



UNIVERSITY MOHAMED BOUDIAF OF M'SILA
FACULTY OF MATHEMATICS AND
INFORMATICS



COMPUTER SCIENCE DEPARTMENT

**A dissertation in fulfillment
For the requirement of
MASTER Degree in Computer Science
Domain: Mathematics and Informatics
Branch: Computer Science
Option: Networks**

By: Ben Hamida Yazid

SUBJECT

**Quantum-Inspired Genetic Algorithm for a
Class of Combinatorial Optimization**

Defended on: / / 2016 in front of the jury made up of:

Mr. Moussaoui Adel.

Mr. Bouhouita Guermech Salah Eddine

University of M'sila supervisor

University of M'sila Chair

University of M'sila Examiner

University of M'sila Examiner

Promotion: 2015 /2016

THANKS

First praise and thanks to Allah ,
for his help and save and facilitating .
praise him that pleased me to complete this work .then my
great thanks are to the one who stood by my side with his support
and guidance in this project my Great supervisor Mr.SallehBouhuita & Mr.AdelMoussaoui
also I thank my colleagues in Bouhuita groups.
To Maria and all Friends
To my family and all those who helped me especially My father and my mother and Zineb
To all members of Samir Pharmacy
finally, I thank all those who participated in a way or another to the success of this work.

table of contents

Introduction	1
--------------------	---

Chapter 1: Quantum Computing.

1. Introduction:.....	4
2. Q_algebra:	4
3. Qbit:.....	5
3.1. Matrix representation :	5
4. Quantum registers:.....	6
5. The classic model inspired by the quantum (CIQ):.....	6
5.1. the Qbit :.....	6
5.2. The Qreg:	7
5.3. Measurement:	10
5.4. Collapse process:	10
6. Interference phenomenon, notion and example	10
6.1. Phase shifting:	11
7. Conclusion	11

Chapter 2: quantum inspired Evolutionary Algorithms

1. Introduction :.....	13
2. Evolutionary algorithms:	13
3. Evolutionary algorithms computational model:	14
4. Fundamental concepts of evolutionary algorithms:	
..... Error! Bookmark not defined.	15
4.1. Selection	15
4.2. Recombination	15
4.3. Mutation.....	16
4.4. Reinsertion	16
5. Quantum inspired evolutionary algorithms concepts:	16
5.1. Principle of superposition:	16
5.2. Representation of a quantum entity:	16

table of contents

6.	Quantum probability:	17
7.	Quantum Measurement:	17
8.	Evolutionary quantum algorithms:	19
8.1.	QEA algorithm:	19
8.2.	Concepts and procedures pivots of the algorithm:.....	19
8.2.2.	The Fitness:.....	20
8.2.3.	Evolution:	20
8.2.4.	Measurement:.....	20
8.2.5.	The selection:	20
9.	the flowchart of the algorithm QEA:	20
10.	Description of the algorithm:	21
11.	conclusion:.....	21

Chapter 3 : quantum inspired genetic algorithm

1.	Introduction:.....	23
2.	Q_algebra:	23
3.	Qbit:.....	24
3.1.	Matrix representation :	24
4.	Quantum registers:.....	25
5.	The classic model inspired by the quantum (CIQ):.....	25
5.1.	the Qbit :.....	25
5.2.	The Qreg:	26
5.3.	Measurement:	29
5.4.	Collapse process:	29
6.	Interference phenomenon, notion and example.....	29
6.1.	Phase shifting:	30
7.	Conclusion	30

chapter 4 : experiments and Analyze.

1. Introduction :	32
2. Programming language:	32
2.1. Fortran :	32
3. Used software:	32
3.2. Gnuplot :	33
3.2.1. Example command line:	34
3.2.2. Programming and application interfaces:	35
4. Collapse process:	35
4.1. Experiment without using collapse process:	35
4.2. Experiment using collapse process:	36
5. Analyze the effect of the interference reflected by crossover operation on the algorithm performance:	37
6. Statics analyze of interference (crossover) experiments:	43
4.2. Average value of solutions found and standard deviation:	44
5. The effect of interference (crossover operation):	45
6. Analyze the effect the phase shifting reflected by mutation operation on the algorithm performance:	46
7. Statics analyses of phase shifting (mutation)experiments:	50
7.1. Maximum solution achieved:	51
8. Fluctuation:	51
8.1. The fitness average value and standard deviation:	52
8.2. The behavior of GIQA over 1000 iteration:	53
9. Qbit behavior :	53
10. Conclusion:	55

General Introduction:

Quantum information and computation is a new exciting field of research, while quantum information investigates the informational power of quantum bit (Qbit) which is the quantum information unit, the quantum computation tries to exploit this quantum informational capability to build a strong quantum algorithms to solve a class of problems that seems to be unsolvable by the conventional computers. Unfortunately, the design of quantum computer faces the difficulty of coherence with environment.

Still we can exploit the concepts of quantum information to inspire classical techniques that could give classical algorithms better performance. A quantum computation is probabilistic by nature, and so it can be more exploitable when combined with random algorithms which are more powerful than exact ones.

Evolutionary algorithms like genetic ones try to deal with probability through nice models inspired of biological real ones in nature. It seems natural that these two different techniques should combine their computational powers in a hope to require better capability to solve hard problems.

Genetic algorithms are stochastic ones; they deal with probability upon the notion of population, rather than quantum information, in which the probability is intrinsic to the quantum bit (Qbit). So, the population will be an ensemble of Qbits.

The algebra of the Qbits is built upon matrices, which suggest, that the inspired quantum algorithms should choose the same algebra. Q_algebra has three notions: Qbit is represented by a binary linear Hilbert space, so each Qbit is represented by a unitary binary column vector; geometrically it can better represented by Bloch sphere. The evolution of the Qbit is represented by a unitary matrix, and the measurement of the Qbit collapsed it in the classical bit (Cbit) $\in \{0, 1\}$.

The algebra of GAs evolves initial population of potential solutions taken randomly from the whole solution space. This evolution is done applying the principle of genetics like crossover and mutation. The fittest individuals make next generation. This process iterate in hope that each generation approaches better the solution seeking than the generation that generate them.

Q_algebra and G_algebra have to interfere constructively to create a better algorithm.

This study proposes a novel genetic algorithm inspired from quantum information and computation concepts (QIGA) which resulted from nice interference between the algebra of quantum computation and genetic algorithm. QIGA still based population with all its dynamic characteristics, like : individual representation, the evolution function, but their definition is inspired by quantum notions, as, linear superposition (Qbit), unitary transformation (crossover and mutation), measurement (Qbit to Cbit).

General Introduction

QIGA offers better diversity, because probability is taken deep to the representation of the individual itself. This manner augments the exploration and mutation because they manipulate probabilities through Qbits.

To validate the performance of QIGA, experiments are carried out on knapsack problem.

To appropriately do our study, we organized the dissertation as follows:

Chapter 1: presentation of quantum information and computation formalism.

Chapter 2 : quantum inspired evolutionary algorithm are introduced

Chapter 3 : the proposed algorithm QIGA is presented.

Chapter 4 : experiments are carried out and results are discussed.

.

CHAPTER 1

Quantum Computing

1. Introduction:

The computer is a device that processes information by executing algorithms. An algorithm is a well-defined procedure, with a finite description for the realization of an information processing task. Each task can always be translated into a physical task.

Quantum computation is trying to explore and exploit the phenomena of quantum mechanics to design quantum algorithms. The advantage of these their capacity to solve a class of problems which is impossible or difficult to solve with conventional algorithms.

Quantum mechanics provides a computational Framework, revealing a new aspect of probability; the probability is obtained in a more fundamental entity, which is the amplitude. Classical probability quantifies the lack of information, in other words the ignorance we have of the system. While quantum probability is intrinsic to the physical system, it expresses that the information is probabilistic in its fundamental nature. In the section below, we present a concise introduction of quantum mechanics.

2. Q_algebra:

Quantum mechanics is formalized upon a linear algebra of the vector space.

Space : Quantum world is modeled by Hilbert space.

The Hilbert space (H) is a vector space defined on the set of complex numbers, where we prefer Dirac notation to represent vectors:

Dirac notation is inspired of the bracket $\langle \varphi | \chi \rangle$ to express the inner-product of two vectors: ket $|\varphi\rangle$ and bra $\langle \chi|$

Where: $|\varphi\rangle, |\chi\rangle \in H$ and $\langle \varphi | \chi \rangle \in \mathbb{C}$

Linearity of Hilbert space involves the following properties [1]:

$$(a) |\alpha\varphi\rangle = \alpha|\varphi\rangle \forall \alpha \in \mathbb{C}, \forall |\varphi\rangle \in H \quad (1)$$

$$(b) |\alpha\varphi + \beta\chi\rangle = \alpha|\varphi\rangle + \beta|\chi\rangle \forall \alpha, \beta \in \mathbb{C}, \forall |\varphi\rangle, |\chi\rangle \in H \quad (2)$$

$$(c) \langle \alpha\varphi + \beta\chi | = \alpha^*\langle \varphi | + \beta^*\langle \chi | \quad \forall \alpha, \beta \in \mathbb{C}, \forall |\varphi\rangle, |\chi\rangle \in H \quad (3)$$

The proprieties (b) and (c) express the linearity of vectors which lead to superposition of states the heart of quantum mechanics and which contains all its secrets and power. The

superposition leads to interference and entanglement; the notions exploited for quantum computation.

3. Qbit:

The simplest quantum system is the two-state system, the Qubit "Quantum bit." [3]. Associated to the Hilbert space H_Q . Its computational basis is $\{|0\rangle, |1\rangle\}$. So the Qbit $|\varphi\rangle$ is written as a superposition of this basis.

$$\begin{cases} |\psi\rangle = \alpha|0\rangle + \beta|1\rangle \\ |\alpha|^2 + |\beta|^2 = 1 : \alpha, \beta \in \mathbb{C} \end{cases} \quad (4)$$

To better understand the behavior of a Qbit we can use a beautiful geometric representation which is the Bloch sphere.

The Qbit is represented as follows:

$$\begin{cases} |\psi\rangle = \cos(\theta)|0\rangle + e^{i\varphi}\sin(\theta)|1\rangle \\ \theta \in [0, \pi], \quad \varphi \in [0, 2\pi] \\ \alpha = \cos(\theta), \beta = e^{i\varphi}\sin(\theta) \end{cases} \quad (5)$$

Where:

$$\begin{cases} |\alpha|^2 = \cos^2(\theta) = p(|0\rangle) \\ |\beta|^2 = \sin^2(\theta) = p(|1\rangle) \end{cases} \quad (6)$$

$|\alpha|^2$ gives the probability that the Qbit will be found in the "0" state and $|\beta|^2$ gives the probability that the Qbit will be found in the "1" state. [22.pdf p 2]. This process is called the measurement of the state.

