

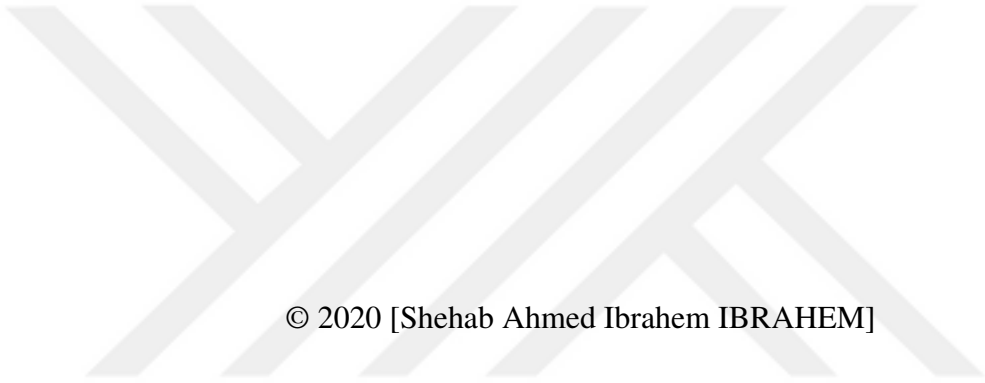
**T.C.
SÜLEYMAN DEMİREL UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES**

**NEW GLOBAL OPTIMIZATION TECHNIQUE BY USING
AUXILIARY FUNCTION METHOD IN DIRECTIONAL SEARCH
WITH COMPUTER APPLICATIONS**

Shehab Ahmed Ibrahim IBRAHEM

**Supervisor
Prof. Dr. Ahmet SAHINER**

**THE DEGREE OF DOCTOR OF PHILOSOPHY
DEPARTMENT OF MATHEMATICS
ISPARTA - 2020**



© 2020 [Shehab Ahmed Ibrahim IBRAHEM]

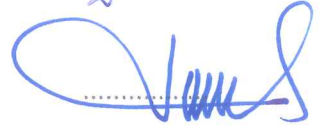
APPROVAL OF THE THESIS

New Global Optimization Technique By Using Auxiliary Function Method In Directional Search With Computer Applications submitted by **Shehab Ahmed Ibrahim IBRAHEM** in partial fulfillment of the requirements for **Doctor of Philosophy** in **Department of Mathematics**, Graduate School of Natural and Applied Sciences, Süleyman Demirel University by,

Supervisor **Prof. Dr. Ahmet SAHINER**
Süleyman Demirel University



Committee Member **Prof. Dr. Kenan TAŞ**
Çankaya University



Committee Member **Prof. Dr. S. Zeynep ALPARSLAN GÖK**
Süleyman Demirel University



Committee Member **Assist. Prof. Dr. Gaber FAISEL**
Süleyman Demirel University



Committee Member **Assist. Prof. Dr. Dumitru BALEANU**
Çankaya University



Director **Assoc. Prof. Dr. Şule Sultan UĞUR**
.....

COMMITMENT

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Shehab Ahmed Ibrahim IBRAHEM



TABLE OF CONTENTS

	Page
TABLE OF CONTENTS.....	i
ABSTRACT	ii
ÖZET	iii
ACKNOWLEDGMENTS	iv
LIST OF FIGURES	v
LIST OF TABLES	vi
LIST OF SYMBOLS AND ABBREVIATION	vii
1. INTRODUCTION	1
1.1. Global Optimization Formula.....	2
1.2. Global Optimization Classification.....	3
1.3. Objective Function Form	4
1.4. Various Methods for Solving Global Optimization Problems.....	7
1.4.1. Stochastic methods	7
1.4.2. Heuristic methods	8
1.4.3. Deterministic methods.....	8
1.5. Thesis Overview	10
2. PRELIMINARIES AND TERMINOLOGIES.....	12
3. NEW SMOOTHING TECHNIQUE WITH APPLICATION IN GLOBAL OPTIMIZATION	14
3.1. Global Smoothing Technique	14
3.2. Global Optimization Technique	18
3.3. Algorithm	22
4. GLOBAL OPTIMIZATION TECHNIQUE IN A DIRECTIONAL SEARCH	23
4.1. Auxiliary Function Method in Global Optimization.....	23
4.2. One-dimensional Minimization Problem	26
4.3. Algorithm	28
5. INCREASING THE EFFECTS OF FILLED FUNCTION IN GLOBAL OPTIMIZATION	30
5.1. Overview of the Filled Function Method	31
5.1.1. Filled Function Methods with Two-parameter	31
5.1.2. Filled Function Methods with One-parameter	32
5.2. New Filled Function and Its Properties.....	34
5.3. Algorithm	37
6. APPLICATION FOR GLOBAL OPTIMIZATION ALGORITHMS	39
6.1. Test Problems	39
6.2. Applications on Test problems	46
7. TOTAL VARIATION APPLICATION IN IMAGE DENOISING	56
7.1. Theoretical part	57
7.2. Smoothed TV-function for Denoising	58
7.3. Algorithm	59
8. Conclusion.....	66
REFERENCES.....	67
CURRICULUM VITAE	75
ÖZGEÇMİŞ.....	77

ABSTRACT

Doctor of Philosophy

NEW GLOBAL OPTIMIZATION TECHNIQUE BY USING AUXILIARY FUNCTION METHOD IN DIRECTIONAL SEARCH WITH COMPUTER APPLICATIONS

Shehab Ahmed Ibrahim IBRAHEM

**Süleyman Demirel University
Graduate School of Natural and Applied Sciences
Department of Mathematics**

Supervisor: Prof. Dr. Ahmet SAHINER

Global optimization has been applied to a wide range of problems related to science and engineering design. Although numerous global optimization techniques have been developed and studied for decades, the results some times are not satisfactory when used in design or complex systems, so there is still an area for developing these techniques to be more accurate, fast, and easy to implement.

The difficult issues for global optimization are how to escape from the current local minimizer and find a lower minimizer of the objective function, then how to evaluate the convergence to the global minimizer and determine the stopping criteria. The purpose of this thesis is to solve and avoid the above difficulties by developing and demonstrate deterministic methods for unconstrained global optimization problems.

The main part of this thesis contains three Sections 3, 4, and 5. Section 3, a new global smoothing approximation technique is proposed for non-smooth functions and use the same technique to construct a smoothing auxiliary function for solving unconstrained global optimization problems. Section 4 presents a new auxiliary function to solve global optimization problems by converting a multi-dimensional problem into a one-dimensional problem and decreasing the number of local minimizers and then finding the global minimizer of the one-dimensional problem, then finding the global minimizer of the multi-dimensional function by using a new algorithm. Section 5 a new filled function is proposed for finding a better minimizer of smooth unconstrained global optimizations and the proposed filled function is continuously differentiable and contains two parameters.

The applications part of this thesis includes two Sections, Section 6 presents results to a set of common test problems for the proposed algorithms in Sections 3, 4 and 5. Image processing is done as a real-life problem, the problem is solved by using the algorithm introduced in Section 3, and numerical results are shown in Section 7.

Keywords: Global optimization, Smoothing technique, Auxiliary function, Directional search.

2020, 78 pages

ÖZET

Doktora Tezi

YARDIMCI FONKSİYON YÖNTEMİNİ YÖNLÜ ARAMA İLE KULLANARAK YENİ BİR GLOBAL OPTİMİZASYON TEKNİĞİ VE BİLGISAYAR UYGULAMALARI

Shehab Ahmed Ibrahim IBRAHEM

Süleyman Demirel Üniversitesi
Fen Bilimleri Enstitüsü
Matematik Anabilim Dalı

Danışman: Prof. Dr. Ahmet ŞAHİNER

Global optimizasyon, bilim ve mühendislik ile ilgili çok çeşitli problemlere uygulanmıştır. Çok sayıda küresel optimizasyon tekniği on yıllardır incelenmiş ve geliştirilmiş olmasına rağmen bu teknikler karmaşık sistemlerde kullanıldığında bazen tatmin edici sonuçlar sunabilmiş değildir. Bu nedenle bu tekniklerden daha doğru sonuçlar elde edilmesi, hızlı ve kolay uygulanabilmesi için daha da geliştirilmesi gerekmektedir.

Global optimizasyonda zor olan konular; mevcut yerel minimumlaştırıcıdan nasıl kurtulunacağı, objektif fonksiyonun daha düşük bir minimumlaştırıcının nasıl bulunacağı, daha sonra küresel minimumlaştırıcıya yakınlaşmanın nasıl değerlendirileceği ve durdurma kriterlerinin nasıl belirleneceği şeklindedir. Bu tezin amacı, kısıtsız küresel optimizasyon problemleri için yukarıdaki zorlukları aşacak şekilde deterministik yöntemler geliştirmektir.

Bu tezin ilk kısmı üç bölüm 3, 4 ve 5 içerir. Bölüm 3'te, düzgün olmayan fonksiyonlar için yeni bir global düzünleştirme tekniği önerilmektedir. Aynı teknik, kısıtlanmamış global optimizasyon problemlerini çözmek için geliştirilen yardımcı fonksiyonu oluşturmak için kullanılmaktadır. Bölüm 4'te, çok boyutlu problemi tek boyutlu bir probleme dönüştürerek ve yerel minimumlaştırıcı sayısını azaltarak yeni bir yardımcı fonksiyon metodu geliştirilmiştir. Tek boyuta indirgenen problemin global minimumlaştırıcısı bulunup ardından çok boyutlu probleme lokal minimum bulma algoritması yardımıyla orjinal problemin global minimumlaştırıcısı bulunur. Bölüm 5'te, düzgün kısıtsız global optimizasyon problemlerinin daha alt bir minimumlaştırıcısını bulmak için yeni bir doldurulmuş fonksiyon önerilmiştir. Önerilen doldurulmuş fonksiyon türevlenebilir yapıdadır ve iki parametre içerir.

Bu tezin uygulama kısmı iki bölümden oluşmaktadır. Bölüm 6'da bölümler 3, 4 ve 5'teki önerilen algoritmaları bir dizi test problemine uygulanmasından elde edilen sonuçlar sunulmuştur. Bölüm 7'de, görüntü işleme problemi bir gerçek yaşam problemi olarak ele alınmış ve 3. bölümde tanıtılan algoritma bu probleme uygulanarak nümerik sonuçlar sunulmuştur.

Anahtar Kelimeler: Global optimizasyon, Düzgünleştirme tekniği, Yardımcı fonksiyon, Yönlü arama

2020, 78 sayfa

ACKNOWLEDGMENTS

I thank almighty Allah for giving me strength and determination to do my dissertation. Now I would like to express my deep thanks and appreciation to my advisor Prof. Dr. Ahmet Sahiner. I have had the wonderful opportunity to interact with him for years and he is an exceptional mentor and professor with excellent expertise. I thank him for all his encouragement, commitment, patience, and support pending my doctoral study. It was also a pleasure to have had Dr. Nurullah Yilmaz, I would like to thank him for introducing me to key researchers relevant to my area of interest, and for the various advice and bright ideas provided during my study. Thank you Dr. Gaber Faisel and Dr. Dumitru Baleanu, members of my committee, for sharing their insights and inspiring me through these years; their support has been absolutely invaluable.

I am very happy to be one of the students at the wonderful University of Süleyman Demirel. I sincerely thank the entire faculty and staff members in the mathematics department for all they've done to help me, it has been a pleasure to share my life with these wonderful people.

I am very grateful to my university (Kirkuk University), which gave me the scholarship to cover my educational expenses. Many thanks to all my wonderful friends here in Turkey or in my country (Iraq) for their endless support and encouragement. Thank you to my friend Idris A. Abdulhamid for sharing his knowledge with me.

I am very thankful to my family who has given me the strength to continue through all this time with their enormous love and support. Thank you for never letting me fall.

Shehab Ahmed Ibrahim IBRAHEM
ISPARTA, 2020

LIST OF FIGURES

	Page
Figure 1.1	Continuous and discontinuous functions..... 5
Figure 1.2	Convex and non-convex functions 6
Figure 1.3	Smooth and non-smooth functions 11
Figure 3.1	The graph of $\tilde{\omega}_\sigma(t)$ (green and solid) and $\omega(t)$ (black and dot) a, b, c and d with $\sigma = 0.8, 0.6, 0.4, 0.2$ respectively. 16
Figure 3.2	The graph of $\varphi(x)$ (blue and solid) and $\tilde{\varphi}_\sigma(x)$ (green and dot) a and b with $\sigma = 0.4, 0.8$ respectively. 17
Figure 3.3	(a) The graph of $\varphi(x, y)$, (b) The graph of $\tilde{\varphi}_\sigma(x, y)$, (c) The contour graph of $\varphi(x, y)$ and (d) The contour graph of $\tilde{\varphi}_\sigma(x, y)$ 18
Figure 3.4	The graph of $f(x)$ and $\phi(x, x_k^*)$ before smoothing it, where x_k^* is the current minimizer. 19
Figure 3.5	After smoothing, the graph of $f(x)$ and $\tilde{\phi}_{\sigma, \mu}(x, x_k^*)$, where x_k^* is the current minimizer. 21
Figure 4.1	The one-dimensional problem..... 28
Figure 5.1	Some different values of the parameter μ and their effect on the function $F(x, x_k^*)$ 35
Figure 5.2	The shape of the functions $F(x, x_k^*)$ and $f(x)$ in two dimensions. .. 35
Figure 7.1	The graph of $\phi(u)$ (black and solid) and $\tilde{\phi}_\sigma(u)$ (green and dashed) with deferent σ 58
Figure 7.2	Test images (a) Cameraman, (b) Barbara, (c) Lena and (d) Pepper 60
Figure 7.3	Smoothed TV-function gradient. 61
Figure 7.4	The denoising experiment with Cameraman image. (a) Noisy image. (b) TVF. (c) SA-DCT. (d) DNSCT. (e) GTV. (f) NSTV. 62
Figure 7.5	The denoising experiment with Barbara image. (a) Noisy image. (b) TVF. (c) SA-DCT. (d) DNSCT. (e) GTV. (f) NSTV. 63
Figure 7.6	The denoising experiment with Lena image. (a) Noisy image. (b) TVF. (c) SA-DCT. (d) DNSCT. (e) GTV. (f) NSTV. 64
Figure 7.7	The denoising experiment with Peppers image. (a) Noisy image. (b) TVF. (c) SA-DCT. (d) DNSCT. (e) GTV. (f) NSTV. 65

LIST OF TABLES

	Page
Table 6.1 The list of test problems	47
Table 6.2 The results of NSA algorithm on problems 1-49	49
Table 6.3 The results of DSA algorithm on problems 1-49	50
Table 6.4 The results of FSA algorithm on problems 1-49	51
Table 6.5 Iteration, function evaluations and execution time for NSA, DSA, and FSA algorithms.	52
Table 6.6 Comparison NSA algorithms with the algorithm in (Bagirov et al. (2009)).	53
Table 6.7 Comparison DSA algorithms with the algorithm in (Wei et al. (2014))	54
Table 6.8 Comparison FSA algorithms with the algorithm in (Sahiner et al. (2019)).	55
Table 7.1 The PSNR value by various methods (unit: dB).	61

LIST OF SYMBOLS AND ABBREVIATION

x_k^*	: k – th Local minimizer
x^*	: Global minimizer
x_{int}	: Start point
\mathbb{R}	: The real numbers
\mathbb{R}^n	: Euclidean n -space
f	: The objective function $f(x)$ with $\mathbb{R}^n \rightarrow \mathbb{R}$ maps
\mathbb{D}	: Domain of the objective function $f(x)$
d	: Direction
n	: Dimension of $f(x)$
B_k^*	: The basin of x_k^*
$\ \cdot\ $: Euclidean norm
$\nabla f(x)$: The gradient of $f(x)$
σ, β	: Smoothing parameters
α, μ	: Parameters

1. INTRODUCTION

Optimization is a process to obtain optimal value (minimal or maximal) for a given problem. Problems can vary from optimization problems with simple objective functions to problems with complex objective functions that include various properties and multiple equations. Optimization can be a local search (or global). Local search finds the optimal value within the neighborhood set of a candidate solution while global search references to locating the optimal value of a specific objective function in the whole domain.

Many real-life problems have been formulated as global optimization problems. They have been applied in many branches of science such as engineering (e. g., architecture, bridge design, dam design, and rocket design), computer and information technology (e.g., artificial intelligence, database design, image processing, and network design), economics (e.g., business prediction and electronic commerce), agriculture (e. g., agricultural production structure, and spatial distribution of agricultural crops), chemistry, geography (e.g., weather and earthquake prediction), physics (e.g., nanotechnology, and metal matrix composition), etc, or even in social life such as vacation planning.

Global optimization problems become more and more complicated from year to year due to the increase in the number variables and the structure of the problems (non-smoothness). The existence of many local minimizers of objective function makes global optimization complicated. Furthermore, the major difficulties for global optimization are listed bellow:

- a- When finding a local minimizer with the help of any local minimization method, and how to override the current local minimizer.
- b- How to ignore the local minimizers of which their values are greater than the value of current minimizer and find a lower minimizer of the objective function.
- c- How to evaluate the convergence to the global minimizer and, determine the stopping criteria.

Global optimization can not always guarantee to gain a global solution for a given set of computational sources; therefore, approximations are vital. sure global optimization algorithms cater to precise problems and this has caused a range of evolved global optimization algorithms. commonly, the focal point of a global optimization algorithm

is to achieve a suitable result in a suitable amount of computational time; but, developing new approaches to solve a selected problem is difficult, and other strategies may be shown to be better. In recent years many sciences depend on global optimization to solve many problems. Global optimization problems had existed since ancient times with many sciences and these problems were ignored until the 1970s. There are many reasons to ignore the global optimization problems before this date, perhaps one of the reasons is that the theories and local optimization methods were not discovered or ready at that time as well as the large computational complications inherent in global optimization problems could be one of the other reasons. There were no proper controls and instructions before the 1970s. After that date, the evolution that took place in computer technologies made problems that had no solution in the past are solvable nowadays. Since that time, the field of global optimization is in the growth and expansion. This Section introduces general information for a global optimization problem, problem properties, preliminaries, and pertinent global optimization methods to solve the problem. Part 1.1 defines a formula of the global optimization problem. Part 1.2 presents a classification of global optimization. Part 1.3 describes a formula of the objective function. Part 1.4 discusses the various methods for solving global optimization. Part 1.5 overview of dissertation.

1.1. Global Optimization Formula

Global optimization is a search process to locate a global minimizer of a function. Generally, the formula of global optimization can be as the following:

$$(P) : \min_{x \in \mathbb{D}} f(x) \tag{1.1}$$

where f is referred to as the objective function of the problem and $\mathbb{D} \subset \mathbb{R}^n$ is the feasible domain of vectors $x = (x_1, x_2, \dots, x_n)$ and which is subject to specific constraints. The solution of problem (P) is to find the global minimizer value x^* , that means

$$f(x^*) = \min_{x \in \mathbb{D}} f(x).$$

There are many new theories and algorithms that have been presented for solving problem (P) . In general, methods of global optimization can be classified into three

main classes: stochastic methods, heuristic methods and deterministic methods.

1.2. Global Optimization Classification

The global optimization problem is subject to two concepts to define it. These concepts are the objective function f and the domain \mathbb{D} (search area). The problem (P) is defined as unconstrained global optimization if the whole domain \mathbb{D} is as a search area to find the global minimizer x^* , that means

$$f(x^*) \leq f(x),$$

for all $x \in \mathbb{D}$. The problem (P) is defined as a constrained global optimization if the minimization of the objective function subject to constraints. Generally, the formula of these problems is

$$\min_{x \in \mathbb{R}^n} f(x),$$

bound by

$$\begin{cases} g_i(x) = 0, & i \in A_1 \\ g_i(x) \geq 0, & i \in A_2, \end{cases}$$

where f and functions $g_i(x)$ are twice continuously differentiable and the sets A_1, A_2 are finite. As previously, f is called the objective function, while

$$g_i(x) = 0, \quad i \in A_1,$$

are equality constraints and

$$g_i(x) \geq 0, \quad i \in A_2,$$

are inequality constraints. So, the set D_c which satisfies the above constraints is called the feasible set, that is

$$D_c = \{x / g_i(x) = 0, \quad i \in A_1; g_i(x) \geq 0, \quad i \in A_2\}.$$

The function $f : D_c \rightarrow R$ with the set $D_c \subset \mathbb{R}^n$ is known as a constrained global optimization problem and it can be written as follows

$$\min_{x \in D_c} f(x).$$

In this thesis, we consider the problem (P) as an unconstrained global optimization problem.

1.3. Objective Function Form

As mentioned before, a global optimization problem subjects to two concepts to define it, search space(domain) and objective function. In general, the objective function can be classified into four main parts based on the properties of function f as follows:

- Continuity
- Convexity
- Differentiability
- Smoothness

Continuity: An objective function $f : \mathbb{D} \rightarrow \mathbb{R}$ is called continuous at a vector $x = c \in \mathbb{D}$ if

$$\lim_{x \rightarrow c} f(x) = f(c),$$

that means the objective function is called continuous if it is continuous at each point of its domain. In other words, a function $f(x)$ is continuous at a vector $x = c$ if it satisfies the following conditions

- $f(c)$ is defined.
- $\lim_{x \rightarrow c} f(x)$ exists.
- $\lim_{x \rightarrow c} f(x) = f(c)$.

If any of the above conditions are not satisfied then we say f is discontinuous (see Fig. 1.1).

Convexity: An objective function $f : \mathbb{D} \rightarrow \mathbb{R}$ is convex on a set $\mathbb{D} \subset \mathbb{R}^n$ if for any $x_1, x_2 \in \mathbb{D}$ and $\lambda \in (0, 1)$ then

$$f(\lambda x_1 + (1 - \lambda)x_2) \leq \lambda f(x_1) + (1 - \lambda)f(x_2).$$

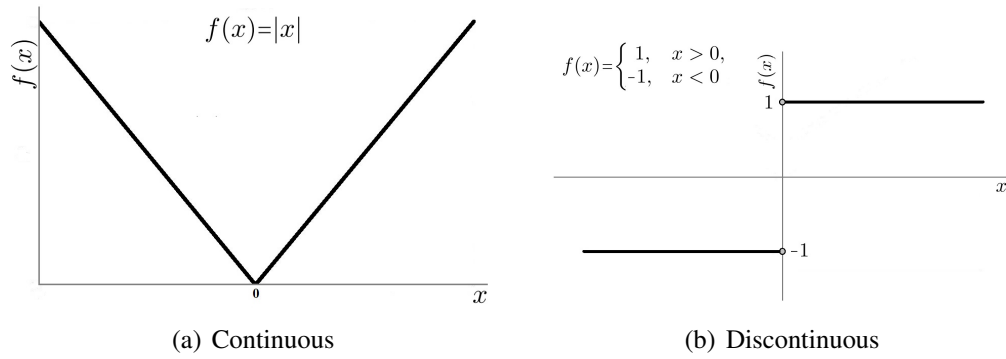


Figure 1.1. Continuous and discontinuous functions

If this inequality does not hold, the function f is non-convex, as shown in Fig. 1.2. If the objective function is convex, the local optimal point is also global optimal point, therefore only one of the local methods is required to solve the problem. However, if the objective function is non-convex, that means there are more than one local extrema, and the search of the global optimal point needs the whole domain, thus the local methods are not enough to find the global optimal point, in this case, the global methods are needed to solve the problem.

Differentiability: The objective function $f : \mathbb{D} \rightarrow \mathbb{R}$ is said to be differentiable at a vector $c \in \mathbb{D}$ if its derivative exists at c . The gradient direction can be used to determine information about the local minimizer, maximizer and stationary point in \mathbb{D} . This information about gradient can be used in optimization solutions and finding the minimizer faster. There are many methods relied on the gradient to find a local minimizer such as Newton's method and gradient-descent. Furthermore, first and second-order optimality conditions are results that give us some structural information about the properties of optimal solutions. For example, if x_k^* is a local minimizer of a twice continuously differentiable function $f : \mathbb{D} \rightarrow \mathbb{R}$ then the first-order condition must have

$$\nabla f(x_k^*) = 0.$$

The second-order condition explains that if $x_k^* \in \mathbb{D}$ is a local minimizer of f , then $\nabla f(x_k^*) = 0$ and

$$\nabla^2 f(x_k^*) \geq 0,$$

that means the Hessian at x_k^* is positive semi-definite. Some global optimization methods use the gradient information of the objective function to find the local minimizer value.

