Simulated Annealing Driven Pattern Search Algorithms for Global Optimization

Musa Nur Gabere

School of Computational and Applied Mathematics

A dissertation submitted to the Faculty of Science,
University of the Witwatersrand, in fulfillment of the requirements
for the degree of Master of Science.

August 16, 2007

Declaration

I declare that this dissertation is my own, unaided work. It is being submitted for the Degree of Master of Science to the University of the Witwatersrand, Johannesburg. It has not been submitted before for any degree or examination to any other University.

Musa Nur Gabere

(Date)

Abstract

This dissertation is concerned with the unconstrained global optimization of nonlinear problems. These problems are not easy to solve because of the multiplicity of local and global minima. In this dissertation, we first study the pattern search method for local optimization. We study the pattern search method numerically and provide a modification to it. In particular, we design a new pattern search method for local optimization. The new pattern search improves the efficiency and reliability of the original pattern search method. We then designed two simulated annealing algorithms for global optimization based on the basic features of pattern search. The new methods are therefore hybrid. The first hybrid method is the hybrid of simulated annealing and pattern search. This method is denoted by MSA. The second hybrid method is a combination of MSA and the multi-level single linkage method. This method is denoted by SAPS. The performance of MSA and SAPS are reported through extensive experiments on 50 test problems. Results indicate that the new hybrids are efficient and reliable.

Keywords: Global optimization, pattern search, simulated annealing, multi level single linkage, non-linear optimization, hybridization.

Dedication

Dedicated to my Mum

Marian Elmi Hassan

Acknowledgement

First and foremost, I give my appreciation to **ALLAH (swt)**, the Cherisher and Sustainer of the Worlds for His Infinite Blessings and Guidance in my life.

I would like to express my sincere gratitude to my supervisor, Prof. Montaz Ali, who has put his enormous effort and time in the supervision of this dissertation. He has helped me to develop the research skills necessary to produce this document. I have relied on his knowledge of the field and guidance for implementing various algorithms. He also provided me with the neccessary tools for research such as routines for the test problems and simulated annealing. His innumerable and insightful suggestions have ultimately shaped this document. Without his help, this dissertation would not have realised.

I cannot miss to mention the love, support and devotion of my family. They have given me the strength and encouragement, that made me complete my studies. Many thanks goes to my Mum, Marian, my Dad, Nur, my one and only sister, Shugri and all my brothers, namely Mahammed, Abdi, Hassan, Abdihakin and Bashir. Special thanks goes to my elder brother, Mahammed, for his constant communication and encouragement.

Helpful email correspondence with Dr. Professor Kaelo is gratefully acknowledged. I also thank my colleagues who have helped me at every step of the way. Special thanks goes to Mihaja, Naval, Gasper, Morgan and Charles.

Lastly, my special gratitude goes to the School of Computational & Applied Mathematics and the African Institute for Mathematical Sciences (AIMS) for their financial support towards my research.

Nomenclature

Acronyms

PS	Pattern search
MPS	Modified pattern search
SA	Simulated annealing
MSA	Modified simulated annealing
SAPS	Simulated annealing driven pattern search
MSL	Multi level single linkage

Superscripts used throughout this dissertation

k	Iteration counter
sa	Simulated annealing
t	Temperature counter

General symbols

Ω	Search region
N	Sample size
n	Dimension of the problem
f	Objective function
x	A vector
min/max	Minimize/Maximize
x_i	The i^{th} component of the vector x
l_i	Lower bound in the i^{th} dimension
u_i	Upper bound in the i^{th} dimension

Symbols related to pattern search

$x^{(k)}$	k^{th} iterate of x.
Δ_k	Step size parameter at iterate k
∇	First order derivative
D	The set of positive spanning directions
θ_k	Expansion factor at iteration k
ϕ_k	Contraction factor at iteration k
$\liminf$	Limit inferior
η	Step factor

Symbols related to simulated annealing

χ	Acceptance ratio
k_B	Boltzmann's constant
m_0	Number of trial points
m_1	Number of successful trial points
m_2	Number of unsuccessful trial points
δ	Cooling rate control parameter
ε_s	Stop parameter
E_i	Energy state of the system configuration i
ΔE_i	Difference in energy between new and current configurations
p	Probability
s_i	State

Symbols related to MSA and SAPS hybrid

RD	Random direction
Δ_0^{sa}	Initial step size parameter used inside SA
x^b	The best point vector
x_i^ρ	Sample point

Contents

1 Introduction 1

1.1 Problem formulation 2

1.2 Classification of global optimization methods 3

1.3 Hybridization 5

1.4 The structure of the dissertation 6

2 Pattern search for unconstrained local optimization 7

2.1 The pattern search (PS) method 7

2.2 Description of the PS method 9

2.3 The convergence properties of the PS method 14

2.4 Pros and cons of the PS method 15

2.5 The modified pattern search method 16

2.6 Summary 19

3 Simulated annealing for unconstrained global optimization 20

3.1 The physical annealing 20

3.2 The Metropolis procedure 21

3.3 The simulated annealing (SA) method 22

3.4 The simulated annealing algorithm for continuous problems 23

3.5 Cooling schedule 25

3.6 Advantages and disadvantages of the SA method 27

3.7 Summary 27

4 The hybrid global optimization algorithms based on PS 28

4.1 Generation mechanism 28

4.2 Proposed hybrid methods 31

4.2.1 Modified simulated annealing (MSA) 31

4.2.2 Simulated annealing driven pattern search (SAPS) 34

4.3 The single iteration-based MSL algortihm. 35

4.4 Summary 38

5 Numerical results 39

5.1 Numerical results for PS and MPS 40

5.1.1 Parameter values 40

5.1.2 Numerical comparison 40

5.2 Numerical results for MSA and SAPS 43

5.2.1 Parameter values 43

5.2.2 Numerical studies of the MSA method 44

5.2.3 Numerical studies of the SAPS method 51

5.3 Overall Performance 66

5.4 Effect of temperature on step size parameter (Δ_t^{sa}) 67

5.5 A study of the critical distance Δ_t^c in the single iteration-based MSL 70

5.6 Summary 71

6 Conclusion and future research 72

Appendix **73**

A The multi level single linkage algorithm **73**

B A collection of benchmark global optimization test problems **75**

Bibliography **90**

List of Figures

1.1 The structure of the dissertation. 6

2.1 Trial points of PS at N, E, W and S positions. 8

2.2 Figures (a)-(f) shows how the POLL steps works in the PS method. 13

2.3 Convergence of the pattern search method. 14

2.4 The first trial point by MPS. 18

2.5 The generation of the second trial point by MPS when the first trial point is unsuccessful. 18

2.6 The generation of the second trial point of MPS when the first trial point is successful. . . 19

4.1 Flowchart for the MSA algorithm. 32

4.2 Flowchart for the SAPS algorithm. 36

5.1 Effect of T_t on Δ_t^{sa} for HSK problem. 68

5.2 Effect of T_t on Δ_t^{sa} for GP problem. 68

5.3 Effect of T_t on Δ_t^{sa} for S5 problem. 69

5.4 Effect of T_t on Δ_t^{sa} for RB problem. 69

5.5 Profile of Δ_t^{c} and Δ_t^{sa} for BP problem. 70

5.6 Profile of Δ_t^{c} and Δ_t^{sa} for KL problem. 71

List of Tables

5.1 Comparison of PS, PS-I and MPS using 32 problems. 42

5.2 Results of MSA for different values of ζ, GM-I. 45

5.3 Comparison of different α values in MSA using 41 problems, GM-I. 46

5.4 Comparison of MSA$_{\alpha=0.15}$ and MSA-I using 43 problems, GM-I. 48

5.5 Results of MSA$_{\alpha=0.15}$ using 43 problems, GM-II. 50

5.6 Results of SAPS for different values of δ, GM-I. 52

5.7 Results of SAPS for different values of β, GM-I. 54

5.8 Results of SAPS for different values of β, GM-I. 56

5.9 Results of SAPS for $\beta = 20$ using 43 problems, GM-II. 58

5.10 Results of SAPS for different sample size N, GM-I. 59

5.11 Results of SAPS using $N = 5n$ & $\gamma = 1$ using 43 problems, GM-I. 61

5.12 Results of SAPS using $N = 5n$ & $\gamma = 0.5$ using 43 problems, GM-I. 63

5.13 Results of SAPS using $N = 5n$ & $\gamma = 0.25$ using 43 problems, GM-I. 65

5.14 Comparison of the algorithms using total results. 66

5.15 Rank order of algorithms. 67

B.1 Epistatic Michalewicz's global optimizers. 78

B.2 Data for Hartman 3 problem. 79

B.3 Data for Hartman 6 problem. 80

B.4 Data for Hartman 6 problem. 80

B.5 Data for Kowalik problem. 81

B.6 Data for Meyer & Roth problem. 82

B.7 Data for modified Langerman problem. 82

B.8 Global optimizers for modified Langerman problem. 83

B.9 Data for Multi-Gaussian problem. 83

B.10 Global minima for Neumaier 3 problem. 84

B.11 Data for Price's transistor modelling problem. 85

B.12 Global optimizers for Schubert problem. 87

B.13 Data for Shekel problem family. 88

B.14 Data for Shekel's foxholes problem. 89

B.15 Global optimizers for Shekel's foxholes problem. 89

B.16 Global optimizers for Storn's Tchebychev problem. 90

Chapter 1

Introduction

Optimization is an important research area. It is the study of problems in which one seeks to minimize or maximize a real function by systematically choosing the values of real or integer variables within an allowed set. Optimization is mainly divided into two branches namely continuous and discrete optimization. A continuous optimization is where the variables used in the objective function assume real values. On the other hand, a discrete optimization is where the variables used in the objective function are restricted to assume only discrete values, such as integers. Continuous optimization problems can be classified according to the mathematical structure of the objective function and constraints. For example, a problem that has linear objective function and linear constraints is called a linear optimization problem. On the other hand, a problem that has nonlinear (linear) objective function with nonlinear or linear (nonlinear) constraints is called a nonlinear optimization problem. In other words, a nonlinear optimization problem is where the objective function or the constraints or both contain nonlinear terms. A nonlinear optimization problem can either be unconstrained or constrained depending on the presence of constraints or limitations on the variables.

A nonlinear optimization problem can have more than one optimal solution. The goal of a local optimization method is to obtain any one of the optimal solutions. On the other hand, the goal of a global optimization method is to obtain the best optimal solution from a number of solutions. The best (global) optimal solution is not only hard to determine but also hard to verify. Despite its inherent difficulties, global optimization is vital to many practical applications. Some of these applications include, but not restricted to engineering design, financial risk management, computational chemistry, molecular biology and economics [8, 28]. Global optimization problems can be classified according to the properties of the objective function and constraints. A problem that has no constraints or constrained by simple lower and/or upper bounds is called unconstrained global optimization problem. A problem that has linear (nonlinear) constraints and nonlinear objective function is called a linearly (nonlinearly) constrained global optimization problem. These problems arise in real-life applications. In many applications, global optimization problems are of black-box type. A black-box scenario occurs whenever the objective function and/or constraints are not given in closed form, i.e., if the objective function values and/or constraints are evaluated via complex computations, simulations or experiments.

Our research is concerned with the design of unconstrained global optimization algorithms for solving both noisy and black-box type global optimization problems. An ideal global optimization algorithm should:

- work for a wide range of problems, be it easy, moderately difficult or difficult problems [47],
- not depend on the properties (e.g., continuity) of the objective function to be optimized,
- be easy to implement, and
- require very little computational effort.

It is not so easy to design an algorithm that satisfies all the above criteria. In any case, progress have been made and a number of global optimization algorithms have been suggested in the literature. We will review these algorithms later in the chapter. In the next section, we will present the global optimization problem mathematically.

1.1 Problem formulation

We consider the problem of finding the global optimum of box-constrained global optimization problems. The mathematical formulation of the global optimization problem is defined as follows

$$\text{optimize} \quad f(x) \quad \text{subject to} \quad x \in \Omega, \tag{1.1}$$

where $x = (x_1, \cdots, x_n)$ is an $n-$dimensional vector of unknowns, $\Omega \subseteq \mathbb{R}^n$ is the search region, and f is a nonlinear continuous real-valued objective function, i.e., $f : \Omega \to \mathbb{R}$. The domain of the search space, Ω, is defined by specifying an upper limit u_i and a lower limit l_i of each i^{th} component of x, i.e.,

$$l_i \leq x_i \leq u_i, \quad l_i, u_i \in \mathbb{R}, \quad i = 1, 2, \cdots, n. \tag{1.2}$$

Without loss of generality, we consider only the global minimization problem since the global maximum can be found in the same way by reversing the sign of f, i.e.,

$$\max_{x\in\Omega} f(x) = -\min_{x\in\Omega}(-f(x)\,). \tag{1.3}$$

A point $x^* \in \Omega$ is called a global minimizer of f with the corresponding global minimum value $f^* = f(x^*)$ if

$$f^* \leq f(x), \quad \text{for all} \quad x \in \Omega. \tag{1.4}$$

On the other hand, a point $x^{loc} \in \Omega$ is referred to as a local minimizer of f over Ω if there is an $\epsilon > 0$ such that

$$f(x^{loc}) \leq f(x), \quad \text{for all} \quad x \in N_\epsilon(x^{loc}) \cap \Omega, \tag{1.5}$$

where $N_\epsilon(x^{loc}) \overset{def}{=} \{x \in \mathbb{R}^n \,:\, \|x - x^{loc}\| < \epsilon\}$.

1.2 Classification of global optimization methods

Global optimization methods can be classified as deterministic and stochastic methods [22, 26]. Deterministic methods usually use gradient information and other properties such as known Lipschitz constant of f. The main disadvantage of deterministic methods is that they cannot be implemented in noisy and black-box type functions. In addition, these methods are very slow as they often perform an exhaustive search. As opposed to deterministic methods, stochastic methods are very easy to implement and in most instances, they do not require any functional properties. Hence these methods are widely applicable. Unlike deterministic methods, which guarantee convergence to the global minimum, stochastic methods assures convergence in a probabilistic sense. In addition, the computational costs of stochastic methods are in general less than those of deterministic methods [41]. For this reason, in this dissertation, we concentrate on stochastic methods, in particular simulated annealing and multi level single linkage with some hybridization. We will briefly present some deterministic and stochastic methods.

One of the best known deterministic method is the interval arithmetic method [27] for global optimization. It is based on the branch-and-bound method [38]. The branch-and-bound method is a technique where the feasible region is relaxed and subsequently split into parts (branching) over which the lower (and often also the upper) bounds of the objective function value can be determined (bounding). Another important deterministic method is the multi-dimensional bisection method [52] which is a generalization of the bisection method [16] to higher dimensions. It begins by generating a sequence of intervals whose infinite intersection is the set of points desired. However, unlike interval arithmetic method, this method never attracted the global optimization researchers and practitioners. In addition to the above deterministic methods, Breiman and Cutler [18] designed a deterministic algorithm for global optimization. This algorithm assumes a bound on the second derivatives of the function and uses this to construct an upper envelope. Successive function evaluations lower this envelope until the value of the global minimum is found. Other deterministic methods include αBB [1], Lipschitz method, methods based on convex envelopes of the objective function over special domain like boxes. There are softwares developed to solve deterministic global optimization. Currently the branch-and-reduce optimization navigator (BARON) [43] is the best software in the field of deterministic global optimization.

Stochastic methods are either single sample (point) based or multiple sample (population) based methods. Within the single sample based methods, tabu search [19], adaptive random search [35] and simulated annealing [2, 21] are well known. Among the population based methods, density clustering [41], multi level single linkage [42] and topographical multilevel single linkage [13] often referred to as two-phase methods. Two-phase methods use both random sampling (global phase) and local search (local phase). In the global phase, the function is evaluated in a number of randomly sampled points while in the local phase, the sample points are scrutinised by a clustering technique in order to identify potential points to start a local search. A more detailed survey of the two-phase methods for stochastic global optimization can be found in [44].

Other population based methods are genetic algorithm [23], controlled random search [6, 10, 40] and differential evolution [12, 45]. These methods start with an initial population set of points, drawn uniformly in the search space Ω and subsequently manipulating this sample in order to obtain a better population set. The better population set is obtained by replacing all or some members of the current set with new trial points. The mechanism used in creating trial points depends on the considered algorithm [11]. For example, in genetic algorithm, trial points are generated by selecting successively a subset of the population and then applying mutation and crossover operations on this set. In controlled random search, a trial point is generated by forming a simplex using $(n+1)$ distinct points, chosen at random with replacement from the population set, and reflecting one of the points in the centroid of the remaining n points of the simplex, as in the Nelder and Mead algorithm [37]. In differential evolution, trial points are generated using mutation and crossover operations. In addition to the above stochastic methods, there exist hybrid methods. The purpose of hybrid methods is to use the complementary strengths of several methods within a single method. Next, we present the main features of the hybrid method for global optimization.

1.3 Hybridization

Hybridization is basically the combination of principles (elements) from different methods so as to give rise to a new method that displays desirable properties of the original methods but not their weaknesses.

There are different ways to hybridize methods which combine two methods [39, 46]. One approach is to run one algorithm until it stops before the next one is started. This is known as sequential hybridization. Another approach is to run the algorithms in parallel in a pre-defined manner, e.g., the next method starts before the previous one ends. This is known as parallel hybridization. However, in most cases, hybridization is achieved by combining algorithmic elements of the original methods to end up with a single algorithmic architecture. For instance, a popular approach is to combine global features of a global method with local features of a local method. Local methods are computationally efficient because they make use of local information around the current point to move to a promising region. This expedites convergence whenever a point is within the region of attraction of a minimum. On the other hand, global methods are more reliable in locating the global minima because they explore the whole search region and have mechanisms to escape being trapped in local minima. Consequently they are computationally expensive.

It is expected that the combination of elements from local methods with those of global methods would result in methods that are more efficient, more accurate and more reliable in finding the global minimum. Efficiency refers to the amount of efforts (be it CPU time or number of function evaluations) required to obtain a solution. Accuracy means how close is the final solution obtained by a global optimization algorithm to the known global minimum of a problem. Reliability is how successful is the method in finding the global minimum. Hence such hybrid methods would result into being more reliable in locating the global minimum than a local method and also more accurate and more efficient than a global method. Examples of hybrid methods include but not restricted to simulated annealing combined with direct search hybrid [9, 31] and tabu search combined with Nelder-Mead simplex hybrid [20].

We aim at designing hybrid methods that will possess only strengths of the original methods. In this dissertation, we will combine global methods (simulated annealing, multi level single linkage) and a local method (pattern search).

1.4 The structure of the dissertation

The dissertation is divided into six chapters as shown in Figure 1.1. In Chapter 2, we address the strengths and weaknesses of the pattern search method. We first review the pattern search method in relation to its description, convergence properties and its limitations in solving global optimization problems. Then we propose a modified pattern search method.

In Chapter 3, we present an overview of the simulated annealing method in regard to its origin. We also present the simulated annealing for continuous problems and the cooling schedule.

In Chapter 4, we propose two new hybrid methods based on the pattern search method, the simulated annealing method and the multi level single linkage method.

In Chapter 5, we report the performance of the proposed hybrid methods using extensive numerical experiments on some well known test problems.

In Chapter 6, we summarize the work in this dissertation and propose further avenues to extend and enhance this research. Finally, we give a description of the multi level single linkage algorithm and a collection of 50 benchmark global optimization test problems in Appendixes A and B, respectively.

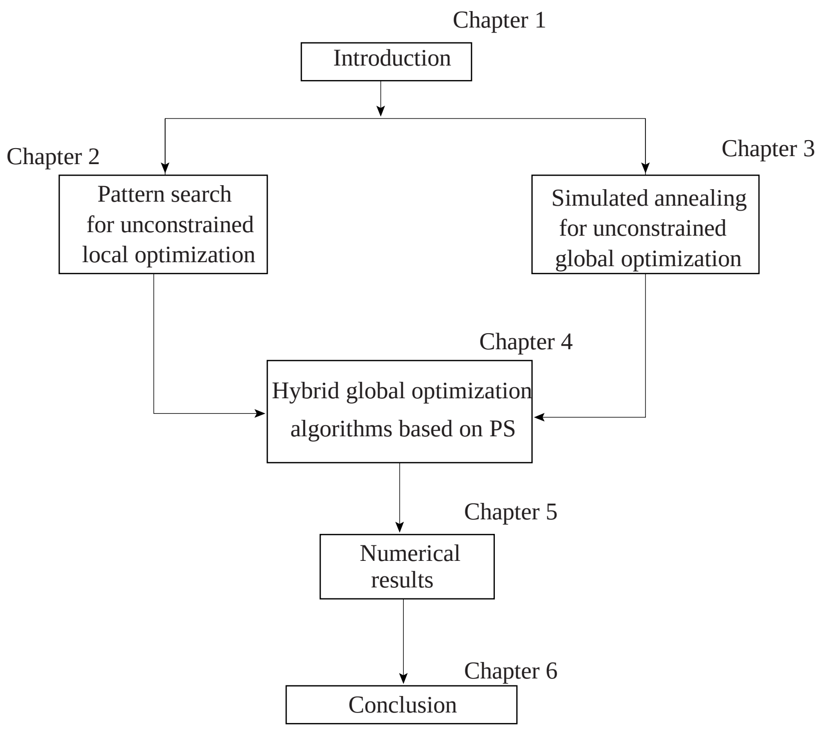
Chapter 1
Introduction
Chapter 2
Pattern search for unconstrained local optimization
Chapter 3
Simulated annealing for unconstrained global optimization
Chapter 4
Hybrid global optimization algorithms based on PS
Chapter 5
Numerical results
Chapter 6
Conclusion

Figure 1.1: The structure of the dissertation.

Chapter 2

Pattern search for unconstrained local optimization

The pattern search method [30, 48] is a recent direct search method for local optimization. In this chapter, we describe the pattern search method for unconstrained local optimization and propose a modification to it.

2.1 The pattern search (PS) method

In its simplest form, the PS method is a variation of the coordinate search method [30]. However, the mathematical formalization presented by Torczon [48] shows that the PS method is a general class of the direct search methods. For instance, the Hooke and Jeeves method [25], the basic coordinate search method [30] and the multi-directional search method [49] also form part of the PS method. As such, in some literature [15], the PS method is referred to as the generalized pattern search (GPS) method. In this dissertation, we only deal with a simple but effective variant of the PS method. Before we describe the PS method, we give two definitions [4] that are essential for understanding the search directions of this method. We also present an example of the search directions used by the PS method in a typical two dimensional problem.

Definition 2.1
A positive combination of the set of vectors $D = \{d_i\}_{i=1}^{r}$ is a linear combination $\sum\limits_{i=1}^{r} \lambda_i d_i$, where $\lambda_i \geq 0,\ i = 1, 2, \cdots, r$.

Definition 2.2
A finite set of vectors $D = \{d_i\}_{i=1}^{r}$, $n + 1 \leq r \leq 2n$, forms a positive spanning set for $\mathbb{R}^n$ if any $\nu \in \mathbb{R}^n$ can be expressed as a positive combination of vectors in D. The set of vectors D is said to positively span $\mathbb{R}^n$. The set D is said to be a positive basis for $\mathbb{R}^n$ if no proper subset of D spans $\mathbb{R}^n$.

Having presented the above definitions, we now describe the directions used by the PS method. The simplest search directions used by the PS method is made up of $r = 2n$ vectors and given by the set

$$D = \{e_1, \cdots, e_n, -e_1, \cdots, -e_n\}, \tag{2.1}$$

where e_i is the i^{th} unit coordinate vector in $\mathbb{R}^n$. The set D in equation (2.1) is an example of a set with a maximal positive spanning directions. In the following example, we present possible trial points generated by the PS method in a typical iteration process, say at the k^{th} iteration in a two dimensional problem.

Example 2.1
In $\mathbb{R}^2$, the set of positive spanning directions, consists of four column vectors of

$$D = \left\{ \begin{array}{rrrr} 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{array} \right\}. \tag{2.2}$$

If the current iterate is $x^{(k)} = (0, 0)$, the PS method may generate up to four trial points (see later the POLL step in section 2.2) located at $E(1, 0)$, $N(0, 1)$, $W(-1, 0)$ and $S(0, -1)$ using four positive spanning directions as shown in Figure 2.1.

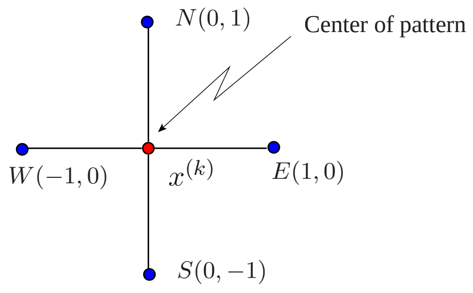

Figure 2.1: Trial points of PS at N, E, W and S positions.

2.2 Description of the PS method

In this section, we present a full description of the PS method. The PS method generates a sequence of iterates $\{x^{(1)}, x^{(2)}, \cdots x^{(k)}, \cdots\}$ with non-increasing objective function values. In each iteration k, there are two important steps of the PS method namely, the SEARCH step and the POLL step. Note that we use the value $r = 2n$ in the description of the PS method.

In the SEARCH step, the objective function is evaluated at a finite number of points (say a maximum of $\mathcal{V}$ points) on a mesh (a discrete subset of $\mathbb{R}^n$) so as to improve the current iterate. The mesh at the current iterate, $x^{(k)}$, is given by

$$M_k = \{m \in \mathbb{R}^n \,|\, m = x^{(k)} + \Delta_k Dq \,:\, q \in \mathbb{Z}^r_+\}, \tag{2.3}$$

where m is a mesh trial point, $\Delta_k > 0$ is a mesh size parameter (also known as the step size control parameter) which depends on the iteration k, and $\mathbb{Z}_+$ is the set of nonnegative integers. There are no specific rules on how to generate trial points of the SEARCH step in the current mesh. Users may generate these points by some heuristic rules. The aim of the SEARCH step is to find a feasible trial point (on a mesh M_k) that yields a lower objective function value than the function value at $x^{(k)}$. A SEARCH step is therefore successful if there exists a feasible trial point $m \in M_k$ (where m is one of the $\mathcal{V}$ points) such that $f(m) < f(x^{(k)})$. In such a case, m is treated as the new iterate and the step size Δ_k is increased so as to choose the next trial points on a magnified mesh than the previous mesh. If the SEARCH step is unsuccessful in improving the current iterate $x^{(k)}$, a second step, called the POLL step, is executed around $x^{(k)}$ with the aim of decreasing the objective function value. This step must be done before terminating the iteration.

The POLL step generates trial points at the poll set around the current iterate, $x^{(k)}$, as shown in Figure 2.1, for the case of a two dimensional problem, where $\Delta_k = 1$. The poll set is composed of trial points that are positioned a step Δ_k away from the current iterate $x^{(k)}$, along the direction designated by the columns of D. This poll set is denoted by P_k and is defined by

$$P_k = \{\, p_i \in \mathbb{R}^n \,|\, p_i = x^{(k)} + \Delta_k d_i \,:\, d_i \in D, \quad i := 1, \cdots, r\,\}, \tag{2.4}$$

where p_i is a trial point in the POLL step. The order in which the points in P_k are evaluated can also differ and has no effect on convergence. We now present the step by step description of the PS algorithm [3] using both the SEARCH and the POLL step.

Algorithm 2.1: The PS algorithm (based on the SEARCH and the POLL steps).

1. **Initialization:** Choose an initial point $x^{(0)} \in \Omega$ and an initial mesh size $\Delta_0 > 0$. Set the iteration counter $k = 0$.

