UNIVERSIDAD NACIONAL DE COLOMBIA

Optimization of Nonlinear Functions with Planar Regions Using Supernova

Eddy Janneth Mesa Delgado

Universidad Nacional de Colombia
Facultad de Minas
Departamento de ciencias de la computación y la decisión
Medellín, Colombia
2017

Optimization of Nonlinear Functions with Planar Regions Using Supernova

Eddy Janneth Mesa Delgado

Tesis presentada como requisito parcial para optar al título de:
Doctor en Ingeniería de Sistemas

Director:
Juan David Velásquez Ph.D.
Codirectora
Patricia Jaramillo Ph. D.

Universidad Nacional de Colombia
Facultad de Minas
Area curricular de sistemas e informática
Medellín, Colombia
2017

For my parents and friends.

Acknowledgements

I would like to thank those persons who made this research possible, worthwhile and enjoyable. A special note to Juan David Velásquez and Patricia Jaramillo who support and guide the work.

Abstract

Nowadays, optimization is begun to be use in different fields, e.g. preference algorithms. These new challenges need a robustness meta heuristics to solve them. Supernova meta heuristic that emules the descent behavior of the gradients and share the same weakness of them. They get stuck planar regions and hardly find the needle minimum. The main objective of this works is to improve the performance of the original version of supernova for the problematic topologies mention above. First, a review of how to these problems are solved in the literature is presented. Second, A criterion to determine planar regions is described . Third, a strategy to choose the parameters agree with the topology of the function is implemented. Supernova 2.0 was tested using the set of benchmarks functions proposed in CEC2013. The new version is significantly better than the original version, no significantly better than SPSO2011 and significantly inferior with SADE. Although, the results are applied to Supernova, most of the strategies can be applied to other methods.

Keywords: Metaheuristics, nonlinear optimization, Supernova

TABLE OF CONTENTS

1. INTRODUCTION

"All men have stars, but they are not the same things for different people. For some, who are travelers, the stars are guides. For others they are no more than little lights in the sky. For others, who are scholars, they are problems... But all these stars are silent."
–Antoine de Saint-Exupery, the Little Prince

Metaheuristics are direct methods for optimization that do not need derivatives. Nowadays, real-world optimization problems are more difficult because there are shorter times to solve them. Moreover, models and solutions need more accuracy; e.i. more variables and interaction among them and values closer to the optimal point (Rothlauf, 2011). Classic methods of optimization, as gradient methods, are not enough to solve all optimization problems (Himmelblau, 1972). Therefore, new strategies have been developed. They do not use the derivative information and evaluate directly the objective function. The first direct methods implemented were systematic searches over feasible space. Step by step, the methods evolved to become more and more effective algorithms until arriving to artificial intelligence (Corne, Dorigo, & Glover, 1999). These new methods were called heuristics, metaheuristics and hyper-heuristics. They work, but they cannot guarantee an optimal solution. All these methods search directly over feasible region following different inspirations, for example, natural phenomena. (Olariu & Zomaya, 2006; Gendreau & Potvin, 2010).

Metaheuristics simulate part of an inspirational phenomenon to get some features, which guide the search according to with these particular characteristics. A good example for a bioinspired metaheuristic is ants. The manner that ants use to communicate among them inspired one algorithm for combinatorial optimization. Ants mark the path towards feed with pheromones. Shortest paths have more pheromone and become the most popular increasing its use by other individuals. In addition, the algorithm uses a similar idea to find new paths and improving the current path. Starting with an incomplete solution, the path is marked with pheromone until found the shortest way. This means; optimal combination (Talbi, 2009). There are diverse phenomena that inspiring the heuristics, but the common factor among them the phenomena improve or order something, for example; Evolution improve a population (Yang, 2010).

Supernovae are a complex phenomenon composed by different interactions, e.g. thermodynamics, kinetics, among others. Focusing in the movement among the particles expelled by the star, a reduction of the phenomenon for this work will be: impulse and gravity. The initial force gives to the particles a random velocity and direction, e.i., impulse. Other bodies around attract these particles to add them. Initially, the explosion breaks the equilibrium, after a while, the different

interactions between the stellar material and the environment arrive at an equilibrium state again (Lieddle, 2003). We use the concept of improvement implicit on the way to the new equilibrium to idealize a search in a previous work. The trajectory followed by the particles is modified by the heavier bodies changing the initial direction given by the explosion. Most of the particles will be attracted to other bigger and closer bodies around the explosion. An analogy between the optimization process and supernovae could be: the valleys or local minimums are the heavier bodies. The mass will be the value of the objective function, and particles are the solutions that become attracted those minimums.

The proposed reduction of the phenomenon has two important parts, impulse and gravity force. Both have been inspired another metaheuristics as: gravitational local search (GLSA)(Webster & Bernhard, 2003), big-bang big-crunch (BBBC) (Erol & Eksin, 2006), gravitational search (GSA) (Rashedi, Nezamabadi-pour, & Saryazdi, 2009) and central force optimization (CFO) (Formato, 2007). These four methods differ each other; the first one is a local method while the others are global. BBBC uses integer variables and calculates an inertial mass point; CFO is deterministic and GSA is stochastic. Other method inspired in central force is electromagnetic algorithm, it is similar to GSA.

Following this idea, we proposed a novel metaheuristic called Supernova (Mesa, 2010). It is a new approach based on the gravity analogy and inspired in the supernova phenomenon. Our methodology shows the following advantages:

- The method can find a solution with smaller objective function in the case of minimization and fewer calls to objective function for different kind of functions.
- It is robust for high dimensional problems convex, quasi-convex, smoothness slopes and high rugosity functions.
- The range of the parameters is known and easy to tune.

Moreover, supernova is a new metaheuristic that has still been developed. When this research began the methodology was immature and had some weakness as: handle of constraints, unsuccessful performance in planar regions and needle minima, control and tune of parameters, high computational costs, among others. We focus to solve two of them: The performance over planar search regions and needle minimum, and the computation cost. Although, there are many ways to improve metaheuristics, the main focus of this work is: how the control and selection of the parameters can increase the effectiveness over the kind of regions mention above. In addition, we proposed different manner of initialization and scaled functions as tools to increase the exploration of the metaheuristic. So, the main goal of this work is to present a research proposal propose a method to improve the performance of Supernove in planar regions.

This chapter is organized as follows. In Section 2, we present the previous and related works. In Section 3, we discuss the problem. In Section 4, we present the requirements to solve the problem, hypothesis, objectives and contributions to this research. In last section, we describe the content of the complete text.

1.1 SUPERNOVA: The previous work

In a previous research, we propose a metaheuristic inspired in supernovae (Mesa, 2010). In this section, a brief summary of the process and results we obtain in the previous work is presented.

Initially, the basic concepts of the metaheuristics are described. Afterward, the implementation of the algorithm from previous work. In the third part, we discuss about the convergence of the algorithm. Finally, the performance and similarity of supernova metaheuristic and another metaheuristics like GSA is discussed.

1.1.1 Supernova Methodology

Supernovae are a very complex phenomenon; it includes aspects to be analyzed from areas such as thermodynamics, quantum mechanics, etc. If we simulate the entire process of supernovae, surely to solve the algorithm we would require more effort than the required to find a solution for the problem itself. Then, our approach only focus on a subset of the phenomenon. When the star explodes, it ejects particles around it. The particles move until achieving an equilibrium as the initial status previous to explosion. The system has a high level of energy that is better distributed after particles are in balance another time out of the star (Lieddle, 2003).

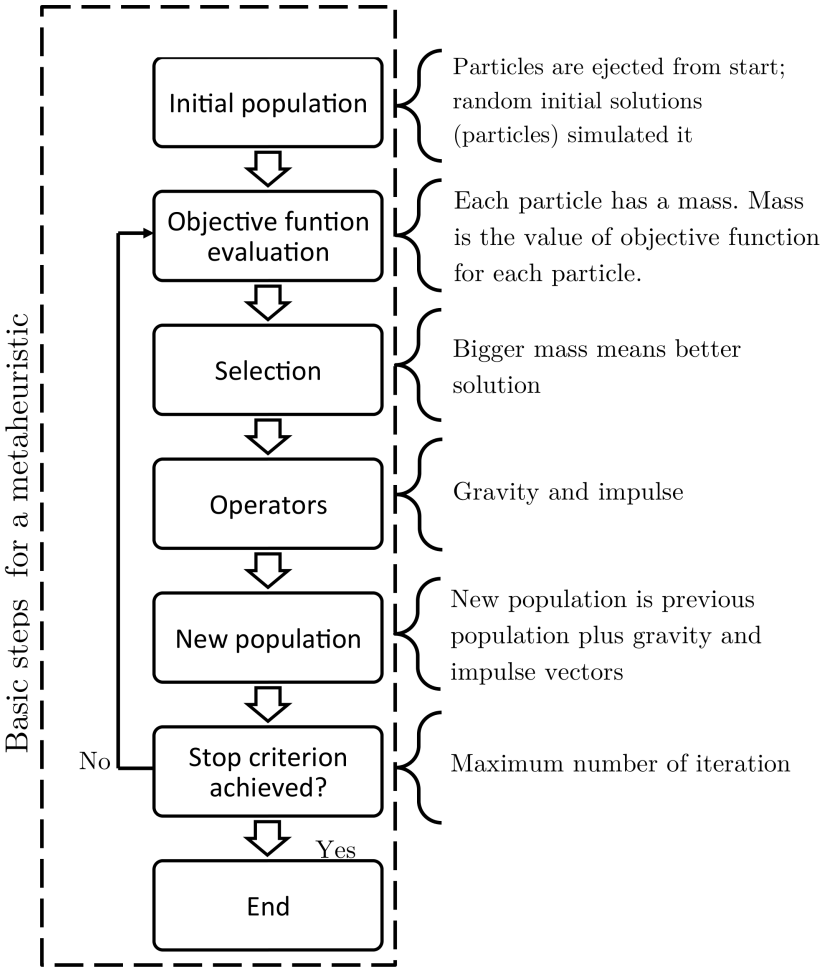


Figure 1.1: Sequence followed by the algorithm

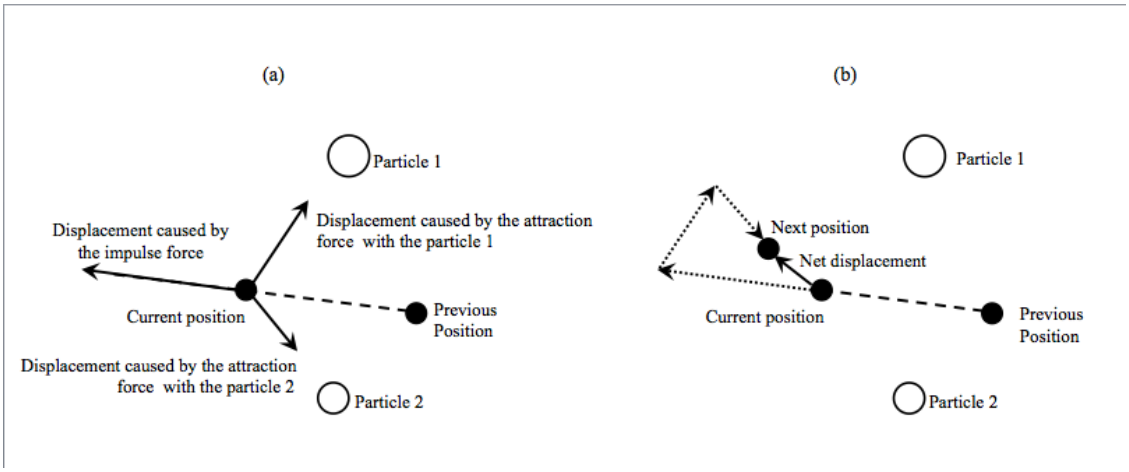


Figure 1.2: Forces actuating over a single particle.

Usually, a metaheuristic can be divided into four main generic steps: initial population, evaluation, selection, operations(gravity and impulse) and new population. Each step of this sequence is different for each method, i.e. each search process varies from one to one according to the approach. In the Figure 1.1, we present the sequence for the proposed algorithm inspired by supernovae. Note that a particle is equivalent to a solution vector. Particles have a property: the mass. Mass is the value of this mass can be the value of the objective function or a scaled value of this function.

The main difference of the proposed algorithm with others metaheuristics is in the operations step. We simplify and idealize the real phenomenon recreating a simple dynamics to control the interaction of the particles in the macro physic part (expel and repel). The movement of the particles is proposed as a strategy for exploring the search space. Thus, only two forces are used in our heuristic: the attraction force generated by the gravity, and the impulse force caused by the explosion of the star (Torn & Zilinskas, 1989; Wolpert & Macready, 1997; Zanakis & Evans, 1981). Accordingly, the algorithm achieves a combination of exploration and exploitation, which is necessary for global optimization (Torn & Zilinskas, 1989). In order to exploit the best regions found, the strategy is restarted with a smaller impulse. Solutions (particles) have a value of objective function (mass), and the distance between solutions can be calculated. Then, a set of solutions moves to agree with gravitational force and impulse, starting with a random explosion. In the supernova phenomenon, the magnitudes are huger compared with typical optimization applications, so, the constants for gravity and impulse are parameters of the metaheuristic.

For the proposed algorithm, we have K particles in each iteration. In Figure 1.2, we analyze the displacement of the particle k assuming that other particles remain static. Black points in Figure 1.2 represent the trajectory of the analyzed particle, whereas white points represent the other particles. In Figure 1.2(a), we show the displacements caused by the impulse and attraction forces with particles 1 and 2. The displacement $\mathbf{i}_k$ of the particle k caused by the impulse force is:

$$\mathbf{i}_k = F \cdot m_k \cdot v_k \cdot \mathbf{d}_k \tag{1.1}$$

where F is a constant parameter for the impulse force; $\mathbf{d}_k$ is the current direction; m_k is the mass and v_k is the velocity. The displacement caused by the attraction force between particles k

and i, $\mathbf{a}_{ki}$, is calculated as (see Figure 1.2(a)):

$$\mathbf{a}_{kj} = G \cdot m_k \cdot m_j \cdot \left(r_{kj}^2\right)^{-1} \cdot \mathbf{u}_{kj} \tag{1.2}$$

where G denotes the gravitational constant, r_{kj} denotes the distance between particles k and j , and $\mathbf{u}_{kj}$ denotes a unitary vector in the same direction between particles k and j. Thus, the net displacement Δ_k is calculated using the displacements caused by the attraction forces among all particles and the inertia:

$$\Delta_k = \mathbf{i}_k + \sum_{\substack{j=1 \\ j \neq k}}^{K} \mathbf{a}_{kj}, \ \forall k \tag{1.3}$$

Thus, the next position of the particle is calculated as the current position plus the new vector of displacement, as show in Figure 1.2(b). Equations 1.1, 1.2 and 1.3 are the equations of the phenomenon, which are very similar to those raised for the metaheuristic.

1.1.2 Algorithm description

In the Algorithm 1, we present the pseudocode of the metaheuristic proposed and called Supernova. We are interested solving problems of the type:

$$\min f(\mathbf{x}) \tag{1.4}$$

$$\text{s.t.} \quad \mathbf{LB} \leq \mathbf{x} \leq \mathbf{UB} \tag{1.5}$$

where $\mathbf{x}$ is a vector of $N \times 1$ components and $f(\cdot)$ is a bounded real-valued and continuous function, such that $f : \Re^n \to \Re$. The solution for this problem is the global minimum.

Definition 1.1.1. Let $\mathbf{x^*}$ a vector space $\Re^n$ and $f(\cdot)$ a function such that $f : \Re^n \to \Re$. $\mathbf{x^*}$ is the global minimum if and only if $f(\mathbf{x^*}) \leq f(\mathbf{x})$ for the entire domain of $f(\cdot)$.

There is not an optimization method that guarantees to find global minimum, there are strategies to increase the probability to find global minimum (Torn & Zilinskas, 1989). In the proposed algorithm, we defined a vector $\mathbf{x}_G$ such that $f(\mathbf{x^*}) \leq f(\mathbf{x}_G) \leq f(\mathbf{x})$. Notice that $\mathbf{x}_G$ is an approximation to global minimum.

Now, we define some notation used in the algorithm. We have a population with K particles. The particle k in the iteration t has the following properties: a current position, $\mathbf{x}_k$ for iteration t; a previous position, $\mathbf{p}_k$ is $\mathbf{x}_k$ at $t-1$ iteration; a next position, $\mathbf{s}_k$ is $\mathbf{x}_k$ at $t+1$; a mass, m_k; a unitary direction, $\mathbf{d}_k$; and a velocity, v_k; m_k, $\mathbf{d}_k$ and v_k are particular for iteration t. In addition, $\mathbf{n}_{kt}$ denotes the best point visited by the particle k until the iteration t; This is a memory vector to control, but it is not used to calculate the next point as in particle swarm optimization (PSO) method. Supernova iterates over $t = 1, \cdots, T$, where T is the maximum number of iterations. Also, the proposed algorithm has a mechanism of restarts that operates between $r = 1, \cdots, R$, where R is the maximum number of restarts.

Algorithm 1 supernova pseudocode

Require: T, R, λ_0, $\mathbf{x}_0$, K, α, η, G, F, $f(\cdot)$

$t \leftarrow 1; r \leftarrow 0; \lambda \leftarrow \lambda_0$
while $t \leq T$ **and** $r \leq R$ **do**
  if $t \neq 1$ **then** $\mathbf{x}_0 \leftarrow \arg\min_{1 \leq k \leq K} f(\mathbf{n}_k)$ **end if**
  for $k = 1$ **to** K **do**
    $v_k \leftarrow \text{runif(K)}$; $\mathbf{d}_k \leftarrow 2 \cdot \text{runif}(N) - 1$; $\mathbf{d}_k \leftarrow \frac{\mathbf{d}_k}{\|\mathbf{d}_k\|}$
    $\mathbf{s}_k \leftarrow \mathbf{x}_0 + \lambda \cdot v_k \cdot \mathbf{d}_k$
    if $t = 1$ **then** $\mathbf{n}_k \leftarrow \arg\min\{f(\mathbf{x}_0), f(\mathbf{s}_k)\}$**end if**
  end for
  $f^* \leftarrow \min_{1 \leq k \leq K} f(\mathbf{s}_k)$
  Restart $\leftarrow$ FALSE
  while Restart=FALSE **or** $t \leq T$ **or** $r \leq R$ **do**
    for $k = 1$ to K **do**
      $\mathbf{p}_k \leftarrow \mathbf{x}_k$; $\mathbf{x}_k \leftarrow \mathbf{s}_k$
    end for
    for $i = 1$ **to** K **do**
      for $j = 1$ **to** K **do** $r_{ij} \leftarrow \|\mathbf{x}_i - \mathbf{x}_j\|$ **end for**
    end for
    for $k = 1$ **to** K **do**
      $m_k \leftarrow f(\mathbf{x}_k) + 1 - f^*$;
      $\mathbf{n}_k \leftarrow \arg\min\{f(\mathbf{n}_k), f(\mathbf{p}_k), f(\mathbf{x}_k)\}$
      $\mathbf{d}_k \leftarrow \frac{\mathbf{p}_k - \mathbf{x}_k}{\|\mathbf{p}_k - \mathbf{x}_k\|}$
      for $i = 1$ **to** K **do** $\mathbf{u}_{ik} \leftarrow \frac{\mathbf{x}_i - \mathbf{x}_k}{\|\mathbf{x}_i - \mathbf{x}_k\|}$ **end for**
      $\mathbf{h}_k \leftarrow \sum_{\substack{i=1 \\ i \neq k}}^{K} G \cdot m_k^{-1} \cdot m_i^{-1} \cdot \left[1 - r_{ik} \cdot \left(\sum_{\substack{j=1 \\ j \neq k}}^{K} r_{jk}\right)^{-1}\right] \cdot \mathbf{u}_k$
      $\mathbf{s}_k \leftarrow \mathbf{x}_k + m_k^{-1} \cdot v_k \cdot \mathbf{d}_k + \mathbf{h}_k$
    end for
    $\mathbf{x}^* \leftarrow \arg\min_{1 \leq k \leq K} f(\mathbf{s}_k)$; $f^* \leftarrow f(\mathbf{x}^*)$
    if $t = 1$ **or** $f^* \leq f(\mathbf{x}_G)$ **then**
      $\mathbf{x}_G \leftarrow \mathbf{x}^*$; $r \leftarrow 0$
    else
      $r \leftarrow r + 1$
    end if
    if $r > \eta \cdot T$ **then** Restart $\leftarrow$ TRUE **end if**
    $t \leftarrow t + 1$
  end while
  $t \leftarrow 1$; $\lambda \leftarrow \frac{\lambda}{\alpha}$
end while
return $\mathbf{x}_G; f(\mathbf{x}_G)$

Initialization

For the first iteration ($t = 1$) we simulate an explosion with randomness. In this case, the particles are ejected from a center ($\mathbf{x}_0$) with an initial velocity v_k such that $0 \leq v_k \leq 1$ and uniformly

distributed (See algorithm 1, line 5), whereas the direction $\mathbf{d}_k$ (line 5) is an unitary random vector. The distance of the particle from the center is given by the expression $v_k \cdot \lambda$. Where λ is a parameter that indicates a new the search space smaller than the initial one (see Line 6).

The initial position $\mathbf{s}_k$ of each expelled particle is a random solution inside of the hyper-sphere with center $\mathbf{x}_0$ and radius λ (line 6); $\mathbf{x}_0$ represents the location of the star before it explodes and the initial value is the limits of the variables or a parameter defined by the user.

Function evaluation

In the developed methodology, the value of objective function evaluated in $\mathbf{x}_k$ is assimilated as the mass m_k of the particle in equations (1.1 to1.3) with $m_k > 0$. Thus, for function minimization or maximization, we use m_k^{-1} as the mass of particle k, where:

$$m_k = f(\mathbf{x}_k) + 1 - \min f(\mathbf{x}_k) \tag{1.6}$$

Previous transformation guarantees values greater or equal than one. In line 19, we use f^* to denote the minimum of $f(\cdot)$ which is calculated in the line 9. Notice that equation 1.6 is linearly and dynamically scaled to guarantee a mass with a minimum value of 1. It was introduced to avoid the division by zero without the modification of the mass interactions that can be induced by amounts near to zero. The value of one increases exploration of the search. Although there are another functions that can be used to scale the objective function(Bäck, 1996). The proposed scale does not introduce any modification over the function's topology. i.e., scale does not introduce an extra bias over search; this allowing us to know the real behavior of the metaheuristic developed.

Selection

The proposed algorithm is elitist. The selection, in this case, is associated with memory. The best solution for each particle is saved in the matrix n_k. For the first iteration, n_k and p_k are initialized using $\mathbf{x}_0$. The next iteration will be changed with solutions that are strictly better than the saved solution. At the end of the process we will choose the best solution for all the particles evaluated.

Operations

For the proposed metaheuristic, the new population is calculated. the displacement is computed for each particle using equation 1.3. In line 1, $\mathbf{d}_k$ is the unitary vector in direction from the previous position to the current position of the particle k. In line 22, the unitary vector in the direction of points i and k , $\mathbf{u}_{ik}$, is calculated. In line 23, we calculate the next displacement caused for all attraction forces (see equation 1.2). However, the straightforward use of equation 1.2 is not possible due to the magnitude of the values used for measuring masses and distances in the real phenomenon are not comparable with optimization problems, e.g. the order of magnitude of gravitational constant is 10^{-11}. Thus, we replace $(r_{ki}^2)^{-1}$ in equation (1.2) for a new expression $\mathbf{a}_{ki}$ in equation 1.7 to guarantee a bigger influence for near particles and lower influence for high distances between particles as in the gravity fields, without the problem of division by zero for close

particles:

$$\mathbf{a}_{ki} = G \cdot m_k \cdot m_i \cdot \left[1 - r_{ki} \cdot \left(\sum_{\substack{j=1 \\ j \neq k}}^{K} r_{jk} \right)^{-1} \right] \cdot \mathbf{u}_k \tag{1.7}$$

Note that the variation introduced in equation (1.2) implies a change between a real measure and a weighting between zero and one. This modification causes change in the magnitude of the calculated vectors, which generates an asymmetry in the interaction. This asymmetry helps to the exploitation process because it reduces the evaluation of repeated points that might be produced by central force. Indeed, equation (1.7) is the most noticeable difference with other approaches inspired by gravitational force. Moreover, this internal procedure is different from other methods such as PSO. While in PSO the strategy focuses on, "Follow the leader", here attraction force throw best solution is smaller than throw wrost.

Restart

In heuristic methods, it is well known that the initial exploration usually is insufficient for certain multimodal regions because there are no enough focus to find the best point (Kirkpatrick, Gelatt, & Vecchi, 1983). When algorithm restarts, a search in a smaller region around the best point found begins, improving the accuracy. When the algorithm reaches $\eta \cdot T$ iterations without improving the current optimum (see line 31), the algorithm is restarted (line 32), and a new set of particles is created. The parameter η is a constant between 0 and 1. For this time, the radius of the hypersphere containing the initial particles is reduced (line 35). Then, the best point found until the current iteration (line 11) is used as the initial point (line 3), and a new explosion occurs (line 4).

Stop criterion

The iterative process finishes when the maximum number of iterations T and restarts R are reached as many other metaheuristics (Talbi, 2009). For last part of test, we include a stop when the algorithm achieves an error equal to 10^{-9}.

