TECHNICAL UNIVERSITY OF CRETE
Department of Electrical and Computer Engineering

A Novel Meta-heuristic Search Algorithm for Global Continuous Optimization

Vasileios Lymperakis

Diploma Thesis

Thesis Committee
Georgios Chalkiadakis (Supervisor)
Athanasios Aris Panagopoulos (Co-Supervisor)
Vasileios Samoladas

Chania, September of 2021

Abstract

Artificial intelligence research in optimization and search is concerned with reaching the maxima or minima of an objective function, while potentially searching among a vast range of value choices for the function's variables. Global continuous optimization methods, in particular, seek to reach the optima of complex continuous mathematical functions. Meta-heuristics are commonly used in order to solve such problems. Typically, however, meta-heuristics originally designed for solving discrete optimization problems are later adapted to continuous tasks, which consumes considerable time. Also, there is a chance that they will get stuck to local optima as the complexity of configuration spaces increases. Furthermore, generally meta-heuristics accept worse solutions, in order to achieve a broader exploration of the configuration space. This results in algorithms that do not improve in an anytime manner, and an arbitrary interruption of the algorithm's flow, can lead to a waste of computation time as the current solution might be worse than a solution discovered earlier on.

In this work, a novel single-point meta-heuristic is proposed, which is specifically designed to tackle continuous optimization problems. Our algorithm, Buggy Pinball, is an anytime algorithm inspired by the well-known pinball game: It employs a trajectory-based search, and each proposed solution ensures the improvement of the configuration space. In order to evaluate our algorithm, we used a number of standard testbed functions, which can also be applied on multiple dimensions. We compared our results to the performance of some of the most widely used meta-heuristics, namely simulated annealing, threshold accepting, and particle swarm optimization. Our systematic evaluation shows that our algorithm is very efficient in global continuous optimization tasks, with performance that is particularly successful in complex configuration spaces.

Ένας εκ των παραδοσιακών πυλώνων της Τεχνητής Νοημοσύνης είναι αυτός που ασχολείται με τεχνικές αναζήτησης και βελτιστοποίησης, που επιχειρούν την εύρεση της βέλτιστης λύσης εντός ενός μεγάλου φάσματος επιλογών. Η καθολική συνεχής βελτιστοποίηση ασχολείται με την επίλυση προβλημάτων βελτιστοποίησης πολύπλοκων συνεχών συναρτήσεων. Οι μετα-ευρετικοί αλγόριθμοι χρησιμοποιούνται ευρέως για την επίλυση τέτοιων προβλημάτων. Οι περισσότεροι όμως μετα-ευρετικοί αλγόριθμοι, έχουν σχεδιαστεί για την επίλυση διακριτών προβλημάτων και αργότερα αναπροσαρμόστηκαν για τη χρήση τους σε συνεχή προβλήματα. Αυτό αυξάνει το χρονικό κόστος εξεύρεσης λύσης. Επίσης, η πιθανότητα να καταλήξουν σε τοπικά βέλτιστα αυξάνεται με την πολυπλοκότητα του χώρου. Ακόμη, τείνουν να αποδέχονται χειρότερες λύσεις, για να επιτύχουν πιο ευρεία εξερεύνηση του χώρου. Αυτό έχει σαν αποτέλεσμα να αποτελούν συνήθως μεθόδους αναζήτησης, οι οποίες δεν βελτιώνονται συνεχώς με την πάροδο του χρόνου. Ως εκ τούτου, μια πιθανή πρόωρη διακοπή της ροής του αλγορίθμου, μπορεί να καταστήσει μεγάλο μέρος της πρότερης υπολογιστικής διαδικασίας κενό νοήματος, αφού οι τελευταίες λύσεις μπορεί να είναι χειρότερες από την καλύτερη που έχει βρεθεί ως εκείνη τη στιγμή.

Στην παρούσα διπλωματική εργασία προτείνεται ένας νέος μετα-ευρετικός αλγόριθμος μονής-λύσης που είναι σχεδιασμένος για να λύνει συνεχή προβλήματα και που μπορεί να διακοπεί ανά πάσα στιγμή με την εγγύηση ότι η τελευταία λύση θα είναι πάντα καλύτερη από τις προηγούμενες. Ο αλγόριθμός μας, επονομαζόμενος "ελαττωματικό φλιπεράκι", είναι εμπνευσμένος από το διαδεδομένο παιχνίδι φλίπερ, όπου εφαρμόζεται ένας τρόπος εύρεσης με τη δημιουργία τροχιάς και κάθε καινούρια λύση βελτιώνει την ήδη υπάρχουσα. Αξιολογήσαμε τον αλγόριθμό μας σε πλείστες κλασσικές συναρτήσεις συνεχούς βελτιστοποίησης, και συγκρίναμε τις επιδόσεις του με αυτές κάποιων από τους πιο διαδεδομένους μετα-ευρετικούς αλγορίθμους αναζήτησης: την προσομοιωμένη ανόπτηση, την αποδοχή ορίου, και την βελτιστοποίηση σμήνους σωματιδίων. Η συστηματική διαδικασία αξιολόγησης που εφαρμόσαμε, αποδεικνύει την αποτελεσματικότητα του αλγορίθμου μας σε καθολικά συνεχή προβλήματα, και ιδιαίτερα την σημαντική υπεροχή του έναντι των ανταγωνιστών του ειδικά σε πολύπλοκους χώρους αναζήτησης.

Acknowledgments

I would like to first thank my professor, Athanasios Aris Panagopoulos, whose motivation and guidance from the first moments of our cooperation, made this thesis possible. His patience throughout our long overnight calls is also much appreciated.
I would like also to express my gratitude to my supervisor, Georgios Chalkiadakis, whose recommendations and expertise were vital from the very beginning of this project.
Last but not least, I would like to thank my family and friends who were supporting me throughout all my academic years. I would also like to specifically thank Anna Christopoulou for her consistent support, because without it, I would have never gotten this far.

Contents

List of figures

List of Algorithms

1 Introduction

Nowadays, we face various issues in many fields of our lives, that are usually hard or even impossible for humans to manually optimize. Over the years, we have managed to develop methods, that can approximate optimal solutions. Such techniques, namely *Optimization*, work towards finding the best choice among various solutions in a problem. Such problems can vary from finding the best route in order to travel in a set of cities, like the Traveling Salesman Problem[22], to minimize highly complex mathematical problems with a number of variables. In computer science, various methods have been developed, with the goal to tackle such problems. The approach of these techniques is to evaluate the objective function that translates into such a problem, and to explore the variables that define the function in various ways. A distinction between optimization problems is focused on the form of its variables, whether it is a countable or a non-countable set, which usually determines the optimization technique that will be used.

1.1 Continuous Optimization

Continuous optimization problems rely on optimization variables that draw their values from a non-countable set—typically a range of real numbers [29]. This is in contrast to discrete (combinatorial) optimization where the optimization variables draw values from a countable set. Many problems in various domains can be formulated and tackled as continuous optimization tasks. These range from image processing [6, 53] and chemical engineering [51, 49] to finance [41, 43, 47] and biology [48, 9]. Additionally, machine learning techniques heavily rely on continuous optimization to optimize internal parameters in order to, typically, minimize an error function [36]. As such, continuous optimization has drawn considerable attention over the years[20].
Continuous optimization methods can be widely classified to either local or global ones.

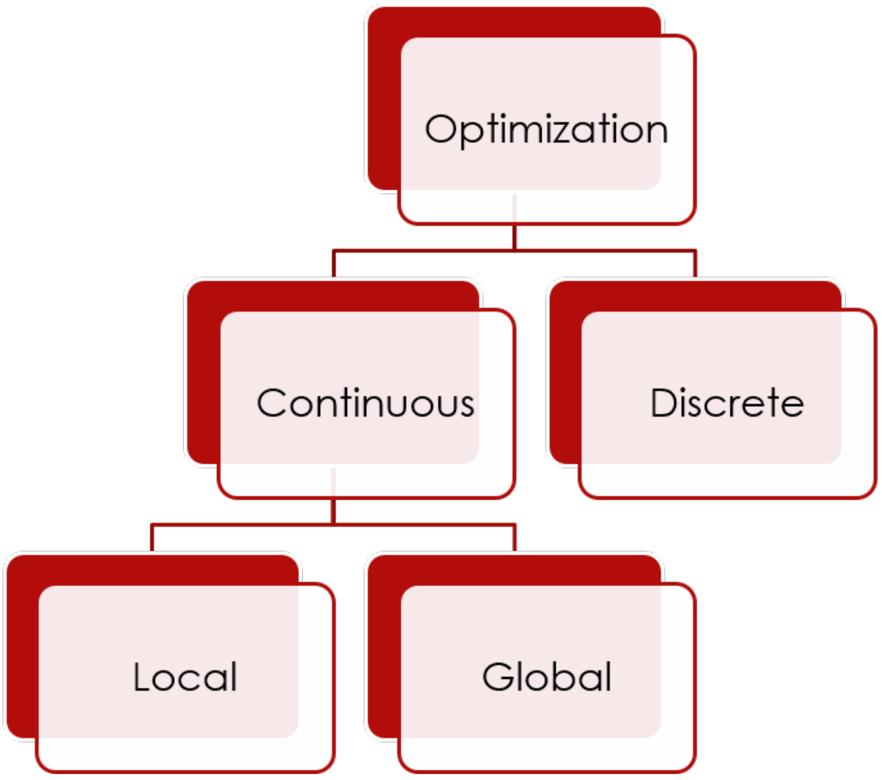


Figure 1.1: Optimization Tree

Local methods, such as naive Gradient Descent and Continuous Hill Climbing, move locally over the optimization space and aim to precisely locate the local optimal solution. As such, they tend to be quite fast and have been widely used in numerous applications (e.g., [19, 5, 40]). However, despite the aforementioned advantages such methods converge to local optima when operating on non-convex configuration spaces. On the other hand, global methods aim to find the global solution, typically by moving in a less restricted manner over the optimization space. Such approaches range from modified local search, such as gradient descent using momentum[33] and Adaptive subgradient methods such as Adagrad [12], to population-based meta-heuristics. In the absence of analytical solutions, such methods approximate the global optimal—instead of precisely locating a local one as local methods do. Given that the objective space is usually non convex and that one is interested in an approximation of the optimal solution, global optimization methods are of great interest and are widely used in practice [2]. Global optimization methods, as discussed, specialize in approximating the global optimum in multimodal functions, i.e. in functions where local optima exist. Such techniques can be classified, as [42] noted, based on their approach on the search that they utilize:

- Blind Search. When the future states are chosen completely random. In such a case, they are distinguished from a simple random search by their strategy in accepting these states.
- Local search. In local search, each future state belongs into a predefined neighbor-

hood of the current state. Such algorithms are simulated annealing and threshold accepting that will analytically be explained below.

- Non-local search. In this search approach, belong techniques that are not constrained within a neighborhood, and they intend to jump from a local optimum into the global or to a better choice of the existing. Particle swarm optimization and our novel buggy pinball algorithm fall within this category.

1.2 Meta-heuristics

Meta-heuristics have long been used for global continuous optimization (e.g.,[45, 10, 39]). They can avoid convergence to local optima and can scale to multidimensional problems. In addition, many meta-heuristic approaches do not require any derivative of any order for the objective function in contrast to what is a restrictive requirement of many continuous optimization approaches (such as gradient descent based optimization). However, many meta-heuristics—such as simulated annealing (SA) [24] and threshold accepting (TA) [13]—have been developed for combinatorial optimization problems, and their deployment in continuous optimization problems requires considerable tuning, which is not always straightforward; while their performance in terms of precision can be limited. Furthermore, even those best tailored for continuous optimization (Particle Swarm Optimization (PSO) [23]) typically accept worse solutions in order to escape local optima and ensure exploration, rather than consistently improve the state of the problem. Due to this reason meta-heuristics can be slow and inefficient when employed to optimize highly complex continuous configuration spaces.

1.3 Our Approach

Against this background, we propose a new single-point meta-heuristic algorithm, designed specifically for global continuous optimization. Our algorithm is inspired by the movement of a ball's collision and descent in the well-known pinball arcade game. Importantly, ours is an anytime algorithm that always improves over the solution, and which can return a valid solution even if it is arbitrarily interrupted. We evaluate our approach against SA, TA, and PSO on a number of commonly employed optimization testbed functions and on a number of dimensions. We show that our approach is able to find the global optima with high accuracy and precision and in shorter time than the benchmark approaches, especially when more complex functions are considered. We believe that the superiority of our approach, namely *buggy pinball* (BP) and the fact that a solution is guaranteed to improve over time makes it a better choice for continuous optimization tasks of high complexity and dimensions. We sum up our main contributions are as follows:

- We propose a novel single point meta-heuristic tailored for continuous optimization

- We experimentally show that ensuring exploration is possible even without having to accept worse solutions—which is a common assumption of meta-heuristic search
- We evaluate our approach against SA, TA, and PSO on a number of testbed functions and on a number of dimensions to demonstrate its efficiency and superior performance in terms of both accuracy and precision, especially in the more complex configuration spaces.

1.4 Thesis Outline

The rest of the paper is structured as follows. We first discuss background material and related work. Then we discuss our approach in detail along with core motivational aspects (that also justify the term ”buggy” in the name of the algorithm). Subsequently, we conduct a systematic evaluation of our approach; discuss the evaluation results; and finally we conclude and present directions for future work.[*]

[*]This work has been submitted for publication to AAAI 2022

2 Background and Related Work

Meta-heuristics can be classified into two broad categories. Single-point and population-based. Single-point approaches focus on modifying and improving a single point while population-based maintain a collection of points and improve them based on population characteristics. *Simulated annealing* is perhaps the most famous single-point meta-heuristic. Numerous simulated annealing variants have been proposed in the literature [38]. A well-known one is *threshold accepting*. A milestone in population-based approaches, is *particle swarm optimization*. Below we discuss these algorithms in detail.

2.1 Simulated Annealing

Simulated annealing (SA) [24] is a global search meta-heuristic algorithm capable of solving discrete and continuous optimization problems, and is widely used in AI [34]. It is especially useful for escaping from local optima. It is inspired by metallurgy, where the process of annealing is widely used, to achieve the altering of the physical properties of a solid. This process relies on the cooling of the preheated solid, which if it is slow enough, the desired structural integrity can be achieved. SA imitates this behavior, avoiding local optimal solutions when searching an objective function's configuration space, by accepting worse solutions with a probability. This probability is calculated using a predetermined ”temperature” value, which is being reduced over time via a ”cooling” schedule, and the energy difference of the last two states (i.e. the current and the candidate solution state of the problem). This probability is expressed by $e^{-\Delta E/T}$: as the temperature decreases over time, the probability of accepting worse solutions will also decrease. SA has shown great success in continuous optimization tasks and as such is considered a benchmark in our work. For instance, [3] used SA to optimize the ma-

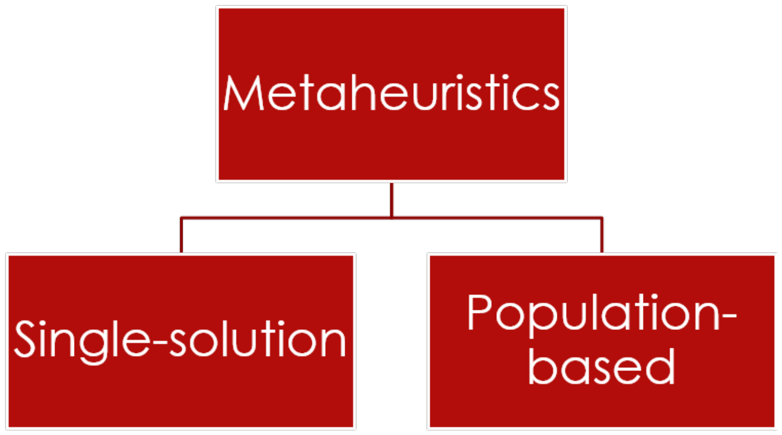


Figure 2.1: Meta-heuristic approaches Tree

chining parameters for continuous profile machining, while [26] used SA to optimize the conditions of the continuous casting process, based on an undesirability function.

2.1.1 Description of Simulated Annealing

The idea of applying this annealing technique into optimization problems drove into the creation of the Simulated Annealing algorithm. The key parameter of this implementation is the so-called temperature value. This parameter is supposed to have the same impact as the temperature of the actual process. A sufficient number of different states must be simulated in order to approach the global optimal value, something that can be achieved by decreasing the temperature slowly.[11]
For each temperature value, there is an energy E that exists with a probability given by the Boltzmann factor $e^{-\Delta E/T}$, where k represents the Boltzmann constant. All the possible energy states constitute the Boltzmann distribution.[11]
In 1953, Metropolis et al.[28] created an algorithm that was simulating the collection of atoms achieving a thermodynamic balance at a given temperature. The procedure occurs by choosing one atom at a time, changing its state, and then calculating the energy difference ($\Delta E = E_{current} - E_{new}$) between the previous and the current state. In case that ΔE is positive or equal to 0, the new state is accepted, and it is replacing the current. When ΔE is negative, the new state is accepted with a probability $P(\Delta E) = e^{-\Delta E/T}$. A random number within the range $(0, 1)$ is chosen every time to be compared with $P(\Delta E)$ and to determine whether the new state will be accepted. When the random number is smaller than this probability, the state is accepted, otherwise the procedure continues with the same state. Applying this process repeatedly, sums up the procedure of moving atoms when heated at a given temperature. The list of all states used in the algorithm creates the Boltzmann distribution.

The algorithm simulates this process of annealing leading to the temperature that brings thermodynamic balance to the solid by using the aforementioned Metropolis algorithm. On every step of the process, a modification of the current state takes place and it is being evaluated. This evaluation occurs by calculating ΔE between the current and the possible future state of the system. When the value of ΔE is negative (i.e. the energy of the future state is smaller than the current) the new state is established; otherwise, it is accepted with a probability $e^{-\Delta E/T}$. This procedure occurs by choosing a random value between 0 and 1, and the new state is accepted when that value is lower than the probability mentioned above. This implementation generates a discrete-time Markov chain constituted by the collection of the algorithm's states, where each state is depended only to its predecessor. What can be concluded is that if the Markov chain is sufficiently long, the system approaches the thermodynamic equilibrium on the final temperature value. [11]

Thus, the temperature parameter is the key to the whole process. At first, when its value is relatively high, the Boltzmann factor is close to 1, and consequently almost every future state will be accepted. On the other hand, as the temperature gets lower, the probability of accepting a worse state approaches 0. What it is achieved is that on an early stage, the configuration space is sufficiently explored, and during the whole process, getting stuck into local optima can be avoided. When the algorithm is about to be terminated, worse states are very unlikely to be accepted, therefore an optimal value has been approximated, when enough iterations have taken place. Each temperature value might also have a number of iterations taking place before decreasing, in order to further expand the algorithm or explore for more time at a given temperature.[11]

The flowchart in Figure 2.2, given by [52], summarizes the process of Simulated Annealing.

As described by [4], the elements of simulated annealing are:

- Finite space S
- Cost function J with limits on S where the global optima of J are contained.
- For every element $i \in S$, there is a subset $S(i) \subset S$ which is called the set of neighbors of i.
- Temperature function $T(t) : t \in (0, \infty)$ described as the cooling schedule. Variable t represents time as the algorithm progresses and $T(t)$ is the temperature for every value of t.
- The initial state $x(0) \in S$.

The Simulated Annealing algorithm creates a discrete-time Markov chain $x(t)$, using the aforementioned elements. On each current state $x(t)$, a neighbor j is chosen randomly. Neighbor j replaces $x(t)$, if $J(j)$ is better than $J(x(t))$ or replaces it with a probability if it is worse.

The simulated annealing algorithms is shown in pseudo-code in Algorithm 1.

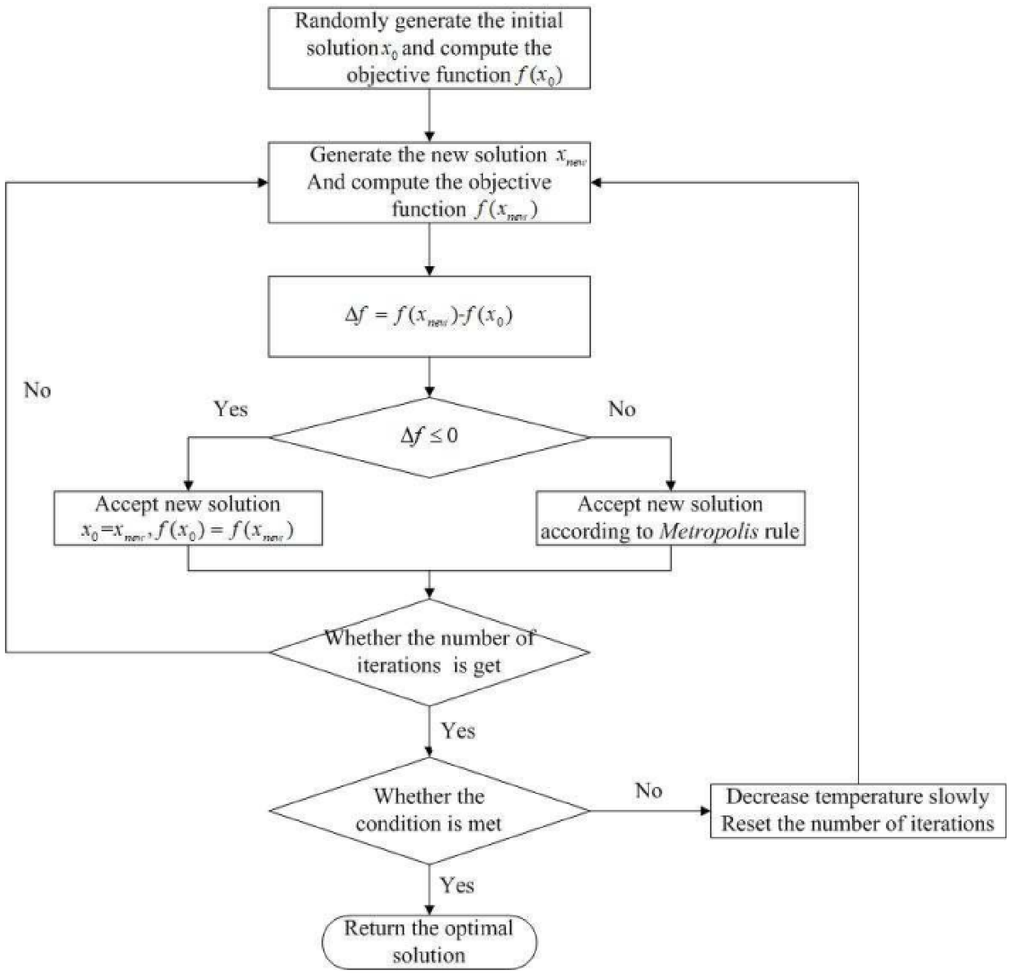

Randomly generate the initial solution x_0 and compute the objective function $f(x_0)$
Generate the new solution x_{new} And compute the objective function $f(x_{new})$
$\Delta f = f(x_{new}) - f(x_0)$
No
Yes
No
$\Delta f \leq 0$
Accept new solution $x_0 = x_{new}, f(x_0) = f(x_{new})$
Accept new solution according to *Metropolis* rule
Whether the number of iterations is get
Yes
Whether the condition is met
No
Decrease temperature slowly Reset the number of iterations
Yes
Return the optimal solution

Figure 2.2: Simulated annealing flowchart

Algorithm 1 Simulated Annealing Algorithm

```
1: $s = s_0$
2: $T = T_{max}$
3: initialize neighbour distance $n_d$
4: for $k = 0$ until terminal condition is met do
5:     $T \leftarrow cooling(T, k)$
6:     $s_{new} \leftarrow neighbour(s)$
7:     if $f(s) \geq f(s_{new})$ then
8:         $s \leftarrow s_{new}$
9:     else if $random(0, 1) < e^{\frac{f(s) - f(s_{new})}{T}}$ then
10:        $s \leftarrow s_{new}$
11: return $s$
```

2.1.2 Cooling Schedule

The cooling schedule that will be used to Simulated Annealing is a crucial factor for the effectiveness of the algorithm. The acceptance probability equation $e^{-\Delta E/T}$ shows the importance of the temperature parameter. When the cooling (i.e. the decrease of the temperature value) occurs in a fast pace, the algorithm will probably get stuck to a local optimal value, whereas on a slow cooling pace, it becomes ineffective. The goal when searching for the best cooling schedule is to find the fastest possible version while the convergence to one of the global optima of the given space is certain. Since the cooling scheme is bound to have a finite time schedule, the convergence of the algorithm cannot be considered certain.[38]
The SA's cooling schedule is consisted of the initial temperature, a function that determines the decreasing rate of the algorithm, the amount of iterations on each temperature value, and the conditions that terminate the procedure, which may be a final temperature value or a number of total iterations or a combination of those.[38]
Over the years, a lot of research has been conducted, regarding the optimal cooling schedule. Some of the most notable publications are mentioned below.
The temperature parameter could terminate the process when reaching zero, therefore this is not compulsory. In [27], was proposed a value for the final temperature calculated by:

$$T_f \leq \frac{\varepsilon}{ln\frac{|y|-1}{P}} \tag{2.1}$$

where T_f is the final temperature, ε is a very small number ($\leq 10^{-3}$), y is the objective function, and P the probability of acceptance of worse solutions.
A cooling schedule can be either static or adaptive. A static schedule is determined at the beginning of the procedure, while an adaptive schedule changes the rate that it decreases the temperature based on the progress of the algorithm. An effective cooling schedule manages to combine a moderate execution time with SA's dependence on asymptotic behavior.[30]
Various cooling schedules have been implemented throughout the years, and sometimes their efficiency varies on the problem they are applied. One of the first cooling schedules is proposed by [24], which was a linear decrease of the temperature value described as:

$$T_i = T_0 - ni \tag{2.2}$$

where T_0 represents the initial temperature of the process, i is the iteration number and n is a factor that determines how fast the temperature will decrease. They also suggested a similar calculation of the temperature by:

$$T_{i+1} = \alpha \mathrm{T}_i - 1 \tag{2.3}$$

Where α is given on the range (0, 1), and determined empirically. This value is later proposed in [32] to be calculated using the initial temperature T_0, the final temperature T_f and the total number of iterations F:

$$a = (\frac{T_f}{T_0})^{\frac{1}{F-1}} \tag{2.4}$$

Another linear cooling scheme is proposed in [15], which depended mainly on the initial temperature:

$$T_i = \frac{T_0 - i}{cT_0} \tag{2.5}$$

With c being a constant. Also, [21] proposed a logarithmic equation:

$$T_i = \frac{c}{1 + log(i)} \tag{2.6}$$

In [44], a formula is created that made the logarithmic decrease faster by adding to the equation the standard deviation of the cost values of the states generated (σ):

$$T_{i+1} = \frac{T_i}{1 + T_i \frac{ln(1+c)}{3\sigma_i}} \tag{2.7}$$

In [27], a nonlinear schedule is created, aiming to a much slower one:

$$T_{i+1} = \frac{T_i}{1 + \beta \mathrm{T}_i} \tag{2.8}$$

Setting β>0. This was followed by [8], where it is proposed for β to be set:

$$\beta = \frac{\mathrm{T}_0 - \mathrm{T}_f}{f T_0 T_F} \tag{2.9}$$

In [35], a cooling schedule is also created that follows slow cooling at the beginning of the algorithm and continues with a faster pace:

$$T_i = \begin{cases} \frac{T_0}{1+i}, & i \leq I_{lim} \\ 0.95 T_{i-1}, & i > I_{lim} \end{cases} \tag{2.10}$$

The I_{lim} parameter is set such that the algorithm will not take too long to terminate, and it is determined empirically.