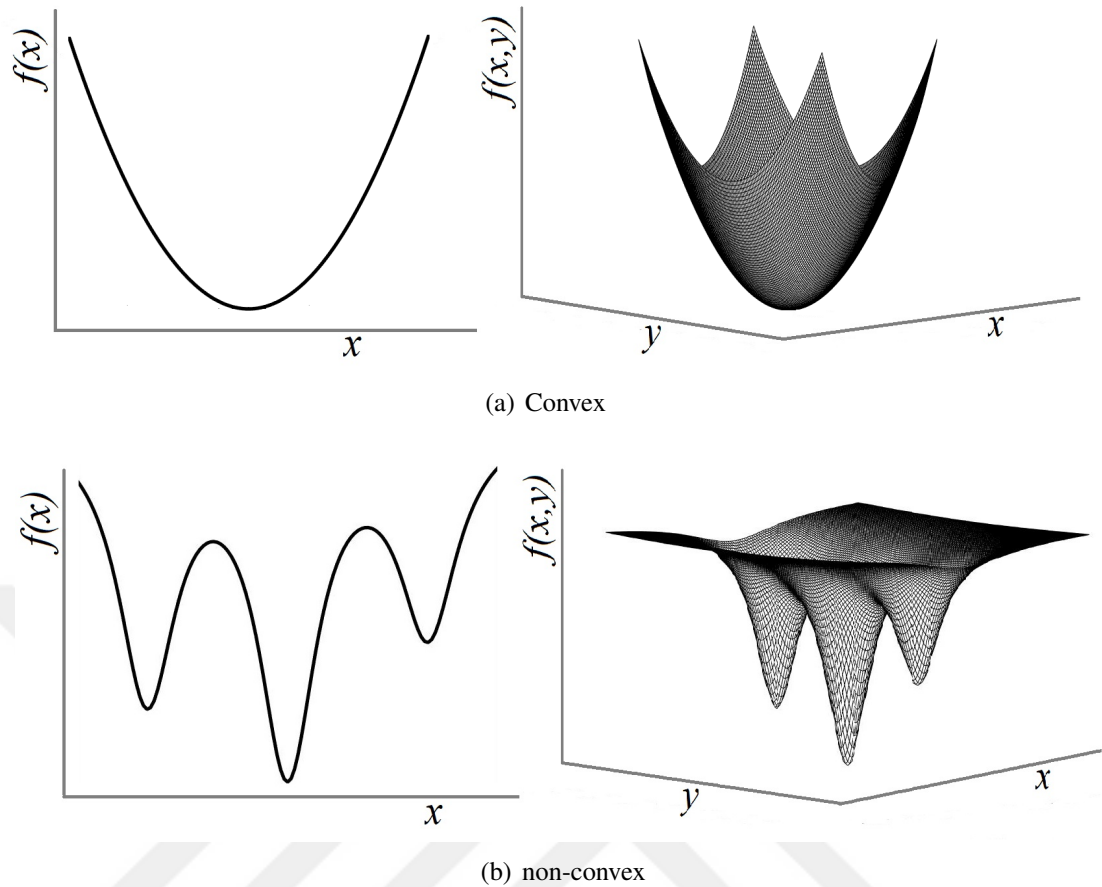


Figure 1.2. Convex and non-convex functions

These methods are called deterministic methods which are using the gradient to find the local minimizer of the objective function and then the global minimizer value with the help of the auxiliary function, which we will mention later in detail.

Smoothness: The objective function $f : \mathbb{D} \subset \mathbb{R}^n \rightarrow \mathbb{R}$ is said to be a smooth function if it is continuously differentiable at all points of domain \mathbb{D} , and if it has continuous derivatives in \mathbb{D} up to a known limit k , then we say that this function is \mathbb{D}_k smooth. If $k = 2$, then the objective function is called a twice continuously differentiable, and that is what we need in optimization problems. On the other hand, if the objective function f is non-differentiable (no matter it is continuous or discontinuous) and then it is called non-smooth. A simple example of a non-smooth function is the absolute value, which is continuous at all domain points but is not differentiable. Fig. 1.3 shows the difference between smoothness and non-smoothness for a set of functions.

In optimization problems, it is easy to find the local extremum when the objective function is smooth and then to find the global optimal solution, but there is a real problem

when the objective function is a non-smooth no matter this function is discontinuous or continuous. In this thesis, we introduce new smoothing techniques to make the objective function and auxiliary function, if necessary, differentiable and smooth to be used in global optimization.

In recent years, non-smooth functions have received considerable attention. Smoothing studies try to make the objective function continuously differentiable before the optimization process. The smoothing process focuses on finding the correct modifications for $f(x)$ that makes it simple and easy to minimize. In the approximation, adjustable parameters are called smoothing parameters that provide the control. In general, smoothing techniques can be classified into two main classes such as local and global smoothing techniques. Smoothing techniques belonging to the first class aim to smooth the objective function locally in a neighborhood of a point at which the objective function is non differentiable (Bertsekas, 1975; Zang, 1980; Chen and Wan, 2015; Sahiner et al., 2018). Smoothing techniques belonging to the second class use the whole domain to approximate the function f globally. However, many important studies related to global smoothing technique have been published by (Xu, 2001; Xavier, 2010; Xiao and Yu, 2010).

1.4. Various Methods for Solving Global Optimization Problems

In recent decades, global optimization problems have received more attention because of their importance in many significant applications, therefore the variety of these methods it easy for us to locate the global optimal value. These methods can generally be divided into three groups which are stochastic, heuristic and deterministic.

1.4.1. Stochastic methods

Stochastic methods use probabilistic tools by generating many points randomly distributed over the feasible domain and use some local search methods from some of these points to converge to the global value. These methods are quite simple, very efficient in black box problems and robust with respect to the increasement of the dimension of the problem but some of the stochastic methods can find only a local solution instead of the global one. The very well-known stochastic approaches are Random search and

Adaptive search, Markovian algorithms, and etc. (Anderssen and Bloomfield, 1975; Huyer and Neumaier, 1999; Zhigljavsky and Zilinskas, 2007; Schäffler, 2012). The population algorithms included stochastic methods but we handle them in the heuristic methods.

1.4.2. Heuristic methods

The heuristic methods are based on the simulation of the biological, physical or chemical processes. These methods are easy applicable and they converge the solution rapidly. However, they can give different results if they are run again. The very well-known methods are Genetic algorithm (Storti et al., 2015), Simulated Annealing algorithm (Suman and Kumar, 2006; Ekren and Ekren, 2010; Samora et al., 2016), Particle Swarm Optimization (Poli et al., 2007; Kennedy, 2010) and Artificial Bee Colony algorithm (Karaboga and Basturk, 2007; Akay and Karaboga, 2012). In recent years, the hybridization of the heuristic global optimization algorithms has come into prominence (Niknam et al., 2009; Mahi et al., 2015; Zheng et al., 2015; Garg, 2016; Liu et al., 2016).

1.4.3. Deterministic methods

In general, stochastic and heuristic methods are easy and fast in the application, but they have no guarantees in obtaining the global optimal solution. Deterministic methods are more effective and reliable, but they need a high cost of computation when compared to the previous methods. Some methods that use deterministic strategy can be divided into:

Branch and bound methods:- The main idea of these methods is to divide the feasible region into several partitions and then dismiss some non-prospective partitions by bounding the objective function at these partitions using predestined lower bounds. Then, the search is in the subregions remaining and treated the same way by dividing them into several smaller partitions based on the lower bounds. There are various methods for evaluating the lower bounds of the objective functions grant different branch and bonding methods. For instance, the primal-dual branch bound approaches used to grant lower boundaries of primal-dual approaches, which was introduced by (Floudas and Visweswaran, 1990, 1993; Androulakis et al., 1995; Adjiman et al., 1998),

whereas the interval branch and bound methods are used interval arithmetic instead, which was presented by (Alefeld and Herzberger, 2012; Hansen and Walster, 2003; Grimstad and Sandnes, 2016).

Auxiliary function methods:- These methods are also called the function modification methods and include all methods that depend on some appropriate modifications of the objective function to override from the current local minimizer to a better solution. In general, the steps work of the auxiliary function can be described as follows:

- 1- **(Initialization)** Minimizing objective function by using a random point as a starting point with the help of any local optimization method to find a local minimizer of the objective function.
- 2- **(Local search)** Construct an auxiliary function based on the current objective function minimizer, and use any point near this point to minimize the auxiliary function. Consequently, an auxiliary function minimizer is obtained. This minimizer will lie in a basin of a good solution to the objective function.
- 3- **(Global search)** Use the minimizer of the auxiliary function obtained in step 2 as a starting point to minimize the objective function and find a better minimizer of f .

Repeating steps 2 and 3 will definitely reduce the number of minimizers and find the global minimizer of the objective function.

Auxiliary function methods are promising and important methods in global optimization. These methods are developed according to deterministic search strategies by constructing an auxiliary function to escape from the current local minimizer to better one. Tunneling methods (Levy and Montalvo, 1985), filled function method (Ge and Qin, 1987; Renpu, 1990; Liu, 2001), and global descent method (Wu et al., 2011) are among these methods.

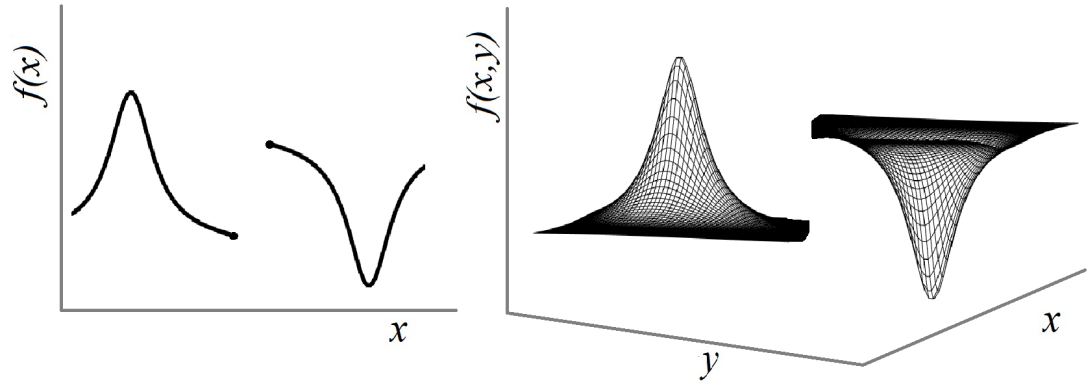
The first auxiliary function method was introduced by (Levy and Montalvo, 1985). (Cetin et al., 1993) developed the tunneling algorithm to resolve constrained global optimization problems. However, many important studies related to tunneling algorithms have been published in (Groenen and Heiser, 1996; Chowdhury et al., 2000; Xu et al., 2015).

Among other methods, the filled function approach can be considered an effective approach to solving multi-model global optimization problems, so it seems to have

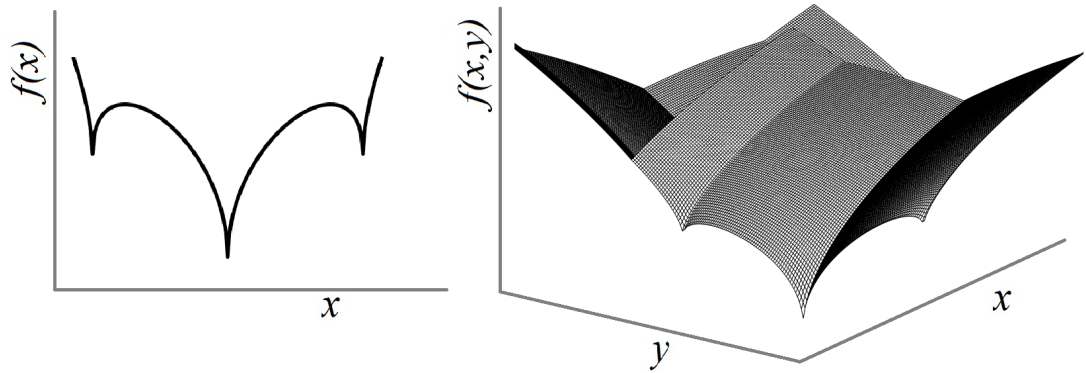
several features over others, for example, finding a better local minimizer sequentially compared to other methods is easier. The filled function method was first presented (Ge and Qin, 1987; Renpu, 1990), and improved by (Xu et al., 2001; Wu et al., 2005, 2007). Many valuable studies have been presented in order to make the filled function applicable for different type of problems such as non-smooth problems (Zhang et al., 2009; Sahiner et al., 2012), constrained optimization problems (Wang et al., 2015), system of nonlinear equations (Yuan et al., 2016b) and (Wei et al., 2014; Lin et al., 2014). Recently, the next generation of filled function or auxiliary function approaches have been developed (Yilmaz and Sahiner, 2017; Sahiner et al., 2017; Lin et al., 2018; Liu et al., 2017; Sahiner and Ibrahim, 2019). In this thesis, we introduce some auxiliary functions to solve multi-model for global optimization problems in different locations. Moreover, there exists important methods depend on deterministic strategy such as Covering methods (Jones et al., 1993), Space Filling Curve methods (Lera and Sergeyev, 2015; Ziadi et al., 2016) and other methods (Basso, 1982).

1.5. Thesis Overview

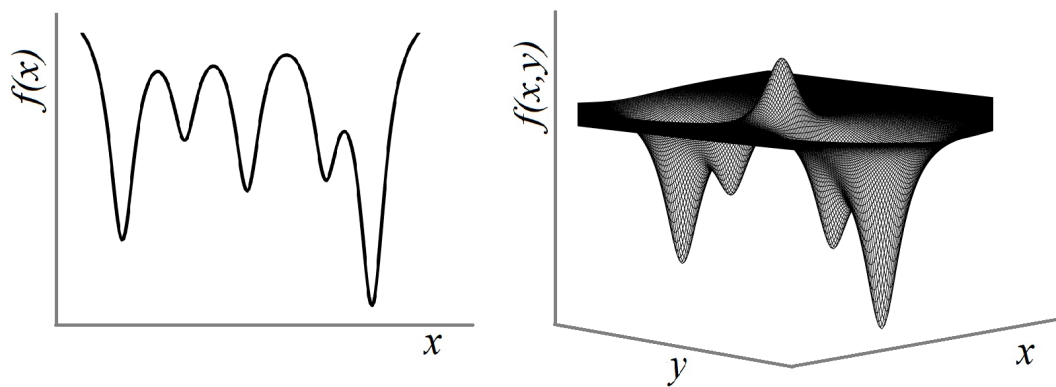
The rest of the thesis is arranged as follows. Section 2 provides some definitions and assumptions regarding the (P) problem. Section 3 presents a new global smoothing technique for non-smooth functions and uses the same technique to construct a smoothing auxiliary function for solving the problem (P). Section 4 describes a new algorithm to solve the (P) problem by converting a multi-dimensional problem into one-dimensional problem partitions and using an auxiliary function to reducing the minimizers number and finding a global minimizer for each partition. Then use these partitions to find the global minimizer of a multidimensional problem. Section 5 documents a new filled function method with two parameters to solve the problem (P). This new proposal is based on putting many stationary points in the lower basin. Section 6 summarizes the results for algorithms in Sections 3, 4, and 5. Section 7 uses the method defined in Section 3 to make TV-function differentiable, and then use it as an application for image denoising. Finally, Section 8 introduces the conclusions.



(a) non-smooth(discontinuous and non-differentiable)



(b) non-smooth(continuous and non-differentiable)



(c) smooth(continuous and differentiable)

Figure 1.3. Smooth and non-smooth functions

2. PRELIMINARIES AND TERMINOLOGIES

Throughout the thesis, we need some definitions and assumptions of the problem (P) defined in 1.1 as follows:

Assumption 2.1. \mathbb{D} is closed bounded and box-shaped domain, $\mathbb{D} = [l_b, u_b] = \{x : l_b \leq x \leq u_b, l_b, u_b \in \mathbb{R}^n\}$ and contains all the local minimizers of f , so we define the sets \mathbb{D}_1 and \mathbb{D}_2 as $\mathbb{D}_1 = \{x \in \mathbb{D} | f(x) \geq f(x_k^*), x \neq x_k^*\}$ and $\mathbb{D}_2 = \{x \in \mathbb{D} | f(x) < f(x_k^*)\}$.

Assumption 2.2. The number of different values of local minimizers of the function f must be finite.

Assumption 2.3. $f : \mathbb{D} \rightarrow \mathbb{R}^n$ is coercive, that means $f(x) \rightarrow +\infty$ as $\|x\| \rightarrow +\infty$.

Definition 2.4. Let $x \in \mathbb{D}$ and $\varepsilon > 0$, the ball

$$N_\varepsilon(x) = \{y : \|y - x\| < \varepsilon\}$$

is called an ε -neighborhood of x .

Definition 2.5. A set $\mathbb{D} \subset \mathbb{R}^n$ is called bounded set if it can be contained within a ball of finite radius.

Definition 2.6. A set $\mathbb{D} \subset \mathbb{R}^n$ is called compact set if it is closed and bounded.

Definition 2.7. Let $\mathbb{D} \subset \mathbb{R}^n$. A point $x^* \in \mathbb{D}$ is said to be a global minimizer of the objective function f if $f(x^*) \leq f(x)$ for all $x \in \mathbb{D}$. If $x_k^* \in \mathbb{D}$ and if there is a neighborhood $N_\varepsilon(x_k^*)$ of x_k^* such that $f(x_k^*) \leq f(x)$ for all $x \in N_\varepsilon(x_k^*)$ then x_k^* is said to be a local minimizer. In a similar way, if $f(x_k^*) < f(x)$ for all $x \in N_\varepsilon(x_k^*) \setminus x_k^*$, then x_k^* is said to be a strict local minimizer.

Definition 2.8. : Suppose that x_k^* is a local minimizer of f over \mathbb{D} . A set $B_k^* \subset \mathbb{D}$ is said to be a basin of f at the point x_k^* if any local minimization approach starting from any point in B_k^* finds local minimizer x_k^* .

Any local minimizer x_{k+1}^* of f is higher (or lower) than x_k^* if $f(x_{k+1}^*) \geq f(x_k^*), f(x_k^*) \geq f(x_{k+1}^*)$, then the basin B_{k+1}^* is said to be higher (lower) than B_k^* .

Definition 2.9. Suppose that $h : \mathbb{R}^n \rightarrow \mathbb{R}$ is a continuous function. We call the function $\tilde{h} : \mathbb{R}^n \times \mathbb{R}_+ \rightarrow \mathbb{R}$ a smoothing function of $h(x)$, if $\tilde{h}(\cdot, \beta)$ is continuously differentiable in \mathbb{R}^n for any constant β , and for any $x \in \mathbb{R}^n$,

$$\lim_{z \rightarrow x, \beta \rightarrow 0} \tilde{h}(z, \beta) = h(x).$$

Definition 2.10. The auxiliary function $F(x, x_k^*)$ is said to be a filled function of $f(x)$ at a local minimizer x_k^* if the function $F(x, x_k^*)$ has the following properties:

- x_k^* is a local maximizer of the function $F(x, x_k^*)$;
- $F(x, x_k^*)$ has no stationary points in the region $\mathbb{D}_1 = \{x \in \mathbb{D} | f(x) \geq f(x_k^*), x \neq x_k^*\}$;
- if x_k^* is not a global minimizer of f , then the function $F(x, x_k^*)$ does have a minimizer in the region $\mathbb{D}_2 = \{x | f(x) < f(x_k^*), x \in \mathbb{D}\}$.

3. NEW SMOOTHING TECHNIQUE WITH APPLICATION IN GLOBAL OPTIMIZATION

The main idea in smoothing is to use smooth functions to approximate a non-smooth objective function. The control is provided in the approximation by adjustable parameters, called smoothing parameters. In recent years, non-smooth functions have been attracting significant attention. Smoothing studies aim to make the objective function continuously differentiable prior to the optimization process. The smoothing methods focus on finding appropriate modifications to smooth the objective function f and creating a smoothing function without difficulties and minimizing it easily. Bertsekas (Bertsekas, 1975) suggests a primary review of the smoothing methodologies. This Section presents a new global smoothing technique for non-smooth functions and uses the same technique to construct a smoothing auxiliary function to solve unconstrained global optimization problems.

3.1. Global Smoothing Technique

The global smoothing technique is one of the smoothing techniques that use the whole domain to make the objective function differentiable and smooth at points where it is non-differentiable. Suppose that f and g are two continuously differentiable functions on \mathbb{R}^n , we define the following function $\varphi(x) = \max\{f(x), g(x)\}$. The function φ is used in many fields of optimization problems.

First of all, we rewrite φ as

$$\varphi(x) = \frac{1}{2}[\{f(x) - g(x)\}\omega(t) + f(x) + g(x)], \quad (3.1)$$

where $t = f(x) - g(x)$ and the function $\omega(t) : \mathbb{R} \rightarrow \mathbb{R}$ is given by

$$\omega(t) = \begin{cases} 1, & t \geq 0, \\ -1, & t < 0. \end{cases}$$

The function ω is obviously not smooth, and neither is the function φ . To consider a smooth approximation $\tilde{\varphi}$ to φ , a smooth approximation $\tilde{\omega}$ to ω is enough, we present a new global smoothing technique that can be extended to a number of non-smooth functions. In addition, any smoothing function $\tilde{\omega}_\sigma(t)$ can be used if it satisfies the

following characteristics:

- i. $\lim_{t \rightarrow \infty} \tilde{\omega}_\sigma(t) = 1$,
- ii. $\lim_{t \rightarrow -\infty} \tilde{\omega}_\sigma(t) = -1$,
- iii. $\forall t \in \mathbb{R}, \quad \tilde{\omega}'_\sigma(t) > 0$,
- iv. $\forall t \in (-\infty, 0), \quad \tilde{\omega}''_\sigma(t) < 0$ and $\forall t \in [0, \infty), \quad \tilde{\omega}''_\sigma(t) \geq 0$.

We consider the following smoothing function of our methodology

$$\tilde{\omega}_\sigma(t) = \frac{2}{1 + \exp(\frac{-1}{2\sigma}t)} - 1, \quad (3.2)$$

where $t = f(x) - g(x)$ and $\sigma > 0$ is a smoothing parameter. Obviously,

$$\lim_{\sigma \rightarrow 0} \tilde{\omega}_\sigma(t) = \begin{cases} 1 & t > 0, \\ 0 & t = 0, \\ -1 & t < 0. \end{cases} \quad (3.3)$$

Finally, we get the smoothing function version for the function $\varphi(x)$ by using (3.2) as:

$$\tilde{\varphi}_\sigma(x) = \frac{1}{2} [\{f(x) - g(x)\} \tilde{\omega}_\sigma(t) + f(x) + g(x)]. \quad (3.4)$$

The parameter σ is used to squeeze the function $\tilde{\omega}_\sigma(t)$, that means when $\sigma \rightarrow 0$ the function $\tilde{\omega}_\sigma(t) \rightarrow \omega(t)$. The graph of the functions $\tilde{\omega}_\sigma(t)$ and $\omega(t)$ with different values of σ are displayed in figure 3.1.

Based on the above features of the function $\tilde{\omega}_\sigma(t)$ we give the following result.