1.1.3 Testing

After implementation, we started to evaluate supernova dividing the test in three parts. In the first part, we evaluated parameters, function of the algorithm, and synergy between strategies of the metaheuristic. In the second part, some functions were rotated and translated to obtain different configuration of the problem to check the robustness. Finally, in the third part, Supernova was compared against another metaheuristics well known for validating the performance of proposed metaheuristic. This section is divided in four parts: the right behavior of the metaheuristic and recommended values for the parameters, performance for specific functions configurations, description of benchmark functions, and results.

Convergence analysis

Subsequently, we describe briefly the results found by supernova algorithm used to evaluate the convergence, consists in three functions: Sphere, Rosenbrock and Ackley for two dimensions (corresponding to benchmark functions f_1, f_5 and f_{10} from test detailed by (Yao, Liu, & Lin, 1999)).

For each test function, we plot the mean and deviation against the number of calls to objective function. Figure 1.3 presents the results; part (a) shows the convergence for each function without the restart part; restart strategy drew in part (b). Finally, the complete algorithm convergence is in part (c) of the Figure 1.3 from those graphics. Then, we conclude that both strategies (movement and restart) present an asymptotic behavior and have synergy, which improves the performance of each one of the parts.

Besides the test described above, we made an empirical test to determine the robustness to initial solution, rotation, asymmetry and dimensions. For this, we transform the three test functions with the aim of reaching asymmetry and negative values for the global minimum in each case. From the results, we could conclude that supernova algorithm is not sensitive to initial solution (complete test reported in (Mesa, 2010)). The increase of dimensions is proportional to minimum quality for Rosenbrock function while for Sphere and Ackley functions the results are similar until 100 dimensions. Supernova is not sensitive to asymmetric functions or negative ones.

Moreover, from this part on the test, we conclude about computational effort from computational time; the parts of the search that increase the computational time, in order of importance, are: dimensions, particles, iterations, and restarts. We obtain an average time of 15.3452 seconds for the optimization of a problem with 30 dimensions in a processor 3.40 GHz and a RAM memory of 2 GB.

Recommended values for the parameters

After the initial test of convergence presented before, we made an experimental design to determine the best set of parameters to use with the algorithm. For this, we optimize the set of benchmark functions proposed by De Jong (1975), and an extra multimodal function called Ackley function (the complete set of functions used to parameterize corresponds to the functions f_1, f_5, f_{10}, f_6, f_7 and f_{14} of the Table 1.1).

Supernova has seven parameters. The constants F and G that tell the search how much gravity or impulse use for the search; this means, for $F > G$ the search increases the amount of the impulse and decreases the importance of gravity for the movement strategy. Iterations (T) and restarts (R) limit how long the search is. The number of particles (K) is the size of search population. And finally, initial radius of the hyper-sphere for generating the particles (λ_0) and the factor for reducing the radius of the hyper-sphere in each restart (α) define the intensity of the exploitation. For a higher α the radius of the hyper-sphere decrease faster than for a small one. This couple is closely related with the original size within the search region.

We varied the parameters in specific ranges, for F and G, the range is $[1, 10]$ with a gap of 0.1 for each function. For this couple of parameters, we obtained that: both parameters work better when they are equal. Moreover, the most of the functions got a good minimum with $F = G$ between 1 and 5. For T we used values between $[100, 600]$ vary each 100 while for restarts we tested between $[10, 100]$ vary each 10. A good configuration, in general, was $T = 100$ and $R \leq 50$. And K was tested between $[10, 100]$ each 10, and we found a big reduction of the minimum from 30 particles until 60 particles. From 60 to 100 the reduction was smaller. So the best configuration for the number of particles is between 30 and 60. In this case, we took 50 to compare with the other metaheuristics.

The last two parameters are: the initial radius of the hyper-sphere (λ_0) and the factor for reducing the radius of the hyper-sphere in each restart (α). We tested λ_0 between $[10, 1000]$ covering the size of the region of each 100 function and α between $[1.1, 1.5]$, agree with the initial

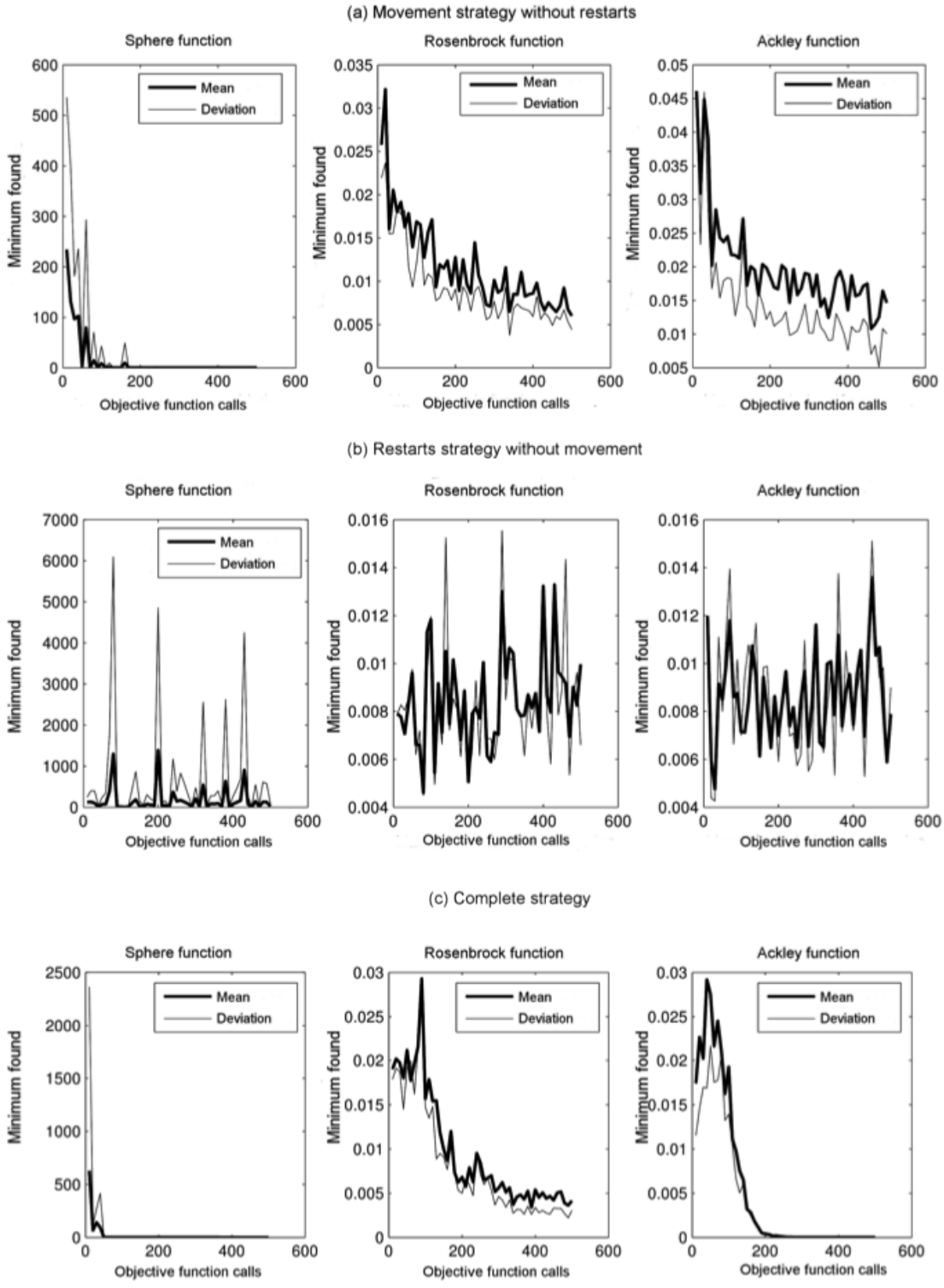


Figure 1.3: Convergence graphic.

idea, those parameters were related with the size of the region, we found that the optimal set for this parameter depends of size of search region and big regions needs big (λ_0). For small search spaces less 30 units or huge spaces over 200 units. For small regions, the most general set of parameters was $\lambda_0 = 30$; for big regions, we use $\lambda_0 = 600$ and $\alpha = 1.1$ or $\alpha = 1.5$ for small and big region respectively. For the sake of brevity, complete results are not reported here, but those parameters recommendations were used for the complete test described and analyzed in next sections.

1.1.4 Experimental setup

Initially, the convergence and a general parameter values was tested. Afterwards, supernova results was compared with other two well-know metaheuristics as: DE and PSO. First, we describe benchmark functions. In the next part, we describe briefly DE and PSO algorithms. Finally, the results are described.

Benchmark functions

The twenty three functions set was used. This benchmark functions was proposed by Yao et al. (1999); for now on we refer to this test as Yao's test. In Table 1.1, the equation of each function, the range of the search space, the number of dimensions (N) and the optimal value for each function are presented. This set of functions has several characteristics, unimodal, multimodal, high dimensions, low dimensions; which provides a lot information about the metaheuristic performance. Those benchmark functions could classify as: the first four functions and 6^{th} one are high dimensional and unimodal functions. The 5^{th} is multimodal for a high number of dimensions (Shang & Qiu, 2006). Seventh is a noisy function. The functions 8^{th} to 13^{th} are high dimensional and multimodal functions and remaining functions are low dimensional and multimodal functions. Yao's test has functions with well-known features that allow a better understanding of the behavior of any metaheuristic and its parameters.

However, the functions mentioned above are not hard enough and there are another functions that could be interesting to prove the real potential of the proposed algorithm. Then, in the second part, we test the supernova with 25 extra functions from CEC's special session for real parameter celebrated in 2005 (Suganthan, Hansen, Liang, Deb, Chen, Auger, & Tiwari, 2005); for now on we refer to this test as CEC's test. Each function was run for 10 dimensions. This functions set has the first five functions are shifted unimodal functions. The next seven functions (6^{th} to 12^{th}) are shifted multimodal functions. 13^{th} and 14^{th} are extended function. The other functions are hybrid functions.

Algorithms descriptions and parameters used for the test

The result was compared with DE and PSO. Both are classical metaheuristics developed for nonlinear optimization. The performances of both algorithms are well-known and widely used. Comparison can show supernova's strengths and weakness and helps to improve the algorithm.

DE is an evolutionary algorithm; the main general steps for evolutionary metaheuristics are used in this method: initialization, mutation, crossover and selection and has two basic parameters F and CR called amplification factor and crossover constant. In this case, we compare with a traditional version called "Rand/1/exp" (Price, 1996). This version has not self adapted parameters for the initial test. Parameters reported for both benchmark sets were $F = 0.5$ and $CR = 0.9$. The stop criterion for Yao's test was maximum iterations, for CEC's 2005 was maximum iterations

Table 1.1: Benchmark function set

f	(N)	limits	minimum
$f_1(x) = \sum_{i=1}^{N} x_i^2$	30	$[-500, 500]^N$	0
$f_2(x) = \sum_{i=1}^{N} \lvert x_i \rvert + \prod_{i=1}^{N} \lvert x_i \rvert$	30	$[-10, 10]^N$	0
$f_3(x) = \sum_{i=1}^{N} (\sum_{j=1}^{i} x_j)^2$	30	$[-100, 100]^N$	0
$f_4(x) = \max_i \{\lvert x_i \rvert, 1 \leq i \leq 30\}$	30	$[-100, 100]^N$	0
$f_5(x) = \sum_{i=1}^{N-1} [100(x_{i+1})] x_i^2$	30	$[-100, 100]^N$	0
$f_6(x) = \sum_{i=1}^{N} (\lfloor x_i + 0.5 \rfloor)^2$	30	$[-100, 100]^N$	0
$f_7(x) = \sum_{i=1}^{N} i(x_i^4) + random[0, 1)$	30	$[-1.28, 1.28]^N$	0
$f_8(x) = \sum_{i=1}^{N} [x_i \sin(\sqrt{\lvert x_i \rvert})]$	30	$[-500, 500]^N$	-12569,5
$f_9(x) = \sum_{i=1}^{N-1} [x_i^2 - 10\cos(2\pi x_i + 10)]$	30	$[-5.12, 5.12]^N$	0
$f_{10}(x) = -20 \cdot exp(-0.2\sqrt{\frac{1}{n} \cdot \sum_{i=1}^{n} x_i^2}) - exp(\frac{1}{n} \cdot \sum_{i=1}^{n} \cos(2\pi x_i)) + 20 + exp(1)$	30	$[-32, 32]^N$	0
$f_{11}(x) = \frac{1}{4000} \sum_{i=1}^{N} x_i^2 - \prod_{i=1}^{N} \cos(\frac{x_i}{\sqrt{i}}) + 1$	30	$[-600, 600]^N$	0
$f_{12}(x) = \frac{\pi}{N} \{10 \sin^2(\pi y_1) + \sum_{i=1}^{N-1} (y_i - 1)^2 \times [1 + \sin^2(3\pi y_{i+1})]$ $+(y_N - 1)^2\} + \sum_{i=1}^{N} u(x_i, 5, 100, 4)$	30	$[-50, 50]^N$	0
$f_{13}(x) = 0.1\{10 \sin^2(\pi 3x_1) + \sum_{i=1}^{N-1} (y_i - 1)^2 [1 + 10 \sin^2(\pi y_i + 1)]$ $+(y_N - 1)^2\} + \sum_{i=1}^{N} u(x_i, 10, 100, 4)$ para las funciones f_{12} y f_{13}	30	$[-50, 50]^N$	0
$u(x_i, a, k, m) = \left\{ \begin{array}{ccc} k(x_i - a)^m, & & x_i > a \\ 0, & & -a \leq x_i \leq a \\ k(-x_i - a)^m, & & x_i < -a \end{array} \right\} y_i = 1 + \frac{1}{4}(x_i + 1)$			
$f_{14}(x) = [\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{N} (x_i - a_{i,j})^6}]$	2	$[-65.5, 65.5]^N$	≈ 1
$f_{15}(x) = \sum_{i=1}^{11} \left[a_i - \frac{x_1 (b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4} \right]^2$	4	$[-5, 5]^N$	$\approx 3.075 \times 10^{-4}$
$f_{16}(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1 x_2 - 4x_2^2 + 4x_2^4$	2	$[-5, 5]^N$	1.0316285
$f_{17}(x) = \left(x_2 - \frac{5.1}{4\pi^2} x_1^2 + \frac{5}{\pi} x_1 - 6 \right)^2 + 10\left(1 - \frac{1}{8\pi}\right) \cos x_1 + 10$	2	$[-5, 10][0, 15]$	0.398
$f_{18}(x) = \left[1 + (x_1 + x_2 + 1)^2 \left(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1 x_2 + 3x_2^2\right) \right]$ $\times \left[30 + (2x_1 - 3x_2)^2 \left(18 - 32x_1 + 12x_1^2 + 48x_2 - 30x_1 x_2 + 27x_2^2\right) \right]$	2	$[-2, 2]^N$	3
$f_{19}(x) = -\sum_{i=1}^{4} c_i \exp\left[-\sum_{j=1}^{3} a_{ij} (x_j - p_{ij})^2 \right]$	3	$[0, 1]^N$	-3.86
$f_{20}(x) = -\sum_{i=1}^{4} c_i \exp\left[-\sum_{j=1}^{6} a_{ij} (x_j - p_{ij})^2 \right]$	6	$[-100, 100]^N$	-3.32
$f_{21}(x) = \sum_{i=1}^{5} \left[(x - a_i)(x - a_i)^T + c_i \right]^{-1}$	4	$[0, 10]^N$	-10
$f_{22}(x) = \sum_{i=1}^{7} \left[(x - a_i)(x - a_i)^T + c_i \right]^{-1}$	4	$[0, 10]^N$	-10
$f_{23}(x) = \sum_{i=1}^{9} \left[(x - a_i)(x - a_i)^T + c_i \right]^{-1}$	4	$[0, 10]^N$	-10

and a minimum error 10^{-9}. The numerical results for comparison are available at (Brest, Greiner, Boskovic, Mernik, & Zumer, 2006; Derrac, García, Molina, & Herrera, 2011).

The version of PSO used to compare the algorithm is the one proposed by Kennedy and Eberhart (Mirjalili & Hashim, 2010), inspired in swarm behavior. "Follow the leader" is the main idea. PSO uses velocity, leadership to guide the search; the individuals are going behind the best individual. The parameters reported for Yao's test were $c_1 = 2$ and $c_2 = 2$ while for the set of CEC's 2005, the parameters were $c_1 = 2.8$ and $c_2 = 1.3$. The stop criterion for the initial benchmark function set was a number of iterations. A second one was a number of iterations to achieve a error of 10^{-9}. The

numerical results for comparison are available at (Mirjalili & Hashim, 2010; Derrac et al., 2011). Finally, parameters for supernova were $G = F = 3$, $\lambda_0 =$ "the limits of search space", $\alpha = 1.2$ and $T = 100$ the stop criterion was a number of restarts for both benchmark function sets. In the second set, a limit for error was added.

For both function sets, we summarize 50 independent runs. In the first set, we report the mean $\overline{f^*}$ and the standard deviation (σ) for the fifty runs and the number of function calls (FO) (see Table 1.2). FO has not changed for each function in the different independent runs. In the second test, we report mean $\overline{f^*}$ for each algorithm to compare them statistical (see Table 1.3).

Statistical tests are necessary to compare metaheuristics. Nonparametric tests are recommended (Derrac et al., 2011). Initially; we compare the means for the three algorithm as a unique set using Friedman's test. The hypotheses are $H_0 : \overline{f^*}_{snova} = \overline{f^*}_{DE} = \overline{f^*}_{PSO}$ and $H_a : \overline{f^*}_i \neq \overline{f^*}_{i'}$ for at least one of the algorithms. For both tests, the rank for ties was calculated as the average.

In Table 1.4, the first three columns show the Friedman ranks, the following two columns present F_F Friedman statistical value and the p-value, which are more conservative statistical values. In the next columns F_{ID} statistical value proposed by Iman y Davenport (Derrac et al., 2011); this statistical value accepts higher differences, which is desired in this case. In the Friedman statistical, in both cases, H_0 is rejected while F_{ID} there is a significant difference only for Yao's test. This means that there is a performance difference between algorithms for Yao's test, and is similar for CEC's test.

So far, we know that there is a difference between performances. However, there is not statistical evidence to confirm which one is better or similar to another. Wilcoxon's test is a nonparametric pair-wise test that allows to know if one algorithm is significantly better than another. Wilcoxon's rank for the test is calculated. The ranges for the Wilcoxon's test are calculated by subtracting both means of the algorithms; in this case, it was always in the following order: Supernova-the algorithm to compare. Thus, for negative ranges, supernova is better than the other algorithm, because the supernova algorithm gives a smaller value than the other algorithm. Besides, we take an one-tailed test, because we want to know which one is strictly better (Sheskin, 2003; Derrac et al., 2011).

In Table 1.5, rows are for different tests, Yao's and CEC's. Columns are for pair the first are for supernova Vs. DE and the second for supernova Vs. PSO. For each pair we can show positive ranks negative ranks and the limit for a significant level of $\alpha = 0.05$; these values are taken from table A.5 (Sheskin, 2003). The ranks are calculated using the means presented in Table 1.3. The Hypotheses for the test were $H_0 : \overline{f^*}_{snova} = \overline{f^*}_i$ and $H_a : \overline{f^*}_{snova} < \overline{f^*}_i$ where i is DE or PSO. The limit present in Table 1.5 is the maximum value for min (R+,R-), if the minimum is greater than this value, we can not reject H_0. According to with the results, we can conclude that there is not a significant difference between supernova and DE. And supernova is significant better than PSO.

1.2 Comparison with other heuristics based on gravitational force

Supernova was inspired on star crafts; BBBC was based on big-bang big-crunch theory. GLSA, GSA and CFO methodologies were developed from the idea of gravitational fields and kinetic in movement of the particles. In all these approaches, gravity law is the fundamental key of the search. That foundation gives to the methods some general advantages. Initially, the search made a fast mapping of the function to focus in the important areas found in the first iterations of the algorithm. The heuristics are sensitive to function's slope. Therefore, a target search, as in a gradient method, exploits the region. Each approach has its own strategy that emphasized these or other particular

Table 1.2: Results and comparison for benchmark functions, Supernova, DE and PSO.

f	Supernova				$F.O$	DE		$F.O$	PSO	
	$F.O$	$\overline{f*}$	σ	best		$\overline{f*}$	σ		$\overline{f*}$	best
f_1	972	6.3×10^{-29}	$1.0x10^{-28}$	$1.8x10^{-33}$	1500	$8.2 \times 10{-}14$	5.9×10^{-14}	1000	2.83×10^{-4}	8.7×10^{-6}
f_2	1812	1.1×10^{-26}	1.1×10^{-26}	9.28×10^{-28}	2000	1.5×10^{-9}	$9.9 \times 10{-}10$	1000	$5.5\times^{-3}$	7.1×10^{-6}
f_3	852	1.0×10^{-13}	8.8×10^{-14}	5.15×10^{-15}	5000	$6.8 \times 10{-}11$	$7.4 \times 10{-}11$	1000	5.19×10^{3}	1.91×10^{-3}
f_4	1212	6.1×10^{-35}	1.7×10^{-34}	2.2×10^{-36}	5000	0	0	1000	$4.4 \times 10{-}7$	$2.4 \times 10{-}8$
f_5	510	28.89	$2.5 \times 10{-}2$	27.98	20000	0	0	1000	201.665	15.5933
f_6	677	0	0	0	1500	0	0	1000	4.9	4.51
f_7	552	3.2×10^{-3}	1.9×10^{-3}	6.9×10^{-4}	3000	4.63×10^{-3}	1.2×10^{-3}	1000	2.6×10^{-1}	1.1×10^{-1}
f_8	612	-7.4×10^{15}	4.28×10^{15}	-6.28×10^{16}	9000	-11080.1	574.7	1000	-5909.5	-7802.34
f_9	612	0	0	0	5000	69.2	38.8	1000	7.3×10^{1}	5.6×10^{1}
f_{10}	972	4.7×10^{-15}	2.3×10^{-15}	8.18×10^{-16}	1500	9.7×10^{-8}	4.2×10^{-8}	1000	$4.9 \times 10{-}10$	2.5×10^{-12}
f_{11}	612	0	0	0	2000	0	0	1000	5.4×10^{-3}	9.9×10^{-7}
f_{12}	100	2.9×10^{-2}	2.5×10^{-2}	1.5×10^{-3}	1500	7.9×10^{-15}	8.0×10^{-15}	1000	2.3×10^{-0}	1.1×10^{-1}
f_{13}	250	1.09	0.55	0.18	1500	5.1×10^{-14}	4.8×10^{-14}	1000	8.9×10^{-2}	1.1×10^{-2}
f_{14}	100	1.39	1.2	0.998	100	0.998004	$3.3 \times 10{-}16$	1000	0.998	0.998
f_{15}	404	4.4×10^{-4}	4.9×10^{-5}	3.3×10^{-5}	4000	4.5×10^{-4}	3.3×10^{-4}	1000	$1.04 \times 10{-}3$	9.8×10^{-4}
f_{16}	204	-1.0315	$1.36 \times 10{-}4$	-1.0316	100	-1.03163	3.1×10^{-13}	1000	-1.0316	-1.0316
f_{17}	204	0.39884	1.2×10^{-3}	0.39788	100	0.397887	9.9×10^{-9}	1000	39.79	39.79
f_{18}	204	3.0024	2.9×10^{-3}	3	100	3	2.0×10^{-15}	1000	3	3
f_{19}	50	-3.862	7.4×10^{-5}	-3.8619	*N.A.*	*N.A.*	*N.A.*	1000	-3.8628	-3.8628
f_{20}	60	-3.26	$3.68 \times 10{-}2$	-3.3105	*N.A.*	*N.A.*	*N.A.*	1000	-1.60	-2.9587
f_{21}	510	-5.914	3.59	-8.354	100	-10.1532	2.5×10^{-6}	1000	-6.5752	-10.1532
f_{22}	510	-6.041	3.72	-9.766	100	-10.4029	3.9×10^{-7}	1000	-8.05607	-10.4029
f_{23}	510	-6.78	3.14	-9.879	100	-10.5364	1.9×10^{-7}	1000	-7.33602	-10.5364

Table 1.3: Results and comparison for benchmark functions, CEC 2005, between Supernova, DE and PSO. 10 dimension and 50 independent runs

f	Error Supernova	Error DE	Error PSO
f_1	2.49×10^{-06}	$\mathbf{8.26 \times 10^{-09}}$	1.23×10^{-04}
f_2	4.41×10^{-04}	$\mathbf{8.18 \times 10^{-09}}$	2.60×10^{-02}
f_3	5.27×10^{05}	$\mathbf{9.94 \times 10^{00}}$	5.17×10^{04}
f_4	6.58×10^{-01}	$\mathbf{8.35 \times 10^{-09}}$	2.49×10^{00}
f_5	2.39×10^{01}	$\mathbf{8.51 \times 10^{-09}}$	4.10×10^{02}
f_6	3.77×10^{01}	$\mathbf{8.39 \times 10^{-09}}$	7.31×10^{02}
f_7	2.46×10^{02}	1.27×10^{03}	$\mathbf{2.68 \times 10^{01}}$
f_8	$\mathbf{2.03 \times 10^{01}}$	2.04×10^{01}	2.04×10^{01}
f_9	1.38×10^{01}	$\mathbf{8.15 \times 10^{-09}}$	1.44×10^{01}
f_{10}	2.28×10^{01}	$\mathbf{1.12 \times 10^{01}}$	1.40×10^{01}
f_{11}	7.99×10^{00}	$\mathbf{2.07 \times 10^{00}}$	5.59×10^{00}
f_{12}	$\mathbf{3.25 \times 10^{01}}$	6.31×10^{01}	6.36×10^{02}
f_{13}	$\mathbf{1.30 \times 10^{00}}$	6.40×10^{01}	1.50×10^{00}
f_{14}	3.71×10^{00}	$\mathbf{3.16 \times 10^{00}}$	3.30×10^{00}
f_{15}	3.40×10^{02}	$\mathbf{2.94 \times 10^{02}}$	3.40×10^{02}
f_{16}	$\mathbf{7.45 \times 10^{01}}$	1.13×10^{02}	1.33×10^{02}
f_{17}	2.49×10^{02}	$\mathbf{1.31 \times 10^{02}}$	1.50×10^{02}
f_{18}	8.96×10^{02}	$\mathbf{4.48 \times 10^{02}}$	8.51×10^{02}
f_{19}	8.92×10^{02}	$\mathbf{4.34 \times 10^{02}}$	8.50×10^{02}
f_{20}	8.83×10^{02}	$\mathbf{4.19 \times 10^{02}}$	8.51×10^{02}
f_{21}	$\mathbf{1.55 \times 10^{01}}$	5.42×10^{02}	9.14×10^{02}
f_{22}	$\mathbf{7.53 \times 10^{02}}$	7.72×10^{02}	8.07×10^{02}
f_{23}	$\mathbf{2.35 \times 10^{01}}$	5.82×10^{02}	1.03×10^{03}
f_{24}	4.43×10^{02}	$\mathbf{2.02 \times 10^{02}}$	4.12×10^{02}
f_{25}	4.49×10^{02}	1.74×10^{03}	$\mathbf{5.10 \times 10^{02}}$

Table 1.4: Results for Friedman test for Supernova, DE and PSO

	Supernova	DE	PSO	F_F	P-value	F_{ID}	P-value
Rank Yao's test	1.833	1.595	2.667	21.357	4.89×10^{-07}	20,69	6.76×10^{-07}
Rank CEC's test	2,060	1,500	2.440	11.2	1.03×10^{-4}	0.576	0.566

Table 1.5: Results for Wilcoxon's test for Supernova, DE and PSO

Benchmark functions	snova-DE			snova-PSO		
	R +	R-	Limit for $\alpha = 0.05$ and one-tailed	R+	R-	Limit for $\alpha = 0.05$ and one-tailed
Yao's test	100.5	52.5	41	36.5	194.5	67
CEC's test	220	105	100	89	211	91

Table 1.6: Comparison among algorithms based on gravity

Feature	Algorithm				
	GLSA	BBBC	GSA	CFO	Supernova
Type of search	Local	Global	Global	Global	Global
Type of variables	Real	Integer	Real	Real	Real
Generation of the initial particles	Random	Random	Random	Deterministic	One point or Random points
Mass	Objective function	Objective function	Objective function	Objective function	Scaled objective function
Strategy to find the next particles	Gravity field For 3 points	Mass center	Gravity Interaction between random particles	Gravity between deterministic interaction particles	Gravity and impulse interaction between particles

advantages (Erol & Eksin, 2006; Webster & Bernhard, 2003; Rashedi et al., 2009; Formato, 2007). All this algorithms present similarities and differences, that we describe in this section. The comparison be done in two parts, the principals parts of metaheuristic (objective function, population, parameters, among others) and step by step for the basic algorithm of metaheuristic from Figure 1.1.

