint handling method will be dominating the other and enter other populations as well. In the following generations, these superior offspring will be parents in other populations. When the constraint handling methods selected to form an ensemble are similar in their nature populations associated with the each of the constraint handling methods might lose diversity and the search ability of *ECHT* could worsen. Hence, the performance of *ECHT* can be developed by getting constraint handling methods with diverse and competitive nature.

### 3.4 Computational Results of Constrained RPO

The EDE-VNS Algorithm was coded in C++ and run on an Intel P4 1.33 GHz Laptop PC with 256 MB memory. The population size is taken as  $NP=50$ . The  $NFT_0$  is fixed at 0.001. The diversification probability is taken as 0.005. For the EC constraint handling method, following parameters are used as  $\theta = 0.95xNP$ ,  $t_c = 0.4 * MaxGen$  and  $cp = 2$ . It was carried out 25 replications for each benchmark problem and average, minimum and standard deviation of 25 replications are provided. It should be noted that real numbers are rounded to zero after 10 digits in the standard deviation calculations.

DE Algorithm with ensemble strategies and VNS (EDE-VNS) was compared to the best performing algorithms from the literature such as GA-MPC (Elsayed et al. 2013), APF-GA (Tessema & Yen, 2009), MDE (Mezura-Montes et al., 2006), ECHT-EP2 (Mallipedi et al., 2010). The presentation of overall analysis and comparison based on the results are given in Table 2. As seen in the table 2, proposed algorithm EDE-VNS was run for 240,000 and 500,000 function evaluations. Since it is possible to find some algorithms run for 500,000 function evaluations in the literature, it is thought that proposed algorithm is supposed to be more likely to find more optimal solutions with zero standard deviations when number of function evaluation increases. It is due to the fact that ensemble of mutation strategies help the functions converge slowly to the optimal solution. Therefore, more function of evaluations could yield better results. EDE-VNS algorithm with 240,000 function evaluations was able to find the optimal solutions with zero standard deviations for 18 out of 22 benchmark problems, while the same algorithm with 500,000 function evaluations was able to find the optimal solutions with zero standard deviations for 21 out of 22 benchmark problems. The EDE-VNS algorithm with 240,000 function evaluations was slightly better than APF-GA and ECHT-EP2 algorithms, since they were able to find 13 and 14 out of 22 benchmark problems respectively. The performance of MDE is obviously better than these two algorithms as well. GA-MPC and MDE algorithms were better than EDE-VNS algorithm with 240,000 function evaluations. However, MDE was run for 500,000 function evaluations and if MDE is compared to EDE-VNS with 500,000 function evaluations, it is obvious that EDE-VNS is better than MDE. The clear winner is GA-MPC algorithm because of the fact that it was able to find 20 optimal solutions with 240,000 function evaluations. It also should be highlighted EDE-VNS algorithm with 500,000 function evaluations is



competitive to best performing algorithm GA-MPC, since it is able to find 21 optimal solutions with zero standard deviations. In 2 benchmark problems, the standard deviation of EDE-VNS algorithm was smaller than both GA-MPC and MDE, respectively. In summary, the EDE-VNS algorithm with both 240,000 and 500,000 functions evaluations are competitive to the best performing algorithms taken from the literature.

Problem		EDE-VNS	EDE-VNS	GA-MPC	APF-GA	MDE	ECHT-EP2
	FEs	240,000	500,000	240,000	500,000	500,000	240,000
g01	Best	<b>-15.0000</b>	<b>-15.0000</b>	<b>-15.0000</b>	<b>-15.0000</b>	<b>-15.0000</b>	<b>-15.0000</b>
	Avg	<b>-15.0000</b>	<b>-15.0000</b>	<b>-15.0000</b>	<b>-15.0000</b>	<b>-15.0000</b>	<b>-15.0000</b>
	Std	<b>0.00E-00</b>	<b>0.00E-00</b>	<b>0.00E-00</b>	<b>0.00E-00</b>	<b>0.00E-00</b>	<b>0.00E-00</b>
g02	Best	<b>-0.8036191</b>	<b>-0.8036191</b>	<b>-0.8036191</b>	<b>-0.803601</b>	<b>-0.8036191</b>	<b>-0.8036191</b>
	Avg	<b>-0.8036191</b>	<b>-0.8036191</b>	-0.802921	-0.803518	-0.78616	<b>0.7998220</b>
	Std	0.0000	<b>0.00E-00</b>	2.4150E-03	1.00E-04	1.26E-02	6.29E-03
g03	Best	<b>-1.0005</b>	<b>-1.0005</b>	<b>-1.0005</b>	<b>-1.001</b>	<b>-1.0005</b>	<b>-1.0005</b>
	Avg	<b>-1.0005</b>	<b>-1.0005</b>	<b>-1.0005</b>	<b>-1.001</b>	<b>-1.0005</b>	<b>-1.0005</b>
	Std	<b>0.00E-00</b>	<b>0.00E-00</b>	<b>0.00E-00</b>	<b>0.00E-00</b>	<b>0.00E-00</b>	<b>0.00E-00</b>
g04	Best	<b>-30665.54</b>	<b>-30665.539</b>	<b>-30665.539</b>	<b>-30665.539</b>	<b>-30665.539</b>	<b>-30665.539</b>
	Avg	<b>-30665.54</b>	<b>-30665.539</b>	<b>-30665.539</b>	<b>-30665.539</b>	<b>-30665.539</b>	<b>-30665.539</b>
	Std	<b>0.00E-00</b>	<b>0.00E-00</b>	<b>0.00E-00</b>	1.00E-04	<b>0.00E-00</b>	<b>0.00E-00</b>
g05	Best	<b>5126.497</b>	<b>5126.497</b>	<b>5126.497</b>	<b>5126.497</b>	<b>5126.497</b>	<b>5126.497</b>
	Avg	<b>5126.497</b>	<b>5126.497</b>	<b>5126.497</b>	5127.5423	<b>5126.497</b>	<b>5126.497</b>
	Std	<b>0.00E-00</b>	<b>0.00E-00</b>	<b>0.00E-00</b>	1.4324E+00	<b>0.00E-00</b>	<b>0.00E-00</b>
g06	Best	<b>-6961.814</b>	<b>-6961.814</b>	<b>-6961.814</b>	<b>-6961.814</b>	<b>-6961.814</b>	<b>-6961.814</b>
	Avg	<b>-6961.814</b>	<b>-6961.814</b>	<b>-6961.814</b>	<b>-6961.814</b>	<b>-6961.814</b>	<b>-6961.814</b>
	Std	<b>0.00E-00</b>	<b>0.00E-00</b>	<b>0.00E-00</b>	<b>0.00E-00</b>	<b>0.00E-00</b>	<b>0.00E-00</b>
g07	Best	<b>24.3062</b>	<b>24.3062</b>	<b>24.3062</b>	<b>24.3062</b>	<b>24.3062</b>	<b>24.3062</b>
	Avg	<b>24.3062</b>	<b>24.3062</b>	<b>24.3062</b>	<b>24.3062</b>	<b>24.3062</b>	24.3063
	Std	<b>0.00E-00</b>	<b>0.00E-00</b>	<b>0.00E-00</b>	<b>0.00E-00</b>	<b>0.00E-00</b>	3.19E-05
g08	Best	<b>-0.095825</b>	<b>-0.095825</b>	<b>-0.095825</b>	<b>-0.095825</b>	<b>-0.095825</b>	<b>-0.095825</b>
	Avg	<b>-0.095825</b>	<b>-0.095825</b>	<b>-0.095825</b>	<b>-0.095825</b>	<b>-0.095825</b>	<b>-0.095825</b>
	Std	<b>0.00E-00</b>	<b>0.00E-00</b>	<b>0.00E-00</b>	<b>0.00E-00</b>	<b>0.00E-00</b>	<b>0.00E-00</b>

g09	Best	<b>680.630</b>	<b>680.630</b>	<b>680.630</b>	<b>680.630</b>	<b>680.630</b>	<b>680.630</b>
	Avg	<b>680.630</b>	<b>680.630</b>	<b>680.630</b>	<b>680.630</b>	<b>680.630</b>	<b>680.630</b>
	Std	<b>0.00E-00</b>	<b>0.00E-00</b>	<b>0.00E-00</b>	<b>0.00E-00</b>	<b>0.00E-00</b>	<b>0.00E-00</b>
g10	Best	<b>7049.248021</b>	<b>7049.24802</b>	<b>7049.24802</b>	<b>7049.24802</b>	<b>7049.24802</b>	7049.2483
	Avg	7049.248022	<b>7049.24802</b>	<b>7049.24802</b>	7077.6821	<b>7049.24802</b>	7049.2490
	Std	7.00E-06	<b>0.00E-00</b>	<b>0.00E-00</b>	5.1240E+01	<b>0.00E-00</b>	6.60E -04
g11	Best	<b>0.7499</b>	<b>0.7499</b>	<b>0.7499</b>	<b>0.7499</b>	<b>0.7499</b>	<b>0.7499</b>
	Avg	<b>0.7499</b>	<b>0.7499</b>	<b>0.7499</b>	<b>0.7499</b>	<b>0.7499</b>	<b>0.7499</b>
	Std	<b>0.00E-00</b>	<b>0.00E-00</b>	<b>0.00E-00</b>	<b>0.00E-00</b>	<b>0.00E-00</b>	<b>0.00E-00</b>
g12	Best	<b>-1.0000</b>	<b>-1.0000</b>	<b>-1.0000</b>	<b>-1.0000</b>	<b>-1.0000</b>	<b>-1.0000</b>
	Avg	<b>-1.0000</b>	<b>-1.0000</b>	<b>-1.0000</b>	<b>-1.0000</b>	<b>-1.0000</b>	<b>-1.0000</b>
	Std	<b>0.00E-00</b>	<b>0.00E-00</b>	<b>0.00E-00</b>	<b>0.00E-00</b>	<b>0.00E-00</b>	<b>0.00E-00</b>
g13	Best	<b>0.053942</b>	<b>0.053942</b>	<b>0.053942</b>	<b>0.053942</b>	<b>0.053942</b>	<b>0.053942</b>
	Avg	<b>0.053942</b>	<b>0.053942</b>	<b>0.053942</b>	<b>0.053942</b>	<b>0.053942</b>	<b>0.053942</b>
	Std	<b>0.053942</b>	<b>0.053942</b>	<b>0.053942</b>	<b>0.00E-00</b>	<b>0.00E-00</b>	<b>0.00E-00</b>
g14	Best	<b>-47.764888</b>	<b>-47.764888</b>	<b>-47.764888</b>	-47.76479	<b>-47.764887</b>	<b>-47.7649</b>
	Avg	<b>-47.764888</b>	<b>-47.764888</b>	<b>-47.764888</b>	-47.76479	-47.764874	-47.7648
	Std	<b>0.00E-00</b>	<b>0.00E-00</b>	<b>0.00E-00</b>	1.00E-04	1.400E-05	2.72E-05
g15	Best	<b>961.7150</b>	<b>961.71502</b>	<b>961.71502</b>	<b>961.71502</b>	<b>961.71502</b>	<b>961.71502</b>
	Avg	<b>961.7150</b>	<b>961.71502</b>	<b>961.71502</b>	<b>961.71502</b>	<b>961.71502</b>	<b>961.71502</b>
	Std	<b>0.00E-00</b>	<b>0.00E-00</b>	<b>0.00E-00</b>	<b>0.00E-00</b>	<b>0.00E-00</b>	<b>0.00E-00</b>
g16	Best	<b>-1.905155</b>	<b>-1.905155</b>	<b>-1.905155</b>	<b>-1.905155</b>	<b>-1.905155</b>	<b>-1.905155</b>
	Avg	<b>-1.905155</b>	<b>-1.905155</b>	<b>-1.905155</b>	<b>-1.905155</b>	<b>-1.905155</b>	<b>-1.905155</b>
	Std	<b>0.00E-00</b>	<b>0.00E-00</b>	<b>0.00E-00</b>	<b>0.00E-00</b>	<b>0.00E-00</b>	<b>0.00E-00</b>
g17	Best	<b>8853.5397</b>	<b>8853.5397</b>	<b>8853.5397</b>	8853.5398	<b>8853.5397</b>	<b>8853.5397</b>
	Avg	<b>8853.5397</b>	<b>8853.5397</b>	<b>8853.5397</b>	8888.4876	<b>8853.5397</b>	<b>8853.5397</b>
	Std	<b>0.00E-00</b>	<b>0.00E-00</b>	<b>0.00E-00</b>	29.0347	<b>0.00E-00</b>	2.13E -08

g18	Best	<b>-0.866025</b>	<b>-0.866025</b>	<b>-0.866025</b>	<b>-0.866025</b>	<b>-0.866025</b>	<b>-0.866025</b>
	Avg	<b>-0.866025</b>	<b>-0.866025</b>	<b>-0.866025</b>	-0.866025	<b>-0.866025</b>	<b>-0.866025</b>
	Std	<b>0.00E-00</b>	<b>0.00E-00</b>	<b>0.00E-00</b>	0000	<b>0.00E-00</b>	<b>0.00E-00</b>
g19	Best	<b>32.655593</b>	<b>32.655593</b>	<b>32.655593</b>	<b>32.655593</b>	32.655693	32.6591
	Avg	32.656792	32.655688	<b>32.655593</b>	<b>32.655593</b>	33.34125	32.6623
	Std	4.935E-03	2.55E-04	<b>0.00E-00</b>	<b>0.00E-00</b>	8.475E-01	3.4E -03
g21	Best	<b>193.72451</b>	<b>193.72451</b>	<b>193.72451</b>	196.63301	<b>193.72451</b>	193.7246
	Avg	<b>193.72451</b>	<b>193.72451</b>	<b>193.72451</b>	196.51581	<b>193.72451</b>	193.7438
	Std	<b>0.00E-00</b>	<b>0.00E-00</b>	<b>0.00E-00</b>	2.3565E+00	<b>0.00E-00</b>	1.65E-02
g23	Best	<b>-400.0551</b>	<b>-400.0551</b>	-400.0527	-399.7624	<b>-400.0551</b>	-398.9731
	Avg	-399.590570	<b>-400.0551</b>	-400.023589	-394.7627	<b>-400.0551</b>	-373.2178
	Std	2.3227E+00	<b>0.00E-00</b>	6.3463E-02	3.8656E+00	<b>0.00E-00</b>	3.37E+01
g24	Best	<b>-5.508013</b>	<b>-5.508013</b>	<b>-5.508013</b>	<b>-5.508013</b>	<b>-5.508013</b>	<b>-5.508013</b>
	Avg	<b>-5.508013</b>	<b>-5.508013</b>	<b>-5.508013</b>	<b>-5.508013</b>	<b>-5.508013</b>	<b>-5.508013</b>
	Std	<b>0.00E-00</b>	<b>0.00E-00</b>	<b>0.00E-00</b>	<b>0.00E-00</b>	<b>0.00E-00</b>	<b>0.00E-00</b>

**Table 2 Computational Results of EDE-VNS, GA-MPC, APF-GA, MDE, ECHT-EP2  
For CEC 2006 Test Problem**

In additional to these comparisons, so as to demonstrate the effect of ensemble of mutation strategies employed in VNS loops, the code was also run for 500,000 function evaluations for each mutation strategy alone. When each mutation strategy was applied one by one and the code was run again, it was observed that each of them yielded at least one infeasible solution for benchmark problems. Table 3 given below provides the information about feasibility rate of each benchmark problem as employing each mutation strategy alone and ensemble of mutation strategies.