2. **SEARCH step:**. Evaluate f at a finite number of points in the mesh M_k as defined by (2.3). **If** $f(m) < f(x^{(k)})$ for some $m \in M_k$ **then** set $x^{(k+1)} = m$ and go to step 4 (the SEARCH step is deemed successful). **If** the SEARCH step is unsuccessful, i.e., $f(x^{(k)}) \leq f(m)$, for all $\mathcal{V}$ points in M_k **then** go to step 3.

3. **POLL step:** This step is executed only if the SEARCH step is unsuccessful.

   - **If** $f(p_i) < f(x^{(k)})$ for some p_i in the poll set P_k defined by (2.4), **then** set $x^{(k+1)} = p_i$ and go to step 4 in order to increase the mesh size Δ_k, (POLL step is declared successful).
   - **Otherwise** if $f(x^{(k)}) \leq f(p_i)$ for all p_i in the poll set P_k defined by (2.4) , set $x^{(k+1)} = x^{(k)}$ and go to step 5 in order to decrease the mesh size Δ_k, (POLL step is declared unsuccessful).

4. **Mesh expansion:** Let $\Delta_{k+1} = \theta_k \Delta_k$, (with $\theta_k > 1$). Increase $k := k + 1$ and go to step 2 for a new iteration.

5. **Mesh reduction:** Let $\Delta_{k+1} = \phi_k \Delta_k$, (with $0 < \phi_k < 1$). Increase $k := k + 1$ and go to step 2 for a new iteration.

In summary, Algorithm 2.1 performs the SEARCH and the POLL step. In the SEARCH step, the objective function f is evaluated at a finite number of trial points $m \in M_k$ with the goal of improving the current iterate $x^{(k)}$. If an improvement is accomplished, then the trial point m becomes the current iterate and the mesh size is increased, i.e., $\Delta_{k+1} = \theta_k \Delta_k$ and the SEARCH step continues. Otherwise, if the SEARCH step is unsuccessful in improving the current iterate $x^{(k)}$, for all $\mathcal{V}$, a second step called the POLL step is invoked. If the POLL step is successful, i.e., $f(p_i) < f(x^{(k)})$ for some $p_i \in P_k$ then p_i becomes the new iterate, the mesh size is increased and the SEARCH step process is invoked. If $f(p_i) \geq f(x^{(k)})$ for all $p_i \in P_k$ then the current iterate $x^{(k)}$ is retained, the mesh size is decreased and the SEARCH step is performed.

In the literature of the PS method, no specific information is given on how to implement the SEARCH step. Indeed, the results of the PS method using only the POLL step are reported in the literature [3, 30]. It is also reported in [15] that the SEARCH step is a liability to convergence. Therefore, in this dissertation, we will only implement the POLL step in the PS method. Before we present the PS algorithm based on the POLL step, we would like to elaborate more on the POLL step. We discuss how the current iterate and the step size are updated in the POLL step.

The POLL step begins by determining a trial point p_i in the poll set P_k defined earlier, i.e.,

$$\{\, p_i \in \mathbb{R}^n \,|\, p_i = x^{(k)} + \Delta_k d_i \,:\, d_i \in D, \text{ where } i := 1, \cdots, r \,\},$$

where $x^{(k)}$ is the current iterate. The trial point p_i is examined so as to determine if it is a better solution than the current iterate $x^{(k)}$. (Here the trial point p_i could be one of the positions, say for $i = 1$, it is $E(1,0)$ of Figure 2.1). If the POLL step produces a successful point $p_i \in P_k$ such that $f(p_i) < f(x^{(k)})$, then the POLL step stops examining the remaining trial points in the current POLL set P_k. This means that if the POLL step is declared successful then a new POLL step starts at this new current iterate $x^{(k+1)} = p_i$. Otherwise, the current iterate is retained, i.e., $x^{(k+1)} = x^{(k)}$, when $f(p_i) \geq f(x^{(k)})$ for all the trial points $p_i \in P_k$, i.e., the POLL step is declared unsuccessful. Thus, the next iterate for the next POLL step is updated as follows:

$$x^{(k+1)} = \begin{cases} p_i & \text{if } f(p_i) < f(x^{(k)}), \text{ for some } p_i \in P_k, \\ x^{(k)} & \text{otherwise}. \end{cases} \tag{2.5}$$

In the case of a successful POLL step, the step size parameter Δ_{k+1} for the next iteration is increased to $\Delta_{k+1} = \theta_k \Delta_k$, where $\theta_k > 1$, in a similar fashion as in mesh expansion of Algorithm 2.1. This enhances exploration of the PS method. However, when the POLL step is unsuccessful, then step size parameter is decreased to $\Delta_{k+1} = \phi_k \Delta_k$, for $0 < \phi_k < 1$, in a similar way as in the mesh reduction of Algorithm 2.1. This in turn enhances exploitation. In summary the step size parameter is updated [48] as follows:

$$\Delta_{k+1} = \begin{cases} \theta_k \Delta_k & \text{if } f(p_i) < f(x^{(k)}), \text{ for some } p_i \in P_k, \\ \phi_k \Delta_k & \text{otherwise}. \end{cases} \tag{2.6}$$

This POLL step is reiterated until the step size parameter Δ_k gets sufficiently small, thus ensuring convergence to a local minimum. Note that as $x^{(k)}$ approaches the optimum, the algorithm reduces the length of steps taken. This turns out to be central to the convergence proof which will be discussed in section 2.3.

In most implementation of the PS method, the initial step size parameter $\Delta_0 = 1$ is used and the updating of the step size parameter is carried out by

$$\Delta_{k+1} = \begin{cases} 2\Delta_k & \text{if } f(p_i) < f(x^{(k)}), \text{ for some } p_i \in P_k,\ \theta_k = 2, \\ \frac{1}{2}\Delta_k & \text{otherwise}, \ \phi_k = \frac{1}{2}. \end{cases} \tag{2.7}$$

The basic PS method based only on the POLL step described above is presented in Algorithm 2.2 below. Note that from now on, the PS algorithm based on the POLL step will be referred to as the PS algorithm.

Algorithm 2.2: The PS algorithm.

1. **Initialization:**

   Choose an initial feasible solution $x^{(0)} \in \Omega$. Select an initial step size $\Delta_0 > 0$. Choose the positive spanning set D defined by equation (2.1). Set the counter numbers $k = 0$ and $i = 1$. Choose the stopping tolerance $\Delta_{tol} > 0$.

2. **POLL step:**

   2(a) Evaluate the objective function f at the trial point $p_i = (x^{(k)} + \Delta_k d_i) \in P_k$, $d_i \in D$.

   2(b) **If** $f(p_i) < f(x^{(k)})$ **then** set $x^{(k+1)} = p_i$ and go to step 3.
   Otherwise, increase $i := i + 1$ and go to step 2(c).

   2(c) **If** $i \leq r$ **then** go to step 2(a).
   Otherwise, set $x^{(k+1)} = x^{(k)}$ and go to step 4.

3. **Mesh expansion:** Increase the step size parameter $\Delta_{k+1} = \theta_k \Delta_k$. Set $i = 1$ and go to step 5.

4. **Mesh reduction:** Decrease the step size parameter $\Delta_{k+1} = \phi_k \Delta_k$. Set $i = 1$ and go to step 5.

5. **Stopping condition:** **If** $\Delta_{k+1} < \Delta_{tol}$ **then** stop. **Otherwise**, increase $k := k + 1$ and go to step 2.

Having described the PS algorithm, we now illustrate a step by step process of this algorithm, using the following example. In this example. we use $\theta_k = 2$ and $\phi_k = \frac{1}{2}$.

Example 2.2

This example illustrates how the previous Algorithm 2.2 works in $\mathbb{R}^2$. In Figure 2.2, $x^{(k)}$ is the current iterate at the k^{th} iteration and is represented by the dotted circle $\odot$. The solid circle $\bullet$ indicates the position of the trial point $p_i \in P_k$ to be examined, where $i = 1, \cdots, r$. The small open circle $\circ$ and the circled asterisk $\circledast$ represent unsuccessful and successful trial points respectively of the POLL step. The POLL step begins by evaluating the function value of the trial point $p_i \in P_k$, point by point, where $i = 1, \cdots, 4$, as shown in Figure 2.2. In Figure 2.2(a), the PS method computes the trial point p_1 by a step of size Δ_k. It computes the function value at p_1. If $f(p_1) > f(x^{(k)})$ then it examines the next trial point p_2 as shown in Figure 2.2(b). If it is not successful at p_2, i.e., $f(p_2) > f(x^{(k)})$ then it computes p_3 as shown in Figure 2.2(c). If p_3 is still unsuccessful then the process is repeated until all the trial points in P_k are examined, i.e., until p_4 is computed as shown in Figure 2.2(d). If all the points in the POLL set P_k (i.e., p_1, p_2, p_3 and p_4) are not successful then the step size is reduced by half as shown in Figure 2.2(e), i.e., the next POLL step begins at $x^{(k+1)} = x^{(k)}$ with $\Delta_{k+1} = \frac{1}{2}\Delta_k$. On the other hand, suppose that the trial point p_2 is successful, i.e., $f(p_2) < f(x^{(k)})$ as shown in Figure 2.2(f), then the whole POLL step process starts anew at $x^{(k+1)} = p_2$ with enlarged step size, i.e., $\Delta_{k+1} = 2\Delta_k$ as shown in Figure 2.2(h). A similar cycle as shown in (a), (b), (c) and (d) of Figure 2.2 will be repeated (if necessary) for the new POLL at $x^{(k+1)}$.

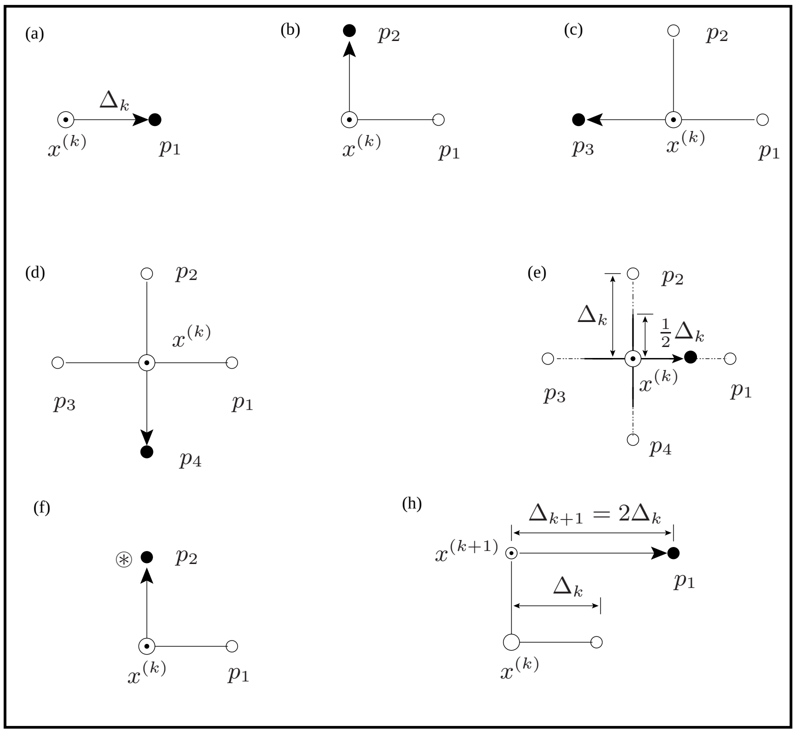

Figure 2.2: Figures (a)-(f) shows how the POLL steps works in the PS method.

2.3 The convergence properties of the PS method

In this section, we will discuss the convergence of the PS method. Before proceeding to a formal statement of the convergence of the PS method, let us first set the stage. We begin by discussing the properties of the PS method which guarantees its convergence. These properties include

1. At any iterate $x^{(k)}$, the positive spanning set D contains at least one descent direction. This means that there exist some $d_i \in D$ for which

$$-\nabla f(x^{(k)})^T d_i \geq \frac{1}{\sqrt{n}} \|\nabla f(x^{(k)})\| \, \|d_i\|, \tag{2.8}$$

where n is the dimension of the problem. This can be illustrated in Figure 2.3 for the case $n = 2$. In Figure 2.3, the search direction pointing towards **S** is a descent direction because it is within 45° of the steepest descent direction $-\nabla f(x)$. It can also be seen that the direction pointing towards **E** is also descent. Furthermore, the PS method can be viewed as a gradient related method. It is shown

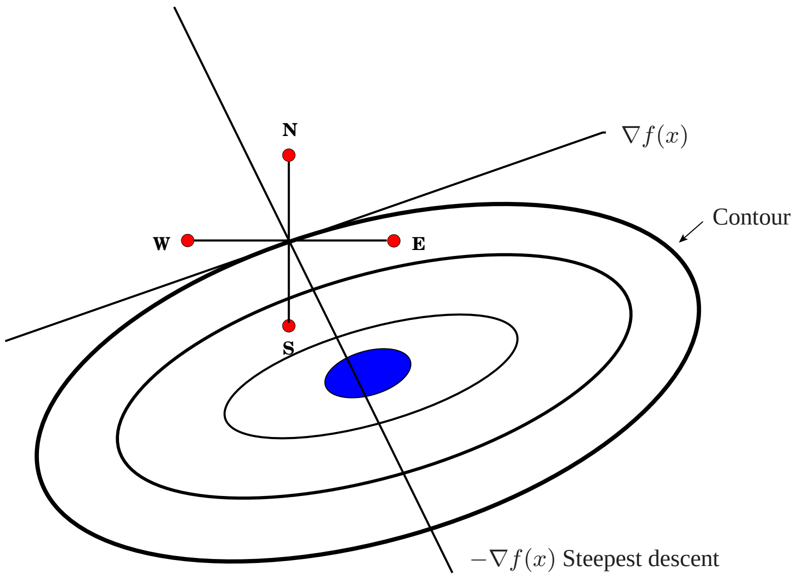

Figure 2.3: Convergence of the pattern search method.

in [30] that if we suppose that f is continuously differentiable, and for simplicity, ∇f is Lipschitz with a constant M, then from equation (2.8), since $\|d_i\| = 1$, we have

$$\|\nabla f(x^{(k)})\| \leq \sqrt{n} M \Delta_k. \tag{2.9}$$

2. As $k \to +\infty$, the total number of successful iterations must be finite. This means that the number of unsuccessful iterations (in POLL) is infinite. Therefore, $\Delta_k \to 0$ as $k \to +\infty$.

Clearly, the convergence analysis of the PS method is based on the standard assumption that all trial points produced by the algorithm lie in a compact set. That is, the level set $\{ x \in \Omega : f(x) < f(x^{(0)}) \}$ is bounded. This boundedness of the level set will ensure that the step size parameter satisfies

$$\lim_{k \to +\infty} \Delta_k = 0. \tag{2.10}$$

It has been established in [48] that the PS method possesses the convergence property i.e.

$$\liminf_{k \to +\infty} \|\nabla f(x^{(k)})\| = 0, \tag{2.11}$$

which follows directly from equation (2.9) and equation (2.10).

After discussing the convergence of the PS method, we now focus our attention in elaborating some of the pros and cons of the PS method with regard to solving global optimization problems. We briefly discuss the pros and cons of the PS method and thereafter propose a modification that will eliminate some of its limitations.

2.4 Pros and cons of the PS method

Associated with the PS method are the following advantages :

- It is a direct search method and does not depend on any properties (continuity or differentiability) of the objective function being optimized.
- It initially makes a rapid progress towards a local solution, i.e., excellent convergence characteristics.
- It is easily programmable and easy to implement.

We studied the numerical efficiency and robustness of the PS method. We applied the PS method on 50 simple bounded global optimization test problems (see Appendix B). Our numerical experiments suggested the following shortcomings of the PS method.

1. **The initial step size parameter** Δ_0. Another problem experienced by the PS method is its traditional use of initial step size $\Delta_0 = 1$. This makes the search very slow in the case of problems with

large search regions and hence takes longer time to converge. Also it is not appropriate for problems with small search regions.

2. **Badly scaled function**. The PS method is very slow to converge when the level sets of the function are extremely elongated. This is because of its use of coordinate directions.

3. **Dimensionality problem**. Lastly, the PS method suffers from curse of dimensionality. As the number of dimension increases, the PS method breaks down.

Having discussed the limitations, we aim at remedying some of these limitations of the PS method.

2.5 The modified pattern search method

In this section we will discuss how to eliminate some of the shortcomings of the PS method. Among the shortcomings of the PS method, the initial step size Δ_0 and searching along coordinate directions were very sensitive. Hence we suggest the following modifications.

To deal with the problem of initial step size parameter ($\Delta_0 = 1$), we decided to use an initial step size parameter which depends on the size of the search region Ω. We propose

$$\Delta_0 = \max\{\, u_i - l_i \,|\, i = 1, \cdots, n \,\}/2, \tag{2.12}$$

where u_i and l_i are upper and lower bounds respectively of the search region Ω for each dimension. The initial stepsize $\Delta_0 = 1$ used in PS for unconstrained local optimization where no bounds exists for the variable of the problem. The property $\Delta_k \to 0$ as $k \to \infty$ is an important ingredient for the convergence of PS. In this study, we used Δ_0 in equation (2.12) to solve bound constrained global optimization problems. The step size Δ_0 is used in such a way that it takes into account on the size of the search space. There are instances whereby the component $p_i^j > u_j$ (or $p_i^j < u_j$) of the trial point $p_i = (p_i^1, \cdots, p_i^n)$ falls outside Ω. In these cases, we re-generate a trial point p_i with the component

$$p_i^j = x^{(k)j} + \omega(u_j - x^{(k)j}),$$

or

$$p_i^j = l_j + \omega(x^{(k)j} - l_j),$$

where ω is a random number $(0, 1)$ and $x^{(k)j}$ is the corresponding component of the current iterate, $x^{(k)}$.

To deal with the problem of searching along coordinate direction, we decided to use a perturbation of the coordinate direction. This modification is described as follows. Starting at the current iterate $x^{(k)}$ at the k^{th} iteration, the POLL step computes the next iterate $p_i \in P_k$ by a step size Δ_k in the same way as in the PS method. However, it does not compute the function value at p_i as in the PS method. Instead it uses this point as a stepping stone to compute the trial point $\tilde{p}_i$ with a step of size $r = \eta\Delta_k$ and this trial point $\tilde{p}_i$ is given by

$$\tilde{p}_i = p_i + r \times U, \tag{2.13}$$

where $U = (U_1, \ldots, U_n)^T$ is a directional cosines with random components

$$U_j = R_j/(R_1^2 + \cdots + R_n^2)^{1/2}, \; j = 1, \cdots, n, \tag{2.14}$$

R_j is a uniform random number in the interval $[-1, 1]$. The PS method equipped with (2.12) and (2.13) is denoted by MPS. The above modifications in equation (2.12)-(2.14) preserves the convergence properties of the PS method. The POLL step of this modification can be explained using the following example.

Example 2.3

The POLL step of this modification is explained as follows using Figures 2.4, 2.5 and 2.6. In these Figures, the definitions of the dotted circle $\odot$, solid circle $\bullet$ and $\circledast$ are the same as in example 2.2 except for the small open circle $\circ$ which represents a stepping point. Given the current iterate $x^{(k)}$ in Figure 2.4, the point p_1 is first computed as in POLL step of Algorithm 2.2. Unlike the PS method, the MPS does not calculate the function value at p_1. In its place, a new neighboring point $\tilde{p}_1$ using equation (2.13) is calculated uniformly on the surface of a hypersphere with radius r. The POLL step then compares the function values of $f(x^{(k)})$ and $f(\tilde{p}_1)$. If it is successful, i.e., $f(\tilde{p}_1) < f(x^{(k)})$ then the new POLL step begins at the new iterate $x^{(k+1)} = \tilde{p}_1$ with $\Delta_{k+1} = 2\Delta_k$ as in the POLL step of Algorithm 2.2. If it is unsuccessful, i.e., $f(\tilde{p}_1) \leq f(x^{(k)})$ then the second coordinate direction is used indirectly to generate the trial point $\tilde{p}_2$ as shown in Figure 2.5. This process is reiterated. If none of the trial points, $\tilde{p}_i$, (for $i = 1, \cdots, r$), is better than the current iterate $x^{(k)}$ then the POLL step begins at $x^{(k+1)} = x^{(k)}$ with $\Delta_{k+1} = \Delta_k/2$. Figure 2.6 shows that the point $\tilde{p}_1$ is successful and the new POLL step begins at $x^{(k+1)} = \tilde{p}_1$ with $\Delta_{k+1} = 2\Delta_k$.

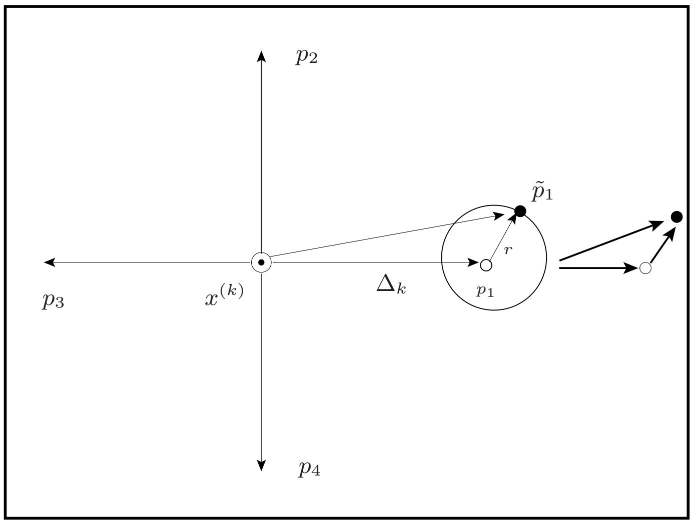

Figure 2.4: The first trial point by MPS.

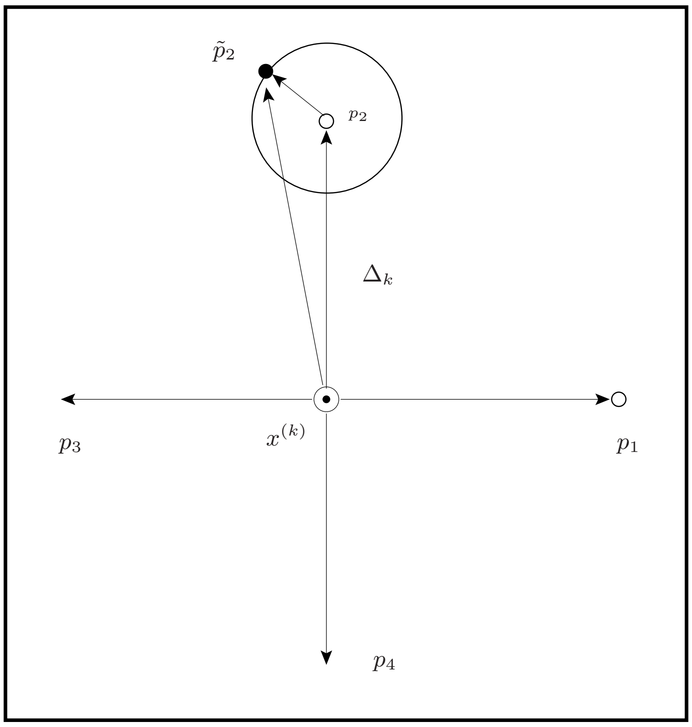

Figure 2.5: The generation of the second trial point by MPS when the first trial point is unsuccessful.

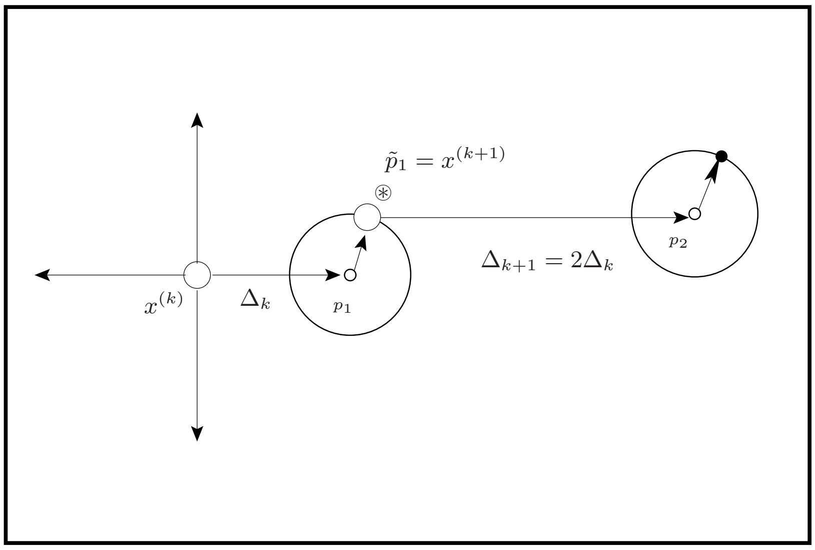

Figure 2.6: The generation of the second trial point of MPS when the first trial point is successful.

2.6 Summary

In this chapter, we have reviewed the PS method. The two main ingredient of the PS method, i.e., the SEARCH and the POLL step were discussed. Thereafter, we give a motivation as to why we discarded the SEARCH step in the PS algorithm. Furthermore, this chapter also elucidates the convergence properties of the PS method. The shortcomings of the PS method are elaborated and some strategies to deal with these shortcomings were suggested. The remaining limitation of PS, i.e., getting trapped in local minimum, will be ameliorated by hybridizing PS with simulated annealing with or without multi level single linkage.

Chapter 3

Simulated annealing for unconstrained global optimization

This chapter forms the core of the hybrid methods that will be designed in Chapter 4. We review the physical annealing and the Metropolis algorithm [36]. We discuss the simulated annealing method [21] for continuous problems. Finally, we present the cooling schedule.

3.1 The physical annealing

The physical annealing is a thermal process for obtaining low energy states of a solid in a heat bath. At first, the solid is heated until all atoms are randomly arranged in a liquid state and then it is cooled by gradually lowering the temperature.

Central to physical annealing is the attainment of the thermal equilibrium. At each temperature, enough time is spent for the solid to reach the thermal equilibrium. If the liquid is cooled slowly enough, then crystals will be formed and the system will have reached its minimum energy at the ground state. However, if the system is cooled quickly, then it will end up in a polycrystalline or amorphous state (local optimal structure), i.e., trapped in a local minimum energy.

Computer simulation of the thermal equilibrium of a collection of atoms at a given temperature was achieved by Metropolis et al. [36]. They suggested an algorithm for obtaining the thermal equilibrium. The algorithm is known as the Metropolis algorithm. The genesis of the simulated annealing method is based on principles of the condensed matter physics, in particular the physical annealing.

3.2 The Metropolis procedure

In 1953, Metropolis et al. [36] used the Monte Carlo method, now known as the Metropolis algorithm, to simulate the collection of particles in thermal equilibrium at a given temperature T. The Metropolis algorithm generates a sequence of states of the system of particles or atoms in the following way. Given a current state, s_i, of the system of particles with corresponding energy E_i, the system is perturbed to a new state s_j with energy E_j. If the change, $\Delta E = E_j - E_i$, represents a reduction in the energy value then the new state s_j is accepted. If the change ΔE represents an increase in the energy value, then the new state is accepted with probability $\exp(-(\Delta E/k_B T)$, where T is the surrounding temperature and k_B is the Boltzmann constant. The acceptance rule described above is called the Metropolis criterion and the algorithm that goes with it, is known as the Metropolis algorithm. The Metropolis algorithm is described as follows:

Algorithm 3.2: The Metropolis Algorithm.

```
set surrounding temperature $T$.
pick initial state $s_i$ at random.
**repeat**
    propose new state $s_j$ picked at random;
    $\Delta E = E_j - E_i$;
    **if** $\Delta E \leq 0$ **then** $p = 1$ **else** $p = \exp(-\Delta E/k_B T)$;
    **if** random$[0, 1) < p$ **then** $s_i = s_j$ ;
**until** thermal equilibrium reached.
```

In the physical annealing, a thermal equilibrium is reached at each temperature if the lowering of the temperature is done sufficiently slowly. Similarly, in the case of the Metropolis algorithm, a thermal equilibrium can be achieved by generating a large number of transitions at a given temperature. At thermal equilibrium, the probability that the system of particles is in state, s_i, with energy E_i is given by the Boltzmann distribution, i.e.,

$$\mathbb{P}_T\{X = s_i\} = \frac{1}{Z(T)} \exp\left(\frac{-E_i}{k_B T}\right), \tag{3.1}$$

where X is a random variable denoting the current state of the system of particles and $Z(T)$ is defined as

$$Z(T) = \sum_j \exp\left(\frac{-E_j}{k_B T}\right). \tag{3.2}$$

3.3 The simulated annealing (SA) method

In 1983, Kirkpatrick et al. [29] designed the simulated annealing algorithm for optimization problems by simulating the physical annealing process. The formulation of the optimization algorithm using the above analogy consists of a series of Metropolis chains used at different values of decreasing temperatures. In this formulation, the system state corresponds to the feasible solution, the energy of the state corresponds to the objective function to be optimized, and the ground state corresponds to the global minimizer.