In Table 1.6 the main characteristics are described, providing a clear idea of similarities and divergences of studied methods. In the first row, the type or search refers the scope of search, local or global. The second row refers to the type of variables used for defining the optimization problem: continuous, discrete or mixed. In the third row, each method, as Supernova, has an initial set of particles. This set can be found randomly o deterministically. All methods have the analogy with mass; the fourth row shows that mass is the goal function or its scaled. Each one has its own particular strategy to calculate the next points to evaluate; the fifth row shows how the search path for the different methods is found.

As all heuristic, the ones inspired by gravitational force uses parameters to control the search. The size of particles' set (K) and number of iteration (T) are the common parameters. There are other parameters to control the force of attraction, the focus of search or regulation for random distributions, and they are different in each case. The differences in initial population objective, function evaluation and selection are associated with the part and parameters, but the biggest

Table 1.7: Comparison of new population functions among algorithms based on gravity

Metaheuristic	New population function
BBBC	$\mathbf{s}_i = \mathbf{x}_C + \text{Rand}\,(\mathbf{x}_C - \lambda, \mathbf{x}_C + \lambda)$, where $\mathbf{x}_C$ is the mass center
CFO	$\mathbf{s}_i = \mathbf{x}_i + \nu_i \mathbf{q} + \frac{1}{2}\mathbf{q}^2 \sum_{\substack{i=1 \\ i \neq j}}^{K} U\,(m_j - m_i)^{1+\alpha} + (r_{ij}\mathbf{d}_{ij})\,\lvert r_{ij}\rvert^{-\beta}$, where α and β are parameters of the algorithm
GSA	$\mathbf{s}_i = \mathbf{x}_i + \nu_i \mathbf{q} + G\mathbf{q}^2 \sum_{\substack{i=1 \\ i \neq j}}^{K} \frac{\left(\frac{m_i}{\sum_{i=1}^{K} m_j}\right)^2 r_{ij}\mathbf{d}_{ij}}{\frac{m_i}{\sum_{i=1}^{K} m_j}\,(r_{ij} + \epsilon)}$, where G and ϵ are parameters of the algorithm
PSO	$\mathbf{s}_i = \mathbf{x}_i + \mathbf{v}_i\omega_i + c_1 q\,(\mathbf{p}_i - \mathbf{x}_i) + c_2 q\,(\mathbf{g} - \mathbf{x}_i)$, where c_1, c_2 andω_i are parameter of the algorithm

difference is focused on operation to found the new population. GLSA is a local algorithm and comparison . In Table 1.7, the function of new population for each algorithm is shown. For each one of them we used the notation of the the proposed algorithm. Each approach has its own idea of gravity, the parameters and calculated values, by the by, in all methods based on gravity, the function slope is found, like in a regular gradient. Only, gravitational heuristics use the mass idea instead of derivate. As well as gradient methods, gravitational heuristics are not successful in total planar regions, but find best minimums for descent functions.

1.3 Discussion

From the previous work, the results obtained for original version of supernova are promising. Of course, this version needs to be improved, increasing their strengths. The future work presented for master degree dissertation was divided in four main parts as follow:

- Improve the performances: computationally and quality for the 40% of unsuccessful test functions present.
- Automatic chosen and control of the parameters.
- To formalize the method presented.
- Extend the metaheuristic to constrained, combinatorial and multiobjective optimization.

Each item of this list is ambitious; therefore, we focus this discussion and the next part of this work in the first item: to improve the performance and particularly to improve quality in terms of minimizing the error of the 40% of unsuccessful functions ($f_5, f_8, f_{12}, f_{13}, f_{14}, f_{16}, f_{17}, f_{18}, f_{21}, f_{22}, f_{23}$).

When does supernova fail?

Initially, we analyzed the problems of each problematic function to determinate if there was a common problem for all these functions. For all problematic function, except for f_8, we just did not achieve the minima. Function f_8 is a no-convex, multimodal and separable function; for the specific region, defined for this function, the minimum is prefect determinate, but outside of this region there are better minima points. In this case, supernova founds a better minima, but outside of the region. For this particular case, we penalized the function but the result was not good enough (Coello, 2002). So, we take in account f_8 as constrainted problem.

Mezura-Montes, Velázquez-Reyes, and Coello Coello (2006) classified the functions as separable or non-separable and unimodal or multimodal. In agreement with that classification, the 23 benchmark functions were categorized in the Table 1.8 (Karaboga & Akay, 2009; Mezura-Montes et al., 2006; Shang & Qiu, 2006). There is not a unique classification that gathers all unsuccessful functions. So, we have not a solid base to classify the problem with these mathematical characteristics.

Table 1.8: Classification of the test functions.

Benchmark function proposed by Yao				
Type of function	Separable		Non-separable	
	Successful	Unsuccessful	Successful	Unsuccessful
Unimodal	f_1, f_4, f_6, f_7		f_2, f_3	f_5(two dimensions)
Multimodal	f_8, f_9	f_{14}, f_{17}	f_{10},f_{11},f_{15},f_{19},f_{20}	$f_5$extended, f_{12},f_{13}, f_{16},f_{18},f_{21},f_{22},f_{23}
Benchmark function from CEC 2005				
Type of function	Separable		Non-separable	
	Successful	Unsuccessful	Successful	Unsuccessful
Unimodal		f_1	f_{15}	f_9
Multimodal		f_2, f_3, f_4, f_5	f_8,f_{10},f_{11},f_{12},f_{13},f_{14}, f_{16},f_{21},f_{22},f_{23}	f_6,f_7, f_{17},f_{18},f_{19},f_{20}, f_{24},f_{25}

Although in Table 1.8, we group the test functions following the proposal of Mezura-Montes et al. (2006), there are other possible classifications based on the geometry of the landscape of the functions. At this point, we found that the classification based on landscape geometry is very difficult due to the impossibility of describe such features, as planar regions or needle minima, based on the mathematical terms in the function. For example, the derivate is commonly used to describe the function's topology, but in many cases, it is not available or it is very difficult to calculate. In agreement with the descriptions given for the functions, we can conclude that: Planar regions and penalties ($f_5, f_{12}, f_{13}, f_{16}, f_{17}, f_{18}$) and Needle minimums ($f_5, f_{14}, f_{21}, f_{22}, f_{23}$) are problematic topologies for Supernova. When we arrive to this point the question was: why the planar region and needle minimums are a big challenge for supernova?

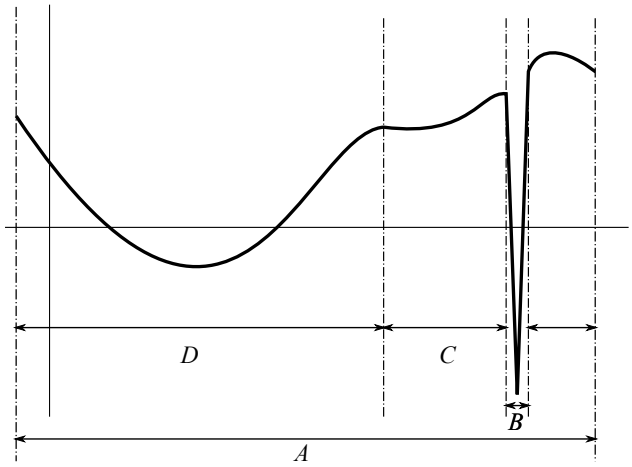


Figure 1.4: Search Area A for a objective function $f(\mathbf{x})$

Why does supernova fail in needles minimums and planar regions?

Before to explain why supernova fails in the regions mentioned above, some basic concepts of global optimization are defined.

Definition 1.3.1. A set A is called a search region if $A = \{\mathbf{x} = (x_1, \ldots, x_n) \, | l_1 \leq x_1 \leq u_1, \ldots, l_n \leq x_n \leq u_n\}$ where l_i and u_i, $i = 1, \ldots, n$, are the lower and upper bounds of the variable i (Bäck, 1996).

Definition 1.3.2. A local minimum for the function $f : \mathbb{R}^n \to \mathbb{R}$ in the subregion $N \subseteq A$, with $N \neq \emptyset$, is defined as $f^* : = (\mathbf{x}^*) > -\infty$, such that $f(\mathbf{x}^*) \leq f(\mathbf{x})$, $\forall \mathbf{x} \in N$ and $\mathbf{x} \neq \mathbf{x}^*$ (Torn & Zilinskas, 1989). For example, the Figure 1.4 has three local minimums for the subregions B, C y D.

Definition 1.3.3. A global minimum for the function $f : \mathbb{R}^n \to \mathbb{R}$ in the region A, with $A \neq \emptyset$ is defined as $f^* : = (\mathbf{x}^*) > -\infty$, such that $f(\mathbf{x}^*) \leq f(\mathbf{x})$, $\forall \mathbf{x} \in A$ and $\mathbf{x} \neq \mathbf{x}^*$ (Bäck, 1996).

Definition 1.3.4. A local search algorithm is an iterative procedure that searches the point $\mathbf{x}_k$in the neighborhood of $\mathbf{x}_{(k-1)}$ such that $f(\mathbf{x}_k) \leq f\left(\mathbf{x}_{(k-1)}\right)$. The sequence of points $\{\mathbf{x}_k\}$ for $k = 0$ to $k = \infty$ is called trajectory of points (Mohd, 2000).

Definition 1.3.5. The attracting region L for the point $\mathbf{x}^*$ where occurs a local minimum for the function $f : \mathbb{R}^n \to \mathbb{R}$ is defined as the region surrounding $\mathbf{x}^*$ such that $\forall \mathbf{x}_0 \in L$, chosen randomly, a local search algorithm starting at $\mathbf{x}_0$ arrives to $\mathbf{x}^*$ in a finite number of iterations (Mohd, 2000; Torn & Zilinskas, 1989; Hendrix & Toth, 2010). In other words, L is the region that provides information about the location of $\mathbf{x}^*$. (See Figure 1.4 region B)

Definition 1.3.6. The point $\mathbf{x}^*$ is a strong minimum of $f(\cdot)$ when $f(\mathbf{x}^*) < f(\mathbf{x})$, $\forall \mathbf{x} \in L$ (Yang, 2010). In other words, there is only one minimum point in the attraction region.

Definition 1.3.7. The point $\mathbf{x}^*$ is a weak minimum when $\mathbf{x}^*$ is not a strong minimum (Yang, 2010). In other words, there is more than one minimum point in the attraction region L.

Needle minimums

A needle minimum is a difficult minimum to reach; often, it is defined as an abrupt change of the behavior of the function in a small attraction region e.g., Attraction region B In Figure 1.4. Usually, it is a strong minimum, and the slopes of the attraction region are big and have opposite sign; although most of the time, the particular point has not a derivate.

Attraction region for the minimum in B gives to Supernova information to follow the search like hill climbing methods. For K random points distributed uniformly over a specific area A, the probability to find a specific point or area B as $B|A$ in this case. The probability $P(x|B) = \frac{KB}{A}$. It means that, for a small B the probability to find the a specific point of B is low, regularly near to 0.

Planar regions

A planar region of a function is an area of weak minimum or minimums. This means that this is a smooth function, with slopes near to 0, where the attraction region of the minimum has little information about the location of the minimum. In this case, all methods would found the attraction region, but the search process stops,but there is not convergence of the algorithm because the information about descendent direction is poor. If we think about classical methods based on gradients, if a gradient is zero or near to it; then, the algorithm has not direction to follow. Supernova has similarities with hill climbing and gradient methods because all of them use the information of the descendent slope like a drop of the water that needs the slope to flow; when the slope information is not available, the drop will be stagnated in one point like the algorithm actually does.

How the other metaheuristics have solved the problem with needle minimums and planar regions

In the literature, the problem of planar regions is not considering in isolation. Usually, metaheuristics is improved in general. For global optimization often calling globalization or escaping of local minimums (Bäck, 1996; Dasgupta, Das, Abraham, Member, & Biswas, 2009; Feoktistov, 2006b; Glover, Kelly, & Laguna, 1995b; Hinterding, Michalewicz, & Eiben, 1997; Hirsch, Meneses, Pardalos, & Resende, 2006; Pant & Et, 2011; Yao et al., 1999). There are two kind of strategies to improve the globalization: internals and externals. The internal strategies are parameters adaptations, better exploration, among others; e.g, 1/5 mutation rule or evolutionary programming made faster (Bäck, 1996; Hinterding et al., 1997; Yao et al., 1999); external strategies are hybrids, hyper heuristics among others; e.g, DE-particle swarm optimization(PSO), C-GRAPS.

There are not particular strategies for particular characteristic because real-world problems have not specific characteristics defined before the starting, like planar regions or needle minimums (Bäck, 1996; Torn & Zilinskas, 1989). Even though, there are some evidences and tries, for optimization and other fields like dynamics systems stability, that show that if search has indicators of the topology for the function then it is redirected to a better regions (Matallana, Blanco, & Bandoni, 2010; Mohd, 2000).

Although, there are evidences of the improvement of the other metaheuristics, it is important to remind that: there are not guarantee of improvement of the performance of Supernova or any other metaheuristic by the direct use of a global strategy, whether it is internal or external. For each particular case is necessary to analyze the possible behavior of the original metaheuristic.

Afterwards, the performance of the global strategy is implemented and validated. Also, there are not specific modifications for the problem of planar regions and needle minimum (Bäck, 1996; Talbi, 2009; Torn & Zilinskas, 1989; Yang, 2010; Macready & Wolpert, 1996).

1.4 Hypothesis

It is possible to find a criterion to identify the planar regions and adapt the parameters and/or strategies of the search improving the performance of supernova for planar regions without impairing the current performance for another topologies.

1.5 Objectives

1.5.1 General objective

To design and to validate a new version of Supernova based on the development of a new strategy specialized in planar regions, and a mechanism for switching between specialized algorithms for descendent and planar regions, such that there is an improvement on the behavior for planar regions without degenerating the current behavior for descendent regions.

1.5.2 Particular objectives

- To choose, modify or design a strategy for improving the search in planar regions.
- To define a criterion for determining when the function have a planar region.
- To design a process for tuning, control and change parameters and strategies for functions planar regions reminding the current for descendent functions.
- To choose or generate a new set of benchmark functions with planar regions for testing behavior of supernovae.
- To validate the behavior of new version of Supernova.

1.6 Contributions

This research contribute:

- Mesa, E., Velásquez J. D. and Jaramillo, P. Supernova utilizando secuencias de baja discrepancia, Congreso latinoamericano de investigación de operaciones CLAIO 2012, Rio de Janerio, septiembre 2012.
- Mesa, E., Velásquez, J. D. and Jaramillo, P. (2014). Nonlinear optimization in landscapes with planar regions. In Terrazas, G., Otero, F. E. B., and Masegosa, A. D. (Eds.), Nature Inspired Co-operative Strategies for Optimization (NICSO 2013), Vol. 512 of Studies in Computational Intelligence, pp. 203-215. Springer International Publishing.
- Mesa, E.; Velásquez, J. D. and Jaramillo, P. (2014). A new self-adaptive PSO based on the identification of planar regions, Evolutionary Computation (CEC), 2014 IEEE Congress on , vol., no., pp.1937,1943, 6-11 July 2014

1.7 Methodology

Agree with requirements of the Section 4.1, we have three challenges to improve supernova algorithm:

- Development a new strategy to optimize planar regions.
- Definition of a novel criterion to determine when a function has planar regions.
- Design a mechanism to switch between both strategies: the original one developed in the master's thesis and new one for planar regions.

Based on these challenges, we propose the following steps:

- The search and analyze of the strategies used for improving other metaheuristics with the same problems of supernova.
- The selection or the redesign of the original strategies used by other metaheuristics, found in the previous step, with the aim to incorporate them inside of the supernova algorithm.
- The design of a novel criterion to know when supernova is in a planar region; and the mechanism to switch between the original strategy and new strategy or strategies proposed to improve the performance.
- The implementation of the strategies, criterion and mechanism described in the steps above.
- The testing of the strategies, switching mechanism and criterion with the aim of establishing the functionality of the proposed improvement of the supernova algorithm. First each one and after together.
- The validation of the complete method.

2. OPERATIONS OF THE ORIGINAL SUPERNOVA ALGORITHM

"There are not more than five cardinal tastes, yet combinations of them yield more flavours than can ever be tasted."
–Sun Tzu, The Art of War

Metaheuristics have internal parts called operations, e.g. Genetic algorithms uses three different operations:crossover, mutation and selection. In other words, an operation is a basic unity with a main objective in the strategy of the metaheuristic. Each metaheuristic has their own operations, which make it be unique, but they are not always formalized for all metaheuristics (Bäck, 1996; Talbi, 2009; Angeline, 1995).

The most known and analyzed operations are, of course, operations for evolutionary metaheuristics paradigm. The operations for this metaheuristics are: Mutation, crossover and selections. For each algorithm, each operation, has the same principal concept, but was developed and implemented in different ways (Bäck, 1996). For example, crossover in genetic algorithms includes the combination of distinct parts of the genes chain of an individual while in differential evolution, the crossover and mutation are non-separable operations that combines randomly the components of one individual between some individuals (Liu, Mernik, Hrnčič, & Črepinšek, 2013; Mezura-Montes et al., 2006). Moreover, for same metaheuristics exist versions with have minimal changes inside operations to get different targets.

The first heuristics and metaheuristics were developed in the 60s, but only until 80s, operations of the metaheuristics become a study issue. The formalization of operations is important because provides a common language that let compare and share the new develops (Yang, 2010). Supernova is a metaheuristics on development, the concept proposed was tested and great results were found. In this chapter, we propose to divide the metaheuristic in two new operations and the traditional selection from evolutionary algorithms. Following we describe and analyze them theoretical.

The First section of this chapter is a background about operations for evolutionary algorithms and swarm optimization. Afterwards, in the next section, we discuss the parameters in the algorithms mentions above. In the third section, the operations of the original supernova algorithm are described and analyzed. Finally, in the last section, we present some conclusion about operations and how improved it.

2.1 Defined Operations

The evolutionary metaheuristics was inspired by the theory of evolution developed by Charles Darwin and Lamarck. The cornerstone concept to these optimization methods was the adaptation

for surviving in the environment. Genetic algorithms (GA) and evolutionary strategies (ES) were developed from a unique concept and almost simultaneously by two different research groups in United stated and Germany (Michalewicz & Schmidt, 2002; Bäck, 1996). Both metaheuristics have the same inspiration, but work differently, e.g. GA was designed for combinatorial problems while EA was designed for continuous problems. The development of this two sophisticated heuristics beginning the first paradigm of the field called evolutionary algorithms. In the 80s, operations for metaheuristics were separated and defined for formalizing the metaheuristics trying to normalize the procedures.

Under the evolutionary paradigm, the following well-known metaheuristics are included: Genetic Algorithm, Evolutionary Strategies, Evolutionary Programming and Differential Evolution. For all algorithms mentioned above, there are three operations defined: mutation, crossover and selection. Each of them came from evolutionary theory, but they are approached differently in each method and new version. In this section, we will present and describe these three operations theoretically from biology, the emulation of the concept and how they impact the algorithm. Moreover, how these operations are related with the parameters of the algorithm. Furthermore, this description is used as a framework for defining operations for another metaheuristics in the next sections of this chapter, particularly: Supernova.

There are some common words for the algorithms of evolutionary paradigm as:

- *Fitness function or quality function* is the measure of individual's adaptation. In optimization, this is related with the objective function pointing which solutions are better than others. This function could be the objective function itself or a modification (Bäck, 1996).
- *Individual*, in optimization, is a single solution for the optimization problem (Bäck, 1996).
- *Genotype* is the genetic material of the individual, in optimization is the solution vector. In this case, the genotype could be the solution itself or a translation of the solution to certain type of code. (Holland, 1992).
- *Phenotype* is a set of observable characteristic of an particular individual. In optimization, this is an attribute of the individual is the solution vector itself. While genotype can be a code, phenotype is the solution (Goldberg, 1989).

2.1.1 Mutation

In biology, mutation is a permanent modification in the DNA of the individual. Several alterations to the genetic chain are possible: small or big changes, mutation to all body cells or just some cells, etc. The metaheuristic's operations are a simplification of the natural phenomena. In this work, we will focus in hereditary mutation with phenotype changes. Theoretically, a mutation starts to be stronger in a population when the new characteristics are positive for fitting into the environment. For example, the passerine birds observed by Darwin in the Galapagos islands. Their beak's form and function were altered comparing with the new shape and of of the beak help to survive in the unknown environment. New birds with the particular beak start to birth and survive better, then these have more chances of reproduction, and the mutation is inheritance.

The algorithm is a very basic code for a mutation operation. There is a population $\mathbf{x}_{k-1}$ the operation offers a new population after mutation.

The concept of mutation is introduced to evolutionary algorithms with the aim to increase the exploration of the algorithm. The small changes in the solution let to the algorithm explore

Algorithm 2 Mutation

Require: $\mathbf{x}_{k-1}$, $\mathbf{m}$, Γ
{$\mathbf{m}$ is a vector result from a mutation function}
1: $\mathbf{s}_k \leftarrow \mathbf{x}_k + \Gamma\mathbf{m}$
2: **return** $\mathbf{s}_k$ {$\mathbf{s}_k$ is new individuals for iteration k}

new areas. Traditionally, the mutation is considered directly proportional to exploration, then the balance between exploration and exploitation is controlled by mutation level (Bäck, 1996). Controlling exploration only with mutation is a concept that is being discussed. Recently, other operations, like crossover, have been studied for knowing their influence in the exploration of the algorithm.

2.1.2 Crossover

Genetic material is replicated generation by generation, and this process is called reproduction. Biologically, It can be sexual, asexual or cloning. For metaheuristics, reproduction can be described as a combination of solutions. For example, in GA the sexual reproduction is a rule to combine two or more solutions called parents for obtaining a new solution.

Nowadays, the relation between crossover and exploration-exploitation balance is under study. The crossover between solutions from opposite zones of the search area can be an exploration strategy. Notice, there are different kinds of crossover each could be an exploration or exploitation strategy associated with the implementation and parameters (Talbi, 2009).

2.1.3 Selection

Which individuals will be the parents of the next generation? Which characteristics of current generation will have the next one? From biology focus, the nature selects the individuals by the adaptation to the environment in terms of skill to survive to reproduce. For example, the color of the beetles can be brown or green. If the a population that living in a old tree trunk, soon the brown beetles will be the majority because brown color make them invisibles to the predators. While most of the green beetles will be eaten by predators because they will be visible in their ecosystem, the old tree trunk. Thus, adaptation to survive in the trunk is better for brown individuals because brown beetles will be a live to reproduce. From metaheuristics, there is a fitness function and this is the criterion to select the better individuals to reproduce. The selection can be elitist, i.e. only the best individuals and them characteristics, agree to the fitness function, will be in the next generation. However, selection can not be absolute elitist and trying to induce exploration in certain parts of the algorithm. Usually, the selection has parameters that increases the exploration for the first iterations and reduces it at the end of the running(Bäck, 1996).