2.1.3 Neighbourhood

Neighborhood $N(x)$ of any point x in a cost function $J(x)$ are considered all the possible states that can succeed x. Determining the neighborhood in Simulated Annealing is a very important aspect to take into consideration, as it directly affects the efficiency of the algorithm.
There is a lot of research that has been dedicated into finding the optimal neighborhood structure. In [16], it is noted that in small neighborhood structures, relatively to the cost function, prevent the Markov chain to move around fast enough, and it might not find the global optima, while in a large neighborhood transforms the algorithm into random sampling. In [7], it is reported that a small neighborhood performs better in some discrete problems like the Traveling Salesman Problem (TSP) and the Quadratic

Algorithm 2 Neighbour Function

```
1: function neighbour($s$)
2:     for each variable $x_i$ of $s$ do
3:         $neighbours[i] = random(x_i - n_d, x_i + n_d)$
       return $f(neighbours)$
4: end function
```

Assignment Problem (QAP)[25]. In contrast, [50] shows that larger neighborhoods are more effective into finding global optima. Generally, we can conclude that the efficiency of the neighborhood is depended on the formulation of the problem.
In [46], an algorithm is introduced, called Opposition-based Simulated Annealing (OSA). This algorithm follows the same procedure as SA, but for every neighbor generation, it also generates an opposite neighbor. Given a configuration space S, two possible states $s_1, s_2 \in S$ are considered opposite, when they have the same distance from a given reference point r, and they are in opposite sides. On this implementation the opposite pairs of values are considered unique. The procedure continues by comparing the two neighbors and keeping the best one to compare it with the current state. By comparing OSA to classic SA and Random Simulated Annealing (RSA) on some continuous optimization problems, is proved that OSA performs better. RSA is using two random neighbors instead of two opposite neighbors like OSA.
Another neighborhood structure is given by [1], using the concept of the variable neighborhood search (VNS) algorithm. This method defines a set of neighborhood structures $N_k, k = 1, 2, ...$, by separating the configuration space and for each neighborhood N_k starting from an initial random point, performs the SA procedure to locate a local optimum. Then, this local optimum is compared to the previous neighborhood's optimal value, and the best of them is chosen. This procedure occurs on every neighborhood defined. The performance of this hybrid VNS-SA algorithm compared to classic SA is tested on the Weibull distribution, and it shows that on the same accuracy levels, this hybrid algorithm requires less CPU time.

2.2 Threshold Accepting

A similar meta-heuristic proposed as a simpler version of SA is threshold accepting (TA)[13]. TA replaces SA's probability of acceptance with a threshold value. This value is set initially at the beginning of the algorithm. Then, the proposed solution is deducted from that of the current, and the new solution is accepted only when the difference is lower than the threshold. Similarly, to the temperature in simulated annealing, the threshold follows a cooling schedule, and it is being reduced over time, so only better solutions will be more likely to be accepted over time. They also claim that, since it is computationally simpler, similar results to SA are achieved in less computational time. A flowchart of threshold accepting procedure is shown in Figure 2.3
Threshold accepting was originally implemented and tested in TSP, so just like simulated annealing, it has been tailored for discrete optimization tasks. TA has also been

Algorithm 3 Threshold Accepting Algorithm

```
1: $s = s_0$
2: $T = T_{max}$
3: initialize neighbour distance $n_d$
4: for $k = 0$ until terminal condition is met do
5:     $T \leftarrow cooling(T, k)$
6:     $s_{new} \leftarrow neighbour(s)$
7:     if $f(s) - f(s_{new}) > T$ then
8:         $s \leftarrow s_{new}$
9: return $s$
```

modified for continuous optimization tasks, i.e. [10], where TA is extended for multiple objective continuous optimization problems. As such it is also considered in our approach. The pseudo-code of threshold accepting is depicted in Algorithm 3.

2.3 Particle Swarm Optimization

Perhaps particle swarm optimization [23] is the most famous population-based meta-heuristic approach used typically for continuous optimization. Particle swarm optimization (PSO) is an evolutionary algorithm that is inspired by the social behaviour of animals. The intention is to imitate the movement of birds and fish in a bird flock and a fish school respectively.
PSO brings this into computer science by initializing various points, i.e. the particles, located all over the configuration space of the examining objective function. These particles move with respect to a d-dimensional velocity parameter. The velocity of each particle is determined in each round and for each variable of the objective function by three factors:

- the velocity of the previous round
- the best position that has been found by this particle
- the best position found by all particles

Each of these factors is controlled by a hyper-parameter, that needs to be determined beforehand and tuned for every problem that is used. The number of the particles is also a parameter that needs to be determined at the beginning of the algorithm. PSO has very high performance among continuous optimization algorithms, and has been used in various fields. For instance, PSO was used by [14] to analyze human tremor by training a neural network, which was able to detect it. Also, [18] used PSO to predict the optimal value of the energy consumption for a smart home in a given weather. In consideration of the above PSO is also considered in this work. The pseudo-code of Particle Swarm Optimization is shown in Algorithm 4. The procedure of PSO can be observed through a flowchart in Figure 2.4.

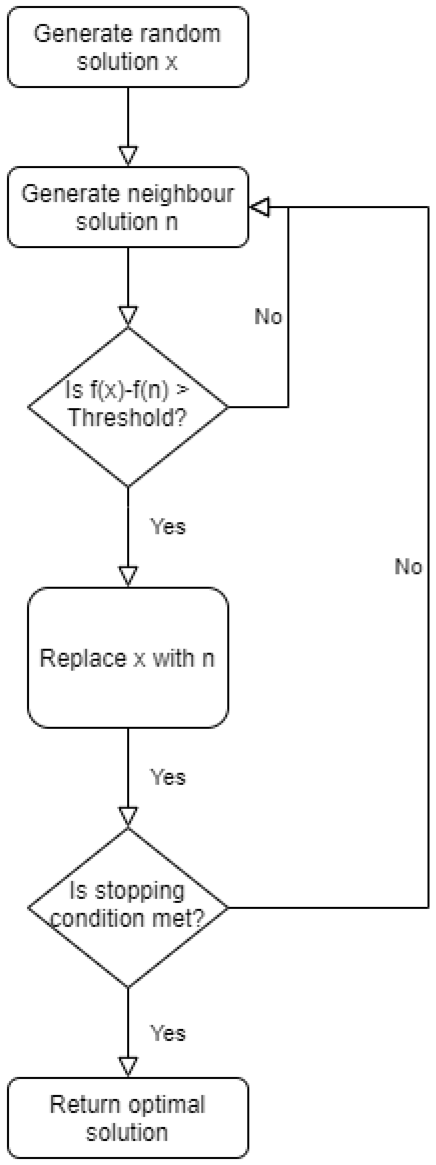

Generate random solution x
Generate neighbour solution n
No
Is f(x)-f(n) > Threshold?
Yes
No
Replace x with n
Yes
Is stopping condition met?
Yes
Return optimal solution

Figure 2.3: Threshold Accepting Flowchart

Algorithm 4 Particle Swarm Optimization Algorithm

```
 1: initialize particles population, weights
 2: for each particle do
 3:     initialize starting point $x_i$, personal best position $pb$, velocity $\mathbf{v}$
 4:     if $f(x_i) < f(\text{global best } gb)$ then
 5:         $gb \leftarrow x_i$
 6: for $i$ in rounds do
 7:     for each particle do
 8:         for each variable $d$ do
 9:             $v_d \leftarrow \omega v_d + c_1 random(0,1)(pb_d - x_{i,d}) + c_2 random(0,1)(gb_d - x_{i,d})$
10:         $x_i \leftarrow x_i + \mathbf{v}$
11:         if $f(x_i) < f(pb)$ then
12:             $pb \leftarrow x_i$
13:             if $f(x_i) < f(gb)$ then
14:                 $gb \leftarrow x_i$
    return $gb$
```

2.3.1 Particles

Particle Swarm Optimization heavily relies on its population-based characteristics. Each particle is usually incapable of solving an objective function on its own. The particles are in need of communicating, in order to solve a problem, thus it is often considered that they are creating a social network. Particles are exchanging information, which significantly affects their updating velocity of each round. The population size of the particles is perhaps the most important parameter that is determined beforehand. It is supposed to be set empirically, considering the number of dimensions and the difficulty of the problem that will be optimized. [31]

2.3.2 Inertia Weight

Inertia weight is the parameter that defines the percentage of the previous velocity value that will be added to the current value. This weight did not exist in the original work and it was later added by [37]. It was added with the purpose to expand the potential search space that particles cover. The optimal value is considered equal or close to 0.9 in most cases, since high values appear to perform adequately. However, values larger than 1 appear to destabilize the algorithm. Various schedules have been proposed over time, in order to regulate the inertia weight, and reach its maximum capabilities.
We use ω as a constant, typically set to large values ≈ 0.9.

2.3.3 Velocity Parameters

The parameters c_1 and c_2 (shown in line 9 of Algorithm 4) determine the weight, along with a random value between 0 and 1, that will be applied into the personal and global best respectively. These weights affect how much of the applied velocity of the particle

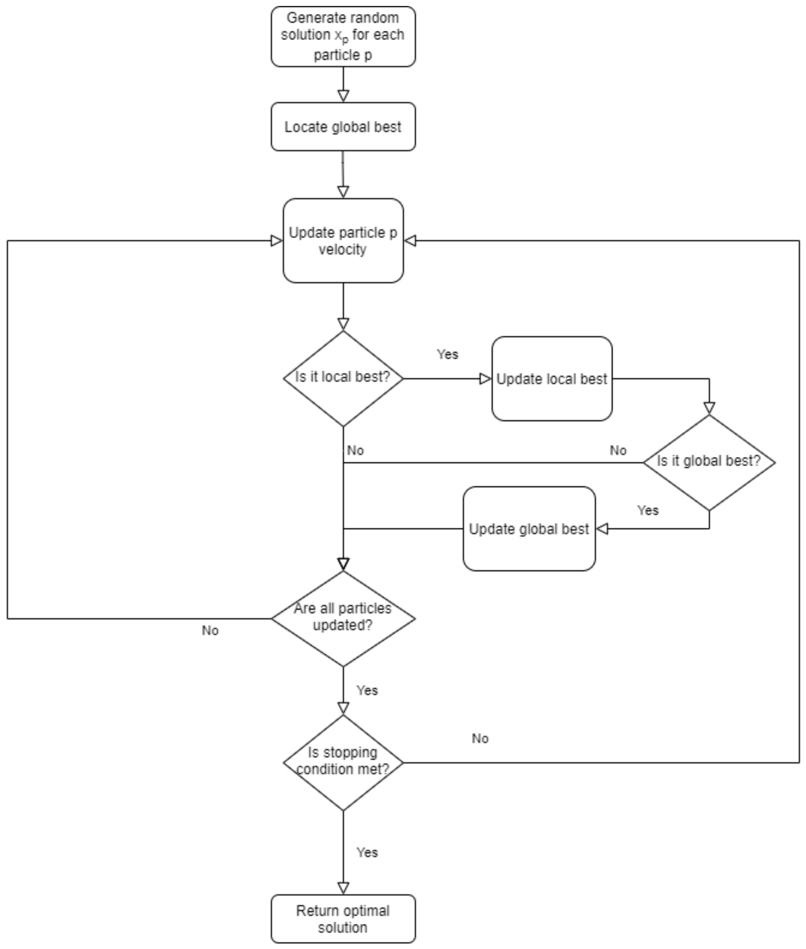


Figure 2.4: Particle Swarm Optimization Flowchart

Figure 2.5: Digital version of pinball

will be stirred towards one of the two aforementioned positions. The velocity parameters play a crucial role in the behaviour of the algorithm. If we explain the movement of the particles using Newton's second law, the half values of the parameters would be the mean stiffness of some springs that are pulling the particles. The original work suggested that c_1 and c_2 should be controlled in such a way, that would keep the velocity within some range. These caused a few problems regarding the efficiency of the algorithm. The optimal limits were problem specific, with no rule of thumb in place. Also, initially, the values $c_1 = c_2 = 2$ were considered as optimal in that case, but it was later abandoned.[31]
The range of the velocity is not used in our work and parameters c_1 and c_2 were initialized empirically in each function.

2.4 The pinball game

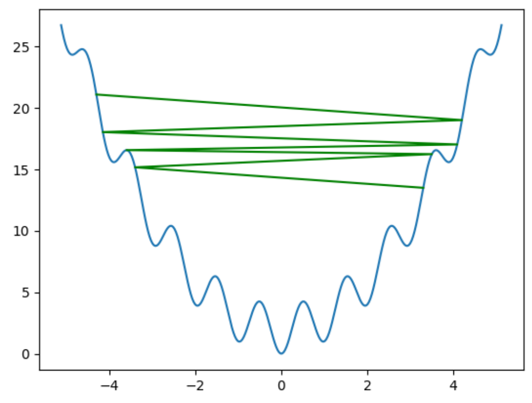
Pinball is a well-known arcade game, which was also later adapted in a digital form. The game's origins are very old. It is believed that it originates from Europe.
The game's goal is to maintain the ball within the play field. The ball is thrown from the top of the field and as it bounces around, the player collects points. The player is using some paddles, known as flippers, which are located at the lowest point of the field, they can manipulate the course of the ball as it sends it upwards, in order to collect more points. The points are collected when the ball collides on various obstacles. The formation of the obstacles and the way that the points are collected vary, depending on the edition of pinball. The game is terminated when the player fails to send the ball upwards, and it lands underneath the flippers. An image of perhaps the most famous digital pinball version is depicted in Figure 2.5.

3 The Buggy Pinball (BP) Algorithm

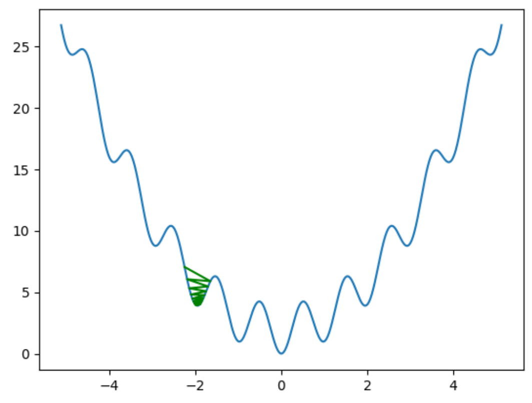
Our work is motivated by the movement of the ball in the pinball game. In this game, a ball is thrown to the highest point, and by moving inside a glass-covered cabinet, heads towards the lowest point. The player uses paddles to avoid the ball from falling in the lowest point and collects points by hitting various targets. The main challenge of the game originates by the multiple collisions of the ball, which will eventually lead the ball to the lowest point. We imitated this movement by creating a trajectory-based search method, where the "ball" is moving until a collision with the objective function takes place, which occurs at the common point between the objective function and the trajectory segment—see e.g. Figure 3.2. The trajectory segments start almost horizontally and become steeper with time. A trajectory segment corresponds to one round of our search algorithm. The intuition behind this movement is that when the ball is moving almost horizontally, the probability of getting into a local optimum is very small, as shown in Figure 3.1a, while steeper trajectories help to speed up convergence to an optimum as time progresses. Anytime algorithms are algorithms that increase the quality of the output as time progresses [17]. In our algorithm, the trajectory segments that are progressively created, direct the search towards values that can only better optimize the cost axis. So, every new point that is detected, is guaranteed that is better than the current. Thus, Buggy Pinball is anytime, in the sense that every new segment can only provide a better solution.

3.1 Why Buggy?

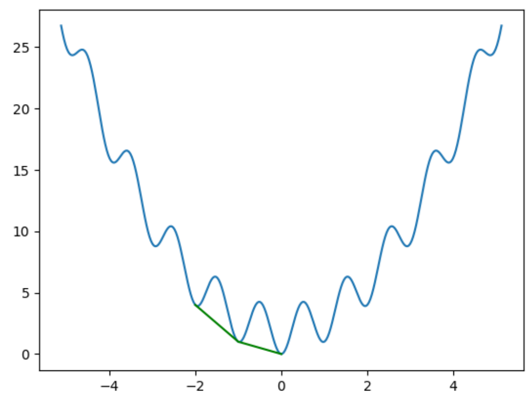
Even though there is a small chance of getting into a local optimum solution, it is not equal to zero. To significantly decrease the possibility of converging to a local opti-



(a) Intuition of Trajectory Movement



(b) Example of entering a local minimum



(c) Avoiding a local minimum

Figure 3.1: Description of trajectory movement

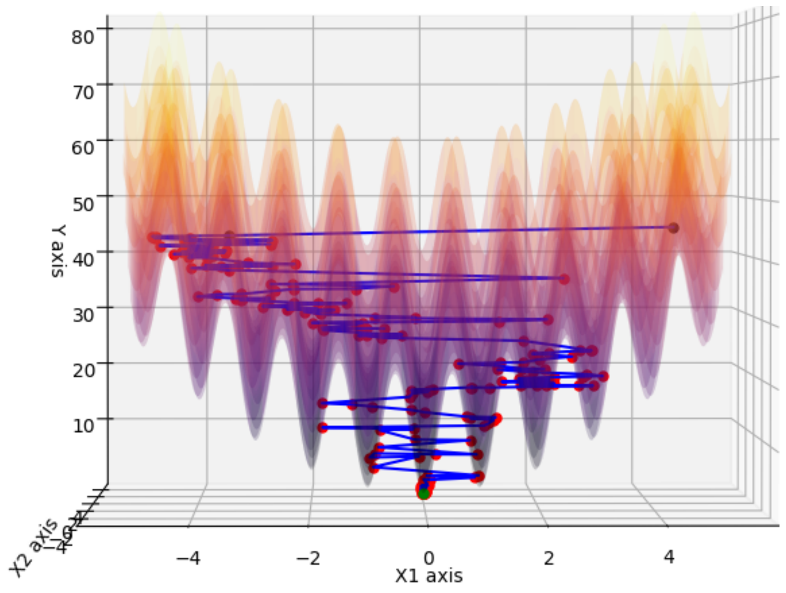
mum, we identified the need for the pinball game to be "buggy". Instead of bouncing away, as in the real pinball, the "ball" in our algorithm is capable of also continuing the search underneath the configuration space. This way, not only local optima are effectively escaped, but also there is no need of accepting worse solutions in order to ensure exploration, unlike most meta-heuristic approaches. This makes BP an any-time algorithm. An illustration of being trapped into a local minimum is shown in Figure 3.1b. In Figure 3.1c, we can also see how Buggy Pinball creates trajectories underneath the configuration space, and avoids convergence to a local minimum. This behavior allows BP to significantly reduce the probability of getting stuck into local optima. The BP effectiveness in avoiding local optima is empirically verified by its increased accuracy, demonstrated in our results.

3.2 Overview of BP

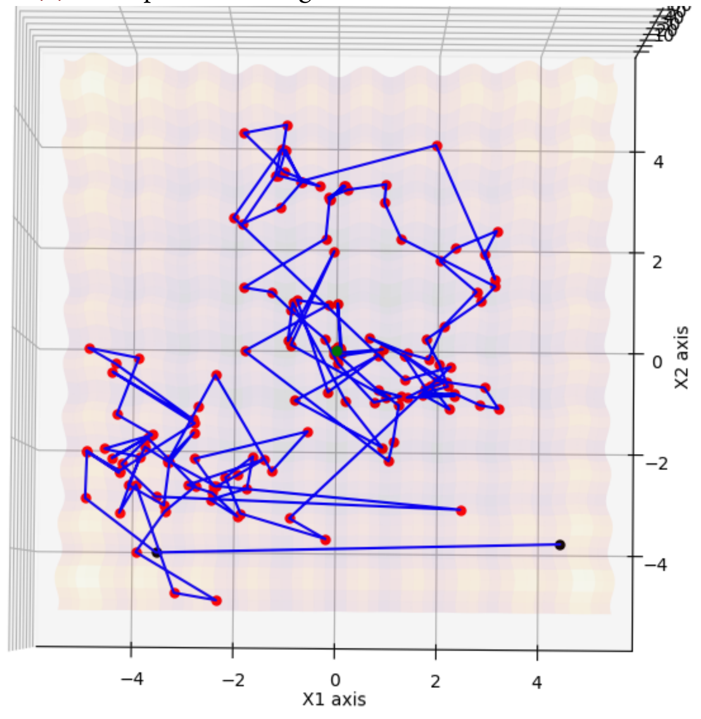
As discussed, Buggy Pinball searches the space by creating trajectories. These are progressing via steps in a continuous simulation manner until a collision between the function and the ball occurs. Once we find that a collision occurs between two steps, a routine to precisely locate the point of collision begins. When the exact point is identified, we create a new trajectory segment starting from that point, and repeat the procedure. The trajectories are created with a random direction, in order to ensure that all of the configuration space is searched. The elevation angle though (i.e., the angle of descend or ascend of the segment depending on whether we minimize or maximize a function) follows a predefined schedule: it starts at an almost horizontal level in order to avoid local optima, and becomes more steep over time to achieve faster convergence rates. By contrast, the step size is reduced over time, to achieve higher precision. As we get to bigger configuration spaces, we choose higher starting step values and smaller elevation angles. As mentioned, the trajectories are even able to advance the ball underneath the configuration space of the function. An example of a BP search on the 3D version of the Rastrigin function (Fig. 4.1h) can be seen in Fig. 3.2. Each blue line represents one trajectory segment, while each red point is the common point between the function and the segment.

3.3 The BP in Detail

BP (Algorithm 5), is composed by five main parts: (1) initialization, (2) trajectory segment creation, (3) stepping forward, (4) recursive refining, and (5) cooling schedules. These parts are detailed below. We note that each round corresponds to one trajectory segment, which progresses in steps. Also, an objective function is characterized by its cost, i.e., its y values, and its x_i variables, which are the parameters to be optimized. Every x_i is an axis in the configuration space (see Fig. 3.2 for an example with two x_i variables).



(a) Example on Rastrigin Function from the side



(b) Example on Rastrigin Function from above

Figure 3.2: Example on a 2D implementation of Buggy Pinball on Rastrigin Function
*The blue line indicates the search direction of the ball in a single round, while red dots show a collision with the function

3.3.1 Initialization

The first part (i.e., Algorithm 5, lines 1-3) is that of initializing our hyper-parameters. First we set the original step and elevation angle, a. The original step size should be set in such a way that we do not make too large steps in the configuration space at the beginning of the process. We have empirically found that a choice of an original step size at ~ 10% of the average variable range performs well (in general the step size should be bigger as the configuration space becomes bigger). We note here that the sign of the step (variable *stepSize* in Alg. 5) should be negative for minimization tasks and positive for maximization.