The general SA consists of two loops. In the inner loop, a number of points in a Markov chain (a Markov chain is a sequence of trial solutions) in the configuration space is produced and some of them are accepted. A trial solution is accepted only if it satisfies the Metropolis criterion. On the other hand, in the outer loop, the temperature is progressively decreased. The whole process depends on the cooling schedule which will be discussed in section 3.5. The original SA algorithm was intended for discrete optimization problem. The general description of the SA algorithm is as follows:

Algorithm 3.3: A general description of the simulated annealing algorithm.

```
Generate the initial configuration $s_i$.
Select an initial temperature $T = T_0$.
while stopping criterion is not satisfied do
begin.
    while no complete Markov chain do
    begin
        generate move $s_j$; compute $f(s_j)$;
        if accept then update solution $s_i$ and $f(s_i)$;
    end;
    decrease $T$;
end.
```

3.4 The simulated annealing algorithm for continuous problems

In this section, we present the SA algorithm for continuous optimization problems. The states s_i and s_j are now denoted by the points x and y respectively in Ω. The corresponding energies of these states, i.e., E_i and E_j are therefore denoted by the function values $f(x)$ and $f(y)$ respectively.

The SA algorithm has been applied to optimization of multimodal continuous functions by fewer authors (Vanderbilt and Louie [50], Alluffi-Pentini et al. [14], Bohachevsky et al. [17] and Wang and Chen [51]) than for the optimization of discrete functions. However, firstly these methods are to some extent different from the original SA approach to discrete optimization. Secondly, their theoretical convergence and sufficient numerical evidences on classified test problems justifying their reliability are missing. Dekkers and Aarts [21] derived a local search-based continuous simulated annealing (LSA) algorithm which is theoretical similar to discrete SA. An aspiration-based simulated annealing (ASA) algorithm [7] and a direct search simulated annealing (DSA) algorithm [9] have also been developed which retains the convergence properties of LSA.

One of the complications arising in going from the discrete to the continuous application of SA is that of the point generation, i.e., generating a new point y from a given point x. One of the possibilities is to generate y using a uniform distribution on Ω; the generation probability distribution function g_{xy}, in this case, is given by $g_{xy} = 1/m(\Omega)$ where $m(\Omega)$ is the Lebesgue measure of the set Ω. However, this choice does not consider the structural information of function values and hence Dekkers and Aarts [21] put forward a mechanism consisting of two possibilities; either a point is drawn uniformly in the search region Ω with probability ψ; or a step is made into a descent direction from the current point x with probability $(1-\psi)$, where ψ is a fixed number in $[0,1)$. Dekkers and Aarts [21] denote this generation mechanism by

$$g_{xy} = \begin{cases} \frac{1}{m(\Omega)} & \text{if} \quad \omega \leq \psi, \\ LS(x) & \text{if} \quad \omega > \psi, \end{cases} \tag{3.3}$$

where ω a uniform random number in $[0,1)$. $LS(x)$ denotes a local technique procedure that generates a point y in a descent direction from x such that $f(y) \leq f(x)$. The local technique $LS(x)$ from x is not a complete local technique but only a few steps of some appropriate descent search. Thus, if $f(y) < f(x)$ then y is not necessarily a local minimum.

Like any other standard SA algorithm based on Markov chains, the essential features of LSA, ASA and DSA are as follows: Starting from a randomly generated initial point $x \in \Omega$ and with an assigned value T_t of the temperature parameter (the temperature counter t is initially set to zero). These methods generates a new trial point, y, using the mechanism (3.3). The objective function $f(y)$ is calculated. If the change $\Delta f_{xy} = f(x) - f(y)$ represents a reduction in the value of the objective function then the new point y is accepted. If the change represents an increase in the objective function value then the new point y is accepted using a Metropolis acceptance probability

$$A_{xy}(T_t) = \min\{\, 1,\, \exp(-(f(x) - f(y)\,)/T_t\,)\,\}. \tag{3.4}$$

This process is repeated for a large enough number of iterations for each T_t. A new Markov chain is then generated (starting from the last accepted point in the previous Markov chain) for a reduced temperature until the algorithm stops. The algorithm for continuous LSA [21] is sketched below.

Algorithm 3.4: The LSA algorithm for the continuous problem.

```
begin
    initialize ($T_0, x$);
    stop criterion := false;
    while stop criterion = false do
    begin
        for $i$ := 1 to $L$ do
        begin
            generate $y$ from $x$ using (3.3);
            if $f(y) - f(x) \leq 0$ then accept;
            else if $\exp(-(f(y) - f(x)\,)/T_t\,)$ >random$[0,1)$ then accept;
            if accept then $x := y$;
        end;
        lower $T_t$;
    end;
end.
```

Remark 3.1:

We will describe the components of the above LSA algorithm, i.e., the values of T_0 and L, the lowering of T_t, and the stop criterion. All these are specified by the cooling schedule which is discussed in the next section

3.5 Cooling schedule

The choice of the cooling schedule (also known as the annealing schedule) is the heart of SA. The cooling schedule affects the number of times the temperature is decreased. We saw earlier in section 3.1 that if a system is cooled hastily, then it will end up with a polycrystalline state, i.e., a system with high energy. Similarly, in the case of an optimization problem, if a fast cooling takes place (i.e., temperature is decreased at a fast rate) then the problem will be trapped in a local minimum. Therefore, in order to avoid being entrapped in a local minimizer, an optimal cooling schedule should be in place. An optimal cooling schedule consists of optimizing four important parameters, namely: the choice of initial temperature T_0, the length L of the Markov chain (the number of trial points for each temperature), stopping criterion and finally the cooling rate of the temperature at each step as cooling proceeds. These parameters are described as follows.

Choice of an initial temperature

The initial temperature value T_0 must be high enough to ensure a large number of acceptances at the initial stages of the algorithm. Using a value that is too high will require more computational effort, while using a low value will rule out the likelihood of an uphill step, thus losing the global feature of the method. Dekker and Aarts [21] suggested an optimal scheme to calculate the initial temperature T_0. In this scheme, a number of trials, say m_0, are generated, and requiring that the initial acceptance ratio $\chi_0 = \chi(T_0)$ be close to 1. The value $\chi(T_0)$is defined as the ratio between the number of accepted trial points and the number of proposed trial points, i.e.,

$$\chi_0 = \frac{m_1 + m_2 \times \exp(-\overline{\Delta f^+}/T_0)}{m_1 + m_2}. \tag{3.5}$$

Here m_1 an m_2 denote the number of trials $(m_0 = m_1 + m_2)$ with $\Delta f_{xy} \leq 0$ and $\Delta f_{xy} > 0$ respectively, and $\overline{\Delta f^+}$ the average value of those Δf_{xy}-values, for which $\Delta f_{xy} > 0$. This initial value of the temperature T_0 given below, is then derived from the equation (3.5), i.e.,

$$T_0 = \overline{\Delta f^+}\left(\ln \frac{m_2}{m_2\chi_0 - m_1(1-\chi_0)}\right)^{-1}. \tag{3.6}$$

Length of the Markov chain

At each temperature, the SA algorithm can be considered as a Markov chain whose length is defined by the number of trial points allowed at this temperature. This number of trial points at each temperature is denoted by the parameter L. Dekkers and Aarts [21] suggested an approach which generates a fixed number of points, i.e.,

$$L = L_0 \times n, \tag{3.7}$$

where n denotes the dimension of the search region Ω and L_0 is a constant.

Cooling rate of the temperature

Once we have the starting temperature, we need to move from one temperature to the other. This can be achieved by using a cooling rate, i.e., the rate at which T decreases at each Markov chain. Dekkers and Aarts [21] suggested the following scheme

$$T_{t+1} = T_t \left(1 + \frac{T_t \times \ln(1+\delta)}{3\sigma(T_t)} \right)^{-1}, \tag{3.8}$$

where $\sigma(T_t)$ is a small positive number and denotes the standard deviation of the values of the cost function at the points in the Markov chain at T_t. The rate of decrease depends on the standard deviation of the objective function values obtained during the Markov chain. The greater the standard deviation, the slower is the decrease. The constant δ is called the distance parameter and determines the speed of decrement of the temperature [21, 33].

Stopping criterion (final temperature)

The algorithm process cannot be performed indefinitely. A stopping criterion must be in place to terminate the algorithm. Dekkers and Aarts [21] proposed a stopping condition based on the idea that the average function value $\overline{f}(T_t)$ over a Markov chain decrease with T_t, so that $\overline{f}(T_t)$ converges to the optimal solution as $T_t \to 0$. If small changes have occurred in $\overline{f}(T_t)$ in two consecutive Markov chains, the procedure will stop. Therefore the simulated annealing algorithm is terminated if

$$\left| \frac{d\overline{f_s}(T_t)}{dT_t} \frac{T_t}{\overline{f}(T_0)} \right| < \varepsilon_s, \tag{3.9}$$

where $\overline{f}(T_0)$ is the mean value of f at the points in the initial Markov chain, $\overline{f_s}(T_t)$ is the smoothed function value of $\overline{f}$ over a number of chains in order to reduce the fluctuations of $\overline{f}(T_t)$, ε_s is a small positive number called the stop parameter. In this dissertation, we will adopt the stopping criterion proposed by Hedar and Fukushima [24], i.e., the algorithm will be terminated after the temperature falls below a certain tolerance, i.e.,

$$T_t \leq \varepsilon. \tag{3.10}$$

The setting of this final temperature in equation (3.10) will give a complete cooling schedule because some problems have high initial temperatures while others have low initial temperatures.

The advantages and disadvantages of the SA method are presented in the next section.

3.6 Advantages and disadvantages of the SA method

In this section, we discuss the advantages and disadvantages of SA method. Some of the advantages of the SA method includes

- SA is able to avoid getting trapped in local minima.
- SA has been proven mathematically to converge to the global minimum given some assumptions on the cooling schedule [21, 32].
- SA is a very simple architecture.

However, SA has some disadvantages, e.g.,

- It is not easy to derive an optimal cooling schedule for SA.
- SA often suffers from slow convergence.

3.7 Summary

In this chapter, we have discussed the physical annealing process, the Metropolis algorithm for simulating such process and the SA algorithm for discrete and continuous variable problems. Finally, we have also mentioned some advantages and disadvantages of SA.

Chapter 4

The hybrid global optimization algorithms based on PS

Up until now, we have not presented how to globalize the PS method. Here by globalization of the PS method, we mean designing a global optimization algorithm based on PS. One way to globalize the PS method is by hybridizing it with a global method. In this chapter, we propose two hybrid methods that combine the PS method and the simulated annealing (SA) method with or without the multi level single linkage method. Both of these hybrid methods use the SA method as the main engine to search for the global minimum. In particular, these hybrid methods use a point generation scheme which is similar to the scheme used in the local search-based simulated annealing (LSA) method [21]. We will briefly describe this generation scheme before the discussion of the hybrid methods.

4.1 Generation mechanism

In any search method, the mechanism for generating a trial point is of paramount importance. In our case, we will propose a generation mechanism through which trial points are generated both globally by using a uniform distribution and locally by using a local technique. A similar strategy was used in LSA [21] and direct search simulated annealing (DSA) [9]. For example LSA uses a gradient-based local technique whereas DSA uses a derivative-free local technique. The local technique used in LSA guarantees local descent while the local technique used in DSA does not. We use the same idea, but unlike LSA, our local technique does not guarantee local descent; it is also entirely different from the local technique used in

DSA. There are several ways of generating trial points from a given point. We present two approaches of the generation mechanism: generation mechanism I (GM-I) and generation mechanism II (GM-II). The two generation mechanisms are described below.

Generation mechanism I (GM-I)

The first approach GM-I is given by the following probability distribution:

$$g_{xy} = \begin{cases} \frac{1}{m(\Omega)} & \text{if} \quad \omega \leq \psi, \\ RD(x) & \text{if} \quad \omega > \psi, \end{cases} \tag{4.1}$$

where ω is a random number in $[0, 1)$, $0 < \psi < 1$ and $RD(x)$ is a local technique which stands for random direction. The procedure involved in $RD(x)$ is described as follows. The local technique $RD(x)$ is invoked if $\omega > \psi$. The direction d_i is first selected randomly from the set of positive spanning directions D defined by equation (2.1) in Chapter 2. Then a trial point in the neighbourhood of x at the t^{th} Markov chain is generated by moving a step of length Δ_t^{sa} along the direction d_i, i.e.,

$$y = x + \Delta_t^{sa} d_i, \tag{4.2}$$

where x is the current iterate and Δ_t^{sa} is a step size parameter (inside the Markov chain of the SA method). The step size parameter Δ_t^{sa} is updated at the end of each Markov chain.

Generation mechanism II (GM-II)

The second approach GM-II is similar to GM-I except that it uses a different local technique. GM-II is given by the following probability distribution:

$$g_{xy} = \begin{cases} \frac{1}{m(\Omega)} & \text{if} \quad \omega \leq \psi, \\ PD(x) & \text{if} \quad \omega > \psi, \end{cases} \tag{4.3}$$

where $PD(x)$ stands for perturbed direction and is described as follows. A trial point y in $PD(x)$ is generated and is given by

$$y = y' + r \times U, \tag{4.4}$$

where y' is the same as the point y in equation (4.2) i.e.,

$$y' = x + \Delta_t^{sa} d_i. \tag{4.5}$$

The direction d_i is chosen randomly from D as in $RD(x)$, U in (4.4) is a normalized directional vector same as in equation (2.14). In essence, the trial point y in equation (4.4) is generated by perturbing the point y generated by $RD(x)$ in equation (4.2).

Calculation of the initial step length

The initial step length in both generation mechanisms is calculated as follows:

$$\Delta_0^{sa} = \zeta \times \max\{\, u_i - l_i \,|\, i = 1, \cdots, n \,\}, \tag{4.6}$$

where $0 < \zeta < 1$, and u_i and l_i are upper and lower bounds of the i^{th} component of x respectively. The initial step length, Δ_0^{sa} in equation (4.6), is independent of the coordinate directions due to the following reasons: it produces, on average, better results and it maintains the step length of the original PS. Notice that Δ_0^{sa} in GM-I and GM-II is much smaller than Δ_0 used in equation (2.12) for MPS. This choice was determined empirically.

Updating of the step size in GM I and II

In both $RD(x)$ and $PD(x)$, the step size parameter Δ_t^{sa} varies with the Markov chain and is updated as follows: At the end of each Markov chain, the following ratio, ra, is computed by

$$ra = \frac{nacp}{nops}, \tag{4.7}$$

where $nops$ is the number of times the local technique $RD(x)$ or $PD(x)$ is invoked to generate trial points and $nacp$ is the number of times the trial points generated by $RD(x)$ or $PD(x)$ are accepted in the Markov chain. The ratio, ra, in equation (4.7) determines whether to increase or decrease the step size parameter Δ_t^{sa}. For instance, if the acceptance rate of the points generated by $RD(x)$ or $PD(x)$ is too high at the t^{th} Markov chain then we increase Δ_{t+1}^{sa} by α% at the end of the t^{th} Markov chain; if the rate is too low we decrease Δ_{t+1}^{sa} by α%. On the other hand, if the rate is close to 50% then we take $\Delta_{t+1}^{sa} = \Delta_t^{sa}$. Thus the next step size parameter Δ_{t+1}^{sa} for the $(t+1)^{\text{th}}$ Markov chain is updated as follows

$$\Delta_{t+1}^{sa} = \begin{cases} (1+\alpha)\Delta_t^{sa} & \text{if} \quad ra \geq \xi, \\ (1-\alpha)\Delta_t^{sa} & \text{if} \quad ra \leq 1-\xi, \\ \Delta_t^{sa} & \text{if} \quad 1-\xi < ra < \xi, \end{cases} \tag{4.8}$$

where ξ is a constant, say $\xi = 0.6$ and the parameter α is such that $0 < \alpha < 1$.

Having discussed the generation mechanism, we are now in a position to present details of the hybrid methods in the following section.

4.2 Proposed hybrid methods

In this section, we present the full details of our main hybrid methods. The first hybrid method is similar to LSA except that it modifies the generation scheme of the LSA method. In particular, it uses the generation mechanism GM-I or GM-II and also updates the step size using equations (4.7)-(4.8). In addition, it keeps a record of the best point found using a singleton set S which is updated with a better point found in the Markov chain. This hybrid is referred to as the modified simulated annealing or MSA.

The second hybrid method extends MSA by incorporating the multi level single linkage (MSL) method [42] within the MSA method. It uses a set S consisting of N points, initially drawn uniformly in the search region Ω. The set S is updated during the course of each Markov chain. This hybrid is referred to as the simulated annealing driven pattern search or SAPS.

4.2.1 Modified simulated annealing (MSA)

Like the LSA method, the MSA method initializes the point x and the parameters of the cooling schedule before the beginning of the first Markov chain. The set S initially contains the point $x_1^\rho = x$.

Structurally, like any other SA method, the MSA method has two loops. In the outer loop, the MSA method, not only decreases the temperature as in LSA, but also updates the step size parameter Δ_t^{sa} using equation (4.8). On the other hand, in the inner loop, MSA differs from LSA in that MSA uses the point generation mechanism GM-I or GM-II and updates the set S as soon as a better point is found in the Markov chain. Therefore the set S contains the best point visited by the MSA method.

The detailed structure of this hybrid is represented in Figure 4.1 using a flowchart.

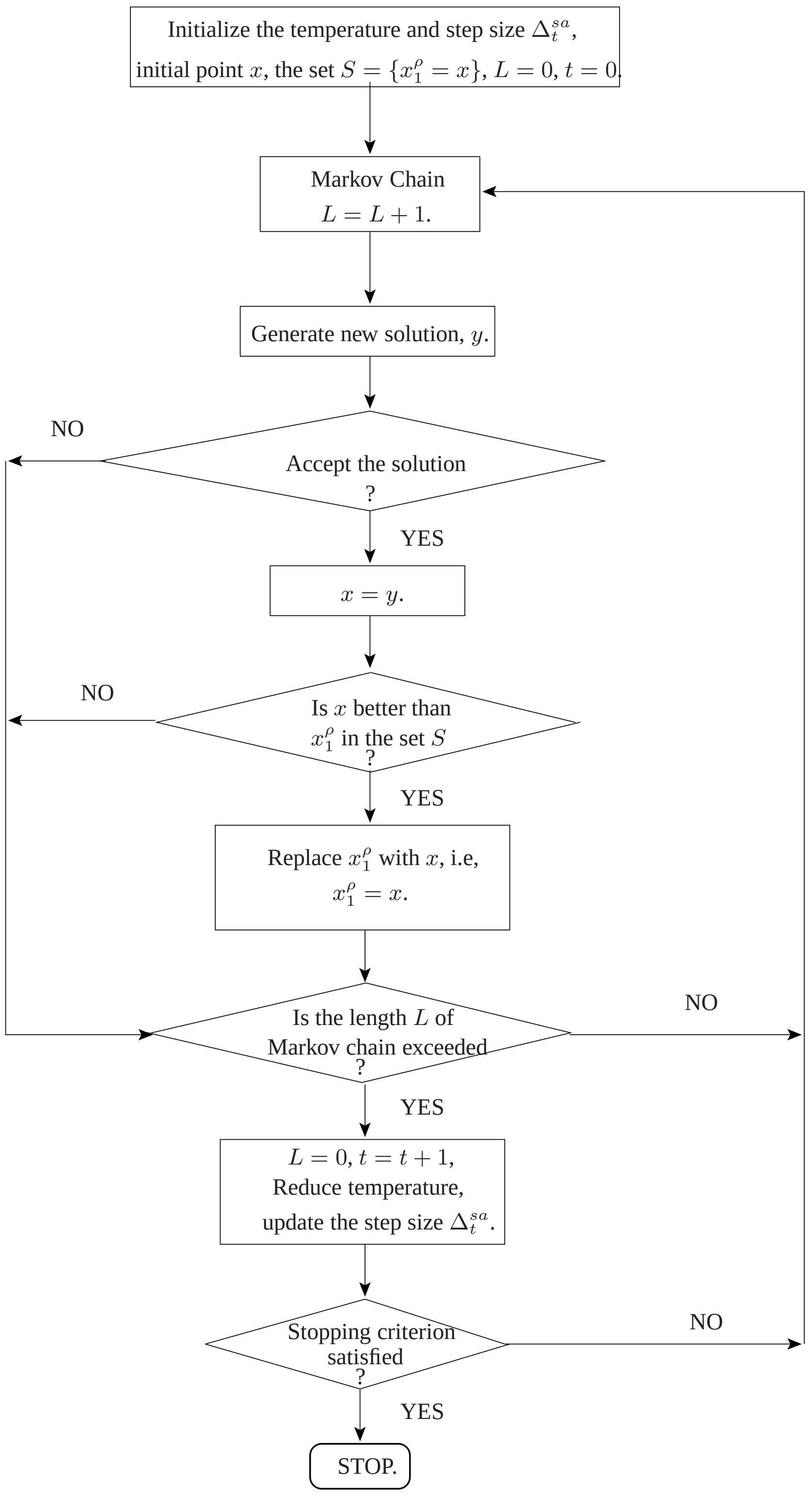
Initialize the temperature and step size Δ_t^{sa},
initial point x, the set $S = \{x_1^\rho = x\}$, $L = 0$, $t = 0$.
Markov Chain
$L = L + 1$.
Generate new solution, y.
NO
Accept the solution ?
YES
$x = y$.
NO
Is x better than x_1^ρ in the set S ?
YES
Replace x_1^ρ with x, i.e, $x_1^\rho = x$.
NO
Is the length L of Markov chain exceeded ?
YES
$L = 0$, $t = t + 1$,
Reduce temperature,
update the step size Δ_t^{sa}.
NO
Stopping criterion satisfied ?
YES
STOP.

Figure 4.1: Flowchart for the MSA algorithm.

The algorithm for MSA is presented below in Algorithm 4.1.

Algorithm 4.1: The MSA Algorithm.

1. **Initialization** : Generate an initial point x. Set $x_1^\rho = x$, $x_1^\rho \in S$. Set the temperature counter $t = 0$. Compute the initial temperature T_0 using equation (3.6). Calculate an initial step size parameter Δ_0^{sa} using equation (4.6).

2. **The inner and outer loops**:

```
while the stopping condition is not satisfied do
begin
    for $i$ := 1 to $L$ do
    begin
        generate $y$ from $x$ using the mechanism in (4.1) or (4.3) ;
        if $f(y) - f(x) \leq 0$ then accept;
        else if $\exp(-(f(y) - f(x))/T_t) >$ random $(0, 1)$ then accept;
        if accept then $x = y$;
        update the set $S$ , i.e., if $f(x) < f(x_1^\rho)$ then $x_1^\rho = x$;
    end;
    $t := t + 1$;
    lower $T_t$ using equation (3.8) ;
    update $\Delta_t^{sa}$ using equation (4.8);
end.
```

Remarks:

4.1. The stopping condition is given by equation (3.10).

4.2. The inner and outer loops of Algorithm 4.1 are similar to those of Algorithm 3.4, i.e., the LSA algorithm, but the significant changes are highlighted in bold.

4.2.2 Simulated annealing driven pattern search (SAPS)

The SAPS hybrid method is the MSA method equipped with the MSL method. Like LSA, it initializes the parameters of the cooling schedule. In addition, it commences by filling the set S with a sample of N ($N >> n$) points uniformly distributed over the search space Ω. This initial set is given by $S = \{x_1^\rho, \cdots, x_N^\rho\}$. The computer implementation of S is done by an array where the best point (having the lowest function value) and the worst point (having the highest function value) are stored in the 1st and the N^{th} positions respectively. Rank ordering of other points between the best point x_1^ρ and the worst point x_N^ρ is not needed.

Structurally, SAPS consists of the inner and the outer loop. It has the same outer loop as MSA where both the temperature and the step size Δ_t^{sa} are updated. The inner loop of SAPS generates trial points using the same generation mechanism as in MSA. However, the inner loop of SAPS differs from that of MSA in the following aspects. At each t^{th} Markov chain of SAPS, the worst point x_N^ρ in S is repeatedly targeted and attempts are made to replace it with the trial point y. That is, if $f(y) < f(x_N^\rho)$ then x_N^ρ in S is replaced by y. The best point x_1^ρ and the worst point x_N^ρ in S are found each time the worst point x_N^ρ is replaced. This process of updating S with new better points continues until all N members of S are replaced. The complete replacement of points in S will require at least N replacements. The replacement process requires more that N replacements especially when a new point y enters the set S (by replacing the worst point x_N^ρ) and becomes the worst point in S. The duration of replacing the whole set S depends on the size S. Therefore the replacement process of S may extend over a number of Markov chains.

When all members of the initial set S, at $t = 0$, are replaced at a (later) Markov chain, the member of S are treated as new. Note that the creation of a new set S can occur either before the completion or at the end of the Markov chain. If a new S is created before the completion of the Markov chain, say at the t^{th} Markov chain, then the t^{th} Markov chain stops temporarily and a single iteration-based MSL is invoked (which is described in section 4.3). After completion of the single iteration-based MSL, the t^{th} Markov chain continues until the length L of the Markov chain is reached. However, if a new S is created at the end of the t^{th} Markov chain, then the single iteration-based MSL is invoked before the next $(t+1)^{\text{th}}$ Markov chain begins. In both cases, the targeting process continues in the subsequent Markov chain(s) until another new S is created and the single iteration-based MSL is invoked. This procedure continues until the stopping criterion is met. Notice that the stopping condition used is that of SA.

4.3 The single iteration-based MSL algortihm.

The process involved in the single iteration-based MSL algorithm is described as follows. The members of S is ordered and a fraction, say γN, $0 < \gamma \leq 1$, of best points is used in the single iteration-based MSL. A local search is carried out from each potential point identified by the single iteration-based MSL algorithm. The best minimizer found by the local search is denoted by x^b. An important parameter of the single iteration-based MSL is the critical distance Δ_t^c and is calculated by

$$\Delta_t^c = \max\{\Delta_t^{sa}, \beta\Delta_0^{sa}\}, \tag{4.9}$$

where $\beta > 1$. Hence when $t = 0$, i.e., initially, $\Delta_0^{sa} = \beta\Delta_0^{sa}$. However, during the initial period of SAPS, the value of Δ_t^{sa} increases. An increase in Δ_t^{sa} indicates that the temperature is high so there is no need to perform a high number of local searches (in order to avoid repetition). This is achieved by setting $\Delta_t^c = \Delta_t^{sa}$, $\Delta_t^{sa} > \beta\Delta_0^{sa}$. This ensures that local searches are performed from few potential points only. A detailed description of the MSL method will be given later in the Appendix A. For a comprehensive literature on MSL, see [42]. Here we present the single iteration-based MSL algorithm.