There is a kind of selection in each metaheuristics, even for non evolutionary ones. Each algorithm has criteria to determine which individual or individuals lead the search and which not. In addition, the convergence of the algorithm is related with the elitism in the selection of the algorithm. Only, elitist algorithms can guarantee the empirical convergence because elitism guarantee the best individuals each iteration and improved each iteration until arrives to an asymptote like an exponential function. i.e. any metaheuristic at least should remember the best solution ever (Villalobos-Arias, Coello, & Hernández-Lerma, 2005).

2.2 How parameters and operations are related

Ideally, parameters and operations work together to obtain the best performance of the algorithm. Parameters are allocated inside operations to modify the operation from one limit of the operation to the another. For example, Mutation from genetic algorithm try to introduce smalls changes in the genetic chain. The parameter is a random roulette to determine how many individual are muted, the extremes of the operation is no mutations at all or whole population is mutated. If the mutation parameter increase more individuals are muted while the parameter decrease less individual are muted. In Figure 2.1, we show how parameters affect population inside operation.

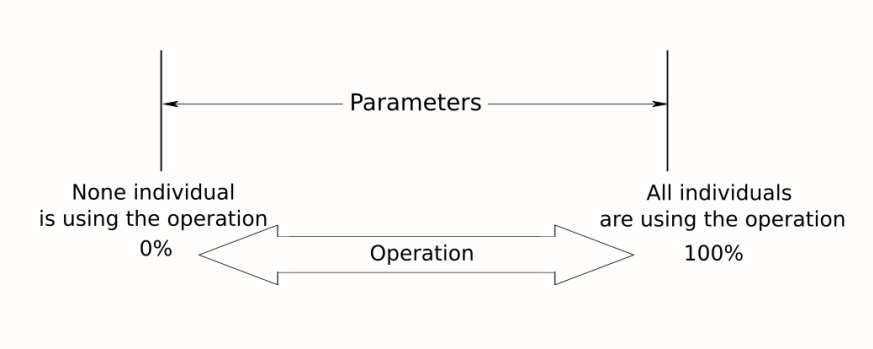


Figure 2.1: Parameter and operations

Indirectly, when parameters changes, the modifications in the search are observed in the balance between exploration and exploitation of the algorithm. There is not a direct measure to determine this balance, but theoretically there there are situation when increase exploration or exploitation is needed.

2.3 Operations in supernova

In the first chapter, the original version of the algorithm was described as a two forces strategy: gravity and impulse. Both emulate the kinetic movement of the supernova phenomenon. In this section, we divide the complete strategy in two operations: gravity and impulse, described and analyze them. Also, there is a third operation that is not clearly defined as both and it is selection. Selection as operation was described in the previous section and conceptually is the same. In this section, we will described the particular uses of this operation for Supernova.

2.3.1 Operation: Gravity

As we described in the first chapter, a simplification of the supernova phenomenon was emulated. In this simplification, gravity force and impulse are the forces chosen. Gravity is an attraction force. In the analogy, every solution has a mass which is related with objective function, i.e. the best solution attracts the nearest solutions and exploits the region. While, the impulse follows the initial direction increasing the exploration.

There are other algorithms that use the gravity force as an operation like: CFO, GSA, BBBC and other metaheuristics described in the previous chapter. Although there is not a unique and

formal definition, the main use of this operation is exploitation of the best solutions found. Each algorithm gives a particular relevance to the mass and the distance for this operation. In Table 1.6, there are the equations for new population for three methods inspired by gravitation force. GSA and CFO found an acceleration value from gravity. The original equation is modified with an exponent that is a parameter (α), i.e. for $\alpha = 0$ the acceleration or gravity influence for new population is null. The distance is indirectly proportional to the mass in CFO. While in CFO there is not a modification for the distance influence, there is for GSA. BBBC does not use the gravity equation. The attraction between particles is calculated using the moment of inertia, which is a weighted mean of the objective function value. The moment of inertia is the influence by the mass (objective function value) and the distance between points, which is conceptually right with gravity operation.

Like in the original equation of the gravity, for Supernova the gravitational interaction was the weighted sum of the vector given by currently particles and the particles from population. The individuals with the better values found to attract the others like in the real phenomenon. From algorithm 1 the equation for the operation is:

$$\mathbf{h}_k = \sum_{\substack{i=1 \\ i \neq k}}^{K} G \cdot m_k^{-1} \cdot m_i^{-1} \cdot \left[1 - r_{ik} \cdot \left(\sum_{\substack{j=1 \\ j \neq k}}^{K} r_{jk} \right)^{-1} \right] \cdot \mathbf{u}_k \tag{2.1}$$

The parameter is G. The distance and mass have the same influence, but the mass is scaled value of the objective function to avoid the negative and null masses.

Algorithm 3 gravity operation

Require: x,$f(\cdot)$, G

1: **for** $i = 1$ **to** K **do** $\mathbf{u}_{ik} \leftarrow \frac{\mathbf{x}_i - \mathbf{x}_k}{\|\mathbf{x}_i - \mathbf{x}_k\|}$ **end for**

2: $\mathbf{h}_k \leftarrow \sum_{\substack{i=1 \\ i \neq k}}^{K} G \cdot m_k^{-1} \cdot m_i^{-1} \cdot \left[1 - r_{ik} \cdot \left(\sum_{\substack{j=1 \\ j \neq k}}^{K} r_{jk} \right)^{-1} \right] \cdot \mathbf{u}_k$

3: $\mathbf{s}_k \leftarrow \mathbf{x}_k + \mathbf{h}_k$

4: **return** $\mathbf{x}_i; f(\mathbf{x}_G)$

For a single iteration, the operation algorithm is shown in 3. An initial unitary vector is calculated (See Line 1). Afterwards, a sum of whole masses is presented in Line 5. Finally, a new population is found using gravity operation. From the algorithm proposed, the complexity of this operation is $O(n^3)$.

2.3.2 Operation: Impulse

Impulse was described as a unitary vector in the direction followed by the particle i.e. the vector that join the past iteration position and currently position. From supernova phenomenon, the particles are eject from the center of the star, this trajectory is emulated for the operation impulse and modified by the other particles (gravity operation). $F * \mathbf{d}_k = F * \frac{\mathbf{p}_k - \mathbf{x}_k}{\|\mathbf{p}_k - \mathbf{x}_k\|}$ Where F is the parameter of the operation. $\mathbf{p}_k$ are the solutions from previous iteration and $\mathbf{x}_k$ are the current solutions.

As in gravity operation, the initial unitary vector is found and a new impulse if found using the

Algorithm 4 Impulse operation

Require: $\mathbf{x}$,$f(\cdot)$, F, $\mathbf{p}$

1: $\mathbf{d}_k \leftarrow \frac{\mathbf{p}_k - \mathbf{x}_k}{\|\mathbf{p}_k - \mathbf{x}_k\|}$
2: $\mathbf{i}_k \leftarrow F \cdot m_k^{-1} \cdot v_i \cdot \mathbf{d}_k$
3: $\mathbf{s}_k \leftarrow \mathbf{x}_k + \mathbf{i}_k$
4: **return** $\mathbf{x}_i$;$f(\mathbf{x}_G)$

unitary vector, mass and a velocity. The velocity is random for the algorithms presented in chapter one. From the algorithm proposed for this operation, the complexity of this operation is $O(n)$

Conceptually, impulse is following the search direction and is about increase the exploration. The impulse in the supernova strategy is clear and separable concept, but we found it immersed in another metaheuristics as well in supernova. For example, the original PSO algorithm the search direction is implicit and non separable from the velocity.

2.3.3 Operation: Selection

As we mention above, selection chooses, agree with some criteria, the best individuals of the population. In the Algorithm 1, we described it as memory that saves the best solution for each particle for previous iterations and restart. Although the concept of choosing the best solutions is preserved, the way it is used is different. In the evolutionary paradigm, the selected individuals are taken as parents while in supernova only the best individual is taken to restart the algorithm. Also, selection operation in supernova metaheuristic provides the best solution found and it provides the elitism that guarantees the asymptotic behaviour of the error. Note that, the name is the same, but it is not identical operation for Supernova and Evolurtionary algorithms. For supernova is a memory, while for GA selection is used for inheritation.

2.4 Conclusions

Supernova has three operation: Selection, impulse and gravity. Divide and define the operations gives a important framework to analyze and modify the metaheuristics. In this case, selection is a very know operation. Gravity is a operation that is not ready a formalized one, but supernova and other approaches results from another metaheuristics indicates a promising operation. The impulse is presented indirectly in other metaheuristics and can be included in other metaheuristics to increment the exploration.

3. LOW DISCREPANCY SEQUENCE START FOR SUPERNOVA

" A thing appears random only through the incompleteness of our knowledge"

–Spinoza, Ethics

Metaheuristics are direct methods. Therefore, the information found each iteration is related with the performance itself. In the minimization case, a good information is defined as information that maximizes of the probability to find the attracting region of the global minimum. The region is unknown in advance, then for increasing the probability to find the attracting region of the global minimum we need to try to cover every section for having as much information as possible. However, continuous and unconstrained problems have infinite solutions, whereas the sample or the size of the population used by population metaheuristics oscillates between forty and one hundred, hence the importance of first population distribution. Often, metaheuristic methods use random sampling to obtain the first set of solutions. This methodology, for initial sampling, is not perfectly uniform due to the large void got because of small samples (Xuan Hoai, Quang Uy, McKay, & Minh Tuan, 2007). The empty spaces can be, or not, relevant for the performance of the metaheuristic in each problem, but the irrelevance of these groupings cannot be guaranteed.

The problem described above is common for different filds as: statistic, numerical methods, among others. Some solutions have been proposed from these areas for example, stratified distribution, low discrepancy sequences and pool of solutions. These approaches have been applied successfully to metaheuristics (Brooks, 1959; Omran, al Sharhan, Salman, & Clerc, 2013; Bäck, 1996). The pool of solutions has been used for evolutionary algorithms. The most important advantage is the possibility to chose the best points from a really large sample, but it does not guarantee the distribution over the search area covers whole spaces and found the attracting region. Stratified distribution has been used for local methods reducing their iterations and evaluations to the objective function. The main problem of this method is determining a right size of the grid for the stratification. Low discrepancy sequences (LDS) are constructed for distributing the solutions uniformly improving the performance of the search as in PSO (Omran et al., 2013). The main problems with LDS are degeneration of distribution when the dimension increase and for some sequences the computational effort is also a issue (Xuan Hoai et al., 2007).

Supernova is a metaheuristic inspired by stars. The proposed algorithm has similar behaviour to other classic metaheuristics (See last chapter). Supernova follows three main steps from initial random population: selection, gravity and impulse. The initial population, as we described above, is important because it gives necessary information for guiding the search. From previous approaches mentioned, we chose LDS as a good manner to improve the methodology. LDS is called quasi-random and each sequence satisfy an inequality where discrepancy is lower that a relation between

dimensions and population size (Knuth, 1998). For uniformly random distribution (URD) , there are not limits in the discrepancy. Small and large samples estimated using URD present big holes which are prevented using LDS.

Supernova initial sampling can be changed by LDS. The main objective of this chapter is proposing a new version using LDS and analyze the effect of LDS for initial sampling in Supernova. First, we introduces a brief review about LDS and propose the appropriate changes in the algorithm for using it with LDS. Afterwards, we present a comparison of both versions (original and new one) and parameters. Finally, we conclude about this new version, the advantages and problems.

3.1 Materials and methods

The main objective of this chapter is presenting the advantages of LDS for calculating initial population and the implementation of the supernova algorithm using them. In this section, the LDS, some antecedents of LDS and the algorithms are decribed. Finally, we describe the modification to the original algorithm to adapt it for LDS.

3.1.1 Low discrepancy sequences

One of the most common uses of LDS is quasi-Monte Carlo (QMC) integration methods. These sequences improve the performance of tradicional Monte Carlo (MC) as shown in (Papageorgiou & Traub, 1997). The initial population of a metaheuristic can be compared with a small sampling of large space for numerical integration. LDS is design to distribute a uniformly spread. Then, from the construction, initial population generated by LDS gets information about whole search space, reducing the risk to ignore regions that could be exploited for the algorithm.

LDS sequences are characterized by small discrepancies. For pseudo-random distributions, the discrepancy is approximately $\frac{\log K^N}{K}$ while the discrepancy for LDS is $\frac{\log \log K^{0.5}}{K^{0.5}}$ which is lower. In other words, the distribution over the search space has less voids. Even so, there is a chance to have empty spaces with important information, but the probability of to happen with LDS is lower than with URD (Gentle, 2003; Knuth, 1998). Besides, the LDS grid is deterministic. Therefore, the independent runs are unnecessary.

A basic definition of the discrepancy between k points in n dimension is:

$$D_k^n = \sup_E \left| \frac{A(E;k)}{k} - \lambda(E) \right| \tag{3.1}$$

Where t_j is a number between 0 and 1 for each component n, $E = [0, t_1) \times \cdots \times [0, t_n)$ and $A(E;k)$ denotes the number of points that belongs to E. Additionally, λ correspondes to Lebesgue measure(Papageorgiou & Traub, 1997). Then, A LDS is the sequence where the discrepancy $D_k^n < c(n)\frac{(log(k))^n}{k}$ $\forall k > 1$ where c is a constant and depends of dimensions. Based on previous definitions, different LDS have been design (Niederreiter, 1992).

Van Der Corput sequence was the first LDS proposed. The strategy is divide the space in b intervals. Also, this b intervals will be divided in another b subintervals and so on. In table 3.1, the first ten elements of the sequence for base 2 and 3 are presented.

Halton, Faure and Sobol LDS are a generalization for higher number of dimensions of Van Der Corput sequence. The Figure 3.1 shows the distribution for each sequence for a population with 10 individuals and two dimensions. For all these sequences, computational effort increases with

Table 3.1: First ten elements for Van Der Corput sequence with two different bases

base	1	2	3	4	5	6	7	8	9	10
2	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{3}{4}$	$\frac{1}{8}$	$\frac{3}{8}$	$\frac{5}{8}$	$\frac{7}{8}$	$\frac{1}{16}$	$\frac{3}{16}$	$\frac{5}{16}$
3	$\frac{1}{3}$	$\frac{2}{3}$	$\frac{1}{9}$	$\frac{4}{9}$	$\frac{7}{9}$	$\frac{2}{9}$	$\frac{5}{9}$	$\frac{8}{9}$	$\frac{1}{27}$	$\frac{4}{27}$

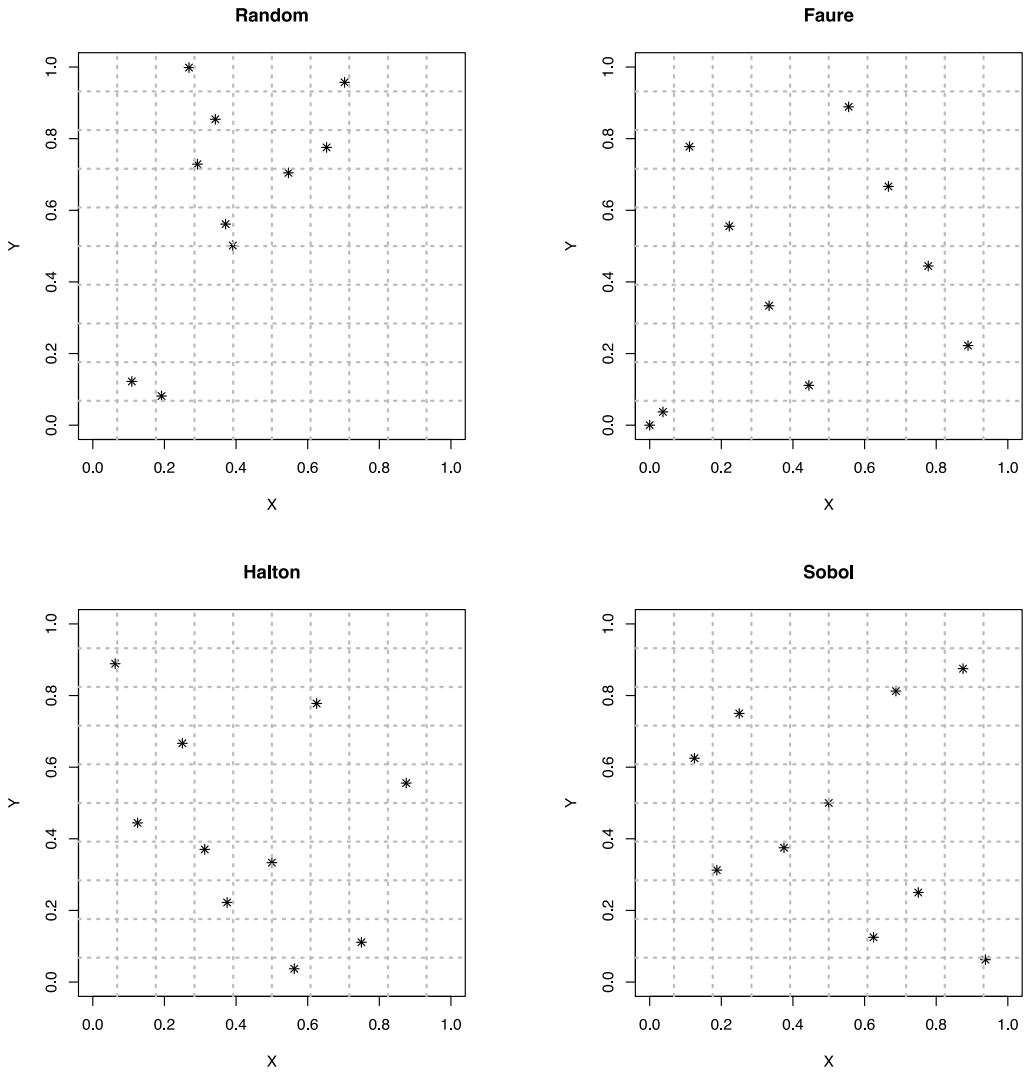


Figure 3.1: First ten terms for Low discrepancy sequences and pseudorandom numbers

Table 3.2: Benchmark functions

f	Equation	Range	$\mathbf{x}^*$	f^*
f_1	$=\sum_{i=1}^{N} x_i^2$	$[-500, 500]^N$	$(0, 0)$	0
f_2	$= -20 \cdot exp(-0.2\sqrt{\frac{1}{N} \cdot \sum_{i=1}^{N} x_i^2}) - exp(\frac{1}{N} \cdot \sum_{i=1}^{N} \cos(2\pi x_i)) + 20 + exp(1)$	$[-32, 32]^N$	$(0, 0)$	0

dimensions, which is one disadvantage of LDS. Halton LDS calculates a Van Der Corput sequence for each each dimension. Each dimension uses a prime number as the base, i.e. the base is 2 for first dimension, 3 for second dimension, etc. But there are some traces that when the dimensions increase the sequence is degenerated and clusters around cero appeared. That is why Halton LDS is recommended for problems with less than 7 dimensions (Sen, Samanta, & Reese, 2006).

Also, Sobol and Faure Sequences are generalization of Van Der Corput LDS, but those sequences use one base for everything and incorporate some mechanism for sorting some elements. For Faure sequence, each dimension uses the closest and higher prime number from dimension, i.e. dimension 50 Halton sequence uses the 50th prime number while Faure sequence uses 53 which is the closest and higher prime number from 50. But, symmetry is a problem(see Figure 3.1). Sobol sequence uses base 2 for every dimension but change the order for certain numbers preventing clusters and degradation with dimension. In theory, the best LDS for metaheuristic is Sobol. In practice, Omrand et al. and Xuan Hoai et al. shaw the Sobol LDS is the best for PSO algorithm (Omran et al., 2013; Xuan Hoai et al., 2007).

3.1.2 Modification to original version

In the last chapter, We described the original version of supernova algorithm. Initial population is calculated using URD. In the implementation process, some details should be changed. The use of LDS affects the initial population calculation. The solutions of LDS are distribute over hypercube with side=1 while supernova algorithm uses hypersphere with radius=1λ, where λ is a parameter of supernova algorithm. For initial population and restart, we modify the implementation as follow: first, we scaled the LDS to hypercube given by upper and lower borders for the objective function, i.e. we define $\lambda_0 = \mathbf{UB} - \mathbf{LB}$ which guarantees that sequence will be distribute over complete feasible area. Second, the center of each restart will be the best point found as in the original algorithm and reduce $\eta\%$ of the original area. When the new center is asymmetric, the new area will be limited for the original limits and the limits found for the reducing the area.

New version of supernova(SLDS) follows the Algorithm 1 excepting line 5 which changes as we described above. After, we use the method proposed in (Mesa, Velasquez, & Jaramillo, 2015) for developed and implementation of metaheuristics. This method has three main steps: verification, parametrization and validation. For the sake of brevity, we describe briefly the test and present the results for each step.

Verification

The algorithm was implemented using the LDS version from R-package called "randtoolbox" (Christophe & Petr, 2015). We implemented two version with Halton LDS and another with Sobol and called them as H-SLDS and S-SLDS. This initial implementation was tested for 2 well-known benchmark functions: Sphere(f_1) and Ackley's (f_2) functions(See Table 3.2).

Table 3.3: Parameters used for SLDS

function	Parameters							
	K	T	R	F	G	λ_0	α	η
f_1	50	100	50	4	4	500	1.5	0.15
f_2	50	100	50	4	4	32	1.2	0.15

Table 3.4: Parameters used for SLDS

Dimensions	f	H-SLDS			S-SLDS		
		f^*	σ	$F.C$	f^*	σ	$F.C$
2	f_1	0	0	224	0	0	224
	f_2	0	0	226	0	0	214
10	f_1	6.16×10^{-30}	0	214	1.10×10^{-46}	1.05×10^{-23}	224
	f_2	0	1.19×10^{-15}	226	0	0	224
20	f_1	1.32×10^{-29}	1.32×10^{-29}	214	2.04×10^{-29}	4.52×10^{-15}	224
	f_2	0	6.72×10^{-16}	226	0	0	224
30	f_1	1.96×10^{-29}	1.96×10^{-29}	214	7.70×10^{-28}	2.77×10^{-14}	224
	f_2	3.55×10^{-15}	3.98×10^{-14}	226	0	0	224

Parametrization

We use the same experimental design proposed in the Section 1.1.3 and the best parameters for two well-known benchmark functions was chosen and translate to prevent problems with symmetry and sobol LDS (See Table 3.2). The new optimal points are: for f_1, $x^* = [450.1, 450.1]$ and $f^* = 0$ and for f_2, $x^* = [-20.1, -20.1]$ and $f^* = 0$.

Validation

With translated benchmark functions, we compare the result for 2, 10, 20 and 30 dimensions, the minimum (f^*), deviation (σ) and objective function calls ($F.C.$). In the Table 3.4, The columns are for SLDS with Halton and Sobol LDS. The rows are for functions and dimensions. For functions f_1 andf_2, the best minimum is for S-LDS. However, deviation for f_1 is worst for S-LDS. Further, the performance is better for S-SLDS.

3.2 Results

The new version S-SLDS was compared with the previous results for benchmark function proposed in CEC2005. The parameters and other algorithms are described in the Section **??**. In the table3.5, we summarized the mean of the error for original supernova, PSO, and DE for the test and the result for S-SLDS. It is important to remind that S-SLDS is deterministic then only one run is needed.

The initial hypothesises are $H_0 : \overline{f^*}_{snova} = \overline{f^*}_{DE} = \overline{f^*}_{PSO} = \overline{f^*}_{S-LDS}$ and $H_a : \overline{f^*}_i \neq \overline{f^*}_{i'}$ for at least one of the algorithms. For both tests, the rank for ties was calculated as the average. In the Table 3.6, the results for the Friedman's test is shown. The P-values are inferior to 0.5, therefore the H_0 is rejected.

