

OPTIMIZATION OF BENCHMARK FUNCTIONS USING CHEMICAL REACTION

OPTIMIZATION

A Paper
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By

Naveen Kumar Dandu

In Partial Fulfillment
for the Degree of
MASTER OF SCIENCE

Major Department:
Computer Science

April 2013

Fargo, North Dakota

North Dakota State University
Graduate School

Title

OPTIMIZATION OF BENCHMARK FUNCTIONS USING CHEMICAL
REACTION OPTIMIZATION

By

Naveen Kumar Dandu

The Supervisory Committee certifies that this *disquisition* complies with North Dakota
State University's regulations and meets the accepted standards for the degree of

MASTER OF SCIENCE

SUPERVISORY COMMITTEE:

Dr. Simone Ludwig

Chair

Dr. Kendall Nygard

Dr. Svetlana Kilina

Approved:

05/3/2013

Date

Dr. Brian M. Slator

Department Chair

ABSTRACT

In recent past, many new nature-inspired optimization techniques have been emerged in the field of science and engineering that have proven to be efficient to solve optimization problems. One such method that makes use of metaheuristics for optimization, inspired by the nature of chemical reactions is Chemical Reaction Optimization (CRO). It is a powerful metaheuristic which mimics the interactions of molecules in chemical reactions to search for the global optimum.

In this paper, we used this technique to solve well-known separable and non-separable benchmark functions. The simulation results showed that all problems showed good performance. A measurement metric is used reporting the best, average, worst, success rate and execution time. Different experiments were conducted by scaling the iterations, population size and energy level.

ACKNOWLEDGMENTS

I would like to thank my advisor, Dr. Simone Ludwig for her mentoring, thoughtful ideas, and continuous support.

I would also like to thank Dr. Kendall Nygard, for his constant support and valuable suggestions throughout my academic career.

I would like to take this opportunity to thank my external mentor Dr. Svetlana Kilina, for her invaluable support.

Last but never least, my special thanks to my family members, friends, and everyone who encouraged, and supported me in the process of completion of my degree.

TABLE OF CONTENTS

ABSTRACT.....	iii
ACKNOWLEDGMENTS	iv
LIST OF TABLES.....	vii
LIST OF FIGURES	viii
1. INTRODUCTION	1
2. LITERATURE REVIEW	4
2.1. Genetic Algorithm	4
2.2. Memetic Algorithm.....	5
2.3. Ant Colony Optimization.....	6
2.4. No-Free-Lunch Theorem	6
2.5. Chemical Reaction Optimization.....	7
3. CHEMICAL REACTION OPTIMIZATION APPROACH	10
3.1. Major Components of the Chemical Reaction Optimization Design	10
3.1.1. Molecules.....	10
3.1.2. Elementary Reactions	11
3.1.3. Energy Handling	16
3.1.4. CRO Algorithm.....	17
4. SIMULATION RESULTS AND DISCUSSIONS.....	23
4.1. Benchmark Functions	23
4.1.1. Unimodal Functions.....	23
4.1.2. High-Dimensional Multimodal Functions	23

4.1.3.	Low-Dimensional Multimodal Functions.....	23
4.2.	Simulation Experiments.....	27
4.2.1.	Parameter Setup and Evaluation Metric	27
4.2.2.	Test Case 1: Simulations Performed using 25 Iterations.....	29
4.2.3.	Test Case 2: Simulations Performed using 25 Iterations by Varying the Dimension of the Functions	30
4.2.4.	Test Case 3: Simulations Performed with Dim = 10 by Varying the Number of Iterations	36
4.2.5.	Test Case 4: Simulations Performed with Dim = 10 and Number of Iterations = 100000; by Varying the Population Size of the Problem	42
4.2.6.	Test Case 5: Simulations Performed with Dim = 10 and Number of Iterations = 100000; by Varying the StepSize of the Problem	44
4.2.7.	Test Case 6: Simulations Performed with Dim = 10 and Number of Iterations = 100000; by Varying the EnergyBuffer Value of the Problem....	46
5.	CONCLUSION AND FUTURE WORK	48
6.	REFERENCES	49

LIST OF TABLES

<u>Table</u>	<u>Page</u>
1. Molecules profile in CRO.....	11
2. Definitions used in CRO.....	12
3. Unimodal functions.....	24
4. High-dimensional multimodal functions	25
5. Low-dimensional multimodal functions.....	26
6. Experimental results showing the results metrics.....	29
7. Best results for functions using 25 iterations varying the dimension	31
8. Average results for functions using 25 iterations varying the dimension.....	33
9. Worst results for functions using 25 iterations varying the dimension	35
10. Best results for functions ran with dim = 10 varying number of iterations	37
11. Average results for functions with 10 dim varying the number of iterations	39
12. Worst results for functions with 10 dim varying the number of iterations.....	41
13. Varying the population size for f1 and evaluating best objective value	43
14. Varying the step size for f1 and evaluating best objective value.....	45
15. Varying the energy buffer for f1 and evaluating best objective value.....	47

LIST OF FIGURES

<u>Figure</u>	<u>Page</u>
1. Potential energy surface [29]	8
2. Molecules in a container	11
3. On-wall ineffective collision [33].....	13
4. Decomposition reaction [33]	15
5. Inter-molecular ineffective collision [33].....	15
6. Synthesis [33]	16
7. Flow chart representing the CRO design.....	22
8. Best results for functions using 25 iterations varying the dimension	32
9. Average results for functions using 25 iterations varying the dimension.....	34
10. Worst results for functions using 25 iterations varying the dimension	36
11. Best results for functions with 10 dim varying the number of iterations	38
12. Average results for functions with 10 dim varying the number of iterations	40
13. Worst results for functions with 10 dim varying the number of iterations.....	42
14. Varying the population size for f1 and evaluating best objective value	44
15. StepSize tuning for f1 and evaluating best objective function	46
16. Energy Buffer tuning for f1 and evaluating best objective function	47

1. INTRODUCTION

Optimization is a process or a technique that is applied to find the best solution from all other feasible solutions. The problem that utilizes this process is known as an optimization problem. It exists in almost all fields of science and engineering ranging from finding an optimized chemical structure in chemical sciences [1] to power generation scheduling in electrical engineering [2]. A simple example of such kind is to find a shortest path to reach from one place to another, with some conditions like avoiding toll gates, minimizing the cost, etc. The goal is to find a better algorithm, which can be applied to solve such problems to obtain an optimal solution.

Generally to perform optimization on such problems, we define several components such as objective functions, variables and constraints. Depending on whether the variables are continuous or discrete, the optimization problems are characterized into two categories: (1) continuous, and (2) combinatorial/discrete or sometimes also known as Nondeterministic Polynomial-time hard (NP-hard) problems [14]. Some examples of real world optimization problems include:

1. Minimizing the transportation cost by varying the locations of facilities [3]: This is a combinatorial optimization problem. The task is to minimize the transportation cost by varying the locations of facilities. If we have n locations and n facilities, we would like minimize the total cost by arranging the locations of the facilities.
2. Resource-constrained project scheduling problem [4]: This is the most intractable, NP-hard problem which needs an optimized solution to allocate a time schedule of the events related to project management.
3. Channel assignment problem in wireless mesh networks [5]: This problem is a NP-

- hard and combinatorial optimization problem that needs the minimization of the induced interference subject to interface constraints, because more channels cannot be assigned to a router than the number of interfaces equipped.
4. Population Transition Problem in peer-to-peer live streaming [6]: The Population Transition Problem maximizes the probability of universal streaming by assigning population transition probabilities among all colonies. The solution required is to provide sufficient streaming data to all peers.
 5. Cognitive radio spectrum allocation problem [7]: Restricted to an interference-free environment, the spectrum allocation problem assigns channels to users in order to maximize system utility subject to hardware constraint.
 6. Grid scheduling problem [8]: schedules tasks to resources as to minimize computational overheads and to utilize the resources effectively.
 7. Standard continuous benchmark function optimization [9]: the standard benchmarks to be optimized comprised of unimodal, high-dimensional and low-dimensional multimodal functions.
 8. Stock portfolio selection problem [10]: It is a multi-objective mixed integer NP-hard problem. The task is to build a portfolio of stocks with the consideration of the two contradicting objectives.
 9. Network coding optimization problem [11]: It is an NP-hard combinatorial problem. The task is to minimize number of coding links while maintaining a certain transmission rate.
 10. Artificial neural network training [12]: The task is to model the input–output relationships of complex systems.

Suppose we have an objective function ‘f’, a set of variables ‘X’ that define the dimensionality of problem from x_1 to x_n and total number of constraints ‘C’ from c_1 to c_m , then we can define a solution space ‘S’ as the set of all possible solutions of the problem. Our goal is to find the minima S^* that belongs to S.

Thus, we can write a generalized equation for optimization as follows:

$$\underbrace{\min}_{X \in R^n} f(X) \text{ is defined to be } \begin{cases} c_i(X) = 0, & i \in E \\ c_i(X) \leq 0, & i \in I \end{cases}$$

This generic form can be used to solve many problems. Since many optimization problems are NP-hard problems [4] there does not exist a known algorithm that can solve and find the optimal solution. For such problems, computational effort/time grows exponentially with the size of the problem, and there is no guarantee that the obtained results are of high quality. Thus, to tackle these problems, we always approximate the algorithm to compute solutions that are categorized as “good” rather than the “best”.

One of the many ways in finding a solution to this problem was found from nature. Nature, as we all know is a complex system, involving all the activities associated with the living beings. This so called complex system balances the environment constantly. Mimicking these methods to solve problems is called nature-inspired computing [15]. The mimicking phenomena from nature have led to the development of many algorithms to solve the problems, more specifically those for which we have limited knowledge to design an effective solving technique.

The arrangement of the chapters is as follows: Chapter 2 briefly introduces the background about the Chemical Reaction Optimization (CRO) technique. Chapter 3 describes the design approach of CRO. Chapter 4 explains the simulation results and discussions. Chapter 5 addresses conclusions drawn and discusses future work.

2. LITERATURE REVIEW

The field of nature-inspired optimization techniques has been emerging extensively since past few decades. These algorithms are generally known to be evolutionary algorithms. The main idea of evolutionary computing was first introduced in the 1960s by I. Rechenberg in his work “Evolution strategies” (originally known to be as Evolutionsstrategie). This idea headed to the invention of many algorithms by other researchers, such as Genetic Algorithm (GA) [17], Ant Colony Optimization (ACO) [18], Particle Swarm Optimization (PSO) [19], Memetic Algorithm [20] (MA), and many more.

2.1. Genetic Algorithm

John Holland was the first to invent Genetic Algorithms (GAs) in 1975. These are adaptive stochastic optimization algorithms developed from the knowledge of natural evolutionary selection. The main idea behind GAs is to simulate processes by mimicking the natural selection system, particularly those that follow the Charles Darwin’s survival of the fittest principle. In brief, it solves a problem by exploiting a random search within a definite search space in two steps. First step is a mutation step, which is done by mutating or arbitrarily changing a given group of sample programs. The next step is to select through measuring against a fitness function. These two steps are repeated until an appropriate solution is established.

Many well-known real world problems were solved successfully using GAs. Listed below are a few fields where GAs are applied:

1. State Assignment Problem [21]: This State Assignment Problem (SAP) belongs to a broader class of combinatorial optimization problems, including the Traveling Salesman Problem (TSP).

2. Game Theory [22]: GA is applied in game theory to find equilibrium points in non-zero sum and non-cooperative situations, and in the game of the iterated prisoner's dilemma to explore the possibility of evolving cooperative behavior.
3. Scheduling [23]: GA is used for the inspection and repair of oil tanks and pipelines. A good inspection schedule for oil installations is constructed. A good schedule ensures that repair times are kept to a minimum and faults are found before they become too serious. An automatic way of assigning maintenance activities to inspectors is devised in such a way as to minimize the loss in capacity, while keeping within resource constraints

Although GAs performs better even in complex, real-world situations, several developments must be made in order to make GAs applicable to all systems.

2.2. Memetic Algorithm

Memetic Algorithm (MA) was first introduced by Moscato in the year 1989. He got the inspiration from both Darwinian natural evolution principles and Dawkins' meme notion. The main motivation is that Memes propagate from brain to brain via imitation and that can be allowed for improvements unlike GA (almost static or unchanged throughout generations). It allows one to introduce new heuristics to search problems for finding the optimal solution, by learning.