3.1. Matrix representation :

Matrix representation is the first mathematical framework proposed by Heisenberg, it is simple and powerful.

$$\text{basis} = \begin{cases} |0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \\ |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \end{cases} \quad (7)$$

So; the representation of the Qbit in a matrix model is as follow:

$$\begin{cases} |\psi\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \cos(\theta) \\ e^{i\varphi} \sin(\theta) \end{pmatrix} \\ \theta \in [0, \pi], \quad \varphi \in [0, 2\pi] \end{cases} \quad (8)$$

4. Quantum registers:

The quantum register is a juxtaposition set of Qbits, the state of a quantum register with m Qbits is given by [3]:

$$|QReg\rangle = |\psi_1\rangle |\psi_2\rangle \dots |\psi_m\rangle \quad (9)$$

Its matrix representation is:

$$|QReg\rangle = \begin{pmatrix} \alpha_1 \\ \beta_1 \end{pmatrix} \begin{pmatrix} \alpha_2 \\ \beta_2 \end{pmatrix} \dots \begin{pmatrix} \alpha_m \\ \beta_m \end{pmatrix} \quad (10)$$

$$\text{Where: } |\alpha_i|^2 + |\beta_i|^2 = 1, \quad \alpha_i, \beta_i \in \mathbb{C}, \quad i = 1, 2, \dots, m.$$

5. The classic model inspired by the quantum (CIQ):

The realization of quantum computer faces enormous challenges. The strategy is to inspire the maximum possible quantum mechanisms that could be implemented on a conventional computer.

5.1. the Qbit :

The Hilbert space H is replaced by the Euclidian space which is a vector space defined upon real ensemble instead of complex one.

ACIbit is represented by:

$$\begin{cases} |\psi\rangle = \alpha|0\rangle + \beta|1\rangle \\ \alpha^2 + \beta^2 = 1 : \alpha, \beta \in R \end{cases} \quad (11)$$

Where:

$$\begin{cases} \alpha^2 = p(\{0\}) \\ \beta^2 = p(\{1\}) \end{cases} \quad (12)$$

Thereby, adequate mathematical representation is the matrix. The state of CIQbit is described by the vector column.

$$|\psi\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}, \quad \alpha^2 + \beta^2 = 1, \alpha, \beta \in R \quad (13)$$

5.2. The Qreg:

The general diagram of a register which simulates the quantum registers with m Qbits[3]:

$$QReg = \begin{pmatrix} \alpha_1 & \dots & \alpha_m \\ \beta_1 & \dots & \beta_m \end{pmatrix} \quad (14)$$

Where: $\alpha_i^2 + \beta_i^2 = 1, \alpha_i, \beta_i \in R, i = 1, 2, \dots, m$.

Example:

We give example of a register of 3 Qbits.

$$QReg = \begin{pmatrix} 1/\sqrt{2} & -1/\sqrt{3} & 1/2 \\ -1/\sqrt{2} & -\sqrt{2}/\sqrt{3} & \sqrt{3}/2 \end{pmatrix}$$

The amplitudes of each bit verify the unitary constraint:

$$\begin{cases} \left(\frac{1}{\sqrt{2}}\right)^2 + \left(-\frac{1}{\sqrt{2}}\right)^2 = 1 \\ \left(-\frac{1}{\sqrt{3}}\right)^2 + \left(-\frac{\sqrt{2}}{\sqrt{3}}\right)^2 = 1 \\ \left(\frac{1}{2}\right)^2 + \left(\frac{\sqrt{3}}{2}\right)^2 = 1 \end{cases}$$

The register itself describes a quantum state, which we present in the table below to seek more clarity.

Interference of Qregister	states
$\left(\frac{1}{\sqrt{2}}\right)^*$ $\left(-\frac{1}{\sqrt{3}}\right)^* \left(\frac{1}{2}\right) = -\frac{1}{2\sqrt{6}}$	$ 000\rangle$
$\left(\frac{1}{\sqrt{2}}\right)^* \left(-\frac{1}{\sqrt{3}}\right)^* \left(\frac{\sqrt{3}}{2}\right) = -\frac{1}{2\sqrt{2}}$	$ 001\rangle$
$\left(\frac{1}{\sqrt{2}}\right)^* \left(-\frac{\sqrt{2}}{\sqrt{3}}\right)^* \left(\frac{1}{2}\right) = -\frac{1}{2\sqrt{3}}$	$ 010\rangle$
$\left(\frac{1}{\sqrt{2}}\right)^* \left(-\frac{\sqrt{2}}{\sqrt{3}}\right)^* \left(\frac{\sqrt{3}}{2}\right) = -\frac{1}{2}$	$ 011\rangle$
$\left(-\frac{1}{\sqrt{2}}\right)^* \left(-\frac{1}{\sqrt{3}}\right)^* \left(\frac{1}{2}\right) = \frac{1}{2\sqrt{6}}$	$ 100\rangle$
$\left(-\frac{1}{\sqrt{2}}\right)^* \left(-\frac{1}{\sqrt{3}}\right)^* \left(\frac{\sqrt{3}}{2}\right) = \frac{1}{2\sqrt{2}}$	$ 101\rangle$
$\left(-\frac{1}{\sqrt{2}}\right)^* \left(-\frac{\sqrt{2}}{\sqrt{3}}\right)^* \left(\frac{1}{2}\right) = \frac{1}{2\sqrt{3}}$	$ 110\rangle$
$\left(-\frac{1}{\sqrt{2}}\right)^* \left(-\frac{\sqrt{2}}{\sqrt{3}}\right)^* \left(\frac{\sqrt{3}}{2}\right) = \frac{1}{2}$	$ 111\rangle$

Tableau 1 :Qregister interference

So, in fact the register confines the following quantum state:

$$\begin{aligned}
 |QReg\rangle = & \frac{-1}{2\sqrt{6}}|000\rangle - \frac{1}{2\sqrt{2}}|001\rangle - \frac{1}{2\sqrt{3}}|010\rangle - \frac{1}{2}|011\rangle \\
 & + \frac{1}{2\sqrt{6}}|100\rangle + \frac{1}{2\sqrt{2}}|101\rangle + \frac{1}{2\sqrt{3}}|110\rangle - \frac{1}{2}|111\rangle
 \end{aligned}$$

The constraint of unitarity is expressed as follow:

$$\begin{aligned}
 & \left(\frac{-1}{2\sqrt{6}}\right)^2 + \left(\frac{-1}{2\sqrt{2}}\right)^2 + \left(\frac{-1}{2\sqrt{3}}\right)^2 + \left(\frac{-1}{2}\right)^2 + \\
 & \left(\frac{1}{2\sqrt{6}}\right)^2 + \left(\frac{1}{2\sqrt{2}}\right)^2 + \left(\frac{1}{2\sqrt{3}}\right)^2 + \left(\frac{1}{2}\right)^2 = 1
 \end{aligned}$$

The angle φ represents the relative phases which have interpretation in quantum mechanics. In our approximation model we use a Bloch circle rather than the all **Bloch sphere**, so, the angle φ is omitted from the formalism.

The **Qbit** representation becomes:

$$\begin{cases} |\psi\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \cos(\theta) \\ \sin(\theta) \end{pmatrix} \\ \theta \in [0, 2\pi] \end{cases} \quad (15)$$

We can compact the representation:

$$\begin{cases} |\psi\rangle = (\theta) \\ \theta \in [0, 2\pi] \end{cases} \quad (16)$$

Consequently the **QReg** with m **Qbits** should be written as:

$$\begin{cases} QReg = (\theta_1 \theta_2 \dots \theta_m) \\ \theta_i \in [0, \pi], i = 1 \dots m \end{cases} \quad (17)$$

5.3. Measurement:

The measurement is the process to extract information from Qbit, the information is “0” or “1”. The measurement process is probabilistic.

$$\left\{ \begin{array}{l} |\psi\rangle = (\theta) \\ \theta \in [0, 2\pi] \end{array} \right\} \xrightarrow{\text{Measurement}} \left\{ \begin{array}{ll} 0, & p(0) = \cos(\theta)^2 \\ "1", & p(0) = \sin(\theta)^2 \end{array} \right\} \quad (18)$$

5.4. Collapse process:

The collapse process destroys superposition of the state of the Qbit and projected its state either $|0\rangle$ to $|1\rangle$ or, regarding the results of the measurement being “0” or “1”.

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \xrightarrow{\text{Collapse}} \begin{cases} |\psi\rangle = |0\rangle \\ |\psi\rangle = |1\rangle \end{cases} \quad (19)$$

The collapse of a Qregister is well expressed in theta representation:

$$|QReg\rangle = (\theta_1 \theta_2 \dots \theta_m) \xrightarrow{\text{Collapse}} \begin{cases} |QReg\rangle = (0 \ 0 \dots 0) \\ |QReg\rangle = \left(\frac{\pi}{2} \ \frac{\pi}{2} \dots \frac{\pi}{2}\right) \end{cases} \quad (20)$$

Where we suppose that all the Qbits of the QReg collapse the same way.

6. Interference phenomenon, notion and example

Interference is a phenomenon which combines the probabilities of two or more Qbits or Qregisters. .we prefer explain it by an example:

$$\left\{ \begin{array}{l} |QReg_i\rangle = (\theta_{i1} \theta_{i2} \dots \theta_{im}) \\ |QReg_k\rangle = (\theta_{k1} \theta_{k2} \dots \theta_{km}) \end{array} \right\} \Rightarrow \text{the interference} \quad (21)$$

$$|QReg_i\rangle + |QReg_k\rangle = (\theta_{i1} + \theta_{k1}) \ (\theta_{i2} + \theta_{k2}) \dots (\theta_{im} + \theta_{km}) \quad (22)$$

6.1. Phase shifting:

We can shift the phases of each *Qbit* of the *Qregister*, it operates like following, where we consider a simple known shifting operation:

$$|QReg_i\rangle = (\theta_{i1}\theta_{i2} \dots \theta_{im}) \Rightarrow \\ New |QReg\psi_i\rangle = ((\theta_{i1} + \pi) (\theta_{i2} + \pi) \dots (\theta_{im} + \pi)) \quad (23)$$

7. Conclusion

Quantum mechanics is beautiful and powerful mathematical framework, which gives strong promises for the design of quantum algorithms. However, building quantum computer faces difficult challenges that are not resolved yet. So, the idea was to inspired much computational power as possible from quantum computing to design an algorithm that runs on conventional computers.

Evolutionary algorithms are natural candidates to handle the inspired techniques and concepts from quantum computing model. In this chapter we tried to introduce them just more enough to serve our aim.

The next chapter will present evolutionary algorithms and argue why they are suitable for this kind of hybridation to produce evolutionary quantum inspired algorithms.

Table of Contents

1. Introduction:.....	4
2. Q_algebra:	4
3. Qbit:.....	5
3.1. Matrix representation :.....	5
4. Quantum registers:	6
5. The classic model inspired by the quantum (CIQ):	6
5.1. the Qbit :	6
5.2. The Qreg:	7
5.3. Measurement:.....	10
5.4. Collapse process:.....	10
6. Interference phenomenon, notion and example	10

6.1. Phase shifting:	11
7. Conclusion	11

CHAPTER 2

QUANTUM INSPIRED EVOLUTIONARY ALGORITHM

1. Introduction :

Evolutionary algorithms are so suitable to be a framework for allowing techniques inspired from the quantum mechanics' mathematical model to apply. The reason is that, these kinds of algorithms are stochastic, based population, and they required no previous knowledge, alone the objective function influences the direct of search, and this one is totally free to define. All these properties made of them the best candidate ever for quantum inspired algorithm.