The elevation angle is defined with respect to the plane perpendicular to the y axis. For the elevation angle we always begin with a value close to 0 (e.g., 0.1) in order to perform a near-horizontal movement.[*] We have also identified that its value should be smaller in large spaces (in contrast to that for step) since a more horizontal movement is required in order to effectively avoid local optima. The same "smaller values" rule applies as the number of dimensions increases.

In line 2, we also set the number of rounds, which corresponds to the number of trajectory segments to be created and followed, as well as the number of steps to be executed in each segment. The number of rounds should be set as high as possible considering the optimization time constraints. With respect to the number of steps, we have found that a reasonable choice is one such as the product of the number of steps and the step size is three to five times bigger than the average variable range. Finally, the initialization of a random starting point is taking place in line 3, and the procedure of the algorithm begins from line 4. As BP is able to escape local optima the algorithm is not very sensitive on the initial points selected with respect to convergence to a global solution. Nevertheless a good initial solution can considerably speed up the algorithm, as further discussed in Results.

3.3.2 Trajectory Creation

Each round corresponds to the creation and execution of a trajectory segment starting from the current point and ending when the maximum number of steps is reached or a collision is detected. At the beginning of each round, the segment's direction is set randomly,[†] in order to ensure the best exploration of the configuration space with only the elevation angle being fixed and following a predetermined schedule. Thus, the step component for each variable is set randomly to a value in [-1, 1]. The corresponding step-towards-optimum component for the y axis that respects the elevation angle is then

[*]The sign of the elevation angle is irrelevant as the square of its sin value is considered.

[†]We experimented with trajectories alternating from one direction to another; however we empirically found this took a toll in exploration. Thus, the algorithm does not behave like a pinball, however the final trajectories do resemble a pinball movement.

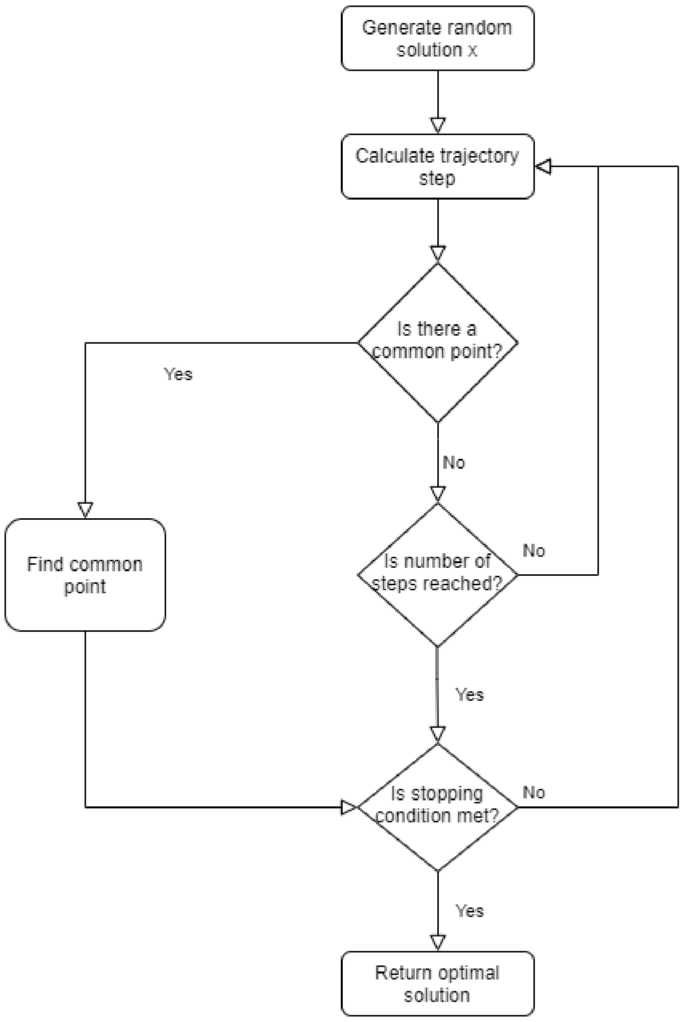

Generate random solution x
Calculate trajectory step
Is there a common point?
Yes
No
Find common point
Is number of steps reached?
No
Yes
Is stopping condition met?
No
Yes
Return optimal solution

Figure 3.3: Buggy pinball flowchart

computed:

$$y_{step} = \sqrt{\frac{\sin^2 a \sum_{j=1}^{d} - x_{step,j}^2}{\sin^2 a - 1}} \tag{3.1}$$

where d is the number of variables—i.e., the problem's dimensions. It is easy to derive this equation with the following procedure. We know for the elevation angle that: $\sin a = \frac{|\mathbf{nu}|}{|\mathbf{n}||\mathbf{u}|}$, where n is the vector perpendicular to the fundamental plane, i.e. parallel to the y axis $(0, 0, ...0, 1)$. Vector u is the trajectory's segment vector $(x_0, x_1, ..., x_{d-1}, y)$. Values $x_0, x_1, ..., x_{d-1}$ and the elevation angle are known. Solving for y gives us Equation 3.1.
Equation 3.1 solved for y.

$$y = \pm \frac{\sqrt{-(x_0^2 + x_1^2 ... + x_n^2) \sin^2 a}}{\sin^2 a - 1} \tag{3.2}$$

With the above procedure we have set the direction of the segment. What remains is to readjust the dimension-wise step components to respect the predetermined overall step size. In order to achieve this, we multiply all step components with a common factor, z, calculated as:

$$z = \frac{stepSize}{\sqrt{x_0^2 + x_1^2 ... + x_{d-1}^2 + y_{step}^2}} \tag{3.3}$$

(used in lines 7, 8 of Alg. 5). Once we have completed this procedure, our step for the current round is ready. We then apply it for the number of steps stated or until a crossing of the objective function is detected.[‡]

3.3.3 Stepping Forward

In this part (i.e. Algorithm 5, line 10 and Algorithm 6), we start our trajectory segment search by applying the step's values on each axis, and we continue until one of the two conditions of stopping is met. As the segment proceeds in the configuration space, it moves downwards with respect to the y axis, as the segment value of y decreases in every step. This results into accepting only better values, i.e. closer to the global optimum, of the current position. The crossing of the objective function by the segment is determined by checking the last two steps taken. As shown in the crossing detected function (Algorithm 6), if the difference between the y value and the function evaluation is of different sign between these steps, we know that a point on the objective function is "internal" to the last trajectory segment drawn—and "recursive refining" is triggered.

[‡]Crossing of the objective function means that a common point of the objective function and the current trajectory segment has been detected, which is closer to the global optimum.

Algorithm 5 BP algorithm

```
1: $stepSize, a = step_{max}, a_{min}$
2: set $\#rounds$, $\#steps$
3: $\boldsymbol{x}, y \leftarrow$ initialize randomly
4: while $i$ in $\#rounds$ do :
5:     $\boldsymbol{x}_{step}$ = random(-1,1)
6:     $y_{step} = \sqrt{\frac{\sin^2(a)\sum_{j=1}^{d} -x^2_{step,j}}{\sin^2(a)-1}}$
7:     $\boldsymbol{x}_{step} = z * \boldsymbol{x}_{step}$
8:     $y_{step} = z * y_{step}$
9:     while $j$ in $\#steps$ do
10:        if crossing_detected($\boldsymbol{x}, y, \boldsymbol{x}_{step}, y_{step}, j$) then
11:            $\boldsymbol{x}, y$ = recursive_refining($\boldsymbol{x}, y, \boldsymbol{x}_{step}, y_{step}, j$)
12:            break
13:    $a$ =elevation_cooling($a_{min}, i, \#rounds$)
14:    $stepSize$ =stepSize_cooling($step_{max}, i, \#rounds$)
15: return $\boldsymbol{x}, y$
```

Algorithm 6 Crossing detected function

```
1: function crossingDetected($\boldsymbol{x}, y, \boldsymbol{x}_{step}, y_{step}, j$)
2:     $A = y + (j-1)y_{step} - f(\boldsymbol{x} + (j-1)\boldsymbol{x}_{step})$
3:     $B = y + jy_{step} - f(\boldsymbol{x} + j\boldsymbol{x}_{step})$
4:     return $A > 0 \wedge B < 0 \vee A < 0 \wedge B > 0$
5: end function
```

3.3.4 Recursive Refining

This is a process of iterative refinement, shown also in pseudo-code in Algorithm 7, used to locate the exact point where a trajectory segment crosses the objective function. It is activated only when a crossing of the configuration space is detected, otherwise that part is skipped. In that case, a loop begins, where the point in the middle between the last two steps is examined in order to determine whether this is the common point between the segment and the function (or a very good approximation). If it is not, then we choose whether we continue the loop on the upper or the lower half of the examined part, depending on where the crossing is identified, according to the signs of the points. We continue by examining the point in the middle of that part as before, and the same process is repeated until the exact location of the common point is found. Once we find this point, we stop the iteration of the current round, as we have reached the closest point to the global optimum so far.

Algorithm 7 Recursive refining function

```
1: function recursiveRefining($\boldsymbol{x}, y, \boldsymbol{x}_{step}, y_{step}, j$)
2:     $\boldsymbol{x} = \boldsymbol{x} + \boldsymbol{x}_{step} j$
3:     $y = y + y_{step} j$
4:     if $y - f(\boldsymbol{x}) \approx 0$ then
5:         return $\boldsymbol{x}, f(\boldsymbol{x})$
6:     else
7:         $\boldsymbol{x}_{step} = \frac{\boldsymbol{x}_{step}}{2}$
8:         $y_{step} = \frac{y_{step}}{2}$
9:         $\boldsymbol{x} = \boldsymbol{x} - \boldsymbol{x}_{step}$
10:        $y = y - y_{step}$
11:        if crossing_detected($\boldsymbol{x}, y, \boldsymbol{x}_{step}, y_{step}, 0$) then
12:            return recursive_refining($\boldsymbol{x}, y, \boldsymbol{x}_{step}, y_{step}, 0$)
13:        else
14:            return recursive_refining($\boldsymbol{x}, y, \boldsymbol{x}_{step}, y_{step}, 1$)
15: end function
```

3.3.5 Cooling Schedules

The last part takes place at the end of each round (i.e. Algorithm 5, line 13-14). It determines the values of the desired step size and the angle for the upcoming round. It simply applies the cooling schedule function determined for each of the parameters. In our experience we used a simple linear cooling schedule, where the final values are a fraction of 1 for the step size, so we have increased precision no matter the structure of the problem, and from 0.1 degree for highly complex many-local-optima functions up to 89 degrees for simple slope no-local-optima functions. This seems to work satisfactorily for each problem tested so far, but further research on the topic is desirable future work.

4 Experimental Evaluation

4.1 Experimental Setup

In order to evaluate our approach, BP, SA, TA and PSO have been tested into the minimization of ten (10) benchmark optimization functions. These functions are classified as unimodal or multimodal, depending on whether there are local optimal points on their configuration space or not. The two unimodal functions are the Sphere and Easom functions. The multimodal functions are: Rastrigin, Ackley, Eggholder, Schwefel, Shubert, Holdertable, Langermann and Dropwave functions. More multimodal functions were chosen because our focus is on global optimization, therefore, optimizing as many complex and diverse multimodal functions as possible was our primary goal. All functions are shown in Table 4.1, and their configuration spaces in Figure 4.1. All functions are considered in our evaluation experiments with two variables, in order to be easy to visualize in 3D. Sphere, Rastrigin, Ackley, and Schwefel can also be defined with different numbers of variables in a straightforward manner—and, as such, are also considered in 2D, 4D, 5D, and 6D.
For each function and dimensions considered, we allowed all algorithms to run for 100 trials for a predefined amount of time. Taking into consideration the dimensional complexity, the predefined time allowance is 1 sec, 5 sec, 1 min, 5 min, and 20 min for 2D, 3D, 4D, 5D, and 6D respectively. For each function-dimensions combination we evaluated the mean accuracy and precision of the algorithms. We calculated the accuracy percentage as the ratio of the trials where the algorithm converged to an approximation of the global minimum over the total number of trials. We consider an algorithm to have converged to an approximation of the global minimum, if the solution discovered is better than the second-best (local) minimum. To evaluate the precision of the algorithms, for those solutions that have converged to an approximation of the global optimum, we calculated the difference between that approximation and the global optimum itself. In

Table 4.1: Test-bed Functions

Function	**Equation**
Dropwave	$-\frac{1+\cos(12\sqrt{x_1^2+x_2^2})}{0.5(x_1^2+x_2^2)+2}$
Eggholder	$-(x_2+47)\sin(\sqrt{\lvert x_2+\frac{x_1}{2}+47\rvert})-x_1\sin(\sqrt{\lvert x_1-x_2-47\rvert})$
Holdertable	$-\lvert\sin(x_1)\cos(x_2)e^{\lvert 1-\frac{\sqrt{x_1^2+x_2^2}}{\pi}\rvert}\rvert$
Langermann*	$\sum_{i=1}^{5} c_i e^{-\frac{1}{\pi}\sum_{j=1}^{d}(x_j-A_{ij})^2}\cos(\pi\sum_{j=1}^{d}(x_j-A_{ij})^2)$
Shubert	$(\sum_{i=1}^{5} i\cos((i+1)x_1+i))(\sum_{i=1}^{5} i\cos((i+1)x_2+i))$
Easom	$-\cos(x_1)\cos(x_2)e^{-(x_1-\pi)^2-(x_2-\pi)^2}$
Ackley	$-20e^{-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{d}x_i^2}}-e^{\sqrt{\frac{1}{d}\sum_{i=1}^{d}\cos(2\pi x_i)}}+20+e^1$
Rastrigin	$10d+\sum_{i=1}^{d}(x_i^2-10\cos(2\pi x_i))$
Schwefel	$418.9829d-\sum_{i=1}^{d}x_i\sin(\sqrt{\lvert x_i\rvert})$
Sphere	$\sum_{i=1}^{d}x_i^2$

*Parameters c and A represent matrices, where they are $c = (1, 2, 5, 2, 3)$ and $A = \begin{pmatrix} 3 & 5 \\ 5 & 2 \\ 2 & 1 \\ 1 & 4 \\ 7 & 9 \end{pmatrix}$

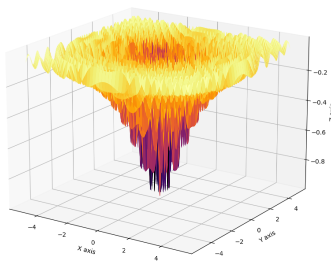
more detail, we calculated the Mean Absolute Error (MAE) as:

$$MAE = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n} \tag{4.1}$$

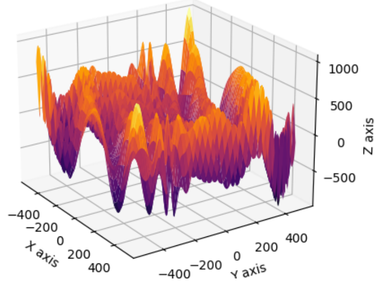
where n is the number of trials, y the global minimum, and x_i the proposed solution on a given trial. Algorithm parameters were optimized for each function-dimension combination via a manual grid search; the experiments ran on a 40-CPU Intel(R) Xeon(R) CPU E5-2680 v2 @ 2.80GHz processor, with 64GB RAM.

4.2 Results

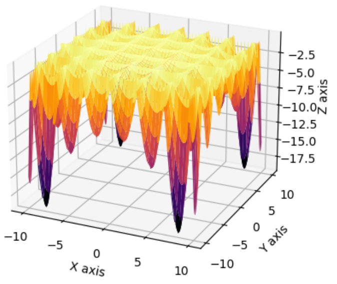
Our results are shown in Table 4.2 for each function, algorithm, and dimensions considered, with the best results in each occasion noted in bold. A higher accuracy percentage indicates better performance (the algorithm discovers and approximates the global optimum more often compared to the rest of the algorithms evaluated), while lower MAE indicates better precision. A MAE of ~ 0 is practically zero. The statistical significance of all results was tested using ANOVA (analysis of variance) and follow up Tukey tests. A p-value threshold of 0.05 is used for significance.
As seen in Table 4.2, BP shows higher or equal accuracy rates compared to all other algorithms in all settings. The starting point for each trial and for all algorithms is random. BP has no trouble approximating the global minimum from any possible starting po-



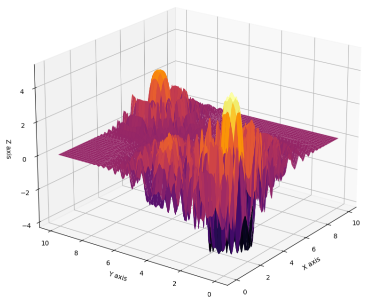
(a) Dropwave function



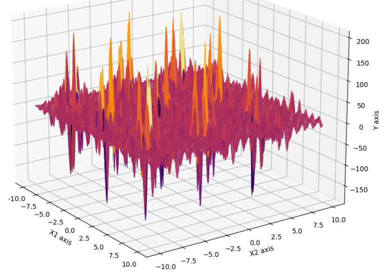
(b) Eggholder function



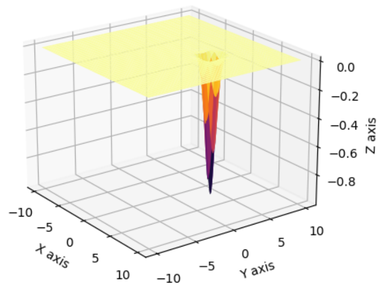
(c) Holdertable function



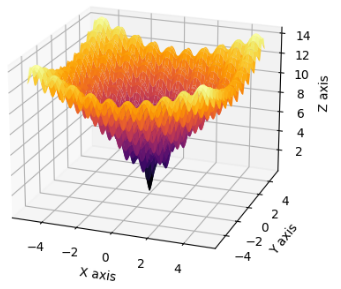
(d) Langermann function



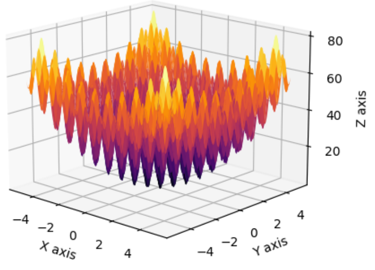
(e) Shubert function



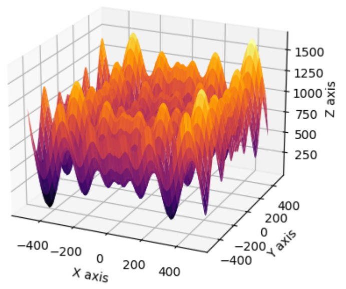
(f) Easom function



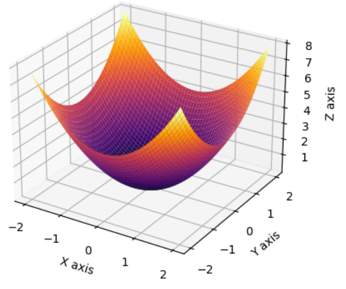
(g) Ackley function



(h) Rastrigin function



(i) Schwefel function



(j) Sphere function

Figure 4.1: Objective functions' graphs

sition. Notably, it is the only algorithm to achieve a 100% accuracy ratio in almost all settings.
There were some cases however where all algorithms achieved almost 100% accuracy. This occurs for the "less complex" functions Ackley, Easom, Rastrigin, Schwefel (2D), Sphere, Holdertable, Shubert, and Dropwave, where all or almost all algorithms exhibit almost 100% accuracy (we elaborate below). The more "complex" functions (as can also be verified visually in Fig. 4.1), are Schwefel (3D, 4D, 5D, 6D), Eggholder, Langermann. In all those cases, BP outperforms others, many times reaching 100% accuracy, and these results are statistically significant, as can be verified in the supplementary material; except results against TA for the Schwefel 3D, 4D, and 6D cases, and the Langermann case (the latter is only marginally not significant).
Now, regarding precision, we clarify it is calculated only for the points that have converged to an approximation of the global minimum. Thus, a high precision demonstrates how well the global optimum is approximated, if the algorithm did not get stuck to a local optimum in the first place. As such, a *high precision-low accuracy* performance is *not* suitable for global optimization—since, although precise, the algorithm is not discovering the global optima often enough. That said, an adequate performance with respect to precision is definitely required for global optimization algorithms. As can be seen, BP's precision is high; and it is higher than that of the other two single-point algorithms considered (SA, TA) in most cases. Follow-up tests (see supplementary material) confirm statistically significant better BP precision against SA and TA in all cases but: Rastrigin 2D against TA; Schwefel 2D, 3D, and 4D against SA; Ackley 4D against SA and TA and 6D against SA; Holdertable 3D against TA; and Dropwave 3D against TA (marginally). Furthermore, although the precision of BP seems to be marginally lower than that of PSO (which is also developed for continuous optimization tasks), follow-up tests indicate that this difference is *not* statistically significant in most cases (i.e., Rastrigin, Ackley, Shubert, Schwefel, Dropwave and Langermann for all respective dimensions considered). Notably, PSO frequently has the worst accuracy among all algorithms, and thus is a poor choice for global optimization in these cases.
As noted already, the advantage of BP compared to all other algorithms seems to become greater when the more complex functions are considered, i.e., Eggholder, Schwefel and Langermann, and when we move to higher dimensions. These functions have many and deep local minima, but only a single global one. The Eggholder and Langermann are also non-symmetric. These facts make them harder to optimize in a global optimization manner, while not getting stuck in a local minimum. The higher dimensionality introduces further challenges for global optimization. That said, BP manages to achieve a 100% accuracy in all occasions except Schwefel for 6D (where an 89% is achieved) and Langermann, 3D (where 73% is achieved). The accuracy results are also always better compared to the rest of the algorithms considered and the improvement ranges to up to 13.6 times better (i.e., compared to SA for Langermann 3D). When simpler functions and lower dimensions are considered the differences between the algorithms become less prominent. For instance, the Dropwave, Ackley, and Rastrigin have few local minima that are relatively shallow, while Shubert and Holdertable have many global optima. As such, most algorithms reach 100% accuracy except SA in Dropwave 3D, PSO in Rastrigin 4D, 5D and 6D, and PSO in Holdertable 3D and Shubert 3D. Finally, when the

Table 4.2: Evaluation Results

Function		**Accuracy (%)**				**Precision (MAE)**			
		BP	SA	TA	PSO	BP	SA	TA	PSO
Dropwave	3D	**100%**	84%	**100%**	**100%**	1e-4	0.008	0.002	**1e-17**
Eggholder	3D	**100%**	73%	77%	48%	0.872	0.454	4.634	**1e-5**
Holdertable	3D	**100%**	**100%**	**100%**	56%	0.006	0.01	0.008	**1e-6**
Langermann	3D	**73%**	5%	59%	56%	7e-5	0.001	0.007	**2e-15**
Shubert	3D	**100%**	**100%**	**100%**	88%	0.032	0.021	0.274	**8e-6**
Easom	3D	**100%**	**100%**	**100%**	99%	0.002	0.001	7e-4	**5e-17**
Ackley	2D	**100%**	**100%**	**100%**	**100%**	2e-4	0.009	0.002	**4e-16**
	3D	**100%**	**100%**	**100%**	**100%**	7e-5	0.017	0.013	**4e-16**
	4D	**100%**	**100%**	**100%**	**100%**	**8e-5**	0.03	0.026	0.021
	5D	**100%**	**100%**	**100%**	**100%**	9e-5	0.044	0.07	**2e-15**
	6D	**100%**	**100%**	**100%**	**100%**	**3e-4**	0.062	0.115	0.063
Rastrigin	2D	**100%**	**100%**	**100%**	**100%**	7e-7	0.005	5e-4	~0
	3D	**100%**	**100%**	**100%**	**100%**	8e-7	0.01	0.01	~0
	4D	**100%**	**100%**	**100%**	99%	6e-7	0.035	0.044	~0
	5D	**100%**	**100%**	**100%**	82%	7e-7	0.116	0.139	~0
	6D	**100%**	**100%**	**100%**	46%	7e-7	0.331	0.36	~0
Schwefel	2D	**100%**	95%	**100%**	95%	2e-4	6e-3	0.11	**1e-5**
	3D	**100%**	90%	93%	91%	3e-4	0.02	0.9	**2e-5**
	4D	**100%**	81%	91%	84%	3e-4	0.142	3.634	**3e-5**
	5D	**100%**	72%	86%	62%	4e-4	0.875	8.69	**3e-5**
	6D	**89%**	71%	77%	60%	4e-4	2.68	16.72	**4e-5**
Sphere	2D	**100%**	**100%**	**100%**	**100%**	0.036	5e-6	2e-12	**1e-117**
	3D	**100%**	**100%**	**100%**	**100%**	0.003	1e-6	2e-7	**4e-16**
	4D	**100%**	**100%**	**100%**	**100%**	8e-4	2e-5	9e-6	~0
	5D	**100%**	**100%**	**100%**	**100%**	7e-4	1e-4	6e-5	~0
	6D	**100%**	**100%**	**100%**	**100%**	7e-4	3e-4	2e-4	~0

unimodal functions, Sphere and Easom are considered, not surprisingly all algorithms achieve a 100% accuracy.