**Table 3 Feasibility Rates of Benchmark Problems for EDE and Each Mutation Strategy**

Problem	D	EDE	Strategy 1	Strategy 2	Strategy 3	Strategy 4	Strategy 5
g1	13	1.00	1.00	1.00	1.00	1.00	1.00
g2	20	1.00	1.00	1.00	1.00	1.00	1.00
g3	10	1.00	1.00	1.00	1.00	1.00	0.56
g4	5	1.00	1.00	1.00	1.00	1.00	1.00
g5	4	1.00	1.00	1.00	1.00	1.00	0.00
g6	2	1.00	1.00	1.00	1.00	1.00	0.00
g7	10	1.00	1.00	1.00	1.00	1.00	0.00
g8	2	1.00	1.00	1.00	1.00	1.00	1.00
g9	7	1.00	1.00	1.00	1.00	1.00	1.00
g10	8	1.00	1.00	1.00	1.00	1.00	0.32
g11	2	1.00	1.00	1.00	1.00	1.00	1.00
g12	3	1.00	0.00	0.00	1.00	1.00	1.00
g13	5	1.00	0.00	0.00	1.00	1.00	0.00
g14	10	1.00	1.00	1.00	1.00	1.00	0.00
g15	3	1.00	1.00	1.00	1.00	1.00	0.00
g16	5	1.00	1.00	1.00	1.00	1.00	1.00
g17	6	1.00	1.00	1.00	1.00	1.00	0.00
g18	9	1.00	1.00	1.00	1.00	1.00	1.00
g19	15	1.00	1.00	1.00	1.00	1.00	1.00
g21	7	1.00	0.64	0.84	0.60	0.60	0.00
g23	9	1.00	0.72	0.68	0.28	0.84	0.00
g24	2	1.00	1.00	1.00	1.00	1.00	1.00

As an interpretation to these feasibility rates in table, it is strictly clear that each mutation strategy has failed to find %100 feasible solutions for all benchmark problems, while ensemble of mutation strategies could provide %100 feasibility rate for all benchmark problems. This situation has shown the power of ensemble of mutation strategies in finding feasible solutions for all benchmark problems.

## 4 APPLICATIONS ON MULTIDIMENSIONAL KNAPSACK PROBLEM

Chu & Beasley (1998) stated in their article that the multidimensional knapsack problem (MKP) is a popular NP-hard and combinatorial optimization problem. The objective of the problem is to maximize the total profit of the selected given item by satisfying all resource constraints. Practical applications of *MKP* arise in a variety of problems such as capital budgeting (Chu & Beasley, 1998), cargo loading (Shih 1979), resource allocating (Gavish & Pirkul 1982), cutting stock (Gilmore & Gomory 1966) etc. Therefore, it is important in the development of effective and efficient algorithms for solving MKPs. The mathematical formulation of MKP can be given as follows:

$$\max z = \sum_{j=1}^n c_j * y_j \quad (46)$$

subjected to:

$$\sum_{j=1}^n a_{ij} * y_j \leq b_i \quad i = 1, 2, \dots, m \quad (47)$$

$$y_j \in \{0, 1\}, \quad j = 1, 2, \dots, n$$

In the formulation given above,  $n$  denotes the number of items and  $m$  denotes the number of knapsack constraints with the capacity  $b_i$  ( $i = 1, 2, 3, \dots, m$ ). Each item  $j$  requires  $a_{ij}$  units of resource consumption in the  $i$ th knapsack and returns  $c_j$  units of profit on incorporation. All entries must be nonnegative.

From the viewpoint of the computation, several proposed algorithms in the literature can be grouped into two main classes: Exact algorithms and heuristic/meta-heuristic algorithms. There have been some exact algorithms applied in order to deal with MKPs in early studies, such as the branch and bound algorithm used by Shih (1979) and dynamic programming used by (DP) Toth (1980). Even though these exact algorithms can yield optimum solutions in solving small-scale problems, they perform badly when the scales of the problem come to be large owing to the NP-hardness property of MKPs. These algorithms cannot provide an optimal solution because of not only high space requirements of the problem but also the limited

computational capacity. In order to remove these limitations, the rule based *B&B* algorithm was proposed by Gavish & Pirkul (1985) so as to reduce the size of the problem and attain better results solutions than previously proposed techniques. In addition to this, Plateau & Elkihel (1985) have introduced *B&B* and *DP* based hybridization method in their article. Lately, an approximated *DP* based approach and a hybridization with a branch and cut procedure were proposed by Bertsimas & Demir (2002) and Boyer et al. (2009). Moreover, exact techniques for solving MKP also include the Lagrangian methods, reduction schemes, special enumeration and surrogate relaxation techniques. Since the search space grows exponentially as the problem size increases, these exact algorithms are generally not useful for solving MKPs. Therefore, some heuristics and meta-heuristic algorithms were introduced to further improve solutions. Simulated annealing is one of the earliest method presented by Drexel (1988) concerning *MKP*. Glover & Kochenberger have proposed tabu search methods and Hanafi & Freville (1998) have further developed this technique to be able to solve all the available public instances in the article of Freville (2004). In the article of Chu & Beasley (1998), a genetic algorithm with large correlated instances was introduced. Their several results were improved by using tabu search algorithm proposed by Vasquez & Hao (2001). Taking the new GA operators and fitness landscapes analysis into consideration, *CBGA* algorithms were used in series of studies as proposed in articles Raidl (1998), Gottlieb (2000), Gottlieb (2001), Tavares et al. (2006), Tavares et al. (2008). More recently, estimation distribution algorithms *EDAs* are used in the articles of Kong et al. (2008), Wang et al. (2012), Martins et al. (2013), Martins & Delbem (2013). Moreover, some other meta-heuristic algorithms were developed for MKP such as particle swarm optimization (*PSO*) in the articles of Behesthi et al. (2012), Chen et al. (2010), Chih et al. (2014) and Azad et al. (2014), differential evolution in Wang et al (2012) and other heuristic algorithms in Angelelli et al. (2012).

#### **4.1 Solution Methodology for MKP**

In this thesis, an ensemble differential evolution algorithm with a variable neighbourhood search in order to solve multidimensional knapsack problem is applied. Unlike the studies that use check and repair operators, some sophisticated constraint handling methods which are further explained in constraint handling methods are to enrich the diversity of the population by taking advantages of infeasible solution within a predetermined threshold. In order to generate the trial

population, *VNS* algorithm with different mutation strategies is proposed. Since this proposed algorithm is employed in a continuous domain, it is required to change these real-values to binary 0,1 values by using *S*-shaped and *V*-shaped transfer functions that will be explained in detail in following sections. So as to be able to develop the solution, the differential evolution algorithm with a variable neighbourhood search is combined with a binary swap local search algorithm as well.

## 4.2 Solution Representation

Since traditional DE algorithm is employed for a continuous space, a unique *DE* with a multi-chromosome solution representation is proposed. This representation is required because of the fact that *MKP* is a binary optimization problem. First of all, *Ensemble DE\_VNS* algorithm works on a continuous domain within the range [0,1]. However, each dimension is converted to binary 0 – 1 values by means of *S*-shaped and *V*-shaped transfer functions. Since *EDE\_VNS* algorithm is used, it is probable to use different transfer functions for each loops of *VNS* algorithm.

Firstly, most well-known *S*-shaped transfer function called sigmoid function inspired from the article of Wang et al. (2013) is used to convert each dimension to 0 – 1. Sigmoid function uses following probability model to provide conversion:

$$\pi_{ij} = \begin{cases} 1 & \text{if } r < 1/(1 + e^{-w(u_{ij}^g - 0.5)}) \\ 0 & \text{otherwise} \end{cases} \quad (48)$$

where  $r$  is a uniform random number in [0,1];  $u_{ij}$  is the real value in [0,1] of the trial individual; and  $w$  is the coefficient of the sigmoid function. Since the selection of  $w$  is quite important, it is tried to take the coefficient  $w$  as 10, 20, 30, 40 and 50 by means of trial-and-error method. In this thesis, two different sigmoid functions are used for converting real values to binary variables. Therefore,  $w$  is taken as 30 like suggested in Wang et al. (2013). In addition to this suggested strategy,  $w$  is also taken as 50 as a second strategy.

On the other hand, such *V*-shaped transfer functions using diverse mathematical equations as *tan* and *arctan* are also employed in *VNS* algorithm. These functions use following probability model in order to provide the conversion:



- $T(x) = |\tanh(x)|$

$$\pi_{ij} = \begin{cases} 1 & \text{if } r < |\tanh(u_{ij})| \\ 0 & \text{otherwise} \end{cases} \quad (49)$$

- $T(x) = |(x)/\sqrt{1+x^2}|$

$$\pi_{ij} = \begin{cases} 1 & \text{if } r < \left| (u_{ij}) / \sqrt{1 + (u_{ij}^g)^2} \right| \\ 0 & \text{otherwise} \end{cases} \quad (50)$$

- $T(x) = \left| \frac{2}{\pi} \arctan\left(\frac{\pi}{2} x\right) \right|$

$$\pi_{ij} = \begin{cases} 1 & \text{if } r < \left| \frac{2}{\pi} \arctan\left(\frac{\pi}{2} u_{ij}^g\right) \right| \\ 0 & \text{otherwise} \end{cases} \quad (51)$$

where  $r$  is a uniform random number in  $[0,1]$ ;  $u_{ij}$  is the real value in  $[0,1]$  of the trial individual.

For conversion to binary values, two  $S$ -shaped and three  $V$ -shaped transfer functions were employed. In summary, when a trial individual is generated by VNS algorithm, binary value of each dimension is determined by one of the equations (48), (49), (50) and (51) given above so as to calculate the fitness functions. Figure below illustrates the multi-chromosome solution representation.

j	1	2	3	4	5
$x_{ij}$	0.2	0.9	0.1	0.8	0.7
$\pi_{ij}$	0	1	0	1	1

*Figure 4.11. Solution Representation*

### 4.3 Families of Transfer Functions

A transfer function is used for defining the probability of changing a position vector's element from 0 to 1 and vice versa. In literature, a transfer function is responsible for mapping a continuous search space to a discrete search space. Therefore a transfer function forces particles to move in a binary space. According to Rashedi et al. (2009), there exist some concepts that should be considered for selecting a transfer function in order to map dimension values of a chromosome to probability values to probability values as follows:

- The range of a transfer function should be bounded in the interval  $[0,1]$ , since they represent the probability of a particle that changes its position.
- A transfer function should be able to provide a high probability of changing the position for a large absolute dimension value. Dimensions with large absolute values for their dimension values are supposedly far from the best solution.
- A transfer function should also provide a small probability of change in the position for small absolute value of dimensions.
- The return value of a transfer function should increase while the value of a dimension rises. Dimensions with a higher probability changing their position must be moving away from the best solution in order to return their previous positions.
- The return value of a transfer function should decrease as the dimension value reduces.

These concepts guarantee that a transfer function can map the process of search in continuous search space to a binary search space as preserving similar concepts of the search for a specific evolutionary algorithm.

**Table 4 Transfer Functions**

No	Transfer Functions
1	$S(x) = \frac{1}{1 + e^{-2x}}$
2 [7]	$S(x) = \frac{1}{1 + e^{-x}}$
3	$S(x) = \frac{1}{1 + e^{\frac{-x}{2}}}$
4	$S(x) = \frac{1}{1 + e^{\frac{-x}{3}}}$
5	$S(x) = \left  \operatorname{erf}\left(\frac{\sqrt{\pi}}{2}x\right) \right  = \left  \frac{\sqrt{2}}{\pi} \int_0^{\frac{\sqrt{\pi}}{2}x} e^{-t^2} dt \right $
6 [14]	$S(x) =  \tanh(x) $
7	$S(x) = \left  \frac{x}{\sqrt{1+x^2}} \right $
8	$S(x) = \left  \frac{2}{\pi} \arctan\left(\frac{\pi}{2}x\right) \right $

In the article of Mirjalili & Lewis (2013), the properties of transfer functions which are given in table 4 are summarized. In this article, authors have used second transfer function (S2) in table for the conventional *BPSO*. As shown in table, first four transfer functions are S-shaped and the rest are V-shaped ones. These transfer functions manipulate the coefficient of  $x$ . If it is necessary to give information about transfer functions in detail, it can be said that S1 sharply increases and reaches its saturation as the dimension value increases much higher than S2, whereas the saturations of S3 and S4 start later than S2. It must be kept on the mind that when the slope of these transfer functions increases, the probability of changing the values of the position vector rises, therefore it can be concluded that S1 returns the highest probability among them for the same dimension value, while S4 yields the lowest one. These transfer functions have been selected with different slopes compared to one used in the article of Mirjalili & Lewis (2013), in order that the efficiency of these characteristics on improving the performance of their proposed algorithm *BPSO* can be investigated. Moreover, to compare this algorithm with the proposed algorithm of this thesis, S3 is used in one of the VNS loops of EDE-VNS algorithm instead of S2. In view of the shapes of the curves of S1, S2, S3 and S4, they are named as S-shaped transfer functions and their group name is given as “S-shaped” family of transfer functions as well.