2. Evolutionary algorithms:

Different main schools of evolutionary algorithms have evolved during the last 40 years: genetic algorithms mainly developed in the USA by J. H. Holland [1], evolutionary strategies, developed in Germany by I. Rothenberg [4] and H.-P. Schwefel [3] and evolutionary programming [2]. Each of these constitutes a different approach; however, they are inspired by the same principles of natural evolution.

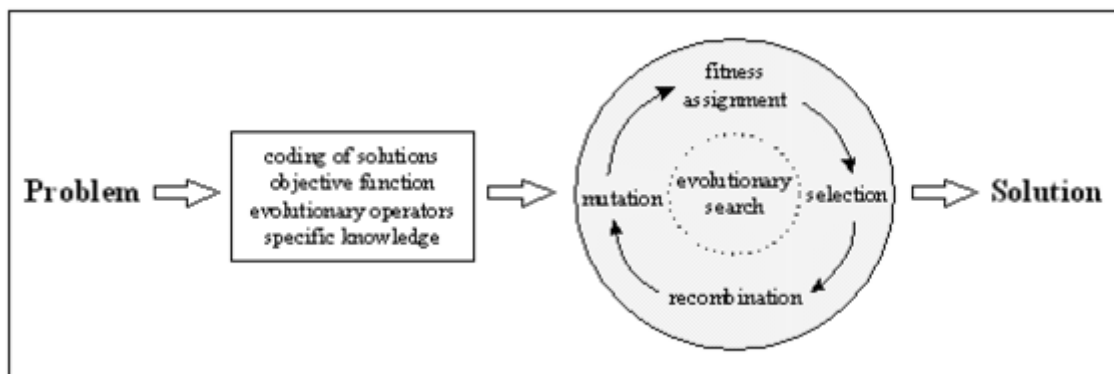


Figure 1 : Diagram of evolutionary algorithm

Evolutionary algorithms are stochastic search methods that mimic the metaphor of natural biological evolution. Evolutionary algorithms operate on a population of potential solutions applying the principle of survival of the fittest to produce better and better approximations to a solution. At each generation, a new set of approximations is created by the process of selecting individuals according to their level of fitness in the problem domain and breeding them together using operators borrowed from natural genetics. This process

leads to the evolution of populations of individuals that are better suited to their environment than the individuals that they were created from, just as in natural adaptation.

3. Evolutionary algorithms computational model:

Evolutionary algorithms model natural processes, such as selection, recombination, mutation, migration, locality and neighborhood. Figure *** shows the structure of a simple evolutionary algorithm. Evolutionary algorithms work on populations of individuals instead of single solutions. In this way the search is performed in a parallel manner.

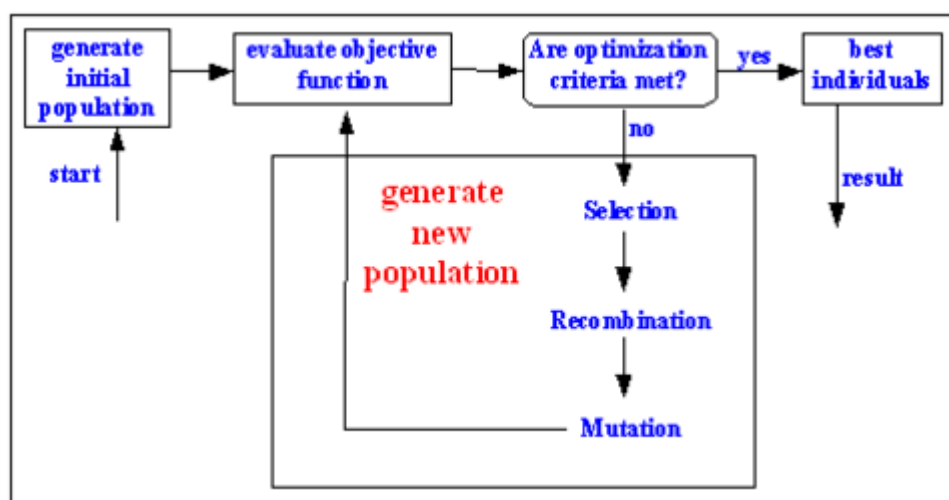


Figure 2 : Evolutionary algorithms computational model.

At the beginning of the computation a number of individuals (the population) are randomly initialized. The objective function is then evaluated for these individuals. The first/initial generation is produced.

If the optimization criteria are not met the creation of a new generation starts. Individuals are selected according to their fitness for the production of offspring. Parents are recombined to produce offspring. All offspring will be mutated with a certain probability. The fitness of the offspring is then computed. The offspring are inserted into the population replacing the parents, producing a new generation. This cycle is performed until the optimization criteria are reached. Evolutionary algorithm is powerful and performs well on a wide variety of problems.

4. Fundamental concepts of evolutionary algorithms:

Evolutionary algorithms differ substantially from more traditional search and optimization methods. The most significant differences are:

- Evolutionary algorithms search a population of points in parallel, not just a single point.
- Evolutionary algorithms do not require derivative information or other auxiliary knowledge; only the objective function and corresponding fitness levels influence the directions of search.
- Evolutionary algorithms use probabilistic transition rules, not deterministic ones.
- Evolutionary algorithms are generally more straightforward to apply, because no restrictions for the definition of the objective function exist.
- Evolutionary algorithms can provide a number of potential solutions to a given problem. The final choice is left to the user.

The following sections list some methods and operators of the main parts of Evolutionary Algorithms. A thorough explanation of the operators will be given in the following chapters.

4.1. Selection

Selection determines which individuals are chosen for mating (recombination) and how many offspring each selected individual produces. The first step is fitness assignment. The actual selection is performed in the next step. Parents are selected according to their fitness.

4.2. Recombination

Recombination produces new individuals in combining the information contained in the parents (parents - mating population). Depending on the representation of the variables of the individuals the adequate algorithms are applied.

4.3. Mutation

After recombination every offspring undergoes mutation. Offspring variables are mutated by small perturbations (size of the mutation step), with low probability. The representation of the variables determines the used algorithm.

4.4. Reinsertion

After producing offspring they must be inserted into the population. This is especially important, if less offspring are produced than the size of the original population. Another case is, when not all offspring are to be used at each generation or if more offspring are generated than needed. By a reinsertion scheme is determined which individuals should be inserted into the new population and which individuals of the population will be replaced by offspring.

5. Quantum inspired evolutionary algorithms concepts:

We have already argue why evolutionary algorithms are suitable to adapt principles from quantum mechanics to allow them being applied in the convention a computer. But, now we have to justify what computational power could we borrow from quantum mathematics to help us design powerful algorithm to solve hard and may be NP-Compleet problems.

5.1. Principle of superposition:

The principle of superposition is what makes quantum mechanics different and confines all it secrets and weirdness. Two things are inspired from this quantum fundamental principle.

5.2. Representation of a quantum entity:

A Qbit is represented as we've already seen, as a column vector. In reality it is a representation much interesting when a Qregister is consider. The Qregister is represented as follow: when we have given the example of:

$$QReg = \begin{pmatrix} 1/\sqrt{2} - 1/\sqrt{3} & 1/2 \\ -1/\sqrt{2} - \sqrt{2}/\sqrt{3} & \sqrt{3}/2 \end{pmatrix}$$

$$|QReg\rangle = \frac{-1}{2\sqrt{6}}|000\rangle - \frac{1}{2\sqrt{2}}|001\rangle - \frac{1}{2\sqrt{3}}|010\rangle - \frac{1}{2}|011\rangle \\ + \frac{1}{2\sqrt{6}}|100\rangle + \frac{1}{2\sqrt{2}}|101\rangle + \frac{1}{2\sqrt{3}}|110\rangle - \frac{1}{2}|111\rangle$$

This example shows us that we have confined the representation of 8 states (solutions) into a representation with only 6 real number. In general we have:

2^m states(numbers) is represented by a matrix of $2m$ numbers

This means that, we have jumped from an exponential number of parameters to linear one. This huge jump means that we have a representation that is potentially compresses an exponential number of parameters into only a linear number extracted from them, and the two sets represent the same information exactly. so, we can do computation on the two sets gives the same results, so, we prefer to work on the compact representation.

6. Quantum probability:

Quantum mechanics states that the probability is intrinsic to the nature and not arising from lack of information, this coincides with well with information theory which defines information upon only the notion of probability.

Quantum probability is more fundamental because it's computed as the square of the coordinates of the Qbit state in Hilbert space, this differs deeply from the classical probability, and the difference can be shown as the following equation:

$$\alpha^2 + \beta^2 = 1, \alpha, \beta \text{ coordinate of the Qbit in Hilbert space}$$

$$\alpha^2 : \text{probability of the event} \ \& \ \beta^2 : \text{probability of the } \neg\text{event}$$

$$p + q = 1, p : \text{probability of the event} \ \& \ q : \text{probability of the } \neg\text{event}$$

Quantum inspired evolutionary algorithms exploit this difference and it works. Because it gives us the possibility to manipulate discrete data (bit) in a continuous manner (θ – angle) of the Qbit.

7. Quantum Measurement:

Qbit offers us the continuous parameter (θ – angle) but in reality after all computations are done. We have to extract the discrete information “0” and “1”. This process

is taken and has been modeled as close as possible to the quantum one. To realize that, we use a function this generates numbers in a uniform random distribution. This procedure is as following:

```

Make( )
    c = rand( )
    if(c > sin(θ)2) x = 1
    else x = 0
end make
    
```

so formally : $make() : \theta \in R \mapsto \{0, 1\}$ [8]

The evolution operator is defined a rotation in space of Qbit as shown in figure .

$$\begin{pmatrix} \alpha' \\ \beta' \end{pmatrix} \equiv \begin{pmatrix} 1 & -\delta \\ \delta & 1 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix}, \quad \delta \ll \pi \text{ is the angle of rotation} \quad (5)$$

Where : $Q\text{-gate} = \begin{pmatrix} 1 & -\delta \\ \delta & 1 \end{pmatrix}$ is the evolution operator with a little angle δ .

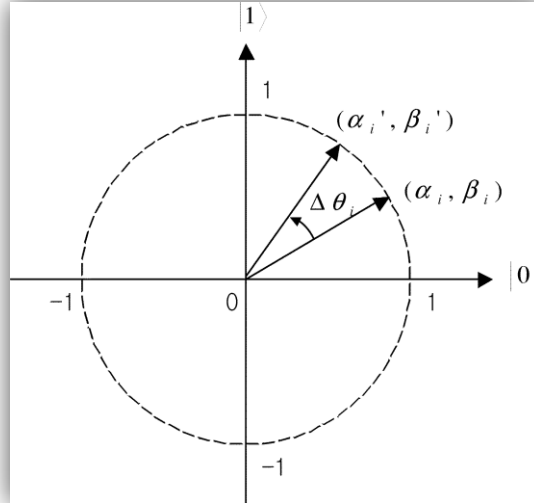


Figure 3 :Q-gate operation of evolution.