Algorithm 4.2: The single iteration-based MSL algorithm.

```
Step 1  Order the sample points such that $f(x_i^\rho) < f(x_{i+1}^\rho)$, $1 \leqslant i \leqslant \gamma N - 1$. Set $i := 1$.

Step 2  Apply a local search procedure to $x_1^\rho$. For every $i = 2, \cdots, \gamma N$, apply a local
        search procedure to the sample point $x_i^\rho$ except if there is another sample point, or
        previous detected local minimum within the critical distance $\Delta_t^c$ of $x_i^\rho$. Update $x^b$,
        if necessary.
```

Remark 4.3:

The local search procedure invoked in Algorithm 4.2 is the MPS algorithm presented in Chapter 2. In addition to the parameter Δ_t^c which determines the number of local search in the single iteration-based MSL algorithm, we also have the initial step size parameter Δ_0 of the local search MPS. We take Δ_0 of MPS to be equal to the current step size Δ_t^{sa} at the t^{th} Markov chain, i.e.,

$$\Delta_0 = \Delta_t^{sa}. \tag{4.10}$$

The main structure of the SAPS hybrid is represented in Figure 4.2 using a flowchart.

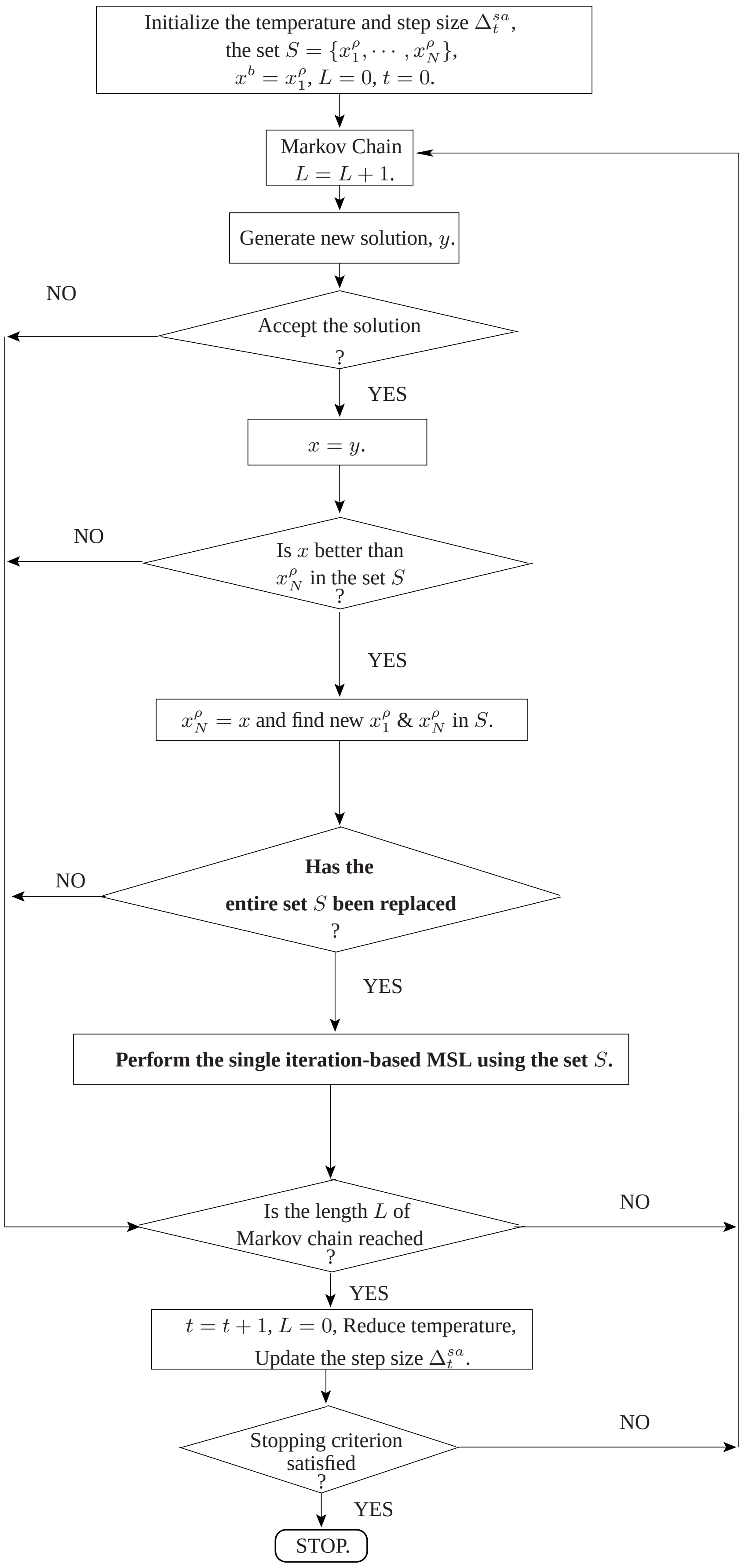
Initialize the temperature and step size Δ_t^{sa}, the set $S = \{x_1^\rho, \cdots, x_N^\rho\}$, $x^b = x_1^\rho$, $L = 0$, $t = 0$.
Markov Chain $L = L + 1$.
Generate new solution, y.
Accept the solution ?
NO
YES
$x = y$.
Is x better than x_N^ρ in the set S ?
NO
YES
$x_N^\rho = x$ and find new x_1^ρ & x_N^ρ in S.
Has the entire set S been replaced ?
NO
YES
Perform the single iteration-based MSL using the set S.
Is the length L of Markov chain reached ?
NO
YES
$t = t + 1$, $L = 0$, Reduce temperature, Update the step size Δ_t^{sa}.
Stopping criterion satisfied ?
NO
YES
STOP.

Figure 4.2: Flowchart for the SAPS algorithm.

The algorithm for SAPS is presented below in Algorithm 4.3.

Algorithm 4.3: The SAPS algorithm.

1. **Initialization**: Same as in step 1 of the MSA algorithm except the initialization of the set $S = \{x_1^\rho, \cdots, x_N^\rho\}$. Let $x^b = x_1^\rho$. Set the parameter value for β.

2. **The inner and outer loops**:

```
while stopping condition is not satisfied do
begin
  for $i$ := 1 to $L$ do
  begin
    generate $y$ from $x$ using the mechanism in (4.1) or (4.3);
    if $f(y) - f(x) \leq 0$ then accept;
    else if $\exp(-(f(y) - f(x))/T_t) > random(0,1)$ then accept;
    if accept then $x = y$;
    **if $f(x_N^\rho) > f(x)$ then $x_N^\rho = x$ and find the best and worst points in the set $S$;**
    **if the set $S$ is replaced entirely then**
    begin
      **if $\Delta_t^{sa} > \beta\Delta_0^{sa}$ then $\Delta_t^c = \Delta_t^{sa}$ else $\Delta_t^c = \beta\Delta_0^{sa}$;**
      **perform the single iteration-based MSL algorithm using the set $S$;**
    end;
  end;
  $t := t + 1$;
  lower $T_t$ using equation (3.8);
  update $\Delta_t^{sa}$ using equation (4.8);
end.
```

Remarks:

4.4. The stopping condition is given by equation (3.10).

4.5. The inner and outer loops of Algorithm 4.3 are similar to those of Algorithm 4.1, i.e., the MSA algorithm, but the significant changes are highlighted in bold.
4.6. The MSL algorithm keeps a record of the number of different minimizer found, as this is needed to stop MSL. On the other hand, Algorithm 4.2, does not keep a record of the number of local minima found. It only keeps a record of the best minimum point, x^b.

4.4 Summary

In this chapter, we have presented two new hybrid global search methods in which the pattern search method is combined with the SA method with and without the multi level single linkage method. Both of these hybrids uses a generation mechanism which is based on the set of positive spanning directions. The performance of these hybrid methods will be discussed in the next chapter.

Chapter 5

Numerical results

In this chapter, we present the computational results in two sections. In the first section, we present the results of the PS method and the MPS method presented in Chapter 2. In the second section, we present results of the two hybrid methods, MSA and SAPS, presented in Chapter 4. We use 50 test problems as benchmark problems to determine the robustness and efficiency of these methods. These problems range from 2 to 20 in dimension and have a variety of inherent difficulties. All the test problems can be found in Appendix B.

The algorithms were run 100 times on each of the 50 test problems to determine the success rate. Therefore there were 5000 runs in total. The success rate, sr, of an algorithm, on a problem is the number of successful runs out of 100 runs. A successful run was counted when the following condition was satisfied,

$$f^* - f_{opt} \leq 0.01, \tag{5.1}$$

where f_{opt} is the known global minimum of the problem and f^* is the best function value obtained when an algorithm terminates. Before we discuss the results on the test problems, we introduce the following notation. We denote the average number of function evaluations and average cpu time by fe and cpu respectively. Note that the average was computed using those runs for which the global minima were obtained, i.e., when sr is positive. We use sr, fe and cpu as the criteria for comparison.

5.1 Numerical results for PS and MPS

In this section, the numerical results of PS and MPS are presented. Initially, we assessed the capabilities of PS in solving the global optimization test problems. We begin by presenting the parameter values of PS and MPS.

5.1.1 Parameter values

In this subsection, we specify suggested parameters values. The initial step size parameters, Δ_0, was set to $\Delta_0 = 1$ and

$$\Delta_0 = \max\{u_i - l_i \,|\, i = 1, \cdots, n\}/2 \tag{5.2}$$

for PS and MPS respectively. We have also tested PS using Δ_0 given by equation (5.2). We denote this version of PS by PS-I. The parameter Δ_0 used by MPS and PS-I depends on the size of the search region Ω. Some of the problems have large search regions, therefore Δ_0 should be proportional to the size of Ω. The MPS method has an additional parameter, namely η, which is used in determining the step r in equation (2.13). We have used $\eta = 0.15$. Our numerical experiments suggest that this is a good choice. A parameter similar to η is used in [17] in the context of local point generation by simulated annealing where $\eta = 0.15$ is also suggested.

Two common parameters of PS, PS-I and MPS are the expansion factor θ_k and the shrinkage factor ϕ_k of equation (2.6). We have taken $\theta_k = 2$ and $\phi_k = \frac{1}{2}$. PS, PS-I and MPS were terminated when the step size parameter Δ_k decreased below a certain tolerance, Δ_{tol}, i.e., when $\Delta_k < \Delta_{tol} = 0.001$.

5.1.2 Numerical comparison

We have implemented PS, PS-I and MPS using the parameter values given in the previous subsection. Each run starts with an initial random point. Rather than using a seed point for the random number generator in all algorithms, we have randomized the initial seed. This means that for each of the 100 runs, we use different initial points in PS, PS-I and MPS. The results of PS, PS-I and MPS are presented in Table 5.1, where the notation, tr, in the last row represents total results, TP denotes the abbreviated names of the test problems and n is the dimension of the test problem. We note that none of the algorithms succeeded in finding the global minimum for the test problems, namely Ackley (ACK), Epistatic Michalewicz (EM), Griewank (GW), Levy and Montalvo 2 (LM2), Miele and Cantrell (MCP), Modified Langerman (ML), Neumaier 2 (NF2), Odd Square (OSP), Paviani (PP), Price's Transistor Modelling (PTM), Rastrigin (RG), Rosenbrock (RB), Salomon (SAL), Schaffer1 (SF1), Schaffer2 (SF2), Schwefel (SWF), Storn's Tchebychev (ST) and Wood (WP). Except for these 18 problems, all other problems were solved by at least one of the algorithms. The total success, sr, is therefore out of 3200 runs. Therefore the results for 32 problems are presented in Table 5.1.

From the total results of Table 5.1, it can be seen that MPS is the best performer. It was successful in 2116 runs out of 3200 runs with total fe=41,900. PS-I is the runner-up. It was successful in 1896 runs out of 3200 runs with total fe=104,553. Finally, PS was the worst performer. It was successful in 1734 runs out of 3200 runs with total fe=115,525. PS-I and MPS perform better than PS because they use an initial step size Δ_0 which takes into account the size of the search regions. This choice is useful especially for problems with large search region. However, Δ_0 used by PS, for unconstrained local optimization, does not consider the size of the search region.

When analysing the numerical results of these algorithms using Table 5.1, the following two questions arise.

- does the choice of initial step size Δ_0 in equation (5.2) improve PS-I?
- does the choice of perturbed coordinate directions improve MPS?

To address the first question, we compare PS and PS-I. The total results of Table 5.1 shows that PS-I is much superior to PS in terms of fe and sr. For instance, PS-I achieved about 9% less fe and 9% more successes in locating the global minimum than PS. This indicates that the choice of the initial step size parameter Δ_0 in equation (5.2) has an effect in improving the convergence rate of PS-I.

Although we have presented the results for PS, here we compare MPS and PS-I to see the effect of perturbed coordinate directions. Table 5.1 shows that out of 50 problems, PS-I and MPS solved 31 and 32 problems respectively. Both PS-I and MPS failed to solve the same 18 problems. In addition, PS-I failed to solve Camel Back6 Hump Problem (CB6). Total results show that MPS has achieved 60% less fe. This difference in the total fe is largely due to two problems, namely Exponential Problem (EXP) and Sinusoidal Problem (SIN). MPS, however, has achieved 220 more successes than PS-I. For the same set of problems, MPS proved its superiority over PS-I. These results demonstrate the effects of the perturbed coordinate directions in PS-I. Hence our modifications to PS-I are fully justified. Finally, we make the observation that despite being a local solver MPS located the global minimum in 2116 runs out of 3200 runs. Hence this potential can be harnessed by incorporating the features of PS-I or MPS in a global solver.

Table 5.1: Comparison of PS, PS-I and MPS using 32 problems.

		PS		PS-I		MPS	
TP	n	fe	sr	fe	sr	fe	sr
AP	2	189	95	196	97	159	88
BL	2	170	100	190	100	160	100
B1	2	221	95	223	94	200	85
B2	2	224	49	229	48	192	57
BR	2	140	100	160	100	150	100
CB3	2	149	57	143	70	142	67
CB6	2	0	0	0	0	149	94
CM	4	306	49	144	97	434	99
DA	2	195	2	170	3	208	4
EP	2	170	3	192	50	174	69
EXP	10	8600	100	7900	100	3200	100
GP	2	193	42	188	49	196	56
GRP	3	833	12	933	15	393	84
H3	3	311	61	300	60	262	65
H6	6	1618	68	1508	61	984	63
HV	3	310	1	290	1	1200	4
HSK	2	158	95	172	99	141	92
KL	4	780	100	640	100	500	100
LM1	3	491	55	482	85	298	84
MC	2	147	75	159	69	141	71
MR	3	3067	75	3400	100	3600	100
MG	4	910	100	1000	100	900	100
MRP	2	187	75	191	68	169	71
MGP	4	173	3	148	5	146	13
NF3	10	9100	100	9192	99	9100	100
PRD	2	167	3	195	4	154	5
PWQ	4	1010	99	1000	100	960	100
SBT	2	150	22	122	27	135	20
S5	4	875	40	897	39	700	40
S7	4	833	24	889	27	641	39
S10	4	848	33	800	25	657	35
SIN	20	83000	1	72500	4	15455	11
tr		115,525	1734	104,553	1896	41,900	2116

5.2 Numerical results for MSA and SAPS

In this section, the numerical results for the two hybrid methods discussed in Chapter 4 are presented in two subsections. Again we have conducted 100 runs on each problem and each run starts with random initial point. Initial members of the set S are also generated randomly. We will first present the numerical results for MSA and for a refinement of MSA. Then we account for the numerical results of SAPS. We begin with the parameter values of MSA and SAPS.

5.2.1 Parameter values

Both MSA and SAPS were implemented using the cooling schedule described in section 3.5, i.e., using equations (3.5)-(3.8). The values of the parameters in the cooling schedule are kept almost the same as those suggested in [21]. Therefore for the cooling schedule of both MSA and SAPS, we use the following common parameter values, namely the acceptance ratio $\chi_0 = 0.9$ and the number of trials $m_0 = 10n$ for the calculation of initial temperature T_0 in (3.6), and constant $L_0 = 10$ for the calculation of the length of the Markov chain in (3.7). We also use the distance parameter $\delta = 0.1$ for determining the decrement of the temperature in (3.8) as suggested in [7, 9, 21]. However, we found δ to be sensitive and hence conducted a number of runs with various values of δ. Note that each run of an algorithm generates a different initial temperature. Hence we present the average initial temperature. The average initial temperature, T_0, for each problem is given in Table 5.4. Note also that each run of an algorithm on a problem, the same initial temperature was generated. This means that the average initial temperature, T_0, for MSA and SAPS on a particular problem is the same. This has been done for a fair comparison. The value of ε in the stopping condition of equation (3.10) is chosen to be $\min(10^{-3}, 10^{-3}T_0)$, as suggested in [24], i.e.,

$$T_t \leq \min(10^{-3}, 10^{-3}T_0). \tag{5.3}$$

The other parameters (other than the cooling schedule parameters) common to both MSA and SAPS include ψ used in the generation scheme (4.1), ζ used in determining the initial step size Δ_0^{sa} in (4.6), and α and ξ used in updating the step size Δ_{t+1}^{sa} in equation (4.8). The parameter $\psi = 0.75$ is used as suggested in [21]. We have carried out numerical testing using a number of values of ξ, e.g., $\xi = 0.5$, $\xi = 0.6$ and $\xi = 0.7$ and the best results were obtained for $\xi = 0.6$. This value produced the overall best results in terms of fe and sr. Hence we use $\xi = 0.6$ for the rest of the numerical experiments. Other parameters are α used in equation (4.8) and ζ used in equation (4.6). We also observe that not all parameters are sensitive.

For example, the parameter α appears to be more sensitive than others, while ζ is less sensitive. Hence we have studied the sensitivity of α and ζ using a series of runs. Each run of MSA or SAPS produces a different number of Markov chains. Hence, we present the average number of Markov chains. We denote the average number of Markov chains by n_{markov}. Note that this average was computed using those runs for which the global minima were obtained.

5.2.2 Numerical studies of the MSA method

In this subsection, we present the results of MSA. We begin with the study of tuning the parameter values of ζ and α in MSA. Fine tuning of parameters is a difficult task and not always easy to see the effects caused by different parameter values. Nonetheless, we try to obtain good values of these parameters. We then compare the MSA algorithm and its refinement. By refinement, we mean that a local search (the MPS algorithm) is performed from the final solution of MSA. Finally, we try to answer an important question. In particular, we answer the question: To what extent does the use of local search affects the performance of MSA.

We begin by studying the effect of varying the parameter ζ. We have used the generation mechanism GM-I for this study. The parameter ζ determines the initial step size Δ_0^{sa} in equation (4.6). For this, we have conducted a series of runs of MSA using the values 0.005, 0.01, 0.03, 0.05 and 0.1 for the parameter ζ. The results are presented in Table 5.2. Although the results are similar for other problems, we present the results for 11 problems as representatives. Table 5.2 shows that the total sr for $\zeta = 0.005$ and $\zeta = 0.1$ are worse than the remaining parameter values. However, the total results in Table 5.2 shows that ζ is less sensitive for the values $\zeta = 0.01,\ 0.03$ and 0.05. All these three values have comparable fe, sr and cpu. We have decided to use the parameter $\zeta = 0.01$ for the rest of the numerical experiments.

Table 5.2: Results of MSA for different values of ζ, GM-I.

	$\zeta = 0.005$		$\zeta = 0.01$		$\zeta = 0.03$		$\zeta = 0.05$		$\zeta = 0.1$	
TP	fe	sr	fe	sr	fe	sr	fe	sr	fe	sr
DA	1981	21	1978	27	2065	27	2026	18	2068	24
GP	2089	19	2064	23	2068	23	2218	20	2370	10
EXP	22258	100	22170	100	22137	100	22142	100	22305	100
GW	38922	100	39427	100	39550	100	39306	100	39712	100
LM2	31455	100	31446	100	31572	100	31635	100	31749	100
NF3	46952	100	47210	100	47250	100	47332	100	47701	100
RG	26817	100	26918	100	26558	100	27094	100	26469	100
RB	50324	100	52046	100	51553	100	52221	100	52128	100
PP	32078	100	31927	100	32629	100	32375	100	32689	100
SAL	22733	80	22907	80	22636	78	22970	90	23920	88
SWF	23675	99	23535	100	23742	100	24408	100	24323	100
tr	299,284	919	301,628	930	301,760	928	303,727	928	305,434	922

Next, we study the effect of α in equation (4.8). The parameter α controls the expansion and reduction of the step size parameter Δ_t^{sa} of equation (4.8). We fix $\zeta = 0.01$ and generate trial points using the generation mechanism GM-I for this study. A series of runs of the MSA algorithm was conducted using the values 0.10, 0.15 and 0.20. We denote the implementation of MSA using $\alpha = 0.10$, $\alpha = 0.15$ and $\alpha = 0.20$ by MSA$_{\alpha=0.10}$, MSA$_{\alpha=0.15}$ and MSA$_{\alpha=0.20}$ respectively. The results for MSA$_{\alpha=0.10}$ and MSA$_{\alpha=0.20}$ are presented in Table 5.3. The total results do not contain the results of 9 problems, namely Epistatic Michalewicz (EM), Gulf Research (GRP), Modified Langerman (ML), Neumaier 2 (NF2), Odd Square (OSP), Price's Transistor Modelling (PTM), Schaffer 2 (SF2), Shekel's Foxholes (FX) and Storn's Tchebychev (ST9) since both MSA$_{\alpha=0.10}$ and MSA$_{\alpha=0.20}$ failed to solve them in all 100 runs. The results of the remaining 41 problems are therefore presented in Table 5.3. The total success, sr, is out of 4100 runs.

A comparison of MSA$_{\alpha=0.10}$ and MSA$_{\alpha=0.20}$ using sr and fe is presented in Table 5.3. MSA$_{\alpha=0.20}$ was successful in 3764 runs out 4100 runs with total fe=513,787. On the other hand, MSA$_{\alpha=0.10}$ was successful in 3541 runs out of 4100 runs with total fe=494,207. The execution time (cpu) for MSA$_{\alpha=0.10}$ and MSA$_{\alpha=0.20}$ are the same. These results shows that MSA$_{\alpha=0.20}$ is superior to MSA$_{\alpha=0.10}$ in terms of sr. The results for MSA$_{\alpha=0.15}$ is presented in a later table. A general trend of the results is that fe and sr increases with α. The reason is because the larger the value of α, the more exploration of the search space is performed. This requires high fe. However, for higher α, the total sr increases. For instance, in Table 5.3 there are at least 3 problems in MSA$_{\alpha=0.20}$, e.g., Dekker (DA), Hartman 3 (H3) and Shubert (SBT), for which sr increased significantly.

Table 5.3: Comparison of different α values in MSA using 41 problems, GM-I.

		$\text{MSA}_{\alpha=0.10}$			$\text{MSA}_{\alpha=0.20}$		
TP	n	fe	sr	cpu	fe	sr	cpu
ACK	10	21139	99	0.080	23240	98	0.090
AP	2	2208	85	0.002	2240	92	0.002
BL	2	2195	100	0.002	2170	100	0.002
B1	2	2684	83	0.004	2427	95	0.003
B2	2	2700	76	0.004	2355	84	0.003
BR	2	2079	91	0.002	1949	95	0.002
CB3	2	2136	100	0.002	2191	100	0.002
CB6	2	2054	87	0.002	2083	98	0.002
CM	4	5519	100	0.008	5808	100	0.008
DA	**2**	**2258**	**2**	**0.002**	**1821**	**81**	**0.002**
EP	2	1417	78	0.002	1166	79	0.002
EXP	10	22324	100	0.060	22101	100	0.060
GP	2	2355	21	0.003	2067	25	0.002
GW	10	38772	100	0.120	39346	100	0.120
H3	**3**	**2349**	**31**	**0.010**	**2203**	**77**	**0.010**
H6	6	8061	97	0.100	8463	96	0.110
HV	3	5036	1	0.006	5391	6	0.006
HSK	2	1479	95	0.002	1234	100	0.002
KL	2	3999	100	0.006	4044	100	0.006
LM1	3	4106	100	0.006	3828	100	0.005
LM2	10	31520	100	0.090	31353	100	0.080
MC	2	1996	99	0.002	1899	99	0.002
MR	3	3436	99	0.004	3317	100	0.004
MCP	4	3256	100	0.008	3352	100	0.008
MRP	2	2486	99	0.003	2334	100	0.002
MGP	2	1845	99	0.007	1581	100	0.005
NF3	10	47357	100	0.110	46880	100	0.110
PP	10	32175	100	0.120	32504	100	0.120
PRD	2	1574	100	0.003	1429	100	0.002
PWQ	4	8359	99	0.010	8274	100	0.010
RG	10	26318	100	0.070	27435	100	0.080
RB	10	50009	100	0.120	52209	100	0.120
SAL	2	21677	82	0.050	23263	84	0.050
SF1	2	1393	100	0.002	1287	100	0.002
SBT	**2**	**1813**	**23**	**0.003**	**1669**	**58**	**0.002**
SWF	10	23451	99	0.060	40633	99	0.110
S5	4	3444	98	0.006	3075	100	0.005
S7	4	3372	100	0.006	3045	99	0.005
S10	4	3474	98	0.007	3233	100	0.006
SIN	20	82204	100	0.610	80190	100	0.580
WP	4	8178	100	0.010	8698	99	0.010
tr		494,207	3541	1.720	513,787	3764	1.748

We have also presented the full results of $\text{MSA}_{\alpha=0.15}$ in Table 5.4. $\text{MSA}_{\alpha=0.15}$ also solved the same 41 problems as solved by $\text{MSA}_{\alpha=0.10}$ and $\text{MSA}_{\alpha=0.20}$. $\text{MSA}_{\alpha=0.20}$ is the best performer in terms of sr but it is the worst performer in terms of fe. $\text{MSA}_{\alpha=0.15}$ performs relatively well in terms of fe and sr. Therefore for the rest of our numerical experiments, we use the parameter value $\alpha = 0.15$.

We now study the results presented in Table 5.4. In particular, we study the effect of the refinement of MSA$_{\alpha=0.15}$. We denote the refined version of MSA$_{\alpha=0.15}$ by MSA-I. The refinement is done by carrying out the local search, MPS, from the final solution of MSA$_{\alpha=0.15}$. The initial step size Δ_0 for MPS is taken as Δ_t^{sa}, where t is the final temperature counter. It also uses the parameter values $\alpha = 0.15$ and $\zeta = 0.01$. We note that MSA-I did not succeed in finding the global minimum for 7 test problems, namely Epistatic Michalewicz (EM), Modified Langerman (ML), Odd Square (OSP), Price's Transistor modelling (PTM) , Schaffer 2 (SF2), Shekel's foxholes (FX) and Storn's Tchebychev (ST9). The results for these 7 problems are not presented in Table 5.4. The average initial temperature for the remaining 43 problems are also presented in Table 5.4. The value of T_0 for the problems, namely EM, ML, OSP, PTM, SF2, FX and ST9 are 0.01, 0.001, 0.47, 8876204, 19.23, 1.14 and 666,623 respectively. Note that some of the problems have high initial temperature, for example, DA, PTM, RB and WP. In Table 5.4, the results in the column under MSA-I have two parts. The results outside the bracket represents the combined fe contributed by MSA$_{\alpha=0.15}$ and the local search (MPS) used for the refinement of the final solution. On the other hand, the results inside the bracket represent the fe contributed by the local search MPS alone.