The application areas of MA are vast, however, there are several that can be highlighted such as the traditional NP optimization problems [24], whereby the Classical NP problems such as Graph Partitioning, Min Number Partitioning, Max Independent Set, Bin-Packing, Min Graph Coloring are optimized. Furthermore, MAs are applied in Machine Learning and Robotics [25], Electronics and Engineering [26], and Molecular Optimization Problems [27].

The more the intensity of individual learning, the greater is the chance to attain local optima. Although it improves the final results, the cost of computational time increases. To balance the available budget for performing computations, and to achieve better search performance, care should be taken in setting these two factors. Also, care must be taken in deciding which meme or memes are used for a given optimization problem as it might favor another neighborhood structure. There might be an issue of utilizing the MA search to maximum, when only a portion of individual populations undergo learning.

2.3. Ant Colony Optimization

Ant Colony Optimization (ACO) is another population-based algorithm that mimics the behavior of ants in searching their food. The paths that lead to food trails denote solutions. These ants, when discovering the path, lay down a chemical called pheromone along their paths such that the other ants get aware about the food trails and follow the same path. The shorter the distance, the more is the pheromone that is produced as more ants travel around that place. These pheromones also evaporate easily aiding to avoid the local optima. The best solution is the one that selects the route which has more pheromone. ACO is more specific to routing based problems. Application areas include scheduling problem [28], vehicle routing problem [29], and quadratic assignment problem [30].

2.4. No-Free-Lunch Theorem

All these algorithms are successful in solving different kinds of optimization problems. According to No-Free-Lunch theorem [31], when averaged over all the involved objective functions, these search algorithms behave exactly the same in terms of performance. This means,

some algorithms work excellent for certain classes of problems, while they are outperformed for other classes of problems. Another two terms we often come across when dealing with the optimization algorithms are heuristics and metaheuristics. Heuristics target to a specific problem and behave poor when applied to other class of problems. But, metaheuristics on the other hand, can be applied to a broader range of problems and shows better performance in terms of results. Compared to metaheuristic algorithm, heuristic is a better option when we are dealing with a specific problem. If the heuristic is not readily available, then metaheuristic may be used to solve that problem.

The heuristics and metaheuristics can be related as accuracy-flexibility tradeoff. A no polynomial-time algorithm can be first solved using a metaheuristic algorithm, if it works well then we can add heuristics and greedy algorithms to make it perform better. But this hybrid algorithm may not be suitable for another class of algorithms, making it more like a heuristic algorithm. This is one of the ways where flexibility is sacrificed for accuracy.

For a broad range of problems, the number of successfully reported meta-heuristics is much less compared to the number of problems.

2.5. Chemical Reaction Optimization

In effort to accomplish this goal, Lam and Li [16] proposed a new meta-heuristic algorithm, called as Chemical Reaction Optimization (CRO), inspired by the nature of chemical reactions. Within a short span of time, CRO has become very popular in solving many problems successfully, overtaking many existing meta-heuristic nature-inspired evolutionary algorithms.

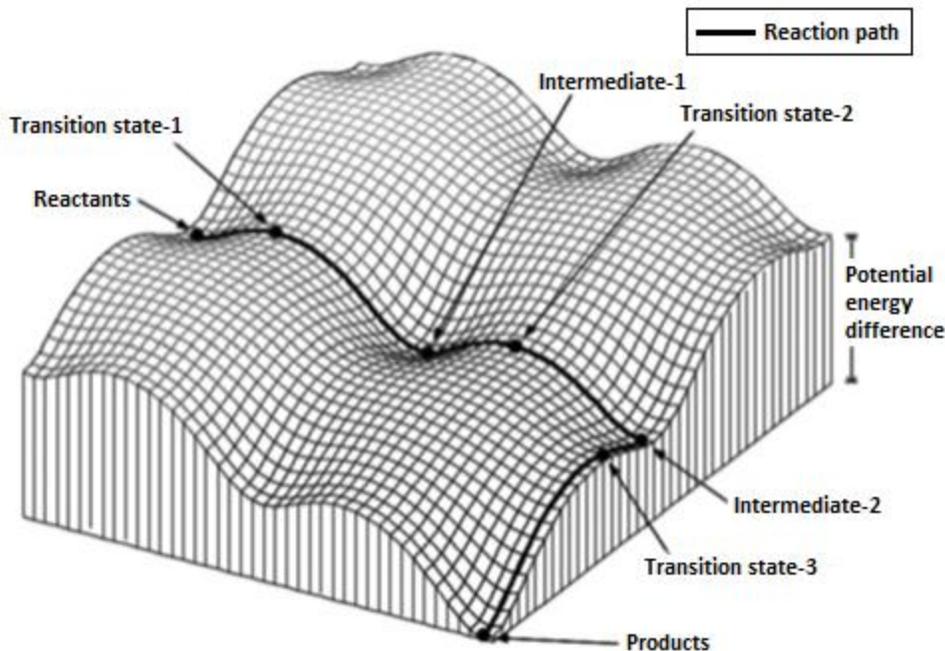


Figure 1: Potential energy surface [29]

Figure 1 represents a potential energy surface model describing the chemical reaction path and the interactions between the molecules/chemicals involved. It describes the changes of potential energy (PE) of the molecules in the chemical system. In this figure, the x-axis and y-axis represents every possible structure of the molecules in their reaction pathway in proceeding to the products, while the z-axis represents the PE.

Depending on the complexity of chemical system, the PES can be a two or more than two-dimensional. A chemical reaction occurs when the initial molecules (called reactants) undergo a bond formation or destruction to form products. These products are formed via a series of intermediate species and transition states, and each step involved in this process is called an elementary reaction.

In Figure 1, a chemical reaction is depicted showing that reactants form products via a solid line that represents the reaction pathway involving three elementary steps, several transition

states and intermediate species. As we can observe in the Figure 1, reactants are always higher in energy than products, implying that reactants release energy and form products that are lesser in energy. The lower the energy of the species, the higher is its stability, showing that the system always tends to reach minima. Each of the intermediate species can be thought to reach local minima and once the product is formed, the system is said to attain global minima.

This system can be compared to our optimization problem in that both the systems try to find the global minima and the process takes place in a step-wise manner. With this innovation, Lam and Li developed a chemical-reaction inspired algorithm for solving challenging optimization problems, by mimicking the molecules behavior in a reaction system/container.

3. CHEMICAL REACTION OPTIMIZATION APPROACH

3.1. Major Components of the Chemical Reaction Optimization Design

There are two major components of the CRO design, which are molecules and elementary reactions.

3.1.1. Molecules

A molecule is made up of two or more atoms. It is generally categorized by its atoms types, bond length, bond angle, dihedral angle and twist. These characteristics make one molecule distinguishable from another, as each molecule has its own set of characteristics. In general, energy is required for bond formation and energy is released for bond breakage. This energy is exchanged in and out from the surroundings, proving the 1st law of thermodynamics [32] stating that energy can neither be created nor destroyed. The following are three properties/attributes that describe a molecule in the CRO:

3.1.1.1. Molecular Structure

The term molecular structure is similar to what we know as a solution in the mathematical domain. In CRO, the molecular structure is denoted by ω , the solution currently held by a molecule. Molecules with minute changes are still considered to belong to the same class. There is no specific requirement on the configuration of ω . Depending on the problem; it can be in the form of a number, a vector, a matrix, or even a graph.

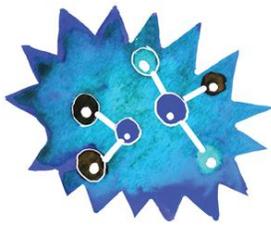
3.1.1.2. Potential Energy (PE)

It represents the energy corresponding to the structure of a molecule, and in CRO it depicts the value of the objective function of the current molecular structure ω , in finding a solution to the problem, i.e., $PE_{\omega} = f(\omega)$.

3.1.1.3. Kinetic Energy (KE)

It represents the level of tolerance for the molecule to change to a less favorable structure, i.e., with higher PE than the existing one.

Table 1: Molecules profile in CRO

Molecule	Terminology	
	Chemical Sciences	Mathematical
	Molecular Structure	Solution
	Potential Energy	Objective function value
	Kinetic Energy	Measuring tolerance of worse solutions
	Number of hits	Current total number of moves
	currentOptimal structure	Current optimal solution
	current optimal value	Current optimal function value
	Minimum hit number	Move number at current optimal solution

3.1.2. Elementary Reactions

Consider that there is a pool of molecules in a container as shown in Figure 2. Those molecules tend to collide with each other or with the walls of the container. These collisions trigger the change in the molecule, i.e., initiates the chemical reaction, making possible a product formation.

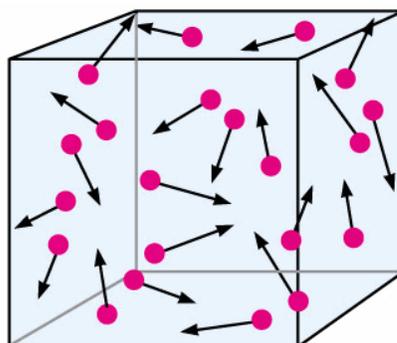


Figure 2: Molecules in a container

Table 2: Definitions used in CRO

Type	Symbol	Algorithm Meaning	Chemical Meaning
Function	F	Objective function	Function defining PES
	Neighbor	Neighbor candidate generator	Neighborhood structure on PES
Variable	Pop	Set of solution; 2-D matrix where each row carries the values of a solution	Set of molecules
	Step	Next neighborhood search	Next neighborhood structure
	PE	Vector of objective function values; $PE = f(\text{Pop})$	Potential energy of all the molecules
	KE	Vector of number measuring the tolerance of the solutions to have worse objective function values afterward	Kinetic energy of all the molecules
	Dim	Number of variables representing a solution; dimensions of the Problem	Total number of characteristic of molecule
Initial Parameter	PopSize	Initial number of solutions maintained; number of rows in Pop	Initial number of molecules in the container
	StepSize	Distance between two neighbors	Distance between two neighboring structures
	KELossRate	Percentage upper limit of reduction of KE in on-wall ineffective Collisions	Percentage upper limit of KE lost to the environment in on-wall ineffective collisions
	MoleColl	Fraction of all elementary reactions corresponding to intermolecular Reactions	Fraction of all elementary reactions corresponding to intermolecular Reactions
	InitialKE	Initial value assigned to each element of KE in the initialization stage	KE of the initial set molecules

Depending on the type of collision, these reactions are divided into four kinds of elementary reactions. They are:

- On-wall ineffective collision;
- Decomposition;
- Intermolecular ineffective collision; and
- Synthesis.

We search the solution space through a sequence of these four elementary reactions. Each of them has their own characteristic features and these features are detailed below:

3.1.2.1. On-Wall Ineffective Collision

This type of collision is considered as a unimolecular collision where a molecule hits the wall of the container and then bounces back as shown in Figure 3. In this case, the molecular structure ω and PE of the molecule that hit the wall undergoes just subtle changes. We can suppose that here the molecule tries to change ω to ω' in the neighborhood of ω , that is $\omega' = N(\omega)$.

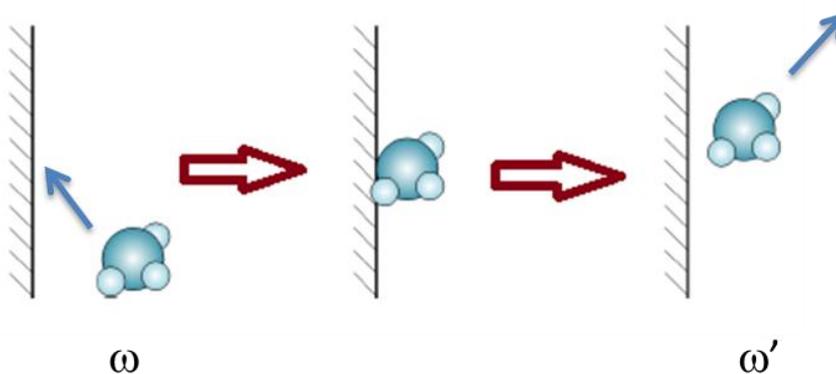


Figure 3: On-wall ineffective collision [33]

As a rule of thumb, when a molecule hits a wall, a portion of its KE will be lost, the lost energy is stored in the central energy buffer (the initial energy buffer size of the container), when the reaction completes.

Its KE is updated by:

$$KE_{\omega'} = (PE_{\omega} - PE_{\omega'} + KE_{\omega}) \times a$$

Where 'a' is a random number that lies in between [KELossRate,1], where KELossRate is a parameter of CRO.