4.2.1 Detailed function evaluation

In order to further evaluate the performance of BP at various time allowance windows, we conducted additional experiments only for 3D, but for all functions, where a range of time allowances were considered. In particular, we evaluated all functions for 100 trials considering 0.05, 0.1, 0.2, 0.5, 1, 2, and 5 seconds. The parameters of the algorithms are optimized manually for the time allowance considered. The results further confirmed that BP is very fast in optimizing complex functions compared to the benchmark. Below we explain analytically the results of the multiple time allowances on each function. We also conduct a detailed analysis on the results of our multidimensional functions, along with their 3D results.

4.2.2 Dropwave

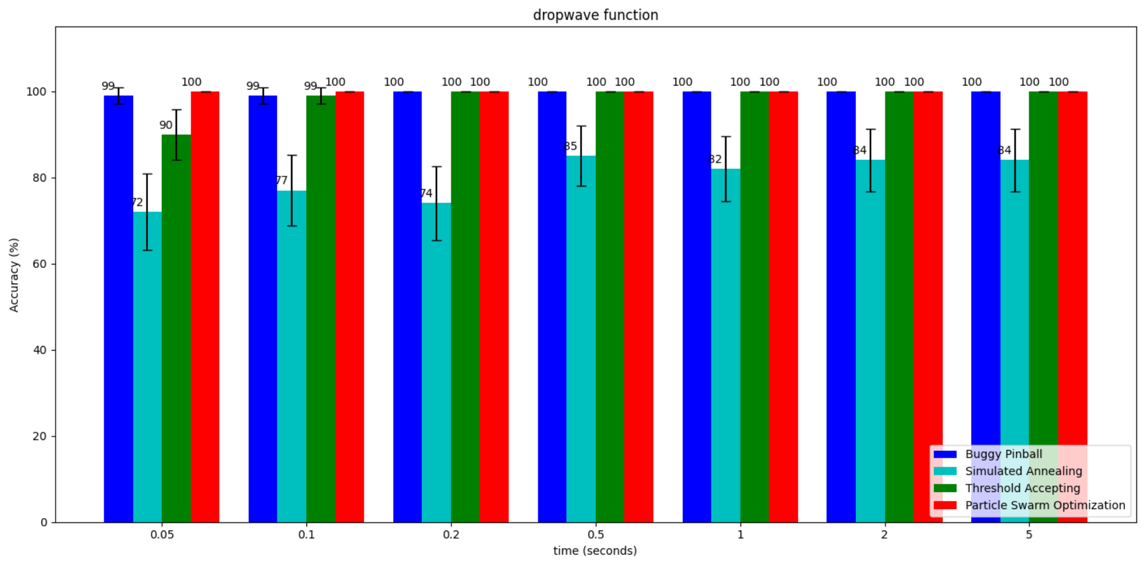
Dropwave is a relatively not complex multimodal function. It is implemented only in 3D space. Results of this function are depicted in Figure 4.2.
Due to its low complexity, all algorithms were able to achieve high accuracy results (Figure 4.1a). PSO in particular had 100% convergence, with BP following with the same results, except the first two measurements where it stopped at 99%. TA had a slightly worse performance with the 0.05 seconds measurement being at 90%. SA was the exception in that case, where it failed to reach the 100% rate at any point. Precision followed the same pattern as that of accuracy. There was a small superiority of PSO against BP, and with higher differences followed TA and SA. Overall statistical significance results, as can be seen in Tables B.1 and B.2, showed that only SA is significantly worse than all the other algorithms in accuracy, while the BP-PSO comparison was the only insignificant precision difference.

4.2.3 Eggholder

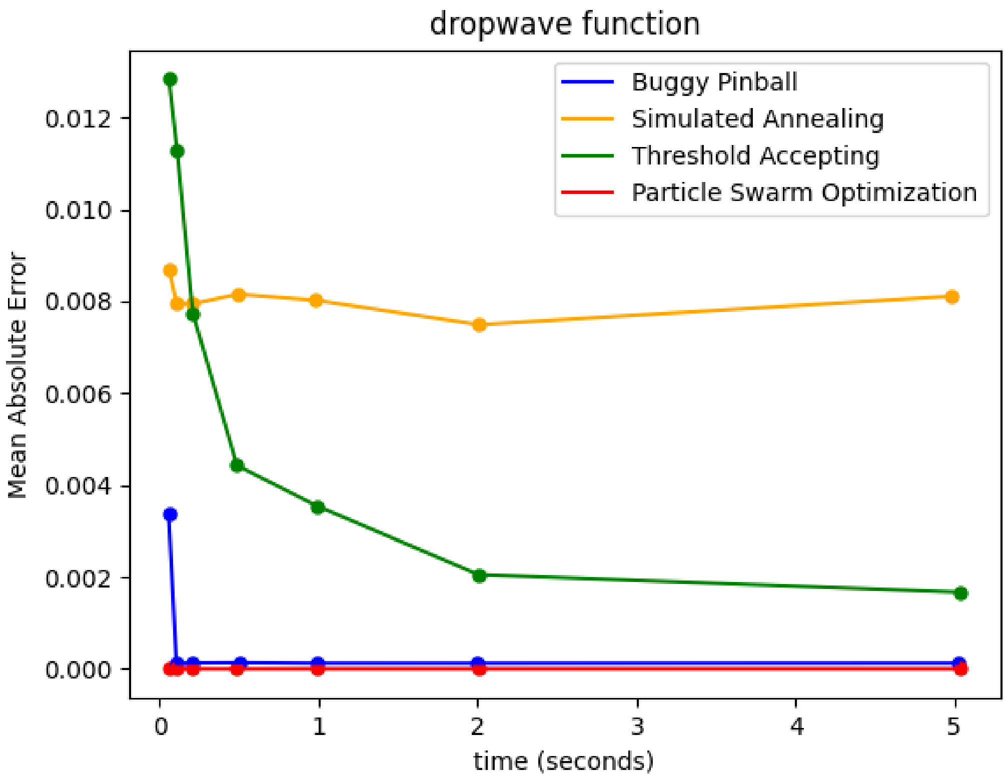
Eggholder is the most complex function in our experiments. It is a relatively large space (range [-512, 512]), consisted of a big amount of local minima which are deep and unevenly distributed, as can be observed in Figure 4.1b. Because of this formation, many algorithms often fail to approach the global solution. As it has already been mentioned in Section 1.3, we have developed BP to successfully tackle problems with that level of complexity.
We can observe in the results, also shown in Figure 4.3, BP succeeds in our goal, since it shows a constantly better accuracy. In precision though we observe a relatively low performance, compared to SA and PSO, but still better than TA. Statistical significance results (Tables B.3 and B.4) confirmed the importance of the accuracy differences, not only against BP, but also among the other algorithms. Precision differences also seem significant, with the exception of the BP-SA comparison, where the p-value exceeds the significance limit.

4.2.4 Holdertable

Holdertable is a relatively simple function with many global minima, depicted in Figure 4.1c. This means that more than one points are accepted as optimal solutions. This structure allows algorithms to easily reach towards a global minimum from multiple starting points.
As can be observed by the results in Figure 4.4, all single-point algorithms achieve 100% accuracy even in small time allowances, while PSO is constrained around 60% in every occasion. Precision results show that PSO is far better than the other algorithms, followed by BP, which has slight differences with SA, and the least precise being TA. Statistical significance analysis (Tables B.5 and B.6) showed that accuracy differences

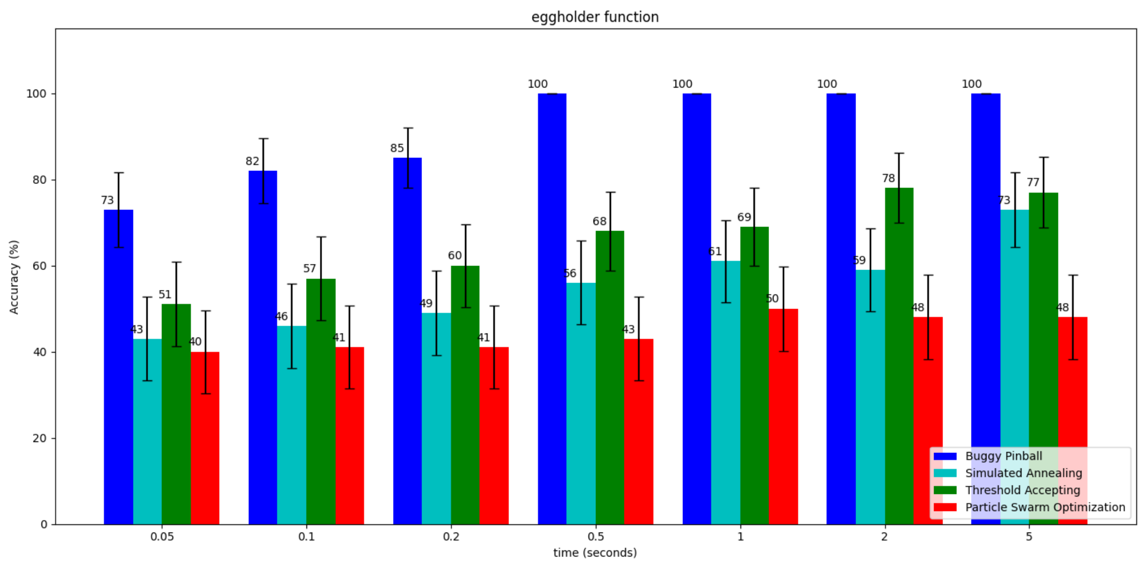


(a) accuracy

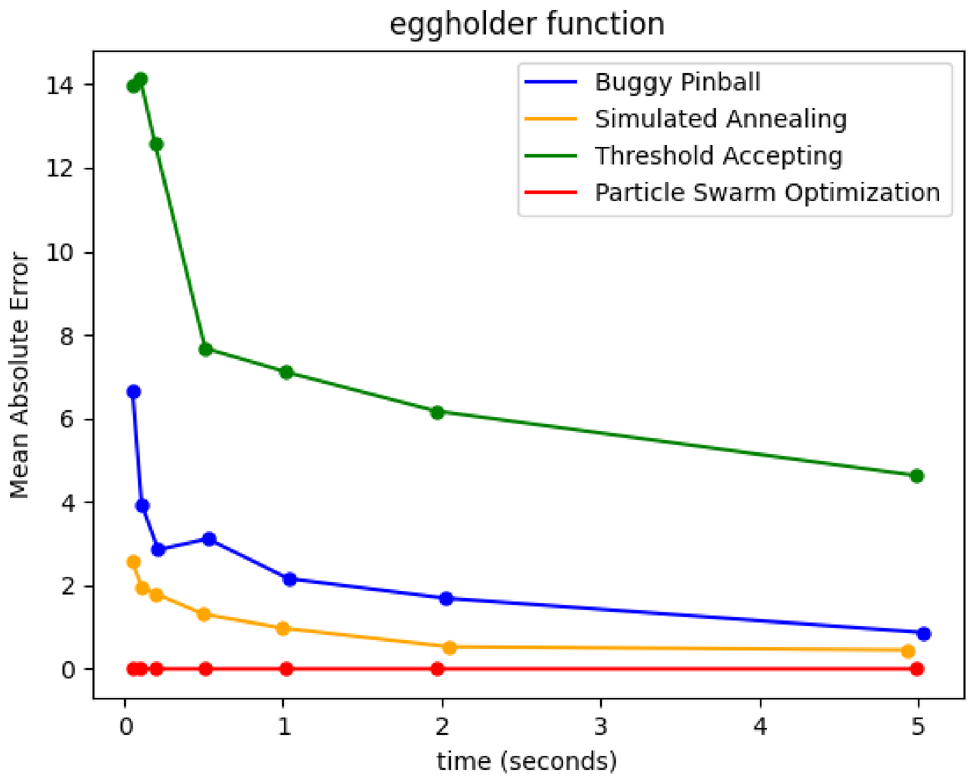


(b) precision

Figure 4.2: Dropwave function analysis



(a) accuracy



(b) precision

Figure 4.3: Eggholder function analysis

are statistically significant, while in precision, the difference of BP and SA is not important.

4.2.5 Langermann

The Langermann function is a unique case of an optimization function. That is because of its unevenly distributed configuration space, as well as the fact that some of its local minima are located very close to the global minimum (local minimum ≈ -4.127577 and global minimum $\approx -4.15580929184779$). The function's configuration space is shown in Figure 4.1d.
Due to that, no algorithm managed to achieve 100% accuracy at any given time, as shown in Figure 4.5. PSO had the best results at an early stage, but BP showed to be constantly improving, achieving 73% at 5 seconds. TA also showed some improvement over time, but at lower values than BP. SA failed to approximate the global minimum adequately, since results did not exceed 10%. Precision was slightly better in PSO than BP, followed by SA and then TA, where as more trials succeeded in approximation, the overall MAE seemed to be getting worse. Statistical significance results showed that all differences between BP and PSO are insignificant (Tables B.7 and B.8).

4.2.6 Shubert

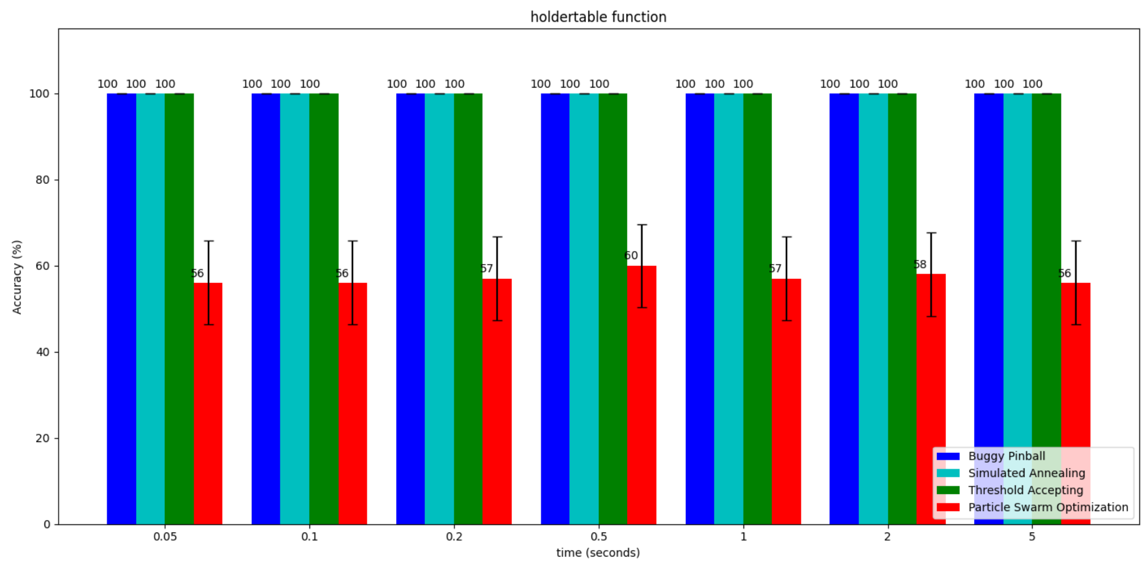
Shubert function is another many-global minima function (Figure 4.1e). Results, similarly to the Holdertable function, show that all algorithms except PSO achieve 100% accuracy.
Accuracy differences are smaller due to the fact that shubert has many global minima, as observed in Figure 4.6. Statistic tests showed that even though smaller, the differences are significant, as can be seen in Table B.9. PSO remains superior compared to all other algorithms, and BP being only second-best, with their difference being not significant, according to the statistical tests (Table B.10).

4.2.7 Easom

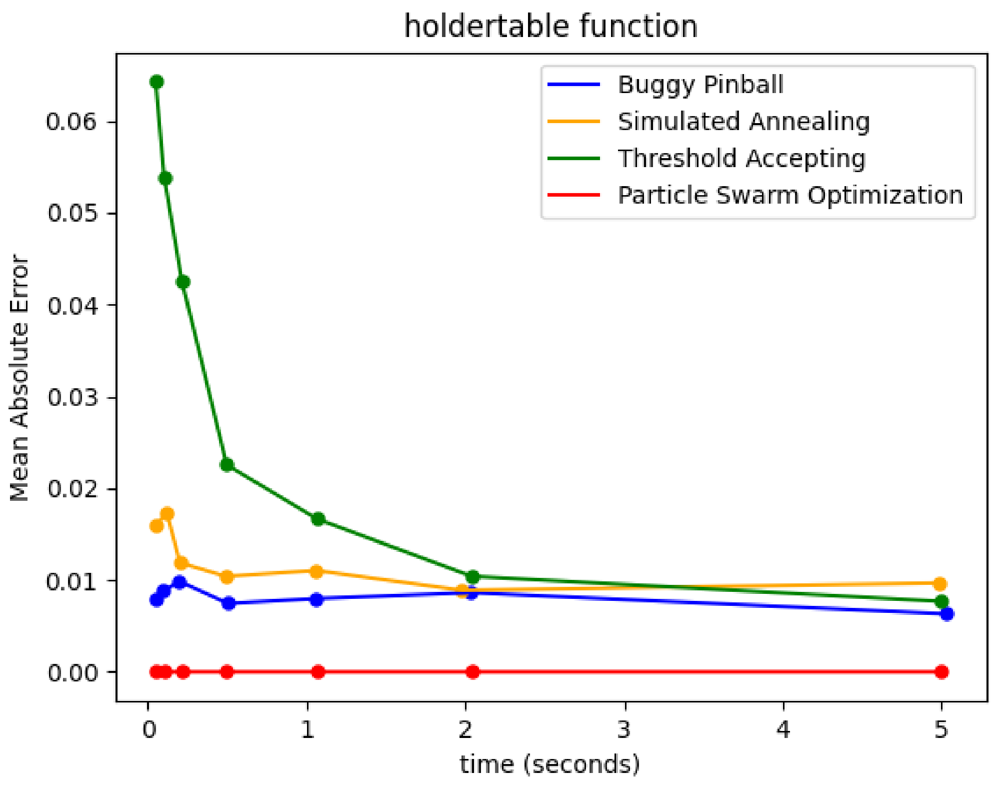
Easom is a unimodal function, where the whole space is flat, and it is possessing only one deep global minimum (Figure 4.1f). Since it is a relatively easy function to approximate, BP does not appear to perform that well as the other algorithms, as depicted in Figure 4.7.
In accuracy, all algorithms reach 100%, with a few exceptions in BP and PSO, where the accuracy is 99%. In precision, BP has the poorest performance, and PSO the best. According to statistical significance results, the differences of SA and TA against PSO are the only significant ones, while all precision results are considered noteworthy (Tables B.11 and B.12).

4.2.8 Ackley

Ackley is a multimodal and multidimensional function. It has a relatively simple structure, with many but not very deep local minima, as can be seen in Figure 4.1g. Similarly