Lemma 3.1. Let $\tilde{\omega}_\sigma(t)$ be a smoothing function of the function $\omega(t)$ and for any $\sigma > 0$

$$\|\tilde{\omega}_\sigma(t) - \omega(t)\|_{L^1} \leq \frac{17}{3}\sigma.$$

Proof. Since $t \in (-\infty, \infty)$ then

$$\begin{aligned} \|\tilde{\omega}_\sigma(t) - \omega(t)\|_{L^1} &= \int_{-\infty}^{\infty} |\tilde{\omega}_\sigma(t) - \omega(t)| d(t) \\ &= \int_{-\infty}^0 |\tilde{\omega}_\sigma(t) - (-1)| d(t) + \int_0^{\infty} |\tilde{\omega}_\sigma(t) - 1| d(t) \\ &= 4\sigma \ln(2) + 4\sigma \ln(2) \\ &\leq \frac{17}{3}\sigma. \end{aligned}$$

□

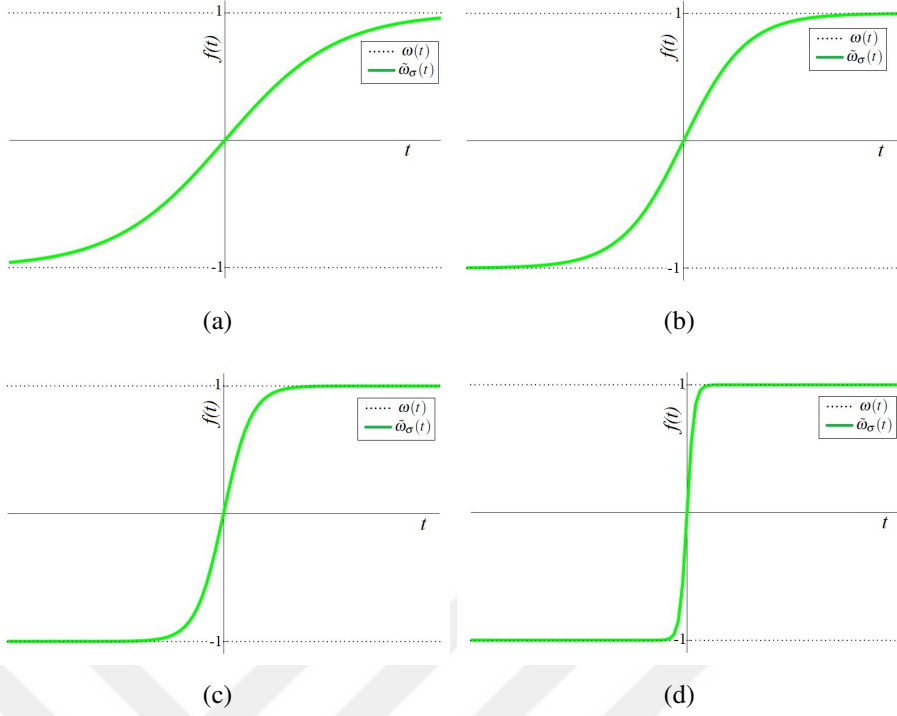


Figure 3.1. The graph of $\tilde{\omega}_\sigma(t)$ (green and solid) and $\omega(t)$ (black and dot) a, b, c and d with $\sigma = 0.8, 0.6, 0.4, 0.2$ respectively.

Theorem 3.2. Let $f(x)$ and $g(x)$ are continuously differentiable functions on \mathbb{R}^n . The difference between the functions $\varphi(x)$ and $\tilde{\varphi}_\sigma(x)$ which are defined in 3.1 and 3.4 can be calculated as follows

$$\|\tilde{\varphi}_\sigma(x) - \varphi(x)\|_{L^1} \leq \frac{20}{3} \sigma^2.$$

Proof. $t = f(x) - g(x)$, then

$$\begin{aligned} \|\tilde{\varphi}_\sigma(x) - \varphi(x)\|_{L^1} &= \int_{-\infty}^{\infty} |\tilde{\varphi}_\sigma(x) - \varphi(x)| dt \\ &= \frac{1}{2} \int_{-\infty}^{\infty} |t(\tilde{\omega}_\sigma(t) - \omega(t))| d(t) \\ &= \frac{1}{2} \int_{-\infty}^0 |t(\tilde{\omega}_\sigma(t) - (-1))| d(t) + \frac{1}{2} \int_0^{\infty} |t(\tilde{\omega}_\sigma(t) - 1)| d(t) \\ &\leq \frac{20}{3} \sigma^2. \end{aligned}$$

□

Theorem 3.3. Suppose that $\tilde{\varphi}_\sigma(x)$ is a smoothing function of the function $\varphi(x)$, then

we have

$$\lim_{\sigma \rightarrow 0} \tilde{\varphi}_\sigma(x) = \varphi(x).$$

Proof. Since the functions $f(x), g(x)$ and $\tilde{\omega}_\sigma(t)$ are smooth, then also the function $\tilde{\varphi}_\sigma(x)$ defined in (3.4) is a smooth for any $\sigma > 0$. From the Theorem 3.2, it can be clearly seen that $\tilde{\varphi}_\sigma(x)$ approaches to $\varphi(x)$ when $\sigma \rightarrow 0$. \square

Example 3.4. For one dimension let $f(x) = 6 - \exp(\frac{1}{2}x)$ and $g(x) = 2x + 2$, we define the function

$$\varphi(x) = \max\{f(x), g(x)\},$$

the smoothing function of the $\varphi(x)$ can be described as follows

$$\tilde{\varphi}_\sigma(x) = \frac{1}{2}[\{f(x) - g(x)\}\tilde{\omega}_\sigma(t) + f(x) + g(x)],$$

where $t = f(x) - g(x)$ and $\sigma > 0$. The graph of the functions $\varphi(x)$ and $\tilde{\varphi}_\sigma(x)$ are shown in figure 3.2.

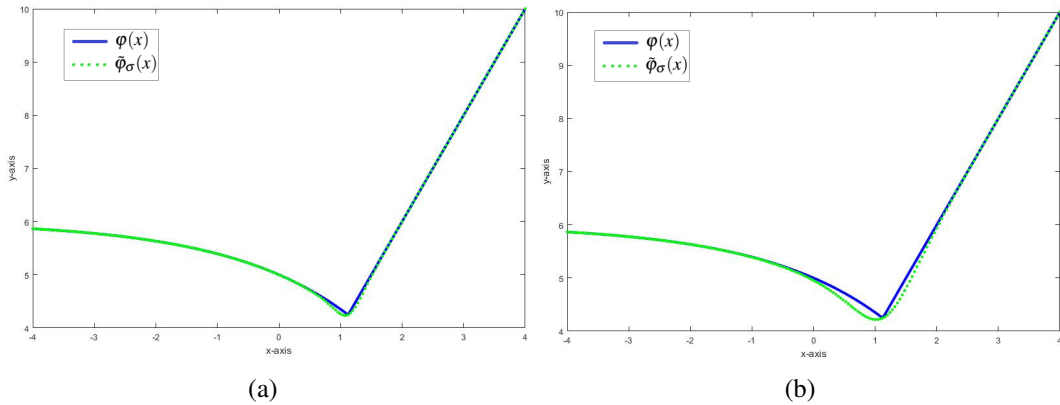


Figure 3.2. The graph of $\varphi(x)$ (blue and solid) and $\tilde{\varphi}_\sigma(x)$ (green and dot) *a* and *b* with $\sigma = 0.4, 0.8$ respectively.

Example 3.5. For two dimension let us define $f(x, y) = 4 - \exp(5\frac{x}{3}) + 3y$ and $g(x, y) = 2x - y + 2$, we define the function

$$\varphi(x, y) = \max\{f(x, y), g(x, y)\},$$

the smoothing function of the $\varphi(x)$ can be defined as follows

$$\tilde{\varphi}_\sigma(x,y) = \frac{1}{2}[\{f(x,y) - g(x,y)\}\tilde{\omega}_\sigma(t) + f(x,y) + g(x,y)],$$

where $t = f(x,y) - g(x,y)$, $\sigma > 0$. The graph of the functions $\varphi(x,y)$ and $\tilde{\varphi}_\sigma(x,y)$ are presented in figure 3.3.

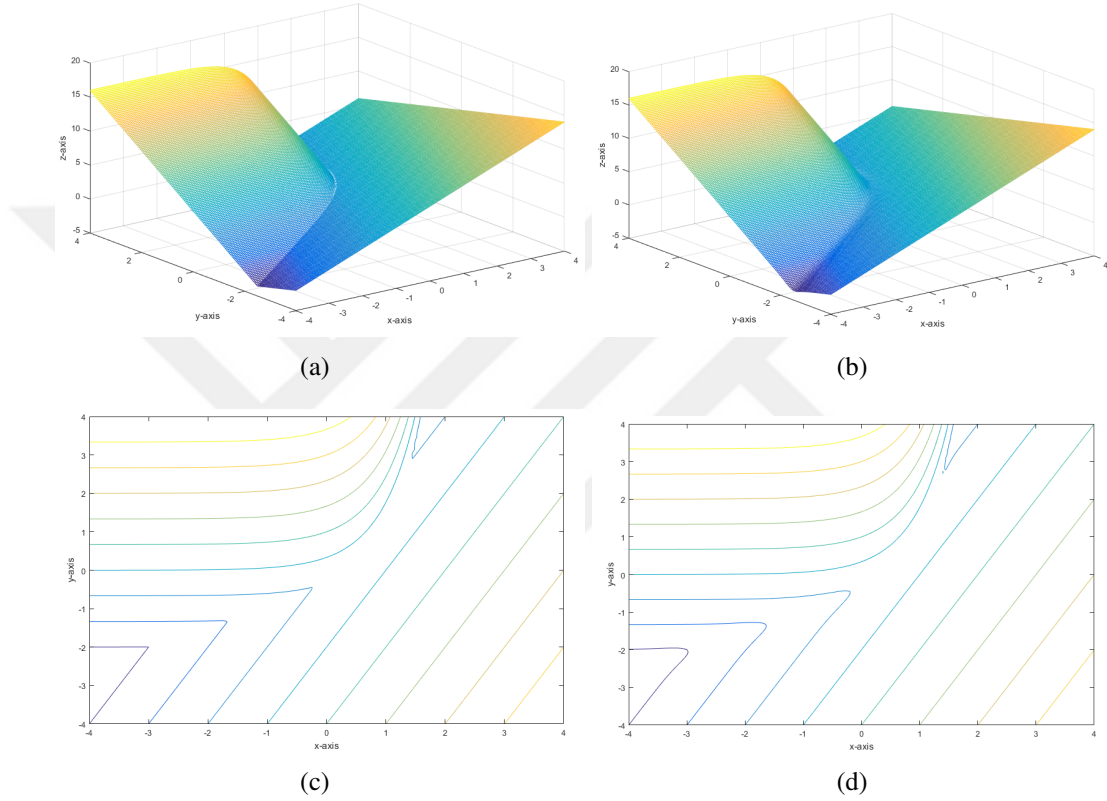


Figure 3.3. (a) The graph of $\varphi(x,y)$, (b) The graph of $\tilde{\varphi}_\sigma(x,y)$, (c) The contour graph of $\varphi(x,y)$ and (d) The contour graph of $\tilde{\varphi}_\sigma(x,y)$.

3.2. Global Optimization Technique

Suppose that f is the function of the problem (P) defined in (1.1), and x_k^* is the current local minimizer of f . In this part, we try to find a global minimizer of the objective function f if it exists. We define the following function to avoid the minimizers that are higher than the current minimizer x_k^*

$$\phi(x, x_k^*) = \min\{f(x), f(x_k^*)\}.$$

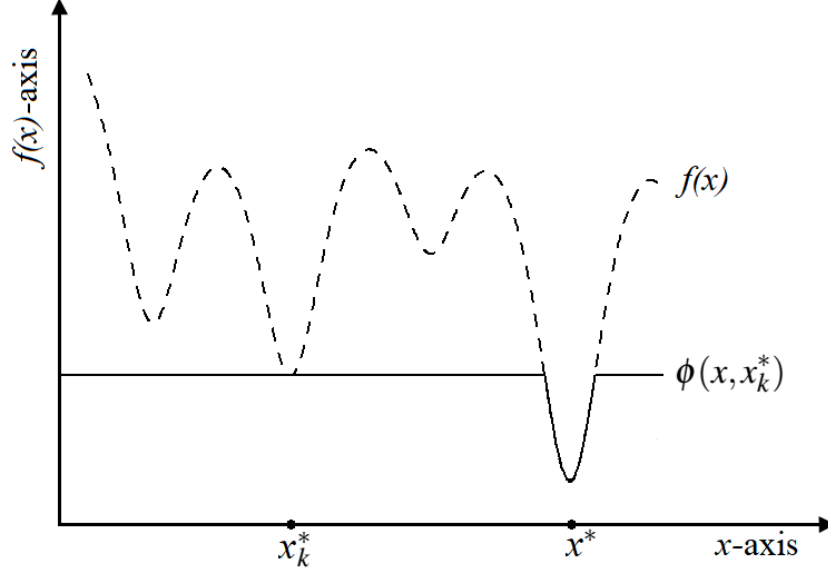


Figure 3.4. The graph of $f(x)$ and $\phi(x, x_k^*)$ before smoothing it, where x_k^* is the current minimizer.

The (min – function) is the best way to avert the higher local minimizers since this function has the same function minimizers of $f(x)$ that are lower than $f(x_k^*)$. The one-dimensional graph of functions $f(x)$ and $\phi(x, x_k^*)$ is given in Fig. 3.4, where the dotted line is $f(x)$, and the solid line is $\phi(x, x_k^*)$. Because we use (min – function) in our approach and this function is non-smooth, first we make $\phi(x, x_k^*)$ smooth, then we create a smooth auxiliary function that can hold lower minimizer of $f(x)$. To render $\phi(x, x_k^*)$ smooth, function ϕ can be reconstructed as follows:

$$\phi(x, x_k^*) = \frac{1}{2}[\{f(x) - f(x_k^*)\}S(t) + f(x) + f(x_k^*)] \quad (3.5)$$

where $t = f(x) - f(x_k^*)$ and the function $S(t) : \mathbb{R} \rightarrow \mathbb{R}$ is defined by

$$S(t) = \begin{cases} 1, & t < 0, \\ -1, & t \geq 0. \end{cases}$$

In order to smooth ϕ , it is sufficient to smooth the function $S(t)$ and by using t instead of $-t$, we can use the function described in (3.2) for that. The smoothing version of $S(t)$ can be defined as follows:

$$\tilde{S}_\sigma(t) = \frac{2}{1 + \exp(\frac{1}{2\sigma}t)} - 1, \quad (3.6)$$

where $t = f(x) - f(x_k^*)$ and $\sigma > 0$. It's clearly,

$$\lim_{\sigma \rightarrow 0} \tilde{S}_\sigma(t) = \begin{cases} 1 & t < 0, \\ 0 & t = 0, \\ -1 & t > 0. \end{cases} \quad (3.7)$$

It is easy to prove that $\lim_{\sigma \rightarrow 0} \tilde{S}_\sigma(t) = S(t)$ (see Lemma 3.1).

Remark 3.6. The function $\tilde{S}_\sigma(t)$ is continuously differentiable and for all $\sigma > 0$, we have

$$\tilde{S}'_\sigma(t) = -\frac{\exp(\frac{1}{2\sigma}t)}{\sigma(\exp(\frac{1}{2\sigma}t) + 1)^2},$$

and

$$\tilde{S}''_\sigma(t) = \frac{(\exp(\frac{1}{2\sigma}t))(\exp(\frac{1}{2\sigma}t) - 1)}{2\sigma^2(\exp(\frac{1}{2\sigma}t) + 1)^3}.$$

The smoothing function of $\phi(x, x_k^*)$ by using the equation (3.6) is defined as:

$$\tilde{\phi}_\sigma(x, x_k^*) = \frac{1}{2}[\{f(x) - f(x, x_k^*)\}\tilde{S}_\sigma(t) + f(x) + f(x_k^*)] \quad (3.8)$$

Theorem 3.7. Suppose that the functions $f(x)$ is twice continuously differentiable on \mathbb{R}^n and x_k^* is a current minimizer of f then

$$\|\tilde{\phi}_\sigma(x, x_k^*) - \phi(x, x_k^*)\|_L \leq \frac{20}{3}\sigma^2,$$

and we have

$$\lim_{\sigma \rightarrow 0} \tilde{\phi}_\sigma(x, x_k^*) = \phi(x, x_k^*).$$

Proof. See Theorems 3.2 and 3.3. □

The smoothing function $\tilde{\phi}_\sigma(x, x_k^*)$ can be very difficult and expensive to transfer from the current minimizer x_k^* throughout the set $\mathbb{D}_1 = \{x \in \mathbb{D} | f(x) \geq f(x_k^*), x \neq x_k^*\}$. To overcome this problem, $\tilde{\phi}_\sigma(x, x_k^*)$ can be added with any escape function. Eventually, the entire auxiliary smoothing function (the graph is shown in Fig. 3.5) that helps to escape from x_k^* and to achieve a better minimizer can be described as follows:

$$\tilde{\phi}_{\sigma, \mu}(x, x_k^*) = \tilde{\phi}_\sigma(x, x_k^*) + \frac{1}{\mu + \|x - x_k^*\|^2} \quad (3.9)$$

where $\mu > 0$ is a parameter of escape function.

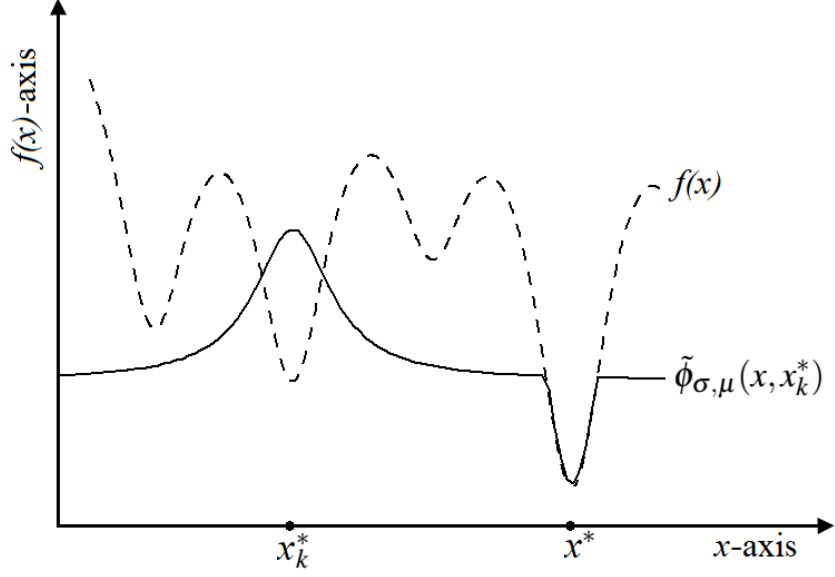


Figure 3.5. After smoothing, the graph of $f(x)$ and $\tilde{\phi}_{\sigma, \mu}(x, x_k^*)$, where x_k^* is the current minimizer.

Theorem 3.8. Suppose that x_k^* is a current local minimizer of f , then $\nabla \tilde{\phi}_{\sigma, \mu}(x, x_k^*) \neq 0$ for any $x \in \mathbb{D}_1$.

Proof. According to the Theorem 3.7 we have

$$\|\tilde{\phi}_{\sigma}(x, x_k^*) - \phi(x, x_k^*)\|_L \leq \frac{20}{3}\sigma^2, \quad \sigma > 0,$$

that means

$$\lim_{\sigma \rightarrow 0} \tilde{\phi}_{\sigma}(x, x_k^*) = \phi(x, x_k^*).$$

Now, for any $x \in \mathbb{D}_1$, $f(x) \geq f(x_k^*)$, if we choose σ close to 0 enough then we obtain

$$\nabla \tilde{\phi}_{\sigma, \mu}(x, x_k^*) = -\frac{2\mu(x - x_k^*)}{(1 + \|x - x_k^*\|^2)^2} \neq 0.$$

□

Theorem 3.9. Let x_k^* is a current minimizer of f but not a global one, and $\mathbb{D}_2 = \{x \in \mathbb{D} | f(x) < f(x_k^*)\}$ not empty then $\tilde{\phi}_{\sigma, \mu}(x, x_k^*)$ has a minimizer when $x \in \mathbb{D}_2$.

Proof. From the equation defined in (3.7) and for all $x \in \mathbb{D}_2$ we have

$$\lim_{\sigma \rightarrow 0} \tilde{S}_\sigma(t) = 1,$$

then

$$\tilde{\Phi}_{\sigma,\mu}(x, x_k^*) = f(x) + \frac{1}{\mu + \|x - x_k^*\|^2}, \quad \mu > 0.$$

Now, if $\mu \rightarrow \infty$ or $\|x - x_k^*\|^2 \rightarrow \infty$ then we obtain

$$\tilde{\Phi}_{\sigma,\mu}(x, x_k^*) \simeq f(x),$$

and since x_k^* is not a global for f , then there exists another local minimizer x_{k+1}^* for f such that $f(x_{k+1}^*) < f(x_k^*)$, that means that there is another local minimizer exists $x' \in \mathbb{D}_2$ for the function $\tilde{\Phi}_{\sigma,\mu}(x, x_k^*)$ such that $x' \in B(x_{k+1}^*)$ and

$$\nabla \tilde{\Phi}_{\sigma,\mu}(x', x_k^*) = 0,$$

for some value of μ . □

3.3. Algorithm

According to the above information, an auxiliary function algorithm is introduced:

Step 1. Put $k = 1$, $\sigma \leq 10^{-2}$, $\varepsilon = 10^{-2}$, $\mu > 0$; directions d_i for $i = 1, 2, 3, \dots, N$ (N number of different directions), and choose $x_{int} \in \mathbb{D}$ as an initial point.

Step 2. Minimize $f(x)$ and use x_{int} as a start point to find any minimizer x_k^* and set $i = 1$.

Step 3. Construct the auxiliary function $\tilde{\Phi}_{\sigma,\mu}(x, x_k^*)$ at x_k^*

$$\tilde{\Phi}_{\sigma,\mu}(x, x_k^*) = \tilde{\Phi}_\sigma(x, x_k^*) + \frac{1}{\mu + \|x - x_k^*\|^2}.$$

Step 4. If $i \leq N$ then put $x = x_k^* + \varepsilon d_i$ and go to Step 5; otherwise go to Step 6.

Step 5. Minimize $\tilde{\Phi}_{\sigma,\mu}(x, x_k^*)$ use x as an initial to find a minimizer x_ϕ of $\tilde{\Phi}_{\sigma,\mu}(x, x_k^*)$, if $x_\phi \in \mathbb{D}$ then set $x_{int} = x_\phi$, $k = k + 1$ and go to Step 2; otherwise $i = i + 1$ then go to Step 4.

Step 6. Take $x^* = x_k^*$ as a global minimizer of $f(x)$ and stop the algorithm.

4. GLOBAL OPTIMIZATION TECHNIQUE IN A DIRECTIONAL SEARCH

In this Section, a new auxiliary function is presented to solve a multi-model of unconstrained global optimization problems, by converting a multidimensional problem into partitions of some one-dimensional problems based on the number of directions, and decreasing minimizers number for each direction as well as finding the global minimizer of the one-dimensional problem. Finally, with the help of a new algorithm, we find the global minimizer of a multi-dimensional problem.

4.1. Auxiliary Function Method in Global Optimization

Suppose f is the objective function of (P) problem and x_k^* is a local minimizer of f . In this part, we try to find the global minimizer of the function f . To achieve the global minimizer, we use the function $\psi(x, x_k^*) = \min\{f(x), f(x_k^*)\}$, which has the same local minimizers with objective function $f(x)$, less than $f(x_k^*)$. The function ψ helps to ignore upper local minimizers but its problem is that it is neither differentiable nor smooth. As the function ψ is not smooth, first, we make it smooth and then we construct a function that can be minimized easily to obtain a better solution of objective function lower than $f(x_k^*)$. The ψ function can be reformulated as follows:

$$\psi(x, x_k^*) = \min\{f(x) - f(x_k^*), 0\} + f(x_k^*) = \{f(x) - f(x_k^*)\} \omega_{A'_k}(t) + f(x_k^*),$$

where $A'_k = \{t \in \mathbb{R} : t = f(x) - f(x_k^*) < 0\}$ and $\omega_{A'_k} : \mathbb{R} \rightarrow \mathbb{R}$ is defined by

$$\omega_{A'_k}(t) = \begin{cases} 1, & t < 0, \\ 0, & t \geq 0. \end{cases} \quad (4.1)$$

Since the $\omega_{A'_k}(t)$ function is not smooth, then the $\psi(x, x_k^*)$ function is also not smooth. For smoothing the $\psi(x, x_k^*)$ function, it is sufficient to try to smooth the $\omega_{A'_k}(t)$ function. Some significant recent studies on this subject include some forms of smoothing functions in the literature see (Yang et al., 2014; Ralph and Xu, 2005; Wu et al., 2015).

In this Section, we use the global smoothing technique that was presented in (Yilmaz and Sahiner, 2019) to smooth the $\omega_{A'_k}(t)$ function. In fact we could use any $\tilde{\omega}_{A'_k}(t, \beta)$ smoothing function with the following features for our methodology:

- i. $\lim_{t \rightarrow -0} \tilde{\omega}_{A'_k}(t, \beta) = 1$,
- ii. $\lim_{t \rightarrow +0} \tilde{\omega}_{A'_k}(t, \beta) = 0$,
- iii. $\forall t \in \mathbb{R}, \quad \tilde{\omega}'_{A'_k}(t, \beta) < 0$,
- iv. $\forall t \in (-\infty, 0], \quad \tilde{\omega}''_{A'_k}(t, \beta) < 0$ and $\forall t \in [0, \infty), \quad \tilde{\omega}''_{A'_k}(t, \beta) > 0$.