On the other hand, Rashedi et al. (2009) has used another type of transfer functions that could be used for their different position updating rules. The transfer

function used by Rashedi et al. (2009) is presented in Table 4 as S6. Since these functions are pretty different from the S-shaped family, they are possible to need new position updating rules. These functions are named V-shaped transfer functions and their group name is given as “V-shaped” family of transfer functions. In order to compare this article with this thesis, not only S6 but also S7 and S8 given in the table above are used in the different loops of proposed EDE-VNS algorithm. These new S7 and S8 transfer functions uses the diverse mathematical equations. To give more information about this V-shaped transfer functions, it can be said that S5 starts with lower absolute values of dimensions compared to S6. This situation makes S5 be able to provide higher probability of switching dimension values of a chromosome than S6 for the same value. In contrast, the S7 and S8 transfer functions’ saturations begin after S5 and S6 providing less probability of change for the same dimension values. In the article of Mirjalili & Lewis (2013), it is possible to see a study about the efficiency of these families.

#### 4.4 Binary Swap Local Search

In this thesis, the neighbourhood search is based on a simple swap neighbourhood. It should be noted that this local search is applied to the best solution  $\pi_b^g$  at each generation  $g$ . The binary swap (*bSWAP*) local search includes two steps:

- *Generate two random integers,  $u$  and  $v$ , in the range  $[1, n]$*
- *If  $\pi_{b,u}^g = \pi_{b,v}^g$  then  $\pi_{b,u}^g = (\pi_{b,u}^g + 1) \bmod 2$*   
*else*  
 *$\pi_{b,u}^g = (\pi_{b,u}^g + 1) \bmod 2$  and  $\pi_{b,v}^g = (\pi_{b,v}^g + 1) \bmod 2$*

The outline of the local search is given in Fig. 4.12. below:

```

produce LocalSearch( $\pi_b^g$ )
 $\pi_1 =$  perturbation ( $\pi_b^g$ )
for ( $l = 1$  to  $n$ )
{
     $\pi_2 =$  bSWAP( $\pi_1$ )
    if ( $f(\pi_2) > f(\pi_1)$ )
    {
         $\pi_1 = \pi_2$ 
        if ( $f(\pi_2) > f(\pi_b^g)$ )
    }
}

```

```

        {
             $\pi_b^g = \pi_2$ 
        } end if
    }end if
}end for
return  $\pi_b^g$ 
endprocedure

```

Figure 4.12. Outline of bSWAP Local Search

In the local search algorithm detailed above,  $\pi_1$  refers to the perturbed best solution up to now  $\pi_b^g$  at each generation  $g$ . It can be also said that, the best so far solution is perturbed by swapping two different items randomly. Then the *bSWAP* operator is applied to  $\pi_1$ . The number of items  $n$  is set as the size of the local search. The full computational procedure of EDE-VNS algorithm is given in Fig. 4.13.

*Procedure EDE\_VNS()*

Set parameters  $g = 0, NP = 50, S_{max} = 5$

Step 1. Establish initial population randomly

$$P^g = \{x_1^g, \dots, x_{NP}^g\} \text{ with } x_i^g = \{x_{i1}^g, \dots, x_{iD}^g\}$$

Step 2. Apply OBL to population and choose the best ones

$$P^g = \{x_1^g, \dots, x_{NP}^g\} \text{ with } x_i^g = \{x_{i1}^g, \dots, x_{iD}^g\}$$

Step 3. Assign a strategy to each individual

$$M_i = \text{rand()} \% M_{max} \text{ for } i = 1, \dots, NP$$

Step 4. Evaluate population and find  $x_{best}^g$

$$f(P^g) = \{f(x_1^g), \dots, f(x_{NP}^g)\}$$

Step 5. Determine  $e(0)$

Step 6. Repeat the following for each individual  $x_i^g$

- obtain  $u_i^g = \text{VNS}_i(x_i^g)$  through NFT
- calculate  $\pi_{ij}$  using transfer functions.
- apply injection to trial individual
- obtain  $x_i^{g+1}$  through  $e - \text{constraint method}$
- if  $(f(x_i^g) \leq f(x_{best}^g))$ ,  $x_{best}^g = x_i^g$  through SF

- Update  $e(g) = e(0) \left(1 - \frac{g}{g_c}\right)$
- Step 7. If the stopping criterion is not met, go to Step 6,  
else stop and return  $\pi_{best}$

**Figure 4.13 Outline of EDE-VNS Algorithm for Knapsack Problem**

## 4.5 Computational Results

The proposed *EDE\_VNS* algorithm was coded in Visual C++ and run on an Intel (R) Core (TM) Duo 2.4 GHz PC with 2GB memory. According the parameters of the *EDE\_VNS* algorithm, the population size was fixed at  $NP = 100$ . As a termination criterion, the *EDE\_VNS* algorithm was run for  $g_{max} = 20000$  generation as in the article of Chih (2014). Next parameters were set in the  $\varepsilon$ -constraint handling method:  $\theta = 0.95 \times NP$   $g_c = 0.4 \times g_{max}$  and  $cp = 2$ . The severity parameter  $\alpha$  and the positive constant  $\gamma$  of the *NFT* penalty function are taken as 2.0 and 0.004, respectively. The  $NFT_0$  is selected as  $1e-8$ , because the probability that algorithm results in an infeasible solution is higher when the  $NFT_0$  threshold distance is larger. As in the article of Chih (2014), 100 runs are carried out for each instance.

In this thesis, the performance of the *EDE\_VNS* algorithm is compared to penalty method based *PSO* and *SACRO* (self-adaptive check and repair) based *PSO* algorithms in Chih (2014) a novel binary fruit fly algorithm (*bFOA2*) in Wang et al. (2013) and the hybrid *EDA*-based algorithm (*HEDA2*) in Wang et al. (2012).

Table 5 provides the information about number of items, number of constraints and best known solutions for Sento and Weing instances.

**Table 5 Details of Sento and Weing Instances**

Instance	n	m	Best Known
SENTO1	60	30	7772
SENTO2	60	30	8722
WEING1	28	2	141278
WEING2	28	2	130883
WEING3	28	2	95677
WEING4	28	2	119337

WEING7	105	2	1095445
WEING8	105	2	624319

Table 6 summarizes the computational results of the *EDE\_VNS* algorithm. As seen in Table 6, penalty function based *BPSO\_TVAC* and *CBPSO\_TVAC* algorithms were not competitive to *EDE\_VNS* algorithm. In addition, *EDE\_VNS* algorithm is able to find the best known solution in each run with zero standard deviation for 7 out of 10 instances whereas *SACRO* based PSO algorithms were only able to do the same thing for 6 problem instances. For the last 2 instances, *EDE\_VNS* algorithm yielded better *Best*, *Avg*, and *Std* values than all the algorithms compared.

**Table 6 Computational Results for Sento and Weing Instances**

P		Penalty function based algorithms			SACRO	
		EDE_VNS	BPSO-TVAC	CBPSO-TVAC	BPSO-TVAC	CBPSO-TVAC
Sento1	Best	<b>7772</b>	<b>7772</b>	<b>7772</b>	<b>7772</b>	<b>7772</b>
	Avg.	<b>7772</b>	7763.26	7635.72	7769.48	7769.48
	Std.	0	11.525	357.784	5.406	5.406
Sento2	Best	<b>8722</b>	<b>8722</b>	<b>8722</b>	<b>8722</b>	<b>8722</b>
	Avg.	<b>8722</b>	8712.58	8668.47	<b>8722</b>	<b>8722</b>
	Std.	0	7.038	101.033	0	0
Weing1	Best	<b>141,278</b>	<b>141,278</b>	<b>141,278</b>	<b>141,278</b>	<b>141,278</b>
	Avg.	<b>141,278</b>	141,278	141,226.8	<b>141,278</b>	<b>141,278</b>
	Std.	0	0	281.978	0	0
Weing2	Best	<b>130,883</b>	<b>130,883</b>	<b>130,883</b>	<b>130,883</b>	<b>130,883</b>
	Avg.	<b>130,883</b>	<b>130,883</b>	130,759.8	<b>130,883</b>	<b>130,883</b>
	Std.	0	0	545.503	0	0
Weing3	Best	<b>95,677</b>	<b>95,677</b>	<b>95,677</b>	<b>95,677</b>	<b>95,677</b>
	Avg.	95,673.5	95,670.58	95,503.93	95,676.39	95,676.39
	Std.	12.82162	25.528	672.423	6.1	6.1
Weing4	Best	<b>119,337</b>	<b>119,337</b>	<b>119,337</b>	<b>119,337</b>	<b>119,337</b>
	Avg.	<b>119,337</b>	<b>119,337</b>	119,294.2	<b>119,337</b>	<b>119,337</b>
	Std.	0	0	378.583	0	0
Weing5	Best	<b>98,796</b>	<b>98,796</b>	<b>98,796</b>	<b>98,796</b>	<b>98,796</b>
	Avg.	<b>98,796</b>	<b>98,796</b>	98,710.4	<b>98,796</b>	<b>98,796</b>
	Std.	0	0	572.82	0	0
Weing6	Best	<b>130,623</b>	<b>130,623</b>	<b>130,623</b>	<b>130,623</b>	<b>130,623</b>
	Avg.	<b>130,623</b>	130,611.3	130,531.3	<b>130,623</b>	<b>130,623</b>
	Std.	0	66.864	343.456	0	0
Weing7	Best	<b>1,095,382</b>	<b>1,095,382</b>	<b>1,095,382</b>	<b>1,095,382</b>	<b>1,095,445</b>
	Avg.	1,095,391	1,095,164	1,084,172	1,094,349	1,094,410
	Std.	21.97031	383.743	30,020.95	2697.39	1795.58

Weing8	Best	<b>624,319</b>	<b>624,319</b>	<b>624,319</b>	<b>624,319</b>	<b>624,319</b>
	Avg.	622,501.40	622,446.6	597,190.6	622,079.9	622,032.5
	Std.	1705.16	2.000.923	75,169.93	1462.09	1454.99

Table 7 provides the information about number of items, number of constraints and best known solutions for Weish instances.

**Table 7 Details of Weish Instances**

Instance	n	m	Best Known
WEISH1	30	5	4554
WEISH2	30	5	4536
WEISH3	30	5	4115
WEISH4	30	5	4561
WEISH5	30	5	4514
WEISH6	40	5	5557
WEISH7	40	5	5567
WEISH8	40	5	5605
WEISH9	40	5	5246
WEISH10	50	5	6339
WEISH11	50	5	5643
WEISH12	50	5	6339
WEISH13	50	5	6159
WEISH14	60	5	6954
WEISH15	60	5	7486
WEISH16	60	5	7289
WEISH17	60	5	8633
WEISH18	70	5	9580
WEISH19	70	5	7698
WEISH20	70	5	9450
WEISH21	70	5	9074
WEISH22	80	5	8947
WEISH23	80	5	8344
WEISH24	80	5	10220
WEISH25	80	5	9939
WEISH26	90	5	9584
WEISH27	90	5	9819
WEISH28	90	5	9492
WEISH29	90	5	9410
WEISH30	90	5	11191

Table 8 gives the information about computational results for *WEISH* instances. It can be concluded that penalty based functions *BPSO\_TVAC* and *CBPSO\_TVAC* are not able to be competitive to proposed algorithm *EDE – VNS*,



whereas *SACRO*-based algorithms yields much better results than penalty based functions. Moreover, It is necessary to highlight that *EDE – VNS* is able to find the best known solution in each run with zero standard deviation for 29 out of 30 instances and only instance WEISH26 is found with 8.089 standard deviation, while *SACRO*-based *CBPSO\_TVAC* and *BPSO\_TVAC* algorithms could find the best known solutions in each run with zero standard deviation for 23 and 25 instances, respectively. Therefore, it can be said that proposed algorithm is competitive to the best performing algorithms in the literature.

**Table 8 Computational results for Weish Instances**

P		Penalty function			SACRO	
		EDE_VNS	BPSO-TVAC	CBPSO-TVAC	BPSO-TVAC	CBPSO-TVAC
Weish1	Best	<b>4554</b>	<b>4554</b>	<b>4554</b>	<b>4554</b>	<b>4554</b>
	Avg.	<b>4554</b>	<b>4554</b>	4548.55	<b>4554</b>	<b>4554</b>
	Std.	0	0	32.808	0	0
Weish2	Best	<b>4536</b>	<b>4536</b>	<b>4536</b>	<b>4536</b>	<b>4536</b>
	Avg.	<b>4536</b>	4534.2	4531.88	<b>4536</b>	<b>4536</b>
	Std.	0	2.412	23.117	0	0
Weish3	Best	<b>4115</b>	<b>4115</b>	<b>4115</b>	<b>4115</b>	<b>4115</b>
	Avg.	<b>4115</b>	4114.37	4105.79	<b>4115</b>	<b>4115</b>
	Std.	0	6.3	52.697	0	0
Weish4	Best	<b>4561</b>	<b>4561</b>	<b>4561</b>	<b>4561</b>	<b>4561</b>
	Avg.	<b>4561</b>	<b>4561</b>	4552.41	<b>4561</b>	<b>4561</b>
	Std.	0	0	85.9	0	0
Weish5	Best	<b>4514</b>	<b>4514</b>	<b>4514</b>	<b>4514</b>	<b>4514</b>
	Avg.	<b>4514</b>	<b>4514</b>	4505.89	<b>4514</b>	<b>4514</b>
	Std.	0	0	74.451	0	0
Weish6	Best	<b>5557</b>	<b>5557</b>	<b>5557</b>	<b>5557</b>	<b>5557</b>
	Avg.	<b>5557</b>	5550.32	5533.79	5553.75	5553.88
	Std.	0	8.196	79.282	5.657	5.58
Weish7	Best	<b>5567</b>	<b>5567</b>	<b>5567</b>	<b>5567</b>	<b>5567</b>
	Avg.	<b>5567</b>	5566.3	5547.83	<b>5567</b>	<b>5567</b>
	Std.	0	3.448	71.946	0	0
Weish8	Best	<b>5605</b>	<b>5605</b>	<b>5605</b>	<b>5605</b>	<b>5605</b>
	Avg.	<b>5605</b>	5064.58	5596.16	5604.92	<b>5605</b>
	Std.	0	0.818	42.809	0.394	0
Weish9	Best	<b>5246</b>	<b>5246</b>	<b>5246</b>	<b>5246</b>	<b>5246</b>
	Avg.	<b>5246</b>	<b>5246</b>	5232.99	<b>5246</b>	<b>5246</b>
	Std.	0	0	65.701	0	0
Weish10	Best	<b>6339</b>	<b>6339</b>	<b>6339</b>	<b>6339</b>	<b>6339</b>
	Avg.	<b>6339</b>	6337.57	6271.84	<b>6339</b>	<b>6339</b>
	Std.	0	9.56	188.63	0	0