If there is high KE processed by the molecule, then there is a possibility that the PE could be increased depicting the worst solution. This change is desirable to make the algorithm to escape from its local minimum. The KE drops as an effect of collision, its level of tolerance in getting a worst solution is lowered, and thus its escape from local minima decreases.

3.1.2.2. Decomposition

Similar to that of on-wall collision, decomposition is also a unimolecular reaction, when the molecule ω hits a wall of the container; it breaks up into two molecules ω_1' and ω_2' as shown in Figure 4. Unlike on-wall ineffective collision, the decomposition reaction is more vigorous and the resultant product molecular structures have greater transformations from that of the reactant molecule. This reaction occurs when the local search around ω is finished and that now we have decided to search other regions corresponding to ω_1' and ω_2' . Energy is transformed from the buffer to ω , such that it will have enough energy to sustain its change to form ω_1' and ω_2' .

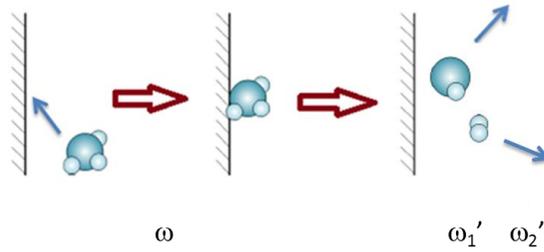


Figure 4: Decomposition reaction [33]

3.1.2.3. Intermolecular Ineffective Collision

This reaction occurs when two molecules collide with each other and bounce back as shown in Figure 5. After collision, the number of molecules remains unchanged. It is similar to that of on-wall collision in that these reactions are also not vigorous. The molecular structures of the reactants ω_1 and ω_2 undergo only subtle changes to form molecules with the molecular structures ω_1' and ω_2' . In this type of collision also, the products form from their own neighborhoods separately, that is:

$$\omega_1' = N(\omega_1) \text{ and } \omega_2' = N(\omega_2)$$

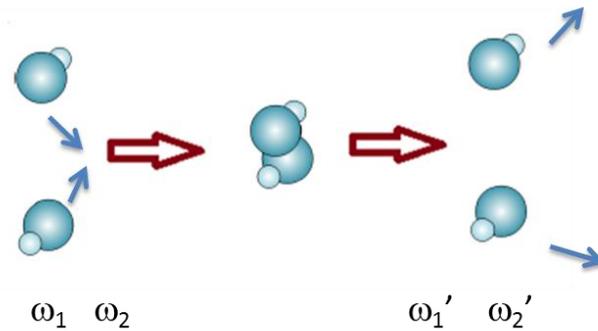


Figure 5: Inter-molecular ineffective collision [33]

3.1.2.4. Synthesis

In this type of reactions, when two molecules collide with each other, they combine to form one new single molecule as shown Figure 6. Unlike in the case of on-wall collision, synthesis reactants involve a vigorous change in their molecular structures. In other words, we give up the search regions in the neighborhoods of ω_1 and ω_2 ; start a new search again in a different territory of solution space S.

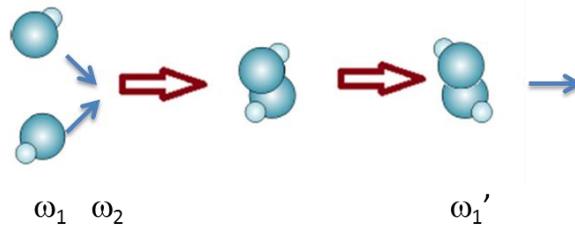
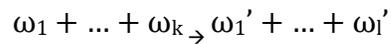


Figure 6: Synthesis [33]

3.1.3. Energy Handling

Energy can be transformed from one type to another type but all energy manipulations must follow the first law of thermodynamics that states that the energy can neither be created nor destroyed.

A generalized form of elementary reaction can be written as follows:



Where k and l are the numbers of molecules involved before and after the reaction, respectively. For k=1 and l=2, the reaction can be said to be a decomposition reaction.

The corresponding energy equation can be written as follows:

$$\begin{aligned} & (PE_{\omega_1} + \dots + PE_{\omega_k}) + (KE_{\omega_1} + \dots + KE_{\omega_k}) + \text{buffer} \\ & = (PE_{\omega_1'} + \dots + PE_{\omega_l'}) + (KE_{\omega_1'} + \dots + KE_{\omega_l'}) + \text{buffer}' \end{aligned}$$

In the above equation, the left hand side of the equality represents the molecules total energy before the change, and the right hand side of the equality represents the molecules total energy after the change.

According to the first law of thermodynamics, the total energy of the molecules before and after the reaction is identical as shown in the above equation. We can determine if the existing molecules with molecular structures $\omega_1, \dots, \omega_k$ are replaced by the new molecules with ω by examining the general new solution acceptance rule as follows:

$$\sum_{\omega} PE_{\omega} + \sum_{\omega} KE_{\omega} - \sum_{\omega'} PE_{\omega'} \geq 0$$

The first two terms in the equation above represents existing molecules total energy and the third term represents the new molecules potential energy.

The total energy of the whole system (PE and KE of the molecules and buffer) is the same for any instance. This total amount is determined by the initial PE of the initial set of molecules, the initial KE (initialKE) assigned to the initial set of molecules, and the initial buffer. This summation affects the convergence speed and the possibility of getting the global minimum.

3.1.4. CRO Algorithm

Imagine that we have a certain number of molecules in a container. They collide, interact, combine, decompose, and finally become stable at the end. The whole process is what we try to mimic with CRO. The flow chart of CRO can be found in Figure 7. It consists of three stages:

- Initialization,
- Iterations, and
- The final stage.

3.1.4.1. Initialization

In initialization, the initial settings for the molecules and the parameters (i.e., PopSize, KELossRate, MoleColl, buffer, InitialKE, α , and β) are configured. We create the initial molecule set with size equal to PopSize by randomly generating solutions in the solution space. Their initial PEs are determined by their corresponding objective function values while their initial KEs are set to InitialKE.

3.1.4.2. Iterations

In each iteration, only one elementary reaction takes place.

3.1.4.2.1. Conditions to check if the reaction is uni- or inter-molecular collision:

We first determine whether it is a uni-molecular or intermolecular reaction by comparing a random number $b \in [0,1]$ against MoleColl. . If $b > \text{MoleColl}$ or there is only one molecule left, we will have a uni-molecular reaction. Otherwise, an intermolecular reaction happens.

3.1.4.2.2. Pseudo-code for uni-molecular collision:

We have $PE = f(x)$

If $PE_{\omega} + KE_{\omega} \geq PE_{\omega'}$,

Then,

“ACCEPT”

$$KE_{\omega'} = (PE_{\omega} + KE_{\omega} - PE_{\omega'}) \times \delta_{\text{loss}}$$

$0 \leq \delta \leq 1$ is pre-defined and lost KE is stored in a central buffer

Else

“REJECT”

Nothing changed

3.1.4.2.3. Pseudo-code for inter-molecular collision:

If $PE_{\omega 1} + PE_{\omega 2} + KE_{\omega 1} + KE_{\omega 2} \geq PE_{\omega 1'} + PE_{\omega 2'}$,

Then,

“ACCEPT”

$temp = (PE_{\omega 1} + PE_{\omega 2} + KE_{\omega 1} + KE_{\omega 2}) - (PE_{x1'} + PE_{x2'})$

$KE_{x1'} = temp \times n$

$KE_{x2'} = temp \times (1-n)$

where $n \in [0,1]$

Else

“REJECT”

Nothing changed

3.1.4.2.4. Criteria for decomposition reaction:

For each unimolecular reaction, we randomly choose one molecule and check if it satisfies the decomposition criterion, i.e., (number of hits – minimum hit number) > α , where α can be interpreted as the tolerance of duration for the molecule without obtaining any new local minimum solution. If so, the molecule will experience decomposition, else it will take an on-wall ineffective collision.

3.1.4.2.5. Pseudo-code for decomposition:

If the selected molecule has stayed in a stable state for a certain period of time, i. e., number of hits – minimum hit number > α ,

Where α is the tolerance of duration for the molecule without obtaining any new local minimum solution

Then, generate two (or more) molecules

If the total energy of the original molecule is not enough to support the change, additional energy can be drawn from the central buffer

else,

Nothing changed

3.1.4.2.6. Criteria for synthesis reaction:

For each intermolecular reaction, two (or more) molecules are selected and they are tested against the synthesis criterion: $(KE \leq \beta)$, where β can be considered as the minimum KE a molecule should have. If it is satisfied by all the selected molecules, they combine through synthesis. Otherwise, they experience an intermolecular ineffective collision. We can see that molecules with little $KE \leq \beta$ always get stuck in local minima and synthesis transforms the inactive molecules into an active one.

3.1.4.2.7. Pseudo-code for synthesis:

If both of the selected molecules have kinetic energy $\leq \beta$,

Where β is the minimum KE a molecule should have;

Generate a new molecule

If $PE_{\omega_1} + PE_{\omega_2} + KE_{\omega_1} + KE_{\omega_2} \geq PE_{\omega'}$,

Then

“ACCEPT”

$KE_{\omega'} = PE_{\omega_1} + PE_{\omega_2} + KE_{\omega_1} + KE_{\omega_2} - PE_{\omega'}$

Else

“REJECT”

Nothing changed

3.1.4.2.8. Stopping criteria:

After the end of each iteration, energy check is performed. The potential energy value of the newly obtained molecule(s) is compared with the original molecules. If the new value(s) can satisfy the energy conservation conditions, the new molecules are accepted and the previous molecules are discarded. Else, the new molecules are discarded and this implies that the reaction I failed at this stage. After the pre-defined total number of function evaluations is reached, the algorithm ends and outputs the so far best value as the global optimum.

We modified the already existing CRO code [16] such that we can record the measurement metrics for the generated results to better understand the performance of the algorithm, while varying the number of iterations, dimensions and other internal parameters used in the CRO code.

The measurement metrics helped us to calculate the estimated run time (ERT), success rate (SR) and the best, average and worst objective values in the output of each function, before it reaches the global minima. All these formulations and the associated results are explained in the next subsection.

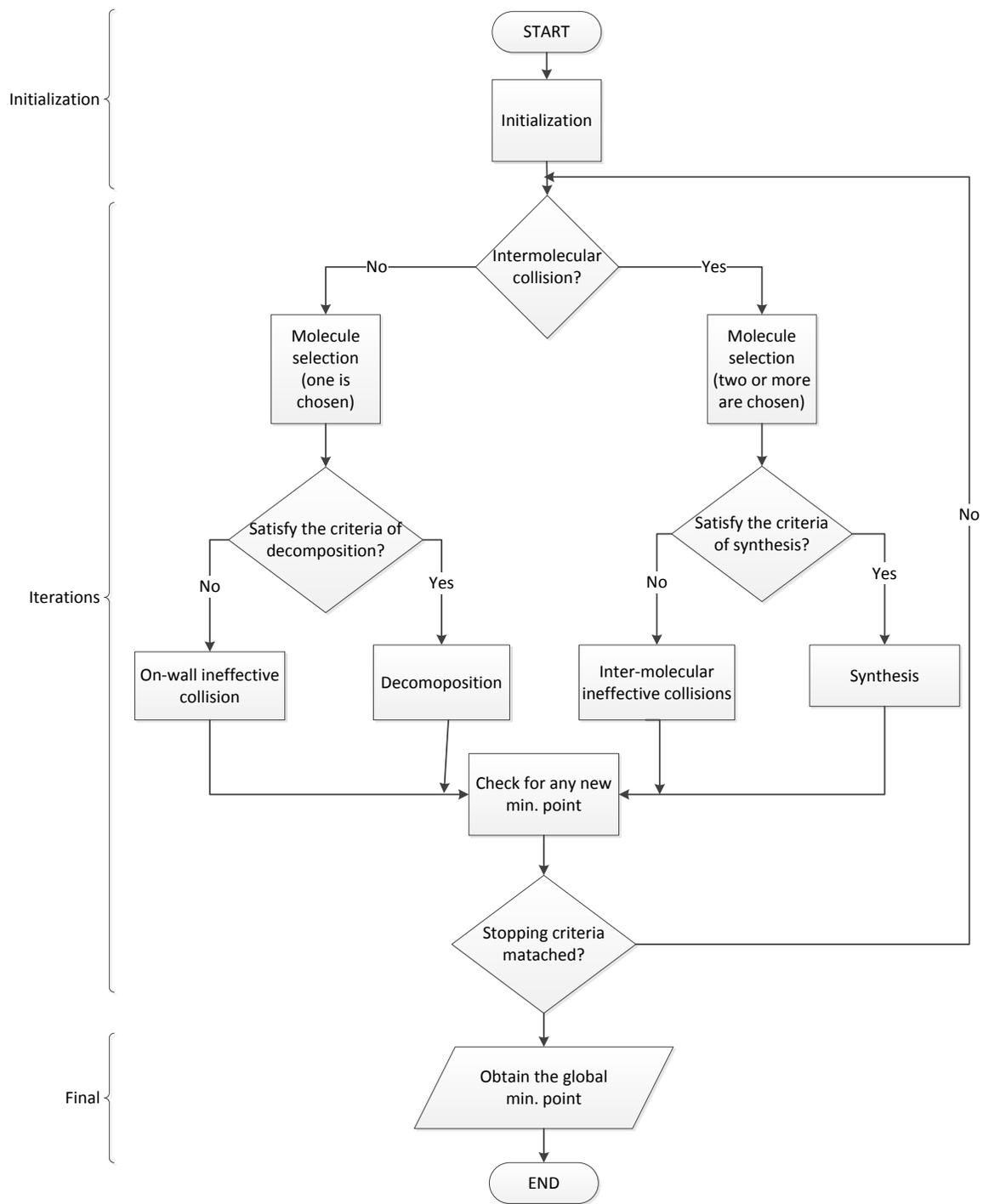


Figure 7: Flow chart representing the CRO design

4. SIMULATION RESULTS AND DISCUSSIONS

4.1. Benchmark Functions

In order to conduct a comprehensive evaluation of the CRO algorithm on problems in the continuous domain, we test the performance of CRO with a large set of standard benchmark functions [34]. These benchmark functions are listed in Table 1-3, which contains their dimension sizes, their feasible solution space S , and the objective function values.