These are the ingredients to construct the evolutionary quantum inspired algorithms.

8. Evolutionary quantum algorithms:

Many evolutionary quantum inspired algorithms were proposed, each one of them exploits the ingredients cited above, and combined them with techniques of an evolutionary algorithm.

The well-known example is the Quantum Evolutionary Algorithm(**QEA**). It is a good algorithm to first introduction to evolutionary quantum inspired algorithms. We present it to get a deep understanding about the subject.

8.1. QEA algorithm:

Evolutionary Algorithm Inspired by Quantum is an algorithm that hybrid evolutionary approach to computation techniques inspired by the quantum. The idea was to import the maximum power supplied by the quantum representation, which provides a linear representation confining an exponential number of possibilities. This representation structure is copied by the matrix formalism $(2, m)$ where m is the number of Qbits in register m . Qreg. The evolution Qbit level is reformulated by angle θ rotation matrix that manipulates the relative phases of Qbit and therefore influenced the probability amplitudes that describe the contribution of both states base: $\{|0\rangle, |1\rangle\}$.

If the representation and evolution are inspired from the quantum mechanics mathematics, the objective of the development is not clearly assumed. To model the objective of the evolution, was used to evolutionary techniques based population and guided by a fitness function.

8.2. Concepts and procedures pivots of the algorithm:

In the following we will present the pivotal concepts and procedures used by the algorithm.

8.2.1. Population:

The population considered is actually a set of Qregisters of $m - Qbits$.

$$Qregister \equiv \sum_{i=0}^{m-1} \frac{1}{\sqrt{2^m}} |i\rangle \quad (1)$$

$$CIQbit \equiv \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \quad (2)$$

$$CIQ - formalism = \begin{pmatrix} \alpha_0 & \dots & \alpha_{m-1} \\ \beta_0 & \dots & \beta_{m-1} \end{pmatrix} \quad (4).$$

8.2.2.The Fitness:

The fitness is guide the evolution to a better solution, and so, it must be well specified. For the problem that we consider knapsack; the fitness is defined as the profit earned. [10]

$$f(x) = f(x_1, x_2, \dots, x_3) \quad (2)$$

8.2.3. Evolution:

The Evolution is made following an evolutionary vision, the strategy is to guide the algorithm by manipulating the relative phase of each Qbit of Qregister. The relative phase is described by the angle θ , so the evolution controls δ in the incrementing or decrementing it by very small value. The control of δ is performed on the basis of knowledge of the following:

$$\langle X: \text{current solution}; b: \text{best solution}; f(x) \geq f(b) \rangle$$

8.2.4. Measurement:

The measurement is the process to get the information “0” *or* “1” from the Qbit. And it is done as said above with the function *make()*. When measurement is done we have to be careful with the resulted value of the fitness.

8.2.5.The selection:

The selection is the best solution is performed at the same time defined as the evolution of the measurement as follows:

- Measure each CIQ-chromosome.
- Calculate the fitness of each chromosome CIQ-: Evaluate.
- Select the best of them: selection and storage.

9. the flowchart of the algorithm QEA:

The flowchart clearly shows the iterative sequence of two programming styles hybridized.

The style inspired by the quantum and evolutionary style. [3]

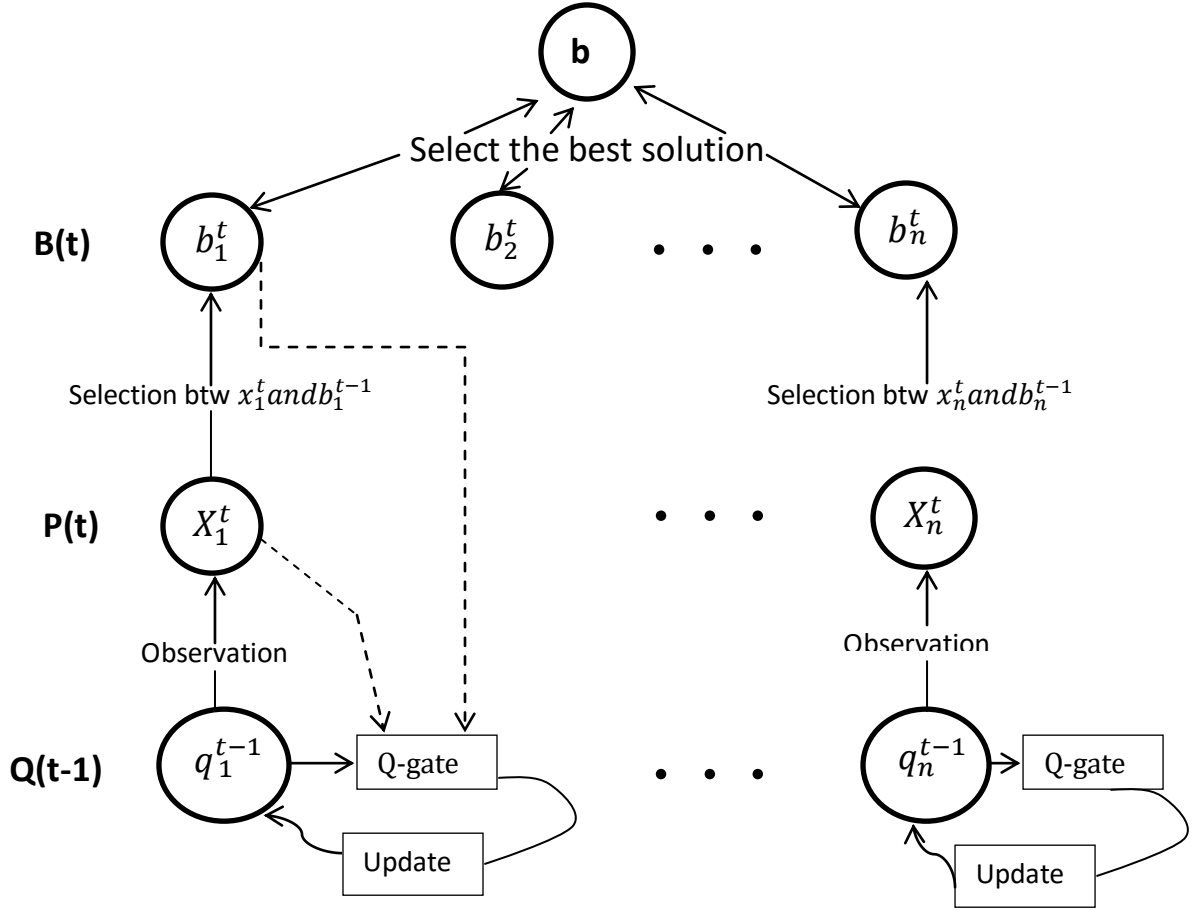


Figure 4 :Structure of QEA[5]

10.Description of the algorithm:

The procedure is QEA. Described as following: [3]

- i. The initial population $Q(0) = \{q_0^0, q_1^0, \dots, q_{n-1}^0\}$

Each **Qregister** $q_j^t, j = 0, 1, \dots, n - 1$. Is initialized with the following superposition:

$$q_j^0 \equiv \sum_{k=0}^{m-1} \frac{1}{\sqrt{2^m}} |X_k\rangle \quad (7)$$

- ii. The phase of the measure applied by **Make()**.
- iii. and produces solutions in: $P(0) = \{X_0^0, X_1^0, \dots, X_{n-1}^0\}$.
- iv. Evaluate the fitness of each solution : $X_j^0, j = 0, 1, \dots, n - 1$.
- v. Selection $B(0) = \{b_0^0, b_1^0, \dots, b_{n-1}^0\}$. We note that $B(0) = P(0)$.
- vi. **Loop while(t criterion number of maximum generation).**
 - a. the observations $P(t) \equiv \text{make}(Q(t - 1)) = \{X_0^t, X_1^t, \dots, X_{n-1}^t\}$.

- b. Evaluate the fitness value of each observation q_j^t is X_j^t , $j = 0, 1, \dots, n - 1$ in $P(t)$.
- c. Update the **Qregister** applying the evolution operator described above.
- d. Select the best solutions as follows:

$$B(t) = \text{meilleur de } P(t) \text{ et } B(t - 1).$$

- e. **Loop.**[6]

11. Conclusion :

We have discussed the evolutionary algorithms and argued their natural adequacy to work with the concepts and techniques of quantum computing. also, we have presented the main idea through the discussion of the Quantum Evolutionary algorithm (QEA). next chapter, will be dedicated, to our proposed algorithm.

CHAPTER 3

QUANTUM INSPIRED GENETIC ALGORITHM

1. Introduction :

We seek a novel algorithm to solve a class of combinatorial problems and for this purpose, so we applied our approaches to one famous problem among them which is the 0-1 Knapsack problem, we're going to define it and discuss its complexity.

2. Basic notion of optimization:

We distinguish two types of optimization problems: minimization problems and maximization problems. An optimization problem (minimization or maximization) is defined as a set of data and a set of constraints. A set of solutions is associated to this optimization problem. among this solutions, there are a subset which represent a realizable solutions which respect the constraints, to each solution it is associated to a value representing its quality. the resolution of optimization problem consists to find a solution which minimize or maximize this value.

3. The complexity theory:

The complexity theory try to evaluate the difficulty of problems via the study of the propose algorithm. it associates a complexity function to each algorithm and measure the resources needed to the resolution of to a given problem. indeed, it classifies the set of problems according to two criteria which are time needed for computation and the required space memory. it classifies the problems according to their complexity in two principal classes: the class P (polynomial time) and the class NP (non-deterministic polynomial time). otherwise, it divides the problems of the NP class into subclasses: NP-complete and NP-hard.

4. Class P problems :

this class of problems could be resolved by the deterministic Turing machine in polynomial time of computation comparing to the size of the problem instance.

5. The class NP problems:

NP regroups the set of problems that could be resolve by a non-deterministic machine and admits a polynomial algorithm to test the validity of the considered problem solution.

6. The class NP-Complete problems:

NP-Complete regroups decision problems which it does not exist an algorithm that allowed a their resolution in polynomial time .the first problem being shown that it is a NP problem ,was the binary logic formula satisfiability.

7. The NP-hard problems:

The class **NP-hard** contains the decision and optimization problems, to prove that a given problem of optimization is NP-hard, it suffices to demonstrate that its associate decision problem is NP-Complete.

NP-complete and hard problems are two difficult problems that none polynomial algorithm allowing their resolution has been found. Indeed, all proposed algorithms in literature are all of exponential complexity or worse. The different between the two problems is in the kind of the expected response .the NP-Complete are decisional problems where the expected response are whither yes or no. such as the cycle Hamiltonian problem the NP-Hard problems concern much more those of optimization where we seek the optimal solution such as the travel salesman problem.[7].

8. The knapsack problem:

Suppose we are planning a hiking trip, and we are, therefore, interested in filling a knapsack with items that are considered necessary for the trip. There are *m* different items types that are deemed desirable; these could include bottle of water, apple, orange, sandwich, and so forth. Each item type has a given set of two attributes, namely a weight (or volume) and a value that quantifies the level of importance associated with each unit of that type of item. Since the knapsack has a limited weight (or volume) capacity, the problem of interest is to figure out how to load the knapsack with a combination of units of the specified types of items that yields the greatest total value. What we have just described is called the Knapsack problem.