To answer the question that we posed at the beginning of this subsection, that is, the effect of the refinement of MSA, we compare MSA$_{\alpha=0.15}$ and MSA-I. The total results in Table 5.4 shows that MSA-I is superior to MSA$_{\alpha=0.15}$ by 7% with respect to sr. On the other hand, MSA$_{\alpha=0.15}$ is superior to MSA-I by 6% and 28% with respect to fe and cpu respectively. MSA-I improved the success rate for some of the problems like Bohachevsky 2 (B2), Dekker (DA), Easom (EP), Hartman 3 (H3), Helical (HV), Salomon (SAL) and Shubert (SBT) which are all highlighted in bold. The increase in function evaluation (fe) for most problems using MSA-I can be attributed to the use of local search. For example, the fe for BL is $2184(52)$, where 2184 represent the combined fe for both MSA$_{\alpha=0.15}$ and the local search (MPS) . The number inside the bracket, i.e., 52 represent the fe for MPS only. The remaining fe, i.e, 2132 represent the fe for MSA$_{\alpha=0.15}$ only.

We now study the total number of Markov chains, n_{markov}, in Table 5.4. Table 5.4 shows that MSA$_{\alpha=0.15}$ and MSA-I incurred $n_{markov} = 6793$ and 7162 respectively. The high n_{markov} in MSA-I justifies why it has higher total fe than MSA$_{\alpha=0.15}$. Note that the n_{markov} values for some problems for MSA$_{\alpha=0.15}$ are higher than those of MSA-I. This is because n_{markov} is the average number of Markov chains where the average is taken over the successful runs. MSA-I has more successful runs than MSA$_{\alpha=0.15}$. Notice that for some problems the values of n_{markov} are the same and this has been indicated with boxes in Table 5.4.

The total results for MSA-I in Table 5.4 includes the results of Gulf Research Problem (GRP) and Neumaier 2 Problem (NF2) where MSA$_{\alpha=0.15}$ failed. We compare MSA$_{\alpha=0.15}$ and MSA-I excluding these two functions. The total cpu and fe for MSA-I, without these two functions, are 1.77 and 510,620 respectively. Therefore, both MSA$_{\alpha=0.15}$ and MSA-I have similar cpu and fe if we exclude the results of these two functions from the total results of Table 5.4.

Table 5.4: Comparison of MSA$_{\alpha=0.15}$ and MSA-I using 43 problems, GM-I.

			MSA$_{\alpha=0.15}$				MSA-I			
TP	n	T_0	fe	sr	cpu	n_{markov}	fe	sr	cpu	n_{markov}
ACK	10	28.84	22594	99	0.080	225	22759 (289)	100	0.090	224
AP	2	3949.00	2154	97	0.002	107	2234 (76)	99	0.003	106
BL	2	96.97	2132	100	0.002	$\boxed{106}$	2184 (52)	100	0.002	$\boxed{106}$
B1	2	15157.49	2436	96	0.003	121	2478 (51)	100	0.003	120
B2	**2**	**15154.47**	**2494**	**80**	**0.003**	**124**	**2522 (51)**	**95**	**0.003**	**123**
BR	2	446.91	2011	95	0.002	100	2044 (64)	100	0.002	98
CB3	2	2394.86	2148	100	0.002	107	2198 (50)	100	0.002	107
CB6	2	8410.76	2099	98	0.002	104	2149 (60)	100	0.002	104
CM	4	7.49	5767	100	0.008	143	5801 (34)	100	0.008	143
DA	**2**	**9469730.00**	**1978**	**27**	**0.002**	**98**	**1900 (21)**	**56**	**0.002**	**93**
EP	**2**	**0.90**	**1148**	**89**	**0.002**	**57**	**1098 (52)**	**99**	**0.002**	**51**
EXP	10	2.20	22170	100	0.06	$\boxed{221}$	22290 (120)	100	0.060	$\boxed{221}$
GP	2	63901.00	2064	23	0.002	102	2215 (70)	82	0.002	106
GW	10	1583.55	39427	100	0.120	393	40561 (1134)	100	0.130	393
GRP	3	23.23	0	0	0.000	0	3646 (360)	15	0.520	109
H3	**3**	**3.68**	**2090**	**52**	**0.010**	**69**	**2074 (154)**	**100**	**0.010**	**63**
H6	6	2.63	8269	97	0.110	137	8432 (311)	100	0.110	135
HV	**3**	**69139.00**	**4809**	**7**	**0.006**	**159**	**5688 (411)**	**26**	**0.007**	**175**
HSK	2	1.97	1324	96	0.002	65	1380 (77)	100	0.002	64
KL	2	0.23	4049	100	0.006	$\boxed{100}$	4212 (163)	100	0.006	$\boxed{100}$
LM1	3	201.55	3963	100	0.005	131	4060 (97)	100	0.005	131
LM2	10	79.16	31446	100	0.090	314	31608 (162)	100	0.090	314
MC	2	11.30	1925	100	0.002	95	1985 (60)	100	0.002	95
MR	3	348426.00	3346	100	0.005	$\boxed{111}$	8870 (5524)	100	0.006	$\boxed{111}$
MCP	4	8.64	3371	100	0.008	83	3967 (595)	100	0.009	83
MRP	2	154127.00	2283	100	0.002	113	2294 (12)	100	0.002	113
MGP	2	2.68	1641	100	0.006	81	1670 (29)	100	0.006	81
NF2	4	689690.00	0	0	0.000	0	12191 (117)	1	0.110	276
NF3	10	18299.00	47210	100	0.110	471	49002 (1792)	100	0.120	471
PP	10	184.43	31927	100	0.110	318	32027 (100)	100	0.110	318
PRD	2	0.19	1483	100	0.002	73	1546 (63)	100	0.002	73
PWQ	4	36236.00	8296	100	0.010	207	8424 (128)	100	0.010	207
RG	10	616.90	26918	100	0.070	268	26971 (53)	100	0.070	268
RB	10	9653091.00	52046	100	0.120	$\boxed{520}$	52100 (54)	100	0.120	$\boxed{520}$
SAL	**2**	**57.17**	**22907**	**80**	**0.050**	**228**	**22978 (610)**	**96**	**0.050**	**222**
SF1	2	0.69	1373	100	0.002	68	1486 (113)	100	0.002	68
SBT	**2**	**212.80**	**1699**	**66**	**0.003**	**84**	**1712 (18)**	**80**	**0.003**	**80**
SWF	10	11908.00	23535	100	0.070	235	24787 (1252)	100	0.070	235
S5	4	10.37	3208	99	0.005	79	3256 (58)	100	0.005	79
S7	4	10.60	3119	99	0.006	77	3172 (68)	100	0.006	77
S10	4	10.64	3248	100	0.007	80	3319 (71)	100	0.008	80
SIN	20	4.99	81700	100	0.610	$\boxed{408}$	82657 (957)	100	0.620	$\boxed{408}$
WP	4	1452635.00	8484	100	0.010	211	8510 (26)	100	0.010	211
tr			496,291	3700	1.728	6793	526,457(15,559)	3949	2.400	7162

Up to now, we have presented the result for MSA$_{\alpha=0.15}$ using the generation mechanism GM-I. It would be interesting to see how MSA$_{\alpha=0.15}$ performs with the generation mechanism GM-II. Therefore, we present the full results of MSA$_{\alpha=0.15}$ using GM-II in Table 5.5. In this table, we note that MSA$_{\alpha=0.15}$ was not successful on the same 9 problems as in MSA$_{\alpha=0.15}$, Table 5.4. MSA$_{\alpha=0.15}$ was successful in 3657 runs out of 4100 runs with total fe=494,469 as shown in Table 5.5. On the other hand, MSA$_{\alpha=0.15}$ was successful in 3700 runs out of 4100 runs with total fe=496,291 as shown in Table 5.4. One can conclude that GM-I and GM-II are comparable in terms of fe, sr and cpu.

Finally, note that the MSA algorithm performed well in separable or closely separable multimodal functions, e.g., Ackley (ACK), Levy and Montalvo (LM 1 & 2) and Rastrigin (RG), and Schwefel (SWF), as opposed to a number of non-separable functions, e.g., Dekkers and Aarts (DA), Schaffer 2 (SF2), and Goldstein and Price (GP). This is evident in Table 5.4. For instance, MSA was successful in 499 runs out of 500 runs for the case of the above 5 separable or closely separable functions. On the other hand, MSA was successful only in 50 runs out of 300 runs for the above 3 nonseparable functions. One important feature of the generation mechanism employed in MSA is that a coordinate step is performed in such a way that a single variable is changed to obtain a trial point in GM-I. We believe that this feature favours the separable functions. A further research can involve understanding the reasons for failure of MSA on some nonseparable functions. We have stated this in the conclusion.

Table 5.5: Results of MSA$_{\alpha=0.15}$ using 43 problems, GM-II.

TP	n	fe	sr	cpu
ACK	10	20972	99	0.080
AP	2	2108	91	0.002
BL	2	2147	100	0.002
B1	2	2484	94	0.003
B2	2	2458	83	0.003
BR	2	1930	96	0.002
CB3	2	2190	100	0.003
CB6	2	2026	96	0.002
CM	4	5633	100	0.009
DA	2	1952	30	0.003
EP	2	1135	85	0.002
EXP	10	22073	100	0.060
GP	2	2093	15	0.003
GW	10	39335	100	0.140
H3	3	2139	38	0.010
H6	6	7843	90	0.010
HV	3	5441	4	0.007
HSK	2	1320	97	0.002
KL	2	4030	100	0.007
LM1	3	3936	100	0.006
LM2	10	31397	100	0.090
MC	2	1922	99	0.002
MR	3	3269	100	0.005
MCP	4	3021	100	0.008
MRP	2	2271	100	0.003
MGP	4	1670	100	0.006
NF3	10	47083	100	0.120
PP	2	32444	100	0.120
PRD	4	1504	100	0.003
PWQ	9	8412	100	0.010
RG	10	26894	100	0.080
RB	10	50932	100	0.130
SAL	2	23599	82	0.060
SF1	2	1355	100	0.002
SBT	2	1668	61	0.003
SWF	10	24839	100	0.070
S5	4	3187	99	0.006
S7	4	3123	100	0.006
S10	4	3260	99	0.007
SIN	20	80892	100	0.620
WP	4	8482	99	0.010
tr		494,469	3657	1.718

5.2.3 Numerical studies of the SAPS method

In this subsection, we present the results of SAPS. The SAPS algorithm is implemented using the same parameter values of the cooling schedule as in MSA. We however study the effect of δ in equation (3.8). In addition to these parameters values, there are two other parameters common to MSA and SAPS, namely ζ in equation (4.6) and α in equation (4.8). Good values of these parameters were empirically obtained in subsection 5.2.2 for MSA. We therefore use the same values in the implementation of SAPS, i.e., we use $\zeta = 0.01$ and $\alpha = 0.15$.

Other parameters of SAPS are β used in equation (4.9), the size N of S and γ. Of these parameters, γ is used by MSL. In this subsection, we study these parameters and the parameter δ in equation (3.8) empirically in order to obtain suitable values for them. The parameter δ was found to be sensitive in our study. Hence, we have presented a series of results with various values of δ. Before we present the results, we introduce some notations. We denote the average number of times the single iteration-based MSL algorithm is performed per run by n_c and the average number of times MPS is performed per MSL by n_{ps}. We also denote the average number of MPS, out of n_{ps}, that obtains the global minimum by n_g.

We begin our numerical investigation with the distance parameter δ. We fix $N = 3n$ and $\gamma = 1$ for this study. We use the generation mechanism GM-I. We run SAPS using different values of δ, namely 0.1, 0.3 and 0.5. The results are presented in Table 5.6. We note that SAPS did not succeed in finding the global minimum of 7 test problems for all δ values, namely Epistatic Michalewicz (EM), Modified Langerman (ML), Odd Square (OSP), Price's Transistor modelling (PTM) , Schaffer 2 (SF2), Shekel's foxholes (FX) and Storn's Tchebychev (ST9). The results of the 43 problems are therefore presented in Table 5.6.

From the total results in Table 5.6, we see that the SAPS algorithm was successful in 4176, 4105 and 4022 runs out of 4300 runs for $\delta = 0.1$, 0.3 and 0.5 respectively. SAPS achieved these successes for total fe equal to 1,021,630, 801,985 and 767,911 for $\delta = 0.1$, 0.3 and 0.5 respectively. The above results shows that both fe and sr decrease as δ increases. This is because the temperature decreases slowly whenever δ is small and hence more fe is needed in order to converge. There are at least 3 problems, e.g., Goldstein and Price (GP), Salomon (SAL) and Shubert (SBT) for which sr differs significantly. We have highlighted these 3 problems in bold. Clearly $\delta = 0.1$ is the best value in terms of sr and $\delta = 0.5$ is the best value in terms of fe. We have decided to choose the best parameter based on success rate, sr. Therefore, we use the parameter $\delta = 0.1$ for the rest of the numerical experiments.

Table 5.6: Results of SAPS for different values of δ, GM-I.

		$\delta = 0.1$			$\delta = 0.3$			$\delta = 0.5$		
TP	n	fe	sr	cpu	fe	sr	cpu	fe	sr	cpu
ACK	10	34797	100	0.140	22706	100	0.080	21819	100	0.070
AP	2	2695	99	0.003	1466	100	0.002	1292	100	0.001
BL	2	2808	100	0.003	1578	100	0.002	1302	100	0.001
B1	2	2873	100	0.003	1818	100	0.002	1559	100	0.002
B2	2	2881	99	0.003	1843	97	0.002	1563	95	0.002
BR	2	2468	100	0.002	1567	100	0.002	1480	100	0.002
CB3	2	2414	100	0.003	1320	100	0.002	1091	100	0.001
CB6	2	2525	100	0.003	1581	100	0.002	1429	100	0.002
CM	4	6515	100	0.009	3263	100	0.005	2643	100	0.004
DA	2	2924	98	0.003	2649	96	0.003	2487	96	0.002
EP	2	1648	99	0.003	1492	94	0.002	1392	96	0.002
EXP	10	26465	100	0.070	12381	100	0.030	9048	100	0.002
GP	**2**	**2634**	**99**	**0.003**	**1678**	**88**	**0.002**	**1497**	**64**	**0.002**
GW	10	52196	100	0.150	28762	100	0.080	22314	100	0.060
GRP	3	6467	100	1.040	10007	100	1.610	11408	100	1.840
H3	3	3120	100	0.020	2600	100	0.020	2154	100	0.010
H6	6	18184	100	0.240	11390	100	0.140	9778	100	0.120
HV	3	12586	81	0.010	8453	66	0.007	7524	64	0.006
HSK	2	2000	100	0.003	1318	100	0.002	1021	99	0.002
KL	2	4811	100	0.007	2056	100	0.003	1505	100	0.002
LM1	3	4929	100	0.006	2684	100	0.003	2208	100	0.003
LM2	10	37091	100	0.110	17177	100	0.050	13399	100	0.040
MC	2	2396	100	0.003	1383	100	0.002	1188	100	0.001
MR	3	17985	100	0.020	16201	100	0.020	13052	100	0.010
MCP	4	12741	100	0.030	10142	100	0.020	8915	100	0.020
MRP	2	2709	100	0.003	1672	100	0.002	1475	100	0.002
MGP	4	1918	100	0.007	1094	100	0.004	913	100	0.003
NF2	10	16967	20	0.140	10433	32	0.100	9952	21	0.100
NF3	10	237368	100	0.410	275763	100	0.430	305831	100	0.540
PP	2	37587	100	0.140	17810	100	0.060	13315	100	0.004
PRD	4	1871	100	0.003	1132	100	0.002	922	100	0.002
PWQ	9	10829	100	0.010	5916	100	0.006	4883	100	0.005
RG	10	44180	100	0.130	28682	100	0.070	25312	100	0.060
RB	10	89715	100	0.190	60083	100	0.110	53265	100	0.090
SAL	**2**	**26641**	**97**	**0.060**	**12230**	**80**	**0.020**	**10255**	**63**	**0.02**
SF1	2	1835	100	0.002	1284	100	0.001	1000	100	0.001
SBT	**2**	**2122**	**84**	**0.003**	**1429**	**52**	**0.002**	**1157**	**24**	**0.002**
SWF	10	45701	100	0.120	39786	100	0.100	33531	100	0.080
S5	4	5782	100	0.008	4482	100	0.006	4240	100	0.006
S7	4	5699	100	0.009	4596	100	0.007	4276	100	0.007
S10	4	5818	100	0.010	4632	100	0.008	4153	100	0.008
SIN	20	206177	100	1.350	157943	100	0.960	145540	100	0.850
WP	4	10558	100	0.010	5503	100	0.006	4823	100	0.006
tr		1,021,630	4176	4.488	801,985	4105	3.987	767,911	4022	3.993

We now study the effect of varying the parameter β used in equation (4.9). We fix $\delta = 0.1$, $\gamma = 1$ and $N = 3n$ use the generation scheme GM-I for this study. We run SAPS algorithm using three values of β, namely 10, 15 and 20. The results are presented in Table 5.7. We note that SAPS failed on the same problems, e.g., EM, ML OSP, PTM, SF2, FX and ST9 for each value of β. SAPS has a positive success rate on the remaining 43 problems for each value of β. Hence Table 5.7 does not contain the results for these 7 problems.

The SAPS algorithm was successful in 4178, 4179 and 4176 runs out of 4300 runs for $\beta = 10$, 15 and 20 respectively. SAPS achieved these successes for total fe equal to 1,384,360, 1,147,738 and 1,021,630 for $\beta = 10$, 15 and 20 respectively. Total results shows that fe decreases as β increases. The decrease in fe as β increases can be justified as follows. We know that the number of local searches performed in the single iteration-based MSL algorithm depends on the length of the critical distance Δ_t^c, which is given by

$$\Delta_t^c = \max\{\Delta_t^{sa}, \beta\Delta_0^{sa}\}.$$

The critical distance Δ_t^c takes the value $\beta\Delta_0^{sa}$ in cases where $\beta\Delta_0^{sa} > \Delta_t^{sa}$. Hence, the parameter β has an effect on the critical distance. Therefore, the larger β is, the lesser the number of local searches are performed resulting in lesser fe. We will explain later why fe decreases with β using the information in Table 5.8. On the other hand, the total results also shows that sr is insensitive to β. We have also tested SAPS with $\beta = 30$. The results have shown a slight decrease in sr for this value. Hence, we use $\beta = 20$ for the rest of numerical study.

Some additional results of the implementation of SAPS that produced the results in Table 5.7 will be presented next in Table 5.8.

Table 5.7: Results of SAPS for different values of β, GM-I.

		$\beta = 10, \gamma = 1$			$\beta = 15, \gamma = 1$			$\beta = 20, \gamma = 1$		
TP	n	fe	sr	cpu	fe	sr	cpu	fe	sr	cpu
ACK	10	46520	100	0.150	39309	100	0.130	34797	100	0.140
AP	2	2778	100	0.003	2714	100	0.003	2695	99	0.003
BL	2	2755	100	0.003	2797	100	0.003	2808	100	0.003
B1	2	2917	100	0.003	2958	100	0.003	2873	100	0.003
B2	2	3029	99	0.003	2883	99	0.003	2881	99	0.003
BR	2	2549	100	0.003	2403	100	0.002	2468	100	0.002
CB3	2	2428	100	0.003	2449	100	0.002	2414	100	0.003
CB6	2	2617	100	0.003	2607	100	0.002	2525	100	0.003
CM	4	6816	100	0.010	6699	100	0.009	6515	100	0.009
DA	2	3283	99	0.003	3068	97	0.003	2924	98	0.003
EP	2	1776	93	0.010	1737	95	0.002	1648	99	0.003
EXP	10	36199	100	0.080	29323	100	0.070	26465	100	0.070
GP	2	2680	98	0.030	2630	99	0.003	2634	99	0.003
GW	10	71112	100	0.190	58503	100	0.160	52196	100	0.150
GRP	3	9331	100	1.400	8170	100	1.300	6467	100	1.040
H3	3	3614	100	0.020	3455	100	0.002	3120	100	0.020
H6	6	21792	100	0.270	19524	100	0.240	18184	100	0.240
HV	3	13310	85	0.020	12969	81	0.010	12586	81	0.010
HSK	2	2062	100	0.003	2116	100	0.003	2000	100	0.003
KL	2	5023	100	0.007	4892	100	0.007	4811	100	0.007
LM1	3	4921	100	0.006	5013	100	0.006	4929	100	0.006
LM2	10	47771	100	0.120	40621	100	0.110	37091	100	0.110
MC	2	2405	100	0.003	2415	100	0.002	2396	100	0.003
MR	3	21297	100	0.020	18251	100	0.010	17985	100	0.020
MCP	4	16840	100	0.040	16308	100	0.003	12741	100	0.030
MRP	2	2887	100	0.003	2717	100	0.003	2709	100	0.003
MGP	4	1909	100	0.007	1913	100	0.007	1918	100	0.007
NF2	10	14804	26	0.140	15395	26	0.140	16967	20	0.140
NF3	10	287709	100	0.460	251521	100	0.390	237368	100	0.410
PP	2	46081	100	0.160	39235	100	0.130	37587	100	0.140
PRD	4	1961	100	0.003	1862	100	0.003	1871	100	0.003
PWQ	9	11844	100	0.010	11318	100	0.010	10829	100	0.010
RG	10	44389	100	0.120	44252	100	0.110	44180	100	0.130
RB	10	179634	100	0.300	130350	100	0.240	89715	100	0.190
SAL	2	28942	96	0.060	27526	99	0.060	26641	97	0.060
SF1	2	1980	100	0.002	1919	100	0.002	1835	100	0.002
SBT	2	2034	82	0.003	2128	83	0.003	2122	84	0.003
SWF	10	63999	100	0.160	45541	100	0.120	45701	100	0.120
S5	4	6593	100	0.009	5913	100	0.008	5782	100	0.008
S7	4	6620	100	0.010	5999	100	0.009	5699	100	0.009
S10	4	6522	100	0.010	5804	100	0.010	5818	100	0.010
SIN	20	325257	100	2.060	248712	100	1.520	206177	100	1.350
WP	4	15370	100	0.020	11819	100	0.010	10558	100	0.010
tr		1,384,360	4178	5.940	1,147,738	4179	4.868	1,021,630	4176	4.488

Having established the effect of β in sr and fe in Table 5.7, we now study the effect of β in n_{ps} and n_c. The values n_{ps} and n_c are the direct consequences of the implementation of MSL in SAPS. We now present the data for n_{ps} and n_c in Table 5.8. Notice that the results in Table 5.7 and 5.8 were obtained using the same implementation of SAPS.

The total results in Table 5.8 shows that SAPS performed $n_c = 175$, 173 and 170 single iteration-based MSL for $\beta = 10$, 15 and 20 respectively. The total number of local searches in the above MSL call were $n_{ps} =$ 299, 258 and 237 respectively for $\beta = 10$, 15 and 20. Although the total number of local searches in each of the above cases is high but the number of local search per MSL is considerably low. For example, there were $\frac{175}{299}$ (=1.7), 1.5 and 1.4 local searches per MSL for $\beta = 10$, 15 and 20 respectively.

On the other hand, we were encouraged to see the results for n_g. For example, the number of successful local searches were $n_g = 227$ out of $n_{ps} = 299$, $n_g = 199$ out of $n_{ps} = 258$, and $n_g = 181$ out of $n_{ps} = 237$ for $\beta = 10$, 15 and 20 respectively. The decrease in value of n_{ps} and n_g as β increases justifies why fe decreases with β as we have seen in Table 5.7. The above results shows that there were 76%, 77% and 76% local searches were successful in locating the global minimum value. Indeed, there are a number of problems, e.g., GW, H3 and MC, where 100% local searches were successful, i.e., $\boxed{n_{ps} = n_g}$. On the other hand, there are some problems where not all local search n_{ps} produced the global minimum, such as the problems MGP, SAL and SBT where $\boxed{n_{ps} \neq n_g}$.

Table 5.8: Results of SAPS for different values of β, GM-I.

		$\beta = 10$		$\beta = 15$		$\beta = 20$	
TP	n	$n_{ps}(n_g)$	n_c	$n_{ps}(n_g)$	n_c	$n_{ps}(n_g)$	n_c
ACK	10	9 (5)	3	7 (4)	3	6 (4)	3
AP	2	5 (4)	4	4 (4)	4	4 (3)	4
BL	2	5 (5)	3	5 (5)	3	5 (5)	3
B1	2	4 (3)	3	4 (3)	3	3 (3)	3
B2	2	4 (2)	3	3 (2)	3	3 (2)	3
BR	2	4 (4)	4	4 (4)	3	4 (4)	4
CB3	2	3 (3)	3	3 (3)	3	3 (3)	3
CB6	2	5 (5)	3	4 (4)	4	4 (4)	4
CM	4	5 (4)	3	5 (4)	3	5 (3)	3
DA	2	9 (6)	7	8 (6)	6	6 (5)	6
EP	2	9 (8)	8	8 (8)	8	8 (7)	7
EXP	10	7 (7)	2	5 (5)	2	4 (4)	2
GP	2	4 (2)	3	4 (2)	3	4 (2)	3
GW	**10**	**6 (6)**	3	**4 (4)**	**3**	**3 (3)**	**3**
GRP	3	4 (4)	3	4 (4)	3	3 (3)	2
H3	**3**	**7 (7)**	**5**	**6 (6)**	**5**	**5 (5)**	**4**
H6	6	16(16)	10	13(13)	10	12(12)	10
HV	3	6 (2)	5	6 (2)	6	6 (2)	5
HSK	2	6 (6)	5	6 (6)	5	5 (5)	5
KL	2	3 (3)	2	3 (3)	2	2 (2)	2
LM1	3	5 (3)	3	5 (3)	3	5 (3)	3
LM2	10	7 (6)	3	5 (4)	2	4 (3)	2
MC	**2**	**5 (5)**	**4**	**5 (5)**	**4**	**4 (4)**	**4**
MR	3	4 (4)	3	4 (3)	3	4 (3)	3
MCP	4	11(11)	4	10(10)	5	8 (8)	4
MRP	2	5 (5)	3	4 (4)	3	4 (4)	3
MGP	**4**	**4 (2)**	**3**	**3 (2)**	**3**	**3 (2)**	**3**
NF2	10	7 (2)	4	6 (2)	3	6 (2)	4
NF3	10	34(34)	24	30(30)	24	29(29)	24
PP	2	6 (6)	3	4 (4)	3	4 (4)	3
PRD	4	4 (4)	3	3 (3)	3	3 (3)	3
PWQ	9	6 (6)	3	5 (5)	3	4 (4)	3
RG	10	9 (3)	3	9 (3)	3	9 (3)	3
RB	10	7 (2)	3	6 (2)	3	5 (2)	3
SAL	**2**	**7 (1)**	**3**	**5 (1)**	**3**	**4 (1)**	**3**
SF1	2	4 (4)	3	4 (3)	3	3 (3)	3
SBT	**2**	**4 (2)**	**2**	**4 (2)**	**2**	**4 (2)**	**2**
SWF	10	6 (3)	3	5 (3)	3	5 (3)	3
S5	4	9 (4)	3	7 (4)	3	7 (3)	3
S7	4	8 (4)	4	7 (3)	3	7 (3)	3
S10	4	8 (4)	3	7 (3)	3	7 (3)	3
SIN	20	13(7)	3	10(5)	3	8 (4)	3
WP	4	5 (3)	3	4 (3)	3	4 (3)	3
tr		299(227)	175	258 (199)	173	237 (181)	170

We have so far conducted numerical testing of SAPS for various parameter values using the generation mechanism GM-I. The results obtained were very satisfactory. We have shown that the best results of SAPS were obtained for $\beta = 20$ and $\delta = 0.1$. It will be interesting to see the results of SAPS for the above parameters values using the generation mechanism GM-II. Hence, the results of SAPS using GM-II are presented in Table 5.9. Note that SAPS was not successful for the same 7 problems as in SAPS of Table 5.7. From the total results in Table 5.9, we see that SAPS was successful in 4181 runs out of 4300 runs with total fe=1,030,017. On the other hand, Table 5.7 shows that SAPS was successful in 4176 runs out 4300 runs with total fe=1,021,630. These results show that SAPS is insensitive to GM-I and GM-II. This is because $r \to 0$ faster than $\Delta_t^{sa} \to 0$. Hence GM-II $\to$ GM-I for a smaller ε in the stoppig condition of equation (3.10). However, our experience have shown that SAPS becomes sensitive to GM-I and GM-II for larger ε in the stopping condition. We have decided to choose GM-I for the rest of our numerical studies.