There are 23 benchmark functions in total, classified into three categories according to their characteristics:

4.1.1. Unimodal Functions

This group consists of functions f_1 – f_7 and they are high-dimensional. There is only one global minimum in each of the functions. They are relatively “easy” to solve when compared with those in the next group.

4.1.2. High-Dimensional Multimodal Functions

This group is composed of functions f_8 – f_{13} . They are high-dimensional and contain many local minima. They are considered as the most difficult problems in the benchmark set.

4.1.3. Low-Dimensional Multimodal Functions

This group includes functions f_{14} – f_{23} . They have lower dimensions and have fewer local minima than the previous group.

Table 3: Unimodal functions

Test Function	Name	S
$f_1(\mathbf{x}) = \sum_{i=1}^n x_i^2$	Sphere	$[-100, 100]^n$
$f_2(\mathbf{x}) = \sum_{i=1}^n x_i + \prod_{i=1}^n x_i $	Schwefel's problem 2.22	$[-10, 10]^n$
$f_3(\mathbf{x}) = \sum_{i=1}^n \left(\sum_{j=1}^i x_j \right)^2$	Schwefel's problem 1.2	$[-100, 100]^n$
$f_4(\mathbf{x}) = \max_i \{ x_i , 1 \leq i \leq n\}$	Schwefel's problem 2.21	$[-100, 100]^n$
$f_5(\mathbf{x}) = \sum_{i=1}^{n-1} (100(x_{i+1} - x_i)^2 + (x_i - 1)^2)$	Generalized Rosenbrock's function	$[-30, 30]^n$
$f_6(\mathbf{x}) = \sum_{i=1}^n (x_i + 0.5)^2$	Step function	$[-100, 100]^n$
$f_7(\mathbf{x}) = \sum_{i=1}^n ix_i^4 + \text{random}[0, 1)$	Quartic function with noise	$[-1.28, 1.28]^n$

Table 4: High-dimensional multimodal functions

Test Function	Name	S
$f_8(x) = \sum_{i=1}^n (x_i \sin(\sqrt{ x_i }))$	Generalized Schwefel's problem 2.26	$[-500, 500]^n$
$f_9(x) = \sum_{i=1}^n (x_i^2 - 10 \cos(2\pi x_i) + 10)$	Generalized Rastrigin's function	$[-5.12, 5.12]^n$
$f_{10}(x) = -20 \exp\left(-0.2 \sqrt{\frac{1}{2} \sum_{i=1}^n x_i^2}\right) - \exp\left(\frac{1}{n} \sum_{i=1}^n \cos 2\pi x_i\right) + 20 + e$	Ackley's function	$[-32, 32]^n$
$f_{11}(x) = \frac{1}{4000} \sum_{i=1}^n x_i^2 - \prod_{i=1}^n \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	Generalized Griewank function	$[-600, 600]^n$
$f_{12}(x) = \frac{\pi}{n} \{10 \sin^2(\pi y_1) + \sum_{i=1}^{29} (y_i - 1)^2 [1 + 10 \sin^2(\pi y_{i+1})] + (y_n - 1)^2\} + \sum_{i=1}^{30} u(x_i, 10, 100, 4)$ $y_i = 1 + \frac{1}{4}(x_i + 1)$ $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a \\ 0, & -a \leq x_i \leq a \\ k(-x_i - a)^m, & x_i < -a \end{cases}$	Generalized penalized function	$[-50, 50]^n$
$f_{13}(x) = 0.1 \left\{ \sin^2(\pi 3x_1) + \sum_{i=1}^{29} (x_i - 1)^2 [1 + \sin^2(3\pi x_{i+1})] + (x_n - 1)^2 [1 + \sin^2(2\pi x_{30})] \right\} + \sum_{i=1}^{30} u(x_i, 5, 100, 4)$	Generalized penalized function	$[-50, 50]^n$

Table 5: Low-dimensional multimodal functions

Test Function	Name	S
$f_{14}(x) = \left[\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^2 (x_i - a_{ij})^6} \right]^{-1}$	Shekel's Foxholes function	$[-65.536, 65.536]^n$
$f_{15}(x) = \sum_{i=1}^{11} \left[a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4} \right]^2$	Kowalik's function	$[-5, 5]^n$
$f_{16}(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	Six-hump camel back function	$[-5, 5]^n$
$f_{17}(x) = (x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6)^2 + 10 \left(1 - \frac{1}{8\pi} \right) \cos x_1 + 10$	Branin function	$[-5, 10] \times [0, 15]^n$
$f_{18}(x) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \times [30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$	Goldstein-Price function	$[-2, 2]^n$
$f_{19}(x) = - \sum_{i=1}^4 c_i \exp[- \sum_{j=1}^4 a_{ij}(x_j - p_{ij})^2]$	Hartman's family	$[0, 1]^n$
$f_{20}(x) = - \sum_{i=1}^4 c_i \exp[- \sum_{j=1}^6 a_{ij}(x_j - p_{ij})^2]$	Hartman's family	$[0, 10]^n$
$f_{21}(x) = - \sum_{i=1}^5 [(x - a_i)(x - a_i)^T + c_i]^{-1}$	Shekel's family	$[0, 10]^n$
$f_{22}(x) = - \sum_{i=1}^7 [(x - a_i)(x - a_i)^T + c_i]^{-1}$	Shekel's family	$[0, 10]^n$
$f_{23}(x) = - \sum_{i=1}^{10} [(x - a_i)(x - a_i)^T + c_i]^{-1}$	Shekel's family	$[0, 10]^n$

4.2. Simulation Experiments

The simulations were run involving various test cases to evaluate the performance of the CRO algorithm by solving the benchmark functions detailed in the previous section. All the properties of the molecules are kept constant in the following test cases unless mentioned. All measurement points given are average of 30 independent runs in order to guarantee normal distribution.

4.2.1. Parameter Setup and Evaluation Metric

Initial parameters defined for the CRO algorithm and functions f1-f7:

- Popsiz = 10;
- Stepsiz = 0.1;
- CollRate = 0.2;
- EnergyBuffer = 0;
- InitialKE = 1000;
- KElossrate = 0.1;
- DecompostionThreshold = 150000;
- Synthesisthrashold = 10.

Initial parameters defined for the CRO algorithm and functions f8-f12:

- Popsiz = 100;
- Stepsiz = 0.5;
- CollRate = 0.2;
- EnergyBuffer = 0;
- InitialKE = 1000;

- KElossrate = 0.1;
- DecompostionThreshold = 500;
- Synthesisthrashold = 10.

Initial parameters defined for the CRO algorithm and functions f13-f23:

- Popsiz = 20;
- Stepsize = 1.0;
- CollRate = 0.2;
- EnergyBuffer = 0;
- InitialKE = 10000000;
- KElossrate = 0.1;
- DecompostionThreshold = 150000;
- Synthesisthrashold = 10.

Furthermore, the following values have been set for:

- **F8:** Stepsize = 300;
- **F11:** Stepsize = 15.

The Expected Running Time (ERT) [13] has been used to evaluate the CRO algorithm for each benchmark function with respect to “Values To Reach” (VTR). The following three VTR values that are used to calculate the performance criteria are:

1. VTR1 = $1e^{-2}$
2. VTR2 = $1e^{-6}$
3. VTR3 = $1e^{-10}$

The Function Evaluations to Success (FES) were calculated according to the definition: “the number of function evaluations spent in one specific successful run until such a candidate

solution was discovered”. The value of \overline{FES} is then calculated by taking the arithmetic mean over all FESs of all successful runs according to the objective function and VTR.

$$Success\ rate(SR) = \frac{\#\ of\ successful\ runs}{\#\ of\ total\ runs}$$

$$ERT = \frac{(SR + \overline{FES}) + ((1 - SR) * Max_FES)}{SR}$$

4.2.2. Test Case 1: Simulations Performed using 25 Iterations

The simulations were performed using 25 iterations. Readings are tabulated below in Table 1 for all 23 functions keeping the dimension (dim) of the problems as 30 and with the same initial parameters mentioned in Section 1.2.1 as constant. Function F6 was ran successfully in all VTR values, in other words success rate was 100% in that function. Maximum functions showed good success rate as well as the better performance in the case of VTR1.

Table 6: Experimental results showing the results metrics

Test Fn.	VTR1 = 1E-02		VTR2 = 1E-06		VTR3 = 1E-10		Best	Average	Worst
	ERT1	SR1	ERT2	SR2	ERT3	SR3			
F1	1.50E+05	1	1.50E+05	0.96	1.80E+308	0	3.05E-07	6.19E-07	1.08E-06
F2	1.50E+05	1	1.80E+308	0	1.80E+308	0	1.57E-03	2.16E-03	2.77E-03
F3	2.50E+05	1	2.50E+05	1	1.80E+308	0	5.43E-08	2.38E-07	4.27E-07
F4	1.73E+05	0.68	1.80E+308	0	1.80E+308	0	5.33E-03	8.98E-03	1.58E-02
F5	1.79E+308	0	1.80E+308	0	1.80E+308	0	1.77E+00	7.55E+01	4.78E+02
F6	1.50E+05	1	1.50E+05	1	1.50E+05	1	0.00E+00	0.00E+00	0.00E+00
F7	1.50E+05	1	1.80E+308	0	1.80E+308	0	2.00E-03	3.74E-03	5.93E-03
F8	1.50E+05	1	1.50E+05	1	1.50E+05	1	1.25E+04	1.24E+04	1.23E+04

(continued)

Table 6: Experimental results showing the results metrics (continued)

Test Fn.	VTR1 = 1E-02		VTR2 = 1E-06		VTR3 = 1E-10		Best	Average	Worst
	ERT1	SR1	ERT2	SR2	ERT3	SR3			
F9	2.50E+05	1	1.80E+308	0	1.80E+308	0	1.20E-03	1.82E-03	3.20E-03
F10	1.50E+05	1	1.80E+308	0	1.80E+308	0	1.68E-03	2.21E-03	3.00E-03
F11	1.80E+308	0	1.80E+308	0	1.80E+308	0	1.00E+00	1.00E+00	1.00E+00
F12	3.21E+05	0.36	4.28E+05	0.28	1.80E+308	0	1.81E-07	4.36E-01	2.19E+00
F13	1.50E+05	1	1.80E+308	0	1.80E+308	0	2.52E-06	7.16E-06	2.17E-05
F14	1.80E+308	0	1.80E+308	0	1.80E+308	0	9.98E-01	2.59E+00	9.80E+00
F15	2.50E+05	1	1.80E+308	0	1.80E+308	0	3.28E-04	5.30E-04	6.87E-04
F16	1.25E+03	1	1.25E+03	1	1.25E+03	1	1.03E+00	9.37E-01	2.94E-01
F17	1.80E+308	0	1.80E+308	0	1.80E+308	0	3.98E-01	4.01E-01	4.56E-01
F18	1.80E+308	0	1.80E+308	0	1.80E+308	0	3.00E+00	3.03E+00	3.13E+00
F19	4.00E+03	1	4.00E+03	1	4.00E+03	1	3.86E+00	3.86E+00	3.86E+00
F20	7.50E+03	1	7.50E+03	1	7.50E+03	1	3.32E+00	3.32E+00	3.31E+00
F21	1.00E+04	1	1.00E+04	1	1.00E+04	1	1.02E+01	8.73E+00	2.63E+00
F22	1.00E+04	1	1.00E+04	1	1.00E+04	1	1.04E+01	9.20E+00	3.31E+00
F23	1.00E+04	1	1.00E+04	1	1.00E+04	1	1.05E+01	9.68E+00	2.870458

4.2.3. Test Case 2: Simulations Performed using 25 Iterations by Varying the Dimension of the Functions

Functions F1 to F12 were ran by CRO by varying the dimension of the benchmark functions from 10 to 20 to 30, keeping the number of iterations = 25. 30 independent repetitions were performed and the best, the average and the worst fitness of these functions are tabulated in Tables 5-7 respectively.