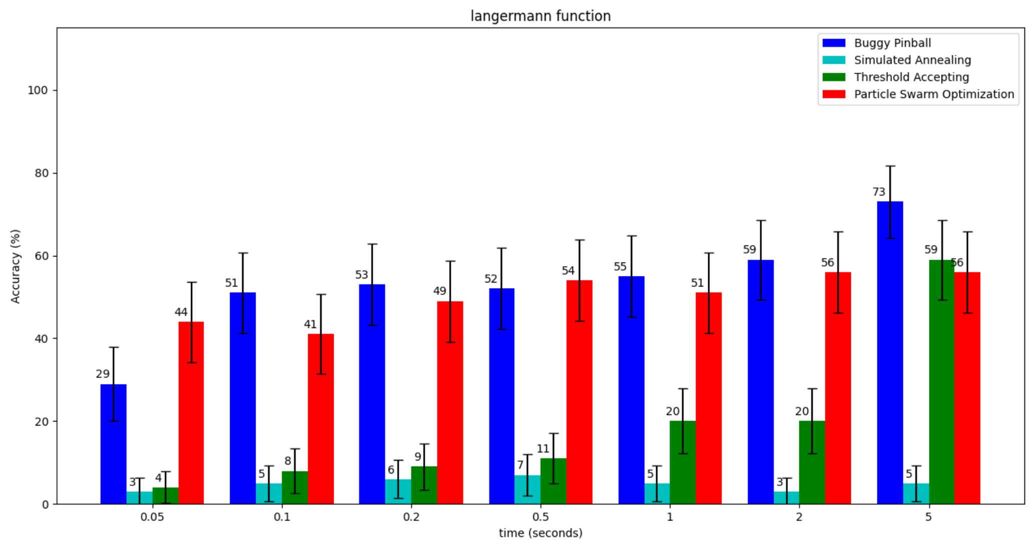


(a) accuracy

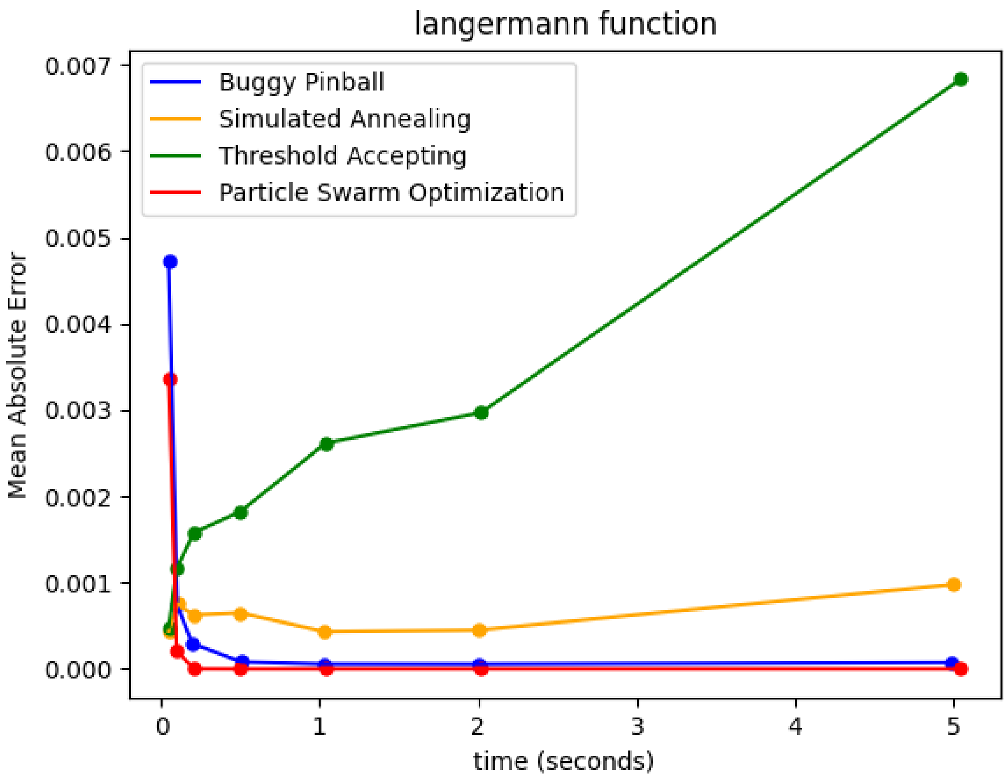


(b) precision

Figure 4.4: Holdertable function analysis

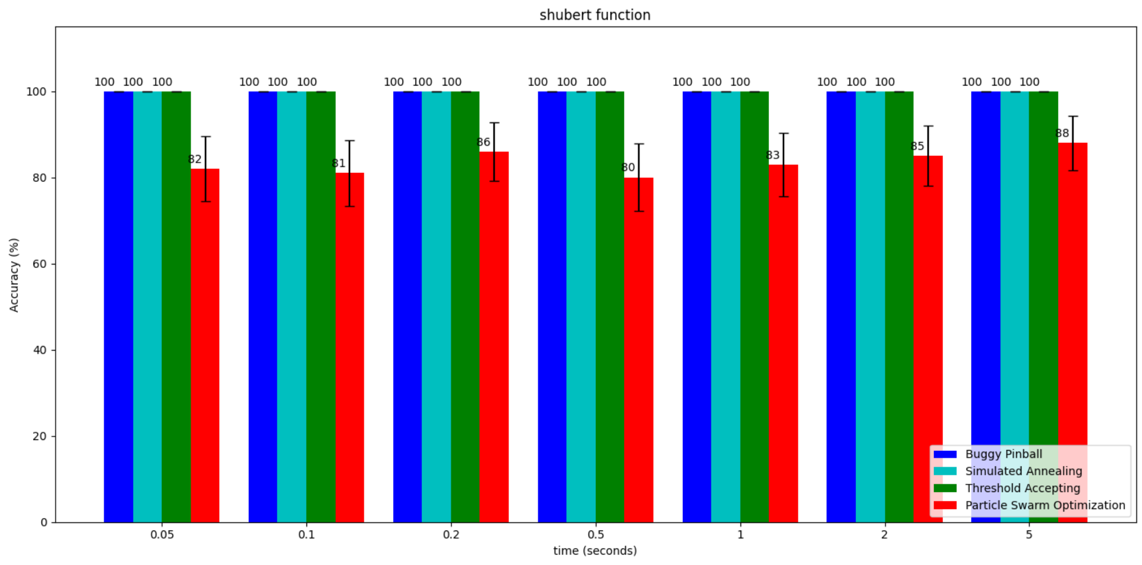


(a) accuracy

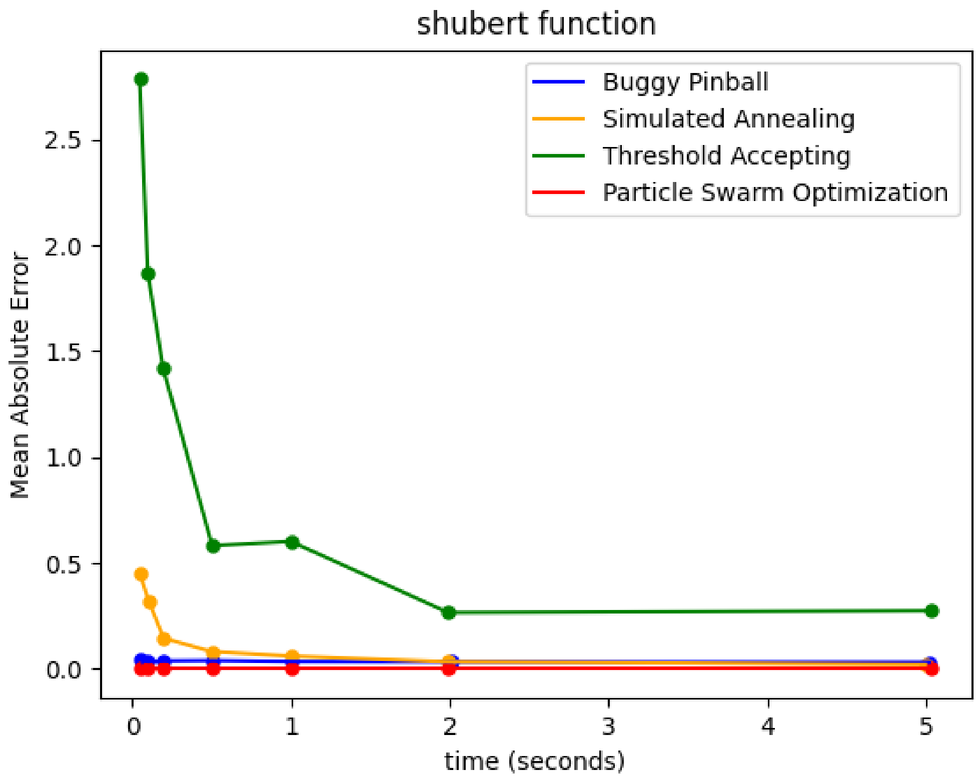


(b) precision

Figure 4.5: Langermann function analysis

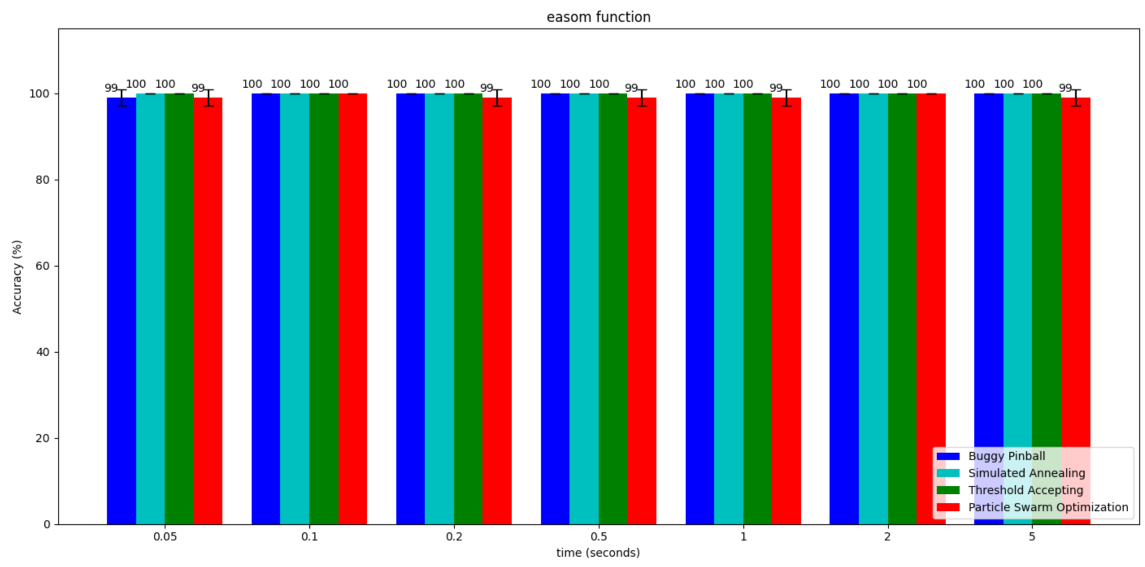


(a) accuracy

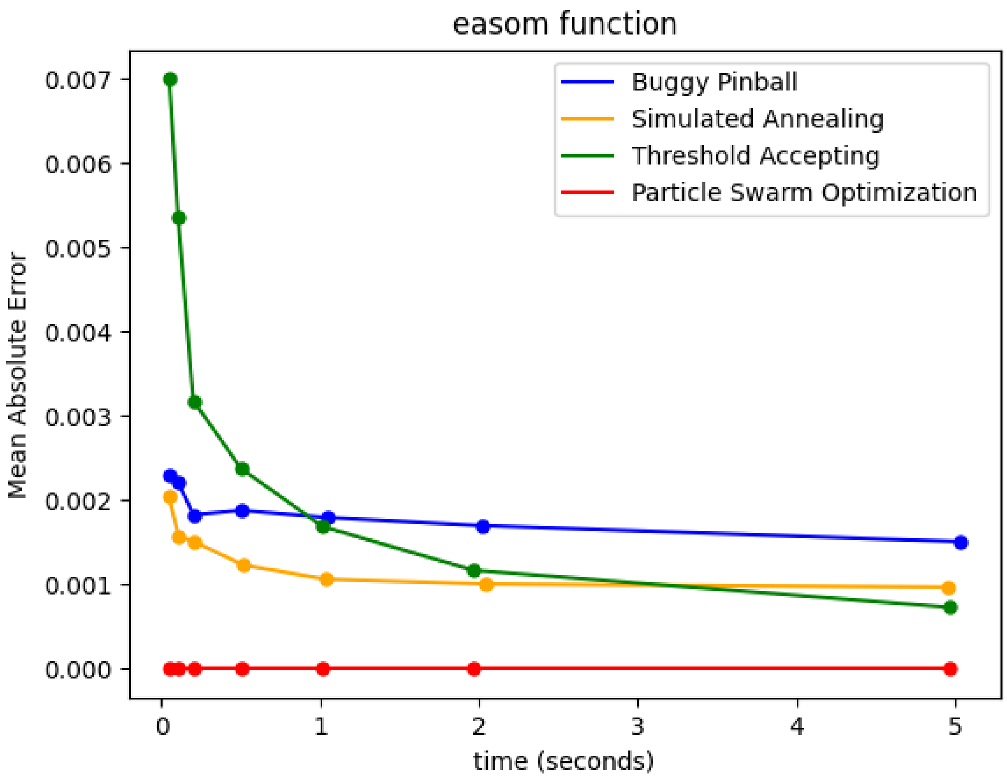


(b) precision

Figure 4.6: Shubert function analysis



(a) accuracy



(b) precision

Figure 4.7: Easom function analysis

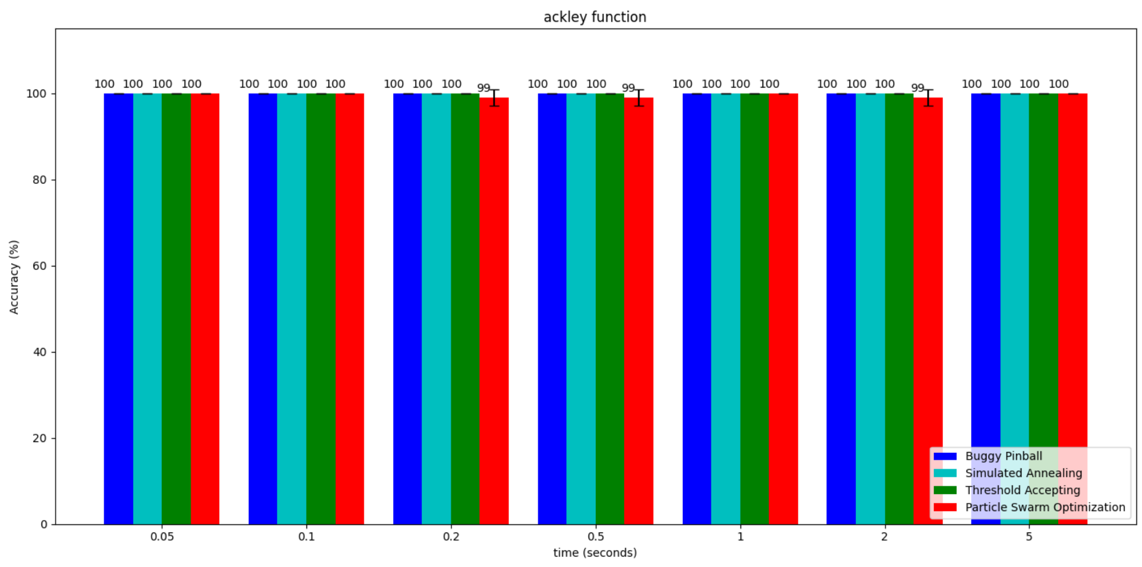
to the aforementioned functions, multiple time allowances were chosen in the 3D space, while for the rest, only one time value is examined.
In 3D space, all algorithms reach 100% convergence, with few exceptions of PSO (Figure 4.8), which are deemed as insignificant, according to the statistical tests (Table B.15). Precision however, is much better in BP and PSO than that of SA and the other three algorithms than TA. Statistical tests confirmed that the only insignificant differences are that of BP and PSO.
In more dimensions, depicted in Figure 4.9, algorithms are still able to achieve 100% accuracy. In precision, we observe that as dimensions increase, SA and TA are becoming less precise, and in a few cases, PSO too. BP on the other hand, preserves the same level of precision, throughout all dimensions. Precision differences of BP are statistically significant in 2D and 5D against SA, all but 4D against TA and none against PSO (Tables B.13-B.22).

4.2.9 Rastrigin

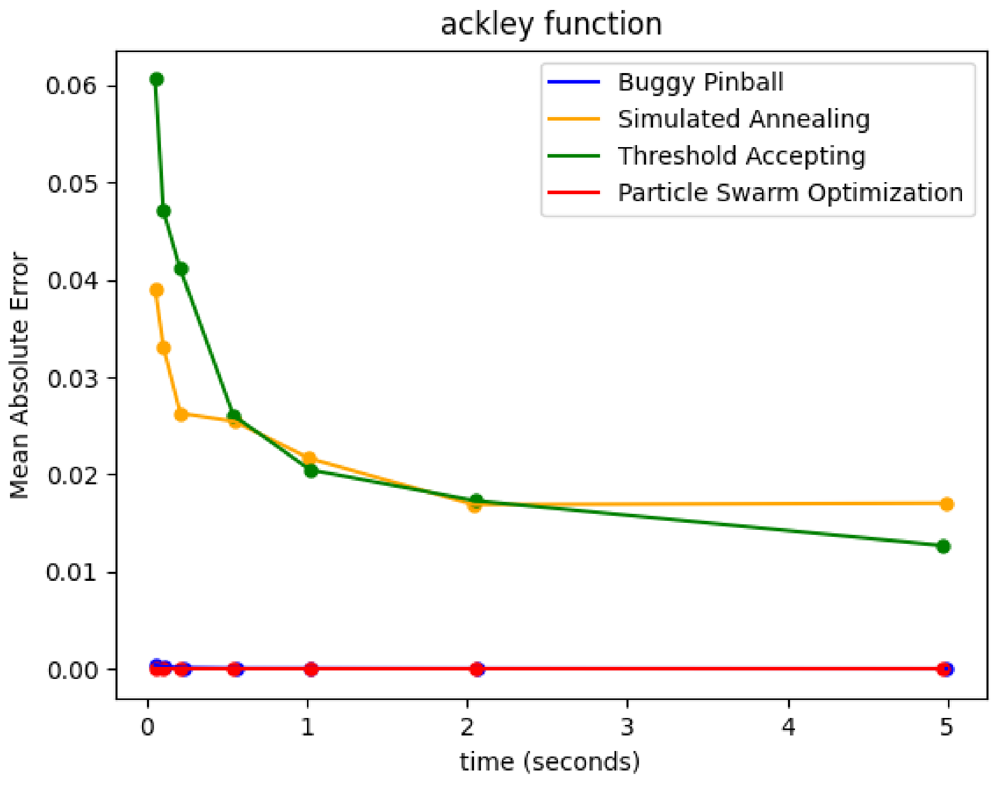
Rastrigin function is very similar to Ackley. We consider it to be slightly more difficult due to deeper local minima. Therefore, there is some similarity to the results of the previous function.
We observed that in 3D space, results in Figure 4.10, algorithms had no trouble to detect the global minimum, with few exceptions of PSO, which were statistically significant, according to Table B.25. In precision, PSO remained the best algorithm, but with insignificant differences against BP. As can be seen in Table B.26, only the differences of these two algorithms against SA and TA were notable.
The single-point algorithms of our experiments, showed the same performance in accuracy, as dimensions were increasing (Figure 4.11), in contrast to PSO, where its performance was deteriorating significantly with the increase of dimensions. In precision though, SA and TA were the ones that its solutions were less precise with that increase. Statistical significant appeared the accuracy differences against PSO in 5D and 6D, and the precision differences of all algorithms against PSO in 2D, and all but the BP-PSO comparison in the rest (Tables B.23-B.32).

4.2.10 Schwefel

Schwefel is one of the highly complex functions of our experiments. It has a similar range to Eggholder ([-500, 500]) for its variables, and deep local minima as well. Their main difference is that Schwefel is symmetric. Schwefel is depicted in Figure 4.1i. Many algorithms appear to be failing in approximating the global minimum in many occasions, when tested in Schwefel, unlike BP.
In our 3D analysis, which we show in Figure 4.12, only BP managed to reach 100% of accuracy, even on an early stage. This is a major difference to the other algorithms, which are much less accurate, especially on smaller time allowances. All accuracy differences appear significant, according to statistical tests (Table B.35). Precision did not have many differences among algorithms, with the exception of TA being much worse than the others, having the only statistical significant differences (Table B.36).