Weish11	Best Avg. Std	<b>5643</b> <b>5643</b> 0	<b>5643</b> 5635.58 25.717	<b>5643</b> 5532.15 403.037	<b>5643</b> <b>5643</b> 0	<b>5643</b> <b>5643</b> 0
Weish12	Best Avg. Std	<b>6339</b> <b>6339</b> 0	<b>6339</b> 6338.71 1.914	<b>6339</b> 6231.5 304.427	<b>6339</b> <b>6339</b> 0	<b>6339</b> <b>6339</b> 0
Weish13	Best Avg. Std	<b>6159</b> <b>6159</b> 0	<b>6159</b> <b>6159</b> 0	<b>6159</b> 6120.38 1725	<b>6159</b> <b>6159</b> 0	<b>6159</b> <b>6159</b> 0
Weish14	Best Avg. Std	<b>6954</b> <b>6954</b> 0	<b>6954</b> 6953.38 4.361	<b>6954</b> 6837.77 364.656	<b>6954</b> <b>6954</b> 0	<b>6954</b> <b>6954</b> 0
Weish15	Best Avg. Std	<b>7486</b> <b>7486</b> 0	<b>7486</b> <b>7486</b> 0	<b>7486</b> 7324.55 554.352	<b>7486</b> <b>7486</b> 0	<b>7486</b> <b>7486</b> 0
Weish16	Best Avg. Std	<b>7289</b> <b>7289</b> 0	<b>7289</b> 7287.84 1.709	<b>7289</b> 7145.71 367.298	<b>7289</b> 7288.7 0.46	<b>7289</b> 7288.71 0.456
Weish17	Best Avg. Std	<b>8633</b> <b>8633</b> 0	<b>8633</b> <b>8633</b> 0	<b>8633</b> 8547.71 227.164	<b>8633</b> <b>8633</b> 0	<b>8633</b> <b>8633</b> 0
Weish18	Best Avg. Std	<b>9580</b> <b>9580</b> 0	<b>9580</b> 9577.21 5.255	<b>9580</b> 9480.86 275.537	<b>9580</b> 9578.46 2.914	<b>9580</b> 9579.02 2.441
Weish19	Best Avg. Std	<b>7698</b> <b>7698</b> 0	<b>7698</b> 7693.1 7.13	<b>7698</b> 7528.55 489.373	<b>7698</b> <b>7698</b> 0	<b>7698</b> <b>7698</b> 0
Weish20	Best Avg. Std	<b>9450</b> <b>9450</b> 0	<b>9450</b> 9446.22 7.527	<b>9450</b> 9332.11 410.738	<b>9450</b> <b>9450</b> 0	<b>9450</b> <b>9450</b> 0
Weish21	Best Avg. Std	<b>9074</b> <b>9074</b> 0	<b>9074</b> 9067.94 10.41	<b>9074</b> 8948.22 378.377	<b>9074</b> <b>9074</b> 0	<b>9074</b> <b>9074</b> 0
Weish22	Best Avg. Std	<b>8947</b> <b>8947</b> 0	<b>8947</b> 8931.88 6.632	<b>8947</b> 8774.2 486.714	<b>8947</b> 8936.92 8.979	<b>8947</b> 8939.8 8.862
Weish23	Best Avg. Std	<b>8344</b> <b>8344</b> 0	<b>8344</b> 8342.89 5.111	<b>8344</b> 8165 437.232	<b>8344</b> <b>8344</b> 0	<b>8344</b> <b>8344</b> 0
Weish24	Best Avg. Std	<b>10,22</b> <b>10,22</b> 0	<b>10,22</b> 10,216.96 6.439	<b>10,22</b> 10,106.28 295.796	<b>10,22</b> 10,219.7 1.193	<b>10,22</b> 10,219.32 2.308
Weish25	Best Avg. Std	<b>9939</b> <b>9939</b> 0	<b>9939</b> 9934.46 7.086	<b>9939</b> 9826.57 361.88	<b>9939</b> <b>9939</b> 0	<b>9939</b> <b>9939</b> 0
Weish26	Best Avg. Std	<b>9584</b> 9581.66 8.089	<b>9584</b> 9527.56 12.806	<b>9584</b> 9313.87 710.769	<b>9584</b> <b>9584</b> 0	<b>9584</b> <b>9584</b> 0

Weish27	Best	<b>9819</b>	<b>9819</b>	<b>9819</b>	<b>9819</b>	<b>9819</b>
	Avg.	<b>9819</b>	9818.61	9607.54	<b>9819</b>	<b>9819</b>
	Std	0	3.9	640.43	0	0
Weish28	Best	<b>9492</b>	<b>9492</b>	<b>9492</b>	<b>9492</b>	<b>9492</b>
	Avg.	<b>9492</b>	9489.01	9123.26	<b>9492</b>	<b>9492</b>
	Std	0	7.774	887.332	0	0
Weish29	Best	<b>9410</b>	<b>9410</b>	<b>9410</b>	<b>9410</b>	<b>9410</b>
	Avg.	<b>9410</b>	9406.81	9025.5	<b>9410</b>	<b>9410</b>
	Std	0	10.094	854.501	0	0
Weish30	Best	<b>11,191</b>	<b>11,191</b>	<b>11,191</b>	<b>11,191</b>	<b>11,191</b>
	Avg.	<b>11,191</b>	11,190.48	10,987.21	11,190.12	11,189.96
	Std	0	1.352	491.815	1.665	1.763

Table 9 provides the information about number of items, number of constraints and best known solutions for some of HP and PB instances.

**Table 9 Details of HP and PB Instances**

Instance	n	m	Best Known
Hp1	28	4	3418
Hp2	35	4	3186
Pb5	20	10	2136
Pb6	40	30	776
Pb7	37	30	1035

Table 10 details the computational results for the *Hp* and *Pb* instances. From table, it can be said that penalty function based *BPSO\_TVAC* and *CBPSO\_TVAC* are not competitive to proposed algorithm *EDE – VNS*. However, it is also worth emphasizing that *EDE – VNS* is able to find best known solution in each run with zero standard deviation for all instances so, *EDE – VNS* is best one among them. On the other hand, *SACRO*-based *CBPSO\_TVAC* and *BPSO\_TVAC* algorithms could find the best known solutions in each run with zero standard deviation for 5 and 5 instances, respectively.

**Table 10 Computational results for HP and PB Instances**

P		Penalty function			SACRO	
		EDE_VNS	BPSO-TVAC	CBPSO-TVAC	BPSO-TVAC	CBPSO-TVAC
Hp1	Best	<b>3418</b>	<b>3418</b>	<b>3418</b>	<b>3418</b>	<b>3418</b>
	Avg.	<b>3418</b>	3406.56	3403.9	3413.38	3414.36
	Std.	0	10.692	13.69	6.616	6.172
Hp2	Best	<b>3186</b>	<b>3186</b>	<b>3186</b>	<b>3186</b>	<b>3186</b>
	Avg.	<b>3186</b>	3179.49	3173.61	3184.74	3185.64
	Std.	0	13.947	21.354	4.615	2.532
Pb1	Best	<b>3090</b>	<b>3090</b>	<b>3090</b>	<b>3090</b>	<b>3090</b>
	Avg.	<b>3090</b>	3081	3079.74	3086.78	3085.95
	Std.	0	9.438	10.521	5.921	6.369
Pb2	Best	<b>3186</b>	<b>3186</b>	<b>3186</b>	<b>3186</b>	<b>3186</b>
	Avg.	<b>3186</b>	3181.5	3171.55	<b>3186</b>	<b>3186</b>
	Std.	0	7.686	18.731	0	0
Pb4	Best	<b>95,168</b>	<b>95,168</b>	<b>95,168</b>	<b>95,168</b>	<b>95,168</b>
	Avg.	<b>95,168</b>	94,939.9	94,863.67	<b>95,168</b>	<b>95,168</b>
	Std.	0	797.051	875.081	0	0
Pb5	Best	<b>2139</b>	<b>2139</b>	<b>2139</b>	<b>2139</b>	<b>2139</b>
	Avg.	<b>2139</b>	2136.28	2135.6	<b>2139</b>	<b>2139</b>
	Std.	0	6.263	6.834	0	0
Pb6	Best	<b>776</b>	<b>776</b>	<b>776</b>	<b>776</b>	<b>776</b>
	Avg.	<b>776</b>	767.3	758.26	<b>776</b>	<b>776</b>
	Std.	0	Ey1.99	40.172	0	0
Pb7	Best	<b>1035</b>	<b>1035</b>	<b>1035</b>	<b>1035</b>	<b>1035</b>
	Avg.	<b>1035</b>	1029.57	1021.95	<b>1035</b>	<b>1035</b>
	Std.	0	5.712	24.247	0	0

According to the analysis above, it can be concluded that the proposed *EDE – VNS* algorithm is superior to both penalty based and *SACRO*-based algorithms.

## 5 CONCLUSIONS & FUTURE WORK

A research about the application of optimization algorithms on one of the most important real-life problem called multidimensional knapsack problem after testing these optimization algorithms on benchmark functions from literature was introduced in this thesis. The underlying reason of using this kind of algorithms is because of the fact that it is not possible to find optimal solutions for NP-hard problems by means of traditional algorithms. Since, multidimensional knapsack problems have real numbers to be found optimally, the main idea of this thesis focused on the term which has been researched and analysed over last decades by engineers named, “Real Parameter Optimization”.

Firstly, 22 benchmark problems from CEC’ 2006 technical report were taken into consideration to solve by using proposed *EDE – VNS* algorithm. Since these benchmark problems are considered as real-life problems, finding optimal solutions for these benchmark problems could demonstrate the applicability of proposed algorithm to real-life problems. *EDE – VNS* algorithm was able to employ multiple mutation strategies in its *VNS* loops to improve the solution quality. In order to take advantage of infeasible solutions, various constrained handling methods (*SF*, *NFT* and  $\epsilon$ -constraint) were handled. Moreover, opposition-based learning algorithm in order to enrich the initial population is also used for improving this proposed heuristic algorithm. A diversification procedure based on the inversion of the target individuals and injection of some good dimensional values from promising areas in the target population by tournament selection with size 2 was presented to provide improved individuals for target population. The computational results showed that the simple *EDE – VNS* algorithm was quite competitive to some of the best performing algorithms from the literature.

Secondly, the proposed algorithm is applied to multidimensional knapsack problem in order to demonstrate the applicability of this algorithm to real-life problems. In this part, some sophisticated constraint handling techniques are utilized to enrich the population diversity within a predetermined threshold instead of making use of the studies employing check and repair operators. So as to generate the trial population, a variable neighbourhood search employing different mutation strategies in each loop is proposed. As the proposed algorithm truly works on a continuous domain, these real-values in chromosomes are converted to 0-1 binary values by

using some S-shaped and V-shaped functions. Moreover, in order to enhance the quality of solution, the differential evolution algorithm with a variable neighbourhood search is combined with a binary swap local search. This proposed algorithm is tested on a benchmark instances in OR-library and computational results are compared to some algorithms called bFOA2 and HEDA2 algorithms from the literature. Computational results show its efficiency in solving benchmark instances and its superiority to other algorithms.

For future work, it is aimed to develop some DE algorithm making use of the idea of neighbourhood change of VNS algorithms for not only unconstrained but also constrained real parameter optimization problems and also apply this algorithm together with some constructive heuristics from literature to solve binary optimization problems in real life in order to demonstrate applicability of evolutionary algorithms.

## REFERENCES

- Angelelli, E., & Mansini, R., 2002.**, The vehicle routing problem with time windows and simultaneous pick-up and delivery, In *Quantitative approaches to distribution logistics and supply chain management* (pp. 249-267). Springer Berlin Heidelberg.
- Azad, M. A. K., Rocha, A. M. A., & Fernandes, E. M., 2014**, “Improved binary artificial fish swarm algorithm for the 0–1 multidimensional knapsack problems,” *Swarm and Evolutionary Computation*, vol. 14, pp. 66-75.
- Babu, B. V., and Jehan, M., 2003**, “Differential evolution for multi-objective optimization,” IEEE, 2003. CEC'03. *The 2003 Congress on* Vol. 4, pp. 2696-2703
- Babu B. V., and Onwubolu G. C., (eds.), 2004**, “New Optimization Techniques in Engineering”, Springer Verlag.
- Beheshti, Z., Shamsuddin, S. M., & Yuhaniz, S. S., 2013**, “Binary accelerated particle swarm algorithm (BAPSA) for discrete optimization problems,” *Journal of Global optimization*, vol.57, no. 2, pp. 549-573.
- Bertsimas, D., and Demir, R., 2002**, “An approximate dynamic programming approach to multidimensional knapsack problems,” *Management Science*, vol.48, no.4, pp. 550-565.
- Boyer, V., Elkihel, M., and El Baz, D., 2009**, Heuristics for the 0–1 multidimensional knapsack problem, *European Journal of Operational Research*, 199(3), pp. 658-664.
- Brest J., Greiner S., Boskovic B., Mernik M., and Zumer V., 2006**, “Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems,” *IEEE Trans. Evol. Comput.*, vol. 10, no. 6, pp. 646–657.
- Chakraborty U K., 2008** (ed.) *Advances in Differential Evolution*. Berlin: Springer.

**Chen, W. N., Zhang, J., Chung, H. S., Zhong, W. L., Wu, W. G., & Shi, Y. H.,** 2010, “A novel set-based particle swarm optimization method for discrete optimization problems,” *Evolutionary Computation, IEEE Transactions*, vol. 14 no. 2, pp. 278-300.