Table 3.5: Results and comparison for benchmark functions, CEC 2005, between Supernova, DE, PSO and S-SLDS. 10 dimension and 50 independent runs except for S-SLDS

f	Error Supernova	Error DE	Error PSO	Error S-SLDS
f_1	2.49×10^{-06}	$\mathbf{8.26 \times 10^{-09}}$	1.23×10^{-04}	5.27×10^{-07}
f_2	4.41×10^{-04}	$\mathbf{8.18 \times 10^{-09}}$	2.60×10^{-02}	5.63×10^{-05}
f_3	5.27×10^{05}	$\mathbf{9.94 \times 10^{00}}$	5.17×10^{04}	6.26×10^{00}
f_4	6.58×10^{-01}	$\mathbf{8.35 \times 10^{-09}}$	2.49×10^{00}	1.22×10^{-03}
f_5	2.39×10^{01}	$\mathbf{8.51 \times 10^{-09}}$	4.10×10^{02}	2.17×10^{01}
f_6	3.77×10^{01}	$\mathbf{8.39 \times 10^{-09}}$	7.31×10^{02}	7.28×10^{00}
f_7	2.46×10^{02}	1.27×10^{03}	$\mathbf{2.68 \times 10^{01}}$	1.59×10^{03}
f_8	$\mathbf{2.03 \times 10^{01}}$	2.04×10^{01}	2.04×10^{01}	2.02×10^{01}
f_9	1.38×10^{01}	$\mathbf{8.15 \times 10^{-09}}$	1.44×10^{01}	4.15×10^{00}
f_{10}	2.28×10^{01}	$\mathbf{1.12 \times 10^{01}}$	1.40×10^{01}	1.19×10^{01}
f_{11}	7.99×10^{00}	$\mathbf{2.07 \times 10^{00}}$	5.59×10^{00}	3.39×10^{00}
f_{12}	$\mathbf{3.25 \times 10^{01}}$	6.31×10^{01}	6.36×10^{02}	8.30×10^{-03}
f_{13}	$\mathbf{1.30 \times 10^{00}}$	6.40×10^{01}	1.50×10^{00}	2.59×10^{-01}
f_{14}	3.71×10^{00}	$\mathbf{3.16 \times 10^{00}}$	3.30×10^{00}	3.44×10^{00}
f_{15}	3.40×10^{02}	$\mathbf{2.94 \times 10^{02}}$	3.40×10^{02}	6.56×10^{01}
f_{16}	$\mathbf{7.45 \times 10^{01}}$	1.13×10^{02}	1.33×10^{02}	1.34×10^{02}
f_{17}	2.49×10^{02}	$\mathbf{1.31 \times 10^{02}}$	1.50×10^{02}	1.51×10^{02}
f_{18}	8.96×10^{02}	$\mathbf{4.48 \times 10^{02}}$	8.51×10^{02}	9.83×10^{02}
f_{19}	8.92×10^{02}	$\mathbf{4.34 \times 10^{02}}$	8.50×10^{02}	8.00×10^{02}
f_{20}	8.83×10^{02}	$\mathbf{4.19 \times 10^{02}}$	8.51×10^{02}	8.00×10^{02}
f_{21}	$\mathbf{1.55 \times 10^{01}}$	5.42×10^{02}	9.14×10^{02}	1.31×10^{02}
f_{22}	$\mathbf{7.53 \times 10^{02}}$	7.72×10^{02}	8.07×10^{02}	7.63×10^{02}
f_{23}	$\mathbf{2.35 \times 10^{01}}$	5.82×10^{02}	1.03×10^{03}	5.47×10^{02}
f_{24}	4.43×10^{02}	$\mathbf{2.02 \times 10^{02}}$	4.12×10^{02}	2.00×10^{02}
f_{25}	4.49×10^{02}	1.74×10^{03}	$\mathbf{5.10 \times 10^{02}}$	1.80×10^{03}

Table 3.6: Results for Friedman test for S-SLDS,Supernova, DE and PSO

	Supernova	DE	PSO	S-SLDS	F_F	P-value	F_{ID}	P-value
Rank CEC's test	2.76	1.94	3.22	2.04	13.67	1.59×10^{-03}	5.3484	2.20×10^{-03}

Table 3.7: Results for Wilcoxon's test for S-SLDS, Supernova, DE and PSO

Algorithms	Values	
S-SLDS-DE	R+	157
	R-	168
	Limit for $\alpha = 0.05$ and one-tailed	100
S-SLDS-PSO	R+	254
	R-	71
	Limit for $\alpha = 0.05$ and one-tailed	100
S-SLDS-Snova	R+	206
	R-	119
	Limit for $\alpha = 0.05$ and one-tailed	100

Now, we know the means are different for at least one algorithm. Compare each algorithm with S-SLDS with Wilcoxon's test (See Table 3.7) we found that S-SLDS is significant better that PSO and there is not significant difference with the previous version and DE. Even so, the difference with previous version are bigger than with DE.

3.3 Conclusions

The initial population distribution helps to improve the performance of the metaheuristic in general. It is an strategy for globalization, but is not particularly good for a particular type of problem like planar regions. Although, it is a interesting improvement for the methodology lost the focus in planar regions. Therefore, We present this part of the search because forward the version of supernova algorithm will be LDS version.

The ongoing work for distribution of initial population will be the change of LDS sequence for tailored nets and/or grids that does not degenerate with dimensions and reduced the computational effort.

4. A CRITERION TO IDENTIFY PLANAR REGIONS[1]

" Would you tell me, please, which way I ought to go from here?" "That depends a good deal on where you want to get to," said the Cat. "I don't much care where –" said Alice. "Then it doesn't matter which way you go," said the Cat."
– Lewis Carroll, Alice in Wonderland

Each type of objective function has a unique topology with different features. One of this features is flat regions. A characteristic of flat regions is small gradients (approximately zero), i.e. the value of the objective function for entire flat region is equal or similar. Several metaheuristics use the differences of the value of objective function between solutions to find a descent direction. Therefore, the information provides by flat regions with the aim of guiding the search is not good enough for determining a clear descent direction. Then, it makes that algorithm get stuck there (Coello, 2005; Törn, Ali, & Viitanen, 1999; Pant & Et, 2011).

In the literature, the problem of flat regions is not consider in isolation. Often, it is approached as a component of the improvement for global optimization for all kind of functions. There is not a solution to improve functions with a flat regions because there are no information in advance to determine if the function has planar regions. Furthermore, there is not a clear idea of the specific characteristics of this region that can be problematic or not for the different algorithms. But, there are benchmark functions that are traditionally problematic and can be consider functions with flat regions. In addition, there is information available about how algorithms work in this functions and how the algorithm has been changed for improving the performance in these functions(Torn & Zilinskas, 1989; Schwefel, 1993; Talbi, 2009; Yao et al., 1999).

The improvement of the algorithm are related with the appropriate balance between exploration and exploration of the metaheuristic. There are different approaches for improving metaheuristics and they are: modification of the parameters and operators and switching between algorithms automatically (Talbi, 2009; Yang, 2010). When Parameters change, the search also change. In this sense, the problem of optimization of objective function with flat regions can be solved by setting the appropriate values for the parameters of the particular metaheuristic (Kramer, 2008).

The presence of large flat regions is not detectable in advance because of there is not available information, as we mentioned before. However, there is information that is being ignore. Several

[1]A preliminar version of this chapter was published on: Mesa, E., Velásquez, J. D. and Jaramillo, P. (2014). Nonlinear optimization in landscapes with planar regions. In Terrazas, G., Otero, F. E. B., and Masegosa, A. D. (Eds.), Nature Inspired Co-operative Strategies for Optimization (NICSO 2013), Vol. 512 of Studies in Computational Intelligence, pp. 203-215. Springer International Publishing.

metaheuristics use a population to search, but they only use the information of the best, some random individuals and/or worst solutions found. Statistics of the value of objective function can be useful for detecting some features of the topology. In this sense, the first objective of this chapter is present a general criterion to identify when there is a problematic flat region in the objective function using the statistics given by population in general. The second one is: to determine the characteristics that can be described as problematic according to the statistic calculated for identify the flat region.

This chapter is organized as follows: in Section 2, we describe the problem of flat regions. In Section 3, we present the developed of a criterion to identify when the objective function has a planar region and if it is can be problematic. In Section 4, we describe a hybrid optimization methodology that switches, between both Supernova and Monte Carlo, using the proposed criterion as parameter control. Furthermore, the test and results are discussed.

4.1 The problem with planar regions

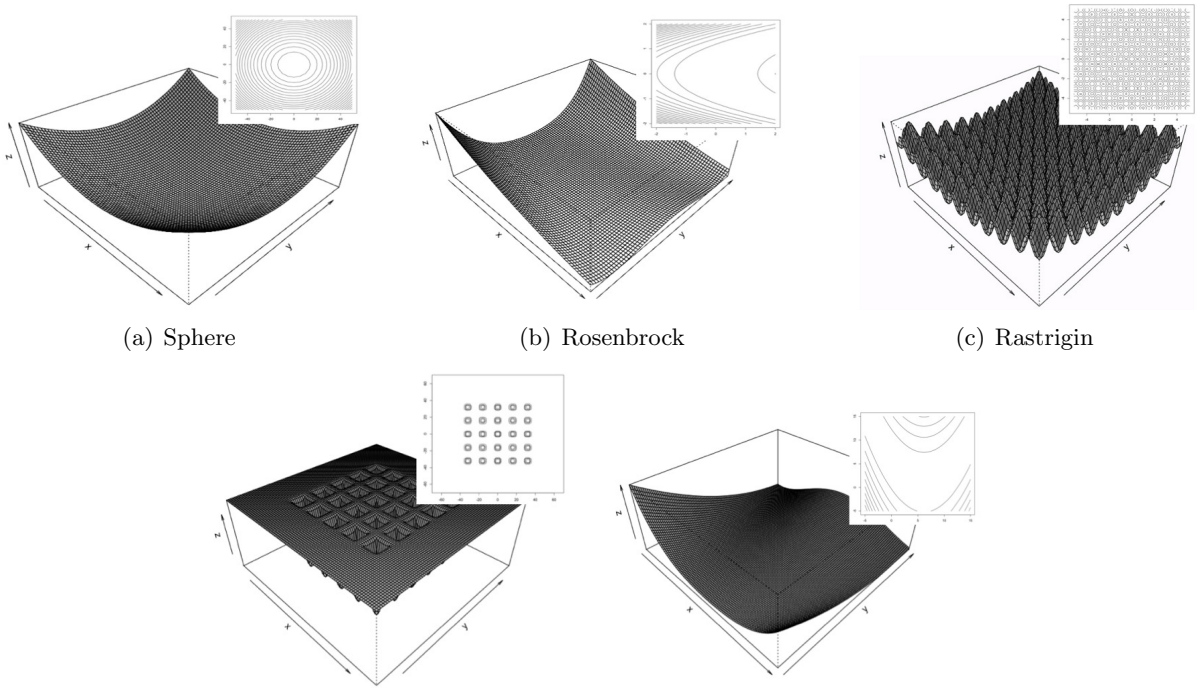
Previous section describes why the optimization of a function that has flat regions can be a problem. In this section, we define what is a flat region for this work. Also, we discuss why this kind of feature is a problem in global optimization and, finally, exemplify the problem using well-known benchmark functions.

4.1.1 The problem with planar regions

A strictly definition of complete flat region, according with definitions of Section 1.3, is: a region $L \in A$, with $L \neq \emptyset$, such that $\forall \mathbf{x} \in L$, $f(\mathbf{x}) = c \in \mathbb{R}$, and all points $\mathbf{x} \in L$ are weak minimums. This implies that $\nabla f(\mathbf{x}) = 0$, $\mathbf{x} \in L$. However, in this work, we consider that the attracting region L for the point $\mathbf{x}^*$ is a flat or planar region when$f(\mathbf{x}^*) \leq f(\mathbf{x})$, $\forall \mathbf{x} \in L$ and$\|f(\mathbf{x}) - f(\mathbf{x}^*)\| \leq \delta$ where δ is an arbitrary quantity near to zero. δ is a small constant that representes the possible amount between complete planar regions, gradient=0, and a region that can be problematic because of small gradients. This implies that $\nabla f(\mathbf{x}) \approx 0$ inside of the attracting region and $f(\cdot)$ is a smooth function with slopes near to zero or zero. Where the attraction region of the minimum has poor information about its location or there are several minimums in the neighborhood. We refer to the initial definition of flat region as a pure planar region and the second definition as a relaxed or non-pure planar region. From this point forward, we will refer to non-pure planar region as planar region.

Classical local optimization based on gradients is not able to optimize functions with the optimal point located in flat regions because the algorithms have not direction to follow. Usually, direct methods based on population emulate the gradient methods, to direct the search using the information of the descendent slope like the water's drop that needs the slope to flow; when the slope information is not available, the drop will be stagnated in one point like the algorithms actually does.

Several metaheuristics have problems for finding the global optimum of a function when such point is located inside of a large flat region, due to: first, there is insufficient information for determining descent directions that is required for guiding the exploratory phase of the algorithm. Second, the local search algorithm would find points with $f(\mathbf{x}_k) \approx f\left(\mathbf{x}_{(k-1)}\right)$ every where. Then, the trajectory of visited points seems to be erratic or random and there is not a clear convergence (Talbi, 2009). Third, the absence of good candidate solutions for being used in the exploitation



(a) Sphere (b) Rosenbrock (c) Rastrigin

(d) Foxholes (e) Branin

Figure 4.1: Benchmark Functions

phase of the used metaheuristic. Thus, metaheuristics presents questionable advantages over pure random optimization when the optimized function has the global optimum inside of a large planar region.

Rarely, we will have a complete planar region for the objective function. Usually, we have a mixture between planar regions and non-planar regions. There is not a unique mathematical criterion to determine when this mixture can be problematic for an algorithm, but we have benchmark functions that are problematic. And they can be consider as functions with large planar regions evaluating their topologies.

4.1.2 Benchmark functions with planar regions

In most relevant literature, commonly benchmark functions are classified in terms of roughness, smoothness, analytic proprieties and/or quantity of minimums, but there is not a classification based on the presence or absence of problematic planar regions. However, several well-known benchmark functions are difficult to optimize due to the presence of large flat regions (Shang & Qiu, 2006; De Jong, 1975; Schwefel, 1993). In Table 4.1 and Figure 4.1, we present five classical functions. Three of them have planar regions.

Functions f_1 and f_3, from Table 4.1, are descendent functions, the other functions present planar regions (see Figure 4.1). In Figure 4.1, we presented the three-dimensional plots and two-dimensional contours of these functions; where we can see the planar regions like bananas for functions f_2 and f_5 and like a foxhole for f_4. Note that there is not a complete planar in any function, there is a mixture between types of regions with a lot of information about location

Table 4.1: Benchmark functions: name, equation, minimum, and search region

Name	Function	Search Area	minimum
Sphere	$f_1(x) = \sum_{i=1}^{D} x_i^2$	$[-500, 500]^{30}$	0
Rosenbrock	$f_2(x) = \sum_{i=1}^{D-1} [100(x_{i+1})]x_i^2$	$[-5.12, 5.12]^2$	0
Rastrigin	$f_3(x) = \sum_{i=1}^{30} [x_i^2 - 10\cos(2\pi x_i + 10)]$	$[-100, 100]^{30}$	0
Foxholes	$f_4(x) = \left[\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j+\sum_{i=1}^{D}(x_i - a_{i,j})^6}\right]$	$[-65.5, 65.5]^2$	≈ 1
Branin	$f_5(x) = \left(x_2 - \frac{5.1}{(4\pi^2)}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2$ $+10\left(1 - \frac{1}{8\pi}\right)\cos x_1 + 10$	$[-5, 10][0, 15]$	0.398

of minimum and zones with a pretty poor clues about minimum location. For these kind of functions, the algorithm can get struck. All these three functions are described in the literature like problematic functions (Shang & Qiu, 2006; De Jong, 1975; Schwefel, 1993).

4.2 A Criterion To Determine Problematic Planar Regions

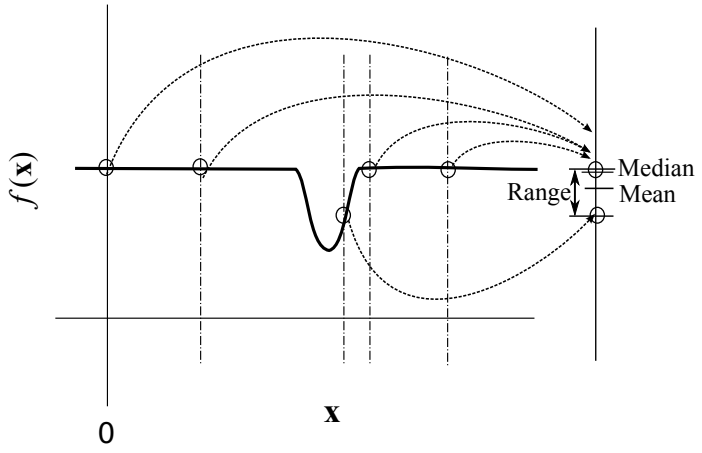
In real-world problems, the geometrical configuration of the objective function is unknown and the identification of a planar region is not a trivial problem. If the topology is identified, this information can be used for tuning the parameters of the metaheuristic changing the exploration-exploitation balance for improving the performance. Furthermore, this can be used for selecting an alternative optimization method best suited for random search. In this Section, we propose an empirical criterion for determining when the objective function has a planar region that can be a problem for the optimizer.

4.2.1 Proposed empirical criterion

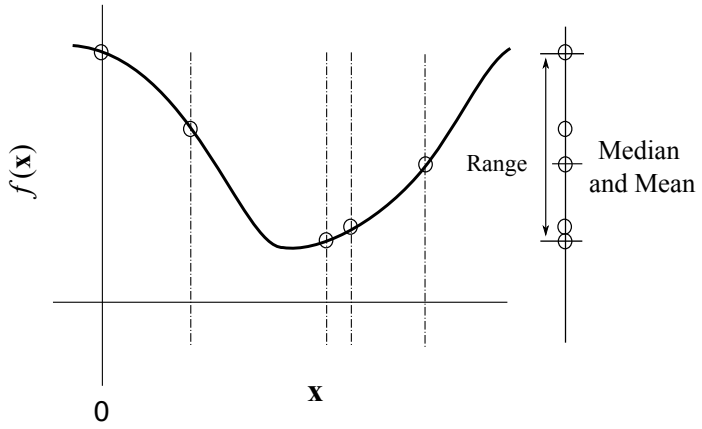
Usually, metaheuristic methods use the known values of objective function to guide the search towards regions where it is possible to find a optimum. Evolutionary strategies (ES) uses the values of objective function to calculate the fitness of the population for finding a next generation, which is improved thanks to the knowledge of previous generation. Moreover, in particle swarm optimization (PSO) algorithm, the values of objective function calculated for the current particles in the swarm are used to calculate the next direction and magnitude of the displacement of each particle. Thus, PSO uses an empirical descendent direction which is similar to the gradient techniques (Bäck, 1996; Kennedy & Eberhart, 1995). In both cases, the similarity with gradient search imposes the same weakness: problems with the performance in presence of large planar regions.

However, the information provided by the points already visited for the metaheuristic is not used to extract important features of the objective function as smoothness, roughness and slopes. For example, when $f(\mathbf{x}_i) = c$, with $c \in \mathbb{R}$, for all $\mathbf{x}_i$ belongs to the current population of a population-based optimization algorithm, the standard deviation, mean and median of the asset value's $f(\mathbf{x}_i)$ will be zero and c respectively, and this information would be used to determine when the population is located inside a pure planar region.

But, what will happen with a scatter measures of objective function values when the region has a large (non-pure) planar region with small descendent areas? Figure 4.2(a) shows the results. Population has a small finite size (approximately 40 to 100). If function have a long flat region,



(a) planar function



(b) non-planar function

Figure 4.2: Scatter of objective function values versus kinds of objective functions: planar regions with descendent areas and purely descendent functions.

Table 4.2: Criterion meaning

Rg	Tc	Meaning
$Rg \approx 0$	$Tc \approx 0$	Weak minimum zone.
$Rg \gg 0$	$Tc \approx 0$	Descend function.
$Rg \gg 0$	$Tc \gg 0$	Mixture between weak and strong minimum zones.

most part of the population will have the same value for the objective function, then median will be around this value. But the solutions out side of the flat region will affect the measures; this affects mean but no the median. In this case, mean and median values will be different.

Accordingly, what will happen when the function is strongly descendent? In Figure 4.2(b), a descendent function is plotted. In this case, we could see that the difference between the mean and median are zero or could be near to zero. That means, we could identify a large planar region from the scatter measures sampled using the current population of the optimization algorithm. We propose two measure of variability as indicators of problematic planar regions. For a set of solutions, we calculate the range R_g, higher values indicate that there are a large variation in the values of the samples. Thus, we divide R_g between the dimension of search space to have an average of the change by one unit of this space. We define R_g as:

$$R_g = \frac{f(\mathbf{x})_{max} - f(\mathbf{x})_{min}}{\|\mathbf{ub} - \mathbf{lb}\|} \tag{4.1}$$

Afterwards, for the same, we calculate the difference, T_c between mean and the median and observe how the sampled are distributed. Low values indicate a well-distributed sample, which implies homogeneity, and it is agree with descendent functions (See Figure 4.2(a) and 4.2(b)). We define T_c as:

$$T_C = \frac{f(\mathbf{x})_{mean} - f(\mathbf{x})_{median}}{f(\mathbf{x})_{max} - f(\mathbf{x})_{min}} \cdot 100 \tag{4.2}$$

In Table 4.2, we show the possible scenarios for both measures R_g and T_c. In both cases, we normalized for a fair comparison between functions and have a range or frontiers to determine when it is or not a problematic function. Until now we have a concept to recognize between a planar region and non-planar region but it is not a criterion because there are not values to determine when function is or not. Initially, when are R_g and T_c near to or differ from zero. To use the criterion first we need to review that:

- The criterion identify between descendent region with small gradient and planar regions.
- The criterion could identify planar regions for high number of dimension.
- The criterion is not sensitive to the sample size used to calculate the criterion it means, the population size.
- The values of R_g and T_c that must be consider problematic for the algorithm, in this case PSO, and if it is the same for all problems and all dimensions.

Table 4.3: Results for one-dimensional functions.

α	f_d		f_p	
	Rg	Tc	Rg	Tc
1	9384.76	7.85%	26818.06	12.6%
0.5	4692.38	7.85%	13409.03	12.6%
0.1	938.48	7.85%	2681.80	12.6%
0.01	93.647	7.85%	268.18	12.6%
0.001	9.39	7.85%	26.81	12.6%

Table 4.4: Results for f_d and f_p as a multi-dimensional functions.

N	f_d		f_p	
	R_g	T_c	R_g	T_c
1	9384.76	7.85%	26818.06	12.6%
2	15312.50	0.59%	39498.16	13.2%
10	51640.63	1.58%	109868.12	19.8%
30	139824.21	0.26%	298131.17	22.9%

4.2.2 Behavior and sensibility of the test for known functions

The criterion proposed is a new and novel approach to identify planar regions. With the aim of analyzing the behavior of the proposed measures of variability, we conduce a test using two tailored functions, a parabola

$$f_d = \alpha x^2 \tag{4.3}$$

and a piecewise function with a planar region (See Figure 4.3), defined as:

$$f_p = \begin{cases} \alpha \mathbf{x}^2 & [-2, 2] \\ 4\alpha & (-70, 2) \cup (2, 70) \\ \alpha \mathbf{x}^2 - 140\alpha \mathbf{x} + 4904\alpha & [-100, -70] \cup [70, 100] \end{cases} \tag{4.4}$$

In both cases, α vary between $[0.001, 1]$ and controls the value of the gradient of the function. In other words, the parabola and the piecewise function with $\alpha = 0.001$ will be flatter than the both functions with $\alpha = 1$. And $\mathbf{x}$ vary between $[-100, 100]^D$ where D is the dimension. The objective of the first experiment is to determine the ranges of values for both measures, if the criterion can identify between a planar region and descendent region with small gradients.

In Table 4.3 we present the calculated values for R_g and T_c for one-dimensional functions version for different values of α. The sample used consists in 50 points obtained from the Sobol low discrepancy sequence(LDS) (Gentle, 2003). For both one-dimensional functions we found that: first, R_g is directly proportional to α and T_c is independent of the gradient, but changes between both functions; this means that the criterion it is not sensitive to small values for gradient of descendent functions for non-random points. Second, we can observe the difference between mean and median that we described above with the values of T_c for both functions. T_c is bigger for f_p which means that f_p is more planar than f_d according with construction of the functions.

Next, we analyze whether the obtained results for one-dimensional functions are sensitive or not to sample's size. For this, we calculate R_g and T_c for different sample sizes and random points for

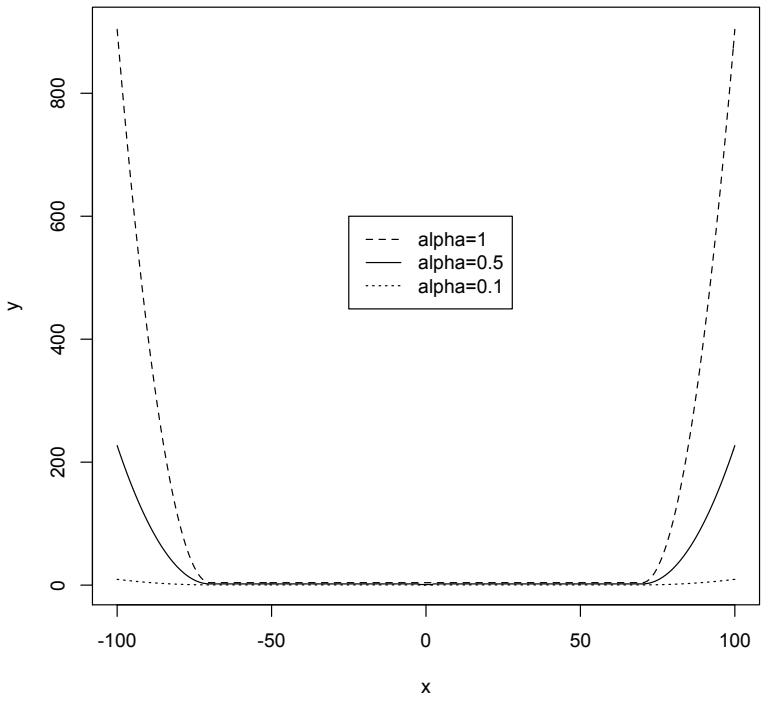
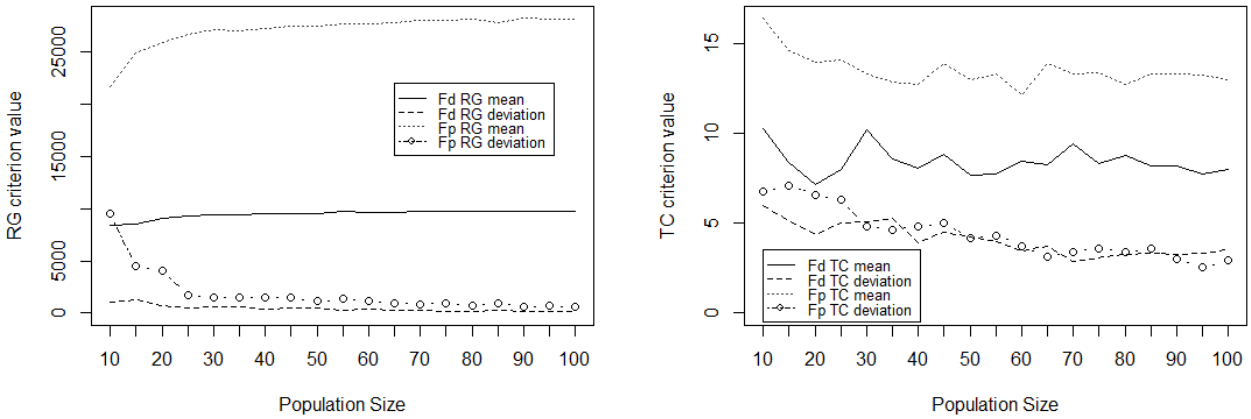


Figure 4.3: piece-wise function f_p



(a) Results for RG

(b) Results for TC

Figure 4.4: Values for the criterion using k numbers to evaluate the function. The numbers are random with uniform distribution.