A large variety of resource allocation problems can be cast in the framework of a knapsack problem. The general idea is to think of the capacity of the knapsack as the available amount of a resource and the item types as activities to which this resource can be allocated. Two quick examples are the allocation of an advertising budget to the promotions of individual products and the allocation of your effort to the preparation of final exams in different subjects.

9. Formal definition of the 0-1 Knapsack problem:

The 0-1 Knapsack problem is the simple variance of the Knapsack problem, where the items has to be selected only one no more, if it is. Formally it is defined as follows:

$$\left\{ \begin{array}{l} \text{Maximize : } \sum_{j=1}^m x_j * p_j \\ \text{Constraint : } \sum_{j=1}^m x_j * w_j \leq Cap \end{array} \right.$$

Where: $x_j \in \{0, 1\}; j = 1 \dots m$; m number of items

Cap : Capacity of the napsack

p_j : Profit of the item j

w_j : Weight of item j

For our model we suppose that the profit is proportional to the weight; and so we can formally assume that the profit and the weight coincide. This assumption is done by researchers to focus on the algorithm mechanism first.

10.The proposed Algorithm *QIGA* :

We proposed a Genetic Quantum Inspired Algorithm to solve a combinatorial optimization problem. As an example we take the Knapsack problem. So we are going to describe the data structure used, and the operations defined on it.

10.1. QGene:

The *QGene* is the *Qbit* which is a *matrix 2 * 1* containing the probability amplitudes of Classical *QGene* :

$$QGene \equiv Qbit$$

10.2. CGene :

is the conventional bit which could take the value “0” or “1” depending on the probabilities which are the squares of the amplitudes defined in *ne* :

$$CGene \equiv bit$$

10.3. QGenotype:

QGenotype is a tuple of m **Qbits**, so, simply the **QGenotype is a QReg** :

$$QGenotype \equiv QReg$$

10.4. Cgenotype :

Cgenotype is a tuple of m **bits** : so, simply the **CGenotype is a Register**

$$CGenotype \equiv m - bits$$

10.5. QPhenotype:

QPhenotype of a **QGenotype** is the amplitudes of probabilities that the register had to get classical informations, which at the end are the classical phenotype **CPhenotype**.

10.6. CPhenotype :

CPhenotype is the information containing in the **CGenotype** which they are the values “0” and “1” that could be measured with the probabilities computed as square of the amplitudes of probabilities defined in the **QPhenotype**.

11.A genetic population:

A genetic population is a set of **Qgenotypes** :

$$Population \equiv \{n \text{ of } QGenotypes\}$$

After the setting of the data structure of the proposed Genetic Quantum inspired Algorithm **GIQA**. So, we proceed to define the adequate operation on the algorithms:

11.1. Initial population:

The population is n **QGenotypes**, and it is generated randomly, traditionally the **QGenotypes** are generated with equal probabilities.

11.2. Evaluation:

To evaluate a **QGenotype** we had to measure it, so, transformed it into a **CGenotype** and after that extracting the information containing in this last one. The measurement is a stochastic process realized with the function **make()** defined earlier in chapter 02. We can describe the whole process as follows :

$$\text{Measurement} : \text{QGenotype} \mapsto \text{CGenotype}$$

$$\text{make}(\): (\theta_1 \theta_2 \dots \theta_m) \mapsto (x_1 x_2 \dots x_m)$$

Now, we can compute the fitness of the **QGenotype** which for our case of study of Knapsack problem is the profit:

$$\text{Fitness}(\text{QGenotype}) = \sum_{j=1}^m x_j w_j$$

11.3. Selection:

After the evaluation has been done, we sorted the **QGenotype** according to the best fitness recorded for each one of them. And then we selected the best ones two by two successively to realize the recombination or the crossover operation which models the interference. To realize the mutation we select randomly an **QGenotype** from the whole resulted set containing parents and offsprings.

11.4. Recombination (Crossover):

After selecting two **QGenotypes** to be parents and reproduce two other offspring, we applied the following crossover operation which we defined as follows:

Let the selected parent being:

$$\begin{cases} |QReg_l\rangle = (\theta_{l1} \theta_{l2} \dots \theta_{lm}) \\ |QReg_k\rangle = (\theta_{k1} \theta_{k2} \dots \theta_{km}) \end{cases} ; l, k = 1 \dots n$$

To create more diversity in our algorithm we introduce a random number generating between (0, 1).

$$\begin{cases} c = rand() \\ |QReg_{l'}\rangle = c * |QReg_l\rangle + (1 - c) * |QReg_k\rangle \\ |QReg_{k'}\rangle = (1 - c) * |QReg_l\rangle + c * |QReg_k\rangle \end{cases}$$

Of course, we should remark that our crossover operation is applied on the **QPhenotypes** of the **QGenotypes** and not on it directly, this is because we are seeking interference effect.

$$\begin{cases} c = rand() \\ |QReg_{l'}\rangle = ((c * \theta_{l1} + (1 - c) * \theta_{k1})(c * \theta_{l2} + (1 - c) * \theta_{k2}) \dots (c * \theta_{lm} + (1 - c) * \theta_{km})) \\ |QReg_{k'}\rangle = (((1 - c) * \theta_{l1} + c * \theta_{k1})((1 - c) * \theta_{l2} + c * \theta_{k2}) \dots ((1 - c) * \theta_{lm} + c * \theta_{km})) \end{cases}$$

11.5. Mutation:

We select random **QGenotypes** from the result population of parents and offsprings with certain low probability, and we applied the mutation which models the phase shifting as follows:

Let the selected QGenotype be : $|QReg_k\rangle = (\theta_{k1}\theta_{k2} \dots \theta_{km})$

The mutation operator had the following effect:

$$|QReg_{k'}\rangle = ((\theta_{k1} + \pi)(\theta_{k2} + \pi) \dots (\theta_{km} + \pi))$$

12.Collapse process:

Collapse process is the heart of our algorithm, it is the process which gives sense and success to our proposed **GIQA**. It is applied when a constraint of the consider problem is violated, in our case study the constraint of the Knapsack that had to be respected and not at all violated is the respect of the capacity of the Knapsack :

$$\sum_{j=1}^m x_j w_j \leq Capacity$$

So, when the constraint is violated, we applied the collapse process to collapse the state (*QGenotype*) to the specific *Zero QGenotype* $\equiv (0, 0, \dots, 0)$, which its measurement generate certainly the Classical Genotype : *CGenotype* $\equiv (0, 0, \dots, 0)$, and so, the fitness is reduce to zero. We can formalize this like following:

Collapse process : QGenotype $\mapsto$ Zero QGenotype

$$: (\theta_1 \theta_2 \dots \theta_m) \mapsto (0, 0, \dots, 0)$$

This was the ingredient of our proposed Genetic Inspired Quantum Algorithm **A**.

13.Flowchart of **A** :

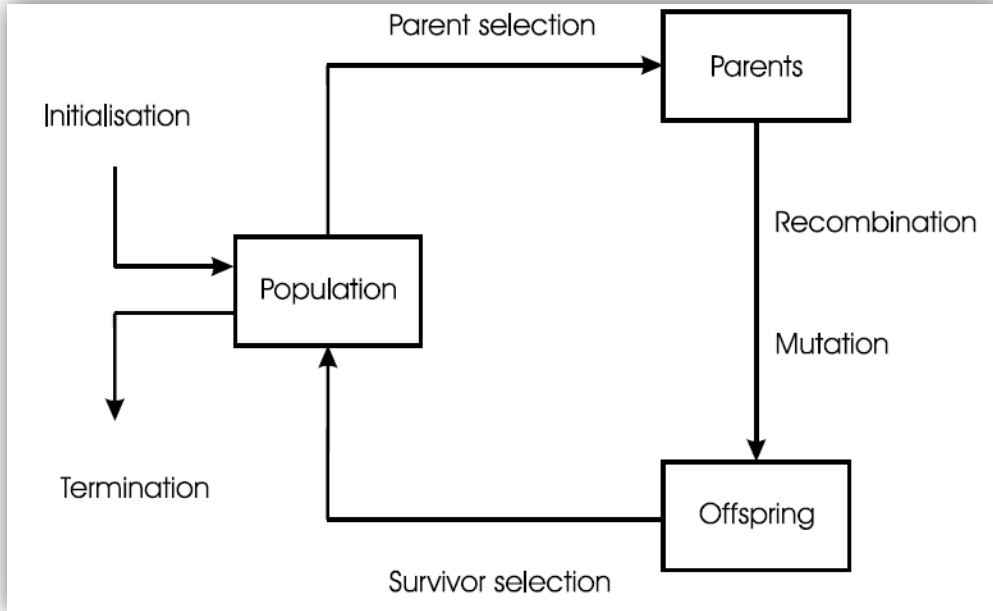


Figure 1: The general scheme of an Evolutionary Algorithm.[6]

14.Conclusion:

In this chapter we have specified our Genetic Inspired Quantum Algorithms **QIGA**, next we are going to investigate its validity and performance with a series of experiments, and to do this we applied it on a well-known NP-Hard problem with is the 0-1 Knapsack problem.

CHAPTER 4

ANALYSE

1. Introduction :

In this work we have proposed an approach, and in order to valid its performance we precede to some experiments. First, we test the value and impact of the novel idea which the heart of our proposed algorithm **GIQA** . Secondly, we experiments the influence of interference modeled by crossover operation on the algorithms. After that, we do the same analyze with the phase shifting. These two experiments are runs 40 executions, to allow a statistics analyze. We also, experiment the behavior of the algorithm over one execution and 1000 iterations. Finally, we explore the behavior of the Qbit over execution, to see how in the basic level of the particle structure how things change.

2. Programming language:

2.1. Fortran :

Fortran (formerly FORTRAN, derived from "Formula Translation"^[1]) is a general-purpose, imperative programming language that is especially suited to numeric computation and scientific computing. Originally developed by IBM^[2] in the 1950s for scientific and engineering applications, Fortran came to dominate this area of programming early on and has been in continuous use for over half a century in computationally intensive areas such as numerical weather prediction, finite element analysis, computational fluid dynamics, computational physics and computational chemistry. It is a popular language for high-performance computing^[3] and is used for programs that benchmark and rank the world's fastest supercomputers.

Fortran95 is a version of the Fortran programming language, however, users have wanted to add over time the interesting ideas used in other languages. This gave the versions Fortran 90 and Fortran 95.

3. Used software:

3.1. Code::Blocks :

Code :: Blocks is a free, opensource crossplatform IDE that supports multiple compilers including GCC, Clang and Visual C++. It is developed in C++ using wx Widgets as the GUI toolkit. Using a plugin architecture, its capabilities and features are defined by the

provided plugins. Currently, Code::Blocks is oriented towards C, C++, and Fortran. It has a custom build system and optional Make support.