**Chih, M., Lin, C. J., Chern, M. S., & Ou, T. Y.,** 2014,” Particle swarm optimization with time-varying acceleration coefficients for the multidimensional knapsack problem,” *Applied Mathematical Modelling*, vol. 38, no. 4, pp. 1338-1350

**Chih, M.** 2015, “Self-adaptive check and repair operator-based particle swarm optimization for the multidimensional knapsack problem,” *Applied Soft Computing*, vol. 26, no. 378-389.

**Chu, P. C., and Beasley, J. E.,** 1998, “A genetic algorithm for the multidimensional knapsack problem,” *Journal of heuristics*, vol. 4, no. 1, pp. 63-86.

**Coello, C. A. C.,** 2002, “Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art,” *Computer methods in applied mechanics and engineering*, vol. 191, no.11, pp. 1245-1287.

**Corne D., Dorigo M., and Glover F., (eds.),** 1999, “Part Two: Differential Evolution,” *New Ideas in Optimization*, McGraw-Hill, pp. 77-158.

**Das S., Abraham A., Chakraborty U. K., and Konar A.,** 2009, “Differential evolution using a neighbourhood based mutation operator,” *IEEE Trans. Evol. Comput.*, vol. 13, no. 3, pp. 526–553.

**Das S., Konar A., and Chakraborty U.,** 2005, “Improved differential evolution algorithms for handling noisy optimization problems,” in *Proc. IEEE Congr. Evol. Comput.*, vol. 2. 2005, pp. 1691–1698.

**Das S., Suganthan P.N.,** 2011, *Differential Evolution: A Survey of the State-of-the-Art*, *IEEE Trans. Evol. Comput.*, Vol 15, No.1.



**Deb, K.**, 2000, An efficient constraint handling method for genetic algorithms, *Comput. Methods Appl. Mech. Eng.*, 186, pp. 311–338.

**Drexl, A.**, 1988, “A simulated annealing approach to the multiconstraint zero-one knapsack problem,” *Computing*, vol. 40, no. 1, pp. 1-8.

**Elsayed, S. M., Sarker, R. A., & Essam, D. L.**, 2014, “A new genetic algorithm for solving optimization problems,” *Engineering Applications of Artificial Intelligence*, vol.27, pp. 57-69.

**Feoktistov, V., & Janaqi, S.**, 2004, “Generalization of the strategies in differential evolution,” *Parallel and Distributed Processing Symposium Proceedings. 18th International*, pp. 165.

**Gämperle, R., Müller, S. D., and Koumoutsakos, P.**, 2002, “A parameter study for differential evolution,” *Advances in intelligent systems, fuzzy systems, evolutionary computation*, vol. 10, pp. 293-298.

**Gavish, B., and Pirkul, H.**, 1985. “Efficient algorithms for solving multiconstraint zero-one knapsack problems to optimality,” *Mathematical programming*, vol. 31, no. 1, pp. 78-105.

**Gilmore, P. C., & Gomory, R. E.**, 1966, “The theory and computation of knapsack functions,” *Operations Research*, vol. 14, no.6, pp. 1045-1074.

**Glover, F., and Kochenberger, G. A.**, 1996, Critical event tabu search for multidimensional knapsack problems. In *Meta-Heuristics*, pp. 407-427, Springer US.

**Gottlieb, J.**, 2001, On the feasibility problem of penalty-based evolutionary algorithms for knapsack problems. *Lecture notes in computer science*, pp. 50-59.

**Hanafi, S., and Freville, A.,** 1998, “An efficient tabu search approach for the 0–1 multidimensional knapsack problem,” *European Journal of Operational Research*, vol. 106, no.2, pp. 659-675.

**Hansen N. and Ostermeier A.,** 2001, “Completely derandomized self-adaptation in evolution strategies”, *Evolutionary Computation*, 9(2) pp. 159–195.

**Huang V. L., Qin A. K., and Suganthan P. N.,** 2006, "Self-adaptive Differential Evolution Algorithm for Constrained Real-Parameter Optimization", Nanyang Technological University, Singapore.

**Jin, Y.** 2005, “A comprehensive survey of fitness approximation in evolutionary computation” *Soft computing*, vol. 9, no. 1, pp. 3-12.

**Kennedy, J., & Eberhart, R. C.,** 1997, “A discrete binary version of the particle swarm algorithm,” In *Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation., 1997 IEEE International Conference*, Vol. 5, pp. 4104-4108.

**Kong, M., Tian, P., and Kao, Y.,** 2008, “A new ant colony optimization algorithm for the multidimensional Knapsack problem,” *Computers & Operations Research*, vol. 35, no. 8, pp. 2672-2683.

**Koziel, S., and Michalewicz, Z.,** 1999, “Evolutionary algorithms, homomorphous mappings, and constrained parameter optimization,” *Evolutionary computation*, vol. 7 no.1, pp. 19-44.

**Lampinen, J.,** 2001, A bibliography of differential evolution algorithm, *Lappeenranta University of Technology, Finland*.

**Lampinen, J.,** 1999, “Differential evolution- New naturally parallel approach for engineering design optimization,” *Developments in computational mechanics with high performance computing*, pp. 217-228.

**Lampinen, J.**, 2002, A constraint handling approach for the differential evolution algorithm., WCCI, pp. 1468-1473.

**Langdon, W. B., and Poli, R.**, 2007, “Evolving problems to learn about particle swarm optimizers and other search algorithms,” *Evolutionary Computation, IEEE Transactions*, vol. 11, no. 5, pp. 561-578.

**Liang J. J. and P. N. Suganthan**, 2006, "Dynamic Multi-Swarm Particle Swarm Optimizer with a Novel Constraint-Handling Mechanism", Nanyang Technological University, Singapore 639798.

**Liang J. J., Runarsson T. P., Mezura-Montes E., Clerc M., Suganthan P. N., Coello Coello C. A., Deb K.**, 2006, "Problem Definitions and Evaluation Criteria for the CEC 2006", Special Session on Constrained Real-Parameter Optimization, Technical Report, Nanyang Technological University, Singapore.

**Langdon, W. B., and Poli, R.**, 2007, “Evolving problems to learn about particle swarm optimizers and other search algorithms,” *Evolutionary Computation, IEEE Transactions on*, vol 11 no. 5, pp. 561-578.

**Liu J. and Lampinen J.**, 2005, “A fuzzy adaptive differential evolution algorithm,” *Soft Computing—A Fusion of Foundations, Methodologies and Applications*, vol. 9, no. 6, pp. 448–462, 2005 [Online]. Available:

<http://springerlink.metapress.com/index/10.1007/s00500-004-0363-x>

**Mallipeddi, R., Suganthan, P. N., Pan, Q. K., & Tasgetiren, M. F.**, 2011., “Differential evolution algorithm with ensemble of parameters and mutation strategies,” *Applied Soft Computing*, vol. 11 no. 2, pp. 1679-1696.

**Mallipeddi, R., & Suganthan, P. N.**, 2011, Ensemble differential evolution algorithm for CEC2011 problems. In *Evolutionary Computation (CEC), 2011 IEEE Congress on* (pp. 1557-1564). IEEE.

**Martins, J. P., Bringel Neto, C., Crocomo, M. K., Vittori, K., & Delbem, A. C.,** 2013, “A comparison of linkage-learning-based genetic algorithms in multidimensional knapsack problems,” In *Evolutionary Computation (CEC), 2013 IEEE Congress*, pp. 502-509.

**Mezura-Montes, E., Velázquez-Reyes, J., & Coello Coello, C.,** 2006, “Modified differential evolution for constrained optimization.” In *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on* pp. 25-32

**Michalewicz, Z., and Schoenauer, M.,** 1996, “Evolutionary algorithms for constrained parameter optimization problems” *Evolutionary computation*, vol. 4 no.1 pp. 1-32.

**Mirjalili, S., & Lewis, A.,** 2013., “S-shaped versus V-shaped transfer functions for binary particle swarm optimization,” *Swarm and Evolutionary Computation*, vol. 9, pp. 1-14.

**Nelder, J. A., and Mead, R.,** 1965, “A simplex method for function minimization”.*The computer journal*, vol 7(4), pp. 308-313.

**Neri F. and Tirronen V.,** 2010, “Recent advances in differential evolution: A review and experimental analysis,” *Artif. Intell. Rev.*, vol. 33, no. 1, pp. 61–106.

**Plateau, G., and Elkihel, M.** 1985, A hybrid method for the 0–1 knapsack problem. *Methods of Operations Research*, vol. 49, pp. 277-293.

**Powell, D., and Skolnick, M. M.,** 1993, “Using genetic algorithms in engineering design optimization with non-linear constraints,” *Proceedings of the 5th international conference on genetic algorithms* pp. 424-431 Morgan Kaufmann Publishers Inc..

**Price, W. L.,** 1977, “Global optimization by controlled random search”. *Journal of Optimization Theory and Applications*, vol 20 no 4, pp. 333-348.

**Price K., Storn R., Lampinen J.,** 2005, *Differential Evolution: A Practical Approach to Global Optimization*, Springer-Verlag, Berlin.

**Price K. V., and Storn R.,** 1997, Differential evolution: A simple evolution strategy for fast optimization, *Dr. Dobb's J.*, vol. 22, no. 4, pp. 18–24.

**Qin A. K. and Suganthan P. N.,** 2005, “Self-adaptive Differential Evolution Algorithm for Numerical Optimization”, *Proc. IEEE Congress on Evolutionary Computation.*

**Rahnamayan S., Tizhoosh H. R., and Salama M. M. A.,** 2008, “Opposition based differential evolution,” *IEEE Trans. Evol. Comput.*, vol. 12, no. 1, pp. 64–79.

**Raidl, G. R.,** 1998, “An improved genetic algorithm for the multiconstrained 0-1 knapsack problem,” In *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference,* pp. 207-211.

**Rashedi, E., Nezamabadi-Pour, H., & Saryazdi, S.,** 2010, “BGSA: binary gravitational search algorithm,” *Natural Computing*, vol. 9 no. 3, pp. 727-745.

**Runarsson, T. P., and Yao, X.,** 2000, “Stochastic ranking for constrained evolutionary optimization,” *Evolutionary Computation, IEEE Transactions*, vol.4, no. 3, pp. 284-294.

**Runarsson, T. P., and Yao, X.,** 2005, “Search biases in constrained evolutionary optimization,” *IEEE Trans. Syst. Man., Cybern.*, vol. 35, no.2, pp. 233-243.

**Shih, W.,** 1979, A branch and bound method for the multiconstraint zero-one knapsack problem, *Journal of the Operational Research Society*, pp. 369-378.

**Smith, A. E., and Tate, D. M.,** 1993, “Genetic optimization using a penalty function,” *Proceedings of the 5th international conference on genetic algorithms*, pp. 499-505 Morgan Kaufmann Publishers Inc.

**Storn R. and Price K. V.,** 1995 “Differential evolution: A simple and efficient adaptive scheme for global optimization over continuous spaces,” *ICSI, USA, Tech. Rep. TR-95-012*, [Online]. Available: <http://icsi.berkeley.edu/~storn/litera.html>

**Storn R. and Price K. V.**, 1997, Differential evolution: A simple and efficient heuristic for global optimization over continuous spaces, *J. Global Optimization*, vol. 11, no. 4, pp. 341–359.

**Storn, R.**, (1999) “System Design by Constraint Adaptation and Differential Evolution,” *IEEE Transactions on Evolutionary Computation*, vol. 3, pp. 22-34.

**Sutton, A. M., Lunacek, M., & Whitley, L. D.**, 2007, “Differential evolution and non-separability: using selective pressure to focus search,” In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pp. 1428-1435, ACM.

**Takahama T. and Sakai S.**, 2006, "Constrained Optimization by the Constrained Differential Evolution with Gradient-Based Mutation and Feasible Elites," in *IEEE Congress on Evolutionary Computation Sheraton Vancouver Wall Centre Hotel*, Vancouver, BC, Canada, pp. 1-8.

**Tasgetiren M. F., and Suganthan P. N.**, 2006, “A Multi-Populated Differential Evolution Algorithm for Solving Constrained Optimization Problems”, Fatih University, 34500, Buyukcekmece, Istanbul, Turkey.

**Tavares, J., Pereira, F. B., & Costa, E.**, 2006, “The role of representation on the multidimensional knapsack problem by means of fitness landscape analysis,” In *Evolutionary Computation, 2006*, pp. 2307-2314.

**Tavares, J., Pereira, F. B., & Costa, E.**, 2008, “Multidimensional knapsack problem: A fitness landscape analysis,” *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions*, vol. 38, no. 3, pp. 604-616.

**Tessema, B., & Yen, G. G.**, 2006, “A self adaptive penalty function based algorithm for constrained optimization,” *Evolutionary Computation, 2006. CEC 2006. IEEE Congress*, pp. 246-253.

**Tessema, B., & Yen, G. G.,** 2009, “An adaptive penalty formulation for constrained evolutionary optimization.” *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions*, vol. 39 no. 3, pp. 565-578.

**Toth, P.,** 1980, “Dynamic programming algorithms for the zero-one knapsack problem,” *Computing*, vol. 25, no.1, pp. 29-45.

**Vasquez, M., and Hao, J. K.,** 2001, “A hybrid approach for the 0-1 multidimensional knapsack problem” *IJCAI*, pp. 328-333.

**Vesterstrom J. and Thomson R.A.,** 2004, “Comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems,” in *Proc. Congr. Evol. Comput.*, pp. 1382-1389.

**Wang, L., and Zheng, D. Z.,** 2001, “ An effective hybrid optimization strategy for job-shop scheduling problems,” *Computers & Operations Research*, vol. 28, no.6, pp. 585-596.

**Wang, Y., Cai, Z., Guo, G., & Zhou, Y.,** 2007, “Multiobjective optimization and hybrid evolutionary algorithm to solve constrained optimization problems,” *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions* vol.37, no.3, pp. 560-575.

**Wang, L., Fu, X., Mao, Y., Menhas, M. I., & Fei, M.,** 2012, “A novel modified binary differential evolution algorithm and its applications,” *Neurocomputing*, vol 98, pp. 55-75.

**Wang, L., Zheng, X. L., & Wang, S. Y.,** 2013., “A novel binary fruit fly optimization algorithm for solving the multidimensional knapsack problem,” *Knowledge-Based Systems*, vol 48, pp. 17-23.

**Wolpert, D. H., & Macready, W. G.,** 1997 “No free lunch theorems for optimization,” *IEEE Trans. Evol. Comput.*, vol.1 no. 1, pp. 67-82.