Table 7: Best results for functions using 25 iterations varying the dimension

Test Function	Best Result		
	Dim = 10	Dim = 20	Dim = 30
F1	3.87E-09	5.29E-08	3.05E-07
F2	1.28E-04	7.43E-04	1.57E-03
F3	7.64E-10	2.44E-08	5.43E-08
F4	1.07E-04	6.81E-04	5.33E-03
F5	3.80E-01	2.02E+00	1.77E+00
F6	0.00E+00	0.00E+00	0.00E+00
F7	8.75E-05	9.36E-04	2.00E-03
F8	4.19E+03	8.38E+03	1.25E+04
F9	1.41E-05	2.96E-04	1.20E-03
F10	1.04E-03	9.01E-04	1.68E-03
F11	1.00E+00	1.00E+00	1.00E+00
F12	5.74E-09	6.55E-08	1.81E-07

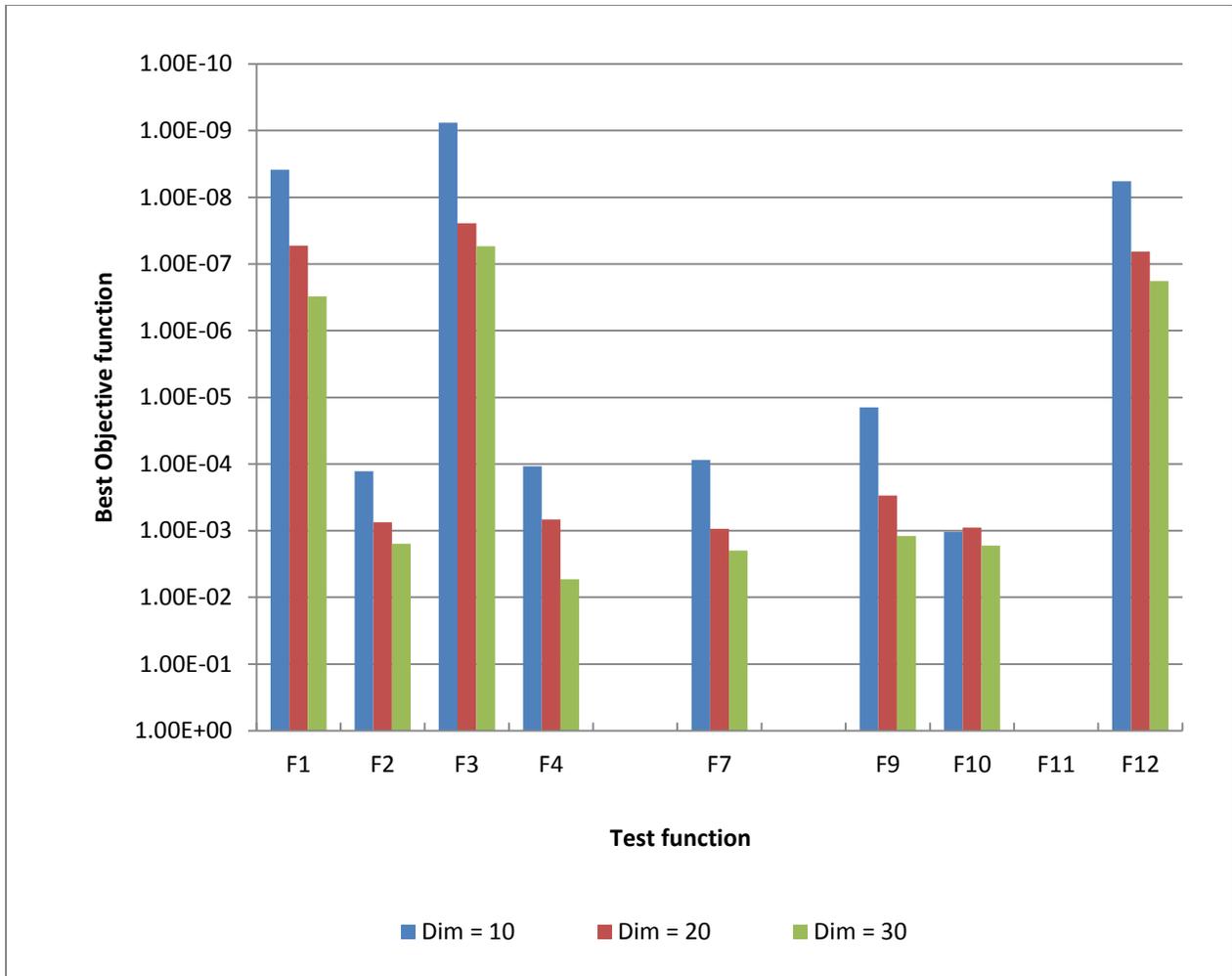


Figure 8: Best results for functions using 25 iterations varying the dimension

Table 8: Average results for functions using 25 iterations varying the dimension

Test Function	Average Result		
	Dim = 10	Dim = 20	Dim = 30
F1	1.60E-08	1.48E-07	6.19E-07
F2	2.31E-04	9.81E-04	2.16E-03
F3	5.12E-09	6.21E-08	2.38E-07
F4	2.05E-04	1.19E-03	8.98E-03
F5	3.45E+01	1.14E+01	7.55E+01
F6	0.00E+00	0.00E+00	0.00E+00
F7	3.87E-04	1.72E-03	3.74E-03
F8	4.19E+03	8.36E+03	1.24E+04
F9	6.20E-05	5.38E-04	1.82E-03
F10	6.59E-04	1.51E-03	2.21E-03
F11	1.00E+00	1.00E+00	1.00E+00
F12	7.46E-02	9.22E-02	4.36E-01

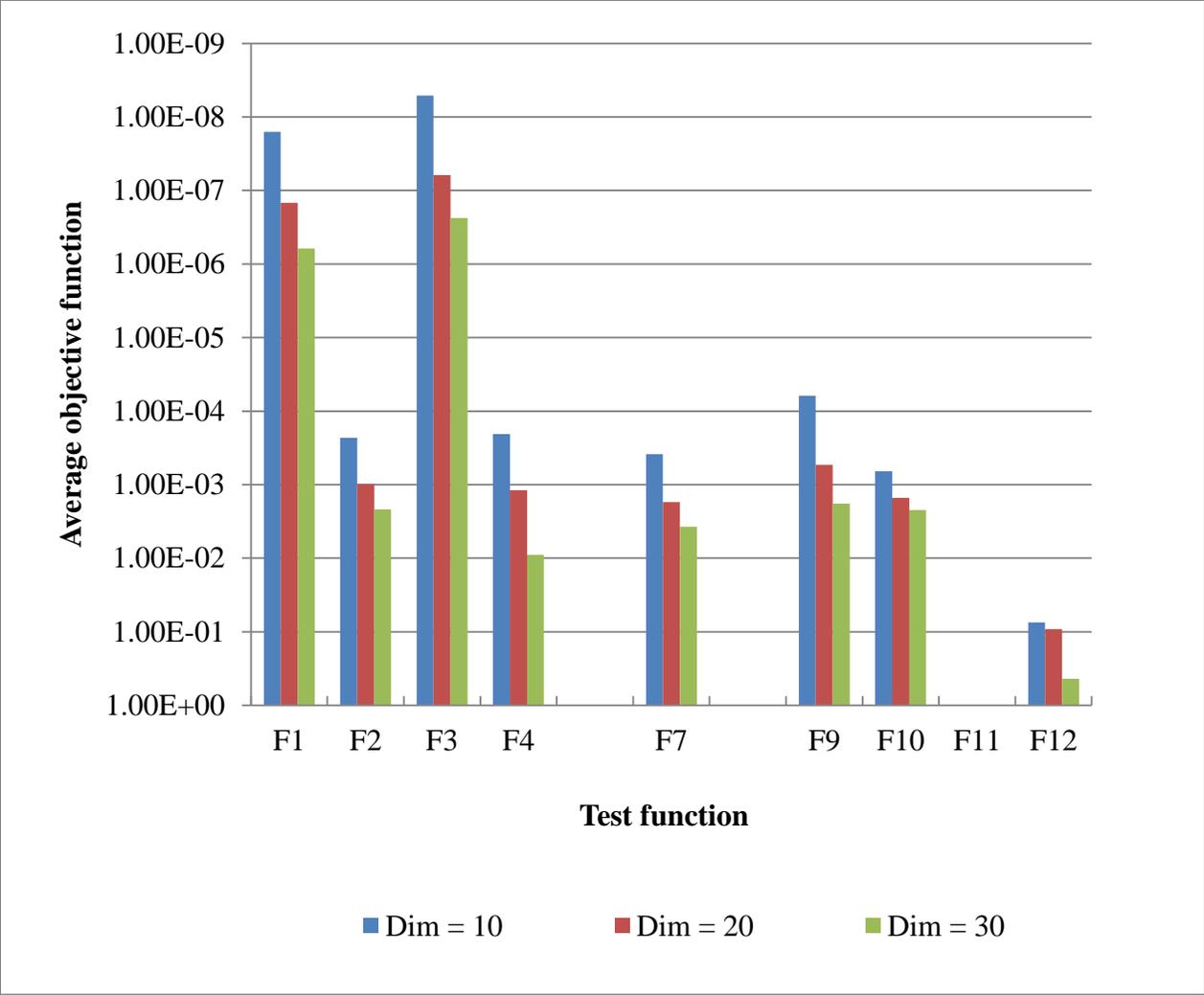


Figure 9: Average results for functions using 25 iterations varying the dimension

Table 9: Worst results for functions using 25 iterations varying the dimension

Test Function	Worst Result		
	Dim = 10	Dim = 20	Dim = 30
F1	4.44E-08	3.35E-07	1.08E-06
F2	3.61E-04	1.26E-03	2.77E-03
F3	1.11E-08	1.18E-07	4.27E-07
F4	3.61E-04	1.75E-03	1.58E-02
F5	2.13E+02	4.79E+02	4.78E+02
F6	0.00E+00	0.00E+00	0.00E+00
F7	9.35E-04	3.34E-03	5.93E-03
F8	4.19E+03	8.30E+03	1.23E+04
F9	1.04E-04	9.56E-04	3.20E-03
F10	1.14E-03	2.22E-03	3.00E-03
F11	1.00E+00	1.00E+00	1.00E+00
F12	9.33E-01	3.11E-01	2.19E+00