Table 5.9: Results of SAPS for $\beta = 20$ using 43 problems, GM-II.

TP	n	fe	sr	cpu
ACK	10	34793	100	0.120
AP	2	2755	100	0.003
BL	2	2754	100	0.003
B1	2	2934	100	0.003
B2	2	2845	98	0.003
BR	2	2487	100	0.003
CB3	2	2406	100	0.003
CB6	2	2534	100	0.003
CM	4	6697	100	0.010
DA	2	3218	98	0.003
EP	2	1742	97	0.003
EXP	10	26559	100	0.070
GP	2	2653	100	0.003
GW	10	53351	100	0.160
GRP	3	2653	100	0.890
H3	3	3505	100	0.020
H6	6	16650	100	0.210
HV	3	12439	84	0.010
HSK	2	2016	100	0.003
KL	2	4816	100	0.007
LM1	3	4973	100	0.006
LM2	10	37314	100	0.110
MC	2	2429	100	0.003
MR	3	18911	100	0.013
MCP	4	14062	100	0.027
MRP	2	2714	100	0.003
MGP	4	1896	100	0.007
NF2	10	16846	20	0.150
NF3	10	231860	100	0.410
PP	2	38428	100	0.140
PRD	4	1837	100	0.003
PWQ	9	10675	100	0.013
RG	10	44730	100	0.120
RB	10	96411	100	0.200
SAL	2	26038	99	0.061
SF1	2	1859	100	0.002
SBT	2	2107	85	0.003
SWF	10	56793	100	0.160
S5	4	5950	100	0.008
S7	4	5757	100	0.009
S10	4	5646	100	0.009
SINF	20	201941	100	1.290
WP	4	11033	100	0.012
tr		1,030,017	4181	4.278

Next, we study the effect of varying the initial sample size N of S. We fix $\delta = 0.1$, $\beta = 20$, $\gamma = 1$ and generate trial points using GM-I for this study. We run SAPS using three values of N, namely $3n$, $5n$ and $7n$. For each value of N, the SAPS algorithm was run 100 times on 12 representative problems. The results are presented in Table 5.10. The SAPS algorithm was successful in 1170, 1171 and 1166 runs out of 1200 runs for $N = 3n$, $5n$ and $7n$ respectively. SAPS accomplished these successes for total fe=590,095, 523,476 and 482,849 for $N = 3n$, $5n$ and $7n$ respectively. From the total results, we can see that the parameter $N = 7n$ is the best in terms of fe and cpu followed by $N = 5n$. Note also that SAPS exhibits similar results for $N = 3n$ and $N = 5n$ in terms of sr. We have decided to use the size $N = 5n$ because it has a slightly higher sr than $N = 7n$. We know that a single iteration-based MSL is invoked when all members of S are replaced. Therefore, intuitively speaking the larger the N is, the smaller the n_c will be. This has been clearly reflected in Table 5.10. For example, the n_c for the parameter $N = 3n$ is 66, while that of $N = 7n$ is 49.

Table 5.10: Results of SAPS for different sample size N, GM-I.

		$N = 3n$				$N = 5n$				$N = 7n$			
TP	n	fe	sr	cpu	n_c	fe	sr	cpu	n_c	fe	sr	cpu	n_c
DA	2	2924	94	0.003	6	2894	99	0.003	6	2759	98	0.003	5
EP	2	1648	98	0.003	8	2676	96	0.003	8	1537	95	0.003	6
GP	2	2634	99	0.003	3	2471	98	0.003	3	2492	98	0.003	2
EXP	10	26465	100	0.070	2	24137	100	0.070	2	23894	100	0.070	2
GW	10	52196	100	0.150	3	48315	100	0.140	2	47747	100	0.14	4
HV	3	12586	79	0.01	5	11435	78	0.01	5	10308	75	0.01	2
LM2	10	37091	100	0.110	2	35049	100	0.110	2	33878	100	0.110	2
NF3	10	237368	100	0.410	25	188142	100	0.350	19	162471	100	0.320	17
RG	10	44180	100	0.130	3	42866	100	0.120	2	41138	100	0.120	2
RB	10	89715	100	0.190	3	86965	100	0.190	3	79829	100	0.170	3
PP	10	37587	100	0.140	3	35364	100	0.140	2	35401	100	0.150	2
SWF	10	45701	100	0.120	3	43162	100	0.120	2	41395	100	0.110	2
tr		590,095	1170	1.343	66	523,476	1171	1.263	56	482,849	1166	1.213	49

Having determined the size N to use, we now investigate the effect of varying the parameter γ of the MSL algorithm. We fix $N = 5n$, $\delta = 0.1$, $\beta = 20.$ for this study. A series of runs of the SAPS algorithm was conducted using the values of γ, namely 1, 0.5 and 0.25. The results for $\gamma = 1$, $\gamma = 0.5$ and $\gamma = 0.25$ are presented in Table 5.11, Table 5.12 and Table 5.13 respectively.

The results for SAPS where MSL was implemented using $\gamma = 1$ is presented in Table 5.11. Here again, it is noteworthy that SAPS was not successful on the same 7 problems, namely Epistatic Michalewicz (EM), Modified Langerman (ML), Odd Square (OSP), Price's Transistor modelling (PTM) , Schaffer 2 (SF2), Shekel's foxholes (FX) and Storn's Tchebychev (ST9). Therefore, the results for these 7 problems are not represented in the total results. From the total results, we see that SAPS with the parameter $\gamma = 1$ was successful in 4167 runs out of 4300 runs with total fe=911,598. The total number of $n_{ps}(n_g)$ is 175(149). This indicates that out of 175 local searches performed 149 attained the global minimum.

Table 5.11: Results of SAPS using $N = 5n$ & $\gamma = 1$ using 43 problems, GM-I.

TP	n	fe	sr	cpu	$n_{ps}(n_g)$	n_c
ACK	10	31961	100	0.130	5(3)	3
AP	2	2600	100	0.003	3(3)	3
BL	2	2647	100	0.003	4(4)	2
B1	2	2743	100	0.003	2(2)	2
B2	2	2800	97	0.004	3(2)	3
BR	2	2338	100	0.003	3(3)	3
CB3	2	2381	100	0.003	2(2)	2
CB6	2	2409	100	0.003	3(3)	3
CM	4	6496	100	0.010	4(3)	2
DA	2	2894	99	0.003	6(5)	6
EP	2	1676	96	0.003	8(8)	8
EXP	10	24137	100	0.070	3(3)	2
GP	2	2471	98	0.003	3(2)	3
GW	10	48315	100	0.150	2(2)	2
GRP	3	5966	100	0.090	3(2)	2
H3	3	2879	100	0.020	4(4)	3
H6	6	16353	99	0.200	9(9)	8
HV	3	11435	78	0.010	5(2)	5
HSK	2	1819	100	0.003	4(4)	3
KL	2	4637	100	0.007	2(2)	1
LM1	3	4809	100	0.007	4(2)	2
LM2	10	35049	100	0.100	3(3)	2
MC	2	2276	100	0.003	3(3)	3
MR	3	14908	100	0.010	3(3)	2
MCP	4	10278	100	0.020	6(6)	3
MRP	2	2572	100	0.003	3(3)	2
MGP	4	1845	100	0.007	2(2)	2
NF2	10	14542	20	0.140	6(1)	3
NF3	10	188142	100	0.340	22(22)	19
PP	2	35364	100	0.130	3(3)	2
PRD	4	1753	100	0.003	2(2)	2
PWQ	9	9707	100	0.020	3(3)	3
RG	10	42866	100	0.130	9(3)	2
RB	10	86965	100	0.190	5(2)	3
SAL	2	25866	96	0.070	3(1)	3
SF1	2	1780	100	0.002	3(3)	3
SBT	2	2053	84	0.003	3(2)	2
SWF	10	43162	100	0.120	5(2)	2
S5	4	5435	100	0.008	6(3)	3
S7	4	5736	100	0.009	7(3)	3
S10	4	5507	100	0.010	6(3)	3
SIN	20	182865	100	1.350	7(3)	2
WP	4	9161	100	0.010	3(3)	3
tr		911,598	4167	3.408	175(149)	

Table 5.12 shows the results for SAPS where MSL is implemented with $\gamma = 0.5$. Note that SAPS was not successful on the same 7 problems as for $\gamma = 1$. Therefore, the results for these 7 problems are not represented in the total results. From the total results, we see that SAPS for $\gamma = 0.5$ was successful in 4156 runs out of 4300 runs with total fe=831,421. The total number of $n_{ps}(n_g)$ is 152(142) which indicates that 93% of total n_{ps} attained the global minimum.

Table 5.12: Results of SAPS using $N = 5n$ & $\gamma = 0.5$ using 43 problems, GM-I.

TP	n	fe	sr	cpu	$n_{ps}(n_g)$	n_c
ACK	10	28790	100	0.120	3(3)	3
AP	2	2612	100	0.003	3(3)	3
BL	2	2620	100	0.003	4(4)	2
B1	2	2743	100	0.003	2(2)	2
B2	2	2786	98	0.003	3(2)	2
BR	2	2258	100	0.003	2(2)	2
CB3	2	2381	100	0.003	2(2)	2
CB6	2	2405	100	0.003	3(3)	3
CM	4	6308	100	0.010	3(3)	2
DA	2	2843	98	0.003	6(4)	5
EP	2	1618	97	0.003	7(7)	6
EXP	10	24209	100	0.070	3(3)	2
GP	2	2447	97	0.003	3(2)	3
GW	10	48335	100	0.160	2(2)	2
GRP	3	8450	100	0.130	4(4)	3
H3	3	2808	100	0.020	4(4)	3
H6	6	15702	99	0.190	9(9)	8
HV	3	11700	77	0.010	5(2)	5
HSK	2	1749	100	0.003	3(3)	3
KL	2	4608	100	0.070	2(2)	1
LM1	3	4594	100	0.006	3(2)	2
LM2	10	34145	100	0.110	2(2)	2
MC	2	2271	100	0.003	3(3)	3
MR	3	15410	100	0.010	3(3)	2
MCP	4	8538	100	0.020	4(4)	3
MRP	2	2577	100	0.003	3(3)	2
MGP	2	1821	100	0.007	2(2)	2
NF2	4	14039	12	0.130	5(2)	4
NF3	10	173192	100	0.300	20(20)	19
PP	10	35219	100	0.140	3(3)	2
PRD	2	1751	100	0.003	2(2)	2
PWQ	4	9628	100	0.010	3(3)	3
RG	10	37953	100	0.110	7(3)	2
RB	10	79549	100	0.170	4(2)	3
SAL	10	25288	98	0.070	3(1)	3
SF1	2	1796	100	0.002	3(2)	2
SBT	2	1979	80	0.003	3(2)	2
SWF	10	38529	100	0.120	3(2)	2
S5	4	5202	100	0.008	5(3)	3
S7	4	5065	100	0.008	5(3)	3
S10	4	4973	100	0.009	5(3)	3
SIN	20	141043	100	1.100	5(3)	2
WP	4	9487	100	0.010	3(3)	3
tr		831,421	4156	3.165	152(142)	136

The results for the SAPS where MSL uses the parameter $\gamma = 0.25$ is presented in Table 5.13. SAPS still unable to solve any of the 7 problems mentioned before. Therefore, the results for these 7 problems are not represented in the total results. From the total results, we see that SAPS with $\gamma = 0.25$ was successful in 4154 runs out of 4300 runs with total fe=766,764. The total number of $n_{ps}(n_g)$ is 148(136) which shows that 92% of the total n_{ps} attained the global minimum.

Table 5.13: Results of SAPS using $N = 5n$ & $\gamma = 0.25$ using 43 problems, GM-I.

TP	n	fe	sr	cpu	$n_{ps}(n_g)$	n_c
ACK	10	28038	100	0.100	3(3)	3
AP	2	2590	100	0.003	3(3)	3
BL	2	2444	100	0.003	3(3)	2
B1	2	2743	100	0.003	2(2)	2
B2	2	2814	100	0.003	3(2)	3
BR	2	2331	100	0.003	3(3)	3
CB3	2	2381	100	0.003	2(2)	2
CB6	2	2400	100	0.003	3(3)	3
CM	4	6143	100	0.010	3(3)	2
DA	2	2751	99	0.003	6(4)	5
EP	2	1607	96	0.002	7(7)	7
EXP	10	23890	100	0.070	3(3)	2
GP	2	2474	97	0.003	3(2)	3
GW	10	48741	100	0.160	2(2)	2
GRP	3	4920	100	0.080	2(2)	2
H3	3	2762	100	0.020	4(4)	3
H6	6	14749	100	0.190	8(8)	7
HV	3	11700	77	0.010	5(2)	5
HSK	2	1759	100	0.003	3(3)	3
KL	2	4504	100	0.007	2(2)	1
LM1	3	4511	100	0.006	3(2)	2
LM2	10	34295	100	0.100	2(2)	2
MC	2	2199	100	0.002	3(3)	3
MR	3	14466	100	0.010	2(2)	2
MCP	4	7718	100	0.020	4(4)	3
MRP	2	2487	100	0.003	3(3)	2
MGP	2	1821	100	0.007	2(2)	2
NF2	4	13615	12	0.120	4(1)	3
NF3	10	167293	100	0.310	19(19)	19
PP	10	35381	100	0.140	3(3)	2
PRD	2	1731	100	0.003	2(2)	2
PWQ	4	9526	100	0.010	3(3)	3
RG	10	34013	100	0.100	5(3)	2
RB	10	69925	100	0.170	4(2)	3
SAL	10	25053	96	0.070	3(1)	3
SF1	2	1724	100	0.002	2(2)	2
SBT	2	1944	77	0.003	2(2)	2
SWF	10	33251	100	0.100	3(2)	2
S5	4	4784	100	0.008	5(3)	3
S7	4	4627	100	0.008	4(3)	3
S10	4	4539	100	0.008	4(3)	3
SIN	20	110820	100	0.870	3(3)	2
WP	4	9300	100	0.010	3(3)	3
tr		766,764	4154	2.758	148(136)	

In summary, SAPS was successful in 4167, 4157 and 4154 runs out of 4300 runs for $\gamma = 1$, 0.5 and 0.25 respectively. SAPS achieved these successes for total fe=911,598, 835,340 and 766,764 for $\gamma = 1$, 0.5 and 0.25 respectively. In conclusion, $\gamma = 1$ is the best in terms of sr and $\gamma = 0.25$ is the best in terms of fe. The total $n_{ps}(n_g)$ for $\gamma = 1$, 0.5 and 0.25 are 175(149), 152(142) and 148(136) respectively. It is clear that the choice of the parameter value γ has an effect on n_{ps}, i.e., n_{ps} decreases as γ decreases. The reason for this trend is because the number of n_{ps} depends on the number of points, γN, used in the single iteration-based MSL. Clearly, the smaller the number of points used in MSL, the smaller the n_{ps} value.

5.3 Overall Performance

We have so far presented the results of MSA, MSA-I and SAPS separately. In this section, we now compare the best results obtained by each of the above algorithms. Note that we use the results of those functions for which all the methods succeeded in finding the global minimum for fair comparison. In other words, we use only 41 problems that were solved by the three algorithms, namely MSA$_{\alpha=0.15}$, MSA-I and SAPS. We extract information for MSA$_{\alpha=0.15}$, MSA-I and SAPS using Table 5.4 and 5.13. These results are summarized in Table 5.14, where we have also presented the total cpu and the number of problems, P_{sol}, solved by an algorithm. Notice that the results presented in Table 5.14 are different from the corresponding total results in Table 5.4 and 5.13. This is because, we are have used the total results for 41 problems that were solved by the three algorithms.

Table 5.14: Comparison of the algorithms using total results.

Algorithm	fe	sr	cpu	P_{sol}
MSA$_{\alpha=0.15}$	496,291	3700	1.73	41
MSA-I	510,620	3933	1.77	43
SAPS	748,229	4082	2.56	43

We rank order the algorithms using the data from Table 5.14 and present in Table 5.15. In Table 5.15, it can be seen that there is no overall best performer in terms of three criteria, namely fe, sr and cpu. In terms of fe, MSA$_{\alpha=0.15}$ is the best performer. In terms of sr, SAPS is the best performer while in terms of cpu, MSA$_{\alpha=0.15}$ is the best performer.

Table 5.15: Rank order of algorithms.

Rank	1	2	3
fe	MSA$_{\alpha=0.15}$	MSA-I	SAPS
sr	SAPS	MSA-I	MSA$_{\alpha=0.15}$
cpu	MSA$_{\alpha=0.15}$	MSA-I	SAPS

5.4 Effect of temperature on step size parameter (Δ_t^{sa})

In this section, we discuss the effect of temperature on the step size parameter Δ_t^{sa} in equation (4.8). At initial stages of the algorithm, most of the trial points are accepted because the temperature is high. As a result, the ratio, ra, of equation (4.7) increases and consequently the step size Δ_t^{sa} increases, so as to explore the search space. On the other hand, as the temperature decreases, few points are accepted. Therefore, the ratio, ra, decreases and consequently Δ_t^{sa} decreases, so that the algorithm focuses more on exploitation. We have demonstrated this phenomena by running MSA once for each of the 4 different problems, namely Hosaki (HSK), Goldstein and Price (GP), Shekel 5 (S5) and Rosenbrock (RB). Results are presented in Figures 5.1, 5.2, 5.3 and 5.4. Each graph presents Δ_t^{sa} and temperature profiles. The x-axis of each graph represents number of Markov chains and y-axis represents Δ_t^{sa} on the left-hand side and temperature on the right-hand side. The figures vary from problem to problem. For example in Figure 5.1, Δ_t^{sa} increases up to its highest peak when $n_{markov} = 30$ with $T_t \approx 0.1$ before it starts to decrease. On the other hand, in Figure 5.3, Δ_t^{sa} increases up to its maximum when $n_{markov} = 18$ with $T_t \approx 1.5$ before it starts to decline.

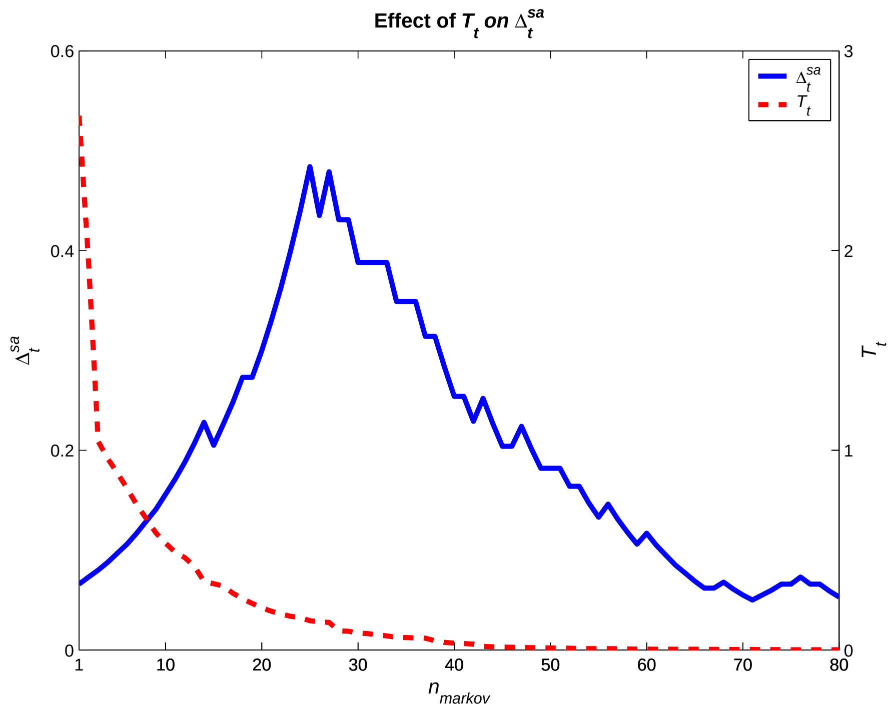

Figure 5.1: Effect of T_t on Δ_t^{sa} for HSK problem.

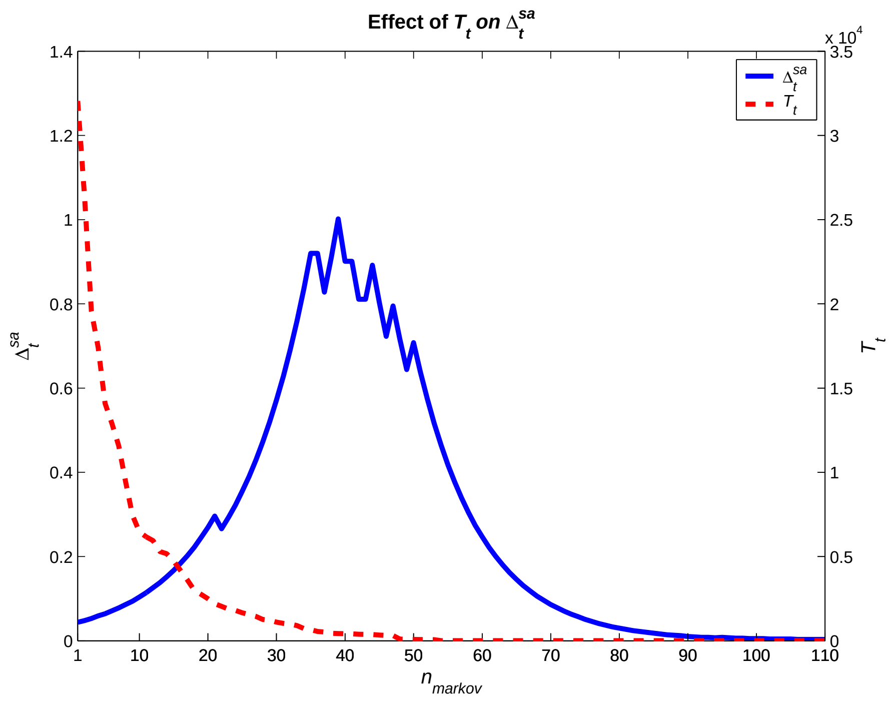

Figure 5.2: Effect of T_t on Δ_t^{sa} for GP problem.

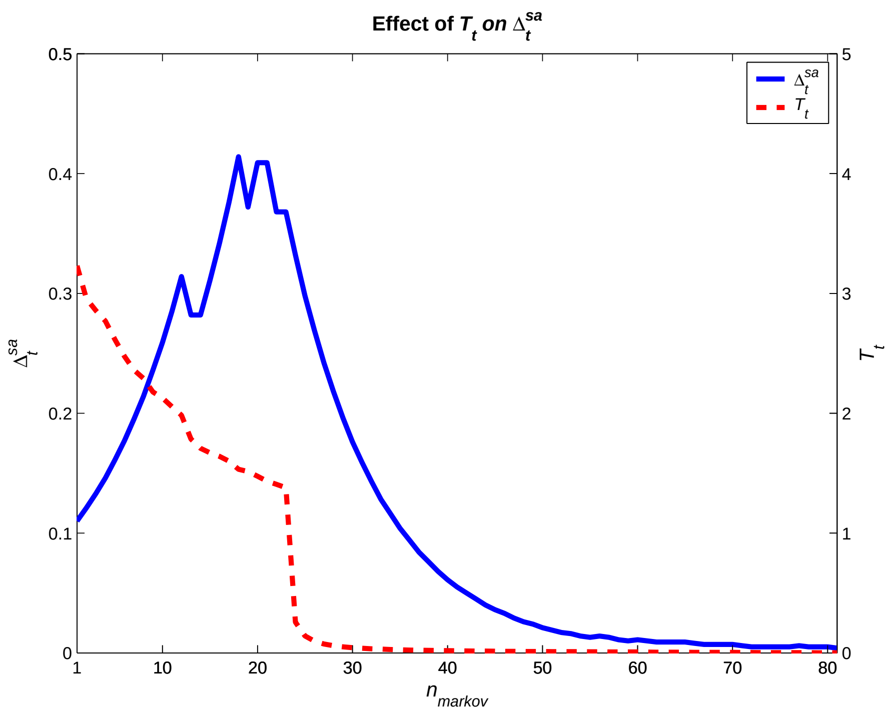

Figure 5.3: Effect of T_t on Δ_t^{sa} for S5 problem.

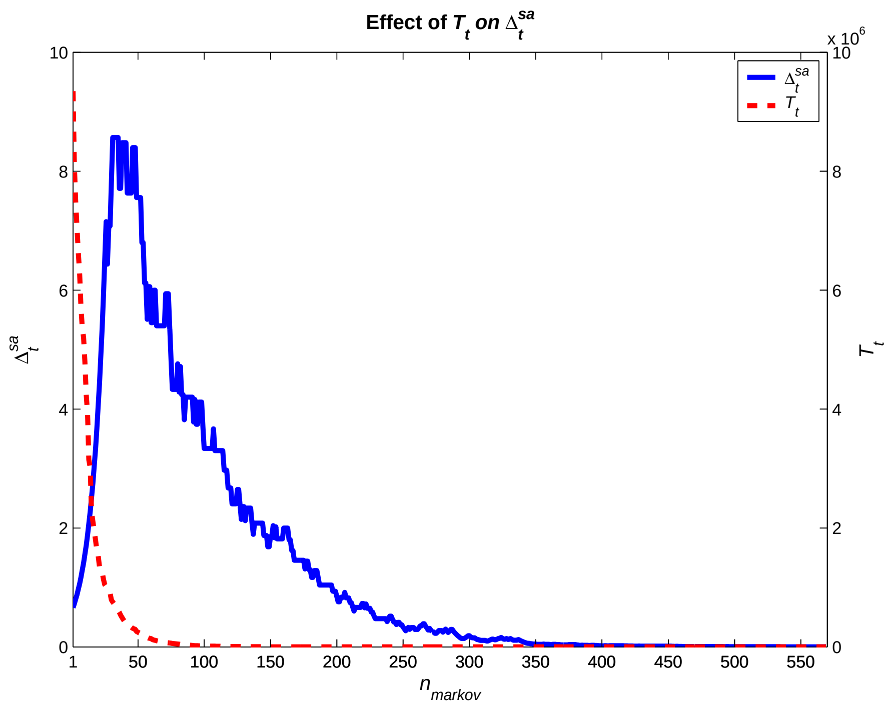

Figure 5.4: Effect of T_t on Δ_t^{sa} for RB problem.

5.5 A study of the critical distance Δ_t^c in the single iteration-based MSL

In this section, we explain how the critical distance Δ_t^c of equation (4.9) changes in the single iteration-based MSL. The distance Δ_t^c is used to control the number of local search made. The value Δ_t^c of equation (4.9) is given by

$$\Delta_t^c = \max\{\Delta_t^{sa}, \beta\Delta_0^{sa}\}, \tag{5.4}$$

where β is equal to 20 in regard to the results of Table 5.7. For this study we used two functions and ran SAPS using the best parameter values found. Results are presented in Figures 5.5 and 5.6. The x-axis represents the number of Markov chains; y-axis represents Δ_t^{sa} on the left-hand side and Δ_t^c on the right-hand side.