Example for $\tilde{\omega}_{A'_k}$ could be $\tilde{\omega}_{A'_k}(t, \beta) = \frac{\arctan\left(\frac{-t}{\beta}\right) + \frac{\pi}{2}}{\pi}$, $\tilde{\omega}_{A'_k}(t, \beta_1, \beta_2) = \frac{\arctan\left(\frac{-t-\beta_1}{\beta_2}\right) + \frac{\pi}{2}}{\pi}$, $\beta, \beta_1, \beta_2 > 0$, etc. For our methodology and algorithm, we consider the smoothing function

$$\tilde{\omega}_{A'_k}(t, \beta) = \frac{\arctan\left(\frac{-t}{\beta}\right) + \frac{\pi}{2}}{\pi}, \quad (4.2)$$

where $t = f(x) - f(x_k^*)$, $\beta > 0$. Clearly,

$$\lim_{\beta \rightarrow 0} \tilde{\omega}_{A'_k}(t, \beta) = \begin{cases} 0 & t > 0, \\ \frac{1}{2} & t = 0, \\ 1 & t < 0. \end{cases} \quad (4.3)$$

Additionally, we gain a smoothing function for $\psi(x, x_k^*)$ by using (4.2) as:

$$\tilde{\psi}(x, x_k^*, \beta) = \{f(x) - f(x_k^*)\} \tilde{\omega}_{A'_k}(t, \beta) + f(x_k^*).$$

Based on the above information we give the following results.

Theorem 4.1. Suppose that x_k^* is any local minimizer of f and $\beta > 0$, then $\lim_{\beta \rightarrow 0} (\tilde{\psi}(x, x_k^*, \beta) - \psi(x, x_k^*)) = 0$.

Proof. From (4.3) and (4.1) we have

$$\lim_{\beta \rightarrow 0} (\tilde{\psi}(x, x_k^*, \beta) - \psi(x, x_k^*)) = t \left(\lim_{\beta \rightarrow 0} \tilde{\omega}_{A'_k}(t, \beta) - \omega_{A'_k}(t) \right).$$

For $t = (f(x) - f(x_k^*))$, now, if $t > 0$ we obtain

$$\lim_{\beta \rightarrow 0} \tilde{\omega}_{A'_k}(t, \beta) = 0 \quad \text{and} \quad \omega_{A'_k}(t) = 0,$$

and, if $t = 0$ we have

$$\lim_{\beta \rightarrow 0} (\tilde{\psi}(x, x_k^*, \beta) - \psi(x, x_k^*)) = 0,$$

and, if $t < 0$ we have

$$\lim_{\beta \rightarrow 0} \tilde{\omega}_{A'_k}(t, \beta) = 1 \quad \text{and} \quad \omega_{A'_k}(t) = 1.$$

□

Throughout the set \mathbb{D}_1 , $\tilde{\psi}(x, x_k^*, \beta)$ function can not leave the current local minimizer. This problem can be solved here by combining a new term to the $\tilde{\psi}(x, x_k^*, \beta)$ function. Finally, to achieve a lower minimizer, the last version of the smooth function can be stated as follows:

$$G(x, x_k^*, \beta, \mu) = \tilde{\psi}(x, x_k^*, \beta) + \mu \frac{1}{1 + \|x - x_k^*\|^2}.$$

Theorem 4.2. Suppose f is a continuously differentiable function on \mathbb{D} , x_k^* is any minimizer of f and $\beta > 0$ is given. If

$$\mu > L \left(\frac{1}{2M} + \frac{1}{2L} \frac{1}{\pi\beta} \right) (1 + M^2)$$

then $G(x, x_k^*, \beta, \mu)$ function cannot have a stationary point when $f(x) > f(x_k^*)$, where $M = \text{diam}\mathbb{D}$, $\|\nabla f(x)\| \leq L$.

Proof. Let $\mu > L \left(\frac{1}{2M} + \frac{1}{2L} \frac{1}{\pi\beta} \right) (1 + M^2)$. If $G(x, x_k^*, \beta, \mu)$ has a stationary point then we have $\nabla G(x, x_k^*, \beta, \mu) = 0$, then

$$\begin{aligned} & \left\| \frac{2\mu(x - x_k^*)}{(1 + \|x - x_k^*\|^2)^2} \right\| = \left\| \nabla f(x)(\tilde{\omega}_{A'_k}(t, \beta) + (f(x) - f(x_k^*))\tilde{\omega}'_{A'_k}(t, \beta)) \right\| \\ \Rightarrow \quad & |\mu| = \left\| \nabla f(x)(\tilde{\omega}_{A'_k}(t, \beta) + (f(x) - f(x_k^*))\tilde{\omega}'_{A'_k}(t, \beta)) \right\| \frac{\|(1 + \|x - x_k^*\|^2)^2\|}{\|2(x - x_k^*)\|} \\ & \leq L \left(\frac{1}{2M} + \frac{1}{2L} \frac{1}{\pi\beta} \right) (1 + M^2). \end{aligned}$$

This is clearly a contradiction. □

Theorem 4.3. Let f be continuously differentiable of a problem (P) and x_k^* is a current minimizer if f has another minimizer lower than x_k^* and

$$\mu = |\mu| \leq \frac{L(1 + LM\frac{1}{\pi\beta})}{0.65},$$

then $G(x, x_k^*, \beta, \mu)$ has a stationary point on \mathbb{D}_2 .

Proof. We have $G(x, x_k^*, \beta, \mu) = \tilde{\psi}(x, x_k^*, \beta) + \mu \frac{1}{1 + \|x - x_k^*\|^2}$, we want the decrement speed of $\frac{\mu}{1 + \|x - x_k^*\|^2}$ to be smaller than or equal to the increment speed of $\tilde{\psi}(x, x_k^*, \beta)$. It is clear that

$$\|\nabla \frac{\mu}{1 + \|x - x_k^*\|^2}\| \leq 0.65|\mu|,$$

and

$$\|\nabla f(x)(\tilde{\omega}_{A_k'}(t, \beta) + (f(x) - f(x_k^*))\tilde{\omega}'_{A_k'}(t, \beta))\| \leq L(1 + LM\frac{1}{\pi\beta}).$$

So if we choose $\mu = |\mu| \leq \frac{L(1 + LM\frac{1}{\pi\beta})}{0.65}$ then our function will have a stationary point on \mathbb{D}_2 . □

In the next part, we work on a one-dimensional global optimization problem, and since this problem is a branch of the multi-variable function and we gave all the proofs in the previous part, we will give our methodology a one-dimensional version without proof.

4.2. One-dimensional Minimization Problem

Line search strategies are the most usable in global optimization. If we want to describe them, let f be a function and suppose x_k be any point in the domain of f , and let d_k be the search direction at x_k . The formulation of a new estimate solution can be described as

$$x_{k+1} = x_k + \alpha d_k,$$

where the step length α is some scalar chosen so that

$$f(x_{k+1}) < f(x_k).$$

If the function value at the next point is less than the function value at the current point, progress toward the minimizer would be made. Let us look closely at the line search formula, and we assume that d_k is a descent direction at x_k , then d_k achieves the

following condition

$$d_k^T \nabla f(x) \leq 0.$$

The method used to calculate the line search direction achieves $f(x_k + \alpha d_k) < f(x_k)$ at least for small positive values of α if d_k is a descent direction. Because of this property, we suppose the step length gets on with $\alpha > 0$. The technique we described is called a "line search" because a search is performed for another point x_{k+1} along with the line $h(\alpha) = x_k + \alpha d_k$. Of course we might want to choose α as an answer for

$$\min H(\alpha) = f(x_k + \alpha d_k).$$

That is, α_k^* would be the outcome of a one-dimensional minimization problem (Griva et al., 2009) in the direction d_k .

To adjust our methodology for multi-variable function to one dimensional-function, we will use $H(\alpha)$ in lieu of $f(x)$, $\tilde{\omega}_{A_k}^\alpha(t, \beta)$ in lieu of $\tilde{\omega}_{A'_k}(t, \beta)$, $\tilde{\Psi}_\alpha(\alpha, \alpha_k, \beta)$ in lieu of $\tilde{\Psi}(x, x_k^*, \beta)$, $G_\alpha(\alpha, \alpha_k, \beta, \mu)$ in lieu of $G(x, x_k^*, \beta, \mu)$ and α_k is a local minimizer of one-dimensional problem. Now we give the details as follows: First of all, we create a single variable function by taking any initial point $x_{int} \in \mathbb{D}$ and $H(\alpha) = f(x_{int} + \alpha d_k)$ in the direction d_k we find first local minimizer α_k^i of $H(\alpha)$, and then we create an auxiliary function $G_\alpha(\alpha, \alpha_k^i, \beta, \mu)$ at this local minimizer α_k^i . After that, we find a minimizer of $G_\alpha(\alpha, \alpha_k^i, \beta, \mu)$ which is certainly in a lower basin of $H(\alpha)$ (This is guaranteed by the Theorems 4.2 and 4.3). After this step, by using the minimizer α_G of the function $G_\alpha(\alpha, \alpha_k^i, \beta, \mu)$ as a starting point, we minimize $H(\alpha)$ and we obtain a lower minimizer α_k^{i+1} . By repeating this process, finally the global minimum point is obtained as α_k^* . Then we find the point x'_k in \mathbb{D} corresponding to the point α_k^* by using $x'_k = x_{int} + \alpha_k^* d_k$ (see Fig. 4.1). By using this point as an initial point we minimize the objective function $f(x)$ to obtain the corresponding minimizer x_k^* of $f(x)$.

Note that a function $H(\alpha)$ corresponds to each of the directions. Therefore the global minimizers α_k^* of functions $H(\alpha)$ are found by repeating the above cycle for each of these functions. Finally, we obtain the global minimizer x^* of $f(x)$ by comparing function values at these minimizers.

One could think that because of the number of directions our algorithm needs more time and potential and use the auxiliary function in each direction. In addition, the

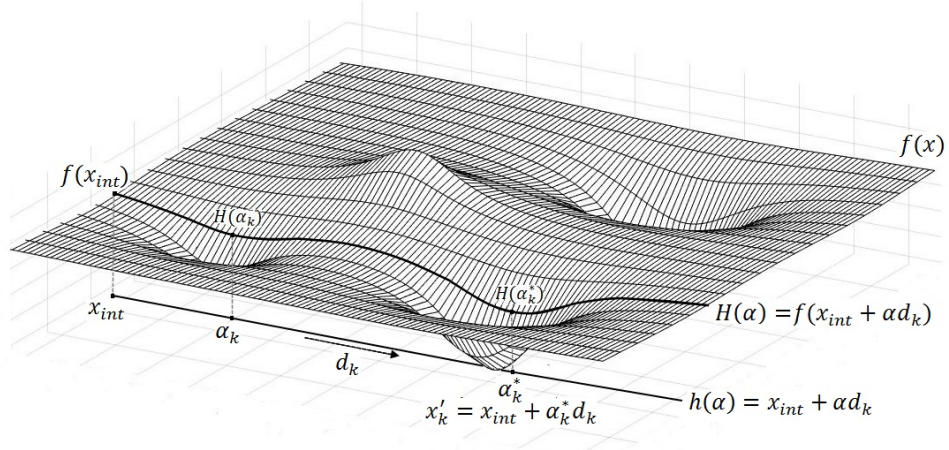


Figure 4.1. The one-dimensional problem

presented approach is effective and positive, as it is seen from the results in Section 6. This provides other advantages such as the auxiliary function that can be used only on a one-dimensional function and obtain the global minimizers of $H(\alpha)$ faster.

4.3. Algorithm

- Step 1. Set $k = 0$, $\varepsilon = 10^{-2}$, $\beta = 10^{-1}$, N the number of directions d_k for $k = 1, 2, \dots, N$, choose an initial point $x_{int} \in \mathbb{D}$, and locate boundary of \mathbb{D} .
- Step 2. Construct the function $H(\alpha) = f(x_{int} + \alpha d_k)$, in a one-dimensional function.
- Step 3. (1) Find a local minimizer α_k^i of the function $H(\alpha)$ from any starting point α_0 . and take $p = -1$.
- (2) Construct an auxiliary function $G_\alpha(\alpha, \alpha_k^i, \beta, \mu)$ at α_k^i .
- (3) Start from $\alpha_0 = \alpha_k^i + p\varepsilon$ to find a minimizer α_G of $G_\alpha(\alpha, \alpha_k^i, \beta, \mu)$.
- (4) If α_G in \mathbb{D} go to (5) otherwise go to (7).
- (5) Start from α_G minimize $H(\alpha)$ to obtain lower minimizer α_k^{i+1} and go to (6).
- (6) If α_k^{i+1} in \mathbb{D} take $\alpha_k^i = \alpha_k^{i+1}$ go to (2).
- (7) If $p = 1$ stop the algorithm take $\alpha_k^* = \alpha_k^i$ otherwise; take $p = 1$ go to (3).
- Step 4. – Calculate x_k' using $x_k' = x_{int} + \alpha_k^* d_k$.
- Find the local minimizer x_k^* of the function $f(x)$ by using x_k' as initial point.
- Step 5. If $k < N$, let $k = k + 1$, generate a new search direction d_{k+1} , and go to Step 2;

otherwise go to Step 6.

Step 6. Find the global minimizer of f as

$$x^* = \min \{f(x_1^*), f(x_2^*), \dots, f(x_N^*)\}.$$



5. INCREASING THE EFFECTS OF FILLED FUNCTION IN GLOBAL OPTIMIZATION

Filled function method can be considered as an effective approach to solve different global optimization problems, so it seems to have several features over other methods, for example, it is more simple to find a better local minimizer sequentially compared to other methods. The filled function method presents a good idea to solve global optimization problems. The main idea of filled function method is to change the objective function into a filled function by using the current local minimizer and then to minimize it to obtain a good initial point for the minimization of the objective function and finally to obtain a better local minimizer from the previous one. In general, the mechanism work of the filled function can be described as follows:

- 1- Minimizing objective function by using any random point as an initial point with using a local optimization method to find a local minimizer of the objective function.
- 2- Construct a filled function based on the current minimizer of the objective function, and use any point near this point (current minimizer) to minimize the filled function. As a result, a minimizer of the filled function will be obtained. This minimizer will be a point into a basin of a better solution of the objective function.
- 3- From the minimizer of filled function which obtained in step 2 and use it as an initial point to minimize the objective function and a better solution of objective function will be found.

By repeating the Step 2 and step 3, the number of minimizers will be surely reduced and global minimizer of objective function will be found. Some of the existing filled function methods have been constructed to have a surface somewhat like a surface of the objective function in the lower basin (when $f(x) < f(x_k^*)$, x_k^* is a current minimizer of the objective function) of the better solution, this situation has drawbacks; it needs more time and function evaluations. In this Section and in order to eliminate the drawbacks in some previous filled function methods we proposed a new filled function method. This new proposal is based on putting many stationary points in the lower basin, in fact, the filled function does not need to go down in the lower basin, only it needs to obtain any stationary point in the lower basin and which can be used as an initial point

for minimizing the objective function to obtain a lower minimizer. This idea helps to reduce the time and function evaluations which are very important in cases of these methods.

5.1. Overview of the Filled Function Method

In 1987 Ge and Qin proposed a first filled function (we call it as G-function) (Ge and Qin, 1987) with two parameters to solve the problem (P) at an isolated local minimizer x_k^* that is defined by

$$G(x, x_k^*, r, \rho) = -(\rho^2 \ln[r + f(x)] + \|x - x_k^*\|^2), \quad (5.1)$$

and, in 1990 Ge introduced another filled function (P-function) (Renpu, 1990) which has the following form

$$P(x, x_k^*, r, \rho) = \frac{1}{r + f(x)} + \exp\left(-\frac{\|x - x_k^*\|^2}{\rho^2}\right), \quad (5.2)$$

where r and ρ are parameters which need to be chosen conveniently. Generally, the G-function and P-function of the objective function $f(x)$ at the current minimizer x_k^* must satisfy the Definition 2.10.

Many important studies are developed the filled function method to solve multi-modal global optimization. These studies can be classified into two categories depending on the number of adjustable parameters.

5.1.1. Filled Function Methods with Two-parameter

(Wu et al., 2006), proposed a filled function with two parameters to progress the activity of numerical computation and overcome several drawbacks of filled functions and the filled function has the following form

$$H_{q,r,x_k^*}(x) = q\left(\exp\left(-\frac{\|x - x_k^*\|^2}{q}\right)g_r(f(x) - f(x_k^*)) + f_r(f(x) - f(x_k^*))\right), \quad (5.3)$$

where $q, r > 0$ are adjustable parameters and f_r, g_r are continuously differentiable functions.

In 2009, (Zhang et al., 2009) introduced a new definition for the filled function, which rectifies several drawbacks of the classic definition. A new filled function with two

parameters defined by

$$P(x, x_k^*, r, a) = \varphi(r + f(x)) - a(\|x - x_k^*\|^2), \quad (5.4)$$

where $a > 0$, r are parameters and the function $\varphi(t)$ is continuously differentiable.

Wei et al., proposed a new continuously differentiable filled function and not sensitive to parameters (Wei et al., 2014). This function has two parameters and has the following formula

$$P(x, x_k^*) = \frac{1}{(1 + \|x - x_k^*\|^2)} g(f(x) - f(x_k^*)), \quad (5.5)$$

and

$$g(t) = \begin{cases} 0, & t \geq 0, \\ r \cdot \arctan(t^\rho), & t < 0, \end{cases}$$

where $r, \rho > 1$ are parameters and r is an adjustable positive number.

5.1.2. Filled Function Methods with One-parameter

According to general opinion, the existence of more than one adjustable parameter in the same filled function makes it difficult to control. So, the first filled function which has only one parameter was Q-function. This function proposed in (1987) by Ge and Qin and has the following formula:

$$Q(x, a) = -(f(x) - f(x_k^*)) \exp(a\|x - x_k^*\|^2). \quad (5.6)$$

Q-function has one adjustable parameter a , if this parameter becomes large and large, the value of exponential function quickly increases which negatively affects the computational results (Ge and Qin, 1987). To overcome this weakness, H-function was introduced by (Liu, 2001) that is given by

$$H(x, a) = \frac{1}{\ln(1 + f(x) - f(x_k^*))} - a\|x - x_k^*\|^2. \quad (5.7)$$

H-function keeps the feature of Q-function with only one adjustable parameter but without exponential function. Shang et al., introduced a filled function with one adjustable parameter in the following

$$F_q(x, x_k^*) = \frac{1}{(1 + \|x - x_k^*\|)} \varphi_q(f(x) - f(x_k^*) + q), \quad (5.8)$$

and

$$\varphi_q(t) = \begin{cases} \exp(-\frac{q^3}{t}), & \text{if } t \neq 0, \\ 0, & \text{if } t = 0, \end{cases}$$

so, q is a parameter subject to certain conditions (Shang et al., 2007). Zhang and Xu constructed a filled function to solve non-smooth constrained global optimization problems (Zhang et al., 2009). This function constructed to overcome several drawbacks of the previous filled functions, and it has one parameter as follows:

$$P(x, x_1^*, q) = \exp(\|x - x_k^*\|) \ln(1 + q(\max\{0, f(x) - f(x_1^*) + r\} + \sum_{i=1}^m \max\{0, g_i(x)\})),$$

where $q > 0$ is the parameter, $g_i(x) > 0$ are constrained conditions and r is prefixed constant.

In 2013, Wei and Wang proposed a new filled function for problem (P) with one adjustable parameter and it is not sensitive to this parameter (Wei and Wang, 2013). The filled function has the following formula:

$$P(x, x_k^*) = -\|x - x_k^*\|^2 g(f(x) - f(x_k^*)), \quad (5.9)$$

where

$$g(t) = \begin{cases} \frac{\pi}{2}, & t \geq 0, \\ r \cdot \arctan(t^2) + \frac{\pi}{2}, & t < 0, \end{cases}$$

and r is an adjustable parameter as large as possible, used as the weight parameter.

Wang et. al. constructed a new filled function for smooth and non-smooth constrained global optimization problems in (Lin et al., 2014). The constructed filled function defined by

$$P(x, x_k^*, q) = -\frac{1}{q} [f(x) - f(x_k^*) + \max\{0, g_i(x)\}]^2 - \arg(1 + \|x - x_k^*\|^2) + q[\min(0, \max(f(x) - f(x_k^*), g_i(x), i \in I))]^3.$$

The above filled function has only one adjustable parameter which can easily control it during the minimization process. A new definition and a new filled function is given in

(Yuan et al., 2016a). This filled function has one parameter, given by

$$F_1(x, x_k^*) = V(\|x - x_k^*\|)W_q(f(x) - f(x_k^*)), \quad (5.10)$$

where $q > 0$ is an adjustable parameter, $V(t) : \mathbb{R} \rightarrow \mathbb{R}$ and $W_q(t) : \mathbb{R} \rightarrow \mathbb{R}$ are continuously differentiable under some properties.

5.2. New Filled Function and Its Properties

In this Section, we propose a new filled function for the problem (P) with two parameters at a local minimizer x_k^* as follows:

$$F(x, x_k^*) = \frac{1}{\alpha + \|x - x_k^*\|^2} h(f(x) - f(x_k^*)),$$

where

$$h(t) = \begin{cases} 1 & t \geq 0, \\ \sin(\mu t + \frac{\pi}{2}) & t < 0, \end{cases}$$

and $0 < \alpha \leq 1$ and $\mu > 1$ are parameters.

The new idea in this filled function is to put many stationary points in the lower basin $\mathbb{D}_2 = \{x | f(x) < f(x_1^*), x \in \mathbb{D}\}$, in fact the filled function does not need to go down in the lower basin, only it needs to obtain any stationary point in \mathbb{D}_2 , which can be used as an initial for minimizing objective function to obtain a lower minimizer.

The above idea has many advantages, for example, it helps to reduce the time and evaluation which are very important in cases like this. Furthermore, the parameter μ is used to increase or decrease the number of stationary points in the interval \mathbb{D}_2 , therefore we have to choose μ carefully, because if it is small there is a possibility that we may lose some of the lower minimizers at which the value of the function is close to the value at the first minimizer (see Fig. 5.1 and Fig. 5.2). The parameter $0 < \alpha \leq 1$ in the term $\frac{1}{\alpha + \|x - x_k^*\|^2}$ is used to control the hat and it is easy to adjustable it. The following theorems references that the function $F(x, x_k^*)$ is a filled function by Definition 2.10.

Theorem 5.1. Suppose that x_k^* is a local minimizer of the function $f(x)$, and $F(x, x_k^*)$ is defined by the Definition 2.10, then x_k^* is a strict local maximizer of $F(x, x_k^*)$.

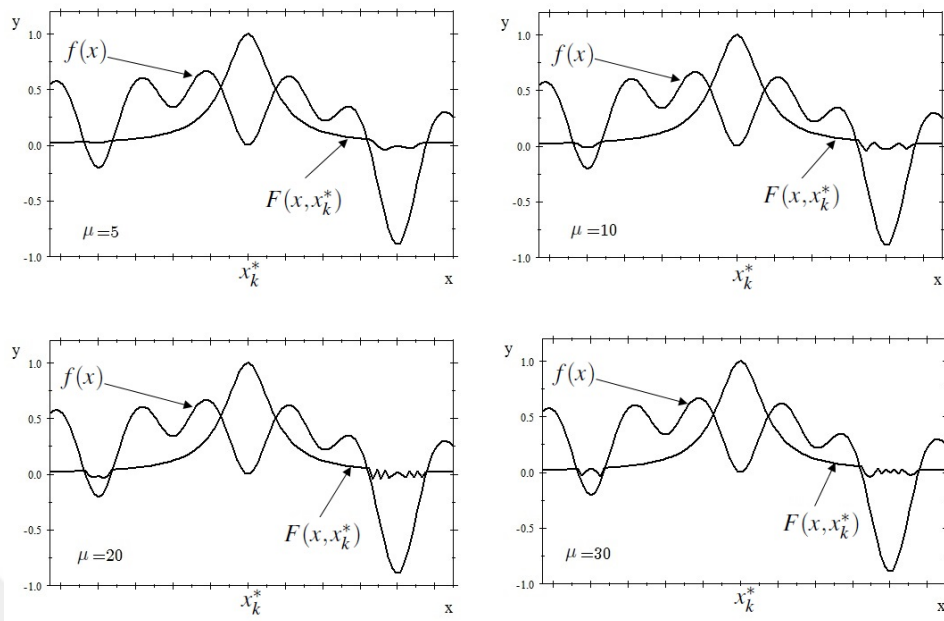


Figure 5.1. Some different values of the parameter μ and their effect on the function $F(x, x_k^*)$.