Code::Blocks is being developed for Windows, Linux, and Mac OS X and has been ported to FreeBSD,[2] Open BSD[3] and Solaris.[4]

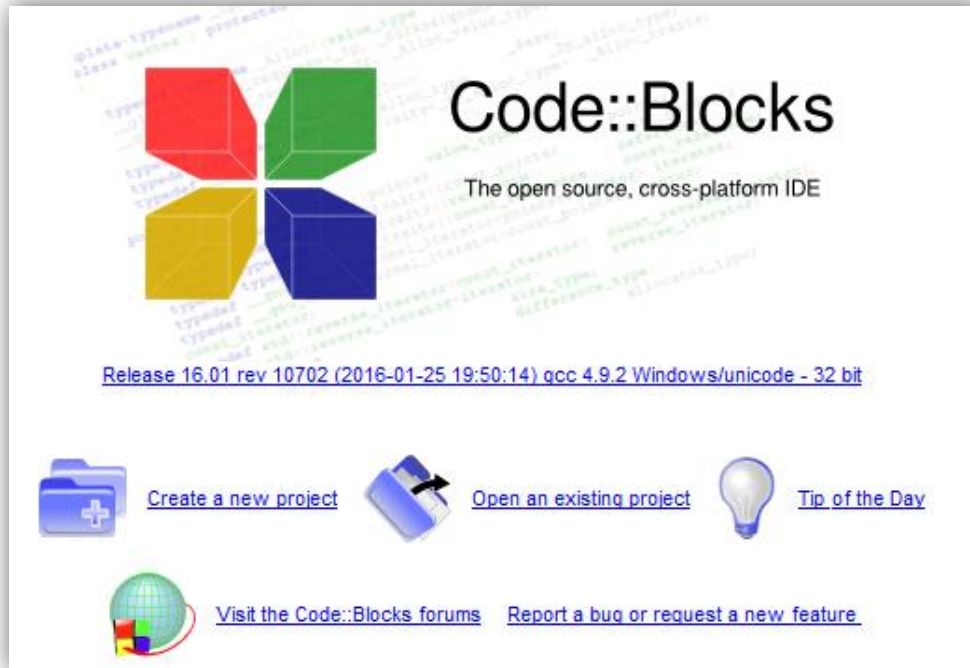


Figure 1code::blocks interface

3.2. Gnuplot :

gnuplot is a command-line program that can generate two- and three-dimensional plots of functions, data, and data fits. It is frequently used for publication-quality graphics as well as education. The program runs on all major computers and operating systems (GNU/Linux, Unix, Microsoft Windows, Mac OS X, and others). It is a program with a fairly long history, dating back to 1986. Despite its name, this software is not distributed under the GNU General Public License (GPL), but its own free software license.^[5]

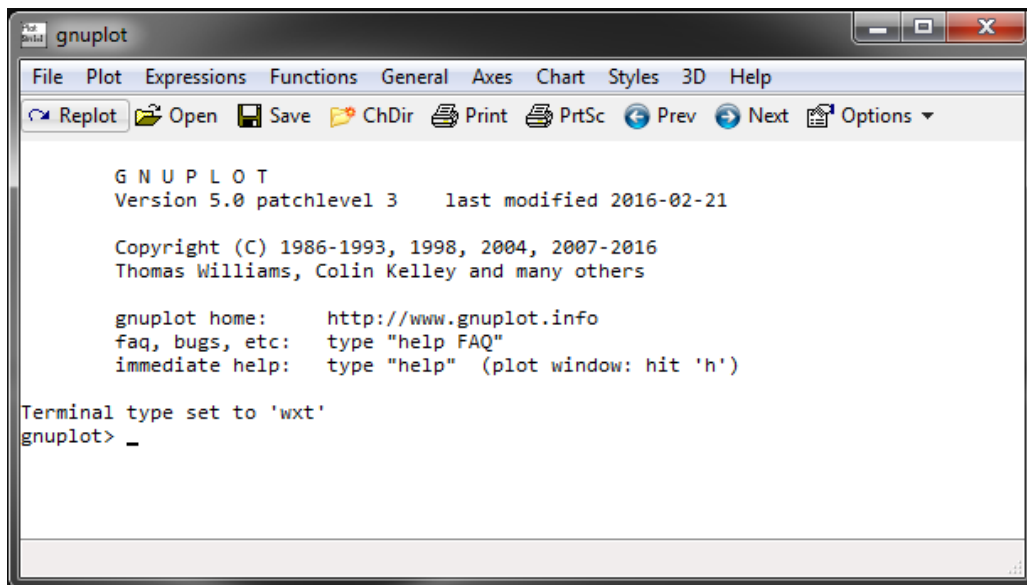


Figure 2:Gnuplot interface

3.2.1. Example command line:

```

gnuplot> set terminal png
Terminal type set to 'png'
Options are 'small color'
gnuplot> set output "exemple1.png"
gnuplot> plot [t=-4:4] sin(t)
Example command line:
    
```

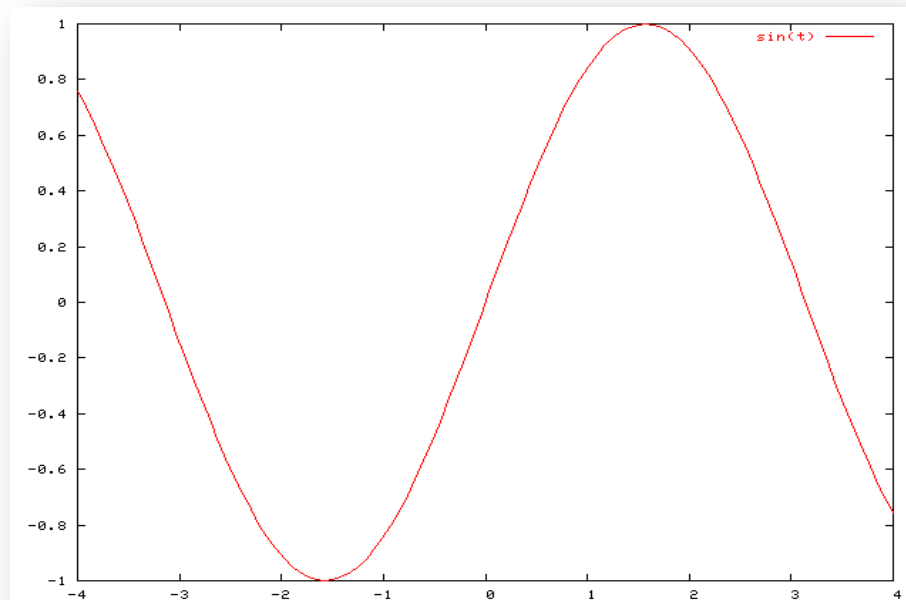


Figure 3Example command line

3.2.2. Programming and application interfaces:

gnuplot can be used from various programming languages to graph data, including Perl (via CPAN), Python (via Gnuplotpy and SAGE), Julia (via Gaston.jl), Java (via JavaGnuplotHybrid and jgnuplot), Ruby (via RubyGnuplot), Ch (via ChGnuplot), Haskell (via a Haskell gnuplot), Fortran 95,^[6] and Smalltalk (Squeak and GNU Smalltalk).

gnuplot also supports piping, which is typical of scripts.^[7] For script-driven graphics, gnuplot is by far the most popular program.

4. Collapse process:

Our genetic algorithm **GIQA** introduces a novel idea which the golden stone of its success, this is the collapse process. To show its value and impact into the success of our algorithm we proceed to do two experiments, the first we launch the **GIQA** without using the collapse process, and in the second experiment we exploit it.

4.1. Experiment without using collapse process:

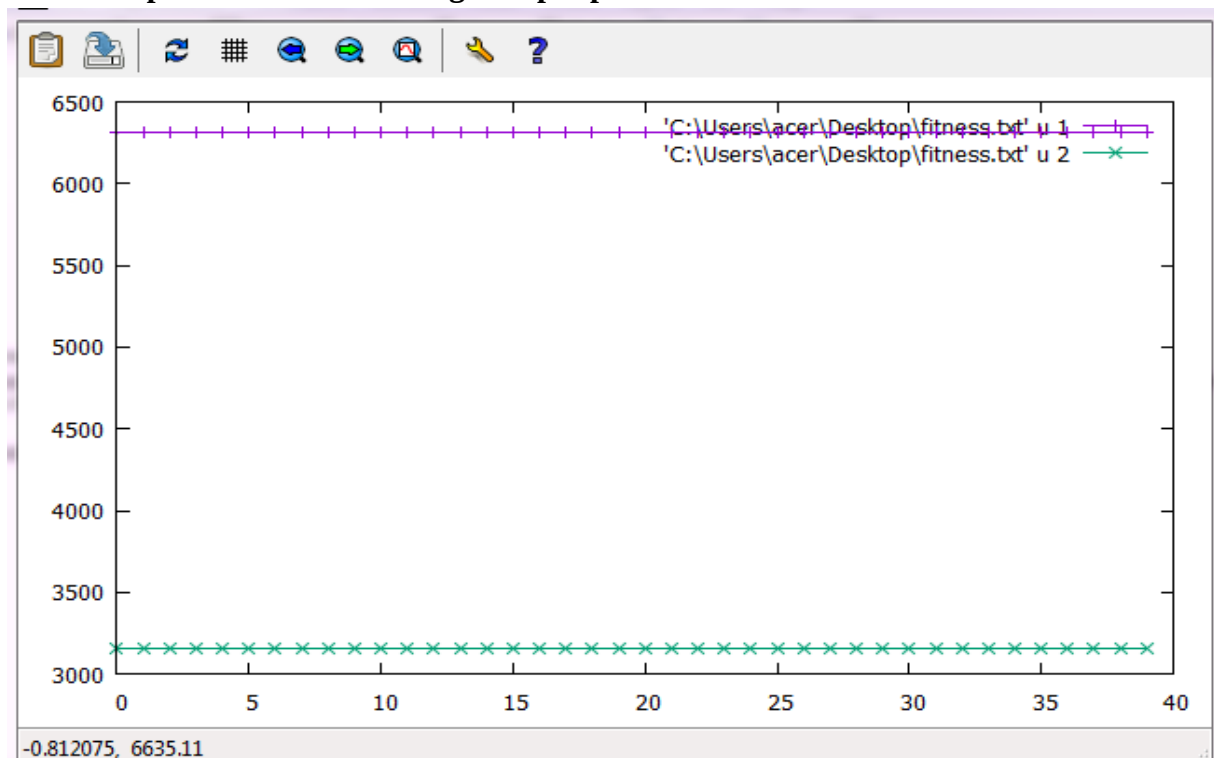


Figure 4<m100n100without collapse process>

The algorithm behaves strangely the same for all 40 executions; and it stabilizes for the double value of profit ($2 * cap$) which means for our choice of the capacity witch is the

half of the whole weight: $cap = \frac{1}{2} wholeweight$. This is explained by the fact that all classical genes (Cbit) of the classical genotype (CGenotype) are measured to the value 1, which means all the items are selected. It seems that the random initialization does favorite the value "1". We conclude that the constructive interference had all chances to happened, rather than, destroyed ones which don't have that chance to be applied that much.