An important feature of both figures is that they use $\Delta_t^c = \beta\Delta_0^{sa}$ for a sizeable number of Markov chains before using $\Delta_t^c = \Delta_t^{sa}$. Towards the end of a run the SAPS algorithm again uses $\Delta_t^c = \beta\Delta_0^{sa}$. For example, in Figure 5.5, Δ_t^c takes the value $\beta\Delta_0^{sa} = 3$ from the 1st to the 22nd Markov chain; it takes the values of Δ_t^{sa} from the 23rd to 45th Markov chain. Finally, Δ_t^c takes the value of $\beta\Delta_0^{sa} = 3$.

The Δ_t^c used by the single iteration-based MSL has been indicated with $*$ in each figure. This feature of the SAPS indicates that more local searches are performed towards the beginning and towards the end of a run.

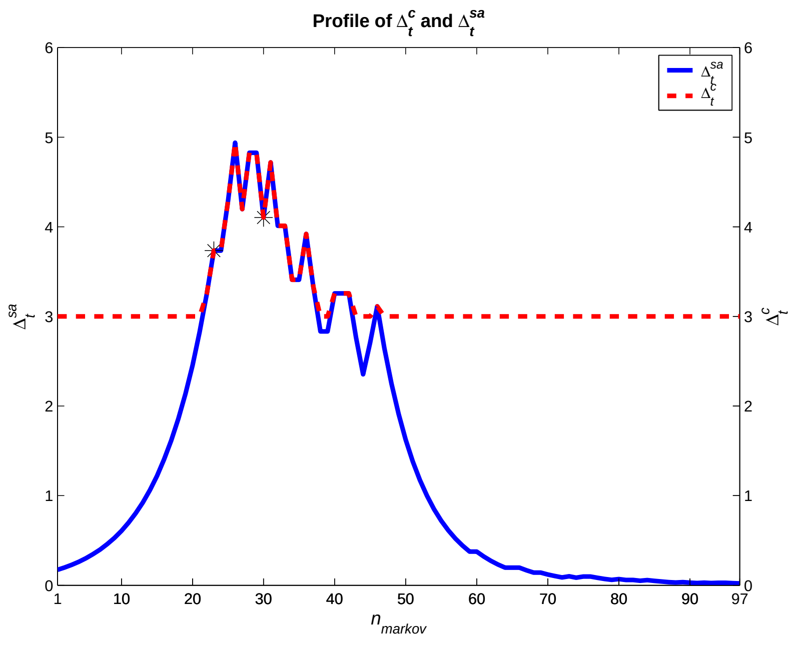

Figure 5.5: Profile of Δ_t^c and Δ_t^{sa} for BP problem.

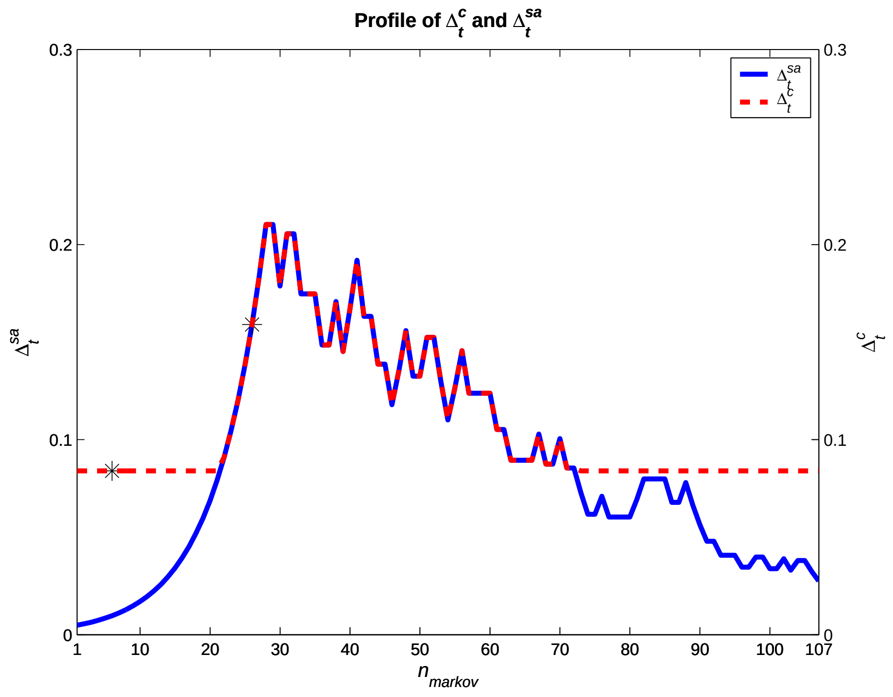

Figure 5.6: Profile of Δ_t^c and Δ_t^{sa} for KL problem.

5.6 Summary

In this chapter, we have presented the numerical results for PS, MPS, and the two hybrids, namely MSA and SAPS. We have applied the algorithm to different test problems and compared the different hybrids developed. Results have shown that the new algorithms are efficient and robust.

Chapter 6

Conclusion and future research

The objective of this dissertation is devoted to design a pattern search based global optimization. To achieve this objective, we have proposed two global optimization based on PS. They are modified simulated annealing (MSA) and simulated annealing driven pattern search (SAPS).

We have carried out an extensive numerical testing of the new algorithms using a large set of test problems. We have first empirically found the optimal values of the parameters of both the algorithms. Sensitivity analysis of some parameters is also performed.

We have conducted numerous runs of each algorithm using more than one value of some parameters. Results obtained by the algorithms for all runs were very satisfactory. Both MSA and SAPS have proved to be efficient and reliable in terms of the number of function evaluations, cpu times and locating the global minimum value.

We have also developed a modified pattern search (MPS) for local minimization. MPS have improved the pattern search method (MPS) considerably in terms of efficiency and reliability.

The approach we adopted in designing the PS based global optimization is new and therefore there will be further scope to develop more efficient and reliable global optimization algorithm for both unconstrained and constrained problems.

We have used a single iteration based MSL algorithm within the framework of simulated annealing. Hence the stopping condition used was that of the simulated annealing. An important aspect that requires further research is to theoretically study the critical distance of the MSL which will also form part of our future work. One can also study the reasons for failure of some nonseparable functions.

Appendix A

The multi level single linkage algorithm

MSL [42] is a modification of the multistart (MS) [41] method which overcomes some of the drawbacks of MS. It consists of two phases in an iteration: a global phase and a local phase. In the global phase, the function is evaluated at N random points. In the local phase, $\gamma v N$, sample points are scrutinized to perform local searches in order to yield a candidate global minimizer, where v is the iteration and $0 < \gamma < 1$. The local search procedure will be applied to a subset of $\gamma v N$ points. We denote the critical distance by r_v.The goal for the MSL algorithm is to find all local minima. We now present the MSL algorithm at the v^{th} iteration in full details.

The MSL algorithm.

1. Sample N points from the search region Ω and calculate the function values $f(x_i^\rho)$, $i = 1, \cdots, N$, of these points. Add these points to the previously drawn $(v-1)N$ points in all earlier iterations. Discard a percentage of worse points.

2. Order the sample points such that $f(x_i^\rho) \leq f(x_{i+1}^\rho)$, $1 \leq i \leq R$, R being the number of remaining points, i.e., $R = \gamma v N$. Start a local search from each new point x_i^ρ except if there is another sample point or previous detected local minimum within the critical distance r_v of x_i^ρ. Add new local minimum point found during the local search to a set of local minima found so far.

3. If the stopping condition is satisfied then stop else go to step 1.

Remark:

1. The distance r_v (computed for every v^{th} iteration) is computed by

$$r_v = \pi^{-\frac{1}{2}} \left[\Gamma(1 + \frac{n}{2}) \mu(\Omega) \sigma \frac{\log(vN)}{vN} \right]^{\frac{1}{n}}, \tag{A.1}$$

where $\mu(\Omega)$ is the Lebesgue measure of the region Ω, $\sigma = 4$, and $\Gamma(n)$ is the gamma function.

Appendix B

A collection of benchmark global optimization test problems

In this appendix, we present 50 well-known benchmark problems which are often used by global optimization researchers. These problems represent various characteristic terrain found in real-world problems,.e.g., unimodal or multimodal, with or without plateaus and ridges, and high or low dimensional. Some of these test problems (TP) can be found in textbooks, in individual research articles, or at different web sites. A collection of these 50 problems is found in Ali et. al. [5]. Please note that in several cases the global minimizer x^* and corresponding global minimum $f(x^*)$ are known only as a numerical approximation.

1. Ackley's Problem (ACK)

$$\min_x f(x) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} {x_i}^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e \tag{B.1}$$

$$\text{subject to} \quad -30 \le x_i \le 30, i \in \{1, 2, \dots, n\}. \tag{B.2}$$

The number of local minima is not known. The global minimum is located at the origin, i.e, with $f(x^*) = 0$. Tests were performed for $n = 10$.

2. Aluffi-Pentini's Problem (AP)

$$\min_x f(x) = 0.25{x_1}^4 - 0.5{x_1}^2 + 0.1x_1 + 0.5{x_2}^2 \tag{B.3}$$

$$\text{subject to} \quad -10 \le x_1, x_2 \le 10. \tag{B.4}$$

The function has two local minima, one of them is global with $f(x^*) \approx -0.3523$ located at $(-1.0465, 0)$.

3. Becker and Lago Problem (BL)

$$\min_x f(x) = (|x_1| - 5)^2 + (|x_2| - 5)^2 \tag{B.5}$$

$$\text{subject to} \quad -10 \leq x_1, x_2 \leq 10. \tag{B.6}$$

The function has four minima located at $x^* = (\pm 5, \pm 5)$, all with $f(x^*) = 0$.

4. Bohachevsky 1 Problem (B1)

$$\min_x f(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2) + 0.7 \tag{B.7}$$

$$\text{subject to} \quad -50 \leq x_1, x_2 \leq 50. \tag{B.8}$$

The number of local minima is unknown but the global minimizer is located at $x^* = (0, 0)$ with $f(x^*) = 0$.

5. Bohachevsky 2 Problem (B2)

$$\min_x f(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1)\cos(4\pi x_2) + 0.3 \tag{B.9}$$

$$\text{subject to} \quad -50 \leq x_1, x_2 \leq 50. \tag{B.10}$$

The number of local minima is unknown but the global minimizer is located at $x^* = (0, 0)$ with $f(x^*) = 0$.

6. Branin Problem (BR)

$$\min_x f(x) = a(x_2 - b{x_1}^2 + cx_1 - d)^2 + g(1 - h)\cos(x_1) + g\,, \tag{B.11}$$

$$\text{subject to} \quad -5 \leq x_1 \leq 10, 0 \leq x_2 \leq 15, \tag{B.12}$$

where $a = 1$, $b = 5.1/(4\pi^2)$, $c = 5/\pi$, $d = 6$, $g = 10$, $h = 1/(8\pi)$. There are three minima, all global, in this region. The minimizers are

$$x^* \approx (-\pi, 12.275), (\pi, 2.275), (3\pi, 2.475)$$

with $f(x^*) = 5/(4\pi)$.

7. Camel Back–3 Three Hump Problem (CB3)

$$\min_x f(x) = 2x_1^2 - 1.05x_1^4 + \tfrac{1}{6}x_1^6 + x_1x_2 + x_2^2 \tag{B.13}$$

$$\text{subject to} \quad -5 \leq x_1, x_2 \leq 5. \tag{B.14}$$

The function has three local minima, one of them is global located at $x^* = (0, 0)$ with $f(x^*) = 0$.

8. Camel Back–6 Six Hump Problem (CB6)

$$\min_x f(x) = 4x_1^2 - 2.1x_1^4 + \tfrac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4 \tag{B.15}$$

$$\text{subject to} \quad -5 \le x_1, x_2 \le 5. \tag{B.16}$$

This function is symmetric about the origin and has three conjugate pairs of local minima with values $f \approx -1.0316,\ -0.2154,\ 2.1042$. The function has two global minima at $x^* \approx (0.089842, -0.712656)$ and $(-0.089842, 0.712656)$ with $f(x^*) \approx -1.0316$.

9. Cosine Mixture Problem (CM)

$$\max_x f(x) = 0.1\sum_{i=1}^{n}\cos(5\pi x_i) - \sum_{i=1}^{n} x_i^2 \tag{B.17}$$

$$\text{subject to} \quad -1 \le x_i \le 1, i \in \{1, 2, \ldots, n\}. \tag{B.18}$$

The global maxima are located at the origin with the function values 0.20 and 0.40 for $n = 2$ and $n = 4$, respectively.

10. Dekkers and Aarts Problem (DA)

$$\min_x f(x) = 10^5x_1^2 + x_2^2 - (x_1^2 + x_2^2)^2 + 10^{-5}(x_1^2 + x_2^2)^4 \tag{B.19}$$

$$\text{subject to} \quad -20 \le x_1, x_2 \le 20. \tag{B.20}$$

The origin is a local minimizer, but there are two global minimizers located at $x^* = (0, 15)$ and $(0, -15)$ with $f(x^*) = -24776.518$.

11. Easom Problem (EP)

$$\min_x f(x) = -\cos(x_1)\cos(x_2)\exp\left(-(x_1 - \pi)^2 - (x_2 - \pi)^2\right) \tag{B.21}$$

$$\text{subject to} \quad -10 \le x_1, x_2 \le 10. \tag{B.22}$$

The minimum value is located at (π, π) with $f(x^*) = -1$. The function value rapidly approaches zero, when away from (π, π).

12. Epistatic Michalewicz Problem (EM)

$$\min_x f(x) = -\sum_{i=1}^{n}\sin(y_i)\left(\sin\left(\frac{iy_i^2}{\pi}\right)\right)^{2m}, \tag{B.23}$$

$$\text{subject to} \quad 0 \le x_i \le \pi, i \in \{1, 2, \ldots, n\}, \tag{B.24}$$

where

$$y_i = \begin{cases} x_i \cos(\theta) - x_{i+1}\sin(\theta), & i = 1,3,5,\ldots,< n \\ x_i \sin(\theta) + x_{i+1}\cos(\theta), & i = 2,4,6,\ldots,< n \\ x_i, & i = n \end{cases}, \tag{B.25}$$

and $\theta = \frac{\pi}{6}$, $m = 10$.

The number of local minima is not known but the global minimizer is presented in Table B.1.

Table B.1: Epistatic Michalewicz's global optimizers.

n	$f(x^*)$	x^*
5	-4.687658	(2.693,0.259,2.074,1.023,1.720)
10	-9.660152	(2.693,0.259,2.074,1.023,2.275,0.500,2.138,0.794,2.219,0.533)

13. Exponential Problem (EXP)

$$\max_x f(x) = \exp\left(-0.5\sum_{i=1}^{n} {x_i}^2\right) \tag{B.26}$$

$$\text{subject to} \quad -1 \le x_i \le 1, i \in \{1,2,\ldots,n\}. \tag{B.27}$$

The optimal value $f(x^*) = 1$ is located at the origin. Our tests were performed with $n = 10, 20$.

14. Goldstein and Price (GP)

$$\begin{aligned} \min_x f(x) = & \left[1 + (x_1 + x_2 + 1)^2\left(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2\right)\right] \\ & \times\left[30 + (2x_1 - 3x_2)^2\left(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2\right)\right] \end{aligned} \tag{B.28}$$

$$\text{subject to} \quad -2 \le x_1, x_2 \le 2. \tag{B.29}$$

There are four local minima and the global minimum is located at $x^* = (0, -1)$, with $f(x^*) = 3$.

15. Griewank Problem (GW)

$$\min_x f(x) = 1 + \tfrac{1}{4000}\sum_{i=1}^{n} {x_i}^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) \tag{B.30}$$

$$\text{subject to} \quad -600 \le x_i \le 600, i \in \{1,2,\ldots,n\}. \tag{B.31}$$

The function has a global minimum located at $x^* = (0,0,\ldots,0)$ with $f(x^*) = 0$. Number of local minima for arbitrary n is unknown, but in the two dimensional case there are some 500 local minima. Tests were performed for $n = 10$. Note that this function becomes simpler and smoother in the numeric space, and easy to solve, as the dimensionality of the search space is increased [34].

16. Gulf Research Problem (GRP)

$$\min_x f(x) = \sum_{i=1}^{99} \left[\exp\left(-\frac{(u_i - x_2)^{x_3}}{x_1} \right) - 0.01 \times i \right]^2, \tag{B.32}$$

$$\text{subject to} \quad 0.1 \le x_1 \le 100, 0 \le x_2 \le 25.6, \text{ and } 0 \le x_3 \le 5, \tag{B.33}$$

where $u_i = 25 + [-50 \ln(0.01 \times i)]^{1/1.5}$. This problem has a global minimizer at $(50, 25, 1.5)$ with $f(x^*) = 0$.

17. Hartman 3 Problem (H3)

$$\min_x f(x) = -\sum_{i=1}^{4} c_i \exp\left[-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2 \right] \tag{B.34}$$

$$\text{subject to} \quad 0 \le x_j \le 1, j \in \{1, 2, 3\} \tag{B.35}$$

with constants a_{ij}, p_{ij} and c_i given in Table B.2. There are four local minima, $x^{loc} \approx (p_{i1}, p_{i2}, p_{i3})$ with $f(x^{loc}) \approx -c_i$. The global minimum is located at

$$x^* \approx (0.114614, 0.555649, 0.852547)$$

with $f(x^*) \approx -3.862782$.

Table B.2: Data for Hartman 3 problem.

i	c_i	a_{ij}			p_{ij}		
		$j = 1$	2	3	$j = 1$	2	3
1	1	3	10	30	0.3689	0.117	0.2673
2	1.2	0.1	10	35	0.4699	0.4387	0.747
3	3	3	10	30	0.1091	0.8732	0.5547
4	3.2	0.1	10	35	0.03815	0.5743	0.8828

18. Hartman 6 Problem (H6)

$$\min_x f(x) = -\sum_{i=1}^{4} c_i \exp\left[-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2 \right] \tag{B.36}$$

$$\text{subject to} \quad -0 \le x_j \le 1, j \in \{1, \ldots, 6\}, \tag{B.37}$$

with constants a_{ij} and c_i given in Table B.3 and constants p_{ij} in Table B.4. There are four local minima, $x^{loc} \approx (p_{i1}, \ldots, p_{i6})$ with $f(x^{loc}) \approx -c_i$. The global minimum is located at $x^* \approx (0.201690, 0.150011, 0.476874, 0.275332, 0.311652, 0.657301)$ with $f(x^*) \approx -3.322368$.

Table B.3: Data for Hartman 6 problem.

i	c_i	a_{ij}					
		$j=1$	2	3	4	5	6
1	1	10	3	17	3.5	1.7	8
2	1.2	0.05	10	17	0.1	8	14
3	3	3	3.5	1.7	10	17	8
4	3.2	17	8	0.05	10	0.1	14

Table B.4: Data for Hartman 6 problem.

i	p_{ij}					
	$j=1$	2	3	4	5	6
1	0.1312	0.1696	0.5569	0.0124	0.8283	0.5886
2	0.2329	0.4135	0.8307	0.3736	0.1004	0.9991
3	0.2348	0.1451	0.3522	0.2883	0.3047	0.665
4	0.4047	0.8828	0.8732	0.5743	0.1091	0.0381

19. Helical Valley Problem (HV)

$$\min_x f(x) = 100\left[(x_2-10\theta)^2+(\sqrt{(x_1^2+x_2^2)}-1)^2\right]+x_3^2 \tag{B.38}$$

$$\text{subject to} \quad -10 \le x_1, x_2, x_3 \le 10 \tag{B.39}$$

where

$$\theta = \begin{cases} \frac{1}{2\pi}\tan^{-1}\frac{x_2}{x_1}, & \text{if } x_1 \ge 0 \\ \frac{1}{2\pi}\tan^{-1}\frac{x_2}{x_1}+\frac{1}{2}, & \text{if } x_1 < 0 \end{cases} \tag{B.40}$$

This is a steep-sided valley which follows a helical path. The minimum is located at $x^* = (1,0,0)$ with $f(x^*) = 0$.

20. Hosaki Problem (HSK)

$$\min_x f(x_1,x_2) = \left(1-8x_1+7x_1^2-\tfrac{7}{3}x_1^3+\tfrac{1}{4}x_1^4\right)x_2^2\exp(-x_2) \tag{B.41}$$

$$\text{subject to} \quad 0 \le x_1 \le 5\,, 0 \le x_2 \le 6. \tag{B.42}$$

There are two minima of which the global minimum is $f(x^*) \approx -2.3458$ with $x^* = (4,2)$.

21. Kowalik Problem (KL)

$$\min_x f(x) = \sum_{i=1}^{11}\left(a_i - \frac{x_1(1+x_2b_i)}{(1+x_3b_i+x_4b_i^2)}\right)^2 \tag{B.43}$$

$$\text{subject to} \quad 0 \le x_i \le 0.42, i \in \{1,2,3,4\}. \tag{B.44}$$

The values for a_i and b_i are given in Table B.5:

Table B.5: Data for Kowalik problem.

i	1	2	3	4	5	6	7	8	9	10	11
a_i	0.1957	0.1947	0.1735	0.16	0.0844	0.0627	0.0456	0.0342	0.0323	0.0235	0.0246
b_i	0.25	0.50	1.0	2.0	4.0	6.0	8.0	10.0	12.0	14.0	16.0

This is a least squares problem with a global optimal value $f(x^*) \approx 3.0748 \times 10^{-4}$ located at $x^* \approx (0.192, 0.190, 0.123, 0.135)$.

22. Levy and Montalvo 1 Problem (LM1)

$$\min_x f(x) = \frac{\pi}{n}\left(10\sin^2(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2\left[1 + 10\sin^2(\pi y_{i+1})\right]\right) + \frac{\pi}{n}(y_n - 1)^2 \tag{B.45}$$

$$\text{subject to} \quad -10 \le x_i \le 10, i \in \{1, 2, \dots, n\} \tag{B.46}$$

where $y_i = 1 + \frac{1}{4}(x_i + 1)$. There are approximately 5^n local minima and the global minimum is known to be $f(x^*) = 0$ with $x^* = (-1, -1, \dots, -1)$. Our tests were performed with $n = 3$.

23. Levy and Montalvo 2 Problem (LM2)

$$\min_x f(x) = 0.1(\sin^2(3\pi x_1) + \sum_{i=1}^{n-1}(x_i - 1)^2[1 + \sin^2(3\pi x_{i+1}] + (x_n - 1)^2[1 + \sin^2(2\pi x_n)]) \tag{B.47}$$

$$\text{subject to} \quad -5 \le x_i \le 5, i \in \{1, 2, \dots, n\}. \tag{B.48}$$

There are approximately 15^n minima and the global minimizer is known to be $x^* = (1, 1, \dots, 1)$ with $f(x^*) = 0$. Our tests were performed with $n = 10$.

24. McCormick Problem (MC)

$$\min_x f(x) = \sin(x_1 + x_2) + (x_1 - x_2)^2 - (3/2)x_1 + (5/2)x_2 + 1 \tag{B.49}$$

$$\text{subject to} \quad -1.5 \le x_1 \le 4, -3 \le x_2 \le 3. \tag{B.50}$$

This problem has a local minimum at $(2.59, 1.59)$ and a global minimum at $x^* \approx= (-0.547, -1.547)$ with $f(x^*) \approx -1.9133$.

25. Meyer and Roth Problem (MR)

$$\min_x f(x) = \sum_{i=1}^{5}\left(\frac{x_1 x_3 t_i}{(1 + x_1 t_i + x_2 v_i)} - y_i\right)^2 \tag{B.51}$$

$$\text{subject to} \quad -20 \le x_i \le 20, i \in \{1, 2, 3\}. \tag{B.52}$$

This is a least squares problem with minimum value $f(x^*) \approx 0.4{\times}10^{-4}$ located at $x^* \approx (3.13, 15.16, 0.78)$. Table B.6 lists the parameter values of this problem.

Table B.6: Data for Meyer & Roth problem.

i	t_i	v_i	y_i
1	1.0	1.0	0.126
2	2.0	1.0	0.219
3	1.0	2.0	0.076
4	2.0	2.0	0.126
5	0.1	0.0	0.186

26. Miele and Cantrell Problem (MCP)

$$\min_x f(x) = (\exp(x_1) - x_2)^4 + 100(x_2 - x_3)^6 + (\tan(x_3 - x_4))^4 + {x_1}^8 \tag{B.53}$$

$$\text{subject to} \quad -1 \le x_i \le 1, i \in \{1, 2, 3, 4\}. \tag{B.54}$$

The number of local minima is unknown but the global minimizer is located at $x^* = (0, 1, 1, 1)$ with $f(x^*) = 0$.

27. Modified Langerman Problem (ML)

$$\min_x f(x) = -\sum_{j=1}^{5} c_j \cos(\pi d_j) \exp(-d_j/\pi) , \tag{B.55}$$

$$\text{subject to} \quad 0 \le x_i \le 10, i \in \{1, 2, \ldots, n\}, \tag{B.56}$$

where $d_j = \sum_{i=1}^{n} (x_i - a_{ji})^2$. The test used $n = 10$. The constants c_j and a_{ji} are given in Table B.7.

Table B.7: Data for modified Langerman problem.

j	c_j	a_{ji}									
		$i = 1$	2	3	4	5	6	7	8	9	10
1	0.806	9.681	0.667	4.783	9.095	3.517	9.325	6.544	0.211	5.122	2.020
2	0.517	9.400	2.041	3.788	7.931	2.882	2.672	3.568	1.284	7.033	7.374
3	0.100	8.025	9.152	5.114	7.621	4.564	4.711	2.996	6.126	0.734	4.982
4	0.908	2.196	0.415	5.649	6.979	9.510	9.166	6.304	6.054	9.377	1.426
5	0.965	8.074	8.777	3.467	1.867	6.708	6.349	4.534	0.276	7.633	1.567

The number of local minima is not known, but the global minima are shown in Table B.8.

Table B.8: Global optimizers for modified Langerman problem.

n	$f(x^*)$	x^*
5	-0.965	(8.074, 8.777, 3.467, 1.867, 6.708)
10	-0.965	(8.074, 8.777, 3.467, 1.867, 6.708, 6.349, 4.534, 0.276, 7.633, 1.567)

28. Modified Rosenbrock Problem (MRP)

$$\min_x f(x) = \quad 100(x_2 - {x_1}^2)^2 + \left[6.4(x_2 - 0.5)^2 - x_1 - 0.6\right]^2 \tag{B.57}$$

$$\text{subject to} \qquad -5 \le x_1, x_2 \le 5. \tag{B.58}$$

This function has two global minima each with $f(x^*) = 0$ (corresponding to the intersection of two parabolas) and a local minimum (where the parabolas approach without intersection). The global minima are located at $x^* \approx (0.3412, 0.1164), (1, 1)$.

29. Multi-Gaussian Problem (MGP)

$$\max_x f(x) = \quad \sum_{i=1}^{5} a_i \exp\left(-((x_1 - b_i)^2 + (x_2 - c_i)^2)/{d_i}^2\right) \tag{B.59}$$

$$\text{subject to} \qquad -2 \le x_1, x_2 \le 2. \tag{B.60}$$

The function has one global maximum at $x^* \approx (-0.01356, -0.01356)$ with $f(x^*) \approx 1.29695$. There are also 4 other local maxima and a saddle point. Values for the parameters a_i, b_i, c_i, and d_i are given in Table B.9.

Table B.9: Data for Multi-Gaussian problem.

i	a_i	b_i	c_i	d_i
1	0.5	0.0	0.0	0.1
2	1.2	1.0	0.0	0.5
3	1.0	0.0	-0.5	0.5
4	1.0	-0.5	0.0	0.5
5	1.2	0.0	1.0	0.5

30. Neumaier 2 Problem (NF2)

$$\min_x f(x) = \quad \sum_{k=1}^{n} \left(b_k - \sum_{i=1}^{n} {x_i}^k\right)^2 \tag{B.61}$$

$$\text{subject to} \quad 0 \le x_i \le n, i \in \{1, 2, \ldots, n\}. \tag{B.62}$$

We consider a case when $n = 4$ and $b = (8, 18, 44, 114)$. The global minimum is $f(1, 2, 2, 3) = 0$.