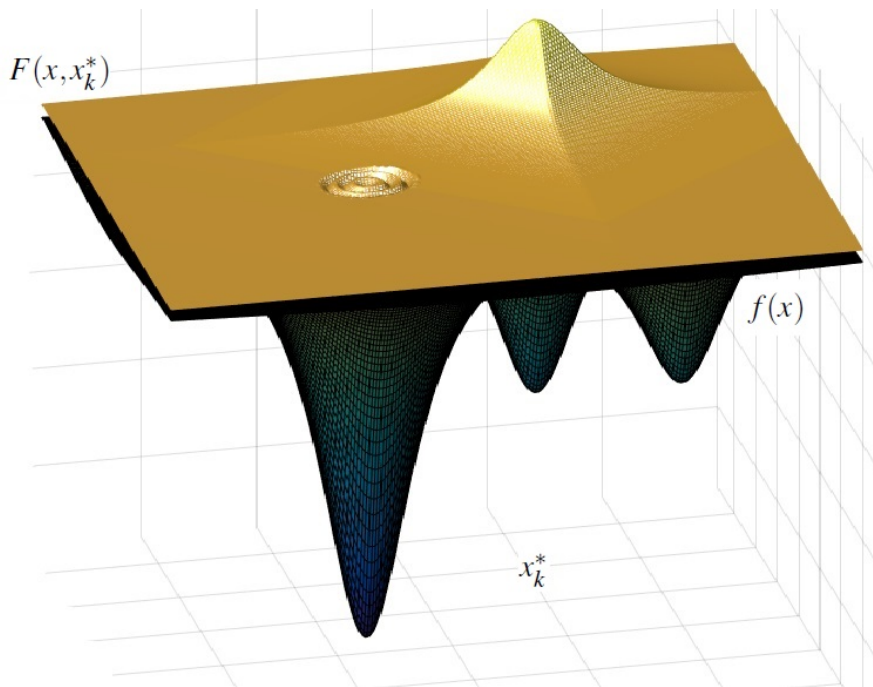


Figure 5.2. The shape of the functions $F(x, x_k^*)$ and $f(x)$ in two dimensions.

Proof. Since x_k^* is a local minimizer of $f(x)$, then there exists neighborhood $N(x_k^*, \varepsilon) \subset \mathbb{D}_1$ of x_k^* for some $\varepsilon > 0$ such that $f(x) \geq f(x_k^*)$ for all $x \in N(x_k^*, \varepsilon)$ and $x \neq x_k^*$, $0 < \alpha \leq 1$.

$$\frac{F(x, x_k^*)}{F(x_k^*, x_k^*)} = \frac{\alpha + \|x_k^* - x_k^*\|^2}{\alpha + \|x - x_k^*\|^2} = \frac{\alpha}{\alpha + \|x - x_k^*\|^2} < 1.$$

That is x_k^* a strict local maximizer of $F(x, x_k^*)$. □

Theorem 5.2. Suppose that x_k^* is a local minimizer of $f(x)$, and x is any point in the set \mathbb{D}_1 , then x is not a stationary point of $F(x, x_k^*)$ for any $0 < \alpha \leq 1$.

Proof. We have $x \in \mathbb{D}_1$, $f(x) \geq f(x_k^*)$ and $x \neq x_k^*$. Then $F(x, x_k^*) = \frac{1}{\alpha + \|x - x_k^*\|^2}$, and $\nabla F(x, x_k^*) = -2 \frac{x - x_k^*}{(\alpha + \|x - x_k^*\|^2)^2} \neq 0$, for each $0 < \alpha \leq 1$. This implies the function $F(x, x_k^*)$ has no stationary point in the set \mathbb{D}_1 . □

Theorem 5.3. Suppose that $L = \min |f(x_i^*) - f(x_j^*)|$, $i, j = 1, 2, \dots, k$, $f(x_i^*) \neq f(x_j^*)$, and assume x_k^* is a local minimizer of $f(x)$ but not global, then there exists a point $x' \in \mathbb{D}_2$ such that the point x' is a local minimizer of the function $F(x, x_k^*)$ when $\mu = \frac{\pi}{2L}$ for each $0 < \alpha \leq 1$.

Proof. Since the current local minimizer x_k^* is not global minimizer of $f(x)$, then there exists second minimizer $x_{(k+1)}^* \in \mathbb{D}_2$ such that $f(x_{(k+1)}^*) < f(x_k^*)$.

For any point $y \in \mathbb{D}_1$ we have $F(y, x_k^*) > 0$, so by the continuity of $f(x)$, and if $\mu = \frac{\pi}{2L}$ we obtain $F(x_{(k+1)}^*, x_k^*) < 0$. Then, by theorem of the intermediate value of continuous function, there exist a point lying between the points y and $x_{(k+1)}^*$ on the part $[y, x_{(k+1)}^*]$, the value of the filled function at this point is equal to 0.

Assuming that z is the nearest point to $x_{(k+1)}^*$ with $F(z, x_k^*) = 0$, then, we obtain the part $[z, x_{(k+1)}^*]$. That means $z \in \partial \mathbb{D}_2$ and z is in the borders of the set $B_{(k+1)}^*$ which is a closed region. By the continuity of the function $F(x, x_k^*)$, there exist a point $x' \in B_{(k+1)}^*$ such that it is a local minimizer of the function $F(x, x_k^*)$ and $F(x', x_k^*) < 0$, since the function $F(x, x_k^*)$ is continuously differentiable, we obtain

$$\nabla F(x', x_k^*) = 0.$$

□

5.3. Algorithm

According to the previous information, we proposed a new filled function algorithm as follows:

Step 1. Set $k = 1$, $\varepsilon = 10^{-2}$, choose $U_\mu = 30$ an upper bound of μ and give $\mu = 5$; N the number of different directions d_i for $i = 1, 2, 3, \dots, N$, choose an initial point $x_{int} \in \mathbb{D}$, and give $0 < \alpha \leq 1$, where n is the dimension of the problem.

Step 2. Minimize $f(x)$ using x_{int} as a starting point to find local minimizer x_k^* .

Step 3. Construct filled function at x_k^*

$$F(x, x_k^*) = \frac{1}{\alpha + \|x - x_k^*\|^2} h(f(x) - f(x_k^*))$$

and set $i = 1$.

Step 4. If $i \leq N$, set $x = x_k^* + \varepsilon d_i$ and go to Step 5; otherwise go to Step 6.

Step 5. Start from x to find a minimizer x_F of $F(x, x_k^*)$, if $x_F \in \mathbb{D}$ then set $x_{int} = x_F$, $k = k + 1$ and go to Step 2; otherwise $i = i + 1$, go to Step 4.

Step 6. If $\mu \leq U_\mu$, then $\mu = \mu + 5$ and go to Step 2; otherwise take x_k^* as a global minimizer of $f(x)$ and stop.

The different set of directions d_i are given as follows: let $\theta_1, \dots, \theta_J \in [0, 2\pi]$ and $\vartheta_1, \dots, \vartheta_J \in [-\frac{\pi}{2}, \frac{\pi}{2}]$, J 's are uniformly distributed. For each direction $d_i, i = 1, 2, \dots, N$ and $j = 1, 2, \dots, J$. If the dimension n is even and $n = 2Q$ we calculate the contents of $d_i^j = (y_1^j, y_2^j, \dots, y_{2Q}^j)$ from

$$y_{2l-1}^j = \frac{\sqrt{2}}{\sqrt{n}} \cos(\theta_j)$$

$$y_{2l}^j = \frac{\sqrt{2}}{\sqrt{n}} \sin(\theta_j)$$

for $l = 1 \sim Q$. If n is odd and $n = 2Q + 1$ we calculate the contents of $d_i^j = (y_1^j, y_2^j, \dots, y_{2Q+1}^j)$ from

$$y_1^j = \frac{\sqrt{2}}{\sqrt{n}} \cos(\vartheta_j) \cos(\theta_j)$$

$$y_2^j = \frac{\sqrt{2}}{\sqrt{n}} \cos(\vartheta_j) \sin(\theta_j)$$

$$y_3^j = \frac{\sqrt{2}}{\sqrt{n}} \sin(\vartheta_j)$$

$$y_{2l}^j = \frac{\sqrt{2}}{\sqrt{n}} \cos(\theta_j)$$

$$y_{2l+1}^j = \frac{\sqrt{2}}{\sqrt{n}} \sin(\theta_j)$$

for $l = 2 \sim Q$ (Wang and Fan, 2010).



6. APPLICATION FOR GLOBAL OPTIMIZATION ALGORITHMS

Results for the proposed algorithms were presented in Sections 3, 4 and 5 which have been applied in this Section to a set of common test problems. A computer program code was created for each of the global optimization methods addressed in this study. For each algorithm, the obtained results are shown in detail in the related tables. Therefore, according to the obtained results, the proposed methods were compared with each other and then compared to some other available methods shown in the corresponding tables, the results of these comparisons are presented. To define algorithms for global optimization, the following abbreviations are used:

NSA : New smoothing auxiliary function (Algorithm which defined in 3.3).

DSA : Directional search algorithm (Algorithm which defined in 4.3).

FSA : Filled function algorithm (Algorithm which defined in 5.3).

6.1. Test Problems

The numerical test of the NSA, DSA and FSA algorithms was performed on the following test problems:

Problem 1-3. dimensional function

$$\min f(x) = [1 - 2x_2 + c \sin(4\pi x_2) - x_1]^2 + [x_2 - 0.5 \sin(2\pi x_1)]^2,$$

for $x_1, x_2 \in [-3, 3]$, where $c = 0.05, 0.2, 0.5$.

Problem 4. Three-hump back camel function

$$\min f(x) = 2x_1^2 - 1.05x_1^4 + \frac{1}{6}x_1^6 - x_1x_2 + x_2^2,$$

for $x_1, x_2 \in [-3, 3]$.

Problem 5. Six-hump back camel function

$$\min f(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 - x_1x_2 - 4x_2^4 + 4x_2^4,$$

for $x_1, x_2 \in [-3, 3]$.

Problem 6. Treccani function

$$\min f(x) = x_1^4 + 4x_1^3 + 4x_1^2 + x_2^2,$$

for $x_1, x_2 \in [-3, 3]$.

Problem 7. Goldstein and Price function

$$\min f(x) = g_1(x)g_2(x),$$

where

$$g_1(x) = 1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2),$$

and

$$g_2(x) = 30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2),$$

for $x_1, x_2 \in [-3, 3]$.

Problem 8. Shubert function

$$\min f(x) = \left\{ \sum_{i=1}^5 i \cos[(i+1)x_1 + i] \right\} \left\{ \sum_{i=1}^5 i \cos[(i+1)x_2 + i] \right\},$$

s. t. $x_1, x_2 \in [-10, 10]$.

Problems 9. n -dimensional function

$$\min f(x) = \frac{\pi}{n} [10 \sin^2 \pi x_1 + g(x) + (x_n - 1)^2],$$

where $g(x) = \sum_{i=1}^{n-1} \left[(x_i - 1)^2 (1 + 10 \sin^2 \pi x_{i+1}) \right]$ and $x_i \in [-10, 10], i = 1, 2, \dots, n$.

Problem 10. (BL_2) Beale function)

$$\min f(x) = (1.5 - x_1 + x_1x_2)^2 + (2.25 - x_1 + x_1x_2^2)^2 + (2.625 - x_1 + x_1x_2^3)^2,$$

s. t. $x_1, x_2 \in [-4.5, 4.5]$.

Problem 11. (B_2) Bohachevsky 1 function

$$\min f(x) = x_1^2 + 2x_2^2 - 0.3 \cos(3\pi x_1) - 0.4 \cos(4\pi x_2) + 0.7,$$

s. t. $x_1, x_2 \in [-100, 100]$.

Problem 12. (B_2) Bohachevsky 2 function

$$\min f(x) = x_1^2 + 2x_2^2 - 0.3 \cos(3\pi x_1) \cos(4\pi x_2) + 0.3,$$

s. t. $x_1, x_2 \in [-100, 100]$.

Problem 13. (B_2) Bohachevsky 3 function

$$\min f(x) = x_1^2 + 2x_2^2 - 0.3 \cos(3\pi x_1 + 4\pi x_2) + 0.3,$$

s. t. $x_1, x_2 \in [-100, 100]$.

Problem 14. (BO_2) Booth function

$$\min f(x) = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2,$$

s. t. $x_1, x_2 \in [-10, 10]$.

Problem 15. (RT_n) Rastrigin function

$$\min f(x) = 20 + \sum_{i=1}^n \left(x_i^2 - 10 \cos(2\pi x_i) \right),$$

for $x_i \in [-5.12, 5.12], i = 1, \dots, n$.

Problem 16. (RC_n) Branin function

$$\min f(x) = (x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6)^2 + 10(1 - \frac{1}{8\pi})\cos(x_1) + 10,$$

for $x_1 \in [-5, 10], x_2 \in [0, 15]$.

Problem 17. (MT_2) Matyas function

$$\min f(x) = 0.26(x_1^2 + x_2^2) - 0.48x_1x_2,$$

s. t. $x_1, x_2 \in [-10, 10]$.

Problem 18. (H_3) Hartmann function

$$\min f(x) = -\sum_{i=1}^4 b_i \exp\left(-\sum_{j=1}^3 A_{ij}(x_j - Q_{ij})^2\right),$$

where

$$b = (1.0, 1.2, 3.0, 3.2)^T$$

$$A = \begin{pmatrix} 3.0 & 10 & 30 \\ 0.1 & 10 & 35 \\ 3.0 & 10 & 30 \\ 0.1 & 10 & 35 \end{pmatrix}$$

$$Q = 10^{-4} \begin{pmatrix} 3689 & 1170 & 2673 \\ 4699 & 4387 & 7470 \\ 1091 & 8732 & 5547 \\ 381 & 5743 & 8828 \end{pmatrix}$$

s. t. $x_i \in [0, 1], i = 1, 2, 3..$

Problem 19. (CV_4) Colville function

$$\min f(x) = 100(x_1^2 - x_2)^2 + (x_3 - 1)^2 + 90(x_3^2 - x_4)^2$$

$$+10.1((x_2 - 1)^2 + (x_4 - 1)^2) + 19.8(x_2 - 1)(x_4 - 1),$$

s. t. $x_i \in [-10, 10], i = 1, \dots, 4.$

Problem 20. (P_n) Perm function

$$\min f(x) = \sum_{i=1}^n \left(\sum_{j=1}^n (j^i + b) \left(\left(\frac{x_j}{j} \right)^i - 1 \right) \right),$$

s. t. $x_i \in [-n, n], i = 1, \dots, n.$

Problem 21. (PS_n) Power sum function

$$\min f(x) = \sum_{i=1}^n \left(\left(\sum_{j=1}^n x_j^i \right) - b_i \right)^2,$$

s. t. $x_i \in [0, n], i = 1, \dots, n.$

Problems 22-24. (S_4) Shekel function

$$\min f(x) = - \sum_{i=1}^m \left(\sum_{j=1}^4 (x_j - a_{i,j})^2 + b_i \right)^{-1},$$

where $m = 5, 7, 10$, b_i is an m -dimensional vector, and $a_{i,j}$ is a $4 \times m$ -dimensional matrix where

$$b_i = 0.1 \begin{pmatrix} 1 & 2 & 2 & 4 & 4 & 6 & 3 & 7 & 5 & 5 \end{pmatrix},$$

$$a_{i,j} = \begin{pmatrix} 4.0 & 1.0 & 8.0 & 6.0 & 3.0 & 2.0 & 5.0 & 8.0 & 6.0 & 7.0 \\ 4.0 & 1.0 & 8.0 & 6.0 & 7.0 & 9.0 & 3.0 & 1.0 & 2.0 & 3.6 \\ 4.0 & 1.0 & 8.0 & 6.0 & 3.0 & 2.0 & 5.0 & 8.0 & 6.0 & 7.0 \\ 4.0 & 1.0 & 8.0 & 6.0 & 7.0 & 9.0 & 3.0 & 1.0 & 2.0 & 3.6 \end{pmatrix},$$

and $x_j \in [0, 10], j = 1, \dots, 4.$

Problem 25. (H_6) Hartmann function

$$\min f(x) = - \sum_{i=1}^4 b_i \exp \left(- \sum_{j=1}^6 A_{ij} (x_j - Q_{ij})^2 \right),$$

where

$$b = (1.0, 1.2, 3.0, 3.2)^T$$

$$A = \begin{pmatrix} 10 & 3.0 & 17 & 3.50 & 1.7 & 8.0 \\ 0.05 & 10 & 17 & 0.1 & 8.0 & 14 \\ 3.0 & 3.5 & 1.7 & 10 & 17 & 8.0 \\ 17 & 8.0 & 0.05 & 10 & 0.1 & 14 \end{pmatrix}$$

$$Q = 10^{-4} \begin{pmatrix} 1312 & 1696 & 5569 & 124 & 8283 & 5886 \\ 2329 & 4135 & 8307 & 3736 & 1004 & 9991 \\ 2348 & 1451 & 3522 & 2883 & 3047 & 6650 \\ 4047 & 8828 & 8732 & 5743 & 1091 & 381 \end{pmatrix}$$

s. t. $x_i \in [0, 1], i = 1, \dots, 6$.

Problem 26. (T_n) Trid function

$$\min f(x) = \sum_{i=1}^n (x_i - 1)^2 - \sum_{i=1}^n x_i x_{i-1},$$

s. t. $x_i \in [-n^2, n^2], i = 1, \dots, n$.

Problem 27. (AK_n) Ackley function

$$\min f(x) = -a \exp \left(-b \sqrt{\frac{1}{n} \sum_{i=1}^n x_i^2} \right) - \exp \left(\frac{1}{n} \sum_{i=1}^n \cos(cx_i) \right) + a + \exp(1),$$

where $a = 20, b = 0.2, c = 2\pi$ and $x_i \in [-32.768, 32.768], i = 1, \dots, n$.

Problem 28. (DP_n) Dixon and Price function

$$\min f(x) = (x_1 - 1)^2 + \sum_{i=2}^n i(2x_i^2 - x_{i-1})^2,$$

s. t. $x_i \in [-10, 10], i = 1, \dots, n$.

Problem 29. (G_n) Griewank function

$$\min f(x) = \sum_{i=1}^n \frac{x_i^2}{4000} - \prod_{i=1}^n \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1,$$

s. t. $x_i \in [-600, 600], i = 1, \dots, n.$

Problems 30-37. (L_n) Levy function

$$\begin{aligned} \min f(x) = & \sin^2(\pi w_1) + \sum_{i=1}^{n-1} (w_i - 1)^2 \left[1 + 10 \sin^2(\pi w_i + 1) \right] \\ & + (w_n - 1)^2 \left[1 + \sin^2(2\pi w_n) \right] \end{aligned}$$

where $w_i = 1 + \frac{x_{i-1}}{4}, i = 1, \dots, n$ and $x_i \in [-10, 10], i = 1, 2, \dots, n.$

Problem 38. (PW_n) Powell function

$$\begin{aligned} \min f(x) = & \sum_{i=1}^{\frac{n}{4}} \left((x_{4i-3} + 10x_{4i-2})^2 + 5(x_{4i-1} - x_{4i})^2 \right. \\ & \left. + (x_{4i-2} - 2x_{4i-1})^4 + 10(x_{4i-3} - x_{4i})^4 \right), \end{aligned}$$

s. t. $x_i \in [-4, 5], i = 1, \dots, n.$

Problem 39. (R_n) Rosenbrock function

$$\min f(x) = \sum_{i=1}^{n-1} \left(100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2 \right),$$

s. t. $x_i \in [-5.12, 5.12], i = 1, \dots, n.$

Problem 40. (SR_n) Sphere function

$$\min f(x) = \sum_{i=1}^n x_i^2,$$

s. t. $x_i \in [-5.12, 5.12], i = 1, \dots, n.$

Problem 41-48. (SS_n) Sum squares function

$$\min f(x) = \sum_{i=1}^n ix_i^2,$$

s. t. $x_i \in [-10, 10], i = 1, \dots, n.$

Problem 49. (Z_n) Zakharov function

$$\min f(x) = \sum_{i=1}^n x_i^2 + \left(\sum_{i=1}^n 0.5ix_i \right)^2 + \left(\sum_{i=1}^n 0.5ix_i \right)^4,$$

s. t. $x_i \in [-10, 10], i = 1, \dots, n.$

The problems above are rearranged and the list of test problems is shown in Table 6.1.

6.2. Applications on Test problems

In this part, the algorithms of NSA, DSA, and FSA are applied to problems 1-49 presented in Table 1. For each of these problems, these algorithms are implemented by independently evaluating ten different points as starting points and uniformly selecting those points from the domain \mathbb{D} on an Intel(R) Core(TM) (i7-3687U CPU and 2.60 GHz) computer in Matlab R2016a. The terms used in the tables are as follows:

- *No.*: the number of the problem,
- *iter_m*: the average number of iteration in the 10 runs,
- *f_{eval}*: the average number of evaluations of functions,
- *time*: the total runtime average in 10 runs (second),
- *f_{mean}*: the average of the function values in 10 runs,
- *f_{best}*: the best performance of function value in 10 runs,
- *Succ*: the success rate of different starting points between 10 implementation.

The NSA, DSA, and FSA algorithms have been applied to test problems 1-49 with dimensions ranging from 2-50 which are listed in Tables 6.2, 6.4, and 6.3 respectively. Ten different points were tested as a starting point for each problem, 50% and upwards were achieved through an analysis of the results of these algorithms to reach the global point at which the overall success rates were 87.77% for NSA, 89.38% for DSA, and 84.69% for FSA starting from these different points. In addition, the average of execution time, the average number of iterations, the average function values, and the

Table 6.1. The list of test problems

<i>No.</i>	<i>n</i>	Problem name	Optimum value	Region
1	2	Two-dimensional function $c = 0.05$	0	$[-3, 3]^2$
2	2	Two-dimensional function $c = 0.2$	0	$[-3, 3]^2$
3	2	Two-dimensional function $c = 0.5$	0	$[-3, 3]^2$
4	2	Three-hump back camel function	0	$[-3, 3]^2$
5	2	Six-hump back camel function	-1.0316	$[-3, 3]^2$
6	2	Treccani function	0	$[-3, 3]^2$
7	2	Goldstein and Price function	3.0000	$[-3, 3]^2$
8	2	Shubert function	-186.73091	$[-10, 10]^2$
9	2	n -dimensional function	0	$[-10, 10]^2$
10	2	(BL_2) Beale function	0	$[-4.5, 4.5]^2$
11	2	(B_2) Bohachevsky 1 function	0	$[-100, 100]^2$
12	2	(B_2) Bohachevsky 2 function	0	$[-100, 100]^2$
13	2	(B_2) Bohachevsky 3 function	0	$[-100, 100]^2$
14	2	(BO_2) Booth function	0	$[-10, 10]^2$
15	2	(RT_n) Rastrigin function	0	$[-5.12, 5.12]^2$
16	2	(RC_n) Branin function	0.3979	$[-5, 10] \times [0, 15]$
17	2	(MT_2) Matyas function	0	$[-10, 10]^2$
18	3	(H_3) Hartmann function	-3.8628	$[0, 1]^3$
19	4	(CV_4) Colville function	0	$[-10, 10]^4$
20	4	(P_n) Perm function	0	$[-4, 4]^4$
21	4	(PS_n) Power sum function	0	$[0, 4]^4$
22	4	(S_4) Shekel function	-10.1532	$[0, 10]^4$
23	4	(S_4) Shekel function	-10.4029	$[0, 10]^4$
24	4	(S_4) Shekel function	-10.5364	$[0, 10]^4$
25	6	(H_6) Hartmann function	-3.3224	$[0, 1]^6$
26	10	(T_n) Trid function	-210	$[-100, 100]^2$
27	10	(AK_n) Ackley function	0	$[-10, 10]^{10}$
28	25	(DP_n) Dixon and Price function	0	$[-10, 10]^{25}$
29	30	(G_n) Griewank function	0	$[-600, 600]^{30}$
30,31,32,33	2,3,5,7	(L_n) Levy function	0	$[-10, 10]^n$
34,35,36,37	10,20,30,50	(L_n) Levy function	0	$[-10, 10]^n$
38	30	(PW_n) Powell function	0	$[-4, 5]^3 0$
39	50	(R_n) Rosenbrock function	0	$[-5.12, 5.12]^{50}$
40	50	(SR_n) Sphere function	0	$[-5.12, 5.12]^{50}$
41,42,43,44	2,3,5,7	(SS_n) Sum squares function	0	$[-10, 10]^n$
45,46,47,48	10,20,30,50	(SS_n) Sum squares function	0	$[-10, 10]^n$
49	50	(Z_n) Zakharov function	0	$[-5, 5]^{50}$

average function evaluation were determined, as well as the best value for the function was included in these ten different points. These values are different from one problem to another according to the type of the problem and its size.