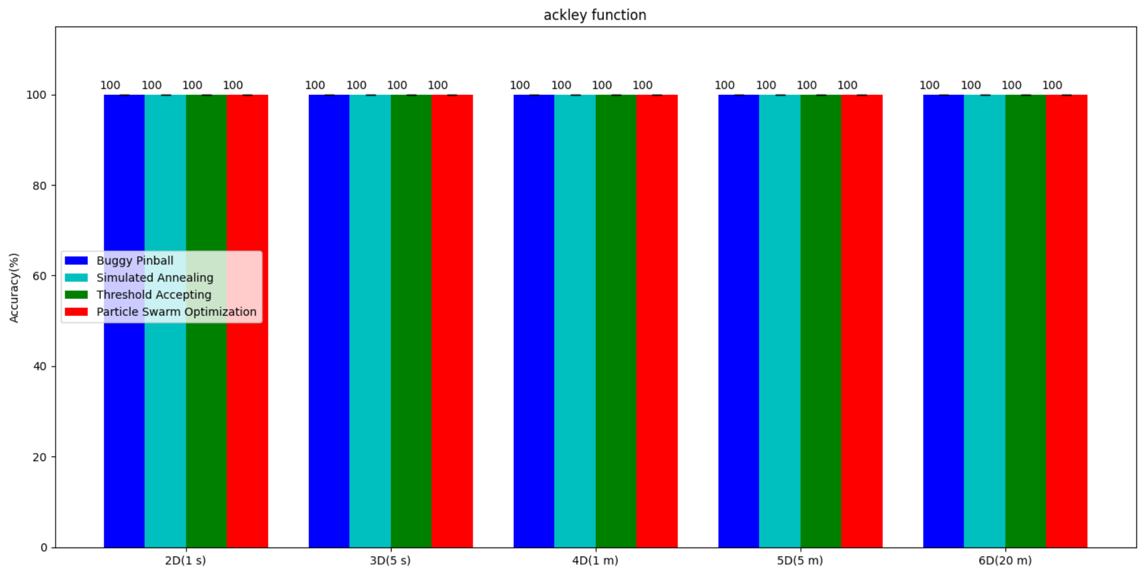


(a) accuracy

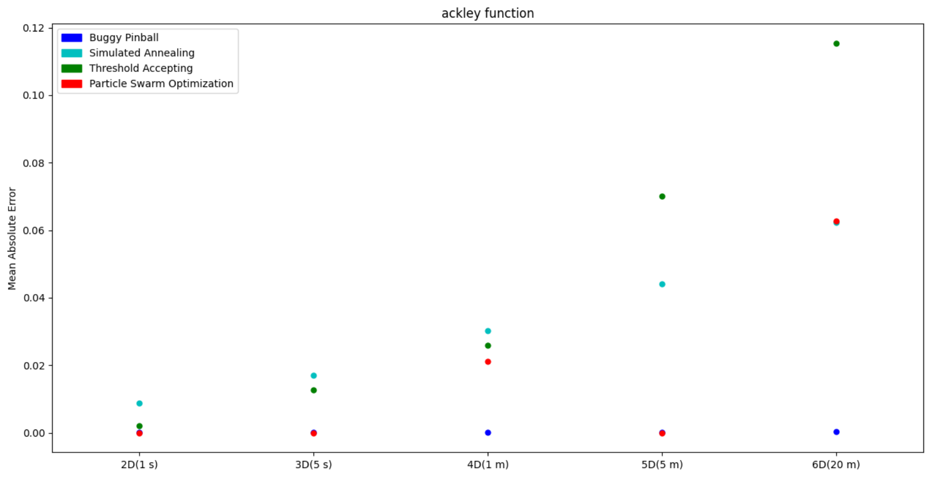


(b) precision

Figure 4.8: Ackley function 3D analysis

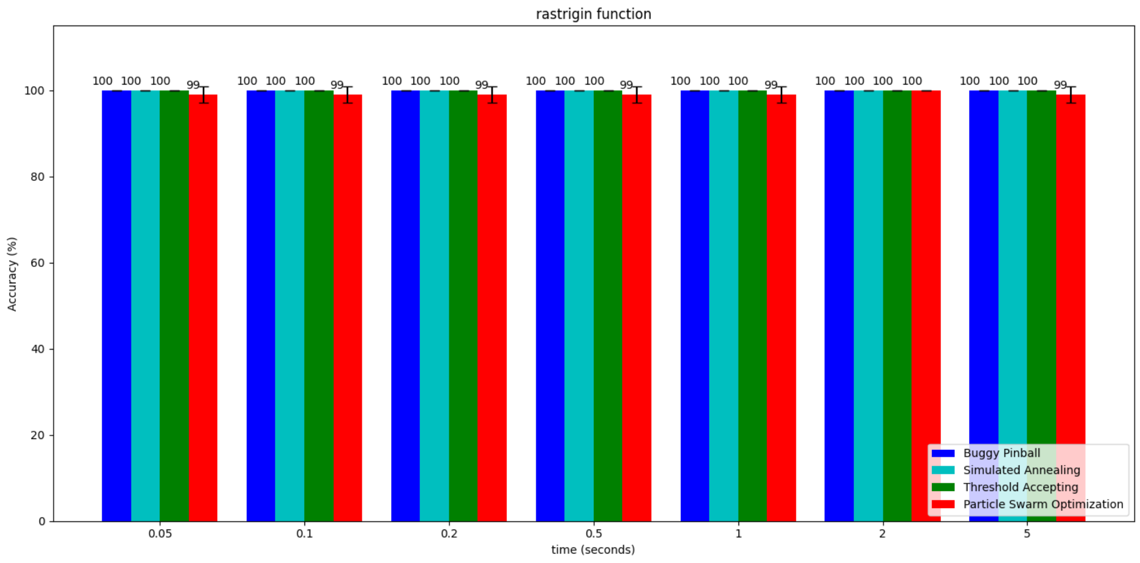


(a) accuracy

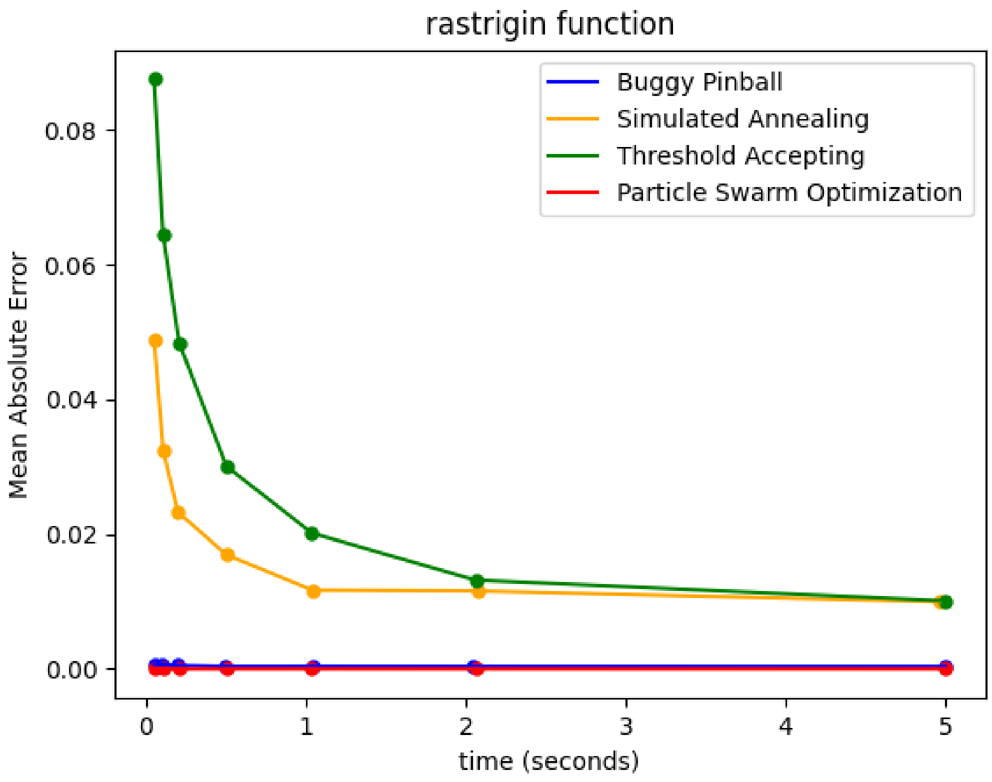


(b) precision

Figure 4.9: Ackley function multi-dimensional analysis

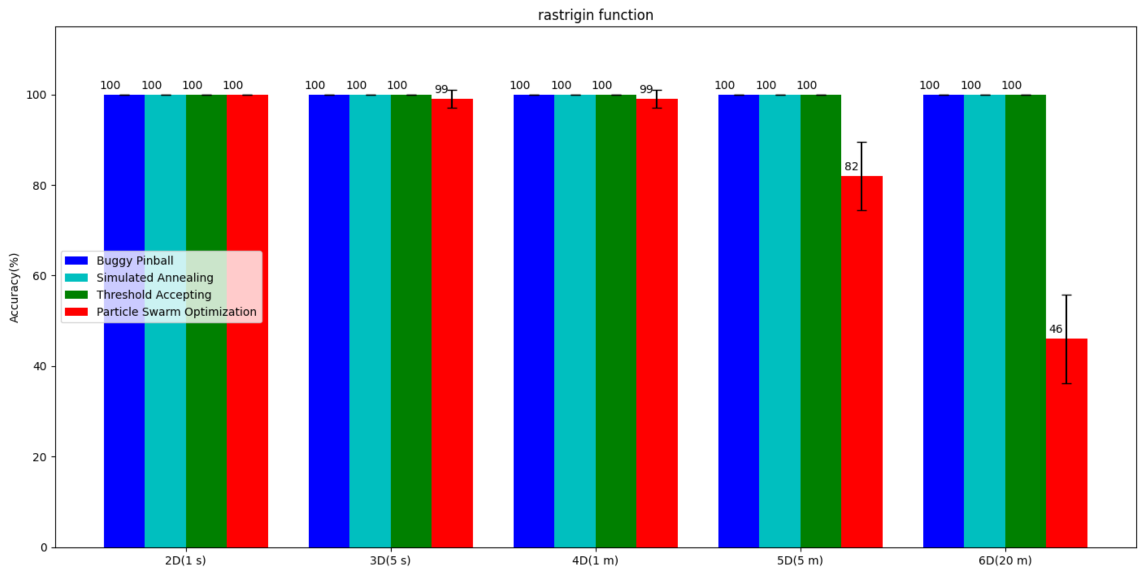


(a) accuracy

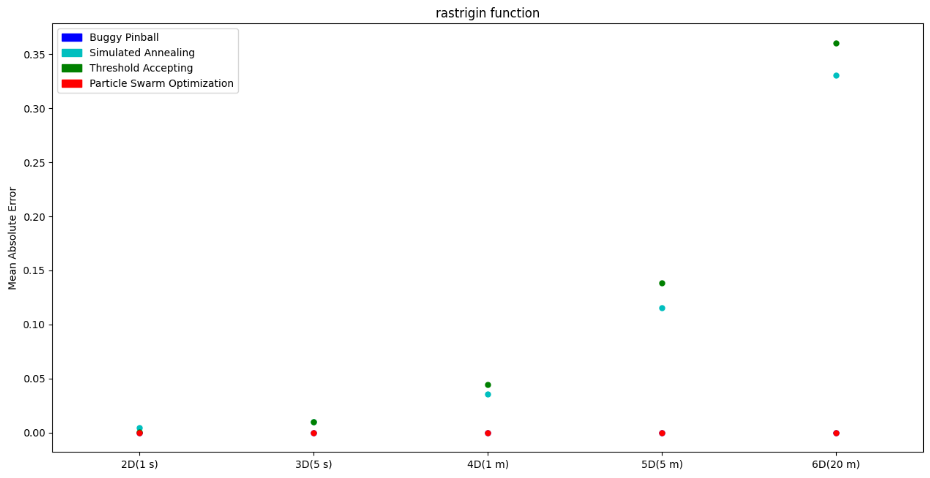


(b) precision

Figure 4.10: Rastrigin function 3D analysis



(a) accuracy



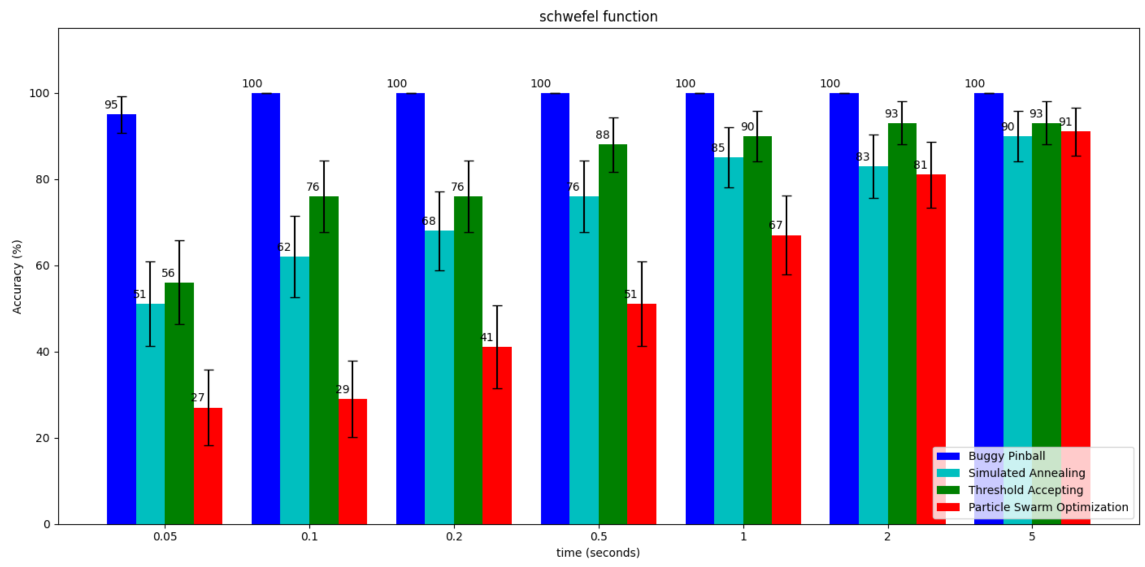
(b) precision

Figure 4.11: Rastrigin function multi-dimensional analysis

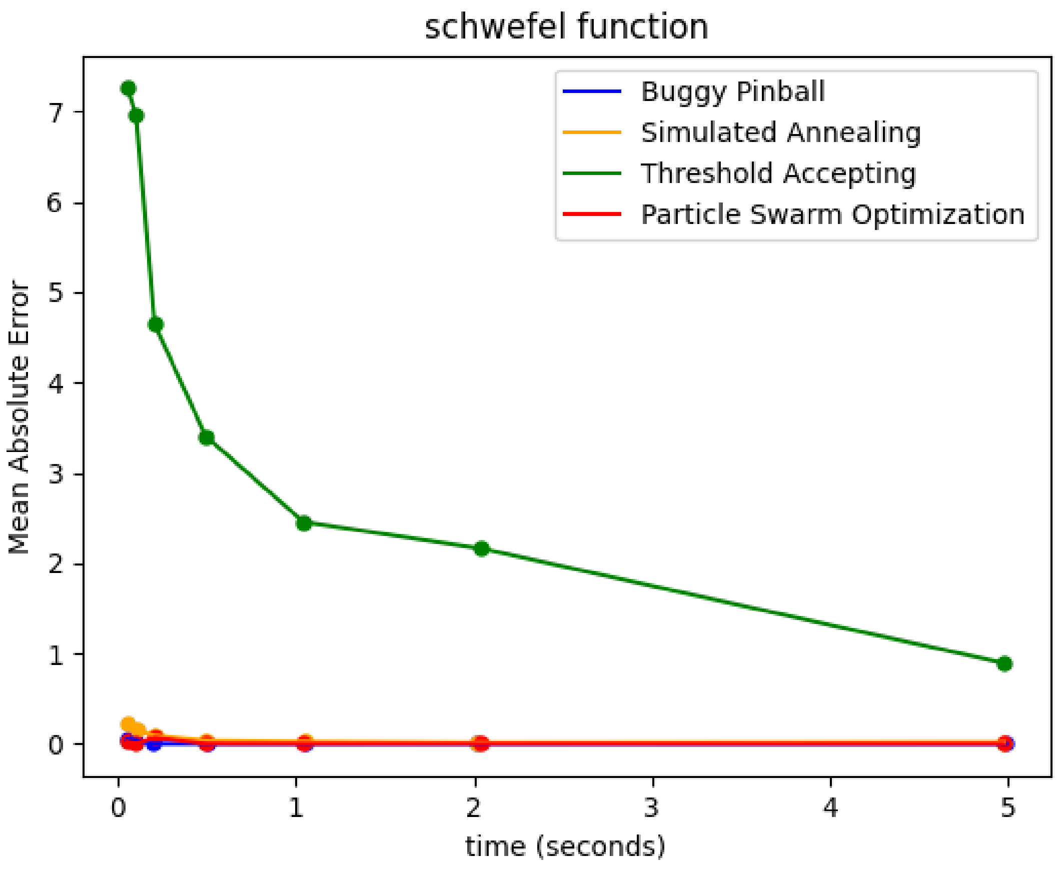
When we increased the dimensions, results remained similar. BP reached 100% accuracy in every number of dimensions, but the 6D, and was always the highest. TA was the only algorithm that was comparable to BP, as the statistical tests showed. In precision, SA and TA had worse results than BP and PSO, and they kept worsening as dimensions increased. For SA, differences were statistically significant from 4D and above. (Tables B.33-B.42)

4.2.11 Sphere

Sphere is the simplest function out of all implemented it this research. It is a simple unimodal valley-shaped function. It is shown in Figure 4.1j. Due to this formation, all solutions proposed are considered accurate.
All results show that BP has the worst performance (Figures 4.14 and 4.15). The simplicity of the function makes the algorithms that are able to perform more rounds to look superior. BP has the most complex structure compared to all other algorithms, therefore, its rounds requiring more time. As dimensions were increasing, and the function's complexity was increasing as well, the difference of BP and the other algorithms was decreasing, but it remained statistically significant as Tables B.44, B.46, B.48, B.50, and B.52 showed.

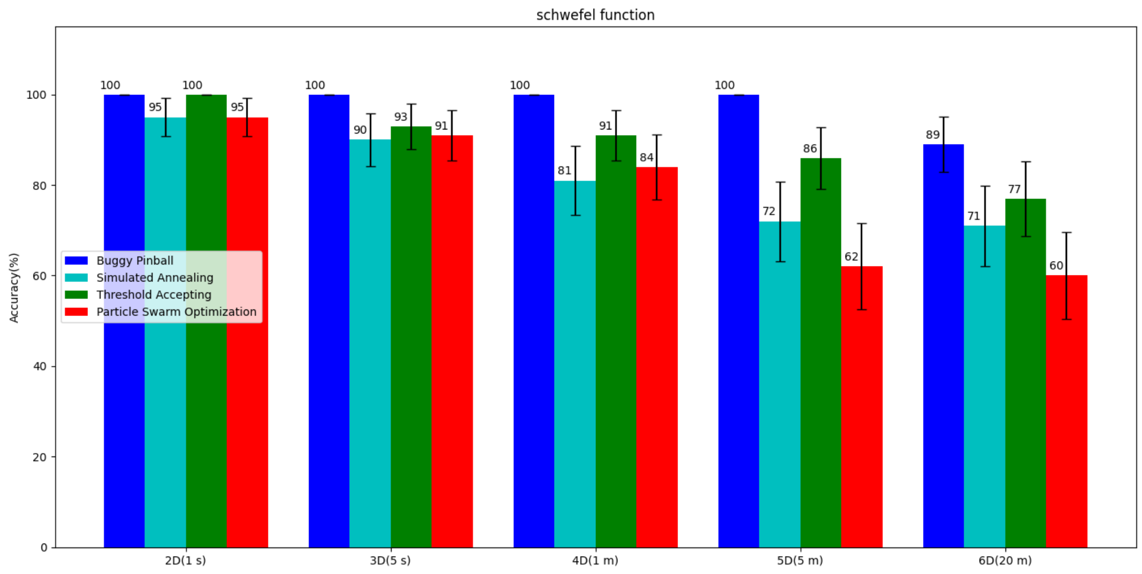


(a) accuracy

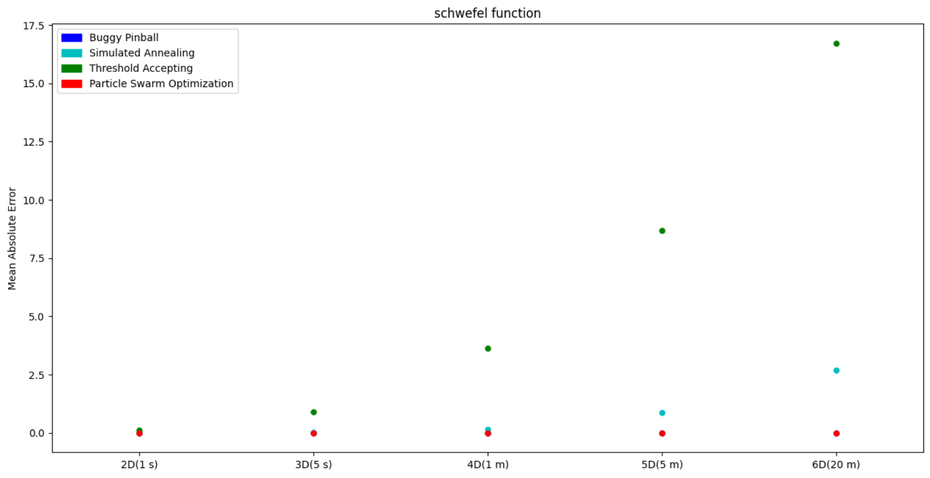


(b) precision

Figure 4.12: Schwefel function 3D analysis

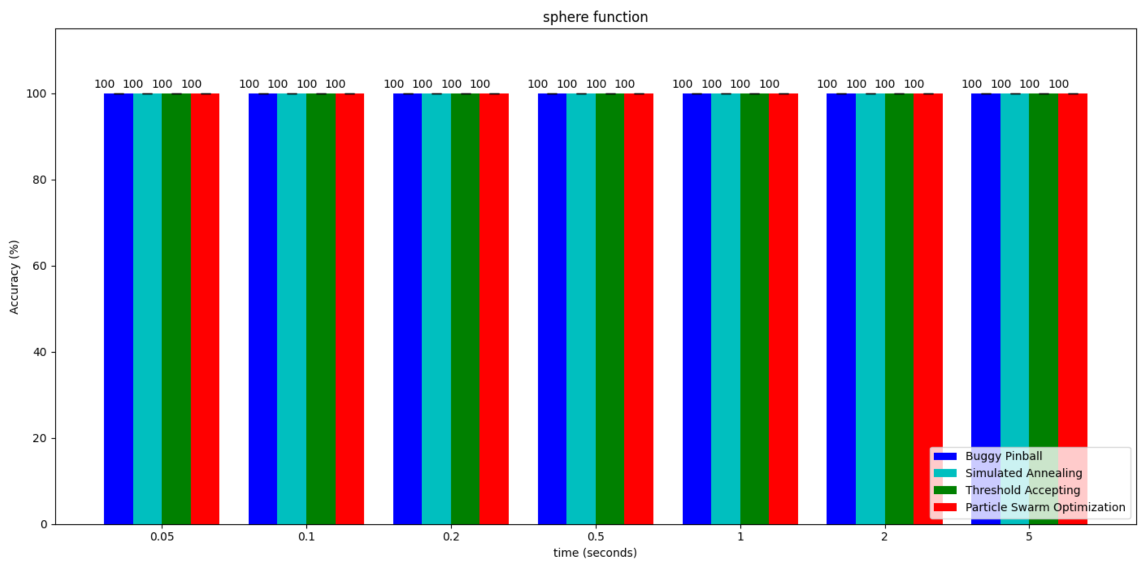


(a) accuracy

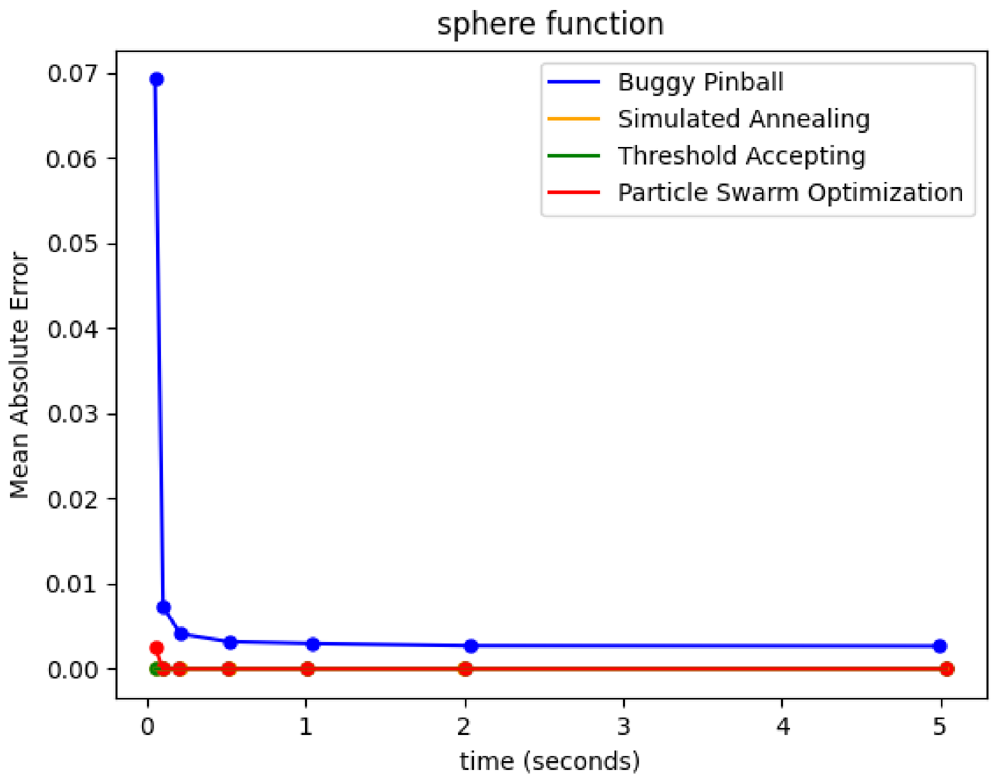


(b) precision

Figure 4.13: Schwefel function multi-dimensional analysis

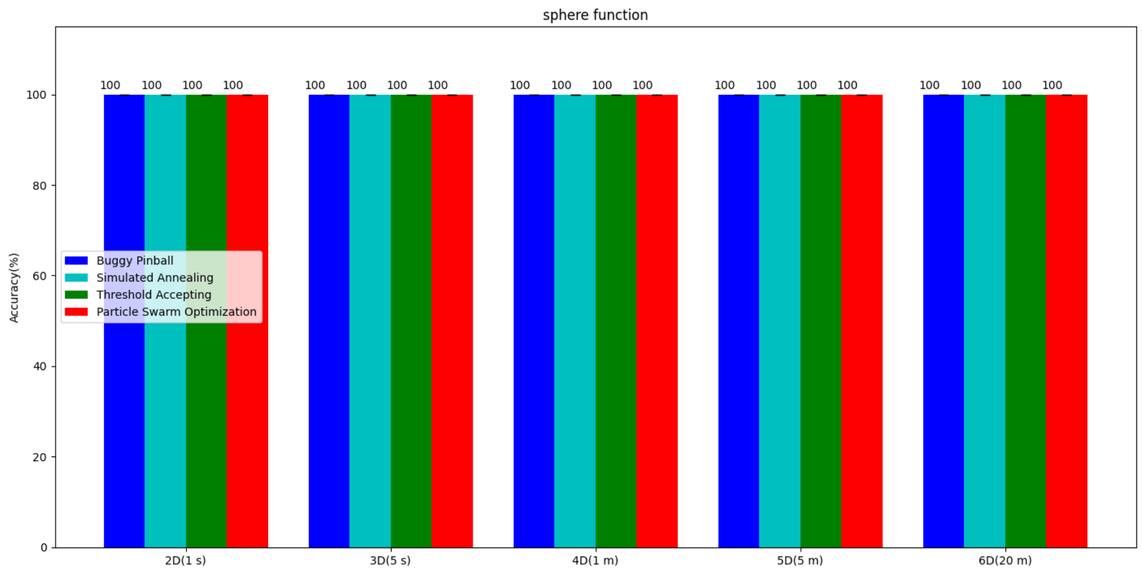


(a) accuracy

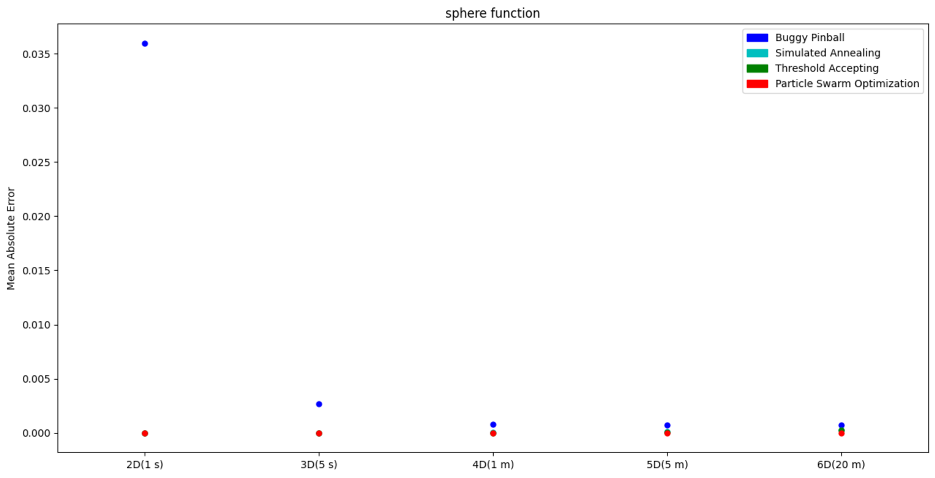


(b) precision

Figure 4.14: Sphere function 3D analysis



(a) accuracy



(b) precision

Figure 4.15: Sphere function multi-dimensional analysis

5 Conclusions

In this work, we have proposed a novel single-point meta-heuristic tailored for global continuous optimization problems. The algorithm, namely *buggy pinball*, is inspired by the pinball arcade game and is able to discover the global optimum in an any-time optimization manner. Through the trajectory-based search, the algorithm avoids local optima efficiently, even when tested on complex optimization problems. We evaluated our algorithm against a number of popular meta-heuristics, namely *simulated annealing*, *threshold accepting* and *particle swarm optimization*, in standard test-bed functions. We showed that our algorithm has better overall performance. We achieved significantly better levels of accuracy in most occasions, while on precision, our performance was comparable to the others. Especially with problems of high complexity, the superiority of our algorithm was vast in terms of accuracy, since we achieved nearly perfect results even in very small time allowances.

5.1 Future work

We believe this thesis opens many possibilities for future work. To begin, the performance of our algorithm in unimodal functions showed that due to its high complexity, other algorithms might perform better in terms of precision. A simpler version of buggy pinball could potentially be created in the future, where it would make BP more competitive in such problems.

A factor that is perhaps consuming more time than necessary on each round is the number of maximum steps for each trajectory. If a schedule was applied into that parameter, in order to decrease or increase it according to the needs of the problem, it would lead to an algorithm, where every round would make only essential steps. Another way to minimize the number of unnecessary steps, is to use the limits of the variables, when these are known.

The parameters of step length and elevation angle are already following a cooling schedule. Similarly to the temperature parameter of simulated annealing, further research could be conducted, in order to find the optimal schedule for each parameters, which it would possibly lead to further increase both the accuracy and the precision of the algorithm.
The precision of buggy pinball might also have room for improvement in possible future work. On the other hand, particle swarm optimization seems to perform adequately on that field, while it significantly loses on accuracy. A potential unification of these algorithms would allow us to benefit from the perks of each one. A hybrid particle swarm optimization - buggy pinball algorithm could be created, where the particles of PSO would perform a trajectory-based search in combination with following the global best.
Last but not least, buggy pinball currently cannot be adapted to discrete optimization problems. Another extension that would translate the trajectory-based exploration into combinatorial problems, would expand significantly the possible uses of the algorithm.

A
Tested Versions and Parameters

During the development of Buggy Pinball we met various obstacles. These obstacles caused various updates on the initial plan of the algorithm, which lead to the development of BP. In this appendix we will explain some of the decisions that we took, so it becomes clear to the reader why some of these choices were made.
The first decision that we made regarding the implementation of the trajectory based search was the way that we would get each point. The progression of the trajectory could be implemented in two ways. One option was to use the derivative of the objective function, in order to locate the exact crossing point with each trajectory segment. This choice would give us the crossing point very fast, since the usage of the recursive refining function would no longer be necessary. The disadvantage of this method though, is that it limits the objective functions that can be searched, only to those that we can calculate their derivative. That is why we chose to use the steps approach, where using the direction of a random vector, we examine points on the objective function, until a crossing point is detected.
The random direction of the vector that we move on each round, is also something that we examined thoroughly. In a pinball game, a ball does not move completely random, but it changes its direction successively. This approach would only be effective in the 2D versions (1 variable, 1 cost axis) of the objective functions that we experimented, as more variables increase the complexity exponentially. After numerous trials of trying to control the direction of the vector, we noticed that setting the direction of the vector (in the variables axes) completely random is the most effective approach.
Moving completely random gives us the best exploration of space, but it also does not allow us to control the rate of increase or decrease in the cost axis (whether we minimize or maximize). That is why it is important to determine the angle that our search vectors move on y. So, using the random variables and the angle on each round, we calculate the y value of the vector regarding these values. At the end, we scale them all, in order to achieve both the step and the angle that we desire on any given round.

The number of steps was also added to overcome another barrier. In most cases, if we do not find a crossing point, and we terminate the round once a variable is off limits, makes our search very accurate. Therefore, there are configuration spaces, where either we do not know their variables' size or they are too large to be set as limits. In order to make our algorithm capable of working on any given problem, we determined that setting the number of steps that would be made at maximum within every round, allows us to search under any circumstances. If the number of steps is reached, the round is terminated without providing a new solution. Then, a new round with the same starting point and a different direction will begin.
Lastly, we will show in Tables A.1, A.2, A.3, and A.4 the parameters that were used in our experiments for each algorithm. These parameters were empirically chosen, with respect to each experiment, so they might not be optimal in every case.