31. Neumaier 3 Problem (NF3)

$$\min_x f(x) = \sum_{i=1}^{n}(x_i - 1)^2 - \sum_{i=2}^{n} x_i x_{i-1} \tag{B.63}$$

$$\text{subject to} \quad -n^2 \le x_i \le n^2, i \in \{1, 2, \dots, n\}. \tag{B.64}$$

The case considered here is $n = 10$. The number of local minima is not known, but the global minima can be expressed as:

$$f(x^*) = -\frac{n(n+4)(n-1)}{6}, \quad x_i^* = i(n+1-i).$$

The global minima for some values of n are presented below.

Table B.10: Global minima for Neumaier 3 problem.

n	10	15	20	25	30
$f(x^*)$	-210	-665	-1520	-2900	-4930

32. Odd Square Problem (OSP)

$$\min_x f(x) = -\left(1.0 + 0.2d/(D+0.01)\right)\cos(D\pi)\, e^{-D/2\pi} \tag{B.65}$$

$$\text{subject to} \quad -15 \le x_i \le 15, i \in \{1, 2, \dots, 20\} \tag{B.66}$$

where

$$d = \sqrt{\sum_{i=1}^{n}(x_i - b_i)^2}, \quad D = \sqrt{n}\left(\max|x_i - b_i|\right),$$

and

$$\underline{b} = (1, 1.3, 0.8, -0.4, -1.3, 1.6, -2, -6, 0.5, 1.4), b_{10+i} = b_i, i = 1, 2, \cdots, 10$$

The number of local minima for a given n is not known but the global minimum is known to be $f(x^*) \approx -1.143833$, $x^* \cong \vec{b} \quad (\text{many solutions near } \underline{b})$. We used $n = 10$ in our experiment.

33. Paviani Problem (PP)

$$\min_x f(x) = \sum_{i=1}^{10}\left[(\ln(x_i - 2))^2 + (\ln(10 - x_i))^2\right] - \left(\prod_{i=1}^{10} x_i\right)^{0.2} \tag{B.67}$$

$$\text{subject to} \quad 2 \le x_i \le 10, i \in \{1, 2, \dots, 10\}. \tag{B.68}$$

This function has a global minimizer at $x_i^* \approx 9.351$ for all i, with $f(x^*) \approx -45.778$.

34. Periodic Problem (PRD)

$$\min_x f(x) = \quad 1 + \sin^2 x_1 + \sin^2 x_2 - 0.1 \exp(-{x_1}^2 - {x_2}^2) \tag{B.69}$$

$$\text{subject to} \quad -10 \leq x_1, x_2 \leq 10. \tag{B.70}$$

There are 49 local minima all with value 1 and global minimum located at $x^* = (0, 0)$ with $f(x^*) = 0.9$.

35. Powell's Quadratic Problem (PWQ)

$$\min_x f(x) = \quad (x_1 + 10x_1)^2 + 5\,(x_3 - x_4)^2 + (x_2 - 2x_3)^4 + 10\,(x_1 - x_4)^4 \tag{B.71}$$

$$\text{subject to} \quad -10 \leq x_i \leq 10, i \in \{1, 2, 3, 4\}. \tag{B.72}$$

This is a unimodal function with $f(x^*) = 0$, $x^* = (0, 0, 0, 0)$. The minimizer is difficult to obtain with accuracy as the Hessian matrix at the optimum is singular.

36. Price's Transistor Modelling Problem (PTM)

$$\min_x f(x) = \quad \gamma^2 + \sum_{k=1}^{4} ({\alpha_k}^2 + {\beta_k}^2) \tag{B.73}$$

$$\text{subject to} \quad -10 \leq x_i \leq 10, i \in \{1, 2, \ldots, 9\}, \tag{B.74}$$

where

$$\begin{aligned}
\alpha_k =& (1 - x_1 x_2) x_3 \{\exp[x_5 (g_{1k} - g_{3k} x_7 \times 10^{-3} - g_{5k} x_8 \times 10^{-3})] - 1\} - g_{5k} + g_{4k} x_2, \\
\beta_k =& (1 - x_1 x_2) x_4 \{\exp[x_6 (g_{1k} - g_{2k} - g_{3k} x_7 \times 10^{-3} + g_{4k} x_9 \times 10^{-3})] - 1\} \\
& - g_{5k} x_1 + g_{4k}, \\
\gamma =& x_1 x_3 - x_2 x_4.
\end{aligned}$$

The values of g_{ik} are given in Table B.11.

Table B.11: Data for Price's transistor modelling problem.

i	g_{ik}			
	$k = 1$	2	3	4
1	0.485	0.752	0.869	0.982
2	0.369	1.254	0.703	1.455
3	5.2095	10.0677	22.9274	20.2153
4	23.3037	101.779	111.461	191.267
5	28.5132	111.8467	134.3884	211.4823

The global minimum occurs very close to $(0.9, 0.45, 1, 2, 8, 8, 5, 1, 2)$ with $f(x^*) = 0$. The number of local minima is unknown.

37. Rastrigin Problem (RG)

$$\min_x f(x) = 10n + \sum_{i=1}^{n} \left[x_i^2 - 10\cos\left(2\pi x_i\right)\right] \tag{B.75}$$

$$\text{subject to} \quad -5.12 \le x_i \le 5.12, i \in \{1, 2, \ldots, n\}. \tag{B.76}$$

The total number of minima for this function is not exactly known but the global minimizer is located at $x^* = (0, 0, \ldots, 0)$ with $f(x^*) = 0$. For $n = 2$, there are about 50 local minimizers arranged in a lattice like configuration. Our tests were performed with $n = 10$.

38. Rosenbrock Problem (RB)

$$\min_x f(x) = \sum_{i=1}^{n-1} \left[100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2\right] \tag{B.77}$$

$$\text{subject to} \quad -30 \le x_i \le 30, i \in \{1, 2, \ldots, n\}. \tag{B.78}$$

Our tests were performed with $n = 10$. This function is known as the extended Rosenbrock function. It is unimodal, yet due to a saddle point it is very difficult to locate the minimizer $x^* = (1, 1, \ldots, 1)$ with $f(x^*) = 0$.

39. Salomon Problem (SAL)

$$\min_x f(x) = 1 - \cos\left(2\pi\|x\|\right) + 0.1\|x\| \tag{B.79}$$

$$\text{subject to} \quad -100 \le x_i \le 100 \tag{B.80}$$

where $\|x\| = \sqrt{\sum_{i=1}^{n} x_i^2}$. The number of local minima (as a function of n) is not known, but the global minimizer is located at $x^* = (0, 0, 0, \ldots, 0)$ with $f(x^*) = 0$. Our tests were performed with $n = 10$.

40. Schaffer 1 Problem (SF1)

$$\min_x f(x) = 0.5 + \frac{\left(\sin\sqrt{x_1^2+x_2^2}\right)^2 - 0.5}{\left(1+0.001\left(x_1^2+x_2^2\right)\right)^2} \tag{B.81}$$

$$\text{subject to} \quad -100 \le x_1, x_2 \le 100. \tag{B.82}$$

The number of local minima is not known, but the global minimum is located at $x^* = (0, 0)$ with $f(x^*) = 0$.

41. Schaffer 2 Problem (SF2)

$$\min_x f(x) = (x_1^2 + x_2^2)^{0.25}\left(\sin^2\left(50(x_1^2 + x_2^2)^{0.1}\right) + 1\right) \tag{B.83}$$

$$\text{subject to} \quad -100 \le x_1, x_2 \le 100. \tag{B.84}$$

The number of local minima is not known, but the global minimum is located at $x^* = (0,0)$ with $f(x^*) = 0$.

42. Schubert Problem (SBT)

$$\min_x f(x) = \prod_{i=1}^n \left(\sum_{j=1}^{5} j \cos\left((j+1)x_i + j\right) \right) \tag{B.85}$$

$$\text{subject to} \quad -10 \le x_i \le 10, i \in \{1, 2, \ldots, n\}. \tag{B.86}$$

Our tests were performed with $n = 2$. The number of local minima for this problem (given n) is not known but for $n = 2$, the function has 760 local minima, 18 of which are global with $f(x^*) \approx -186.7309$. All two dimensional global minimizers are listed in Table B.12:

Table B.12: Global optimizers for Schubert problem.

x^*				
(-7.0835,4.8580),	(-7.0835,-7.7083),	(-1.4251,-7.0835),	(5.4828,4.8580),	(-1.4251,-0.8003),
(4.8580,5.4828),	(-7.7083,-7.0835),	(-7.0835,-1.4251),	(-7.7083,-0.8003),	(-7.7083,5.4828),
(-0.8003,-7.7083),	(-0.8003,-1.4251),	(-0.8003,4.8580),	(-1.4251,5.4828),	(5.4828,-7.7083),
(4.8580,-7.0835),	(5.4828,-1.4251),	(4.8580,-0.8003)		

43. Schwefel Problem (SWF)

$$\min_x f(x) = -\sum_{i=1}^{n} x_i \sin\left(\sqrt{|x_i|}\right) \tag{B.87}$$

$$\text{subject to} \quad -500 \le x_i \le 500, i \in \{1, 2, \ldots, n\}. \tag{B.88}$$

The number of local minima for a given n is not known, but the global minimum value $f(x^*) \approx -418.9829n$ is located at $x^* = (s, s, \ldots, s)$, $s \approx 420.97$. Our tests were performed with $n = 10$.

44. Shekel 5 Problem (S5)

$$\min_x f(x) = -\sum_{i=1}^{5} \frac{1}{\sum_{j=1}^{4} (x_j - a_{ij})^2 + c_i} \tag{B.89}$$

$$\text{subject to} \quad 0 \le x_j \le 10, j \in \{1, 2, 3, 4\}, \tag{B.90}$$

with constants a_{ij} and c_j given in Table B.13 below. There are five local minima and the global minimizer is located at $x^* = (4.00, 4.00, 4.00, 4.00)$ with $f(x^*) \approx -10.1532$.

Table B.13: Data for Shekel problem family.

	i	a_{ij}				c_i
		$j=1$	2	3	4	
S5	1	4	4	4	4	0.1
	2	1	1	1	1	0.2
	3	8	8	8	8	0.2
	4	6	6	6	6	0.4
	5	3	7	3	7	0.4
S7	6	2	9	2	9	0.6
	7	5	5	3	3	0.3
S10	8	8	1	8	1	0.7
	9	6	2	6	2	0.5
	10	7	3.6	7	3.6	0.5

45. Shekel 7 Problem (S7)

$$\min_x f(x) = -\sum_{j=1}^{7} \frac{1}{\sum_{i=1}^{4} (x_j - a_{ij})^2 + c_i} \tag{B.91}$$

$$\text{subject to} \quad 0 \le x_j \le 10, j \in \{1,2,3,4\}, \tag{B.92}$$

with constants a_{ij} and c_j given in Table B.13. There are seven local minima and the global minimizer is located at $x^* = (4.00, 4.00, 4.00, 4.00)$ with $f(x^*) \approx -10.4029$.

46. Shekel 10 Problem (S10)

$$\min_x f(x) = -\sum_{j=1}^{10} \frac{1}{\sum_{i=1}^{4} (x_j - a_{ij})^2 + c_i} \tag{B.93}$$

$$\text{subject to} \quad 0 \le x_j \le 10, j \in \{1,2,3,4\} \tag{B.94}$$

with constants a_{ij} and c_j given in Table B.13. There are 10 local minima and the global minimizer is located at $x^* = (4.00, 4.00, 4.00, 4.00)$ with $f(x^*) \approx -10.5364$.

47. Shekel's Foxholes (FX)

$$\min_x f(x) = -\sum_{j=1}^{30} \frac{1}{c_j + \sum_{i=1}^{n} (x_i - a_{ji})^2} \tag{B.95}$$

$$\text{subject to} \quad 0 \le x_i \le 10, i \in \{1,2,\ldots,10\}. \tag{B.96}$$

Our tests were performed with $n = 5$ and 10. The constants c_j and a_{ji} are given in Table B.14. The number of local minima is not known, but the global minima are presented in Table B.15.

Table B.14: Data for Shekel's foxholes problem.

j	c_j	a_{ji}									
		$i=1$	2	3	4	5	6	7	8	9	10
1	0.806	9.681	0.667	4.783	9.095	3.517	9.325	6.544	0.211	5.122	2.020
2	0.517	9.400	2.041	3.788	7.931	2.882	2.672	3.568	1.284	7.033	7.374
3	0.100	8.025	9.152	5.114	7.621	4.564	4.711	2.996	6.126	0.734	4.982
4	0.908	2.196	0.415	5.649	6.979	9.510	9.166	6.304	6.054	9.377	1.426
5	0.965	8.074	8.777	3.467	1.863	6.708	6.349	4.534	0.276	7.633	1.567
6	0.669	7.650	5.658	0.720	2.764	3.278	5.283	7.474	6.274	1.409	8.208
7	0.524	1.256	3.605	8.623	6.905	4.584	8.133	6.071	6.888	4.187	5.448
8	0.902	8.314	2.261	4.224	1.781	4.124	0.932	8.129	8.658	1.208	5.762
9	0.531	0.226	8.858	1.420	0.945	1.622	4.698	6.228	9.096	0.972	7.637
10	0.876	7.305	2.228	1.242	5.928	9.133	1.826	4.060	5.204	8.713	8.247
11	0.462	0.652	7.027	0.508	4.876	8.807	4.632	5.808	6.937	3.291	7.016
12	0.491	2.699	3.516	5.874	4.119	4.461	7.496	8.817	0.690	6.593	9.789
13	0.463	8.327	3.897	2.017	9.570	9.825	1.150	1.395	3.885	6.354	0.109
14	0.714	2.132	7.006	7.136	2.641	1.882	5.943	7.273	7.691	2.880	0.564
15	0.352	4.707	5.579	4.080	0.581	9.698	8.542	8.077	8.515	9.231	4.670
16	0.869	8.304	7.559	8.567	0.322	7.128	8.392	1.472	8.524	2.277	7.826
17	0.813	8.632	4.409	4.832	5.768	7.050	6.715	1.711	4.323	4.405	4.591
18	0.811	4.887	9.112	0.170	8.967	9.693	9.867	7.508	7.770	8.382	6.740
19	0.828	2.440	6.686	4.299	1.007	7.008	1.427	9.398	8.480	9.950	1.675
20	0.964	6.306	8.583	6.084	1.138	4.350	3.134	7.853	6.061	7.457	2.258
21	0.789	0.652	2.343	1.370	0.821	1.310	1.063	0.689	8.819	8.833	9.070
22	0.360	5.558	1.272	5.756	9.857	2.279	2.764	1.284	1.677	1.244	1.234
23	0.369	3.352	7.549	9.817	9.437	8.687	4.167	2.570	6.540	0.228	0.027
24	0.992	8.798	0.880	2.370	0.168	1.701	3.680	1.231	2.390	2.499	0.064
25	0.332	1.460	8.057	1.336	7.217	7.914	3.615	9.981	9.198	5.292	1.224
26	0.817	0.432	8.645	8.774	0.249	8.081	7.461	4.416	0.652	4.002	4.644
27	0.632	0.679	2.800	5.523	3.049	2.968	7.225	6.730	4.199	9.614	9.229
28	0.883	4.263	1.074	7.286	5.599	8.291	5.200	9.214	8.272	4.398	4.506
29	0.608	9.496	4.830	3.150	8.270	5.079	1.231	5.731	9.494	1.883	9.732
30	0.326	4.138	2.562	2.532	9.661	5.611	5.500	6.886	2.341	9.699	6.500

Table B.15: Global optimizers for Shekel's foxholes problem.

n	$f(x^*)$	x^*
5	-10.4056	(8.025, 9.152, 5.114, 7.621, 4.564)
10	-10.2088	(8.025, 9.152, 5.114, 7.621, 4.564, 4.771, 2.996, 6.126, 0.734, 4.982)

48. Sinusoidal Problem (SIN)

$$\min_x f(x) = -\left[A\prod_{i=1}^n \sin(x_i - z) + \prod_{i=1}^n \sin(B(x_i - z))\right] \tag{B.97}$$

$$\text{subject to} \quad 0 \le x_i \le 180, i \in \{1, 2, \ldots, n\}. \tag{B.98}$$

The variable x is in degrees. Parameter A affects the amplitude of the global optimum; B affects the periodicity and hence the number of local minima; z shifts the location of the global minimum; and n indicates the dimension. Our tests were performed with $A = 2.5, B = 5, z = 30$, and $n = 10$ and 20. The location of the global solution is at $x^* = (90 + z, 90 + z, \ldots, 90 + z)$ with the global optimum value of $f(x^*) = -(A+1)$. The number of local minima increases dramatically in dimension, and when $B = 5$ the number of local minima is equal to:

$$\sum_{i=0}^{\lfloor n/2 \rfloor} \left(\frac{n!}{(n-2i)!(2i)!} 3^{n-2i} 2^{2i} \right). \tag{B.99}$$

49. Storn's Tchebychev Problem (ST)

$$\min_x f(x) = p_1 + p_2 + p_3, \tag{B.100}$$

where

$$p_1 = \begin{cases} (u-d)^2 & \text{if } u < d \\ 0 & \text{if } u \geq d \end{cases} \qquad u = \sum_{i=1}^{n} (1.2)^{n-i} x_i$$

$$p_2 = \begin{cases} (v-d)^2 & \text{if } v < d \\ 0 & \text{if } v \geq d \end{cases} \qquad v = \sum_{i=1}^{n} (-1.2)^{n-i} x_i$$

$$p_3 = \sum_{j=0}^{m} \begin{cases} (w_j - 1)^2 & \text{if } w_j > 1 \\ (w_j + 1)^2 & \text{if } w_j < -1 \\ 0 & \text{if } -1 \leq w_j \leq 1 \end{cases} \qquad w_j = \sum_{i=1}^{n} \left(\frac{2j}{m} - 1 \right)^{n-i} x_i,$$

for $n = 9$: $x_i \in [-128, 128]^n$, $d = 72.661$, and $m = 60$

for $n = 17$: $x_i \in [-32768, 32768]^n$, $d = 10558.145$, and $m = 100$.

The number of local minima is not known but the global minimum is known to be as shown in Table B.16. Our tests were performed with $n = 9$.

Table B.16: Global optimizers for Storn's Tchebychev problem.

n	$f(x^*)$	x^*
9	0	(128, 0, -256, 0, 160, 0, -32, 0, 1)
17	0	(32768, 0, -1331072, 0, 21299, 0, -180224, 84480, 0, -2154, 0, 2688, 0, -128, 0, 1)

50. Wood's Problem (WP)

$$\min_x f(x) = 100(x_2 - x_1^2)^2 + (1-x_1)^2 + 90(x_4 - x_3^2)^2 + (1-x_3)^2 \tag{B.101}$$
$$+10.1[(x_2-1)^2 + (x_4-1)^2] + 19.8(x_2-1)(x_4-1)$$

$$\text{subject to} \quad -10 \leq x_i \leq 10, i \in \{1,2,3,4\}. \tag{B.102}$$

The function has a saddle near $(1,1,1,1)$. The only minimum is located at $x^* = (1,1,1,1)$ with $f(x^*) = 0$.

Bibliography

[1] Adjiman C.S., Dallwig S., Floudas C.A., and Neumaier A. A global optimization method, αBB, for general twice-differentiable constrained NLPs: I. Theoretical advances. *Computers & Chemical Engineering, 22(9)* (1998), 1137–1158.

[2] Aarts, E. and Korst, J. *Simulated Annealing and Boltzmann Machines*. Wiley, Chichester, 1989.

[3] Alberto, P., Nogueira, F., Rocha, H. and Vicente, L.N. Pattern search methods for user-provided points: Application to molecular geometry problems, *SIAM Journal on Optimization*, *14* (2004), 1216–1236.

[4] Abramson, M.A., Audet, C. and Dennis, J. E. Generalized pattern searches with derivative information. *Mathematical Programming*, *100* (2004) 3–25.

[5] Ali, M.M., Khompatraporn, C. and Zabinsky, Z.B. A numerical evaluation of several stochastic algorithms on selected continuous global optimization test problems. *Journal of Global Optimization 31* (2005), 35–672.

[6] Ali, M.M. and Storey, C. Modified controlled random search algorithms. *Intern. Journal of Computer Mathematics 53* (1994), 229–235.

[7] Ali, M.M. and Storey, C. Aspiration based simulated annealing algorithm. *Journal of Global Optimization 11* (1997), 181–191.

[8] Ali, M.M., Storey, C. and Törn, A. Application of some stochastic global optimization algorithms to practical problems. *Journal of Optimization Theory and Applications, 95* (1997), 545–563.

[9] Ali, M.M., Törn, A. and Viitanen, S. A direct search variant of the simulated annealing algorithm for optimization involving continuous variables. *Computers & Operations Research 29* (2002), 87–102.

[10] Ali, M.M., Törn, A. and Viitanen, S. A numerical comparison of some modified controlled random search algorithms. *Journal of Global Optimization 11* (1997), 377–385.

[11] Ali, M.M. and Törn, A. Population set-based global optimization algorithms: Some modifications and numerical studies. *Computers & Operations Research 31(10)* (2004), 1703–1725.

[12] Ali, M.M. and Törn, A. Topographical differential evolution using pre-calculated differentials. In *Stochastic and Global Optimization*, G. Dzemyda, V. Saltenis and A. Zilinskas, Eds. Kluwer Academic Publishers, Dordrecht (2002), 1–17.

[13] Ali, M.M. and Storey, C. Topographical multilevel single linkage. *Journal of Global Optimization 5* (1994), 349–358.

[14] Alluffi-Pentini, F., Parisi, V. and Zirilli, F. Global optimization and stochastic differential equations. *Journal of Optimization Theory and Applications 47* (1985), 1–16.

[15] Audet, C. and Dennis, J. E. Analysis of generalized pattern search. *SIAM Journal on Optimization, 13(3)* (2004), 889–903.

[16] Benjamin, A. The Bisection Method: Which Root? *American Mathematical Monthly, 94(9)* (1987), 861–863, Jstor.

[17] Bohachevsky, M.E., Johnson, M.E. and Stein, M.L. Generalized simulated annealing for function optimization. *Technometrics, 28* (1986), 209–217.

[18] Breiman, L. and Cutler, A. A deterministic algorithm for global optimization. *Mathematical Programming 58* (1993), 179–199.

[19] Chelouah, R. and Siarry, P. Tabu search applied to global optimization. *European Journal of Operational Research 123(2)* (2000), 256–270.

[20] Chelouah, R. and Siarry, P. A hybrid method combining continuous tabu search and Nelder-Mead simplex algorithms for the global optimization of multiminima functions. *European Journal of Operational Research 161(3)* (2005), 636–654.

[21] Dekkers, A. and Aarts, E. Global optimization and simulated annealing. *Mathematical Programming 50* (1991), 367–393.

[22] Floudas, A. and Pardalos, M., Eds. *Recent Advances in Global Optimization*. Princeton University Press, Princeton, (1992).

[23] Goldberg, D.E. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Publishing Company, Reading, (1989).

[24] Hedar, A. and Fukushima, M. Heuristic Pattern Search and its hybridization with simulated annealing for nonlinear global optimization. *Optimization Methods and Software, 19* (2004), 291–308.

[25] Hooke, R. and Jeeves, T.A. Direct search solution of numerical and statistical problems. *Journal of the Association for Computing Machinery 5* (1961), 212–229.

[26] Horst, R. and Tuy, H. *Global Optimization: Deterministic Approaches*, 3rd edition Springer-Verlag, Berlin, (1996).

[27] Ichida, K. and Fujii, Y. An interval arithmetic method for global optimization. *Journal of Computing 23(1)* (1979), 85–97.

[28] Pintér J.P. Global optimization: Software, test problems, and applications. *Pintér Consulting Services Inc. and Dalhousie University*.

[29] Kirkpatrick, S., Gelatt, C.D. and Vecchi, M.P. Optimization by simulated annealing. *Science 220* (1983), 671–680.

[30] Kolda, T.G., Lewis, R.M. and Torczon, V. Optimization by direct dearch: New perspective on classical and modern methods. *SIAM Review, 45(3)* (2003), 385–482.

[31] Kvasnička, V. and Pospichal, J. A hybrid of simplex method and simulated annealing. *Chemometrics and Intelligent Laboratory Systems 39* (1997), 161–173.

[32] Locatelli, M. Convergence of a simulated annealing algorithm for continuous global optimization. *Journal of Global Optimization 18* (2000), 219–233.

[33] Locatelli, M. Simulated annealing algorithms for continuous global optimization. *Handbook of Global Optimization II*, Kluwer Academic Publishers, (2002), 179–230.

[34] Locatelli, M. A note on the Griewank test function. *Journal of Global Optimization 25* (2003), 169–174.

[35] Masri S., Bekey G., and Safford F. A global optimization algorithm using adaptive random search. *Applied Mathematics and Computation*, *7* (1980), 353–375.

[36] Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N, Teller, A.H. and Teller, E. Equation of state calculation by fast computing machines. *Journal of Chemical Physics 2(6)* (1953), 1087–1091.

[37] Nelder, J. A. and Mead, R. A simplex method for function minimization. *Computer Journal 7* (1965), 308–313.

[38] Nemhauser, G.L., Rinnooy Kan, A.H.G. and Todd, N.J. *Optimization*. North-Holland, 1989.

[39] Preux, P.H. and Talbi, E.G. Towards hybrid evolutionary algorithms. *International Transaction in Operational Research 6(6)* (1999), 557–570.

[40] Price, W.L. A controlled random search procedure for global optimization. *The Computer Journal 20* (1978), 367–370.

[41] Rinnooy Kan, A.H.G. and Timmer, G.T. Stochastic global optimization methods, Part I: Clustering methods. *Mathematical Programming 39* (1987), 27–56.

[42] Rinnooy Kan, A.H.G. and Timmer, G.T. Stochastic global optimization methods, Part II: Multi level methods. *Mathematical Programming 39* (1987), 57–78.

[43] Sahinidis, N.V. BARON: A general purpose global optimization software package. *Journal of Global Optimization, 8(2)* (1996), 201–205.

[44] Schoen F. Stochastic global optimization: Two phase methods. *In Chris Floudas and Panos Pardalos, editors, Encyclopedia of Optimization*, 301–305 V. Kluwer Academic Publishers, Dordrecht, 2001.

[45] Storn, R. and Price, K. Differential evolution – A simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization 11* (1997), 341–359.

[46] Talbi, E. G. Taxonomy of hybrid metaheuristics. *Journal of Heuristics 8(5)* (2002), 541–564.

[47] Törn, A., Ali, M.M. and Viitanen, S. Stochastic global optimization: Problem classes and solution techniques. *Journal of Global Optimization 14* (1999), 437–447.

[48] Torczon, V. On the convergence of pattern search algorithms. *SIAM Journal on Optimization, 12(4)* (1997), 1075–1089.

[49] Torczon, V. *Multi-directional search. A direct search algorithm for parallel machines, Ph.D thesis,* (1989), Department of Mathematical sciences, Rice University, Houston.

[50] Vanderbilt, D. and Louie, S.G. A Monte Carlo simulated annealing approach to Optimization over Continuous Variables. *Journal of Computational Physics 56* (1984), 259–271.

[51] Wang, P.P. and Chen, D.S. Continuous optimization by a variant of simulated annealing. *Computational Optimization and Applications 6(1)* (1996), 59–71.

[52] Zhang, B., Wood, G.R. and Baritompa, W.P. Multidimensional bisection: The performance and the context. *Journal of Global Optimization 3(3)* (1993), 337–358.