**Zhang J. and Sanderson A. C.**, 2009, “JADE: Adaptive differential evolution with optional external archive,” *IEEE Trans. Evol. Comput.*, vol. 13, no. 5, pp. 945–958.

**Zhang, J., and Sanderson, A. C.**, 2007, “JADE: Self-adaptive differential evolution with fast and reliable convergence performance,” *IEEE Evol. Comput.*, 2007. CEC 2007. pp. 2251-2258.



## **CURRICULUM VITEA**

Mert Paldrak was born in İzmir/Konak in 1991. He has been working as a research assistant at the Department of Industrial Engineering at Yaşar University in İzmir since October 2013. He has received his BSc degree in Industrial Engineering with the 2nd degree from Yaşar University Engineering Faculty in 2013 and then he continued his MSc degree in Industrial Engineering in Yaşar University Engineering Faculty. During his master degree period, he has taken such various courses as System Simulation, Optimization Models and Algorithms, Heuristic Optimization, Scheduling Theory, Probabilistic Analysis and Applied Stochastic Processes, Dynamic Programming, Mathematics of Operational Research and Supply Chain and Management.

## APPENDIX 1 CEC 2006 BENCHMARKS

Function1: g01 (Floudas C. and Pardalos P., 1987)

g01

Minimize

$$f(\vec{x}) = 5 \sum_{i=1}^4 x_i - 5 \sum_{i=1}^4 x_i^2 - \sum_{i=5}^{13} x_i$$

subject to:

$$g_1(\vec{x}) = 2x_1 + 2x_2 + x_{10} + x_{11} - 10 \leq 0$$

$$g_2(\vec{x}) = 2x_1 + 2x_3 + x_{10} + x_{12} - 10 \leq 0$$

$$g_3(\vec{x}) = 2x_2 + 2x_3 + x_{11} + x_{12} - 10 \leq 0$$

$$g_4(\vec{x}) = -8x_1 + x_{10} \leq 0$$

$$g_5(\vec{x}) = -8x_2 + x_{11} \leq 0$$

$$g_6(\vec{x}) = -8x_3 + x_{12} \leq 0$$

$$g_7(\vec{x}) = -2x_4 - x_5 + x_{10} \leq 0$$

$$g_8(\vec{x}) = -2x_6 - x_7 + x_{11} \leq 0$$

$$g_9(\vec{x}) = -2x_8 - x_9 + x_{12} \leq 0$$

where the bounds are  $0 \leq x_i \leq 1$  ( $i = 1, \dots, 9$ ),  $0 \leq x_i \leq 100$  ( $i = 10, 11, 12$ ) and  $0 \leq x_{13} \leq 1$ . The global minimum is at  $\vec{x}^* = (1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3, 1)$  where six constraints are active ( $g_1, g_2, g_3, g_7, g_8$  and  $g_9$ ) and  $f(\vec{x}^*) = -15$ .

Function2: g02 (Kozieland S., Michalewicz Z., 1999)

g02

Minimize

$$f(\vec{x}) = - \left| \frac{\sum_{i=1}^n \cos^4(x_i) - 2 \prod_{i=1}^n \cos^2(x_i)}{\sqrt{\sum_{i=1}^n i x_i^2}} \right|$$

subject to:

$$g_1(\vec{x}) = 0.75 - \prod_{i=1}^n x_i \leq 0$$

$$g_2(\vec{x}) = \sum_{i=1}^n x_i - 7.5n \leq 0$$

where  $n = 20$  and  $0 < x_i \leq 10$  ( $i = 1, \dots, n$ ). The global minimum  $\vec{x}^* = (3.16246061572185, 3.12833142812967, 3.09479212988791, 3.06145059523469, 3.02792915885555, 2.99382606701730, 2.95866871765285, 2.92184227312450, 0.49482511456933, 0.48835711005490, 0.48231642711865, 0.47664475092742, 0.47129550835493, 0.46623099264167, 0.46142004984199, 0.45683664767217, 0.45245876903267, 0.44826762241853, 0.44424700958760, 0.44038285956317)$ , the best we found is  $f(\vec{x}^*) = -0.80361910412559$  (which, to the best of our knowledge, is better than any reported value), constraint  $g_1$  is close to being active.

Function 3: g03 (Michalewicz Z., Nazhiyath G., and Michalewicz M., 1996)

g03

Minimize

$$f(\vec{x}) = -(\sqrt{n})^n \prod_{i=1}^n x_i$$

subject to:

$$h_1(\vec{x}) = \sum_{i=1}^n x_i^2 - 1 = 0$$

where  $n = 10$  and  $0 \leq x_i \leq 1$  ( $i = 1, \dots, n$ ). The global minimum is at  $\vec{x}^* = (0.31624357647283069, 0.316243577414338339, 0.316243578012345927, 0.316243575664017895, 0.316243578205526066, 0.31624357738855069, 0.316243575472949512, 0.316243577164883938, 0.316243578155920302, 0.316243576147374916)$  where  $f(\vec{x}^*) = -1.00050010001000$ .

Function 4: g04 (Himmelblau D., 1972)

g04

Minimize

$$f(\vec{x}) = 5.3578547x_3^2 + 0.8356891x_1x_5 + 37.293239x_1 - 40792.141$$

subject to:

$$g_1(\vec{x}) = 85.334407 + 0.0056858x_2x_5 + 0.0006262x_1x_4 - 0.0022053x_3x_5 - 92 \leq 0$$

$$g_2(\vec{x}) = -85.334407 - 0.0056858x_2x_5 - 0.0006262x_1x_4 + 0.0022053x_3x_5 \leq 0$$

$$g_3(\vec{x}) = 80.51249 + 0.0071317x_2x_5 + 0.0029955x_1x_2 + 0.0021813x_3^2 - 110 \leq 0$$

$$g_4(\vec{x}) = -80.51249 - 0.0071317x_2x_5 - 0.0029955x_1x_2 - 0.0021813x_3^2 + 90 \leq 0$$

$$g_5(\vec{x}) = 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 + 0.0019085x_3x_4 - 25 \leq 0$$

$$g_6(\vec{x}) = -9.300961 - 0.0047026x_3x_5 - 0.0012547x_1x_3 - 0.0019085x_3x_4 + 20 \leq 0$$

where  $78 \leq x_1 \leq 102$ ,  $33 \leq x_2 \leq 45$  and  $27 \leq x_i \leq 45$  ( $i = 3, 4, 5$ ). The optimum solution is  $\vec{x}^* = 78, 33, 29.9952560256815985, 45, 36.7758129057882073)$  where  $f(\vec{x}^*) = -3.066553867178332e + 004$ . Two constraints are active ( $g_1$  and  $g_6$ ).

Function 5: g05 (Hock W. and Schittkowski K., 1981)

g05

Minimize

$$f(\vec{x}) = 3x_1 + 0.000001x_1^3 + 2x_2 + (0.000002/3)x_2^3$$

subject to:

$$g_1(\vec{x}) = -x_4 + x_3 - 0.55 \leq 0$$

$$g_2(\vec{x}) = -x_3 + x_4 - 0.55 \leq 0$$

$$h_3(\vec{x}) = 1000 \sin(-x_3 - 0.25) + 1000 \sin(-x_4 - 0.25) + 894.8 - x_1 = 0$$

$$h_4(\vec{x}) = 1000 \sin(x_3 - 0.25) + 1000 \sin(x_3 - x_4 - 0.25) + 894.8 - x_2 = 0$$

$$h_5(\vec{x}) = 1000 \sin(x_4 - 0.25) + 1000 \sin(x_4 - x_3 - 0.25) + 1294.8 = 0$$

where  $0 \leq x_1 \leq 1200$ ,  $0 \leq x_2 \leq 1200$ ,  $-0.55 \leq x_3 \leq 0.55$  and  $-0.55 \leq x_4 \leq 0.55$ . The best known solution [4]  $\vec{x}^* = (679.945148297028709, 1026.06697600004691, 0.118876369094410433, -0.39623348521517826)$  where  $f(\vec{x}^*) = 5126.4967140071$ .

Function 6: g06 (Floudas C. and Pardalos P., 1987)

g06

Minimize

$$f(\vec{x}) = (x_1 - 10)^3 + (x_2 - 20)^3$$

subject to:

$$g_1(\vec{x}) = -(x_1 - 5)^2 - (x_2 - 5)^2 + 100 \leq 0$$

$$g_2(\vec{x}) = (x_1 - 6)^2 + (x_2 - 5)^2 - 82.81 \leq 0$$

where  $13 \leq x_1 \leq 100$  and  $0 \leq x_2 \leq 100$ . The optimum solution is  $\vec{x}^* = (14.09500000000000064, 0.8429607892154795668)$  where  $f(\vec{x}^*) = -6961.81387558015$ . Both constraints are active.

Function 7: g07 (Hock W. and Schittkowski K., 1981)

g07

Minimize

$$f(\vec{x}) = x_1^2 + x_2^2 + x_1x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2 + 4(x_4 - 5)^2 + (x_5 - 3)^2 \\ + 2(x_6 - 1)^2 + 5x_7^2 + 7(x_8 - 11)^2 + 2(x_9 - 10)^2 + (x_{10} - 7)^2 + 45$$

subject to:

$$g_1(\vec{x}) = -105 + 4x_1 + 5x_2 - 3x_7 + 9x_8 \leq 0$$

$$g_2(\vec{x}) = 10x_1 - 8x_2 - 17x_7 + 2x_8 \leq 0$$

$$g_3(\vec{x}) = -8x_1 + 2x_2 + 5x_9 - 2x_{10} - 12 \leq 0$$

$$g_4(\vec{x}) = 3(x_1 - 2)^2 + 4(x_2 - 3)^2 + 2x_3^2 - 7x_4 - 120 \leq 0$$

$$g_5(\vec{x}) = 5x_1^2 + 8x_2 + (x_3 - 6)^2 - 2x_4 - 40 \leq 0$$

$$g_6(\vec{x}) = x_1^2 + 2(x_2 - 2)^2 - 2x_1x_2 + 14x_5 - 6x_6 \leq 0$$

$$g_7(\vec{x}) = 0.5(x_1 - 8)^2 + 2(x_2 - 4)^2 + 3x_5^2 - x_6 - 30 \leq 0$$

$$g_8(\vec{x}) = -3x_1 + 6x_2 + 12(x_9 - 8)^2 - 7x_{10} \leq 0$$

where  $-10 \leq x_i \leq 10$  ( $i = 1, \dots, 10$ ). The optimum solution is  $\vec{x}^* = (2.17199634142692, 2.3636830416034, 8.77392573913157, 5.09598443745173, 0.990654756560493, 1.43057392853463, 1.32164415364306, 9.82872576524495, 8.2800915887356, 8.3759266477347)$  where  $g07(\vec{x}^*) = 24.30620906818$  (The recorded results may suffer from rounding errors which may cause slight infeasibility sometimes in the best given solutions). Six constraints are active ( $g_1, g_2, g_3, g_4, g_5$  and  $g_6$ ).

Function 8: g08 (Kozieland S., Michalewicz Z., 1999)

g08

Minimize

$$f(\vec{x}) = -\frac{\sin^3(2\pi x_1) \sin(2\pi x_2)}{x_1^3(x_1 + x_2)}$$

subject to:

$$g_1(\vec{x}) = x_1^2 - x_2 + 1 \leq 0$$

$$g_2(\vec{x}) = 1 - x_1 + (x_2 - 4)^2 \leq 0$$

where  $0 \leq x_1 \leq 10$  and  $0 \leq x_2 \leq 10$ . The optimum is located at  $\vec{x}^* = (1.22797135260752599, 4.24537336612274885)$  where  $f(\vec{x}^*) = -0.0958250414180359$ .

Function 9: g09 (Hock W. and Schittkowski K., 1981)

**g09**

Minimize

$$f(\vec{x}) = (x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^4 + 3(x_4 - 11)^2 \\ + 10x_5^6 + 7x_6^2 + x_7^4 - 4x_6x_7 - 10x_6 - 8x_7$$

subject to:

$$g_1(\vec{x}) = -127 + 2x_1^2 + 3x_2^4 + x_3 + 4x_4^2 + 5x_5 \leq 0$$

$$g_2(\vec{x}) = -282 + 7x_1 + 3x_2 + 10x_3^2 + x_4 - x_5 \leq 0$$

$$g_3(\vec{x}) = -196 + 23x_1 + x_2^2 + 6x_6^2 - 8x_7 \leq 0$$

$$g_4(\vec{x}) = 4x_1^2 + x_2^2 - 3x_1x_2 + 2x_3^2 + 5x_6 - 11x_7 \leq 0$$

where  $-10 \leq x_i \leq 10$  for  $(i = 1, \dots, 7)$ . The optimum solution is  $\vec{x}^* = (2.33049935147405174, 1.95137236847114592, -0.477541399510615805, 4.36572624923625874, -0.624486959100388983,$

$1.03813099410962173, 1.5942266780671519)$  where  $f(\vec{x}^*) = 680.630057374402$ . Two constraints are active ( $g_1$  and  $g_4$ ).

Function 10: g10 (Hock W. and Schittkowski K., 1981)

**g10**

Minimize

$$f(\vec{x}) = x_1 + x_2 + x_3$$

subject to:

$$g_1(\vec{x}) = -1 + 0.0025(x_4 + x_6) \leq 0$$

$$g_2(\vec{x}) = -1 + 0.0025(x_5 + x_7 - x_4) \leq 0$$

$$g_3(\vec{x}) = -1 + 0.01(x_8 - x_5) \leq 0$$

$$g_4(\vec{x}) = -x_1x_6 + 833.33252x_4 + 100x_1 - 83333.333 \leq 0$$

$$g_5(\vec{x}) = -x_2x_7 + 1250x_5 + x_2x_4 - 1250x_4 \leq 0$$

$$g_6(\vec{x}) = -x_3x_8 + 1250000 + x_3x_5 - 2500x_5 \leq 0$$

where  $100 \leq x_1 \leq 10000$ ,  $1000 \leq x_i \leq 10000$  ( $i = 2, 3$ ) and  $10 \leq x_i \leq 1000$  ( $i = 4, \dots, 8$ ). The optimum solution is  $\vec{x}^* = (579.306685017979589, 1359.97067807935605, 5109.97065743133317, 182.01769963061534, 295.601173702746792, 217.982300369384632, 286.41652592786852, 395.601173702746735)$ , where  $f(\vec{x}^*) = 7049.24802052867$ . All constraints are active ( $g_1, g_2$  and  $g_3$ ).