Table 4.5: Criterion results for benchmark function

	f_1	f_2	f_3	f_4	f_5
Rg	3495605.47	60301.24	775.55	495.02	204.07
TC	0.26%	13.12%	0.67%	8.65%	12.06%

both functions. We consider sample sizes from 10 to 100 points every 10 points. For each sample size, we realize 50 independent runs where the points used for calculating R_g and T_c are drawn from a uniform distribution. Figure 4.3 shows the values of R_g and T_c for both test functions in one-dimension; the mean of the value for criterion and the correspondent standard deviation are shown. We found that R_g is more sensitive to small sample sizes than T_c and both criteria tend to be constant for large sample sizes. Figure 4.4(b) shows a pretty clear difference between T_c for f_d which is smaller than T_c for f_p. So, the minimal number of points recommended is 30.

Following, we analyze the behavior of the criterion when the number of variables (dimensions of the problem) changes. In Table 4.4, we show the values of the proposed measures. These were calculated for both test functions and for 1, 2, 10 and 30 dimensions. As in the first experiment, we used a Sobol low discrepancy sequence for obtaining 50 solutions of the test functions. According to the results, we can conclude that criterion is sensitive to the number of variable; which was expected because with dimensions the difference between plans augment. Nevertheless, the difference T_c is increase with number of dimensions which is positive. In other words, for high number of dimensions the difference identified by the criterion between planar regions and non planar regions will be stronger. However, the relationship between criterion values and dimension is unknown. Based on the results, we did a polynomial regression and obtain that:

$$T_c \geq 7.675(N)^{-0.588} \tag{4.5}$$

We use the equation 4.5 to determine when it is a problematic planar region.

4.2.3 Behavior of the test for benchmark functions

In last sections, we explain the concepts and define the criterion. Now we will test it. Initially, we choice three benchmark functions that present problematic weak minimums and two the descendent functions as control functions. These functions, and correspondent, dimensions, search areas, and minimums are presented in Table 4.1 and Figure 4.1. Table 4.5 shows the results; functions f_1 and f_3 have not problematic planar region and the value of T_c are smaller than the other functions. R_g in both cases is greater than zero. For the other function T_c is greater than the limit 7.6. The criterion identify different function as planar o non planar regions.

4.3 Using criterion to select optimization method

In this Section, we applied the criterion to select between two different algorithms according with balance between exploration and exploitation. The first one is Supernova (high exploitation) the proposed algorithm in the latest version which is previous chapter version. The second one is Monte Carlo (high exploration). The aim of this test is to show how this criterion can use to decide about the optimization strategy. In this case, the criterion will switch between two algorithms, Supernova and Monte Carlo sampling. Monte Carlo is a purely stochastic method that has been

used for different applications and has not a sophisticated method inside. It uses the random sampling to find the optimums, but it has been proved to be superior when the information about localization of minimum is poor because it is exploring different regions (Yang, 2010). In this approach, the proposed criterion is used to decide when use supernova or Monte Carlo sampling. The sequence of the algorithm is shown in the Pseudocode 1.

Algorithm 5 Hybrid proposed Supernova-MC

1: $S_k = randomx_1, ..., x_k$
2: Evaluation of $f(x)$ for S_k
3: Evaluation of Rg and Tc
4: **if** $TC \geq 7.675\,(dimensions)^{-0.588}$ and $Rg > 0$ **then**
5: Use monte carlo algorithm
6: **else**
7: use Supernova algorithm
8: **end if**

4.3.1 Results for the Hybrid between Monte Carlo and Supernova

In this test, we used supernova implementation described in last chapter and our own implementation of the most simple version of the Monte Carlo methodology (Yang, 2010). The proposed optimization methodology is used for optimizing the five functions presented in Table 4.1. In addition, we optimized other five functions from CEC'05 benchmark test, correspondent to functions f_{11} to f_{15} for details see (Suganthan et al., 2005). The additional functions are for two and ten dimensions. In step 3, the proposed criterion is evaluated using 50 random points drawn from a uniform distribution. In step 4, we sample 12.000 random points drawn from a uniform distribution and use as a result the best point found. In step 5, we run supernova with 50 iterations 6 restarts and 40 individuals and stop criterion describe in chapter 1; we use the parameters described in last chapter for supernova.

In Table 4.6, we present the results found by each algorithm (mean and standard deviation for 50 independent runs) and the values for the criterion. The bold results are the ones taken by the hybrid proposed. The results for functions f_2 and f_5 are pretty similar, but the deviation is better in both cases. In terms of computational effort, supernova has a maximum of 300 calls to objective function and a population of 40 individuals, but it has internal operations. Monte Carlo has 20000 calls for 2 dimensions 200000 calls to the objective function without internal operations. The characteristics of the processor are 2.9 Ghz intel i7 processor. For the complete set of function for 2 dimensions, we have six functions with problematic planar areas, and four with non planar areas. The first five we are sure that the topology that they have and the functions from CEC'05 are hybrids and rotated functions which makes difficult to determine the topology. Nevertheless, criterion shows the problematic functions and let use the best strategy for the function. For 10 dimensions, function 4th are not available for 10 dimensions and Rosenbrock function change the topology, it is multimodal and according with criterion it is not a problematic planar region.

Table 4.6: Results of hybrid proposed for benchmark functions.

2 dimensions							
Function	Criterion Rg	Tc	Type of function	Monte Carlo Mean	Desviation	Supernova Mean	desviation
f_1	3495605.47	0.26%	Non Planar	4,23E-01	3,85E-01	**2,96E-08**	**5,60E-08**
f_2	60301.24	13.12%	Planar	**9,87E-04**	**1,25E-03**	1,64E-03	3,18E-03
f_3	775.55	0.67%	Non Planar	6,53E-01	5,06E-01	**6,70E-06**	**2,12E-05**
f_4	495.02	8.65%	Planar	**9,98E-01**	**4,03E-06**	1,28E+00	5,17E-01
f_5	204.07	12.06%	Planar	**1,00E+01**	2,40E-09	1,00E+01	5,88E-07
f_6	4.45	0.41%	Non Planar	90.29	0,85E-00	**90.01**	**2.44E-02**
f_7	4662.31	7.5%	Planar	**-459.87**	**8,25E-02**	458.67	3,83E-01
f_8	1.11E05	104%	Planar	**-127,59**	**2.3E-01**	-125.67	2,12E00
f_9	0.018	1.8%	Planar	**298.99**	**5,4E-01**	289.99	5,47E01
f_{10}	195.4	0.74%	Non Planar	194.73	5.5E01	**123.12**	**2.19E00**
10 dimensions							
Function	Criterion Rg	Tc	Type of function	Monte Carlo Mean	Desviation	Supernova Mean	desviation
f_1	9.831E05	0.56%	Non Planar	7,57E-02	5,35E-01	**2,96E-08**	**5,60E-08**
f_2	2.335E13	1,79%	Non Planar	9,87E-04	1,25E-03	**1,64E-03**	**3,18E-03**
f_3	5.319E04	0.8887%	Non Planar	6,53E-01	5,06E-01	**6,70E-06**	**2,12E-05**
f_5	1.001E03	6.758%	Planar	**1,00E+01**	2,40E-09	1,00E+01	5,88E-07
f_6	10.53	0.94%	Non Planar	9.746	0,85E-00	**9.55E-01**	**2.44E-02**
f_7	1.127E06	3.34%	Planar	**48262.4**	**1.659E04**	458.67	3,83E-01
f_8	1.097E07	6.46%	Planar	**12.67**	**3.025E01**	1,30	2,12E00
f_9	0.4928	2.2577%	Planar	**21.491**	**7.177E01**	3,71	5,47E01
f_{10}	1740.72	1.824%	Non Planar	622.064	2.042E01	**3.40E02**	**2.19E00**

4.4 Conclusions

We conclude that the criterion proposed works successfully as a strategy to identify problematic planar regions of functions with different dimensions, configurations and for different populations. As a strategy of control, it levers the advantages for each algorithm agrees with the topology of the function. It improves the individual behavior without any parameter change.

In this case, the main goal of this work was test de criterion as a successful method to improve the performance of the algorithm. And the results for the proposed small benchmark set are enough to determine the potential of the criterion as a general strategy to get good information from function. The computational effort of the criterion proposed is small and it is a good way to learn how the search strategy must to change, which is desirable. Nevertheless, this is the first approach to the idea then it is immature yet. But now the criterion shows potential as a successful strategy for hybrids and could be used for control and tuning parameters for different metaheuristics not only supernova or Monte Carlo.

5. DISCUSSION ABOUT THE RELATIONSHIP BETWEEN THE LEVEL OF THE PARAMETERS AND DIVERSITY IN THE POPULATION

"The essential is to think that anything you are doing has to become the occasion for slashing. You must examine this well."
– Miyamoto Musashi,Five Rings

Metaheuristics use parameters for adjusting the method to new problems, but the choice the right set of parameters is an optimization problem itself. Each parameter has its own role inside the strategy of search. Each one, alone or mixed with other parameters, increases or decreases the intensity and diversity of the search around to the solutions found and selected. For example, genetic algorithms use a crossover operator to increase or decrease the intensity of the search around solutions chosen as parents and the number of the parents is the parameter for controlling it (Bäck, 1996).

The performance of a metaheuristic is related to the set of parameters. There is not a unique method to choose the right set of parameters and guarantee the best performance of the metaheuristic for a particular problem. Often, parametrization uses empiric knowledge to adapt or self-adapt each parameter for the different strategies. The contribution of each parameter is difficult to determine and depends on the problem and metaheuristic. In addition, the landscape of the objective function is the best indicator for determining which balance is better, but it is unknown in advance (Törn et al., 1999).

A parameter of the metaheuristics has three different characteristics: affectation level, range and sensibility (Angeline, 1995; Kramer, 2008). Each characteristic influences the search in diverse manners. Often, the sensibility and range are evaluated and validated in the original versions of the metaheuristics which give us an idea about how they affect the search. However, the level of affectation is described, but is not evaluated (Zambrano-Bigiarini, Clerc, & Rojas, 2013; Mezura-Montes et al., 2006). There are studies about different values and range where the level is treated indirectly. For example, when parameters change the level from one version to another. When a parameter changes the level version by version, these tests show some traces about influence of the parameter level in the balance of exploration and exploitation (Zaharie, 2002).

In this chapter, we focus on the influence level of the parameter because it is unknown. Agree with Angeline (1995), Eiben, Hinterding, and Michalewicz (1999) there are two type of parameters, external and internal, we focus on interal ones. For the internals, there are three kind of influence levels. The first level is population level. This means that parameter affects all population, e.g. size of population. The second level is individual level (solution level). In this case, the parameter affects individual by individual, e.g. ω from the original version of particle swarm optimization (PSO).

Where ω is a factor of the velocity of the individual and affects the whole solution (Eberhart & Kennedy, 1995). Finally, component level this is the lowest one. In this case, the parameter could influences or not each component of the solution individually, e.g. F for differential evolution metaheuristic. Here, the parameter affect only some components of the solutions and not whole individual (Storn & Price, 1997b). For this research, we use the classification of influence level of internal parameter for study the relationship between influence level and balance of exploration and exploitation.

The main goal of this chapter is shown how the level of the parameter affects search. The analysis focuses on affectation level and if it changes how the search is affected. Also, how this level changes can be introduced to control and tuning the parameters using the three variables: Level, range and sensibility. Different version of three classical algorithms, PSO, DE and ES, are compared and analized how the modification to the level of the parameter between version affects balance between exploration and exploitation for both, method and problems.

This chapter is organized as follows. First, a brief summary about balance between exploration and exploitation is presented. Afterwards, we describe the algorithms, their modifications and parametrization, focussing in their exploration and exploitation strategies. Also, we present a experimental design for determining the increasing and decreasing of search areas. Finally, we analyse and discuss the results for different algorithms.

5.1 Exploration Vs. Exploitation

Each problem and method there is a right balance between exploration and exploitation. However, this balance is unknown in advance and found it is an optimization problem by itself. In literature, some authors use the words diversity and similarity as synonymous of exploration and exploitation respectively (Črepinšek, Liu, & Mernik, 2013). Exploration or diversity is described as the amount of new regions included in the search. While, exploitation or similarity is the increasing of the research surrounded to a point found. That means diversity tries to avoid trap of local optimums while similarity improves the quality of the optimal solution (Torn & Zilinskas, 1989).

Both, diversity and similarity, are opposite and linked with the neighbourhood. Multiobjective paradigm has some measures of diversity. In this case, diversity try to guarantee the quality of the Pareto front. Thinking about these measures we found some important topics (Zhou, Qu, Li, Zhao, Suganthan, & Zhang, 2011):

- The diversity has two point of view solution $\mathbf{x}$ and the value of objective function $f(\mathbf{x})$
- Diversity measure can be from one point, a set of points a population or a subset of points.
- A measure of similarity can be defined as a close neighbourhood.

Definitions for exploration and exploitation in this work are:

Definition 5.1.1. Similarity of the Neighbourhood(SN) The similarity of of individual $\mathbf{x}$ with a population or subset of this population P is denoted for a function $d(\cdot)$ such as $SN(\mathbf{x}) = d(\mathbf{x}, P)$ where $\mathbf{x}$ could be also $f(\mathbf{x})$ and function $d(\cdot)$ can be a function of any attribute e.g. distance, variances (Glover, Ching-Chung, & Dhir, 1995a).

Definition 5.1.2. Exploration If $SN(\mathbf{x}) > TH$ where TH is a threshold defined for the particular case and be constant, variable or a function (Črepinšek et al., 2013).

Definition 5.1.3. Exploitation If $SN(\mathbf{x}) \leq TH$where TH is a threshold defined for the particular case and be constant, variable or a function (Črepinšek et al., 2013).

5.2 The level of parameters and the probability of exploration in known metaheuristics

The robustness of a metaheuristic is directed linked with parameters and how much the metaheuristic can be ajusted to different problems, i.e. the flexibility of the metaheuristic which is a parameter problem. Parametrization is a transversal problem for metaheuristics since 70s. Though, the different approaches developed to solve this problem, there is not a perfect and unique method to do it. In this section, we present a brief background of a few methods, the main differences between them and focus on parameter level to compare the performance of the parameters in different levels.

5.2.1 how to control and tune the parameters for metaheuristics

Since first developes in the 70s, metaheuristics use parameters, but the analysis and novel approach are focus only in the strategy. In 90s, some authors started to review the different paradigms for metaheuristics and made a taxonomy for the two principal point of view for parameters: characteristics of the parameters and how to define the parameters (Angeline, 1995; Eiben et al., 1999). As we mention above the characteristic for each parameter was defined as: affectation level, range and sensibility. The parametrization was focus on when the parameters are chosen and defined. There are two moments: before to run the algorithm and inside the running. Both, characteristics and parametrization, are applied for each algorithm and strategy focus on the parameters. For example: $\frac{1}{5}$ rule is a classical self-adaptation for ES. In this rule, the parameter changes the range to increase the exploration after 5 iterations without improvement.

In Figure 5.1, we present the classification summarized by Kramer(2008) where describe the methods to control and tune the parameters. For tuning, the setting of the parameters is doing before the algorithm started. Otherwise, the setting of parameters in the controlling part is doing when the algorithm is running. Each method has their own strategies. Tuning has 3: Ad hoc, experimental design and evolutionary, and controlling has 4: again evolutionary, deterministic, adapt and self-adapt.

- Ad Hoc: the parameters are chosen by expert judgement.
- Experimental design (DOE): a test for parameters is designed and agrees with statistics of the test the best set of parameters is chosen.
- Evolutionary: Parameters are optimized by a metaheuristic.
- Deterministic: Parameters change inside the algorithm agree with iteration or time independently of the behaviour.
- Adapt: parameters follow a rule to adapt, e.g. most part of stop criteria.
- self-adapt parameters change when there are some conditions like $\frac{1}{5}$ rule.

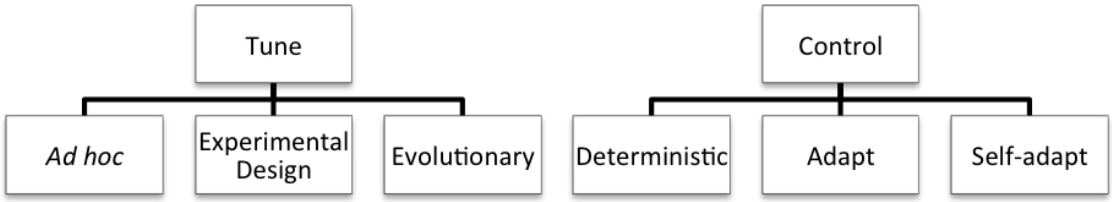


Figure 5.1: Classification for parametrization

5.2.2 PSO Vs. SPSO2011

In this section, we compare the original version of PSO and standar PSO proposed by Clerc in 2011 known as SPSO2011 (Zambrano-Bigiarini et al., 2013). Although the main differences are described, we focus in the parameters of each method and how they affect the search for each version.

From original PSO, the velocity equation is

$$\mathbf{v}_k \leftarrow \omega \mathbf{v}_k + c_1 * \mathbf{u}_1(\mathbf{x}_G - \mathbf{x}_k) + c_2 * \mathbf{u}_2(\mathbf{n}_k - \mathbf{x}_k) \tag{5.1}$$

where $\mathbf{u}_1$ and $\mathbf{u}_2$ are diagonal matrices $n \times n$ whose components are random numbers uniformly distributed ($\sim U(0,1)$). $\mathbf{n}_k$ is the best solution found for each particle k and $\mathbf{x}_G$ is the best solution found for the algorithm in the previous iterations. In this case, the three parameters (ω ,c_1 and c_2) affect the complete solution, i.e. the parameter ω modify the amplitude of the previous velocity. The random vectors increase the exploration and avoid a complete deterministic scatter. The Equation 5.1 correspond to the Line 18 of the Pseudocode 6

Six points of the algorithm have been described as problematic topics and there are several strategies to try to solve those. We can divide them in three kinds. First, algorithm problems: Stagnation, dimensional stagnation and rotation variant. Second, Computational effort: population size. Third, exploration problems: local trap and scale problem. In sake of brevity, we only presented few examples of proposed version of PSO. Some solutions for local trapped is boarded for different versions like: comprehensive-learning PSO (CLPSO) (Liang, Qin, Suganthan, & Baskar, 2006) , LAPSO, the hybrid between differential evolution and PSO (DE-PSO), among others (Zhang & Xie, 2003). Computational effort (hybrid particle swarm with differential evolution operator., 2006) is improved by μ-PSO, coordinate PSO(cle, 2006; Chen, Huang, Jia, & Min, 2006).

Algorithm 6 PSO pseudocode

Require: ω, c_p, c_g, K, T

1: $t \leftarrow 2$;$c_p = c_g \leftarrow 1.193$;$\omega \leftarrow 0.721$
2: $\mathbf{v}_0 \leftarrow$ runif(1)
3: **for** $k = 1$ to K **do**
4: $\quad \mathbf{x}_k \leftarrow$ runif(N)
5: $\quad \mathbf{x}_G \leftarrow \arg\min f(\mathbf{x}_k)$
6: $\quad \mathbf{v}_k \leftarrow \omega v_0 + c_1 * \mathbf{u}_1(\mathbf{x}_G - \mathbf{x}_k)$
7: $\quad \mathbf{s}_k \leftarrow \omega \mathbf{v}_k + \mathbf{x}_k$
8: $\quad \mathbf{n}_k \leftarrow \mathbf{x}_k$
9: **end for**
10: **while** $t \leq T$ **do**
11: $\quad \mathbf{x}_k \leftarrow \mathbf{s}_k$
12: $\quad$ **for** $k = 1$ to K **do**
13: $\quad\quad \mathbf{n}_k \leftarrow \arg\min\{f(\mathbf{n}_k), f(\mathbf{x}_k)\}$
14: $\quad$ **end for**
15: $\quad$ **if** $\min\{f(\mathbf{x}_k), \forall k\} \leq f(\mathbf{x}_G)$ **then**
16: $\quad\quad \mathbf{x}_G \leftarrow \arg\min\{f(\mathbf{x}_k), \forall k\}$;
17: $\quad$ **end if**
18: $\quad \mathbf{v}_k$ % velocity of calculation agree with algorithm version.
19: $\quad \mathbf{s}_k \leftarrow \omega \mathbf{v}_k + \mathbf{x}_k$
20: $\quad t \leftarrow t + 1$
21: **end while**
22: **return** $\mathbf{x}_G$;$f(\mathbf{x}_G)$

The standard versions of PSO, proposed by Clerc (Zambrano-Bigiarini et al., 2013; cle, 2006; Zambrano-Bigiarini et al., 2013), summarize and analyze the best strategies for the problematic function. The standard version used in this work is SPSO 2011 which is invariant to the rotation. The main difference between both version of PSO (original PSO and SPSO2011) is the calculation of new velocity (See Line 18). While original version uses the best solution found and the best solution for each particle to calculate the new population, SPSO2011 asses a hypersphere[1] based on b random solutions. b is a new parameter include in this standard version and regularly are three.

The main improvements of SPSO-2011 version are: random topology paradigm(here topology are referring to neighborhood of the particles) and the change from the hyper-cube to hyper-sphere (Zambrano-Bigiarini et al., 2013). The mixture of these improvements change Line 1 of Pseudocode 6. In this case, the velocity is calculated using random topology paradigm (cle, 2006). Also, Line 5 is changed by introducing a new term in the Eq. 5.1 given by

$$\mathbf{G}_i^t = \frac{b\mathbf{x}_i^t + c_1\mathbf{u}_1 \otimes (\mathbf{p}_i^t - \mathbf{x}_i^t) + c_2\mathbf{u}_2 \otimes (\mathbf{g}_i^t - \mathbf{x}_i^t)}{b} \tag{5.2}$$

Then, hyper-sphere $\mathcal{H}_i(\mathbf{G}_i^t, \|\mathbf{G}_i^t - \mathbf{x}_i^t\|)$, where $\mathbf{G}_i^t$ is the center of hyper-sphere and the radius of the hype-sphere is $\|\mathbf{G}_i^t - \mathbf{x}_i^t\|$

[1] The original version works with hypercube

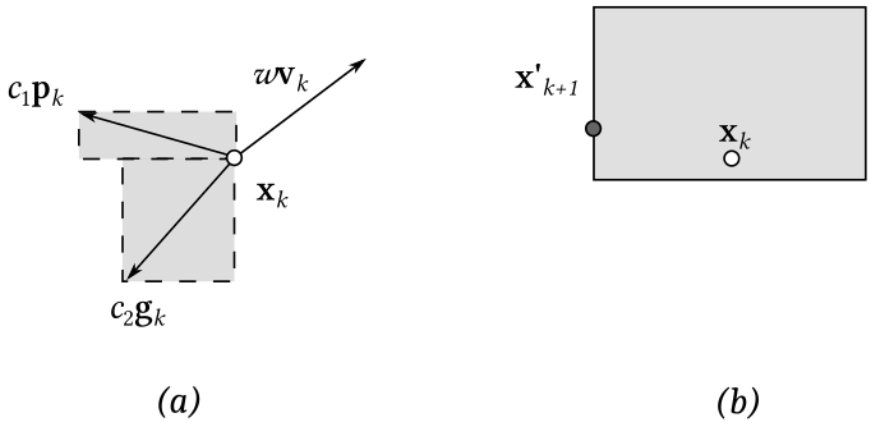


Figure 5.2: For given vector factors the probability square for next generation (a) show the vectors for a solution $\mathbf{x}_k$ and (b)the probable area for next generation solution or $\mathbf{s}_k$

In Line 5 of Pseudocode 6 the equation is modified from Equation 5.1 to Equation **??** while Line 6 does not change. Thus, the version that we use to test and validate the proposed strategy is SPSO-2011.

$$label v_2 011\mathbf{v}_i^{(t+1)} = \omega\mathbf{v}_i^{(t)} + \mathcal{H}_i(\mathbf{G}_i^t, \|\mathbf{G}_i^t - \mathbf{x}_i^t\|) - \mathbf{x}_i^t \tag{5.3}$$

How parameters work in SPSO2011 and PSO

For the original PSO, new swarm is calculated as a vector sum and use three parameters. ω is the parameter that scaled the velocity of the individual. ω is constant and its level is individual. If $\omega < 1$ the magnitude of the velocity is reduced in other case, the magnitude of this vector is equal or bigger. The other two parameters, c_1 and c_2, are factors of the influence between local best and global best found for the algorithm, both are individual level too. In Figure 5.2 part (a),we present the vectors of the inertia and the influence of the best local and global multiplicand by parameter factors. In the part (b) we represent a $\mathbf{x}'_k$ that will be the next solution without random matrices $\mathbf{u}_1$ and $\mathbf{u}_2$. The gray square in Figure 5.2 part (b) represent the area where the new individual would be located randomly. The perturbation induced by this random matrices to the components of the individuals increases the exploration of the algorithm.