Table A.1: Experiments' Buggy Pinball Parameters

Function		Rounds	Steps	Step Size	Angle
Dropwave	3D	61000	40	$[-0.2, 10^{-4}]$	$[1, 10]$
Eggholder	3D	90000	100	$[-60, 10^{-4}]$	$[0.1, 1]$
Holdertable	3D	168000	20	$[-0.5, 10^{-4}]$	$[1, 45]$
Langermann	3D	34000	100	$[-0.8, 10^{-4}]$	$[1, 10]$
Shubert	3D	72000	100	$[-2, 10^{-4}]$	$[45, 89]$
Easom	3D	125000	100	$[-1, 10^{-4}]$	$[1, 10]$
Ackley	2D	12500	40	$[-0.5, 10^{-4}]$	$[10, 60]$
	3D	55000	40	$[-0.5, 10^{-4}]$	$[10, 60]$
	4D	340000	40	$[-0.5, 10^{-4}]$	$[10, 60]$
	5D	1360000	40	$[-0.5, 10^{-4}]$	$[10, 60]$
	6D	5350000	40	$[-0.5, 10^{-4}]$	$[10, 60]$
Rastrigin	2D	23000	30	$[-0.5, 10^{-4}]$	$[30, 60]$
	3D	77000	30	$[-0.5, 10^{-4}]$	$[10, 60]$
	4D	512000	30	$[-0.5, 10^{-4}]$	$[1, 10]$
	5D	2090000	30	$[-0.5, 10^{-4}]$	$[1, 10]$
	6D	7200000	30	$[-0.5, 10^{-4}]$	$[0.5, 1]$
Schwefel	2D	21000	100	$[-50, 10^{-4}]$	$[0.1, 1]$
	3D	79000	100	$[-50, 10^{-4}]$	$[0.1, 1]$
	4D	470000	100	$[-87, 10^{-4}]$	$[0.1, 1]$
	5D	2470000	100	$[-100, 10^{-4}]$	$[0.1, 1]$
	6D	9312000	100	$[-125, 10^{-4}]$	$[0.05, 0.1]$
Sphere	2D	54000	10	$[-0.4, 10^{-4}]$	$[1, 89]$
	3D	274000	10	$[-0.1, 10^{-4}]$	$[1, 89]$
	4D	1332000	10	$[-0.05, 10^{-4}]$	$[1, 89]$
	5D	5400000	10	$[-0.05, 10^{-4}]$	$[1, 89]$
	6D	21340000	10	$[-0.05, 10^{-4}]$	$[1, 89]$

Table A.2: Experiments' Simulated Annealing Parameters

Function		Cooling Factor	Neighbor Distance	Temperature
Dropwave	3D	0.9999979	0.5	$[0.1, 0.01]$
Eggholder	3D	0.9999917	300	$[100, 0.01]$
Holdertable	3D	0.9999979	4	$[0.1, 0.01]$
Langermann	3D	0.9999906	4	$[1, 0.1]$
Shubert	3D	0.999983	1	$[100, 0.01]$
Easom	3D	0.999998	3	$[0.01, 10^{-3}]$
Ackley	2D	0.999968	0.5	$[10, 0.01]$
	3D	0.9999919	0.5	$[10, 0.01]$
	4D	0.99999887	0.5	$[10, 0.01]$
	5D	0.99999971	0.5	$[10, 0.01]$
	6D	0.9999999163	0.5	$[10, 0.01]$
Rastrigin	2D	0.999979	1	$[10, 0.01]$
	3D	0.9999942	1	$[10, 0.01]$
	4D	0.999983	1	$[100, 0.01]$
	5D	0.9999997	1	$[100, 0.01]$
	6D	0.999999925	1	$[100, 0.01]$
Schwefel	2D	0.99998	200	$[100, 0.01]$
	3D	0.9999928	200	$[100, 0.01]$
	4D	0.9999977	200	$[100, 0.01]$
	5D	0.999999535	200	$[10000, 0.01]$
	6D	0.99999986	200	$[10000, 0.01]$
Sphere	2D	0.999995	0.5	$[10^{-4}, 10^{-6}]$
	3D	0.9999986	0.5	$[10^{-4}, 10^{-6}]$
	4D	0.99999978	0.5	$[10^{-4}, 10^{-6}]$
	5D	0.999999535	0.5	$[10^{-4}, 10^{-6}]$
	6D	0.9999999828	0.5	$[10^{-4}, 10^{-6}]$

Table A.3: Experiments' Threshold Accepting Parameters

Function		Rounds	Neighbor Distance	Threshold
Dropwave	3D	1070000	0.5	$[0.5, 0]$
Eggholder	3D	1090000	400	$[500, 0]$
Holdertable	3D	1150000	2	$[5, 0]$
Langermann	3D	255000	4	$[1, 0]$
Shubert	3D	560000	0.5	$[100, 0]$
Easom	3D	1250000	3	$[0.1, 0]$
Ackley	2D	240000	0.5	$[1, 0]$
	3D	900000	0.5	$[1, 0]$
	4D	6200000	0.5	$[1, 0]$
	5D	24300000	0.5	$[1, 0]$
	6D	78400000	0.5	$[1, 0]$
Rastrigin	2D	400000	1	$[1, 0]$
	3D	1240000	1	$[1, 0]$
	4D	8000000	1	$[1, 0]$
	5D	30000000	1	$[1, 0]$
	6D	98500000	1	$[1, 0]$
Schwefel	2D	500000	300	$[800, 0]$
	3D	1300000	300	$[800, 0]$
	4D	7500000	300	$[800, 0]$
	5D	45000000	300	$[800, 0]$
	6D	98000000	300	$[800, 0]$
Sphere	2D	528000	0.5	$[0, 0]$
	3D	1800000	0.5	$[0, 0]$
	4D	11000000	0.5	$[0, 0]$
	5D	45000000	0.5	$[0, 0]$
	6D	137000000	0.5	$[0, 0]$

Table A.4: Experiments'Particle Swarm Optimization Parameters

Function		Rounds	w	c1	c2	Number of Particles
Dropwave	3D	12000	0.9	0.1	0.5	100
Eggholder	3D	2500	0.9	5	10	500
Holdertable	3D	12400	0.9	0.5	1	100
Langermann	3D	4300	0.9	0.5	1	100
Shubert	3D	42000	0.9	0.5	1	20
Easom	3D	62000	0.9	0.05	0.5	20
Ackley	2D	15300	0.9	0.01	0.1	20
	3D	56000	0.9	0.01	0.1	20
	4D	134100	0.9	0.01	0.1	50
	5D	268000	0.9	0.01	0.1	100
	6D	914000	0.9	0.01	0.1	100
Rastrigin	2D	3800	0.9	0.05	0.5	100
	3D	13000	0.9	0.05	0.5	100
	4D	16650	0.9	0.05	0.5	500
	5D	31600	0.9	0.05	0.5	1000
	6D	52100	0.9	0.05	0.5	2000
Schwefel	2D	2800	0.9	1	2	200
	3D	6500	0.9	1	2	200
	4D	8500	0.9	1	2	200
	5D	64200	0.9	1	2	500
	6D	56000	0.9	1	2	2000
Sphere	2D	4100	0.9	0.5	1	100
	3D	10700	0.9	0.5	1	100
	4D	86000	0.9	0.5	1	100
	5D	337500	0.9	0.5	1	100
	6D	1080000	0.9	0.5	1	100

B
Statistical Significance Tables

Here we present the statistical significance tables that were used to interpret the results of our experiments.
As mentioned in the paper, the statistical significance of all results was tested using ANOVA (analysis of variance) and follow up Tukey tests. We consider results to be statistically significant using a 0.05 p-value threshold.
BP outperforms others in terms of accuracy, many times reaching 100% accuracy, and these results are statistically significant, except results against TA for the Schwefel 3D, 4D, and 6D cases, and the Langermann case (the latter is only marginally not significant).
BP's precision is higher than that of the other two single-point algorithms considered (SA,TA) in most cases. Our tests confirm statistically significant better BP precision against SA and TA in all cases but: Rastrigin 2D against TA; Schwefel 2D, 3D, and 4D against SA; Ackley 4D against SA and TA and 6D against SA; Holdertable 3D against TA; and Dropwave 3D against TA (marginally). Furthermore, although the precision of BP seems to be marginally lower than that of PSO (which is also developed for continuous optimization tasks), our tests indicate that this difference is not statistically significant in most cases (i.e., Rastrigin, Ackley, Shubert, Schwefel, Dropwave and Langermann for all respective dimensions considered). As noted in the main paper, PSO frequently has the worst accuracy among all algorithms, and thus is a poor choice for global optimization in these cases.

Table B.1: Dropwave 3D Statistical Significance Accuracy Results

	group1	group2	Diff	Lower	Upper	q-value	p-value
0	BP	SA	0.200000	0.170830	0.229170	24.925005	0.001000
1	BP	TA	0.012857	-0.016313	0.042027	1.602322	0.649250
2	BP	PSO	0.002857	-0.026313	0.032027	0.356071	0.900000
3	SA	TA	0.187143	0.157973	0.216313	23.322683	0.001000
4	SA	PSO	0.202857	0.173687	0.232027	25.281076	0.001000
5	TA	PSO	0.015714	-0.013456	0.044885	1.958393	0.507947

Table B.2: Dropwave 3D Statistical Significance Precision Results

	group1	group2	Diff	Lower	Upper	q-value	p-value
0	BP	SA	0.009515	0.008617	0.010414	38.501871	0.001000
1	BP	TA	0.005734	0.004884	0.006583	24.533287	0.001000
2	BP	PSO	0.000588	-0.000258	0.001434	2.526069	0.280341
3	SA	TA	0.003781	0.002880	0.004682	15.256670	0.001000
4	SA	PSO	0.010103	0.009205	0.011001	40.907267	0.001000
5	TA	PSO	0.006322	0.005473	0.007171	27.068564	0.001000

Table B.3: Eggholder 3D Statistical Significance Accuracy Results

	group1	group2	Diff	Lower	Upper	q-value	p-value
0	BP	SA	0.361429	0.300022	0.422835	21.396893	0.001
1	BP	TA	0.257143	0.195736	0.318550	15.223086	0.001
2	BP	PSO	0.470000	0.408593	0.531407	27.824418	0.001
3	SA	TA	0.104286	0.042879	0.165693	6.173807	0.001
4	SA	PSO	0.108571	0.047165	0.169978	6.427525	0.001
5	TA	PSO	0.212857	0.151450	0.274264	12.601332	0.001

Table B.4: Eggholder 3D Statistical Significance Precision Results

	group1	group2	Diff	Lower	Upper	q-value	p-value
0	BP	SA	0.844217	-0.233499	1.921933	2.848672	0.182977
1	BP	TA	11.077680	10.054642	12.100719	39.377614	0.001000
2	BP	PSO	3.323224	2.166354	4.480095	10.446422	0.001000
3	SA	TA	11.921898	10.767456	13.076339	37.554855	0.001000
4	SA	PSO	2.479007	1.204458	3.753556	7.073164	0.001000
5	TA	PSO	14.400904	13.172242	15.629567	42.623558	0.001000

Table B.5: Holdertable 3D Statistical Significance Accuracy Results

	group1	group2	Diff	Lower	Upper	q-value	p-value
0	BP	SA	0.000000	-0.034023	0.034023	0.000000	0.900
1	BP	TA	0.000000	-0.034023	0.034023	0.000000	0.900
2	BP	PSO	0.428571	0.394549	0.462594	45.793013	0.001
3	SA	TA	0.000000	-0.034023	0.034023	0.000000	0.900
4	SA	PSO	0.428571	0.394549	0.462594	45.793013	0.001
5	TA	PSO	0.428571	0.394549	0.462594	45.793013	0.001

Table B.6: Holdertable 3D Statistical Significance Precision Results

	group1	group2	Diff	Lower	Upper	q-value	p-value
0	BP	SA	0.004032	0.001033	0.007031	4.887421	0.003128
1	BP	TA	0.023008	0.020009	0.026008	27.890252	0.001000
2	BP	PSO	0.008118	0.004601	0.011635	8.391827	0.001000
3	SA	TA	0.018977	0.015977	0.021976	23.002831	0.001000
4	SA	PSO	0.012150	0.008633	0.015667	12.559834	0.001000
5	TA	PSO	0.031126	0.027609	0.034643	32.176714	0.001000

Table B.7: Langermann 3D Statistical Significance Accuracy Results

	group1	group2	Diff	Lower	Upper	q-value	p-value
0	BP	SA	0.482857	0.425446	0.540269	30.574831	0.001000
1	BP	TA	0.344286	0.286874	0.401697	21.800397	0.001000
2	BP	PSO	0.030000	-0.027412	0.087412	1.899620	0.531273
3	SA	TA	0.138571	0.081160	0.195983	8.774434	0.001000
4	SA	PSO	0.452857	0.395446	0.510269	28.675211	0.001000
5	TA	PSO	0.314286	0.256874	0.371697	19.900778	0.001000

Table B.8: Langermann 3D Statistical Significance Precision Results

	group1	group2	Diff	Lower	Upper	q-value	p-value
0	BP	SA	0.011091	0.008745	0.013437	17.209450	0.001000
1	BP	TA	0.011713	0.010382	0.013043	32.050335	0.001000
2	BP	PSO	0.000602	-0.000372	0.001576	2.248737	0.385846
3	SA	TA	0.000622	-0.001898	0.003142	0.898129	0.900000
4	SA	PSO	0.011693	0.009341	0.014045	18.098133	0.001000
5	TA	PSO	0.012315	0.010974	0.013655	33.437878	0.001000

Table B.9: Shubert 3D Statistical Significance Accuracy Results

	group1	group2	Diff	Lower	Upper	q-value	p-value
0	BP	SA	0.000000	-0.025474	0.025474	0.000000	0.900
1	BP	TA	0.000000	-0.025474	0.025474	0.000000	0.900
2	BP	PSO	0.164286	0.138811	0.189760	23.444424	0.001
3	SA	TA	0.000000	-0.025474	0.025474	0.000000	0.900
4	SA	PSO	0.164286	0.138811	0.189760	23.444424	0.001
5	TA	PSO	0.164286	0.138811	0.189760	23.444424	0.001

Table B.10: Shubert 3D Statistical Significance Precision Results

	group1	group2	Diff	Lower	Upper	q-value	p-value
0	BP	SA	0.122499	0.017744	0.227254	4.251238	0.014207
1	BP	TA	1.075749	0.970994	1.180504	37.333075	0.001000
2	BP	PSO	0.036840	-0.072942	0.146623	1.219970	0.800981
3	SA	TA	0.953250	0.848495	1.058005	33.081837	0.001000
4	SA	PSO	0.159339	0.049557	0.269122	5.276519	0.001117
5	TA	PSO	1.112590	1.002807	1.222372	36.843350	0.001000

Table B.11: Easom 3D Statistical Significance Accuracy Results

	group1	group2	Diff	Lower	Upper	q-value	p-value
0	BP	SA	0.001429	-0.004917	0.007774	0.818450	0.900000
1	BP	TA	0.001429	-0.004917	0.007774	0.818450	0.900000
2	BP	PSO	0.005714	-0.000631	0.012060	3.273802	0.094786
3	SA	TA	0.000000	-0.006345	0.006345	0.000000	0.900000
4	SA	PSO	0.007143	0.000798	0.013488	4.092252	0.020050
5	TA	PSO	0.007143	0.000798	0.013488	4.092252	0.020050

Table B.12: Easom 3D Statistical Significance Precision Results

	group1	group2	Diff	Lower	Upper	q-value	p-value
0	BP	SA	0.000548	0.000262	0.000835	6.953154	0.001
1	BP	TA	0.001181	0.000894	0.001468	14.974374	0.001
2	BP	PSO	0.001889	0.001602	0.002176	23.910785	0.001
3	SA	TA	0.001729	0.001443	0.002016	21.935370	0.001
4	SA	PSO	0.001341	0.001054	0.001628	16.976140	0.001
5	TA	PSO	0.003070	0.002783	0.003357	38.872164	0.001

Table B.13: Ackley 2D Statistical Significance Accuracy Results

	group1	group2	Diff	Lower	Upper	q-value	p-value
0	BP	SA	0.0	-4.695276e-16	4.695276e-16	0.0	0.9
1	BP	TA	0.0	-4.695276e-16	4.695276e-16	0.0	0.9
2	BP	PSO	0.0	-4.695276e-16	4.695276e-16	0.0	0.9
3	SA	TA	0.0	-4.695276e-16	4.695276e-16	0.0	0.9
4	SA	PSO	0.0	-4.695276e-16	4.695276e-16	0.0	0.9
5	TA	PSO	0.0	-4.695276e-16	4.695276e-16	0.0	0.9

Table B.14: Ackley 2D Statistical Significance Precision Results

	group1	group2	Diff	Lower	Upper	q-value	p-value
0	BP	SA	0.008684	0.007075	0.010294	19.689190	0.001000
1	BP	TA	0.002000	0.000391	0.003609	4.534711	0.007892
2	BP	PSO	0.000171	-0.001438	0.001781	0.388799	0.900000
3	SA	TA	0.006684	0.005075	0.008294	15.154480	0.001000
4	SA	PSO	0.008856	0.007246	0.010465	20.077989	0.001000
5	TA	PSO	0.002172	0.000562	0.003781	4.923510	0.003101

Table B.15: Ackley 3D Statistical Significance Accuracy Results

	group1	group2	Diff	Lower	Upper	q-value	p-value
0	BP	SA	0.000000	-0.004491	0.004491	0.000000	0.900000
1	BP	TA	0.000000	-0.004491	0.004491	0.000000	0.900000
2	BP	PSO	0.004286	-0.000205	0.008777	3.469068	0.067841
3	SA	TA	0.000000	-0.004491	0.004491	0.000000	0.900000
4	SA	PSO	0.004286	-0.000205	0.008777	3.469068	0.067841
5	TA	PSO	0.004286	-0.000205	0.008777	3.469068	0.067841

Table B.16: Ackley 3D Statistical Significance Precision Results

	group1	group2	Diff	Lower	Upper	q-value	p-value
0	BP	SA	0.025463	0.023217	0.027710	41.209638	0.001
1	BP	TA	0.032040	0.029794	0.034287	51.853678	0.001
2	BP	PSO	0.000174	-0.002075	0.002423	0.281147	0.900
3	SA	TA	0.006577	0.004331	0.008823	10.644040	0.001
4	SA	PSO	0.025637	0.023389	0.027886	41.446514	0.001
5	TA	PSO	0.032214	0.029966	0.034463	52.079119	0.001

Table B.17: Ackley 4D Statistical Significance Accuracy Results

	group1	group2	Diff	Lower	Upper	q-value	p-value
0	BP	SA	0.0	-4.695276e-16	4.695276e-16	0.0	0.9
1	BP	TA	0.0	-4.695276e-16	4.695276e-16	0.0	0.9
2	BP	PSO	0.0	-4.695276e-16	4.695276e-16	0.0	0.9
3	SA	TA	0.0	-4.695276e-16	4.695276e-16	0.0	0.9
4	SA	PSO	0.0	-4.695276e-16	4.695276e-16	0.0	0.9
5	TA	PSO	0.0	-4.695276e-16	4.695276e-16	0.0	0.9

Table B.18: Ackley 4D Statistical Significance Precision Results

	group1	group2	Diff	Lower	Upper	q-value	p-value
0	BP	SA	0.030220	-0.008642	0.069081	2.837325	0.187343
1	BP	TA	0.025889	-0.012973	0.064751	2.430716	0.315367
2	BP	PSO	0.021121	-0.017740	0.059983	1.983092	0.498605
3	SA	TA	0.004331	-0.034531	0.043192	0.406610	0.900000
4	SA	PSO	0.009098	-0.029764	0.047960	0.854233	0.900000
5	TA	PSO	0.004768	-0.034094	0.043629	0.447624	0.900000

Table B.19: Ackley 5D Statistical Significance Accuracy Results

	group1	group2	Diff	Lower	Upper	q-value	p-value
0	BP	SA	0.0	-4.695276e-16	4.695276e-16	0.0	0.9
1	BP	TA	0.0	-4.695276e-16	4.695276e-16	0.0	0.9
2	BP	PSO	0.0	-4.695276e-16	4.695276e-16	0.0	0.9
3	SA	TA	0.0	-4.695276e-16	4.695276e-16	0.0	0.9
4	SA	PSO	0.0	-4.695276e-16	4.695276e-16	0.0	0.9
5	TA	PSO	0.0	-4.695276e-16	4.695276e-16	0.0	0.9

Table B.20: Ackley 5D Statistical Significance Precision Results

	group1	group2	Diff	Lower	Upper	q-value	p-value
0	BP	SA	0.043992	0.036743	0.051240	22.144300	0.001
1	BP	TA	0.069916	0.062667	0.077164	35.194038	0.001
2	BP	PSO	0.000091	-0.007157	0.007340	0.045936	0.900
3	SA	TA	0.025924	0.018676	0.033173	13.049739	0.001
4	SA	PSO	0.044083	0.036834	0.051331	22.190236	0.001
5	TA	PSO	0.070007	0.062759	0.077256	35.239974	0.001

Table B.21: Ackley 6D Statistical Significance Accuracy Results

	group1	group2	Diff	Lower	Upper	q-value	p-value
0	BP	SA	0.0	-4.695276e-16	4.695276e-16	0.0	0.9
1	BP	TA	0.0	-4.695276e-16	4.695276e-16	0.0	0.9
2	BP	PSO	0.0	-4.695276e-16	4.695276e-16	0.0	0.9
3	SA	TA	0.0	-4.695276e-16	4.695276e-16	0.0	0.9
4	SA	PSO	0.0	-4.695276e-16	4.695276e-16	0.0	0.9
5	TA	PSO	0.0	-4.695276e-16	4.695276e-16	0.0	0.9

Table B.22: Ackley 6D Statistical Significance Precision Results

	group1	group2	Diff	Lower	Upper	q-value	p-value
0	BP	SA	0.062052	-0.004846	0.128951	3.384423	0.080176
1	BP	TA	0.115042	0.048143	0.181940	6.274553	0.001000
2	BP	PSO	0.062465	-0.004433	0.129364	3.406948	0.077143
3	SA	TA	0.052990	-0.013909	0.119888	2.890130	0.174063
4	SA	PSO	0.000413	-0.066485	0.067311	0.022525	0.900000
5	TA	PSO	0.052577	-0.014322	0.119475	2.867605	0.179503

Table B.23: Rastrigin 2D Statistical Significance Accuracy Results

	group1	group2	Diff	Lower	Upper	q-value	p-value
0	BP	SA	0.0	-4.695276e-16	4.695276e-16	0.0	0.9
1	BP	TA	0.0	-4.695276e-16	4.695276e-16	0.0	0.9
2	BP	PSO	0.0	-4.695276e-16	4.695276e-16	0.0	0.9
3	SA	TA	0.0	-4.695276e-16	4.695276e-16	0.0	0.9
4	SA	PSO	0.0	-4.695276e-16	4.695276e-16	0.0	0.9
5	TA	PSO	0.0	-4.695276e-16	4.695276e-16	0.0	0.9

Table B.24: Rastrigin 2D Statistical Significance Precision Results

	group1	group2	Diff	Lower	Upper	q-value	p-value
0	BP	SA	4.768148e-03	0.003466	0.006070	13.359526	0.001000
1	BP	TA	4.781578e-04	-0.000824	0.001780	1.339715	0.753715
2	BP	PSO	7.332082e-07	-0.001302	0.001303	0.002054	0.900000
3	SA	TA	4.289991e-03	0.002988	0.005592	12.019811	0.001000
4	SA	PSO	4.768882e-03	0.003467	0.006071	13.361581	0.001000
5	TA	PSO	4.788910e-04	-0.000823	0.001781	1.341770	0.752899

Table B.25: Rastrigin 3D Statistical Significance Accuracy Results

	group1	group2	Diff	Lower	Upper	q-value	p-value
0	BP	SA	0.000000	-0.006338	0.006338	0.000000	0.900000
1	BP	TA	0.000000	-0.006338	0.006338	0.000000	0.900000
2	BP	PSO	0.008571	0.002234	0.014909	4.916595	0.002899
3	SA	TA	0.000000	-0.006338	0.006338	0.000000	0.900000
4	SA	PSO	0.008571	0.002234	0.014909	4.916595	0.002899
5	TA	PSO	0.008571	0.002234	0.014909	4.916595	0.002899

Table B.26: Rastrigin 3D Statistical Significance Precision Results

	group1	group2	Diff	Lower	Upper	q-value	p-value
0	BP	SA	0.021672	0.018049	0.025296	21.741986	0.001
1	BP	TA	0.038716	0.035092	0.042340	38.840617	0.001
2	BP	PSO	0.000418	-0.003214	0.004049	0.418187	0.900
3	SA	TA	0.017044	0.013420	0.020667	17.098632	0.001
4	SA	PSO	0.022090	0.018458	0.025721	22.113331	0.001
5	TA	PSO	0.039134	0.035502	0.042765	39.175126	0.001

Table B.27: Rastrigin 4D Statistical Significance Accuracy Results

	group1	group2	Diff	Lower	Upper	q-value	p-value
0	BP	SA	0.00	-0.018244	0.018244	0.0	0.900000
1	BP	TA	0.00	-0.018244	0.018244	0.0	0.900000
2	BP	PSO	0.01	-0.008244	0.028244	2.0	0.491457
3	SA	TA	0.00	-0.018244	0.018244	0.0	0.900000
4	SA	PSO	0.01	-0.008244	0.028244	2.0	0.491457
5	TA	PSO	0.01	-0.008244	0.028244	2.0	0.491457