Comparison is made of the average iterations ($iter_m$), the average function evaluations (f_{eval}) and the average execution time (time) for test problems for the proposed algorithms. It can be seen that all the proposed algorithms have achieved successful results in test problems. By observing the results for some of the above fields, there are some advantages from one algorithm to another, as the DSA algorithm achieved a success rate of about 89.38 percent of the number of attempts and also a lower percentage for the average of function evaluations compared to others. Whereas in the field of (time), the FSA algorithm has the advantage as shown in Table 6.5. As mentioned before the DSA algorithm has a preference over the others because the DSA algorithm is presented to solve the (P) problem by converting a multi-dimensional problem into one-dimensional problem partitions and using an auxiliary function to reduce the number of minimizers and finding a global minimizer for each partition. Then use these partitions to find the global minimizer of a multidimensional problem. Since the test problem is divided from a multidimensional problem to one dimension problem, it means that there are no more function evaluations, and this is one of the DSA algorithm's features.

Tables 6.6, 6.7, and 6.8 provide a summary of the various algorithms described in this study. Table 6.6 provides a comparison of the NSA with the algorithm described in (Bagirov et al. (2009)), the results of the NSA algorithm can be seen from this table in some places, especially in the columns dedicated to function evaluations (f_{eval}) and success rates ($Succ$) compared to the results of the algorithm in (Bagirov et al. (2009)). Both of the methods are fairly efficient with respect to values (f_{best}). Table 6.7 offers a comparison of DSA with the algorithm provided in (Wei et al. (2014)), from this table the DSA algorithm results obtain an advantage in a column dedicated to function evaluations (f_{eval}). The algorithm described in Table 6.7 has an advantage in a column devoted to execution time (time). All methods in terms of (f_{best}) values and iterations ($iter_m$) are efficient enough. Whereas Table 6.8 of the algorithm offers an advantage for FSA in columns dedicated to function evaluations (f_{eval}) and success rates ($Succ$), especially in broad problem sizes.

Table 6.2. The results of NSA algorithm on problems 1-49

<i>No</i>	<i>n</i>	<i>iter_m</i>	<i>f_{eval}</i>	<i>f_{mean}</i>	<i>f_{best}</i>	<i>time</i>	<i>Succ</i>
1	2	2	187	1.4536e-14	6.5432e-16	0.0911	8/10
2	2	2.5	216	2.3610e-13	2.4535e-14	0.1196	10/10
3	2	1	303	1.2796e-11	3.3378e-15	0.1257	9/10
4	2	1	675	1.6748e-14	0	0.0907	8/10
5	2	2	358	-1.0316	-1.0316	0.0464	10/10
6	2	2	145	3.7911e-10	1.1915e-16	0.0204	10/10
7	2	1.5	460	3.0000	3.0000	0.0623	9/10
8	2	3.5	1400	-186.7309	-186.7309	0.8857	10/10
9	2	2.25	4044	9.7321e-14	8.6220e-15	0.5452	10/10
10	2	1	274	1.2552e-08	4.1242e-14	0.0427	10/10
11	2	3.25	280	1.1634e-13	2.5887e-15	0.9828	7/10
12	2	2.25	306	4.6210e-13	2.6872e-17	0.9619	9/10
13	2	1.5	540	5.4220e-14	8.9750e-16	1.0623	8/10
14	2	1	258	1.2778e-10	1.0089e-14	0.0389	10/10
15	2	2	845	2.2649e-14	0	0.1110	8/10
16	2	1	213	0.3979	0.3979	0.0285	10/10
17	2	1	193	7.4099e-14	4.0478e-16	0.0254	10/10
18	3	1	484	-3.8628	-3.8628	0.0468	8/10
19	4	2.25	910	2.5446e-06	6.0306e-13	0.0758	10/10
20	4	1	1426	1.4358e-04	2.9421e-06	0.3263	6/10
21	4	1	1733	3.5915e-04	1.7632e-07	1.8948	6/10
22	4	2.75	1772	-10.1532	-10.1532	0.1777	10/10
23	4	2.5	1353	-10.4029	-10.4029	0.1429	10/10
24	4	2.5	1436	-10.5321	-10.5321	0.1515	10/10
25	6	1	585	-3.0425	-3.0425	0.0406	10/10
26	10	1	888	-210	-210	0.3263	6/10
27	10	6.5	2912	8.5594e-10	5.3731e-10	0.1774	10/10
28	25	1	55728	3.8734e-8	5.1243e-11	1.9834	6/10
29	30	1	22981	2.8754e-13	1.8112e-13	0.9761	6/10
30	2	3	502	5.0804e-13	4.9451e-17	0.0846	10/10
31	3	2.5	938	1.7220e-12	4.6827e-15	0.1241	8/10
32	5	5.25	1721	2.9544e-13	9.5472e-15	0.1726	7/10
33	7	6.5	2529	3.2636e-12	9.3510e-16	0.2084	6/10
34	10	3	2856	3.7409e-13	7.4937e-15	0.2115	7/10
35	20	2.5	6813	5.6417e-13	7.3256e-15	0.4574	9/10
36	30	4	10810	2.3839e-13	2.1908e-15	0.7524	10/10
37	50	6.5	20445	2.2094e-12	5.6789e-14	1.6759	8/10
38	30	1	3403	4.8804e-06	9.3569e-10	0.2217	10/10
39	50	1	14561	1.1169e-07	7.6706e-11	0.4321	6/10
40	50	1	5523	3.3790e-10	2.3738e-15	0.1943	10/10
41	2	1	129	7.1418e-14	2.5517e-14	0.0184	10/10
42	3	1	244	1.2744e-13	5.4187e-14	0.0289	10/10
43	5	1	600	1.9263e-11	1.0445e-12	0.0528	10/10
44	7	1.5	825	2.4339e-11	9.1360e-13	0.0554	10/10
45	10	1	1262	2.6180e-10	8.3906e-13	0.0737	10/10
46	20	1	2608	8.0341e-10	2.3277e-11	0.1050	10/10
47	30	1.25	3915	1.0642e-08	1.1326e-10	0.1344	10/10
48	50	1	6864	3.3977e-08	4.8552e-10	0.2142	10/10
49	50	1.5	8506.8	5.6232e-06	1.0476e-10	0.2706	10/10

Table 6.3. The results of DSA algorithm on problems 1-49

<i>No</i>	<i>n</i>	<i>iter_m</i>	<i>f_{eval}</i>	<i>f_{mean}</i>	<i>f_{best}</i>	<i>time</i>	<i>Succ</i>
1	2	1	145	2.8106e-12	1.2929e-15	0.0589	10/10
2	2	1.25	120	8.9169e-13	8.1862e-16	0.0699	10/10
3	2	1.15	192	1.1776e-14	7.7555e-16	0.2073	10/10
4	2	1.2	297	3.0290e-16	1.0703e-16	0.1214	10/10
5	2	1.5	114	-1.0316	-1.0316	0.1162	10/10
6	2	2	140	1.2374e-16	9.1354e-18	0.1122	10/10
7	2	1.75	162	3.0000	3.0000	0.1803	9/10
8	2	5.57	207	-186.7309	-186.7309	0.2517	8/10
9	2	1.9	154	6.1757e-13	1.7833e-15	0.1355	10/10
10	2	2.95	146	5.6903e-14	2.0928e-14	0.1180	10/10
11	2	4.4	125	1.1968e-11	1.5543e-15	0.1437	10/10
12	2	3.45	118	2.5438e-13	4.9405e-15	0.1422	9/10
13	2	3.2	102	9.7353e-13	1.1657e-15	0.1514	10/10
14	2	2.4	138	9.7144e-14	2.5354e-15	0.0543	10/10
15	2	2.3	248	3.2330e-13	2.4869e-14	0.1909	6/10
16	2	1.4	152	0.3979	0.3979	0.0638	10/10
17	2	1	137	5.4987e-14	2.6805e-16	0.0686	10/10
18	3	1.35	111	-3.8628	-3.8628	0.0742	10/10
19	4	1.25	187	5.8521e-10	3.3950e-13	0.1918	10/10
20	4	1.25	599	1.9779e-05	8.4586e-06	0.9533	7/10
21	4	1	275	0.0161	0.0161	0.2288	10/10
22	4	1.75	310	-10.1532	-10.1532	0.1812	8/10
23	4	1.5	638	-10.4029	-10.4029	0.4108	9/10
24	4	1.87	667	-10.5321	-10.5321	0.3929	8/10
25	6	1	183	-3.0425	-3.0425	0.0785	10/10
26	10	1	645	-210	-210	0.1690	9/10
27	10	4.5	347	4.6465e-06	7.9323e-07	0.5138	8/10
28	25	1.3	765	3.5502e-05	3.3802e-10	1.2282	6/10
29	30	1.59	1551	2.8975e-11	2.1213e-15	1.4813	5/10
30	2	1.5	189	1.3085e-13	6.7715e-17	0.1011	10/10
31	3	1.7	227	6.2925e-14	1.8777e-15	0.2456	9/10
32	5	2.7	437	2.9336e-13	2.5302e-15	0.3301	8/10
33	7	1.7	350	4.6065e-13	1.8490e-15	0.2431	6/10
34	10	5.3	512	2.8582e-13	1.1873e-15	0.7028	6/10
35	20	1.7	418	1.0709e-12	4.8271e-16	0.4228	7/10
36	30	3.3	759	1.4902e-12	9.7442e-16	1.0371	7/10
37	50	5.6	950	1.7661e-14	2.5524e-15	1.8705	6/10
38	30	1.2	269	1.7461e-09	1.2717e-13	0.6275	10/10
39	50	1	540	7.3042e-05	6.5167e-11	1.9243	7/10
40	50	1	649	2.7006e-16	5.9498e-18	0.2883	10/10
41	2	1.3	110	3.8924e-14	3.2620e-17	0.0683	10/10
42	3	1	150	5.5689e-14	9.9508e-16	0.0814	10/10
43	5	1.5	177	2.8539e-12	7.6193e-16	0.1072	10/10
44	7	1	216	2.7366e-11	6.3709e-13	0.1612	10/10
45	10	3.7	265	9.9834e-12	3.4184e-13	0.2110	10/10
46	20	2.75	313	5.7654e-10	8.7459e-12	0.3173	10/10
47	30	1.6	388	1.1967e-09	2.4248e-11	0.5861	10/10
48	50	1	479	6.3092e-09	1.3800e-10	1.0965	10/10
49	50	5.2	478	3.7096e-07	1.0042e-10	0.7815	10/10

Table 6.4. The results of FSA algorithm on problems 1-49

<i>No</i>	<i>n</i>	<i>iter_m</i>	<i>f_{eval}</i>	<i>f_{mean}</i>	<i>f_{best}</i>	<i>time</i>	<i>Succ</i>
1	2	1	414	5.5023e-13	1.0416e-15	0.0951	7/10
2	2	1.3	426	5.0637e-13	2.4897e-16	0.0949	10/10
3	2	3.25	312	2.1450e-12	2.4059e-15	0.0798	8/10
4	2	1.28	297	4.2567e-14	2.3059e-16	0.0652	8/10
5	2	1.8	249	-1.0316	-1.0316	0.0690	10/10
6	2	1	195	1.8911e-10	2.3324e-16	0.0547	10/10
7	2	1.25	324	3.0000	3.0000	0.0799	8/10
8	2	5.5	990	-186.7309	-186.7309	0.1781	10/10
9	2	2.25	216	1.3655e-12	8.0334e-15	0.0527	10/10
10	2	1	210	2.1415e-09	3.9292e-14	0.0473	10/10
11	2	2.5	260	1.7612e-14	1.3887e-16	0.8727	6/10
12	2	2	290	5.1310e-15	7.6872e-17	0.8615	9/10
13	2	2.5	560	2.1322e-13	3.8751e-15	0.9623	7/10
14	2	1	207	2.5312e-12	2.9352e-15	0.0258	10/10
15	2	1	252	1.3269e-12	0	0.0373	10/10
16	2	1	192	0.3979	0.3979	0.0291	10/10
17	2	1	183	1.1829e-13	3.2168e-17	0.0257	10/10
18	3	1	360	-3.8628	-3.8628	0.0358	8/10
19	4	1	340	4.2280e-06	1.7297e-11	0.0289	9/10
20	4	1	930	4.4280e-06	4.4280e-08	0.0733	6/10
21	4	1	1445	0.0161	0.0161	0.1310	10/10
22	4	1	2550	-10.1532	-10.1532	0.2107	7/10
23	4	1	1550	-10.4029	-10.4029	0.1395	6/10
24	4	1.5	2090	-10.5321	-10.5321	0.2020	8/10
25	6	1	343	-3.0425	-3.0425	0.0287	10/10
26	10	1	1243	-210	-210	0.4378	6/10
27	10	5.4	1100	1.2101e-07	4.4409e-15	0.0714	10/10
28	25	6	5122	1.4691e-8	3.8934e-11	0.1791	5/10
29	30	1.5	20119	1.1213e-10	6.7832e-13	1.0812	5/10
30	2	2.5	249	4.4883e-12	2.8451e-16	0.0326	10/10
31	3	3.57	340	4.9828e-12	4.8602e-16	0.0370	7/10
32	5	2.57	1560	9.6303e-14	7.2187e-15	0.1503	7/10
33	7	2.66	2552	1.8162e-12	1.2567e-14	0.1835	6/10
34	10	3	2662	3.0363e-13	2.8589e-16	0.1632	5/10
35	20	2.57	7602	4.8446e-12	7.4225e-15	0.4800	8/10
36	30	2.33	9579	4.1103e-12	7.4225e-15	0.4570	6/10
37	50	4	6528	1.2428e-11	9.3490e-15	0.4241	8/10
38	30	1	3265	5.6936e-06	1.0917e-09	0.1742	10/10
39	50	1.25	13566	1.1169e-07	1.8954e-11	0.1594	5/10
40	50	1	5253	1.3804e-11	1.2346e-16	0.1943	10/10
41	2	1	210	7.1418e-14	2.5517e-14	0.0304	10/10
42	3	1.25	300	1.2744e-13	5.4187e-14	0.0293	10/10
43	5	1	577	1.7427e-11	3.2479e-14	0.0614	10/10
44	7	1.5	1144	1.4534e-11	2.7828e-14	0.0679	10/10
45	10	1	1584	3.0956e-10	2.5392e-14	0.0786	10/10
46	20	1.5	4515	6.1714e-10	6.0884e-12	0.1985	10/10
47	30	2	12834	8.5200e-09	9.4700e-12	0.4910	10/10
48	50	1.5	21624	3.3257e-08	4.6143e-11	0.6605	10/10
49	50	1	7497	4.1186e-06	1.3670e-10	0.2318	10/10

Table 6.5. Iteration, function evaluations and execution time for NSA, DSA, and FSA algorithms.

<i>No</i>	<i>n</i>	NSA			DSA			FSA		
		<i>iter_m</i>	<i>f_{eval}</i>	<i>tim</i>	<i>iter_m</i>	<i>f_{eval}</i>	<i>time</i>	<i>iter_m</i>	<i>f_{eval}</i>	<i>tim</i>
1	2	2	187	0.0911	1	145	0.0589	1	414	0.0951
2	2	2.5	216	0.1196	1.25	120	0.0589	1.3	426	0.0949
3	2	1	303	0.1257	1.15	192	0.2073	3.25	312	0.0798
4	2	1	675	0.0907	1.2	297	0.1214	1.28	297	0.0652
5	2	2	358	0.0464	1.5	114	0.1162	1.8	249	0.0690
6	2	2	145	0.0204	2	140	0.1122	1	195	0.0547
7	2	1.5	460	0.0623	1.75	162	0.1803	1.25	324	0.0799
8	2	3.5	1400	0.8857	5.57	207	0.2517	5.5	990	0.1781
9	2	2.25	4044	0.5452	1.9	154	0.1355	2.25	216	0.0527
10	2	1	274	0.0427	2.95	146	0.1180	1	210	0.0473
11	2	3.25	280	0.9828	4.4	125	0.1437	2.5	260	0.8727
12	2	2.25	306	0.9619	3.45	118	0.1422	2	290	0.8615
13	2	1.5	540	1.0623	3.2	102	0.1514	2.5	560	0.9623
14	2	1	258	0.0389	2.4	138	0.0543	1	207	0.0258
15	2	2	845	0.1110	2.3	248	0.1909	1	252	0.0373
16	2	1	213	0.0285	1.4	152	0.0638	1	192	0.0291
17	2	1	193	0.0254	1	137	0.0686	1	183	0.0257
18	3	1	484	0.0468	1.35	111	0.0742	1	360	0.0358
19	4	2.25	910	0.0758	1.25	187	0.1918	1	340	0.0289
20	4	1	1426	0.3263	1.25	599	0.9533	1	930	0.0733
21	4	1	1733	1.8948	1	275	0.2288	1	1445	0.1310
22	4	2.75	1772	0.1777	1.75	310	0.1812	1	2550	0.2107
23	4	2.5	1353	0.1429	1.5	638	0.4108	1	1550	0.1395
24	4	2.5	1436	0.1515	1.87	667	0.3929	1.5	2090	0.2020
25	6	1	585	0.0406	1	183	0.0785	1	343	0.0287
26	10	1	888	0.3263	1	645	0.1690	1	1243	0.4378
27	10	6.5	2912	0.1774	4.5	347	0.5138	2.5	1100	0.0714
28	25	1	55728	0.1774	1.3	765	1.2282	6	5122	0.1791
29	30	1	22981	0.9761	1.59	1551	1.4813	1.5	20119	1.0812
30	2	3	502	0.0846	1.5	189	0.1011	2.5	249	0.0326
31	3	2.5	938	0.1241	1.7	227	0.2456	3.57	340	0.0370
32	5	5.25	1721	0.1726	2.7	437	0.3301	2.57	1560	0.1503
33	7	6.5	2529	0.2084	1.7	350	0.2431	2.66	2552	0.1835
34	10	3	2856	0.2115	5.3	512	0.7028	3	2662	0.1632
35	20	2.5	6813	0.4574	1.7	418	0.4228	2.57	7602	0.4800
36	30	4	10810	0.7524	3.3	759	1.0371	2.33	9579	0.4570
37	50	6.5	20445	1.6759	5.6	950	1.8705	4	6528	0.4241
38	30	1	3403	0.2217	1.2	269	0.6275	1	3265	0.1742
39	50	1	1456	0.4321	1	540	1.9243	1.25	13566	0.1594
40	50	1	5523	0.1943	1	649	0.2883	1	5253	0.1943
41	2	1	129	0.0184	1.3	110	0.0683	1	210	0.0304
42	3	1	244	0.0289	1	150	0.0814	1.25	300	0.0293
43	5	1	600	0.0528	1.5	177	0.1072	1	577	0.0614
44	7	1.5	825	0.0554	1	216	0.1612	1.5	1144	0.0679
45	10	1	1262	0.0737	3.7	265	0.2110	1	1584	0.0786
46	20	1	2608	0.1050	2.75	313	0.3173	1.5	4515	0.1985
47	30	1.25	3915	0.1344	1.6	388	0.5861	2	12834	0.4910
48	50	1	6864	0.2142	1	479	1.0965	1.5	21624	0.6605
49	50	1.5	8506	0.2706	5.2	478	0.7815	1	7497	0.2318

Table 6.6. Comparison NSA algorithms with the algorithm in (Bagirov et al. (2009)).

<i>No</i>	<i>n</i>	NSA				Algorithm in (Bagirov et al. (2009))		
		<i>iter_m</i>	<i>f_{eval}</i>	<i>f_{best}</i>	<i>Succ</i>	<i>f_{eval}</i>	<i>f_{best}</i>	<i>Succ</i>
1	2	2	187	6.543E-16	8/10	nil	nil	nil
2	2	2.5	216	2.454E-14	10/10	nil	nil	nil
3	2	1	303	3.338E-15	9/10	nil	nil	nil
4	2	1	675	0	8/10	nil	nil	nil
5	2	2	358	-1.0316	10/10	1783	-1.03163	10/10
6	2	2	145	1.192E-16	10/10	nil	nil	nil
7	2	1.5	460	3.0000	9/10	7032	3.0000	10/10
8	2	3.5	1400	-186.7309	10/10	2732	-185.4678	10/10
9	2	2.25	4044	8.622E-15	10/10	nil	nil	nil
10	2	1	274	4.124E-14	10/10	nil	nil	nil
11	2	3.25	280	2.589E-15	7/10	1879	0.0000	6/10
12	2	2.25	306	2.687E-17	9/10	1917	0.0000	7/10
13	2	1.5	540	8.975E-16	8/10	1672	0.0000	7/10
14	2	1	258	1.009E-14	10/10	2053	0.0000	9/10
15	2	2	845	0	8/10	4859	0.0000	8/10
16	2	1	213	0.3979	10/10	nil	nil	nil
17	2	1	193	4.048E-16	10/10	nil	nil	nil
18	3	1	484	-3.8628	8/10	nil	nil	nil
19	4	2.25	910	6.031E-13	10/10	nil	nil	nil
20	4	1	1426	2.942E-06	6/10	nil	nil	nil
21	4	1	1733	1.763E-07	6/10	nil	nil	nil
22	4	2.75	1772	-10.1532	10/10	nil	nil	nil
23	4	2.5	1353	-10.4029	10/10	nil	nil	nil
24	4	2.5	1436	-10.5321	10/10	nil	nil	nil
25	6	1	585	-3.0425	10/10	nil	nil	nil
26	10	1	888	-210	6/10	nil	nil	nil
27	10	6.5	2912	5.373E-10	10/10	nil	nil	nil
28	25	1	55728	5.124E-11	6/10	nil	nil	nil
29	30	1	22981	1.811E-13	6/10	106290	0.0172	10/10
30	2	3	502	4.945E-17	10/10	3038	0.4354	8/10
31	3	2.5	938	4.683E-15	8/10	nil	nil	nil
32	5	5.25	1721	9.547E-15	7/10	16574	0.1368	8/10
33	7	6.5	2529	9.351E-16	6/10	nil	nil	nil
34	10	3	2856	7.494E-15	7/10	42393	0.0187	9/10
35	20	2.5	6813	7.326E-15	9/10	nil	nil	nil
36	30	4	10810	2.191E-15	10/10	143294	0.0000	10/10
37	50	6.5	20445	5.679E-14	8/10	336367	0.0137	8/10
38	30	1	3403	9.357E-10	10/10	nil	nil	nil
39	50	1	14561	7.671E-11	6/10	nil	nil	nil
40	50	1	5523	2.374E-15	10/10	nil	nil	nil
41	2	1	129	2.552E-14	10/10	3362	0.0311	10/10
42	3	1	244	5.419E-14	10/10	nil	nil	nil
43	5	1	600	1.045E-12	10/10	41825	0.0000	10/10
44	7	1.5	825	9.136E-13	10/10	nil	nil	nil
45	10	1	1262	8.391E-13	10/10	90110	0.0000	10/10
46	20	1	2608	2.328E-11	10/10	nil	nil	nil
47	30	1.25	3915	1.133E-10	10/10	321748	0.000003	10/10
48	50	1	6864	4.855E-10	10/10	342501	0.0011	9/10
49	50	1.5	8506.8	1.048E-10	10/10	nil	nil	nil

Table 6.7. Comparison DSA algorithms with the algorithm in (Wei et al. (2014)) .