Function 11: g11 (Kozieland S., Michalewicz Z., 1999)

**g11**

Minimize

$$f(\vec{x}) = x_1^2 + (x_2 - 1)^2$$

subject to:

$$h(\vec{x}) = x_2 - x_1^2 = 0$$

where  $-1 \leq x_1 \leq 1$  and  $-1 \leq x_2 \leq 1$ . The optimum solution is  $\vec{x}^* = (-0.707036070037170616, 0.500000004333606807)$  where  $f(\vec{x}^*) = 0.7499$ .

Function 12: g12 (Kozieland S., Michalewicz Z., 1999)

**g12**

Minimize

$$f(\vec{x}) = -(100 - (x_1 - 5)^2 - (x_2 - 5)^2 - (x_3 - 5)^2)/100$$

subject to:

$$g(\vec{x}) = (x_1 - p)^2 + (x_2 - q)^2 + (x_3 - r)^2 - 0.0625 \leq 0$$

where  $0 \leq x_i \leq 10$  ( $i = 1, 2, 3$ ) and  $p, q, r = 1, 2, \dots, 9$ . The feasible region of the search space consists of  $9^3$  disjointed spheres. A point  $(x_1, x_2, x_3)$  is feasible if and only if there exist  $p, q, r$  such that the above inequality holds. The optimum is located at  $\vec{x}^* = (5, 5, 5)$  where  $f(\vec{x}^*) = -1$ . The solution lies within the feasible region.

Function 13: g13 (Hock W. and Schittkowski K., 1981)

**g13**

Minimize

$$f(\vec{x}) = e^{x_1 x_2 x_3 x_4 x_5}$$

subject to:

$$h_1(\vec{x}) = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 - 10 = 0$$

$$h_2(\vec{x}) = x_2 x_3 - 5 x_4 x_5 = 0$$

$$h_3(\vec{x}) = x_1^3 + x_2^3 + 1 = 0$$

where  $-2.3 \leq x_i \leq 2.3$  ( $i = 1, 2$ ) and  $-3.2 \leq x_i \leq 3.2$  ( $i = 3, 4, 5$ ). The optimum solution is  $\vec{x}^* = (-1.71714224003, 1.59572124049468, 1.8272502406271, -0.763659881912867, -0.76365986736498)$  where  $f(\vec{x}^*) = 0.053941514041898$ .

Function 14: g14 (Himmelblau D. M., 1972)

g14

Minimize

$$f(\vec{x}) = \sum_{i=1}^{10} x_i \left( c_i + \ln \frac{x_i}{\sum_{j=1}^{10} x_j} \right)$$

subject to:

$$h_1(\vec{x}) = x_1 + 2x_2 + 2x_3 + x_6 + x_{10} - 2 = 0$$

$$h_2(\vec{x}) = x_4 + 2x_5 + x_6 + x_7 - 1 = 0$$

$$h_3(\vec{x}) = x_3 + x_7 + x_8 + 2x_9 + x_{10} - 1 = 0$$

where the bounds are  $0 < x_i \leq 10$  ( $i = 1, \dots, 10$ ), and  $c_1 = -6.089$ ,  $c_2 = -17.164$ ,  $c_3 = -34.054$ ,  $c_4 = -5.914$ ,  $c_5 = -24.721$ ,  $c_6 = -14.986$ ,  $c_7 = -24.1$ ,  $c_8 = -10.708$ ,  $c_9 = -26.662$ ,  $c_{10} = -22.179$ . The best known solution is at  $x^* = (0.04066684113216282, 0.147721240492452, 0.783205732104114, 0.00141433931889084, 0.485293636780388, 0.000693183051556082, 0.0274052040687766, 0.0179509660214818, 0.0373268186859717, 0.0968844604336845)$  where  $f(x^*) = -47.7648884594915$ .

Function 15: g15 (Himmelblau D. M., 1972)

g15

Minimize

$$f(\vec{x}) = 1000 - x_1^2 - 2x_2^2 - x_3^2 - x_1x_2 - x_1x_3$$

subject to:

$$h_1(\vec{x}) = x_1^2 + x_2^2 + x_3^2 - 25 = 0$$

$$h_2(\vec{x}) = 8x_1 + 14x_2 + 7x_3 - 56 = 0$$

where the bounds are  $0 \leq x_i \leq 10$  ( $i = 1, 2, 3$ ). The best known solution is at  $x^* = (3.51212812611795133, 0.216987510429556135, 3.55217854929179921)$  where  $f(x^*) = 961.715022289961$ .



Function 16: g16 (Himmelblau D. M., 1972)

g16

Minimize

$$f(\vec{x}) = 0.000117y_{14} + 0.1365 + 0.00002358y_{13} + 0.000001502y_{16} + 0.0321y_{12} \\ + 0.004324y_5 + 0.0001\frac{c_{15}}{c_{16}} + 37.48\frac{y_2}{c_{12}} - 0.0000005843y_{17}$$

subject to:

$$g_1(\vec{x}) = \frac{0.28}{0.72}y_5 - y_4 \leq 0$$

$$g_2(\vec{x}) = x_3 - 1.5x_2 \leq 0$$

$$g_3(\vec{x}) = 3496\frac{y_2}{c_{12}} - 21 \leq 0$$

$$g_4(\vec{x}) = 110.6 + y_1 - \frac{62212}{c_{17}} \leq 0$$

$$g_5(\vec{x}) = 213.1 - y_1 \leq 0$$

$$g_6(\vec{x}) = y_1 - 405.23 \leq 0$$

$$g_7(\vec{x}) = 17.505 - y_2 \leq 0$$

$$g_8(\vec{x}) = y_2 - 1053.6667 \leq 0$$

$$g_9(\vec{x}) = 11.275 - y_3 \leq 0$$

$$g_{10}(\vec{x}) = y_3 - 35.03 \leq 0$$

$$g_{11}(\vec{x}) = 214.228 - y_4 \leq 0$$

$$g_{12}(\vec{x}) = y_4 - 665.585 \leq 0$$

$$g_{13}(\vec{x}) = 7.458 - y_5 \leq 0$$

$$g_{14}(\vec{x}) = y_5 - 584.463 \leq 0$$

$$g_{15}(\vec{x}) = 0.961 - y_6 \leq 0$$

$$g_{16}(\vec{x}) = y_6 - 265.916 \leq 0$$

$$g_{17}(\vec{x}) = 1.612 - y_7 \leq 0$$

$$g_{18}(\vec{x}) = y_7 - 7.046 \leq 0$$

$$g_{19}(\vec{x}) = 0.146 - y_8 \leq 0$$

$$g_{20}(\vec{x}) = y_8 - 0.222 \leq 0$$

$$g_{21}(\vec{x}) = 107.99 - y_9 \leq 0$$

$$g_{22}(\vec{x}) = y_9 - 273.366 \leq 0$$

$$g_{23}(\vec{x}) = 922.693 - y_{10} \leq 0$$

$$g_{24}(\vec{x}) = y_{10} - 1286.105 \leq 0$$

$$g_{25}(\vec{x}) = 926.832 - y_{11} \leq 0$$

$$g_{26}(\vec{x}) = y_{11} - 1444.046 \leq 0$$

$$g_{27}(\vec{x}) = 18.766 - y_{12} \leq 0$$

$$g_{28}(\vec{x}) = y_{12} - 537.141 \leq 0$$

$$g_{29}(\vec{x}) = 1072.163 - y_{13} \leq 0$$

$$\begin{aligned}
g_{30}(\vec{x}) &= y_{13} - 3247.039 \leq 0 \\
g_{31}(\vec{x}) &= 8961.448 - y_{14} \leq 0 \\
g_{32}(\vec{x}) &= y_{14} - 26844.086 \leq 0 \\
g_{33}(\vec{x}) &= 0.063 - y_{15} \leq 0 \\
g_{34}(\vec{x}) &= y_{15} - 0.386 \leq 0 \\
g_{35}(\vec{x}) &= 71084.33 - y_{16} \leq 0 \\
g_{36}(\vec{x}) &= -140000 + y_{16} \leq 0 \\
g_{37}(\vec{x}) &= 2802713 - y_{17} \leq 0 \\
g_{38}(\vec{x}) &= y_{17} - 12146108 \leq 0
\end{aligned}$$

where:

$$\begin{aligned}
y_1 &= x_2 + x_3 + 41.6 \\
c_1 &= 0.024x_4 - 4.62 \\
y_2 &= \frac{12.5}{c_1} + 12 \\
c_2 &= 0.0003535x_1^2 + 0.5311x_1 + 0.08705y_2x_1 \\
c_3 &= 0.052x_1 + 78 + 0.002377y_2x_1 \\
y_3 &= \frac{c_2}{c_3} \\
y_4 &= 19y_3 \\
c_4 &= 0.04782(x_1 - y_3) + \frac{0.1956(x_1 - y_3)^2}{x_2} + 0.6376y_4 + 1.594y_3 \\
c_5 &= 100x_2 \\
c_6 &= x_1 - y_3 - y_4 \\
c_7 &= 0.950 - \frac{c_4}{c_5} \\
y_5 &= c_6c_7 \\
y_6 &= x_1 - y_5 - y_4 - y_3 \\
c_8 &= (y_5 + y_4)0.995
\end{aligned}$$

$$\begin{aligned}
y_7 &= \frac{c_8}{y_1} \\
y_8 &= \frac{c_8}{3798} \\
c_9 &= y_7 - \frac{0.0663y_7}{y_8} - 0.3153 \\
y_9 &= \frac{96.82}{c_9} + 0.321y_1 \\
y_{10} &= 1.29y_5 + 1.258y_4 + 2.29y_3 + 1.71y_6 \\
y_{11} &= 1.71x_1 - 0.452y_4 + 0.580y_3 \\
c_{10} &= \frac{12.3}{752.3} \\
c_{11} &= (1.75y_2)(0.995x_1) \\
c_{12} &= 0.995y_{10} + 1998 \\
y_{12} &= c_{10}x_1 + \frac{c_{11}}{c_{12}} \\
y_{13} &= c_{12} - 1.75y_2 \\
y_{14} &= 3623 + 64.4x_2 + 58.4x_3 + \frac{146312}{y_9 + x_5}
\end{aligned}$$

$$c_{13} = 0.995y_{10} + 60.8x_2 + 48x_4 - 0.1121y_{14} - 5095$$

$$y_{15} = \frac{y_{13}}{c_{13}}$$

$$y_{16} = 148000 - 331000y_{15} + 40y_{13} - 61y_{15}y_{13}$$

$$c_{14} = 2324y_{10} - 28740000y_2$$

$$y_{17} = 14130000 - 1328y_{10} - 531y_{11} + \frac{c_{14}}{c_{12}}$$

$$c_{15} = \frac{y_{13}}{y_{15}} - \frac{y_{13}}{0.52}$$

$$c_{16} = 1.104 - 0.72y_{15}$$

$$c_{17} = y_9 + x_5$$

and where the bounds are  $704.4148 \leq x_1 \leq 906.3855$ ,  $68.6 \leq x_2 \leq 288.88$ ,  $0 \leq x_3 \leq 134.75$ ,  $193 \leq x_4 \leq 287.0966$  and  $25 \leq x_5 \leq 84.1988$ . The best known solution is at  $x^* = (705.174537070090537, 68.5999999999999943, 102.899999999999991, 282.324931593660324, 37.5841164258054832)$  where  $f(x^*) = -1.90515525853479$ .

## Function 17: g17 (Himmelblau D. M., 1972)

**g17**

Minimize

$$f(\vec{x}) = f(x_1) + f(x_2)$$

where

$$f_1(x_1) = \begin{cases} 30x_1 & 0 \leq x_1 < 300 \\ 31x_1 & 300 \leq x_1 < 400 \end{cases}$$

$$f_2(x_2) = \begin{cases} 28x_2 & 0 \leq x_2 < 100 \\ 29x_2 & 100 \leq x_2 < 200 \\ 30x_2 & 200 \leq x_2 < 1000 \end{cases}$$

subject to:

$$h_1(\vec{x}) = -x_1 + 300 - \frac{x_3x_4}{131.078} \cos(1.48477 - x_6) + \frac{0.90798x_3^2}{131.078} \cos(1.47588)$$

$$h_2(\vec{x}) = -x_2 - \frac{x_3x_4}{131.078} \cos((1.48477 + x_6) + \frac{0.90798x_4^2}{131.078} \cos(1.47588)$$

$$h_3(\vec{x}) = -x_5 - \frac{x_3x_4}{131.078} \sin((1.48477 + x_6) + \frac{0.90798x_4^2}{131.078} \sin(1.47588)$$

$$h_4(\vec{x}) = 200 - \frac{x_3x_4}{131.078} \sin((1.48477 - x_6) + \frac{0.90798x_3^2}{131.078} \sin(1.47588)$$

where the bounds are  $0 \leq x_1 \leq 400$ ,  $0 \leq x_2 \leq 1000$ ,  $340 \leq x_3 \leq 420$ ,  $340 \leq x_4 \leq 420$ ,  $-1000 \leq x_5 \leq 1000$  and  $0 \leq x_6 \leq 0.5236$ . The best known solution is at  $x^* = (201.784467214523659, 99.9999999999999005, 383.071034852773266, 420, -10.9076584514292652, 0.0731482312084287128)$  where  $f(x^*) = 8853.53967480648$ .