In order to remedy at this situation we introduced a process inspired from quantum mechanics called collapse process, it's when we projected the state of Qbit, as example, to one of its computational basis $|0\rangle$ or $|1\rangle$, and destroys their superposition. The collapse process could formalized as follow:

$$where(fitness > capacity)|Qbit\rangle \xrightarrow{Collapses} |0\rangle$$

4.2. Experiment using collapse process:

The presents of the collapse function gives sense to and power to the proposed algorithm **GIQA** to reach the full profit (capacity) of the knapsack. The graph below shows well this fact, where the computed fitness is around (3154.46168 ± 0.12134) , and it reaches the full profit (full capacity) of the knapsack ($profit = 3154.5$) in 20 executions over 40 which is half of the executions, the remain half is in the σ scope, except for one execution where the algorithm gives $profit = 3153.72$ which remains very high accuracy. This is more reflected by the fact that: $maxprofit = 3154.5 \in 3154.46168 \pm 0.12134$, and also, we have to remark that : $maxprofit - averageprofit = 0.03832 \ll \sigma$, which means that, when the standard deviation is smallest, the average fitness value tend to equal the maximum solution ($maxprofit, capacity$). All this show a great performance of the algorithm.

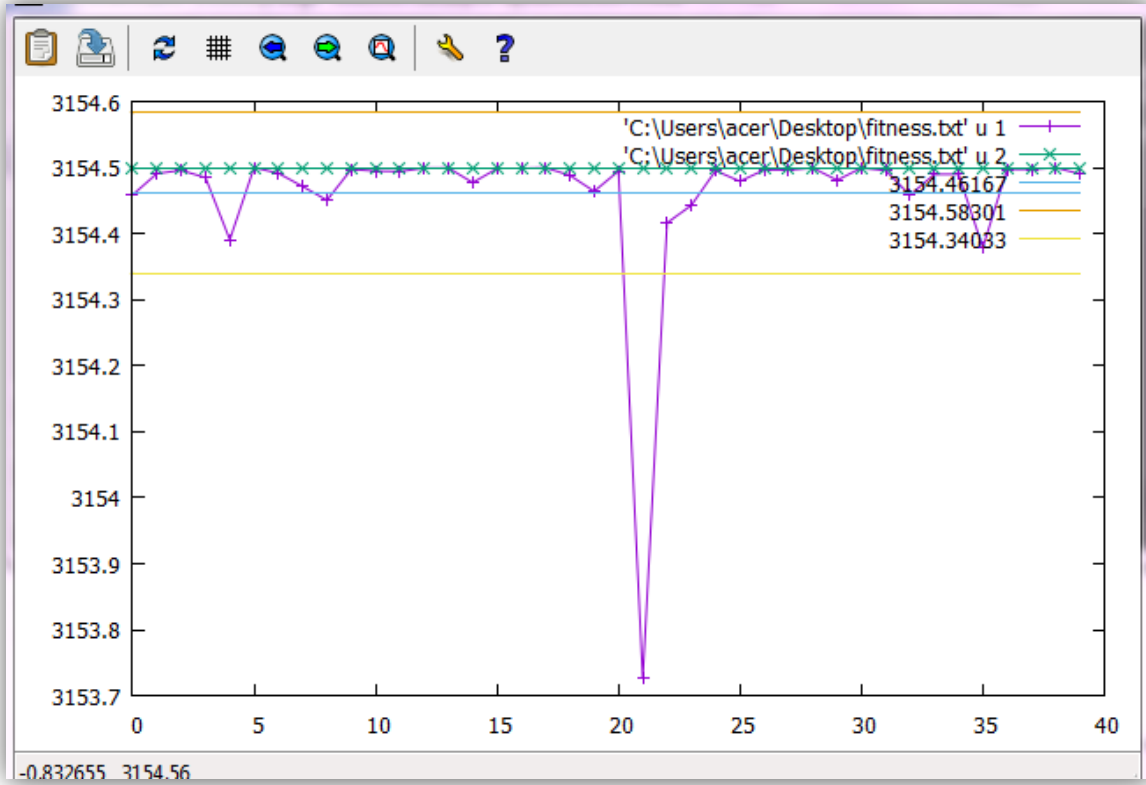


Figure 5:<m=100 ; n=100 ; pcross =1.0 ; pmut =0 ; maxgen = 1000 ; exe = 40>

$$\lim_{\sigma \rightarrow 0} avgfitness \approx Solution$$

5. Analyze the effect of the interference reflected by crossover operation on the algorithm performance:

In order to show the effect of interference on our proposed algorithm *IQA*, we proceed gradually to intensify the crossover operation probability which modeled interference:

(*pcross* = 0.05, 0.125, 0.25, 0.375, 0.5, 0.5, 0.625, 0.75, 0.825, 1.0,) and to really deal with only the effect of interference we isolated from the contribution of mutation to the algorithm performance, so we illuminate it (*pmut* = 0.0). We generate graphs representing these experiments to better analyze and interpret the results.