<i>No</i>	<i>n</i>	DSA				Algorithm in (Wei et al. (2014))			
		<i>iter_m</i>	<i>f_{eval}</i>	<i>f_{best}</i>	<i>time</i>	<i>iter_m</i>	<i>f_{eval}</i>	<i>f_{best}</i>	<i>time</i>
1	2	1	145	1.2929E-15	0.0589	3	255	8.6721e-18	0.241103
2	2	1.25	120	8.1862E-16	0.0699	2	297	0	0.085445
3	2	1.15	192	7.7555E-16	0.2073	1	374	2.1518e-10	0.348272
4	2	1.2	297	1.0703E-16	0.1214	1	74	3.6795e-15	0.046905
5	2	1.5	114	-1.0316	0.1162	2	74	-1.0316	0.062735
6	2	2	140	9.1354E-18	0.1122	2	72	0	0.050563
7	2	1.75	162	3.0000	0.1803	3	80	3.0000	0.185288
8	2	5.57	207	-186.7309	0.2517	4	78	-186.7309	0.056097
9	2	1.9	154	1.783E-15	0.1355	6	634	0	0.046059
10	2	2.95	146	2.0928E-14	0.118	nil	102	0	nil
11	2	4.4	125	1.5543E-15	0.1437	nil	226	0	nil
12	2	3.45	118	4.9405E-15	0.1422	nil	nil	nil	nil
13	2	3.2	102	1.1657E-15	0.1514	nil	nil	nil	nil
14	2	2.4	138	2.5354E-15	0.0543	nil	77	0	nil
15	2	2.3	248	2.4869E-14	0.1909	nil	nil	nil	nil
16	2	1.4	152	0.3979	0.0638	nil	202	0.3979	nil
17	2	1	137	2.6805E-16	0.0686	nil	207	0	nil
18	3	1.35	111	-3.8628	0.0742	nil	3241	-38628	nil
19	4	1.25	187	3.395E-13	0.1918	nil	167	0	nil
20	4	1.25	599	8.4586E-06	0.9533	nil	103	3.1750e-08	nil
21	4	1	275	0.0161	0.2288	nil	103	7.9352e-09	nil
22	4	1.75	310	-10.1532	0.1812	nil	116	-10.1532	nil
23	4	1.5	638	-10.4029	0.4108	nil	209	-10.4029	nil
24	4	1.87	667	-10.5321	0.3929	nil	507	-10.5364	nil
25	6	1	183	-3.0425	0.0785	nil	827	-3.3224	nil
26	10	1	645	-210.0000	0.169	nil	642	-210	nil
27	10	4.5	347	7.9323E-07	0.5138	nil	647	4.4409e-15	nil
28	25	1.3	765	3.3802E-10	1.2282	nil	16266	6.6175e-03	nil
29	30	1.59	1551	2.1213E-15	1.4813	nil	643	2.6527e-16	nil
30	2	1.5	189	6.7715E-17	0.1011	nil	nil	nil	nil
31	3	1.7	227	1.8777E-15	0.2456	nil	nil	nil	nil
32	5	2.7	437	2.5302E-15	0.3301	nil	nil	nil	nil
33	7	1.7	350	1.849E-15	0.2431	nil	nil	nil	nil
34	10	5.3	512	1.1873E-15	0.7028	nil	nil	nil	nil
35	20	1.7	418	4.8271E-16	0.4228	nil	nil	nil	nil
36	30	3.3	759	9.7442E-16	1.0371	nil	6206	1.8424e-10	nil
37	50	5.6	950	2.5524E-15	1.8705	nil	nil	nil	nil
38	30	1.2	269	1.2717E-13	0.6275	nil	681517	6.6014e-09	nil
39	50	1	540	6.5167E-11	1.9243	nil	316406	3.2891e-10	nil
40	50	1	649	5.9498E-18	0.2883	nil	6215	1.3952e-09	nil
41	2	1.3	110	3.262E-17	0.0683	nil	nil	nil	nil
42	3	1	150	9.9508E-16	0.0814	nil	nil	nil	nil
43	5	1.5	177	7.6193E-16	0.1072	nil	nil	nil	nil
44	7	1	216	6.3709E-13	0.1612	nil	nil	nil	nil
45	10	3.7	265	3.4184E-13	0.211	nil	nil	nil	nil
46	20	2.75	313	8.7459E-12	0.3173	nil	nil	nil	nil
47	30	1.6	388	2.4248E-11	0.5861	nil	nil	nil	nil
48	50	1	479	1.38E-10	1.0965	nil	6277	5.9246e-12	nil
49	50	5.2	478	1.0042E-10	0.7815	nil	316356	3.8675e-03	nil

Table 6.8. Comparison FSA algorithms with the algorithm in (Sahiner et al. (2019)).

No	n	FSA				(Sahiner et al. (2019))			
		$iter_m$	f_{eval}	f_{best}	Succ	$iter_m$	f_{eval}	f_{best}	Succ
1	2	1	414	1.0416E-15	7/10	1.5	214	2.6630E - 154	8/10
2	2	1.3	426	2.4897E-16	10/10	1.13	290.6250	3.4336E - 16	8/10
3	2	3.25	312	2.4059E-15	8/10	1.75	414.2857	4.7243E - 16	8/10
4	2	1.28	297	2.3059E-16	8/10	1.4	411	2.8802E - 16	10/10
5	2	1.8	249	-1.0316	10/10	1.5	234	-1.0316	10/10
6	2	1	195	2.3324E-16	10/10	1.	216.5000	1.6477E - 15	10/10
7	2	1.25	324	3	8/10	1.22	487.8889	3.0000	9/10
8	2	5.5	990	-186.7309	10/10	2.7	813.5000	-186.7309	10/10
9	2	2.25	216	8.0334E-15	10/10	nil	nil	nil	nil
10	2	1	210	3.9292E-14	10/10	nil	nil	nil	nil
11	2	2.5	260	1.3887E-16	6/10	nil	nil	nil	nil
12	2	2	290	7.6872E-17	9/10	nil	nil	nil	nil
13	2	2.5	560	3.8751E-15	7/10	nil	nil	nil	nil
14	2	1	207	2.9352E-15	10/10	nil	nil	nil	nil
15	2	1	252	0	10/10	nil	nil	nil	nil
16	2	1	192	0.3979	10/10	nil	nil	nil	nil
17	2	1	183	3.2168E-17	10/10	nil	nil	nil	nil
18	3	1	360	-3.8628	8/10	nil	nil	nil	nil
19	4	1	340	1.7297E-11	9/10	nil	nil	nil	nil
20	4	1	930	4.428E-08	6/10	nil	nil	nil	nil
21	4	1	1445	0.0161	10/10	nil	nil	nil	nil
22	4	1	2550	-10.1532	7/10	nil	nil	nil	nil
23	4	1	1550	-10.4029	6/10	nil	nil	nil	nil
24	4	1.5	2090	-10.5321	8/10	nil	nil	nil	nil
25	6	1	343	-3.0425	10/10	nil	nil	nil	nil
26	10	1	1243	-210	6/10	nil	nil	nil	nil
27	10	5.4	1100	4.4409E-15	10/10	nil	nil	nil	nil
28	25	6	5122	3.8934E-11	5/10	nil	nil	nil	nil
29	30	1.5	20119	6.7832E-13	5/10	nil	nil	nil	nil
30	2	2.5	249	2.8451E-16	10/10	nil	nil	nil	nil
31	3	3.57	340	4.8602E-16	7/10	nil	nil	nil	nil
32	5	2.57	1560	7.2187E-15	7/10	nil	nil	nil	nil
33	7	2.66	2552	1.2567E-14	6/10	nil	nil	nil	nil
34	10	3	2662	2.8589E-16	5/10	nil	nil	nil	nil
35	20	2.57	7602	7.4225E-15	8/10	nil	nil	nil	nil
36	30	2.33	9579	7.4225E-15	6/10	nil	nil	nil	nil
37	50	4	6528	9.349E-15	8/10	nil	nil	nil	nil
38	30	1	3265	1.0917E-09	10/10	nil	nil	nil	nil
39	50	1.25	13566	1.8954E-11	5/10	nil	nil	nil	nil
40	50	1	5253	1.2346E-16	10/10	nil	nil	nil	nil
41	2	1	210	2.5517E-14	10/10	2.75	743.2500	9.4192E - 15	8/10
42	3	1.25	300	5.4187E-14	10/10	1.9	3027	5.6998E - 15	10/10
43	5	1	577	3.2479E-14	10/10	1.8	4999.3	3.7007E - 15	10/10
44	7	1.5	1144	2.7828E-14	10/10	1.75	8171	1.3790E - 14	8/10
45	10	1	1584	2.5392E-14	10/10	2.78	8895.4	3.0992eE - 14	9/10
46	20	1.5	4515	6.0884E-12	10/10	2.71	18242	3.0016E - 13	7/10
47	30	2	12834	9.47E-12	10/10	3.5	43232	1.7361E - 12	6/10
48	50	1.5	21624	4.6143E-11	10/10	2.5	83243	9.8531E - 13	6/10
49	50	1	7497	1.367E-10	10/10	nil	nil	nil	nil

7. TOTAL VARIATION APPLICATION IN IMAGE DENOISING

Image denoising is a central field in computer vision and image processing. Most technologies are used for the preservation, recovery or transfer of images, so that noise from the image can somehow be eliminated during these operations and that the original image can be retrieved. All images in size $n \times n$ are considered. Let $x' \in R^{n^2}$ is the original image, and let $\delta \in R^{n^2}$ is the noise of Gaussian. Any image $y \in R^{n^2}$ degraded can be described below

$$y = x' + \delta. \quad (7.1)$$

In order to remove the noise from the image, the inverse of the model in (7.1) can be taken with considering that the regularization techniques try to integrate both the terms data-fidelity and the model in (7.1) into a same objective function and the appropriate solutions for this function can be considered as a minimization problem as the following:

$$\min_x \{ \|y - x\| + \lambda J(x) \} \quad (7.2)$$

where $J(x)$ is a regularization term and λ is a positive scalar that acts as an equalizer in (7.2). For regularization techniques, the main issue to consider is the correct choice for the suggested image model and sufficient mathematical definition. Literature includes several different schemes to define the suggested image (Bruckstein et al., 2009; Ao et al., 2013; Phillips, 1962). TV-function is one of the most important regularization functions due to its ability to recover edges and maintain image texture. TV-function was first implemented by Rudin, Osher, and Fatemi (Rudin and Osher, 1994) as an image denoising regularizer, then extended and used in deblurring (Beck and Teboulle, 2009; Chan et al., 2013) and segmentation (He et al., 2012; Chan et al., 2006). Depending on the TV-function feature equation (7.2) can be written as follows:

$$\min_x \{ \|y - x\| + \lambda TV(x) \}, \quad (7.3)$$

and

$$TV(x) = \int \|D(x)\| dx = \sum_i \sqrt{(D_i^h x)^2 + (D_i^v x)^2},$$

where D_i^h, D_i^v denote first-order horizontal and vertical finite-difference of x at $i - th$ pixels. The TV-function defined in (7.3) is convex and this function makes the minimizer unique, but the problem is that this function is non-differentiable due to the nature of the norm-term, which is difficult to minimize and its gradient flow is not well-defined.

This challenge forced researchers to find alternatives (Chambolle, 2005; Chan et al., 1999). There are various methods to solve the TV-function, see (Rudin and Osher, 1994; Marquina and Osher, 2000) for more detail. A gradient descent method is one of the significant methods proposed to solve optimization with a double formulation for denoising TV-function in several different studies (Chambolle, 2004; Guo et al., 2009; Yang et al., 2009). In this Section, we use the method defined in Section (3) to make TV-function differentiable, and then we use it as an application for image denoising.

7.1. Theoretical part

To make TV-function differentiable let $\|v\| = \|D(x)\|$, and in one-dimension $\|v\| = |v|$. From the equation (3.1), if $f(x) = v$, $h(x) = -v$ then

$$\varphi(v) = |v| = \frac{1}{2} \{ (v - (-v))\omega(v) + v + (-v) \} = v\omega(v),$$

where the function $\omega(v) : R \rightarrow R$ is defined by

$$\omega(v) = \begin{cases} 1, & v \geq 0, \\ -1, & v < 0. \end{cases}$$

Because the $\omega(v)$ function is non-smooth it is clear that the $\varphi(v)$ function is non-smooth too. In order to smooth the $\varphi(v)$ function, it is necessary for $\omega(v)$ to be smooth. We use equation (3.2) to make $\varphi(v)$ as follows:

$$\tilde{\varphi}_\sigma(v) = v\tilde{\omega}_\sigma(v), \tag{7.4}$$

where

$$\tilde{\omega}_\sigma(v) = \frac{2}{1 + \exp(-\frac{1}{2\sigma}v)} - 1.$$

Fig. 7.1 displays the diagram of the functions $\tilde{\varphi}_\sigma(v)$ and $\varphi(v)$ in one-dimensions with different σ values.

The following results are given in accordance with the above features of the functions $\tilde{\omega}_\sigma(v)$ and $\tilde{\varphi}_\sigma(v)$.

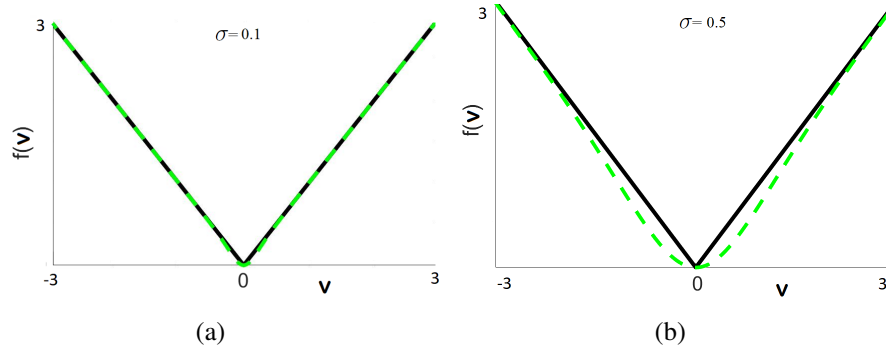


Figure 7.1. The graph of $\phi(u)$ (black and solid) and $\tilde{\phi}_\sigma(u)$ (green and dashed) with different σ .

Lemma 7.1. Let $\tilde{\omega}_\sigma(v)$ and $\omega(v)$ functions, then for any $\sigma > 0$

$$\|\tilde{\omega}_\sigma(v) - \omega(v)\|_{L^1} \leq \frac{17}{3}\sigma.$$

Proof. See Lemma 3.1. □

Theorem 7.2. Suppose that $\tilde{\varphi}_\sigma(v)$ function is a smoothing function of $\varphi(v)$, then

$$\|\tilde{\varphi}_\sigma(v) - \varphi(v)\|_{L^1} \leq 13.16\sigma^2.$$

Proof. See Theorem 3.2. □

Theorem 7.3. Assume that $\tilde{\varphi}_\sigma(v)$ is a smoothing function of $\varphi(v)$, then

$$\lim_{\sigma \rightarrow 0} \tilde{\varphi}_\sigma(v) = \varphi(v).$$

Proof. See Theorem 3.3. □

7.2. Smoothed TV-function for Denoising

We use the results and the properties of the previous part to make the TV-function smooth. As we described before, we have

$$TV(x) = \int \|D(x)\| dx,$$

the function gradient can be computed as

$$\nabla TV(x) = \text{div}\left(\frac{D(x)}{\|D(x)\|}\right).$$

The TV-function gradient is not defined if one has $D(x) = 0$ at a pixel x . This means that it is hard to minimize the TV-function and its gradient flow is not well defined. To overcome this problem, instead of a smooth TV function, we consider

$$TV_\sigma(x) = \int D(x)\tilde{S}(x)dx \quad (7.5)$$

and

$$\tilde{S}(x) = \frac{2}{1 + \exp\left(-\frac{1}{2\sigma}D(x)\right)} - 1,$$

where $\sigma > 0$. The gradient of the smoothed TV-function is

$$\nabla TV_\sigma(x) = \text{div}\left(\frac{\sigma \exp\left(\frac{1}{\sigma}D(x)\right) + x \exp\left(\frac{1}{2\sigma}D(x)\right) - \sigma}{\sigma \left(\exp\left(\frac{1}{2\sigma}D(x)\right) + 1\right)^2}\right).$$

Now, the problem set out in (7.3) can be reformulated using smooth TV-function as

$$\min_x \{\|y - x\| + \lambda TV_\sigma(x)\} \quad (7.6)$$

By using (7.6) with the help of a gradient descent minimization method as in the next algorithm, we can obtain a good image denoising solution.

7.3. Algorithm

Step 1- Set $k = 0$, choose $\sigma > 0$, $\lambda > 0$, $\varepsilon > 0$ as a stopping condition, and $\tau > 0$ as a step.

Step 2- Let $x_1 = y$, y is a noisy image.

Step 3- Calculate a better solution $x^{(k)+1}$ by

$$x^{(k)+1} = x_1 - \tau(y - x_1 + \lambda \nabla TV_\sigma(x_1)).$$

Step 4- If $\|x^{(k)+1} - x_1\| > \varepsilon$ then take $x_1 = x^{(k)+1}$, set $k = k + 1$ and go to (**Step 3**); else stop the algorithm and go to (**Step 5**).

Step 5- Take $x^{(k)+1}$ as the best solution of image denoising operations.

We use some test images size of 256×256 as shown in Fig. 7.2 to show the advantages of the presented method (NSTV). Fig. 7.3 shows the gradient of the smooth TV-function with different σ).

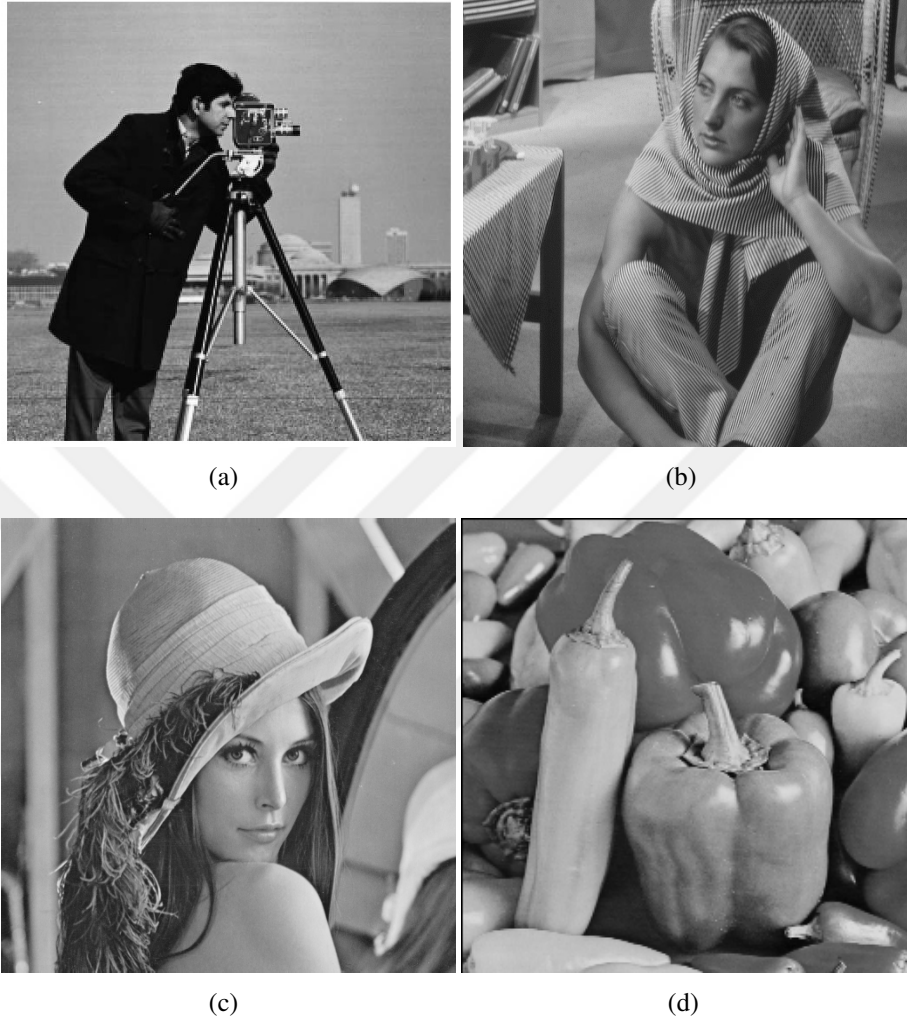


Figure 7.2. Test images (a) Cameraman, (b) Barbara, (c) Lena and (d) Pepper

The denoising results are shown in Figs. 7.4, 7.5, 7.6 and 7.7. The efficiency of the results is calculated in Table 7.1 by peak signal-noise ratio (PSNR) with noise deviation $\rho = 20, 25$ and the higher PSNR the higher quality of the restoration. We compared proposed (NSTV) with some current image restore algorithms, such as FASTA denoising TV (Beck and Teboulle, 2009), the SA-DCT method in (Foi et al., 2007), the non-sampled contourlet denoising method in (Da Cunha et al., 2006)(DNS-CT), and generalized TV regularization (Wu et al., 2017) (GTV). The proposed method shows some advantages in PSNR.

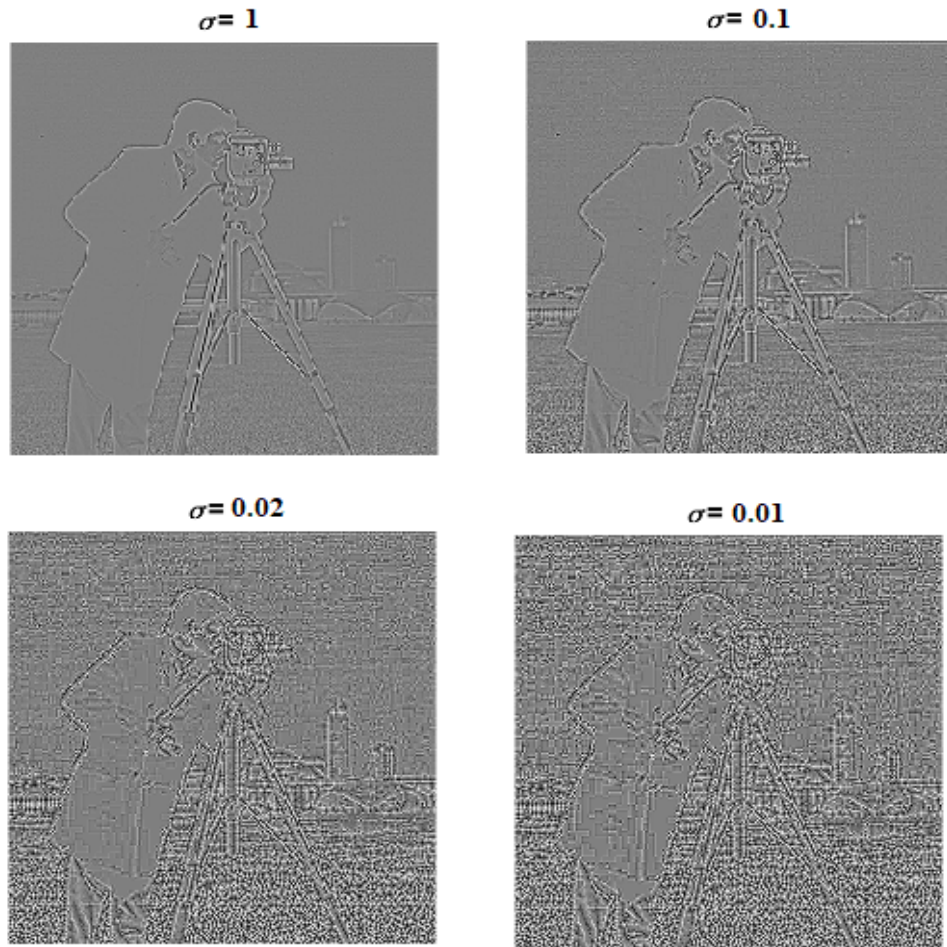


Figure 7.3. Smoothed TV-function gradient.

Table 7.1. The PSNR value by various methods (unit: dB).