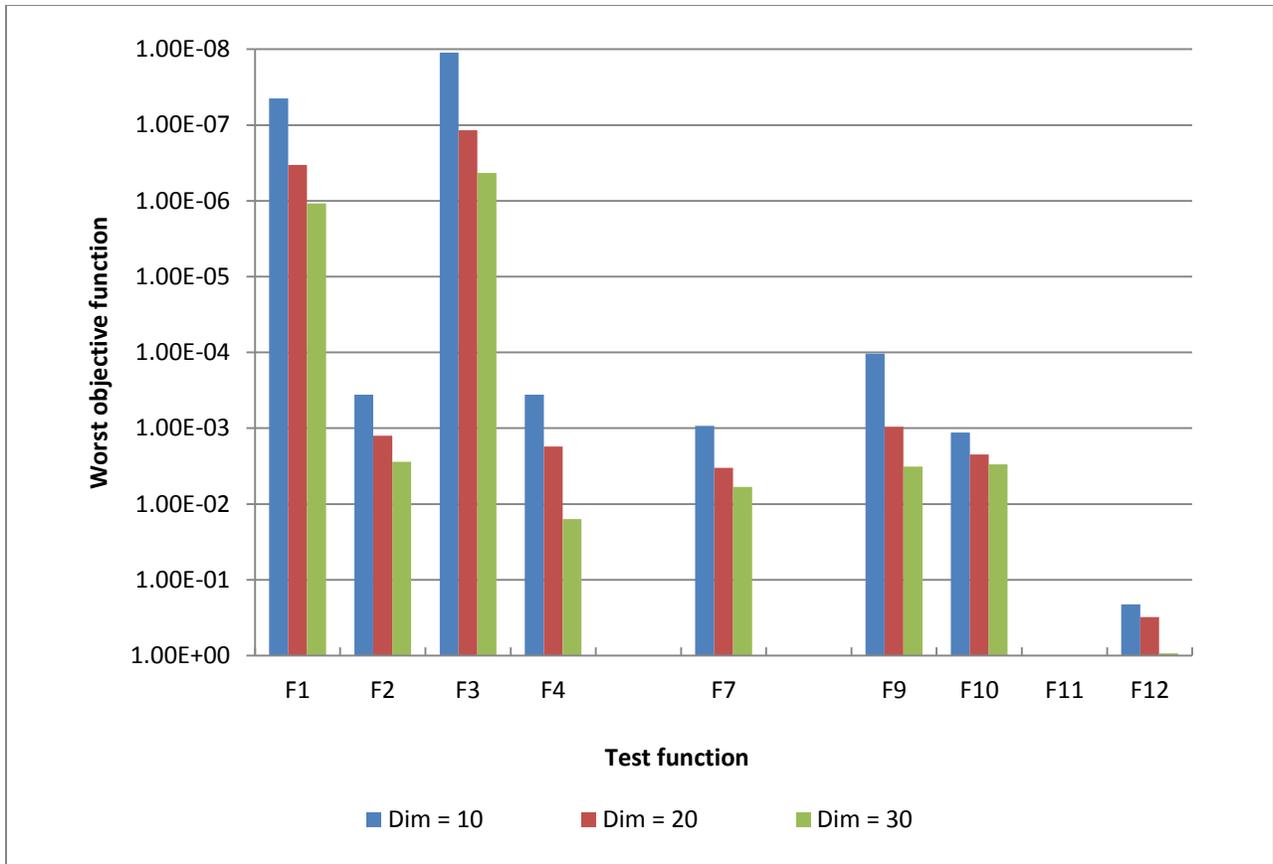


Figure 10: Worst results for functions using 25 iterations varying the dimension

We can clearly see from the tables, that the lower the dimension of the functions, the better is the performance of the algorithm. This is a well-known observation of all known algorithms that the lesser the dimensionality of the problem, the better solution is the obtained in the same amount of time, i.e., iterations.

4.2.4. Test Case 3: Simulations Performed with Dim = 10 by Varying the Number of Iterations

Functions F1 to F12 were ran by CRO with dim = 10, varying the number of iterations from 10^3 to 10^4 to 10^5 . 30 independent repetitions were performed and the best, the average and the worst fitness of these functions were tabulated in Tables 8-10 respectively.

Table 10: Best results for functions ran with dim = 10 varying number of iterations

Test Function	Best result for dim = 10		
	Iteration = 10³	Iteration = 10⁴	Iteration = 10⁵
F1	1.81E-09	6.97E-10	5.77E-10
F2	8.17E-05	5.77E-05	4.27E-05
F3	3.80E-10	2.41E-10	6.25E-11
F4	4.68E-05	5.09E-05	3.05E-05
F5	8.67E-03	2.18E-05	4.79E-06
F6	0.00E+00	0.00E+00	0.00E+00
F7	3.04E-05	2.28E-05	5.21E-06
F8	4.19E+03	4.19E+03	4.19E+03
F9	5.66E-06	2.86E-06	2.60E-06
F10	2.48E-04	1.89E-04	1.46E-04
F12	1.80E-09	1.57E-09	7.85E-10

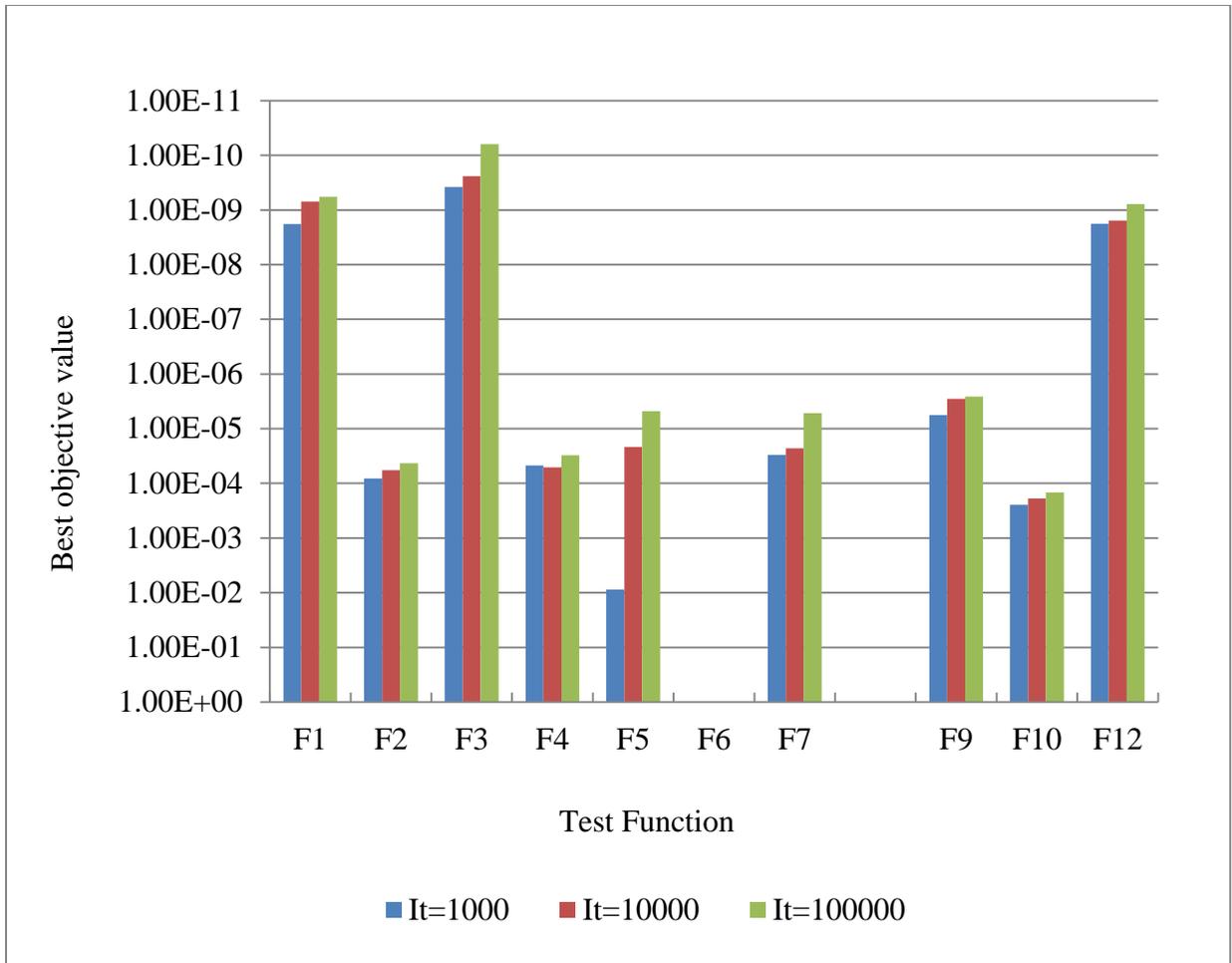


Figure 11: Best results for functions with 10 dim varying the number of iterations

Table 11: Average results for functions with 10 dim varying the number of iterations

Test Function	Average result for dim = 10		
	Iteration = 10³	Iteration = 10⁴	Iteration = 10⁵
F1	1.56E-08	1.54E-08	1.55E-08
F2	2.33E-04	2.34E-04	2.34E-04
F3	5.09E-09	4.93E-09	4.94E-09
F4	2.12E-04	2.15E-04	2.16E-04
F5	2.37E+01	2.60E+01	2.57E+01
F7	4.83E-04	4.69E-04	4.69E-04
F8	4.19E+03	4.19E+03	4.19E+03
F9	5.88E-05	5.93E-05	5.95E-05
F10	7.13E-04	7.01E-04	7.01E-04
F11	1.00E+00	1.00E+00	1.00E+00
F12	3.36E-02	3.32E-02	3.44E-02

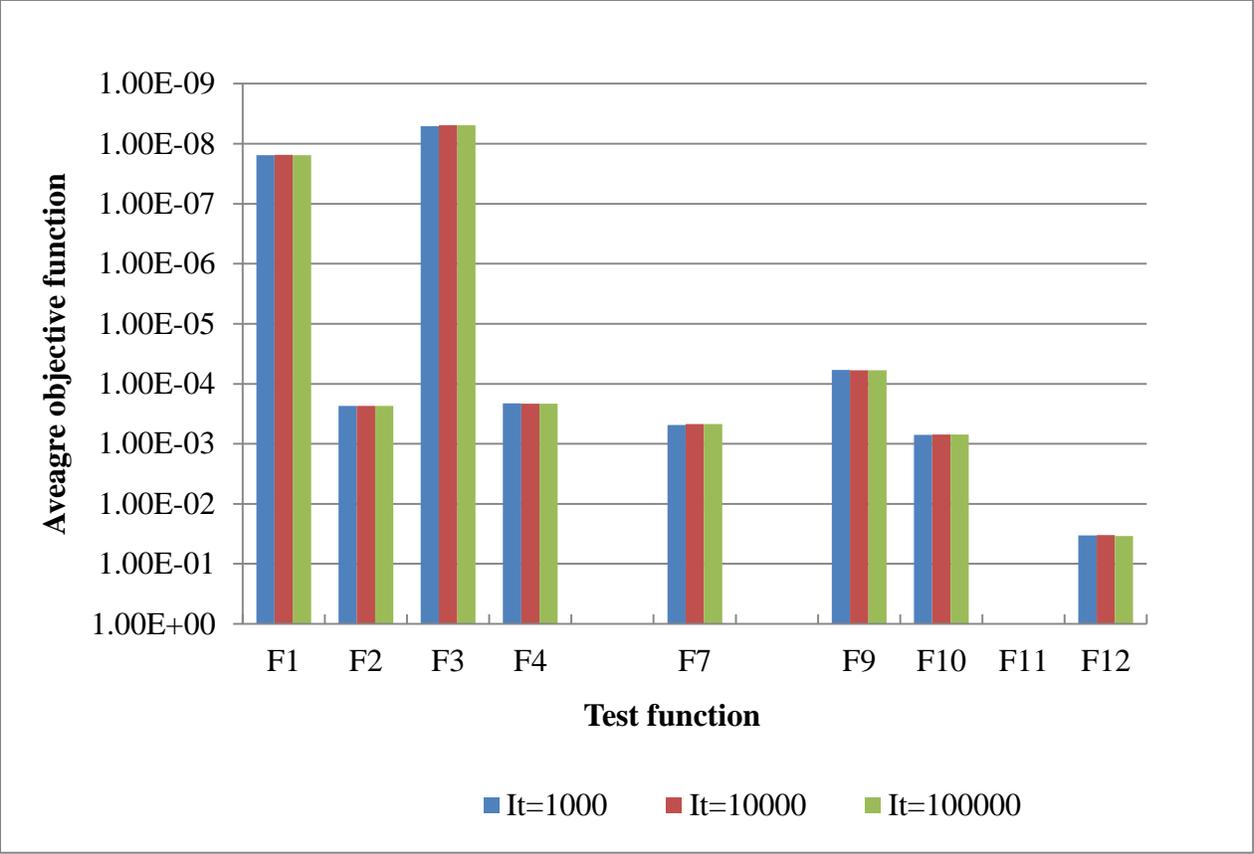


Figure 12: Average results for functions with 10 dim varying the number of iterations

Table 12: Worst results for functions with 10 dim varying the number of iterations

Test	Worst result for dim = 10		
Function	Iteration = 10^3	Iteration = 10^4	Iteration = 10^5
F1	6.32E-08	7.81E-08	9.30E-08
F2	5.55E-04	5.35E-04	6.50E-04
F3	2.16E-08	2.92E-08	4.42E-08
F4	0.000561	0.000553	5.97E-04
F6	0.00E+00	0.00E+00	0.00E+00
F7	1.87E-03	2.16E-03	2.60E-03
F9	2.32E-04	2.82E-04	3.39E-04
F10	1.89E-03	1.88E-03	2.30E-03
F11	1.00E+00	1.00E+00	1.00E+00