In the last paragraphs, we described both versions of the PSO. The original version has same weakness of a hill climb method, good to find the descent direction, but it gets tramp in local minimal solutions. Furthermore, PSO is a central force algorithm, i.e. poor exploration for intricate topologies (cle, 2006). SPSO2011 joins different develops from original version. The main improvement is the invariance to rotation (Zambrano-Bigiarini et al., 2013). However, the changes in the parameters modify the probability to find the attraction area to the global minimum. In the first case, the probability will be distributed over vectors, and new population will be a factor of the resultant vector. SPSO2011 version works with a hypersphere which is building from global best solution ever, the best local solution and a random neighbor. This hypersphere is invariant to

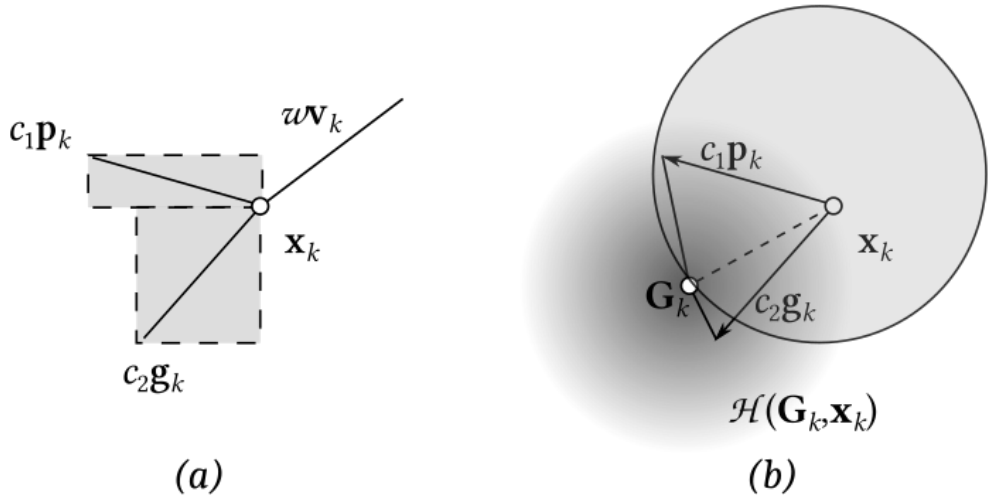


Figure 5.3: For given vector factors the probability sphere for next generation (a) show the vectors for a solution $\mathbf{x}_k$ and (b) the probable area for next generation solution or $\mathbf{s}_k$

the rotation, but also the distribution over the feasible area is bigger.

5.2.3 Differential evolution vs. Self-adaptive DE (SaDE)

In this section, we compare the original version of DE and a self-adaptive version. We discuss how the strategies modify the behavior of the algorithm focusing in the parameters. The original algorithm and the changes made for the new version are described. And how the differences between original version and self-adaptive version lever to the control over the balance between exploration and exploitation.

Differential Evolution

In 1996, DE was proposed by Storn and Price, as a new algorithm of the evolutionary paradigm (Storn & Price, 1997a). Evolutionary metaheuristics have a structure: Evaluation of Objective function, Selection, Operators (crossover and mutation) New population and start over until a stop criterion. DE follows this structure, but operators are not complete independent as in other algorithm of evolurtionary paradigm. i.e. There is one operation for crossover and mutation. As PSO method, DE calculated a velocity that is a combination of both operators. In the Pseudocode 7, there is the basic structure of the algorithm (Feoktistov, 2006a).

The original version of DE propose the following equation for velocity (Line 11 of the Pseudocode 7):

$$\mathbf{v}_i^{(t+1)} = \mathbf{x}_{best}^{(t)} + F(\mathbf{x}_2^t - \mathbf{x}_3^t) \tag{5.4}$$

where $\mathbf{x}_{best}^{(t)}$ is the best solution found, $\mathbf{x}_2^t$ and $\mathbf{x}_3^t$ are two different solution chosen randomly, and F is a parameter. And this equation is mutation operation for the metaheuristic. if a component

Algorithm 7 DE pseudocode

Require: F, cr, K, T
1: $t \leftarrow 2$;$cr = 0.5$;$F = 0.9$
2: **for** $k = 1$ to K **do**
3: $\mathbf{x}_k \leftarrow$ runif(N)
4: **end for**
5: **while** $t \leq T$ **do**
6: $\mathbf{x}_k \leftarrow \mathbf{s}_k$
7: $\mathbf{x}_{best} \leftarrow \arg\min\{f(\mathbf{x}_{best}), f(\mathbf{x}_k)\}$
8: **if** $\min\{f(\mathbf{x}_k), \forall k\} \leq f(\mathbf{x}_G)$ **then**
9: $\mathbf{x}_G \leftarrow \arg\min\{f(\mathbf{x}_k), \forall k\}$;
10: **end if**
11: $\mathbf{v}_k$ % velocity of calculation agree with algorithm version.
12: $\mathbf{s}_k$ % New population is the result of crossover and mutation
13: $t \leftarrow t + 1$
14: **end while**
15: **return** $\mathbf{x}_G$;$f(\mathbf{x}_G)$

uses or not the mutation (there is a change in the individual or not) is defined as crossover for this algorithm. Then, new population are determined by the following equation (Line 12 of the Pseudocode 7):

$$\mathbf{s}_{j,i}^{t+1} = \begin{cases} \mathbf{v}_{j,i}^{t+1} & \text{if } rand(1) < Cr \text{ or } i = j \\ \mathbf{x}_{j,i}^{t} & \text{else} \end{cases} \tag{5.5}$$

where sub-index j denotes each component of the solution vector and Cr is a parameter called crossover parameter. As we mention above, both operators are non separable as a individual operators as in other algorithms as GA or ES. Then both parameter affect the general behavior. In any case, we can construct the probability region. In this case, the probability of the new population have specific points given by F and the Cr. In Figure 5.4, the gray points represent the possible solutions for next generation from $\mathbf{x}_i$, which is the black point in Figure. The other three points $\mathbf{x}_1$,$\mathbf{x}_2$, $\mathbf{x}_3$ are solutions from current generation and were chosen randomly. In the algorithm, if F increase, the diversity in terms of distance between solutions increase and *vice versa*. The another parameter Cr is a ratio between 0 and 1, when $Cr = 0$ there are only one change in each indivitual while $Cr = 1$ all crossover are generated for all components. Both limits for Cr reduce the exploration over search space changing the level of the parameter from component to individual.

SaDE

Several approaches has been proposed from the original version of DE. These new versions are different in the crossover and mutation. The principal advantage of this version (SaDE) is the self-adaptation of the parameters. It was proposed by Qin and Suganthan in 2005 (Qin & Suganthan, 2005). This version combines two different versions of the mutation for the algorithm using a learning process that control the parameters for the algorithm.

The original version of DE was called as: *rand/1/bin*. Rand means that solutions for the new

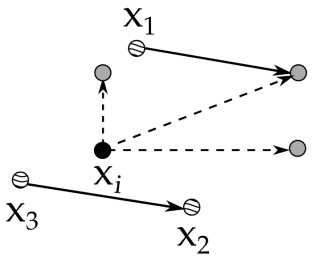


Figure 5.4: For given factors, possible solutions for the new population for original version of DE

population are chosen randomly. The number one means you must only have one vector $F(\mathbf{x}_2^t - \mathbf{x}_3^t)$ and bin denotes binomial crossover. In this case, the parameter for the crossover is chosen in a tournament. There are another option which is exp in this case, the tournament, but also is function of iterations. In this work we only use the versions bin. The most popular version are: the original, *best/1/bin*, *rand/2/bin best/2/bin*. When the version is denoted with the word best, that means a elitism strategy. While rand version uses 3 solutions chose randomly, best version uses the best solution and $\mathbf{x}_2$ and $\mathbf{x}_3$. For the number 2, we use the whole five solutions chose randomly or the best and 4 chose randomly.

SaDE use two operation of mutation, *rand/1/bin* and *best/2/bin*. Also, this version increase the diversity with a random F, the parameter will be distribute normally with $x = 0.5$ and $\sigma = 0.3$ for the range $(0, 2]$. The dynamical parameters allocated in the level of component made SADE one of the best algorithms in terms of quality.

5.2.4 ES vs. CMA-ES

Genetic Algorithms and ES are the first approaches to population metaheuristics. Proposed in Germany in the 60's by Rechenberg and deeply modified in the 70's by Schwefel (Beyer & Schwefel, 2002), ES is one of the most efficient and known metaheuristic. The original algorithm proposed in the 60's was individual algorithm, in the 70's the population was proposed. The initials modifications of ES increase the exploration of the algorithm implemented a population. The population evolved using two mechanisms mutations and crossover.

The parameters of the algorithm are: the number of children (λ) and the number of the parents (μ), the operator comma(,) or plus (+) for the selection, σ and ξ are deviation and mean respectively of a normal distribution and are linked with mutation operator. c is a constant factor to increase or reduce the deviation σ agree with the successful of the mutation (Beyer & Schwefel, 2002).

ES

In the pseudocode 8, we present the version (μ+λ)-ES. The initial population is random and uniformly distribute over search space. Afterwards, we apply mutation operator and later the crossover. For the crossover, the parents are chosen using parents and children of the generation because + operator. The original algorithm uses the randomly parents, some elitism was included using the best i. Crossover have a lot versions, using the mean between both solutions using one or another, among others. In Figure 5.5, the line pointed with crossover is the probably line for a child between $\mathbf{x}_1$ and $\mathbf{x}_2$. Mutation is a "perturbation" of the solution where $\mathbf{x}_1$ mutation is

Algorithm 8 ES pseudocode (μ+λ)

Require: σ, ξ, λ, μ, T
$t \leftarrow 1$;
$\mathbf{x}_k = rand(n) \forall k \in \{1, ..., \lambda\}$;
while $t \leq T$ **do**
  $f(\mathbf{x}_k) \forall k \in \{1, ..., \lambda\}$;
  $\mathbf{x}_G = \text{ARGMIN}(f(\mathbf{x}_k) \forall k$;
  $\mathbf{s}_k = \mathbf{x}_k + RandN(0, \sigma) \forall k \in \{1, ..., \lambda\}$;
  if $f(\mathbf{x}_k) < f(\mathbf{s}_k)$ **then**
    $\mathbf{s}_k = \mathbf{x}_k$;
  end if
  choose μ solutions randomly from parents chosen and previous children;
  crossover λ to obtain solutions from the parents chosen;
  select the best solutions for a final $\mathbf{s}_k$
  $\mathbf{x}_k = \mathbf{s}_k$
end while
return $\mathbf{x}_G; f(\mathbf{x}_G)$

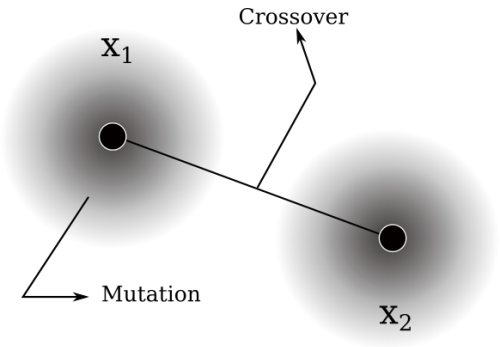
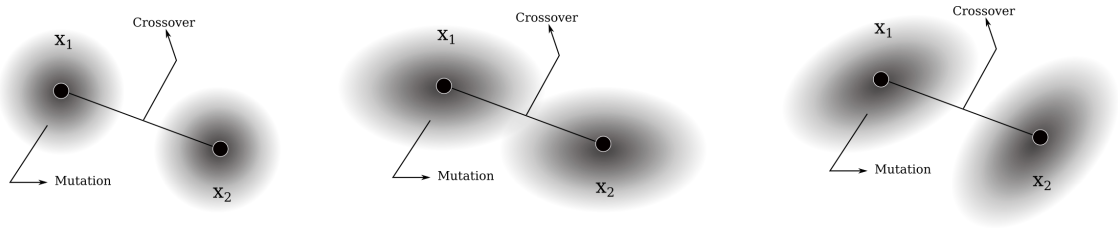


Figure 5.5: For given factors, possible solutions for the new solution of the population for (μ + λ)-ES

$\mathbf{x}'_1 = \mathbf{x}_1 + randN(0, \sigma)$. The new individual is part of the population if its fitness is better than non mutate solution. the deviation σ changes, as we mention in the parameters description, with the successful mutation. In figure 5.5, the gray circles around solutions are the probable are for new individual after mutation. Then, the line and circles are the exploration area for next generation solutions given $\mathbf{x}_1$ and $\mathbf{x}_2$.

CMA-ES

The version called CMA-ES changes the type of the random distribution used. The original version of ES uses normal distribution while CMA-ES uses multivariate normal distribution, which is the main difference. with this particular change, CMA-ES tries to emulate a gradient, using the correlation matrix as the approximation of a kind of Hessian (Hansen, 2016). Parameters does not change in quantity but in formate i.e. while in the original version we have a mean, CMA-ES use



(a) Normal distribution

(b) Multivariate normal distribution without correlation

(c) Multivariate normal distribution with correlation

Figure 5.6: For given factors, possible solutions for the new solution of the population for (CMA-ES

a vector of means.

The changes mention above has a interesting results in terms of the next generation probability. In Figure 5.6 part (a), the mutations is a perfect circle. In terms of geometry, the projection of iso-density line changes in a multivariate normal distribution because there is not a unique mean or deviation is a vector of them (see Figure 5.6 part (b)). The shape of the level for the same value change from a circle to a ellipse where de axis directions are related with correlation. For a matrix of low correlation(separable functions) the axis of iso-density ellipsoid will be parallel to the axis of the variables. If variables are highly correlated the axis of the new ellipsoid emulate to a gradient calculated by the Hessian (Bäck, 1996)(see Figure 5.6 part (c)). Most of the times, the parameters introduced in this version, increase the area from the original circle, but the correlate matrix introduce a indirect exploitation in the algorithm, which induces similarity into the population. This is a particular case in the literature, where the parameters and strategy are mixed efficiently to produce more exploration and exploitation.

5.3 Conclusions

Previously, we follow the most successful modifications for known metaherustics. Although, the three metaheuristic described are different, there are some common point in the modifications.

- The parameters change level to increase the control of the balance between exploration and exploitation.
- The improvement of the performance are related with the control and equilibrium of the diversity and similarity.
- Theoretically it is clear how the parameter must to be control, if we know the objective function. The objective function is unknown in advance. Then, parameters are controlled randomly or as a function of the iterations using the concept of asymptotic behavior.

6. SUPERNOVA 2.0

" Would you tell me, please, which way I ought to go from here?" "That depends a good deal on where you want to get to," said the Cat. "I don't much care where –" said Alice. "Then it doesn't matter which way you go," said the Cat."

– Lewis Carroll, Alice in Wonderland

The past five chapters give a solid background to propose a new version of supernova. In the first chapter, the weakness was described and the focus of this work was defined. In the second chapter the three main operators, Selection, gravity and impulse, was characterized. In the third chapter, a version with a LDS initialization was proposed as a general improvement for the algorithm. In the fourth chapter, a criterion to identify planar regions is presented. Finally, in the fifth chapter, a discussion about parameters their characteristics from other metaheuristics and their improvements is reported. A new version of the algorithm Supernova, called Supernova 2.0, is proposed based on the previous chapters. This version tries to reduce one of the weaknesses mentions in chapter one, the planar regions. Self-adaptive parameters that follows a topology criterion and LDS start are the main innovations developed for this new version. The results show a reduction of iterations for some benchmark functions and minimum significantly lower than first version.

In the first section of this chapter, we present the modifications made to the original algorithm. In the second section, we present the implementation of the algorithm and the first empirical convergence tests. In the third section, we present the results of Supernova 2.0 for well-known benchmark functions and the results of the same functions for two popular metaheuristics and original version of Supernova. Finally, we discuss the results found.

6.1 Algorithm Modification

In this section, the modification made to the original algorithm is presented. The changes are described for each operation mentioned in the second chapter: gravity, impulse and selection. Finally, how parameters and modification of the operations can affect the balance between exploration and exploitation of the algorithm is analyzed (Liu et al., 2013). For the gravity operation, the calculation of the distance was modified. For selection, a non-elitism procedure is proposed. Last but not least meaning, the parameters are adapted for following the topology criterion proposed in chapter 4.

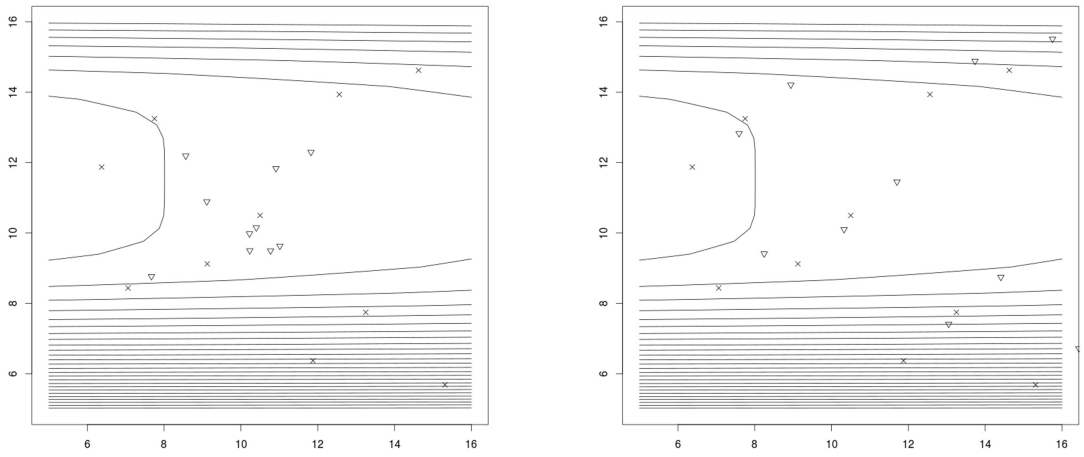


Figure 6.1: comparison in a 10 individual populations between original gravity operation and new version of gravity operation

6.1.1 Modification to Gravity operation

The original operations of Supernova, described in the Chapter 2, work properly for convex functions, but not for planar regions as we described in Chapter 1. The original version of the operation called gravity implies a high computational effort. The first modification to this operation is related with reduction of the computational effort keeping the same performance of the operator.

The high computation effort is introduced by distance calculation in the original gravity operation. For this second version, the distance vectors are not calculated. Instead, inertial momentum is calculated for the different particles involved. In Figure 6.1 part (a), we show the behavior of a random population with original operation. Triangles represent the new population and x represent the initial population. Also, in Figure 6.1 part (b), we present the identical population for new same function and found the next population obtained by inertial momentum calculation. The most important advantage of this change is the reduction of algorithm's complexity from $O(n^3)$ to $O(n)$, without losing the diversity.

Algorithm 9 gravity operation

Require: x,$f(\cdot)$, G
1: **for** i=1 **to** i=K **do**
2: $\quad \mathbf{R}_{i+1,i} = \mathbf{x}_{i+1} - \mathbf{x}_i$
3: $\quad \mathbf{R}_{i,i+1} = -\mathbf{R}_{i+1,i}$
4: **end for**
5: $\mathbf{h}_k \leftarrow \sum_{\substack{i=1 \\ i \neq k}}^{K} G \cdot m_i \cdot \mathbf{R}_{i,1:k}$
6: $\mathbf{s}_k \leftarrow \mathbf{x}_k + \mathbf{h}_k$
7: **return** **s**

In the algorithm 9, the sequence of the new operation is presented. The unitary vector of

distance calculated in the original version is replaced by the difference between solutions, which is quite similar to inertial moment $I = mr^2$. We do not use the square of the distance because when the distances are less than 1 there is a distortion. The inertial moment is like a weight average where mass is the weight. That means, bigger mass implicates that the inertial moment will be near to this point. If the difference between the particles' masses is very large, the moment of inertia can be absorbed by a single particle which drastically reduces the exploration. For keeping the exploration, we proposed to rank masses and use this ordered integer index to find the moment of inertia. $m = rank_k(f(\mathbf{x}_k)$

6.1.2 Modification to Selection

In the previous version, selection was a memory, i.e. a vector keeps the best solution found by individual, which is important to guarantee an asymptotic behaviour for the error(Villalobos-Arias et al., 2005). In this version, we propose a random parameter to keep the best solution found for the individual or keep the new solution when this is worse than the best solution for the next iteration. This kind of selection was proposed for ES(Bäck, 1996). The elitism is very important to get the convergence to a particular solution, but when a unique best solution is used the algorithm can arrive to a local minimum. In the original version, the best solution is used for each iteration reducing the exploration of new regions.

6.1.3 Parameters

From previous version, we keep the parameters: size of population, number of iterations, number of restarts, F and G and the new parameter to determine when keep the best solution or use the worst that we call ω. Furthermore, the last three will be controlled by the topology of the region found by the criterion described in Chapter 4. Notice that for planar regions, the exploration should be increased. How parameters increase or reduce the exploration is discussed in chapter 5, and the modifications to the Supernova algorithm's parameters are shown in this section.

The criterion propose found two values from particles, the first one is:

$$R_g = \frac{f(\mathbf{x})_{max} - f(\mathbf{x})_{min}}{\|\mathbf{ub} - \mathbf{lb}\|} \tag{6.1}$$

That is related with Range of objective function's values. The second one is:

$$T_C = \frac{f(\mathbf{x})_{mean} - f(\mathbf{x})_{median}}{f(\mathbf{x})_{max} - f(\mathbf{x})_{min}} \cdot 100 \tag{6.2}$$

This is the difference between the mean and median of the objective function's values.

The Table 4.2 shows the possible scenarios for R_g and T_c for populations between 30 and 100 individuals. Agree with this criterion, the algorithm uses parameters for planar regions and other parameters for convex functions. The next parameters are taken from the original parameters (See Chapter1) and do not change for the different kind of topologies: population $K = 50$, restarts $R = 50$, maximum iterations $T = 500$ and $\alpha = 1.1$. The initial λ_0 is the range of the variables. The other two parameters and the new selection parameter change for planar or convex regions, and start with the begining values: $F = G = 1$ and $\omega = 0.5$. The criterion is described in the algorithm 5 the parameters will modify from the initial values to $F = G = 2$ and $\omega = 0.2$ for non-planar regions like sphere function and $F = G = 0.5$ $\omega = 0.8$ for planar regions.

The main modification for parameters is a level change. For planar regions, a new configuration for the parameters is added. F and G are individual-level parameters for previous version of the meta heuristic. From chapter 5 discussion, we propose to change both parameters from individual to component level, i.e. the parameter will be used only in some component randomly chosen. This increase the exploration of the algorithm. For example, we think in 2D function, when F and G are individual-lever parameters the options for the new solution is a line while both are component-level parameters the line become a square of solutions (See Figure 5.2).

6.2 Results

The error medians for the set of 28 benchmark function in 10 dimensions, known as a CEC 2013 test, are presented in Table 6.1. Each column is the result of a metaheuristic. SaDE and DE are versions of differential evolution, which are the best results reported in the literature for the set of benchmark functions chosen. And SaDE is included because this is the standard version of self-adaptive algorithm of DE. The finest results for a similar algorithm are SPSO2011. SPSO2011 is a standard version of PSO proposed by Clerc and the result are significantly better than the original version of PSO. The results for Supernova and Supernova 2.0 are unedited while the results for the other algorithms are from (Zambrano-Bigiarini et al., 2013; Qin & Li, 2013; Qin, Li, Pan, & Xia, 2013) respectively. The algorithms were used for previous comparisons in this work. Between the original Supernova proposed in 2010 and Supernova 2.0 there is a version which was presented in the Chapter 3. This version is a deterministic version that works better than original version presented in Chapter 1 and is the one used in this comparision.

For all algorithms, except supernova, the median is calculate from 50 independent runs. Supernova is a unique value because it is deterministic algorithm. The parameters for each algorithm are SaDE and snova 2.0 are self-adaptive, DE(K=50,CR=0.9 and F=0.5), SPSO2011 (K=40,informants:3, $c_1 = c_2 = 0.5 + ln(2)$,$\omega = \frac{1}{2*ln(2)}$) and Supernova (K=50,$F = G = 4$,$\alpha = 1.5$,$\eta = 0.15$).

The initial hypothesis is $H_0 : \overline{f^*}_{supernova2.0} = \overline{f^*}_{DE} = \overline{f^*}_{SADE} = \overline{f^*}_{SPSO2011} = \overline{f^*}_{Supernova}$ and $H_a : \overline{f^*}_i \neq \overline{f^*}_{i'}$ for at least one of the algorithms. To check this hypothesis, the Friedman's test will be used. For both tests, the rank for ties was calculated as the average. In the Table 3.6, the result for the Friedman's test is shown. The P-values are inferior to 0.5, then the H_0 is rejected. This means that at least one of the algorithm behaviour is different from others.