Table B.28: Rastrigin 4D Statistical Significance Precision Results

	group1	group2	Diff	Lower	Upper	q-value	p-value
0	BP	SA	3.546609e-02	0.028149	0.042783	17.685494	0.001000
1	BP	TA	4.448206e-02	0.037165	0.051799	22.181389	0.001000
2	BP	PSO	6.492275e-07	-0.007335	0.007336	0.000323	0.900000
3	SA	TA	9.015966e-03	0.001699	0.016333	4.495895	0.008632
4	SA	PSO	3.546674e-02	0.028131	0.042802	17.641325	0.001000
5	TA	PSO	4.448270e-02	0.037147	0.051818	22.125910	0.001000

Table B.29: Rastrigin 5D Statistical Significance Accuracy Results

	group1	group2	Diff	Lower	Upper	q-value	p-value
0	BP	SA	0.00	-0.070443	0.070443	0.000000	0.900
1	BP	TA	0.00	-0.070443	0.070443	0.000000	0.900
2	BP	PSO	0.18	0.109557	0.250443	9.323456	0.001
3	SA	TA	0.00	-0.070443	0.070443	0.000000	0.900
4	SA	PSO	0.18	0.109557	0.250443	9.323456	0.001
5	TA	PSO	0.18	0.109557	0.250443	9.323456	0.001

Table B.30: Rastrigin 5D Statistical Significance Precision Results

	group1	group2	Diff	Lower	Upper	q-value	p-value
0	BP	SA	1.156989e-01	0.098643	0.132755	24.756064	0.00100
1	BP	TA	1.386910e-01	0.121635	0.155747	29.675672	0.00100
2	BP	PSO	6.810272e-07	-0.017967	0.017968	0.000138	0.90000
3	SA	TA	2.299207e-02	0.005936	0.040048	4.919607	0.00315
4	SA	PSO	1.156996e-01	0.097732	0.133667	23.500137	0.00100
5	TA	PSO	1.386917e-01	0.120724	0.156659	28.170135	0.00100

Table B.31: Rastrigin 6D Statistical Significance Accuracy Results

	group1	group2	Diff	Lower	Upper	q-value	p-value
0	BP	SA	0.00	-0.091384	0.091384	0.000000	0.900
1	BP	TA	0.00	-0.091384	0.091384	0.000000	0.900
2	BP	PSO	0.54	0.448616	0.631384	21.560834	0.001
3	SA	TA	0.00	-0.091384	0.091384	0.000000	0.900
4	SA	PSO	0.54	0.448616	0.631384	21.560834	0.001
5	TA	PSO	0.54	0.448616	0.631384	21.560834	0.001

Table B.32: Rastrigin 6D Statistical Significance Precision Results

	group1	group2	Diff	Lower	Upper	q-value	p-value
0	BP	SA	3.308128e-01	0.287498	0.374127	27.885891	0.001000
1	BP	TA	3.603789e-01	0.317064	0.403693	30.378169	0.001000
2	BP	PSO	6.807842e-07	-0.054565	0.054566	0.000046	0.900000
3	SA	TA	2.956611e-02	-0.013748	0.072881	2.492278	0.293632
4	SA	PSO	3.308135e-01	0.276248	0.385379	22.136192	0.001000
5	TA	PSO	3.603796e-01	0.305814	0.414945	24.114591	0.001000

Table B.33: Schwefel 2D Statistical Significance Accuracy Results

	group1	group2	Diff	Lower	Upper	q-value	p-value
0	BP	SA	0.05	-0.006514	0.106514	3.228166	0.103828
1	BP	TA	0.00	-0.056514	0.056514	0.000000	0.900000
2	BP	PSO	0.05	-0.006514	0.106514	3.228166	0.103828
3	SA	TA	0.05	-0.006514	0.106514	3.228166	0.103828
4	SA	PSO	0.00	-0.056514	0.056514	0.000000	0.900000
5	TA	PSO	0.05	-0.006514	0.106514	3.228166	0.103828

Table B.34: Schwefel 2D Statistical Significance Precision Results

	group1	group2	Diff	Lower	Upper	q-value	p-value
0	BP	SA	0.005724	-0.023612	0.035060	0.712001	0.900
1	BP	TA	0.104921	0.075964	0.133878	13.221992	0.001
2	BP	PSO	0.000191	-0.029145	0.029527	0.023742	0.900
3	SA	TA	0.099197	0.069862	0.128533	12.339378	0.001
4	SA	PSO	0.005915	-0.023795	0.035624	0.726488	0.900
5	TA	PSO	0.105112	0.075776	0.134448	13.075121	0.001

Table B.35: Schwefel 3D Statistical Significance Accuracy Results

	group1	group2	Diff	Lower	Upper	q-value	p-value
0	BP	SA	0.257143	0.203971	0.310315	17.580669	0.001
1	BP	TA	0.175714	0.122542	0.228886	12.013457	0.001
2	BP	PSO	0.440000	0.386828	0.493172	30.082478	0.001
3	SA	TA	0.081429	0.028256	0.134601	5.567212	0.001
4	SA	PSO	0.182857	0.129685	0.236029	12.501809	0.001
5	TA	PSO	0.264286	0.211114	0.317458	18.069021	0.001

Table B.36: Schwefel 3D Statistical Significance Precision Results

	group1	group2	Diff	Lower	Upper	q-value	p-value
0	BP	SA	0.094264	-0.379072	0.567600	0.724101	0.900
1	BP	TA	4.843520	4.383930	5.303110	38.318878	0.001
2	BP	PSO	0.010816	-0.505527	0.527159	0.076167	0.900
3	SA	TA	4.749256	4.254734	5.243778	34.919017	0.001
4	SA	PSO	0.083448	-0.464220	0.631115	0.554013	0.900
5	TA	PSO	4.832704	4.296872	5.368535	32.793210	0.001

Table B.37: Schwefel 4D Statistical Significance Accuracy Results

	group1	group2	Diff	Lower	Upper	q-value	p-value
0	BP	SA	0.19	0.078439	0.301561	6.214168	0.001000
1	BP	TA	0.09	-0.021561	0.201561	2.943553	0.161086
2	BP	PSO	0.16	0.048439	0.271561	5.232984	0.001401
3	SA	TA	0.10	-0.011561	0.211561	3.270615	0.096804
4	SA	PSO	0.03	-0.081561	0.141561	0.981184	0.895828
5	TA	PSO	0.07	-0.041561	0.181561	2.289430	0.369920

Table B.38: Schwefel 4D Statistical Significance Precision Results

	group1	group2	Diff	Lower	Upper	q-value	p-value
0	BP	SA	0.174557	-0.314800	0.663913	1.302226	0.768607
1	BP	TA	3.993026	3.518757	4.467294	30.736395	0.001000
2	BP	PSO	0.000310	-0.484194	0.484814	0.002335	0.900000
3	SA	TA	3.818469	3.318401	4.318538	27.876296	0.001000
4	SA	PSO	0.174867	-0.334920	0.684653	1.252258	0.788408
5	TA	PSO	3.993336	3.498014	4.488657	29.432282	0.001000

Table B.39: Schwefel 5D Statistical Significance Accuracy Results

	group1	group2	Diff	Lower	Upper	q-value	p-value
0	BP	SA	0.28	0.143083	0.416917	7.461811	0.001000
1	BP	TA	0.14	0.003083	0.276917	3.730905	0.042876
2	BP	PSO	0.38	0.243083	0.516917	10.126743	0.001000
3	SA	TA	0.14	0.003083	0.276917	3.730905	0.042876
4	SA	PSO	0.10	-0.036917	0.236917	2.664932	0.236512
5	TA	PSO	0.24	0.103083	0.376917	6.395838	0.001000

Table B.40: Schwefel 5D Statistical Significance Precision Results

	group1	group2	Diff	Lower	Upper	q-value	p-value
0	BP	SA	1.214328	0.004306	2.424349	3.665695	0.048815
1	BP	TA	10.103092	8.951755	11.254429	32.052767	0.001000
2	BP	PSO	0.000307	-1.265177	1.265791	0.000886	0.900000
3	SA	TA	8.888765	7.638194	10.139335	25.962517	0.001000
4	SA	PSO	1.214635	-0.141758	2.571027	3.270949	0.097199
5	TA	PSO	10.103399	8.799089	11.407710	28.294380	0.001000

Table B.41: Schwefel 6D Statistical Significance Accuracy Results

	group1	group2	Diff	Lower	Upper	q-value	p-value
0	BP	SA	0.18	0.02432	0.33568	4.218738	0.015965
1	BP	TA	0.12	-0.03568	0.27568	2.812492	0.193958
2	BP	PSO	0.29	0.13432	0.44568	6.796856	0.001000
3	SA	TA	0.06	-0.09568	0.21568	1.406246	0.727340
4	SA	PSO	0.11	-0.04568	0.26568	2.578118	0.264119
5	TA	PSO	0.17	0.01432	0.32568	3.984364	0.026057

Table B.42: Schwefel 6D Statistical Significance Precision Results

	group1	group2	Diff	Lower	Upper	q-value	p-value
0	BP	SA	3.781060	1.464498	6.097622	5.964393	0.001
1	BP	TA	21.717131	19.451330	23.982932	35.024933	0.001
2	BP	PSO	0.000002	-2.431813	2.431817	0.000003	0.900
3	SA	TA	17.936071	15.540747	20.331396	27.362737	0.001
4	SA	PSO	3.781058	1.228128	6.333988	5.412165	0.001
5	TA	PSO	21.717129	19.210170	24.224089	31.655681	0.001

Table B.43: Sphere 2D Statistical Significance Accuracy Results

	group1	group2	Diff	Lower	Upper	q-value	p-value
0	BP	SA	0.0	-4.695276e-16	4.695276e-16	0.0	0.9
1	BP	TA	0.0	-4.695276e-16	4.695276e-16	0.0	0.9
2	BP	PSO	0.0	-4.695276e-16	4.695276e-16	0.0	0.9
3	SA	TA	0.0	-4.695276e-16	4.695276e-16	0.0	0.9
4	SA	PSO	0.0	-4.695276e-16	4.695276e-16	0.0	0.9
5	TA	PSO	0.0	-4.695276e-16	4.695276e-16	0.0	0.9

Table B.44: Sphere 2D Statistical Significance Precision Results

	group1	group2	Diff	Lower	Upper	q-value	p-value
0	BP	SA	3.595145e-02	0.034106	0.037797	7.107533e+01	0.001
1	BP	TA	3.595685e-02	0.034111	0.037802	7.108599e+01	0.001
2	BP	PSO	3.595685e-02	0.034111	0.037802	7.108599e+01	0.001
3	SA	TA	5.395104e-06	-0.001840	0.001851	1.066601e-02	0.900
4	SA	PSO	5.395106e-06	-0.001840	0.001851	1.066602e-02	0.900
5	TA	PSO	1.910873e-12	-0.001846	0.001846	3.777759e-09	0.900

Table B.45: Sphere 3D Statistical Significance Accuracy Results

	group1	group2	Diff	Lower	Upper	q-value	p-value
0	BP	SA	0.0	-6.684743e-16	6.684743e-16	0.0	0.9
1	BP	TA	0.0	-6.684743e-16	6.684743e-16	0.0	0.9
2	BP	PSO	0.0	-6.684743e-16	6.684743e-16	0.0	0.9
3	SA	TA	0.0	-6.684743e-16	6.684743e-16	0.0	0.9
4	SA	PSO	0.0	-6.684743e-16	6.684743e-16	0.0	0.9
5	TA	PSO	0.0	-6.684743e-16	6.684743e-16	0.0	0.9

Table B.46: Sphere 3D Statistical Significance Precision Results

	group1	group2	Diff	Lower	Upper	q-value	p-value
0	BP	SA	0.013151	-0.001467	0.027768	3.270561	0.095304
1	BP	TA	0.013152	-0.001465	0.027769	3.270880	0.095253
2	BP	PSO	0.012779	-0.001839	0.027396	3.178036	0.111083
3	SA	TA	0.000001	-0.014616	0.014619	0.000318	0.900000
4	SA	PSO	0.000372	-0.014245	0.014989	0.092525	0.900000
5	TA	PSO	0.000373	-0.014244	0.014991	0.092844	0.900000

Table B.47: Sphere 4D Statistical Significance Accuracy Results

	group1	group2	Diff	Lower	Upper	q-value	p-value
0	BP	SA	0.0	-4.695276e-16	4.695276e-16	0.0	0.9
1	BP	TA	0.0	-4.695276e-16	4.695276e-16	0.0	0.9
2	BP	PSO	0.0	-4.695276e-16	4.695276e-16	0.0	0.9
3	SA	TA	0.0	-4.695276e-16	4.695276e-16	0.0	0.9
4	SA	PSO	0.0	-4.695276e-16	4.695276e-16	0.0	0.9
5	TA	PSO	0.0	-4.695276e-16	4.695276e-16	0.0	0.9

Table B.48: Sphere 4D Statistical Significance Precision Results

	group1	group2	Diff	Lower	Upper	q-value	p-value
0	BP	SA	0.000737	0.000660	0.000814	34.952150	0.001000
1	BP	TA	0.000751	0.000674	0.000828	35.608279	0.001000
2	BP	PSO	0.000760	0.000683	0.000836	36.016033	0.001000
3	SA	TA	0.000014	-0.000063	0.000091	0.656129	0.900000
4	SA	PSO	0.000022	-0.000055	0.000099	1.063884	0.863049
5	TA	PSO	0.000009	-0.000068	0.000086	0.407754	0.900000

Table B.49: Sphere 5D Statistical Significance Accuracy Results

	group1	group2	Diff	Lower	Upper	q-value	p-value
0	BP	SA	0.0	-4.695276e-16	4.695276e-16	0.0	0.9
1	BP	TA	0.0	-4.695276e-16	4.695276e-16	0.0	0.9
2	BP	PSO	0.0	-4.695276e-16	4.695276e-16	0.0	0.9
3	SA	TA	0.0	-4.695276e-16	4.695276e-16	0.0	0.9
4	SA	PSO	0.0	-4.695276e-16	4.695276e-16	0.0	0.9
5	TA	PSO	0.0	-4.695276e-16	4.695276e-16	0.0	0.9

Table B.50: Sphere 5D Statistical Significance Precision Results

	group1	group2	Diff	Lower	Upper	q-value	p-value
0	BP	SA	0.000606	0.000551	0.000662	39.981321	0.001000
1	BP	TA	0.000644	0.000589	0.000700	42.465009	0.001000
2	BP	PSO	0.000706	0.000651	0.000761	46.550744	0.001000
3	SA	TA	0.000038	-0.000018	0.000093	2.483687	0.296413
4	SA	PSO	0.000100	0.000044	0.000155	6.569423	0.001000
5	TA	PSO	0.000062	0.000007	0.000117	4.085736	0.021155

Table B.51: Sphere 6D Statistical Significance Accuracy Results

	group1	group2	Diff	Lower	Upper	q-value	p-value
0	BP	SA	0.0	-4.695276e-16	4.695276e-16	0.0	0.9
1	BP	TA	0.0	-4.695276e-16	4.695276e-16	0.0	0.9
2	BP	PSO	0.0	-4.695276e-16	4.695276e-16	0.0	0.9
3	SA	TA	0.0	-4.695276e-16	4.695276e-16	0.0	0.9
4	SA	PSO	0.0	-4.695276e-16	4.695276e-16	0.0	0.9
5	TA	PSO	0.0	-4.695276e-16	4.695276e-16	0.0	0.9

Table B.52: Sphere 6D Statistical Significance Precision Results

	group1	group2	Diff	Lower	Upper	q-value	p-value
0	BP	SA	0.000440	0.000382	0.000499	27.448808	0.001
1	BP	TA	0.000455	0.000396	0.000513	28.348265	0.001
2	BP	PSO	0.000704	0.000645	0.000762	43.868313	0.001
3	SA	TA	0.000014	-0.000044	0.000073	0.899457	0.900
4	SA	PSO	0.000263	0.000205	0.000322	16.419505	0.001
5	TA	PSO	0.000249	0.000190	0.000307	15.520048	0.001

References

[1] Abbasi, B., Niaki, S. T. A., Khalife, M. A., & Faize, Y. (2011). A hybrid variable neighborhood search and simulated annealing algorithm to estimate the three parameters of the weibull distribution. *Expert Systems with Applications*, 38(1), 700–708.

[2] Ali, M. M., Khompatraporn, C., & Zabinsky, Z. B. (2005). A numerical evaluation of several stochastic algorithms on selected continuous global optimization test problems. *Journal of global optimization*, 31(4), 635–672.

[3] Asokan, P., Saravanan, R., & Vijayakumar, K. (2003). Machining parameters optimisation for turning cylindrical stock into a continuous finished profile using genetic algorithm (ga) and simulated annealing (sa). *The International Journal of Advanced Manufacturing Technology*, 21(1), 1–9.

[4] Bertsimas, D., Tsitsiklis, J., et al. (1993). Simulated annealing. *Statistical science*, 8(1), 10–15.

[5] Biehl, M. & Schwarze, H. (1995). Learning by on-line gradient descent. *Journal of Physics A: Mathematical and general*, 28(3), 643.

[6] Chambolle, A. & Pock, T. (2016). An introduction to continuous optimization for imaging. *Acta Numerica*, 25, 161–319.

[7] Cheh, K. M., Goldberg, J. B., & Askin, R. G. (1991). A note on the effect of neighborhood structure in simulated annealing. *Computers & Operations Research*, 18(6), 537–547.

[8] Connolly, D. T. (1990). An improved annealing scheme for the qap. *European Journal of Operational Research*, 46(1), 93–100.

[9] Deng, Z. & Tian, T. (2014). A continuous optimization approach for inferring parameters in mathematical models of regulatory networks. *BMC bioinformatics*, 15(1), 1–12.

[10] Dhouib, S., Kharrat, A., & Chabchoub, H. (2010). A multi-start threshold accepting algorithm for multiple objective continuous optimization problems. *International journal for numerical methods in engineering*, 83(11), 1498–1517.

[11] Dréo, J., Pétrowski, A., Siarry, P., & Taillard, E. (2006). *Metaheuristics for hard optimization: methods and case studies*. Springer Science & Business Media.

[12] Duchi, J., Hazan, E., & Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7).

[13] Dueck, G. & Scheuer, T. (1990). Threshold accepting: A general purpose optimization algorithm appearing superior to simulated annealing. *Journal of computational physics*, 90(1), 161–175.

[14] Eberhart, R. C. & Hu, X. (1999). Human tremor analysis using particle swarm optimization. In *Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406)*, volume 3 (pp. 1927–1930).: IEEE.

[15] Golden, B. L. & Skiscim, C. C. (1986). Using simulated annealing to solve routing and location problems. *Naval Research Logistics Quarterly*, 33(2), 261–279.

[16] Goldstein, L. & Waterman, M. (1988). Neighborhood size in the simulated annealing algorithm. *American Journal of Mathematical and Management Sciences*, 8(3-4), 409–423.

[17] Grass, J. & Zilberstein, S. (1996). Anytime algorithm development tools. *ACM SIGART Bulletin*, 7(2), 20–27.

[18] Gupta, D., Tilwalia, R., & Jain, A. (2020). Optimization of electricity consumption using evolutionary algorithms. *Available at SSRN 3565796*.

[19] Hochreiter, S., Younger, A. S., & Conwell, P. R. (2001). Learning to learn using gradient descent. In *International Conference on Artificial Neural Networks* (pp. 87–94).: Springer.

[20] Jeyakumar, V. & Rubinov, A. M. (2006). *Continuous Optimization: Current Trends and Modern Applications*, volume 99. Springer Science & Business Media.

[21] Johnson, D. S., Aragon, C. R., McGeoch, L. A., & Schevon, C. (1989). Optimization by simulated annealing: An experimental evaluation; part i, graph partitioning. *Operations research*, 37(6), 865–892.

[22] Jünger, M., Reinelt, G., & Rinaldi, G. (1995). The traveling salesman problem. *Handbooks in operations research and management science*, 7, 225–330.

[23] Kennedy, J. & Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of ICNN'95-international conference on neural networks*, volume 4 (pp. 1942–1948).: IEEE.

[24] Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *science*, 220(4598), 671–680.

[25] Koopmans, T. C. & Beckmann, M. (1957). Assignment problems and the location of economic activities. *Econometrica: journal of the Econometric Society*, (pp. 53–76).

[26] Kulkarni, M. & Babu, A. S. (2005). Managing quality in continuous casting process using product quality model and simulated annealing. *Journal of Materials Processing Technology*, 166(2), 294–306.

[27] Lundy, M. & Mees, A. (1986). Convergence of an annealing algorithm. *Mathematical programming*, 34(1), 111–124.

[28] Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., & Teller, E. (1953). Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6), 1087–1092.

[29] Munoz, M. A., Kirley, M., & Halgamuge, S. K. (2013). The algorithm selection problem on the continuous optimization domain. In *Computational intelligence in intelligent data analysis* (pp. 75–89). Springer.

[30] Nikolaev, A. G. & Jacobson, S. H. (2010). Simulated annealing. In *Handbook of metaheuristics* (pp. 1–39). Springer.

[31] Poli, R., Kennedy, J., & Blackwell, T. (2007). Particle swarm optimization. *Swarm intelligence*, 1(1), 33–57.

[32] Potts, C. & Van Wassenhove, L. N. (1991). Single machine tardiness sequencing heuristics. *IIE transactions*, 23(4), 346–354.

[33] Qian, N. (1999). On the momentum term in gradient descent learning algorithms. *Neural networks*, 12(1), 145–151.

[34] Russell, S. J., Norvig, P., Canny, J., Malik, J., & Edwards, D. (1995). Iterative improvement algorithms. *Artificial Intelligence: A Modern Approach*, (pp. 111–114).

[35] Sekihara, K., Haneishi, H., & Ohyama, N. (1992). Details of simulated annealing algorithm to estimate parameters of multiple current dipoles using biomagnetic data. *IEEE transactions on medical imaging*, 11(2), 293–299.

[36] Shalev-Shwartz, S. & Ben-David, S. (2014). *Understanding machine learning: From theory to algorithms*. Cambridge university press.

[37] Shi, Y. & Eberhart, R. (1998). A modified particle swarm optimizer. In *1998 IEEE international conference on evolutionary computation proceedings. IEEE world congress on computational intelligence (Cat. No. 98TH8360)* (pp. 69–73).: IEEE.

[38] Siddique, N. & Adeli, H. (2016). Simulated annealing, its variants and engineering applications. *International Journal on Artificial Intelligence Tools*, 25(06), 1630001.

[39] Socha, K. & Dorigo, M. (2008). Ant colony optimization for continuous domains. *European journal of operational research*, 185(3), 1155–1173.

[40] Soltanolkotabi, M. (2017). Learning relus via gradient descent. *arXiv preprint arXiv:1705.04591*.

[41] Taylan, P., Weber, G.-W., & Yerlikaya, F. (2008). Continuous optimization applied in mars for modern applications in finance, science and technology. In *ISI Proceedings of 20th mini-EURO conference continuous optimization and knowledge-based technologies* (pp. 317–322).: Citeseer.

[42] Törn, A. & Zilinskas, A. (1989). *Global optimization*. Springer.

[43] Tütüncü, R. H. (2003). *Optimization in finance*. Citeseer.

[44] Van Laarhoven, P. J., Aarts, E. H., & Lenstra, J. K. (1992). Job shop scheduling by simulated annealing. *Operations research*, 40(1), 113–125.

[45] Vanderbilt, D. & Louie, S. G. (1984). A monte carlo simulated annealing approach to optimization over continuous variables. *Journal of computational physics*, 56(2), 259–271.

[46] Ventresca, M. & Tizhoosh, H. R. (2007). Simulated annealing with opposite neighbors. In *2007 IEEE Symposium on Foundations of Computational Intelligence* (pp. 186–192).: IEEE.

[47] Weber, G.-W. (2009). Continuous optimization in finance.

[48] Weber, G.-W., Özöğür-Akyüz, S., & Kropat, E. (2009). A review on data mining and continuous optimization applications in computational biology and medicine. *Birth Defects Research Part C: Embryo Today: Reviews*, 87(2), 165–181.

[49] Xiong, Q. & Jutan, A. (2003). Continuous optimization using a dynamic simplex method. *Chemical Engineering Science*, 58(16), 3817–3828.

[50] Yao, X. (1991). Simulated annealing with extended neighbourhood. *International journal of computer mathematics*, 40(3-4), 169–189.

[51] Zamora, J. M. & Grossmann, I. E. (1998). Continuous global optimization of structured process systems models. *Computers & chemical engineering*, 22(12), 1749–1770.

[52] Zhou, A.-H., Zhu, L.-P., Hu, B., Deng, S., Song, Y., Qiu, H., & Pan, S. (2019). Traveling-salesman-problem algorithm based on simulated annealing and gene-expression programming. *Information*, 10(1), 7.

[53] Zibulevsky, M. & Elad, M. (2010). L1-l2 optimization in signal and image processing. *IEEE Signal Processing Magazine*, 27(3), 76–88.