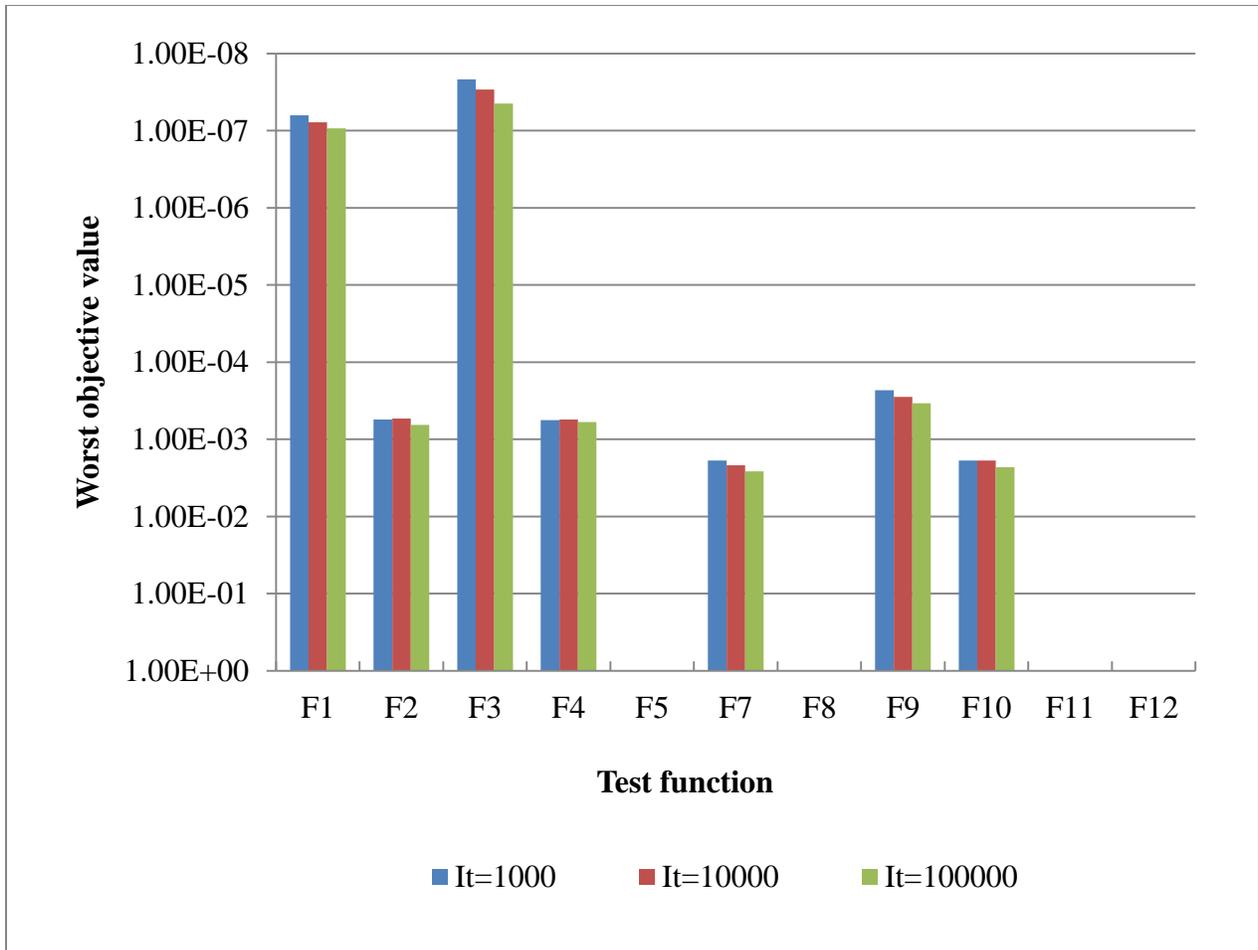


Figure 13: Worst results for functions with 10 dim varying the number of iterations

We can clearly see that there is an increase in the performance of the values with the increase in the number of iterations. But the increase is not very significant with the change in the number of iterations, proving that CRO performs better with as low as 25 iterations.

4.2.5. Test Case 4: Simulations Performed with Dim = 10 and Number of Iterations = 100000; by Varying the Population Size of the Problem

Function F1 was simulated using CRO by varying the population size (PopSize) of the algorithm.

Table 13: Varying the population size for f1 and evaluating best objective value

PopSize	Best Objective Value	Standard Deviation
5	1.19E-09	8.55E-09
10	6.97E-10	6.76E-10
15	8.46E-10	3.17E-09
20	9.56E-10	6.55E-07
25	1.03E-09	8.49E-09
30	1.18E-09	6.65E-09
35	1.35E-09	3.22E-11
40	1.36E-09	6.56E-09

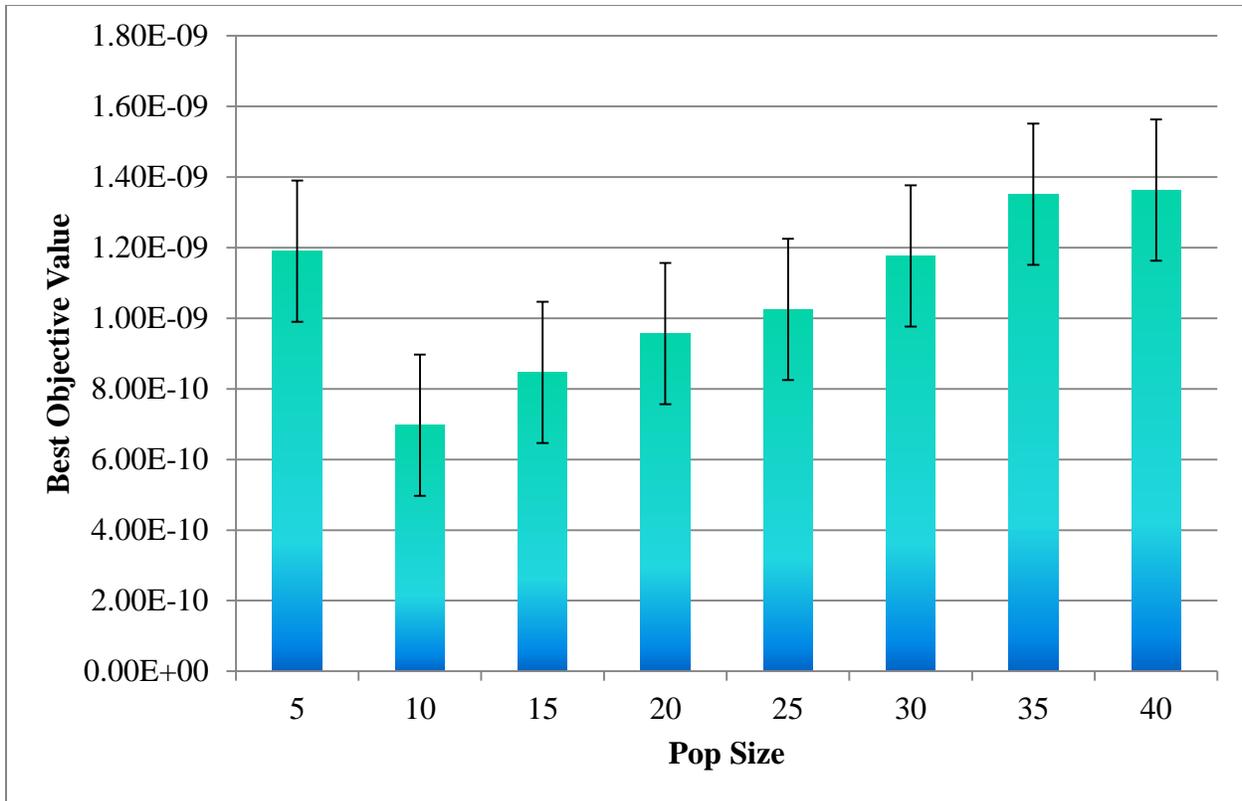


Figure 14: Varying the population size for f1 and evaluating best objective value

Figure 14 shows clearly that an optimal solution was found with the population size of 10 in comparison with the other population size values.

4.2.6. Test Case 5: Simulations Performed with Dim = 10 and Number of Iterations = 100000; by Varying the StepSize of the Problem

Function F1 was simulated using CRO by varying the StepSize of the problem.

Table 14: Varying the step size for f1 and evaluating best objective value

StepSize	Best Objective Value	Standard Deviation
0.04	6.57E-11	3.43E-12
0.06	1.57E-10	2.32E-11
0.08	2.23E-10	4.65E-11
0.1	1.92E-10	3.20E-11
0.14	5.47E-10	6.50E-11
0.2	8.90E-10	2.03E-10
0.3	1.69E-09	4.68E-11

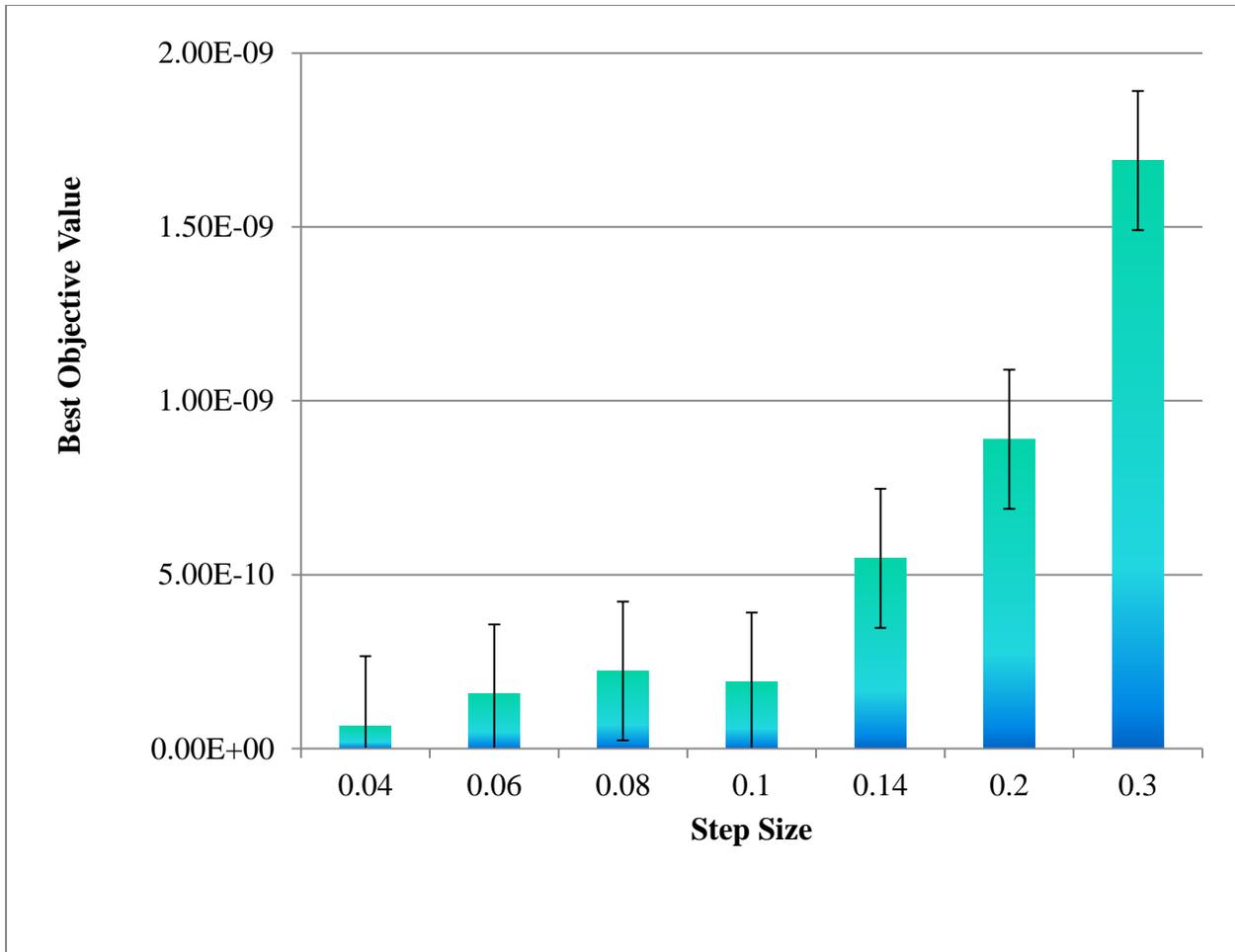


Figure 15: StepSize tuning for f1 and evaluating best objective function

Figure 15 shows clearly that optimal solutions were found when the StepSize is less than 10 in comparison with the other StepSize values.