Methods	$\rho = 20$				$\rho = 25$			
	Cameraman	Barbara	Lena	Peppers	Cameraman	Barbara	Lena	Peppers
NSTV	29.4642	29.7295	30.3214	30.1947	29.7547	28.5112	29.6008	29.4844
TVF	28.6591	29.3121	29.4092	30.2731	27.5397	27.0854	28.3415	28.4686
SA-DCT	30.0145	30.8906	30.9197	31.8967	28.8963	29.0789	29.7450	29.9224
DNSCT	28.4210	29.1285	29.1131	29.9971	27.3357	26.9872	28.2382	28.2696
GTV	30.3569	31.9690	31.6027	31.7693	29.0712	29.7826	30.5201	30.2162



(a)



(b)



(c)



(d)



(e)



(f)

Figure 7.4. The denoising experiment with Cameraman image. (a) Noisy image. (b) TVF. (c) SA-DCT. (d) DNSCT. (e) GTV. (f) NSTV.



(a)



(b)



(c)



(d)



(e)



(f)

Figure 7.5. The denoising experiment with Barbara image. (a) Noisy image. (b) TVF. (c) SA-DCT. (d) DNSCT. (e) GTV. (f) NSTV.



(a)



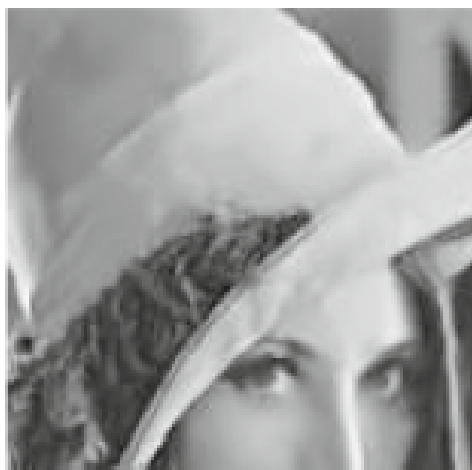
(b)



(c)



(d)

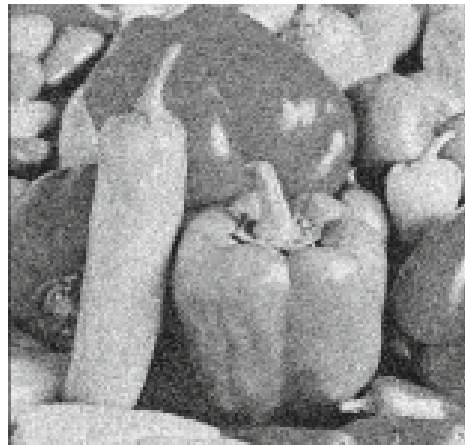


(e)



(f)

Figure 7.6. The denoising experiment with Lena image. (a) Noisy image. (b) TVF. (c) SA-DCT. (d) DNSCT. (e) GTV. (f) NSTV.



(a)



(b)



(c)



(d)



(e)



(f)

Figure 7.7. The denoising experiment with Peppers image. (a) Noisy image. (b) TVF. (c) SA-DCT. (d) DNSCT. (e) GTV. (f) NSTV.

8. Conclusion

In this study, we have presented numerous algorithms aimed at solving a wide range of global optimization problems and non-smoothing functions. Stochastic and heuristic methods are easy and fast in the application, but they have no guarantees in obtaining the global optimal point. Deterministic methods are more effective and reliable, this thesis focused on deterministic global optimization methods. We presented three algorithms of global optimization, namely New Smoothing Auxiliary Function (NSA), Directional Search Algorithm (DSA), and Filled Function Algorithm (FSA). The preliminary results were very encouraging for all those algorithms.

The NSA algorithm was introduced in Section 3 and includes two parts, a new smoothing approximation technique for non-smooth functions and the new smoothing auxiliary method by using the same technique to solve unconstrained global optimization problems with multi-model. This algorithm is efficient for non-smooth functions and solves multi-model global optimization problems.

In Section 4, the DSA algorithm was introduced, this algorithm presented a new technique of global optimization by reducing a multidimensional problem to a one-dimensional problem. The results of the calculation and comparison with NSA and FSA or with another algorithm showed that this algorithm obtained impressive results as can be seen from the tables concerned. As a future work, we can extend this algorithm by adding some restricted on the domain or by reducing the number of directions to be more efficient.

In Section 5, a new filled function for unconstrained global optimization was presented to the FSA algorithm. The proposed filled function contains two parameters, which can be easily adjusted in the process of minimization. Furthermore, it was based on numerical tests to show the efficacy of the algorithm presented. From empirical results, it can be seen that the present approach is promising. This algorithm is an effective approach to solve global multi-modal optimization issues. This algorithm is based on putting many stationary points in the lower basin. This idea helps to reduce the time and function evaluations in the minimizing process. These two important properties make it an advantageous algorithm among the other methods.

Image denoising is an important computer vision field. The NSA algorithm was used in

Section 7 as a real-problem to render TV-function smooth and then apply it in image denoising. This algorithm has been shown to be efficient in image denoising capability. For the future work, all of the algorithms in this thesis can be applied in the future as an extension study to many real-life issues such as data mining, chemical processes, aerospace industries, and image processing such as deblurring and segmentation.



REFERENCES

- Adjiman, C.S., Dallwig, S., Floudas, C.A., Neumaier, A., 1998. A global optimization method, α BB, for general twice-differentiable constrained NLPs-I. Theoretical advances. *Computers & Chemical Engineering*, 22(9), 1137–1158.
- Akay, B., Karaboga, D., 2012. A modified artificial bee colony algorithm for real-parameter optimization. *Information sciences*, 192, 120–142.
- Alefeld, G., Herzberger, J., 2012. *Introduction to interval computation*. Academic press.
- Anderssen, R., Bloomfield, P., 1975. Properties of the random search in global optimization. *Journal of Optimization Theory and Applications*, 16(5-6), 383–398.
- Androulakis, I.P., Maranas, C.D., Floudas, C.A., 1995. α BB: A global optimization method for general constrained nonconvex problems. *Journal of Global Optimization*, 7(4), 337–363.
- Ao, L., Yibing, L., Xiaodong, Y., Yue, L., 2013. Image restoration with dual-prior constraint models based on Split Bregman. *Optical Review*, 20(6), 491–495.
- Bagirov, A.M., Rubinov, A.M., Zhang, J., 2009. A multidimensional descent method for global optimization. *Optimization*, 58(5), 611–625.
- Basso, P., 1982. Iterative methods for the localization of the global maximum. *SIAM Journal on Numerical Analysis*, 19(4), 781–792.
- Beck, A., Teboulle, M., 2009. Fast gradient-based algorithms for constrained total variation image denoising and deblurring problems. *IEEE transactions on image processing*, 18(11), 2419–2434.
- Bertsekas, D.P., 1975. *Nondifferentiable optimization via approximation*. *Nondifferentiable optimization*, Springer. (pp. 1–25).
- Bruckstein, A.M., Donoho, D.L., Elad, M., 2009. From sparse solutions of systems of equations to sparse modeling of signals and images. *SIAM review*, 51(1), 34–81.
- Cetin, B., Barhen, J., Burdick, J., 1993. Terminal repeller unconstrained subenergy tunneling (TRUST) for fast global optimization. *Journal of Optimization Theory and Applications*, 77(1), 97–126.
- Chambolle, A., 2004. An algorithm for total variation minimization and applications. *Journal of Mathematical imaging and vision*, 20(1-2), 89–97.
- Chambolle, A., 2005. Total variation minimization and a class of binary MRF models. *International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*. Springer, (pp. 136–152).

- Chan, R.H., Tao, M., Yuan, X., 2013. Constrained total variation deblurring models and fast algorithms based on alternating direction method of multipliers. *SIAM Journal on imaging Sciences*, 6(1), 680–697.
- Chan, T.F., Esedoglu, S., Nikolova, M., 2006. Algorithms for finding global minimizers of image segmentation and denoising models. *SIAM journal on applied mathematics*, 66(5), 1632–1648.
- Chan, T.F., Golub, G.H., Mulet, P., 1999. A nonlinear primal-dual method for total variation-based image restoration. *SIAM journal on scientific computing*, 20(6), 1964–1977.
- Chen, Y., Wan, Z., 2015. A locally smoothing method for mathematical programs with complementarity constraints. *The ANZIAM Journal*, 56(3), 299–315.
- Chowdhury, P.R., Singh, Y.P., Chansarkar, R., 2000. Hybridization of gradient descent algorithms with dynamic tunneling methods for global optimization. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 30(3), 384–390.
- Da Cunha, A.L., Zhou, J., Do, M.N., 2006. The nonsubsamped contourlet transform: theory, design, and applications. *IEEE transactions on image processing*, 15(10), 3089–3101.
- Ekren, O., Ekren, B.Y., 2010. Size optimization of a PV/wind hybrid energy conversion system with battery storage using simulated annealing. *Applied energy*, 87(2), 592–598.
- Floudas, C.A., Visweswaran, V., 1990. A global optimization algorithm (GOP) for certain classes of nonconvex NLPs-I. Theory. *Computers & chemical engineering*, 14(12), 1397–1417.
- Floudas, C.A., Visweswaran, V., 1993. Primal-relaxed dual global optimization approach. *Journal of Optimization Theory and Applications*, 78(2), 187–225.
- Foi, A., Katkovnik, V., Egiazarian, K., 2007. Pointwise shape-adaptive DCT for high-quality denoising and deblocking of grayscale and color images. *IEEE transactions on image processing*, 16(5), 1395–1411.
- Garg, H., 2016. A hybrid PSO-GA algorithm for constrained optimization problems. *Applied Mathematics and Computation*, 274, 292–305.
- Ge, R., Qin, Y., 1987. A class of filled functions for finding global minimizers of a function of several variables. *Journal of Optimization Theory and Applications*, 54(2), 241–252.
- Grimstad, B., Sandnes, A., 2016. Global optimization with spline constraints: a new branch-and-bound method based on B-splines. *Journal of Global Optimization*, 65(3), 401–439.

- Griva, I., Nash, S.G., Sofer, A., 2009. Linear and nonlinear optimization, Volume 108. Siam.
- Groenen, P.J., Heiser, W.J., 1996. The tunneling method for global optimization in multidimensional scaling. *Psychometrika*, 61(3), 529–550.
- Guo, X., Li, F., Ng, M.K., 2009. A fast 1-TV algorithm for image restoration. *SIAM Journal on Scientific Computing*, 31(3), 2322–2341.
- Hansen, E., Walster, G.W., 2003. Global optimization using interval analysis: revised and expanded, Volume 264. CRC Press.
- He, Y., Hussaini, M.Y., Ma, J., Shafei, B., Steidl, G., 2012. A new fuzzy c-means method with total variation regularization for segmentation of images with noisy and incomplete data. *Pattern Recognition*, 45(9), 3463–3471.
- Huyer, W., Neumaier, A., 1999. Global optimization by multilevel coordinate search. *Journal of Global Optimization*, 14(4), 331–355.
- Jones, D.R., Perttunen, C.D., Stuckman, B.E., 1993. Lipschitzian optimization without the Lipschitz constant. *Journal of optimization Theory and Applications*, 79(1), 157–181.
- Karaboga, D., Basturk, B., 2007. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. *Journal of global optimization*, 39(3), 459–471.
- Kennedy, J., 2010. Particle swarm optimization. *Encyclopedia of machine learning*, 760–766.
- Lera, D., Sergeyev, Y.D., 2015. Deterministic global optimization using space-filling curves and multiple estimates of Lipschitz and Hölder constants. *Communications in Nonlinear Science and Numerical Simulation*, 23(1-3), 328–342.
- Levy, A.V., Montalvo, A., 1985. The tunneling algorithm for the global minimization of functions. *SIAM Journal on Scientific and Statistical Computing*, 6(1), 15–29.
- Lin, H., Gao, Y., Wang, Y., 2014. A continuously differentiable filled function method for global optimization. *Numerical Algorithms*, 66(3), 511–523.
- Lin, H., Wang, Y., Gao, Y., Wang, X., 2018. A filled function method for global optimization with inequality constraints. *Computational and Applied Mathematics*, 37(2), 1524–1536.
- Liu, H., Wang, Y., Guan, S., Liu, X., 2017. A new filled function method for unconstrained global optimization. *International Journal of Computer Mathematics*, 94(12), 2283–2296.
- Liu, J., Zhang, S., Wu, C., Liang, J., Wang, X., Teo, K.L., 2016. A hybrid approach to constrained global optimization. *Applied Soft Computing*, 47, 281–294.

- Liu, X., 2001. Finding global minima with a computable filled function. *Journal of Global Optimization*, 19(2), 151–161.
- Mahi, M., Baykan, Ö.K., Kodaz, H., 2015. A new hybrid method based on particle swarm optimization, ant colony optimization and 3-opt algorithms for traveling salesman problem. *Applied Soft Computing*, 30, 484–490.
- Marquina, A., Osher, S., 2000. Explicit algorithms for a new time dependent model based on level set motion for nonlinear deblurring and noise removal. *SIAM Journal on Scientific Computing*, 22(2), 387–405.
- Niknam, T., Amiri, B., Olamaei, J., Arefi, A., 2009. An efficient hybrid evolutionary optimization algorithm based on PSO and SA for clustering. *Journal of Zhejiang University-SCIENCE A*, 10(4), 512–519.
- Phillips, D.L., 1962. A technique for the numerical solution of certain integral equations of the first kind. *Journal of the ACM (JACM)*, 9(1), 84–97.
- Poli, R., Kennedy, J., Blackwell, T., 2007. Particle swarm optimization. *Swarm intelligence*, 1(1), 33–57.
- Ralph, D., Xu, H., 2005. Implicit smoothing and its application to optimization with piecewise smooth equality constraints. *Journal of optimization theory and applications*, 124(3), 673–699.
- Renpu, G., 1990. A filled function method for finding a global minimizer of a function of several variables. *Mathematical programming*, 46(1-3), 191–204.
- Rudin, L.I., Osher, S., 1994. Total variation based image restoration with free local constraints. *Proceedings of 1st International Conference on Image Processing. IEEE, Volume 1*, (pp. 31–35).
- Sahiner, A., Gokkaya, H., Yigit, T., 2012. A new filled function for nonsmooth global optimization. *AIP Conference Proceedings. AIP, Volume 1479*, (pp. 972–974).
- Sahiner, A., Ibrahim, S.A., 2019. A new global optimization technique by auxiliary function method in a directional search. *Optimization Letters*, 13(2), 309–323.
- Sahiner, A., Yilmaz, N., Ibrahim, S.A., 2018. Smoothing Approximations to Non-smooth Functions. *Journal of Multidisciplinary Modeling and Optimization*.
- Sahiner, A., Yilmaz, N., Kapusuz, G., 2017. A descent global optimization method based on smoothing techniques via Bezier curves. *Carpathian Journal of Mathematics*, 33(3), 373–380.
- Sahiner, A., Yilmaz, N., Kapusuz, G., 2019. A novel modeling and smoothing technique in global optimization. *Journal of Industrial & Management Optimization*, 15(1), 113–130.
- Samora, I., Franca, M.J., Schleiss, A.J., Ramos, H.M., 2016. Simulated annealing in

- optimization of energy production in a water supply network. *Water resources management*, 30(4), 1533–1547.
- Schäffler, S., 2012. *Global optimization: a stochastic approach*. Springer Science & Business Media.
- Shang, Y.I., Pu, D.g., Jiang, A.p., 2007. Finding global minimizer with one-parameter filled function on unconstrained global optimization. *Applied Mathematics and Computation*, 191(1), 176–182.
- Storti, G.L., Paschero, M., Rizzi, A., Mascioli, F.M.F., 2015. Comparison between time-constrained and time-unconstrained optimization for power losses minimization in smart grids using genetic algorithms. *Neurocomputing*, 170, 353–367.
- Suman, B., Kumar, P., 2006. A survey of simulated annealing as a tool for single and multiobjective optimization. *Journal of the operational research society*, 57(10), 1143–1160.
- Wang, W., Zhang, X., Li, M., et al., 2015. A filled function method dominated by filter for nonlinearly global optimization. *Journal of Applied Mathematics*, 2015.
- Wang, Y., Fan, L., 2010. A smoothing evolutionary algorithm with circle search for global optimization. 2010 Fourth International Conference on Network and System Security. IEEE, (pp. 412–418).
- Wei, F., Wang, Y., 2013. A new filled function method with one parameter for global optimization. *Mathematical Problems in Engineering*, 2013.
- Wei, F., Wang, Y., Lin, H., 2014. A new filled function method with two parameters for global optimization. *Journal of Optimization Theory and Applications*, 163(2), 510–527.
- Wu, H., Zhang, P., Lin, G.H., 2015. Smoothing approximations for some piecewise smooth functions. *Journal of the Operations Research Society of China*, 3(3), 317–329.
- Wu, Q., Li, Y., Lin, Y., 2017. The application of nonlocal total variation in image denoising for mobile transmission. *Multimedia Tools and Applications*, 76(16), 17179–17191.
- Wu, Z., Zhang, L., Teo, K., Bai, F., 2005. New modified function method for global optimization. *Journal of Optimization Theory and Applications*, 125(1), 181–203.
- Wu, Z.Y., Bai, F., Lee, H.J., Yang, Y., 2007. A filled function method for constrained global optimization. *Journal of Global Optimization*, 39(4), 495–507.
- Wu, Z.Y., Lee, H.J., Zhang, L.S., Yang, X., 2006. A novel filled function method and quasi-filled function method for global optimization. *Computational Optimization and Applications*, 34(2), 249–272.

- Wu, Z.Y., Li, D., Zhang, L.S., 2011. Global descent methods for unconstrained global optimization. *Journal of Global Optimization*, 50(3), 379–396.
- Xavier, A.E., 2010. The hyperbolic smoothing clustering method. *Pattern Recognition*, 43(3), 731–737.
- Xiao, Y., Yu, B., 2010. A truncated aggregate smoothing Newton method for minimax problems. *Applied Mathematics and Computation*, 216(6), 1868–1879.
- Xu, S., 2001. Smoothing method for minimax problems. *Computational Optimization and Applications*, 20(3), 267–279.
- Xu, Y.T., Zhang, Y., Wang, S.G., 2015. A modified tunneling function method for non-smooth global optimization and its application in artificial neural network. *Applied Mathematical Modelling*, 39(21), 6438–6450.
- Xu, Z., Huang, H.X., Pardalos, P.M., Xu, C.X., 2001. Filled functions for unconstrained global optimization. *Journal of Global Optimization*, 20(1), 49–65.
- Yang, J., Yin, W., Zhang, Y., Wang, Y., 2009. A fast algorithm for edge-preserving variational multichannel image restoration. *SIAM Journal on Imaging Sciences*, 2(2), 569–592.
- Yang, Y., Pang, L., Ma, X., Shen, J., 2014. Constrained nonconvex nonsmooth optimization via proximal bundle method. *Journal of Optimization Theory and Applications*, 163(3), 900–925.
- Yilmaz, N., Sahiner, A., 2017. New global optimization method for non-smooth unconstrained continuous optimization. *AIP Conference Proceedings*. AIP Publishing, Volume 1863, (s. 250002).
- Yilmaz, N., Sahiner, A., 2019. New Smoothing Approximations to Piecewise Smooth Functions and Applications. *Numerical Functional Analysis and Optimization*, 40(5), 513–534.
- Yuan, L., Wan, Z., Tang, Q., 2016a. A criterion for an approximation global optimal solution based on the filled functions. *Journal of Industrial & Management Optimization*, 12(1), 375–387.
- Yuan, L.y., Wan, Z.p., Tang, Q.h., Zheng, Y., 2016b. A class of parameter-free filled functions for box-constrained system of nonlinear equations. *Acta Mathematicae Applicatae Sinica, English Series*, 32(2), 355–364.
- Zang, I., 1980. A smoothing-out technique for min—max optimization. *Mathematical Programming*, 19(1), 61–77.
- Zhang, Y., Zhang, L., Xu, Y., 2009. New filled functions for nonsmooth global optimization. *Applied Mathematical Modelling*, 33(7), 3114–3129.
- Zheng, Y.J., Xu, X.L., Ling, H.F., Chen, S.Y., 2015. A hybrid fireworks optimization

method with differential evolution operators. *Neurocomputing*, 148, 75–82.

Zhigljavsky, A., Zilinskas, A., 2007. *Stochastic global optimization*, Volume 9. Springer Science & Business Media.

Ziadi, R., Bencherif-Madani, A., Ellaia, R., 2016. Continuous global optimization through the generation of parametric curves. *Applied Mathematics and Computation*, 282, 65–83.



CURRICULUM VITAE

Name Surname : Shehab A. IBRAHEM

Place and Date of Birth : Iraq, Kirkuk, 1979

Marital Status : Married

Foreign Language : English, Turkish

Nationality : Iraqi

E-mail : mullaiq@gmail.com



Education

High School : Al-Hawija High School, 1997

B.Sc. : University of Musel, Department of Mathematics, 2001

M.Sc. : University of Tikrit, Department of Mathematics, 2008

Employment Experiences

Teaching Assistant, Department of Mathematics, Kirkuk University : 2003-2014

Publications

Sahiner, A., Ibrahim, S.A., 2019. A new global optimization technique by auxiliary function method in a directional search. *Optimization Letters*, 13(2), pp. 309-323.

Sahiner, A., Ibrahim, S.A., Yilmaz, N., Increasing the Effects of Auxiliary Function by Multiple Extrema in Global Optimization. In *Numerical Solutions of Realistic Nonlinear Phenomena*, pp. 125-143. Springer, Cham, 2020.

Sahiner, A., Yilmaz, N., Ibrahim, S.A., 2018. Smoothing Approximations to Non-smooth Functions. *Journal of Multidisciplinary Modeling and Optimization*, 1(2), pp. 69-74.

Sahiner, A., Abdulhamid, I.A.M., Ibrahim, S.A., 2019. A new filled function method with two parameters in a directional search. *Journal of Multidisciplinary Modeling and Optimization*, 2(1), pp. 34-42.

Sahiner, A., Ibrahim, S.A., A new single-strand smoothing technique and its usage in global optimization. on *Mathematics and Mathematics Education (ICMME*

2019), p. 129.

Sahiner, A., Abdulhamid, I.A., Ibrahem, S.A., New multimodal auxiliary function and directional search for global optimization. on Mathematics and Mathematics Education (ICMME 2019), p. 337.

Ibrahem, S.A., Sahiner, A., Ibrahim, A.A., 2018. Fuzzy Logic Modeling for Prediction of the Nuclear Tracks. Journal of Multidisciplinary Modeling and Optimization, 1(1), pp. 33-40.



ÖZGEÇMİŞ

Adı Soyadı : Shehab A. IBRAHEM
Doğum Yeri ve Yılı : Irak, Kirkuk, 1979
Medeni Hali : Evli
Yabancı Dili : İngilizce, Türkçe
Uyruğu : Irak
E-posta : mllaiq@gmail.com



Eğitim Durumu

Lise : Al-Hawija Lisesi, 1997
Lisans : Musel Üniversitesi, Matematik Bölümü, 2001

Mesleki Deneyim

Kirkuk Üniversitesi, Matematik Bölümü, Araştırma Görevlisi : 2003-2014

Yayımlar

- Şahiner, A., Ibrahim, S.A., 2019. A new global optimization technique by auxiliary function method in a directional search. *Optimization Letters*, 13(2), pp. 309-323.
- Şahiner, A., Ibrahim, S.A., Yilmaz, N., Increasing the Effects of Auxiliary Function by Multiple Extrema in Global Optimization. In *Numerical Solutions of Realistic Nonlinear Phenomena*, pp. 125-143. Springer, Cham, 2020.
- Şahiner, A., Yilmaz, N., Ibrahim, S.A., 2018. Smoothing Approximations to Non-smooth Functions. *Journal of Multidisciplinary Modeling and Optimization*, 1(2), pp. 69-74.
- Şahiner, A., Abdulhamid, I.A.M., Ibrahim, S.A., 2019. A new filled function method with two parameters in a directional search. *Journal of Multidisciplinary Modeling and Optimization*, 2(1), pp. 34-42.
- Şahiner, A., Ibrahim, S.A., A new single-strand smoothing technique and its usage in global optimization. on *Mathematics and Mathematics Education (ICMME 2019)*, p. 129.

Şahiner, A., Abdulhamid, I.A., Ibrahem, S.A., New multimodal auxiliary function and directional search for global optimization. on Mathematics and Mathematics Education (ICMME 2019), p. 337.

Ibrahem, S.A., Şahiner, A., Ibrahim, A.A., 2018. Fuzzy Logic Modeling for Prediction of the Nuclear Tracks. Journal of Multidisciplinary Modeling and Optimization, 1(1), pp. 33-40.