The initial hypothesises are $H_0 : \overline{f^*}_{snova2.0} = \overline{f^*}_i$ and $H_a : \overline{f^*}_i \neq \overline{f^*}_{i'}$ where i is each algorithm, in this case: Supernova, DE, SaDE, SPSO2011. The median of the i algorithms are compared with the new version of Supernova called Supernova 2.0. The Wilcoxon test is non-parametric test to determine if there is a significant difference between medians, for this test we choose one tailored test and positive rank indicates a superior performance of the Supernova 2.0. In Table 6.3, the rows are the positive and negative rank, the columns are the algorithms. In the last row the level of the test for $\alpha = 0.01$ is given. Agree with the level, we can conclude that medians from original version and second version of supernova are significant different. The positive ranking is greater which means that Supernova 2.0 is significantly better than Supernova. For SPSO2011 and Snova 2.0 the H_0 is not rejected which means the performance is non significantly different. Finally for DE and SaDE, the negative ranking is greater and the minimum is smaller than 101 which means both algorithm have a better performance than Supernova 2.0.

The test was running in a processor intel core i7, 2.36 Ghz and 8 cores. The time depends of the kind of objective function, the first functions are faster because the calculation of the objective

Table 6.1: Results and comparison for benchmark functions, CEC 2013, between Supernova v.1, v.2, DE and PSO. 10 dimension and 50 independent runs

f	Error SPSO2011	Error DE	Error CDE	Error Supernova	Error Supernova 2.0
f_1	$\mathbf{0.00 \times 10^{+00}}$	1.00×10^{-08}	1.00×10^{-08}	$1.23 \times 10^{+03}$	$5.70 \times 10^{+02}$
f_2	$3.63 \times 10^{+04}$	$2.85 \times 10^{+02}$	$\mathbf{1.00 \times 10^{-08}}$	$3.59 \times 10^{+06}$	$2.47 \times 10^{+06}$
f_3	$2.68 \times 10^{+05}$	3.11×10^{-01}	$\mathbf{4.23 \times 10^{-03}}$	$1.07 \times 10^{+09}$	$1.24 \times 10^{+09}$
f_4	$8.87 \times 10^{+03}$	$1.92 \times 10^{+00}$	$\mathbf{3.91 \times 10^{-07}}$	$2.58 \times 10^{+03}$	$4.04 \times 10^{+03}$
f_5	$1.19 \times 10^{+03}$	$\mathbf{1.00 \times 10^{-08}}$	$\mathbf{1.00 \times 10^{-08}}$	$1.58 \times 10^{+02}$	$8.88 \times 10^{+01}$
f_6	$9.80 \times 10^{+00}$	$\mathbf{7.10 \times 10^{-01}}$	$9.81 \times 10^{+00}$	$6.23 \times 10^{+01}$	$5.43 \times 10^{+01}$
f_7	$2.11 \times 10^{+01}$	$\mathbf{8.41 \times 10^{-06}}$	7.06×10^{-03}	$6.21 \times 10^{+01}$	$3.69 \times 10^{+01}$
f_8	$2.03 \times 10^{+01}$	$2.04 \times 10^{+01}$	$2.04 \times 10^{+01}$	$\mathbf{2.01 \times 10^{+01}}$	$\mathbf{2.01 \times 10^{+01}}$
f_9	$4.80 \times 10^{+00}$	$\mathbf{1.03 \times 10^{+00}}$	$1.11 \times 10^{+00}$	$6.53 \times 10^{+00}$	$6.16 \times 10^{+00}$
f_{10}	3.00×10^{-01}	4.92×10^{-02}	$\mathbf{2.46 \times 10^{-02}}$	$6.30 \times 10^{+01}$	$6.07 \times 10^{+01}$
f_{11}	$1.09 \times 10^{+01}$	9.95×10^{-01}	$\mathbf{1.00 \times 10^{-08}}$	$6.23 \times 10^{+01}$	$5.44 \times 10^{+01}$
f_{12}	$1.39 \times 10^{+01}$	$6.96 \times 10^{+00}$	$\mathbf{4.16 \times 10^{+00}}$	$6.79 \times 10^{+01}$	$1. \times 10^{+01}$
f_{13}	$2.08 \times 10^{+01}$	$1.18 \times 10^{+01}$	$\mathbf{4.87 \times 10^{+00}}$	$6.47 \times 10^{+01}$	$3.22 \times 10^{+01}$
f_{14}	$8.34 \times 10^{+02}$	$7.48 \times 10^{+01}$	$\mathbf{1.00 \times 10^{-08}}$	$1.17 \times 10^{+03}$	$1.04 \times 10^{+03}$
f_{15}	$7.74 \times 10^{+02}$	$1.19 \times 10^{+03}$	$7.41 \times 10^{+02}$	$1.02 \times 10^{+03}$	$\mathbf{7.02 \times 10^{+02}}$
f_{16}	$\mathbf{5.00 \times 10^{-01}}$	$1.03 \times 10^{+00}$	$1.11 \times 10^{+00}$	6.18×10^{-01}	5.88×10^{-01}
f_{17}	$1.89 \times 10^{+01}$	$1.67 \times 10^{+01}$	$\mathbf{1.01 \times 10^{+01}}$	$8.50 \times 10^{+01}$	$6.14 \times 10^{+01}$
f_{18}	$\mathbf{1.78 \times 10^{+01}}$	$3.12 \times 10^{+01}$	$2.28 \times 10^{+01}$	$9.15 \times 10^{+01}$	$5.24 \times 10^{+01}$
f_{19}	9.00×10^{-01}	9.74×10^{-01}	$\mathbf{3.94 \times 10^{-01}}$	$1.02 \times 10^{+01}$	$1.38 \times 10^{+01}$
f_{20}	$3.40 \times 10^{+00}$	$2.33 \times 10^{+00}$	$\mathbf{2.20 \times 10^{+00}}$	$2.87 \times 10^{+00}$	$3.57 \times 10^{+00}$
f_{21}	$\mathbf{4.00 \times 10^{+02}}$	$\mathbf{4.00 \times 10^{+02}}$	$\mathbf{4.00 \times 10^{+02}}$	$4.40 \times 10^{+02}$	$4.20 \times 10^{+02}$
f_{22}	$9.06 \times 10^{+02}$	$1.16 \times 10^{+02}$	$\mathbf{8.84 \times 10^{+00}}$	$1.18 \times 10^{+03}$	$5.02 \times 10^{+02}$
f_{23}	$9.10 \times 10^{+02}$	$1.01 \times 10^{+03}$	$6.59 \times 10^{+02}$	$1.30 \times 10^{+03}$	$\mathbf{6.50 \times 10^{+02}}$
f_{24}	$2.14 \times 10^{+02}$	$2.05 \times 10^{+02}$	$2.00 \times 10^{+02}$	$\mathbf{1.67 \times 10^{+02}}$	$1.94 \times 10^{+02}$
f_{25}	$2.09 \times 10^{+02}$	$2.00 \times 10^{+02}$	$2.00 \times 10^{+02}$	$1.68 \times 10^{+02}$	$\mathbf{1.55 \times 10^{+02}}$
f_{26}	$2.00 \times 10^{+02}$	$2.00 \times 10^{+02}$	$\mathbf{1.06 \times 10^{+02}}$	$1.70 \times 10^{+02}$	$1.69 \times 10^{+02}$
f_{27}	$3.36 \times 10^{+02}$	$\mathbf{3.00 \times 10^{+02}}$	$\mathbf{3.00 \times 10^{+02}}$	$5.88 \times 10^{+02}$	$4.54 \times 10^{+02}$
f_{28}	$\mathbf{3.00 \times 10^{+02}}$	$\mathbf{3.00 \times 10^{+02}}$	$\mathbf{3.00 \times 10^{+02}}$	$4.50 \times 10^{+02}$	$5.12 \times 10^{+02}$

Table 6.2: Results for Friedman test for Supernova 2.0, supernova, SaDE DE and PSO

	Values
SPSO2011	3.05
DE	2.5
SADE	1.84
SUPERNOVA	4.14
SNOVA 2.0	3.4
F_F	32.2071
P-value	1.85×10^{-07}
F_{ID}	10.8981
P-value	1.74×10^{-06}

Table 6.3: Results for Wilcoxon's test for SPSO2011, DE, SaDE and supernova

	Snova2.0-SPSO2011	Snova2.0-DE	Snova2.0-SaDE	Snova2.0-Supernova
Positive Ranking	158	78	43	307
Negative Ranking	248	328	363	99
Level for $\alpha = 0.01$ one tailored=101				

function are easy and make complex to the last function. Taking a model the funtion f_8 for 10 dimenstion and 1000 iteration the average time was 3.87 seconds.

6.3 Conclusion

In this chapter, we presented a new version of Supernova using the criterion presented in chapter four for identify planar regions and self-adapt the parameters of the new version called supernova 2.0. This criterion was used in PSO to control its parameters sucessfully. The criterion proposed use a central tendecy statistic measures and works for population between 40 and 100. This criterion is alredy published in the contributions presented in the Chapter 1.

In this Chapter, the Friedman's and Wilcoxon's testes showed that the results of Supernova 2.0 are significantly better than the original Supernova. This means that the new version of the algorithm does not degenerate the performance of previous version of the algorithm. Furthermore, The performance of Supernova 2.0 is similar to SPSO2011, which is a well-known and older algorithm.

Besides the numerical results, we focus in the analysis of exploration and exploitation balance in the search, How they affect the search in planar regions and how parameters controlling this balance. These thoughts are general understanding of the direct methods of optimization and can be used for improving any metaheuristic. Furthermore, the criterion to identify a planar region can be used for any metaheuristic or to choose a different metaheuristics or classical methods to start new search. Finally, the self-adaptation of the parameters using the information about topology of the goal function given by population can be extended to other methods.

For future work, the self-adaptation of the parameters can be improved. The metaheuritics can be combined with classical methodologies to improve the result and guarantee the local minimum using the criterion proposed.

7. CONCLUSIONS

The six chapters of this work summarize the improvements made to Supernova. In chapter one, the advantages and disadvantages found are listed and compared with other known metaheuristics. Among the advantages, we found the few iterations necessary for descending convex functions, and the quality of the answers found. The most marked disadvantages were the time spends for problems with many variables and the performance of the algorithm in planar regions or needle minimum. These problems are common with other optimization methods. When finished the analysis of the first version of the algorithm, general objective was defined as:

> "To design and to validate a new version of Supernova based on the development of a new strategy specialized in planar regions, and a mechanism for switching between specialized algorithms for descendent and planar regions, such that there is an improvement on the behaviour for planar regions without degenerating the current behaviour for descendent regions."

The following five chapters record the advances made in the research until Supernova 2.0.

Often, the populations of metaheuristics have between one to hundred individuals because the size of the population is related with computational. In many cases, the uniform distribution is used to estimate the first population, but the spaces between points are large. For direct methods, the points of the first population can bias the search to one area to another one. Then, the initial population should be evenly as possible. Low discrepancy sequences are not random sequences, but the constructions of this sequences guarantee a similar space between points even for small population. In chapter three, the original version of the algorithm is compared with version with LDS. The implementation with LDS is significantly better than previous version. However, there is not significantly better for problematic functions.

A criterion to identify planar regions is one of the specific objectives. In chapter four, a general criterion is presented; it uses the difference between measures of central tendency of the population as an indicator to increase or decrease the diversity of the search. Regularly, the best solution, worst and random solutions are used to guide the search. In this case, the measures of central tendency take into account the information found for whole population for controlling parameters for increasing or decreasing the diversity or intensity adapting the algorithm to the problem.

In chapter five, a discussion about how parameters can be self-adapted successfully is presented. The examples are limited, but they help to understand how to improve known metaheuristics and plan strategies for controlling parameters. The most common strategy is changing parameters randomly when the search meets certain requirements, e.g. 1/5 rule. Also, in different new version of the algorithm change the level of the parameters as manner improve the performance of the metaheuristics.

Finally, Supernova 2.0 is presented in chapter six. In this chapter, the progress of the research is collected and summarized. From chapter two the operations of the new version are included. . From chapter three the use of low discrepancy sequences to estimate initial population. the criterion for controlling intensity and diversity of the search is implemented for this version. Also, Version 2.0 includes self-adaptation of the parameters. In conclusion, the version presented in chapter six is a self-adapted metaheuristic that use the central tendency measures for controlling the parameters. Supernova 2.0 was tested using the 28 benchmark functions from CEC2013 and compared with the original version, SPSO2011, DE and SADE. In general, Supernova 2.0 was significantly better than original version. There is no significantly difference with SPSO2011 and work significantly worse that DE and SADE.

As future work, the advances achieved by this research are associated with supernova, but they can be used to improve other methods. Also, the criterion can be used to choose methods in different points of the search for example change to a gradient method. The process can be extent to self-adapt the parameters of other metaheuristics. The computational efficiency of the method can also be improved by parallelizing it. Also, the algorithm can be extended to combinatorial problems. The statistic measures of the population can be explore and extended and included inside the operation of other metaheuristics.

BIBLIOGRAPHY

(2006). *Particle swarm optimization*. ISTE.

Angeline, P. (1995). Adaptative and Self-Adaptative Evolutionary Computations. In *IEEE, Computational Intelligence*, pp. 152–163. IEEE.

Bäck, T. (1996). *Evolutionary Algorithms in Theory and Practice*. Oxford University press.

Beyer, H. G., & Schwefel, H. P. (2002). Evolution strategies. *Natural computing*, *1*(1), 3–52.

Brest, J., Greiner, S., Boskovic, B., Mernik, M., & Zumer, V. (2006). Self-Adapting Control Parameters in Differential Evolution: A Comparative Study on Numerical Benchmark Problems. *IEEE Transactions on Evolutionary Computation*, *10*(6), 646–657.

Brooks, S. H. (1959). A Comparison of Maximum-Seeking Methods. *Operations Research*, *7*(4), 430–457.

Chen, G., Huang, X., Jia, J., & Min, Z. (2006). Natural Exponential Inertia Weight Strategy in Particle Swarm Optimization. In *2006 6th World Congress on Intelligent Control and Automation*, Vol. 1, pp. 3672–3675.

Christophe, D., & Petr, S. (2015). *randtoolbox: Generating and Testing Random Numbers*. R package version 1.17.

Coello, C. (2005). *Evolutionary Multiobjective Optimization Theoretical Advances And Applications*, chap. Recent Trends in Evolutionary Multiobjective Optimization, pp. 7–32. Springer-Verlag.

Coello, C. A. (2002). Theoretical and Numerical Constraint-Handling Techniques used with Evolutionary Algorithms: A Survey of the State of the Art. *Computer Methods in Applied Mechanics and Engineering*, *191*(11-12), 1245–1287.

Corne, D., Dorigo, M., & Glover, F. (1999). *New Ideas in Optimization*. McGraw Hill.

Dasgupta, S., Das, S., Abraham, A., Member, S., & Biswas, A. (2009). Adaptive Computational Chemotaxis in Bacterial Foraging Optimization : An Analysis. *IEEE Transactions on Evolutionary Computation*, *13*(4), 919–941.

De Jong, K. (1975). *Analysis of the behavior of a class of genetic adaptative systems*. Ph.D. thesis, The University of Michigan.

Derrac, J., García, S., Molina, D., & Herrera, F. (2011). A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation*, *1*(1), 3–18.

Eberhart, R., & Kennedy, J. (1995). A new optimizer using particle swarm theory. In *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, pp. 39–43. IEEE.

Eiben, E., Hinterding, R., & Michalewicz, Z. (1999). Parameter control in evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, *3*(2), 124–141.

Erol, O. K., & Eksin, I. (2006). A new optimization method: Big bang-Big crunch. *Advances in Engineering Software*, *37*(2), 106–111.

Feoktistov, V. (2006a). *Differential Evolution*. Springer.

Feoktistov, V. (2006b). *Differential evolution: in search of solutions*, Vol. 5. Springer, New York.

Formato, R. A. (2007). Central force optimization: a new metaheuristic with applications in applied electromagnetics. *Progress In Electromagnetics Research*, *77*, 425–491.

Gendreau, M., & Potvin, J.-Y. (2010). Handbook of Metaheuristics. *Int. Ser. Oper. Res. Man*, *146*, 648.

Gentle, J. E. (2003). *Random number generation and Monte Carlo methods*, Vol. 4 of *Statistics and computing*. Springer.

Glover, F., Ching-Chung, K., & Dhir, K. S. (1995a). A discrete optimization model for preserving biological diversity. *Applied Mathematical Modelling*, *19*(11), 696–701.

Glover, J. F., Kelly, J., & Laguna, M. (1995b). Genetic algorithms and tabu search: hybrids for optimization. *computer Operation Research*, *22*(1), 111–134.

Goldberg, D. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley.

Hansen, N. (2016). The CMA Evolution Strategy: A Tutorial. *CoRR*, *abs/1604.00772*.

Hendrix, E., & Toth, B. (2010). *Introduction to nonlinear and global optimization*. Springer, London.

Himmelblau, D. (1972). *Applied Nonlinear Programming*. McGraw hill.

Hinterding, R., Michalewicz, Z., & Eiben, A. (1997). Adaptation in evolutionary computation: a survey. In *Proceedings of 1997 IEEE International Conference on Evolutionary Computation (ICEC '97)*, pp. 65–69. Ieee.

Hirsch, M., Meneses, C., Pardalos, P., & Resende, M. (2006). Global optimization by continuous GRASP. *Optimization letters*, *1*(2), 201–212.

Holland, J. H. (1992). *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. MIT Press, Cambridge, MA, USA.

Karaboga, D., & Akay, B. (2009). A comparative study of Artificial Bee Colony algorithm. *Applied Mathematics and Computation*, *214*(1), 108–132.

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. In *IEEE International Conference on Neural Networks, 1995*, Vol. 11, pp. 1942–1948.

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by Simulated Annealing. *Science, New Series*, *220*(4598), 671–680.

Knuth, D. E. (1998). *The Art of Computer Programming: Seminumerical Algorithms* (3rd edition)., Vol. 2. Addison-Wesley, Massachusetts.

Kramer, O. (2008). *Self-adaptative Heuristics for Evolutionary Computation* (First Ed. edition). springer, Berlin.

Liang, J. J., Qin, A., Suganthan, P., & Baskar, S. (2006). Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. *IEEE Trans. Evol. Comput.*, *10*(3), 281–295.

Lieddle, A. (2003). *Introduction to Modern Cosmology*. Wiley.

Liu, S.-H., Mernik, M., Hrnčič, D., & Črepinšek, M. (2013). A parameter control method of evolutionary algorithms using exploration and exploitation measures with a practical application for fitting Sovova's mass transfer model. *Applied Soft Computing*, *13*(9), 3792–3805.

Macready, W. G., & Wolpert, D. H. (1996). What makes an optimization problem hard?. *Complex.*, *1*(5), 40–46.

Matallana, L. G., Blanco, A. M., & Bandoni, J. A. (2010). Estimation of domains of attraction: A global optimization approach. *Mathematical and Computer Modelling*, *52*(3-4), 574 – 585.

Mesa, E., Velasquez, J., & Jaramillo, P. (2015). Evaluation and implementation of heuristic algorithms for non-restricted global optimization. *Latin America Transactions, IEEE (Revista IEEE America Latina)*, *13*(5), 1542–1549.

Mesa, E. (2010). *Supernova : un algoritmo novedoso de optimización global*. Msc. thesis, National University.

Mezura-Montes, E., Velázquez-Reyes, J., & Coello Coello, C. (2006). A comparative study of differential evolution variants for global optimization. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pp. 485–492. ACM.

Michalewicz, Z., & Schmidt, M. (2002). *Evolutionary Optimization*, chap. Evolutiona. Kluwer Academic Press.

Mirjalili, S., & Hashim, S. (2010). A new hybrid PSOGSA algorithm for function optimization. In *Computer and Information Application (ICCIA), 2010 International Conference on*, pp. 374 –377.

Mohd, I. B. (2000). Identification of region of attraction for global optimization problem using interval symmetric operator. *Applied Mathematics and Computation*, *110*(2-3), 121–131.

Niederreiter, H. (1992). *Random Number Generation and Quasi-Monte Carlo Methods*. SIAM.

Olariu, S., & Zomaya, A. Y. (2006). *Handbook of bioinspired algorithms and applications*. Chapman & Hall/CRC computer and information science series. Chapman & Hall/CRC.

Omran, M., al Sharhan, S., Salman, A., & Clerc, M. (2013). Studying the effect of using low-discrepancy sequences to initialize population-based optimization algorithms. *Computational Optimization and Applications*, *56*(2), 457–480.

Pant, M., & Et, A. (2011). DE-PSO: a new hybrid meta-heuristic for solving global optimization problems. *New Mathematics and Natural Computation*, *7*(3), 363–381.

Papageorgiou, A., & Traub, J. F. (1997). Faster Evaluation of Multidimensional Integrals. *Comput. Phys.*, *11*(6), 574–578.

Price, K. (1996). Differential evolution: a fast and simple numerical optimizer. In *Proceedings of North American Fuzzy Information Processing*, pp. 524–527. Ieee.

Qin, A. K., & Li, X. (2013). Differential evolution on the CEC-2013 single-objective continuous optimization testbed. In *2013 IEEE Congress on Evolutionary Computation*, pp. 1099–1106.

Qin, A. K., Li, X., Pan, H., & Xia, S. (2013). Investigation of self-adaptive differential evolution on the CEC-2013 real-parameter single-objective optimization testbed. In *2013 IEEE Congress on Evolutionary Computation*, pp. 1107–1114.

Qin, A. K., & Suganthan, P. N. (2005). Self-adaptive differential evolution algorithm for numerical optimization. In *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, Vol. 2, pp. 1785–1791. IEEE.

Rashedi, E., Nezamabadi-pour, H., & Saryazdi, S. (2009). GSA: A Gravitational Search Algorithm. *Information Sciences*, *179*(13), 2232–2248.

Rothlauf, F. (2011). *Design of Modern Heuristics*. Natural Computing Series. Springer.

Schwefel, H. P. (1993). *Evolution and Optimun Seeking*. John Wiley and sons Inc.

Sen, S., Samanta, T., & Reese, A. (2006). Quasi- versus pseudo-random generators: discrepancy, complexity and integration-error based comparison. *International Journal of Innovative Computing, Information and Control*, *2*(3), 621–651.

Shang, Y.-W., & Qiu, Y.-H. (2006). A Note on the Extended Rosenbrock Function. *Evolutionary Computation*, *14*(1), 119–126.

Sheskin, D. J. (2003). *Handbook of parameter and nonparametric statistical procedures*. Chapman and hall/CRC.

Storn, R., & Price, K. (1997a). Differential Evolution - A simple evolution strategy for fast optimization. *Dr. Dobb's J. Software Tools*, *22*(4), 18–24.

Storn, R., & Price, K. (1997b). Differential Evolution: A Simple and Efficient Heuristic for global Optimization over Continuous Spaces. *Journal of Global Optimization*, *11*(4), 341–359.

Suganthan, P. N., Hansen, N., Liang, J. J., Deb, K., Chen, Y. P., Auger, A., & Tiwari, S. (2005). Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization. Tech. rep., Nanyang Technological University, Singapore.

Talbi, E. G. (2009). *Metaheuristics: from design to implementation*. Wiley.

Törn, A., Ali, M. M., & Viitanen, S. (1999). Stochastic Global Optimization : Problem Classes and Solution Techniques. *Journal of Global Optimization*, *14*(4), 437–447.

Torn, A., & Zilinskas, A. (1989). *Global optimization*, Vol. 350 of *Lecture Notes in Computer Science*. Springer-Verlag.

Črepinšek, M., Liu, S.-H., & Mernik, M. (2013). Exploration and Exploitation in Evolutionary Algorithms: A Survey. *ACM Comput. Surv.*, *45*(3), 35:1–35:33.

Villalobos-Arias, M., Coello, C. A. C., & Hernández-Lerma, O. (2005). *Asymptotic Convergence of Some Metaheuristics Used for Multiobjective Optimization*, pp. 95–111. Springer Berlin Heidelberg, Berlin, Heidelberg.

Webster, B., & Bernhard, P. (2003). A local search optimization algorithm based on natural principles of gravitation. Tech. rep. CS-2003-10, Florida Institute of Technology, Melbourne.

Wolpert, D. H., & Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE Transactions on evolutionary computation*, *1*(1), 67–82.

Xuan Hoai, N., Quang Uy, N., McKay, R., & Minh Tuan, P. (2007). Initialising \uppercase{PSO} with randomized low-discrepancy sequences. In *GECCO*, pp. 173–181.

Yang, X.-S. (2010). *Engineering Optimization An Introduction with Metaheuristics Applications*. Wiley, Hoboken.

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

Zaharie, D. (2002). Critical values for the control parameters of differential evolution algorithms. In *Proceedings of MENDEL*, Vol. 2002.

Zambrano-Bigiarini, M., Clerc, M., & Rojas, R. (2013). Standard Particle Swarm Optimisation 2011 at CEC-2013: A baseline for future PSO improvements. In *Evolutionary Computation (CEC), 2013 IEEE Congress on*, pp. 2337–2344.

Zanakis, S., & Evans, J. (1981). Heuristic "optimization": why, when and how to use it. *Interfaces*, *11*(5), 84–91.

Zhang, W.-J., & Xie, X.-F. (2003). DEPSO: hybrid particle swarm with differential evolution operator.. In *SMC*, pp. 3816–3821. IEEE.

Zhou, A., Qu, B.-Y., Li, H., Zhao, S.-Z., Suganthan, P. N., & Zhang, Q. (2011). Multiobjective evolutionary algorithms: A survey of the state of the art. *Swarm and Evolutionary Computation*, *1*(1), 32 – 49.