4.2.7. Test Case 6: Simulations Performed with Dim = 10 and Number of Iterations = 100000; by Varying the EnergyBuffer Value of the Problem

Function F1 was simulated using CRO by varying the EnergyBuffer value of the problem.

Table 15: Varying the energy buffer for f1 and evaluating best objective value

EnergyBuffer	Best Objective Value	Standard Deviation
0	1.92E-10	2.08E-07
10	2.38E-10	3.56E-06
100	2.95E-10	8.79E-08
1000	3.54E-10	1.55E-08

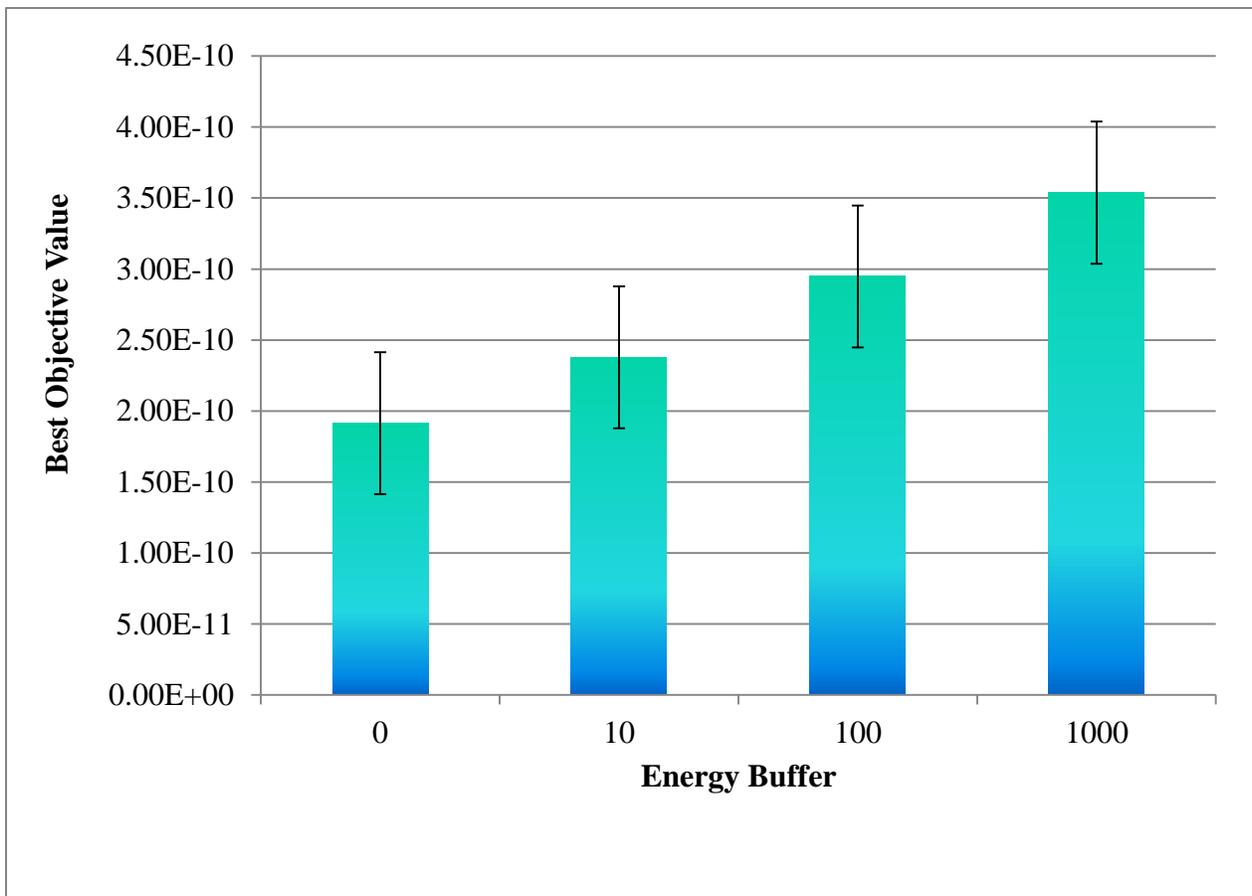


Figure 16: Energy Buffer tuning for f1 and evaluating best objective function

Figure 16 shows clearly that an optimal solution was found when the EnergyBuffer value = 0 in comparison with the other EnergyBuffer values.

5. CONCLUSION AND FUTURE WORK

In a chemical reaction, reactants naturally tend to undergo transformations to form more stable (less energetic) species called products via a sequence of elementary steps, involving transition states and intermediates. Chemical Reaction Optimization captures this idea to solve optimization problems in a metaheuristic manner.

Past research has observed that CRO is not restricted to just continuous or just discrete optimization problem. It has successfully solved both continuous and discrete problems.

In this paper, the CRO algorithm has been evaluated on a variety of benchmark functions of different complexity. The findings revealed that CRO performs well and different test cases were run in order to evaluate the performance of CRO.

CRO performed better with the increase in number of iterations, but the level of improvement is less, which suggests that even with the lesser number of iterations, we can have better results. Varying the size of the variable of the functions increased the efficiency of the results.

Also, initial parameter effects were studied to benchmark the parameters that could hold good for all other search problems. Results have showed that effect of change in PopSize, StepSize and Buffer drastically affect the performance of the algorithm solving these functions.

In the future, we plan to apply CRO to more continuous problems. We can extend the study of parameter values of CRO using a calibration method. Furthermore, we implement a self-adaptation of the parameter values to each molecule, so that the efficiency of the system would be improved.

6. REFERENCES

1. S. F. Boys, G. B. Cook, C. M. Reeves and I. Shavitt, "Automatic fundamental calculations of molecular structure", *Nature*, vol. 178, no. 2, pp. 1207, 1956.
2. M. AlRashidi and M. El-Hawary, "A survey of particle swarm optimization applications in electric power systems", *IEEE Trans Evol Comput*, vol. 13, no. 4, pp. 913–918, 2009.
3. E.M. Loiola, N.M.M. de Abreu, P.O. Boaventura-Netto, P. Hahn and T. Querido, "A survey for the quadratic assignment problem" *Eur J Oper Res*, vol. 176, no. 2, pp. 657–690, 2007.
4. E.L. Demeulemeester and W.S. Herroelen, "Project scheduling: a research handbook", Academic Publishers, Boston, MA, USA, 2002.
5. A.P. Subramanian, H. Gupta, S.R. Das and J. Cao, "Minimum interference channel assignment in multiradio wireless mesh networks", *IEEE Trans Mobile Comput*, vol. 7, no. 12, pp. 1459–1473, 2008.
6. A. Y.S. Lam, J. Xu and V. O.K. Li, "Chemical reaction optimization for population transition in peer-to-peer live streaming", In: *Proceedings of the IEEE congress on evolutionary computation*. Barcelona, Spain, 2010.
7. C. Peng, H. Zheng and B.Y. Zhao, "Utilization and fairness in spectrum assignment for opportunistic spectrum access" *ACM/Kluwer Mobile Netw Appl*, vol. 11, no. 4, pp. 555–576, 2006.
8. G. Ritchie and J. Levine, "A hybrid ant algorithm for scheduling independent jobs in heterogeneous computing environments", In: *Proceedings of 23rd workshop of the UK planning and scheduling special interest group*. Cork, Ireland, 2004.

9. X. Yao, Y. Liu and G. Lin, "Evolutionary programming made faster", *IEEE Trans Evol Comput*, vol. 3, no. 2, pp. 82–102, 1999.
10. G.D. Tollo and A. Roli, "Metaheuristics for the portfolio selection problem", *J Financial Quant Anal*, vol. 8, no. 4, pp. 621–636, 2008.
11. M. Kim, M. Medard, V. Aggarwal, U.M. O'Reilly, W. Kim and C.W. Ahn, "Evolutionary approaches to minimizing network coding resources", In: *Proceedings of the 26th annual IEEE conference on computer Communications*, Anchorage, AK, USA, 2007.
12. P.P. Palmes, T. Hayasaka and S. Usui, "Mutation-based genetic neural network", *IEEE Trans Neural Netw*, vol. 16, no. 3, pp. 587–600, 2005.
13. S.-Y. Shin, I.-H. Lee, D. Kim and B.-T. Zhang, "Multiobjective evolutionary optimization of DNA sequences for reliable DNA computing," *IEEE Trans. Evol. Comput.*, vol. 9, no. 2, pp. 143–158, Apr. 2005.
14. M.R. Garey and D.S. Johnson, "Computers and intractability: A guide to the theory of NP-completeness" WH Freeman & Co Ltd, New York, 1979.
15. N. Shadbolt, "Nature-inspired computing" *IEEE Intell Syst*, vol. 19, no. 1, pp. 2–3, 2004.
16. A. Y. S. Lam and V. O. K. Li, "Chemical-reaction-inspired metaheuristic for optimization," *IEEE Trans. Evol. Comput.*, vol. 14, no. 3, pp. 381– 399, Jun. 2010.
17. D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, MA: Addison-Wesley, 1989.
18. M. Dorigo and T. Stutzle, "Ant Colony Optimization. Cambridge, MA: MIT Press, 2004.
19. J. Kennedy and R. Eberhart, *Swarm Intelligence*. San Francisco, CA: Morgan Kaufmann, 2001

20. X.S. Chen, Y.S. Ong, M.H. Lim and K.C. Tan, "A multi-facet survey on memetic computation", IEEE Trans Evol Comput, vol. 15, no. 5, pp. 591– 607, 2011.
21. D. Whitley, T. Starkweather and D. Fuquay, "Scheduling problems and traveling salesmen: The genetic edge recombination operator", Proceedings of the Third International Conference on Genetic Algorithms, pp.133 -140, 1989.
22. I.A. Ismail, N.A. El_Ramly, M.M. El_Kafrawy and M.M. Nasef, "Game Theory Using Genetic Algorithms", Proceedings of the World Congress on Engineering 2007 Vol I WCE 2007, London, U.K., July 2 - 4, 2007.
23. G. Syswerda, "The Application of Genetic Algorithms to Resource Scheduling." Proceedings from the Fourth International Conference on Genetic Algorithms: pp. 502-508, 1990.
24. R.M. Karp, "Reducibility among combinatorial problems", In: R.E. Miller and J.W. Thatcher (eds.), Complexity of Computer Computations. Plenum, New York NY, pp. 85–103, 1972.
25. T. Ichimura and Y. Kuriyama, "Learning of neural networks with parallel hybrid ga using a royal road function", In: 1998 IEEE International Joint Conference on Neural Networks, Vol. 2, IEEE, New York, NY, pp. 1131–1136, 1998.
26. I.C. Yeh, "Hybrid genetic algorithms for optimization of truss structures. Computer Aided Civil and Infrastructure Engineering", vol. 14, no. 3, pp. 199–206, 1999.
27. R.J.W. Hodgson, "Memetic algorithms and the molecular geometry optimization problem", In: Proceedings of the 2000 Congress on Evolutionary Computation. IEEE Service Center, Piscataway, NJ, pp. 625–632, 2000.

28. D. Martens, M. De Backer, R. Haesen, J. Vanthienen, M. Snoeck and B. Baesens, "Classification with Ant Colony Optimization", IEEE Transactions on Evolutionary Computation, vol. 11, no. 5, pp. 651—665, 2007.
29. N. Secomandi, "Comparing neuro-dynamic programming algorithms for the vehicle routing problem with stochastic demands," Computers & Operations Research, vol.27, no.11, pp.1201-1225, 2000.
30. T. Stützle, "MAX-MIN Ant System for the quadratic assignment problem," Technical Report AIDA-97-4, FB Informatik, TU Darmstadt, Germany, 1997
31. D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization" IEEE Trans Evol Comput, vol. 1, no. 1, pp. 67–82, 1997.
32. E.A. Guggenheim, "Thermodynamics: an advanced treatment for chemists and physicists", 5th edn. Wiley, North Holland, 1967.
33. A. Y.S. Lam and V. O.K. Li, "Chemical reaction optimization: A tutorial," Memetic Computing, vol. 14, no. 1, pp. 3-17, Mar. 2012.
34. A. Y.S. Lam and V. O.K. Li, "Chemical-reaction-inspired metaheuristic for optimization", IEEE Trans Evol Comput, vol. 14, no. 3, pp. 381–399, 2010.