Function 18: g18 (Himmelblau D. M., 1972)

g18

Minimize

$$f(\vec{x}) = -0.5(x_1x_4 - x_2x_3 + x_3x_9 - x_5x_9 + x_5x_8 - x_6x_7)$$

subject to:

$$\begin{aligned} gg_1(\vec{x}) &= x_3^2 + x_4^2 - 1 \leq 0 \\ gg_2(\vec{x}) &= x_9^2 - 1 \leq 0 \\ gg_3(\vec{x}) &= x_5^2 + x_6^2 - 1 \leq 0 \\ gg_4(\vec{x}) &= x_1^2 + (x_2 - x_9)^2 - 1 \leq 0 \\ gg_5(\vec{x}) &= (x_1 - x_5)^2 + (x_2 - x_6)^2 - 1 \leq 0 \\ gg_6(\vec{x}) &= (x_1 - x_7)^2 + (x_2 - x_8)^2 - 1 \leq 0 \\ gg_7(\vec{x}) &= (x_3 - x_5)^2 + (x_4 - x_6)^2 - 1 \leq 0 \\ gg_8(\vec{x}) &= (x_3 - x_7)^2 + (x_4 - x_8)^2 - 1 \leq 0 \\ gg_9(\vec{x}) &= x_7^2 + (x_8 - x_9)^2 - 1 \leq 0 \\ gg_{10}(\vec{x}) &= x_2x_3 - x_1x_4 \leq 0 \\ gg_{11}(\vec{x}) &= -x_3x_9 \leq 0 \\ gg_{12}(\vec{x}) &= x_5x_9 \leq 0 \\ gg_{13}(\vec{x}) &= x_6x_7 - x_5x_8 \leq 0 \end{aligned}$$

where the bounds are  $-10 \leq x_i \leq 10$  ( $i = 1, \dots, 8$ ) and  $0 \leq x_9 \leq 20$ . The best known solution is at  $x^* = (-0.657776192427943163, -0.153418773482438542, 0.323413871675240938, -0.946257611651304398, -0.657776194376798906, -0.753213434632691414, 0.323413874123576972, -0.346462947962331735, 0.59979466285217542)$  where  $f(x^*) = -0.866025403784439$ .

Function 19: g19 (Himmelblau D. M., 1972)

g19

Minimize

$$f(\vec{x}) = \sum_{j=1}^5 \sum_{i=1}^5 c_{ij}x_{(10+i)}x_{(10+j)} + 2 \sum_{j=1}^5 d_jx_{(10+j)}^3 - \sum_{i=1}^{10} b_ix_i$$

subject to:

$$g_j(\vec{x}) = -2 \sum_{i=1}^5 c_{ij}x_{(10+i)} - 3d_jx_{(10+j)}^2 - e_j + \sum_{i=1}^{10} a_{ij}x_i \leq 0 \quad j = 1, \dots, 5$$

where  $\vec{b} = [-40, -2, -0.25, -4, -4, -1, -40, -60, 5, 1]$  and the remaining data is on Table 1. The bounds are  $0 \leq x_i \leq 10$  ( $i = 1, \dots, 15$ ). The best known solution is at  $x^* = (1.66991341326291344e - 17, 3.95378229282456509e - 16, 3.94599045143233784, 1.06036597479721211e - 16, 3.2831773458454161, 9.9999999999999822, 1.12829414671605333e - 17, 1.2026194599794709e - 17, 2.50706276000769697e - 15, 2.24624122987970677e - 15, 0.370764847417013987, 0.278456024942955571, 0.523838487672241171, 0.388620152510322781, 0.298156764974678579)$  where  $f(x^*) = 32.6555929502463$ .

j	1	2	3	4	5
$e_j$	-15	-27	-36	-18	-12
$c_{1j}$	30	-20	-10	32	-10
$c_{2j}$	-20	39	-6	-31	32
$c_{3j}$	-10	-6	10	-6	-10
$c_{4j}$	32	-31	-6	39	-20
$c_{5j}$	-10	32	-10	-20	30
$d_j$	4	8	10	6	2
$a_{1j}$	-16	2	0	1	0
$a_{2j}$	0	-2	0	0.4	2
$a_{3j}$	-3.5	0	2	0	0
$a_{4j}$	0	-2	0	-4	-1
$a_{5j}$	0	-9	-2	1	-2.8
$a_{6j}$	2	0	-4	0	0
$a_{7j}$	-1	-1	-1	-1	-1
$a_{8j}$	-1	-2	-3	-2	-1
$a_{9j}$	1	2	3	4	5
$a_{10j}$	1	1	1	1	1

Data set for test problem g19

Function 20: g20 (Himmelblau D. M., 1972)

g20

Minimize

$$f(\vec{x}) = \sum_{i=1}^{24} a_i x_i$$

subject to:

$$g_i(\vec{x}) = \frac{(x_i + x_{(i+12)})}{\sum_{j=1}^{24} x_j + e_i} \leq 0 \quad i = 1, 2, 3$$

$$g_i(\vec{x}) = \frac{(x_{(i+3)} + x_{(i+15)})}{\sum_{j=1}^{24} x_j + e_i} \leq 0 \quad i = 4, 5, 6$$

$$h_i(\vec{x}) = \frac{x_{(i+12)}}{b_{(i+12)} \sum_{j=13}^{24} \frac{x_j}{b_j}} - \frac{c_i x_i}{40 b_i \sum_{j=1}^{12} \frac{x_j}{b_j}} = 0 \quad i = 1, \dots, 12$$

$$h_{13}(\vec{x}) = \sum_{i=1}^{24} x_i - 1 = 0$$

$$h_{14}(\vec{x}) = \sum_{i=1}^{12} \frac{x_i}{d_i} + k \sum_{i=13}^{24} \frac{x_i}{b_i} - 1.671 = 0$$

where  $k = (0.7302)(530)(\frac{14.7}{40})$  and the data set is detailed on Table 2. The bounds are  $0 \leq x_i \leq 10$  ( $i = 1, \dots, 24$ ). The best known solution is at  $x^* = (1.28582343498528086e - 18, 4.83460302526130664e - 34, 0, 0, 6.30459929660781851e - 18, 7.57192526201145068e - 34, 5.03350698372840437e - 34, 9.28268079616618064e - 34, 0, 1.76723384525547359e - 17, 3.55686101822965701e - 34, 2.99413850083471346e - 34, 0.158143376337580827, 2.29601774161699833e - 19, 1.06106938611042947e - 18, 1.31968344319506391e - 18, 0.530902525044209539, 0, 2.89148310257773535e - 18, 3.34892126180666159e - 18, 0, 0.310999974151577319, 5.41244666317833561e - 05, 4.84993165246959553e - 16)$ . This solution is a little infeasible and no feasible solution is found so far.

i	$a_i$	$b_i$	$c_i$	$d_i$	$e_i$
1	0.0693	44.094	123.7	31.244	0.1
2	0.0577	58.12	31.7	36.12	0.3
3	0.05	58.12	45.7	34.784	0.4
4	0.2	137.4	14.7	92.7	0.3
5	0.26	120.9	84.7	82.7	0.6
6	0.55	170.9	27.7	91.6	0.3
7	0.06	62.501	49.7	56.708	
8	0.1	84.94	7.1	82.7	
9	0.12	133.425	2.1	80.8	
10	0.18	82.507	17.7	64.517	
11	0.1	46.07	0.85	49.4	
12	0.09	60.097	0.64	49.1	
13	0.0693	44.094			
14	0.0577	58.12			
15	0.05	58.12			
16	0.2	137.4			
17	0.26	120.9			
18	0.55	170.9			
19	0.06	62.501			
20	0.1	84.94			
21	0.12	133.425			
22	0.18	82.507			
23	0.1	46.07			
24	0.09	60.097			

Data set for test problem g20

### Function 21: g21 (Epperly T.)

**g21**

Minimize

$$f(\vec{x}) = x_1$$

subject to:

$$g_1(\vec{x}) = -x_1 + 35x_2^{0.6} + 35x_3^{0.6} \leq 0$$

$$h_1(\vec{x}) = -300x_3 + 7500x_5 - 7500x_6 - 25x_4x_5 + 25x_4x_6 + x_3x_4 = 0$$

$$h_2(\vec{x}) = 100x_2 + 155.365x_4 + 2500x_7 - x_2x_4 - 25x_4x_7 - 15536.5 = 0$$

$$h_3(\vec{x}) = -x_5 + \ln(-x_4 + 900) = 0$$

$$h_4(\vec{x}) = -x_6 + \ln(x_4 + 300) = 0$$

$$h_5(\vec{x}) = -x_7 + \ln(-2x_4 + 700) = 0$$

where the bounds are  $0 \leq x_1 \leq 1000$ ,  $0 \leq x_2, x_3 \leq 40$ ,  $100 \leq x_4 \leq 300$ ,  $6.3 \leq x_5 \leq 6.7$ ,  $5.9 \leq x_6 \leq 6.4$  and  $4.5 \leq x_7 \leq 6.25$ . The best known solution is at  $x^* = (193.724510070034967, 5.56944131553368433e-27, 17.3191887294084914, 100.047897801386839, 6.68445185362377892, 5.99168428444264833, 6.21451648886070451)$  where  $f(x^*) = 193.724510070035$ .

Function 22: g22 (Epperly T.)

g22

Minimize

$$f(\vec{x}) = x_1$$

subject to:

$$g_1(\vec{x}) = -x_1 + x_2^{0.6} + x_3^{0.6} + x_4^{0.6} \leq 0$$

$$h_1(\vec{x}) = x_5 - 100000x_8 + 1 \times 10^7 = 0$$

$$h_2(\vec{x}) = x_6 + 100000x_8 - 100000x_9 = 0$$

$$h_3(\vec{x}) = x_7 + 100000x_9 - 5 \times 10^7 = 0$$

$$h_4(\vec{x}) = x_5 + 100000x_{10} - 3.3 \times 10^7 = 0$$

$$h_5(\vec{x}) = x_6 + 100000x_{11} - 4.4 \times 10^7 = 0$$

$$h_6(\vec{x}) = x_7 + 100000x_{12} - 6.6 \times 10^7 = 0$$

$$h_7(\vec{x}) = x_5 - 120x_2x_{13} = 0$$

$$h_8(\vec{x}) = x_6 - 80x_3x_{14} = 0$$

$$h_9(\vec{x}) = x_7 - 40x_4x_{15} = 0$$

$$h_{10}(\vec{x}) = x_8 - x_{11} + x_{16} = 0$$

$$h_{11}(\vec{x}) = x_9 - x_{12} + x_{17} = 0$$

$$h_{12}(\vec{x}) = -x_{18} + \ln(x_{10} - 100) = 0$$

$$h_{13}(\vec{x}) = -x_{19} + \ln(-x_8 + 300) = 0$$

$$h_{14}(\vec{x}) = -x_{20} + \ln(x_{16}) = 0$$

$$h_{15}(\vec{x}) = -x_{21} + \ln(-x_9 + 400) = 0$$

$$h_{16}(\vec{x}) = -x_{22} + \ln(x_{17}) = 0$$

$$h_{17}(\vec{x}) = -x_8 - x_{10} + x_{13}x_{18} - x_{13}x_{19} + 400 = 0$$

$$h_{18}(\vec{x}) = x_8 - x_9 - x_{11} + x_{14}x_{20} - x_{14}x_{21} + 400 = 0$$

$$h_{19}(\vec{x}) = x_9 - x_{12} - 4.60517x_{15} + x_{15}x_{22} + 100 = 0$$

where the bounds are  $0 \leq x_1 \leq 20000$ ,  $0 \leq x_2, x_3, x_4 \leq 1 \times 10^6$ ,  $0 \leq x_5, x_6, x_7 \leq 4 \times 10^7$ ,  $100 \leq x_8 \leq 299.99$ ,  $100 \leq x_9 \leq 399.99$ ,  $100.01 \leq x_{10} \leq 300$ ,  $100 \leq x_{11} \leq 400$ ,  $100 \leq x_{12} \leq 600$ ,  $0 \leq x_{13}, x_{14}, x_{15} \leq 500$ ,  $0.01 \leq x_{16} \leq 300$ ,  $0.01 \leq x_{17} \leq 400$ ,  $-4.7 \leq x_{18}, x_{19}, x_{20}, x_{21}, x_{22} \leq 6.25$ . The best known solution is at  $x^* = (236.430975504001054, 135.82847151732463, 204.818152544824585, 6446.54654059436416, 3007540.83940215595, 4074188.65771341929, 32918270.5028952882, 130.075408394314167, 170.817294970528621, 299.924591605478554, 399.258113423595205, 330.817294971142758, 184.51831230897065, 248.64670239647424, 127.658546694545862, 269.182627528746707, 160.000016724090955, 5.29788288102680571, 5.13529735903945728, 5.59531526444068827, 5.43444479314453499, 5.07517453535834395)$  where  $f(x^*) = 236.430975504001$ .

Function 23: g23 (Xia Q.)

g23

Minimize

$$f(\vec{x}) = -9x_5 - 15x_8 + 6x_1 + 16x_2 + 10(x_6 + x_7)$$

subject to:

$$g_1(\vec{x}) = x_9x_3 + 0.02x_6 - 0.025x_5 \leq 0$$

$$g_2(\vec{x}) = x_9x_4 + 0.02x_7 - 0.015x_8 \leq 0$$

$$h_1(\vec{x}) = x_1 + x_2 - x_3 - x_4 = 0$$

$$h_2(\vec{x}) = 0.03x_1 + 0.01x_2 - x_9(x_3 + x_4) = 0$$

$$h_3(\vec{x}) = x_3 + x_6 - x_5 = 0$$

$$h_4(\vec{x}) = x_4 + x_7 - x_8 = 0$$

where the bounds are  $0 \leq x_1, x_2, x_6 \leq 300$ ,  $0 \leq x_3, x_5, x_7 \leq 100$ ,  $0 \leq x_4, x_8 \leq 200$  and  $0.01 \leq x_9 \leq 0.03$ . The best known solution is at  $x^* = (0.00510000000000259465, 99.99470000000000514, 9.01920162996045897e - 18, 99.99990000000000535, 0.000100000000027086086, 2.75700683389584542e - 14, 99.999999999999574, 2000.0100000100000100008)$  where  $f(x^*) = -400.055099999999584$ .

Function 24: g24 (Floudas C., 1999)

g24

Minimize

$$f(\vec{x}) = -x_1 - x_2$$

subject to:

$$g_1(\vec{x}) = -2x_1^4 + 8x_1^3 - 8x_1^2 + x_2 - 2 \leq 0$$

$$g_2(\vec{x}) = -4x_1^4 + 32x_1^3 - 88x_1^2 + 96x_1 + x_2 - 36 \leq 0$$

where the bounds are  $0 \leq x_1 \leq 3$  and  $0 \leq x_2 \leq 4$ . The feasible global minimum is at  $x^* = (2.329520197477623, 17849307411774)$  where  $f(x^*) = -5.50801327159536$ . This problem has a feasible region consisting on two disconnected sub-regions.