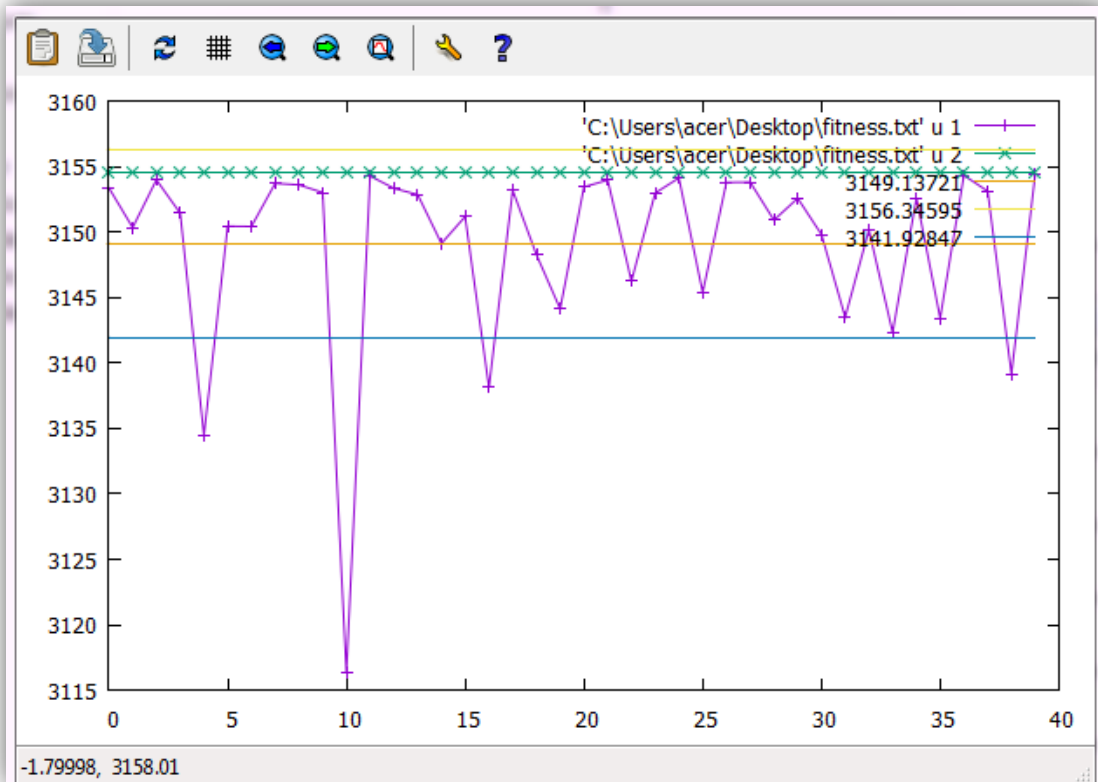


Figure 6:<m=100 ; n=100 ; pcross =0.0 ; pmul = 0.0 ; maxgen=1000 ; exe = 40 >

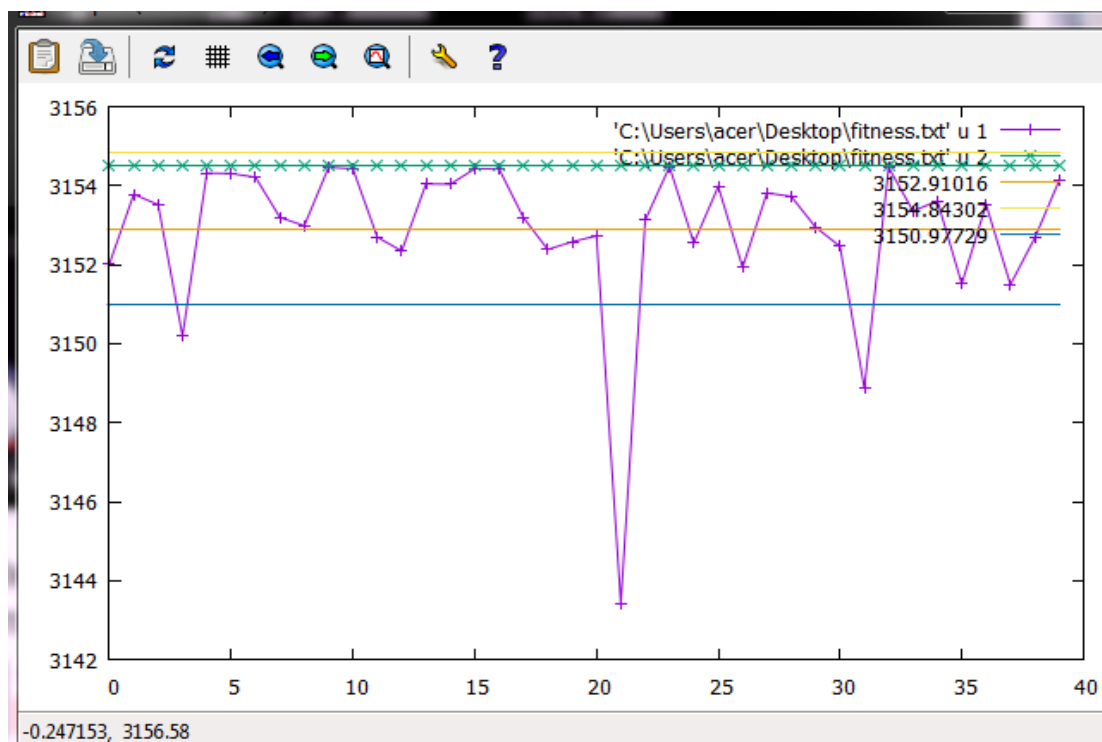


Figure 7:<m=100 ; n=100 ; pcross =0.05 ; pmul = 0.0 ; maxgen=1000 ; exe = 40 >

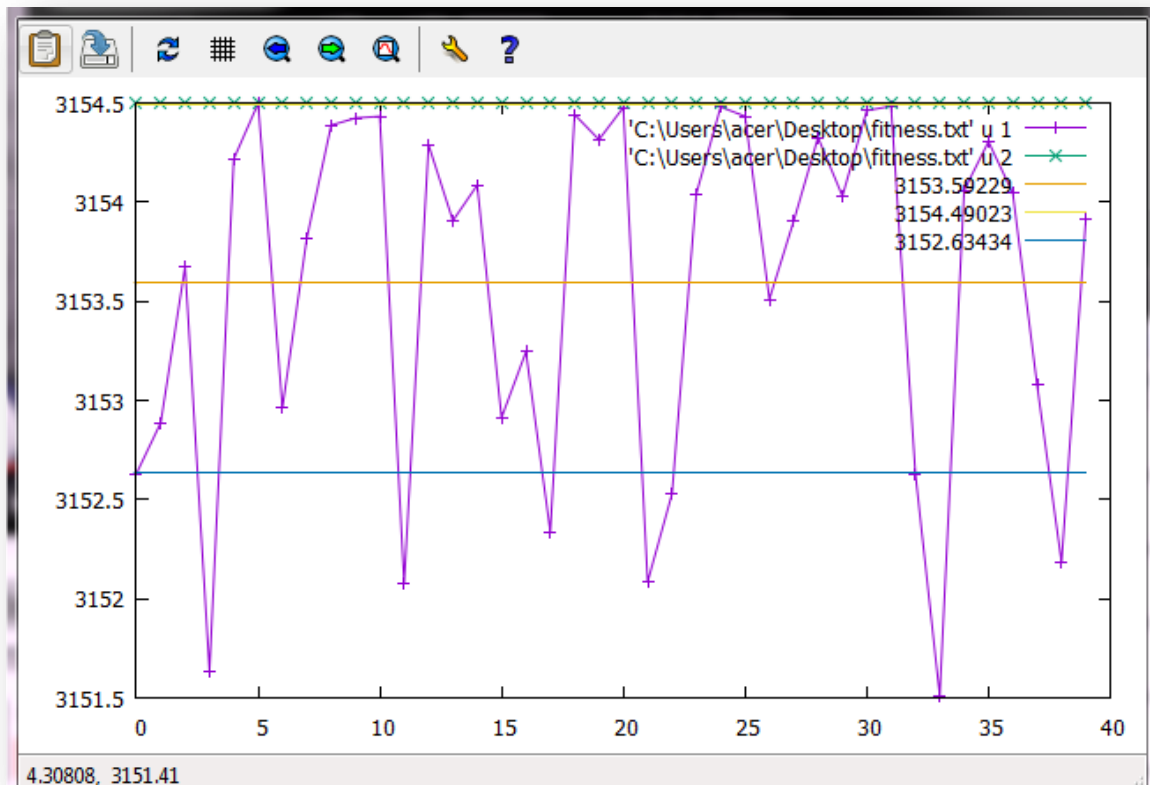


Figure 8:<m=100 ; n=100 ; pcross =0.125 ; pmut = 0.0 ; maxgen=1000 ; exe = 40 >

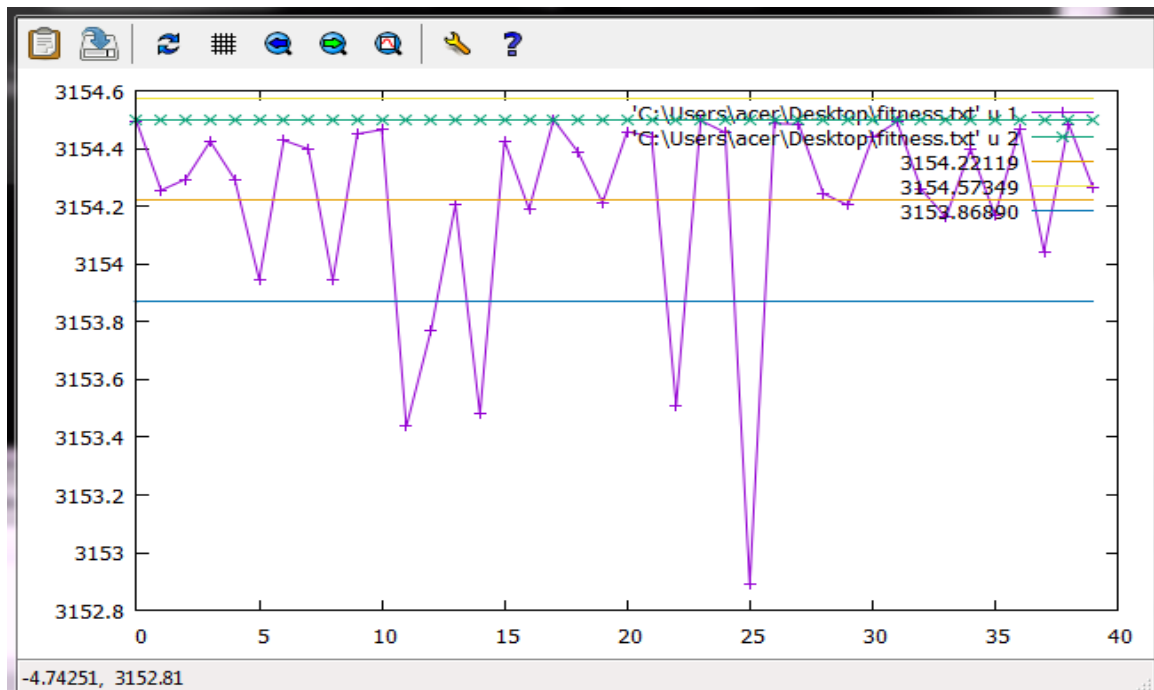


Figure 9:<m=100 ; n=100 ; pcross =0.25 ; pmut = 0.0 ; maxgen=1000 ; exe = 40 >

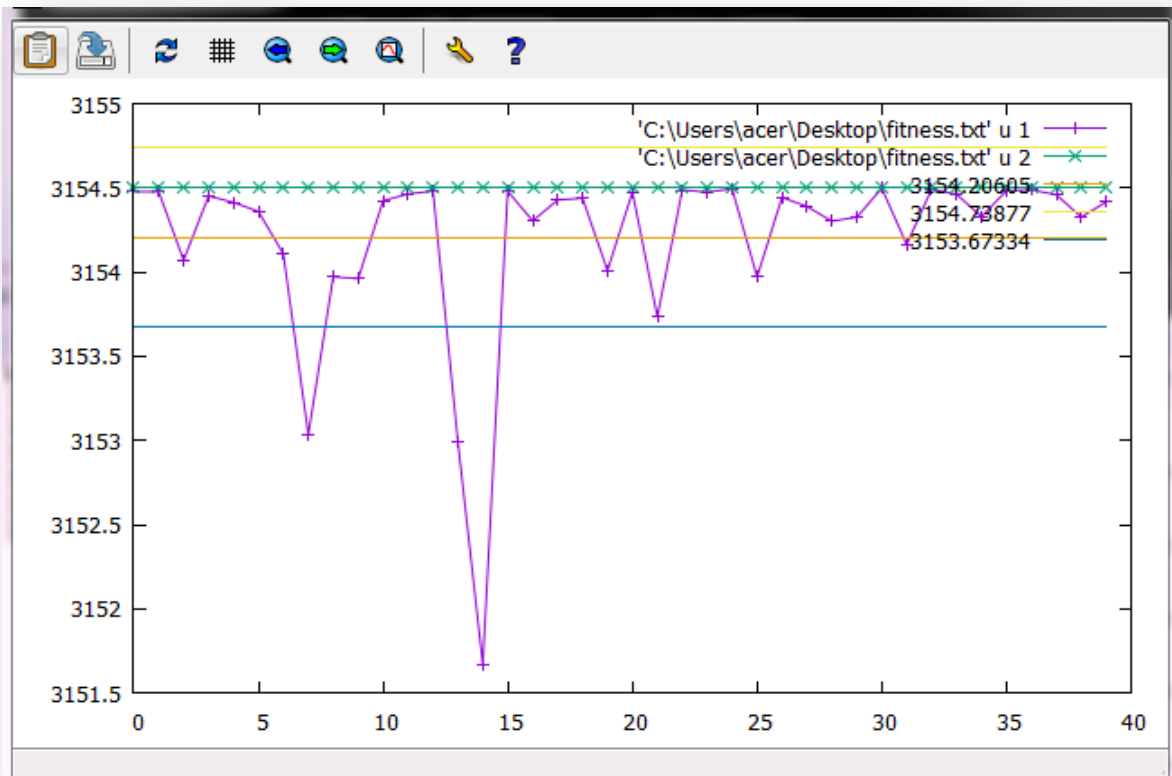


Figure 10:<m=100 ; n=100 ; pcross =0.375 ; pmul = 0.0 ; maxgen=1000 ; exe = 40 >

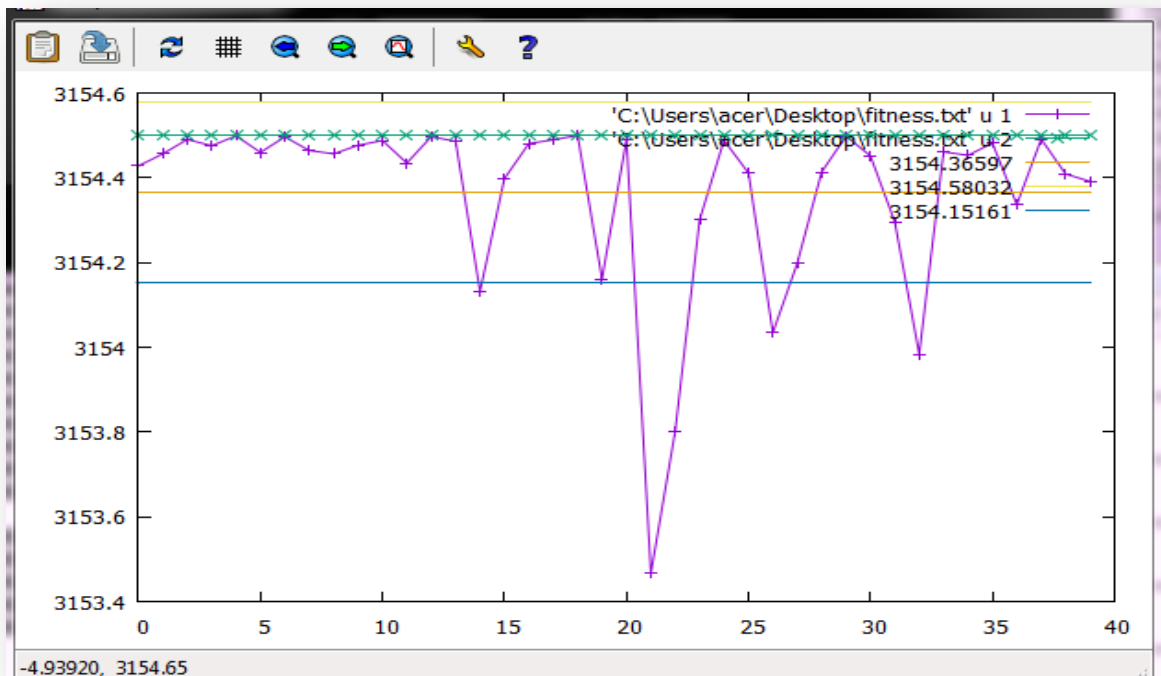


Figure 11:<m=100 ; n=100 ; pcross =0.5 ; pmul = 0.0 ; maxgen=1000 ; exe = 40 >

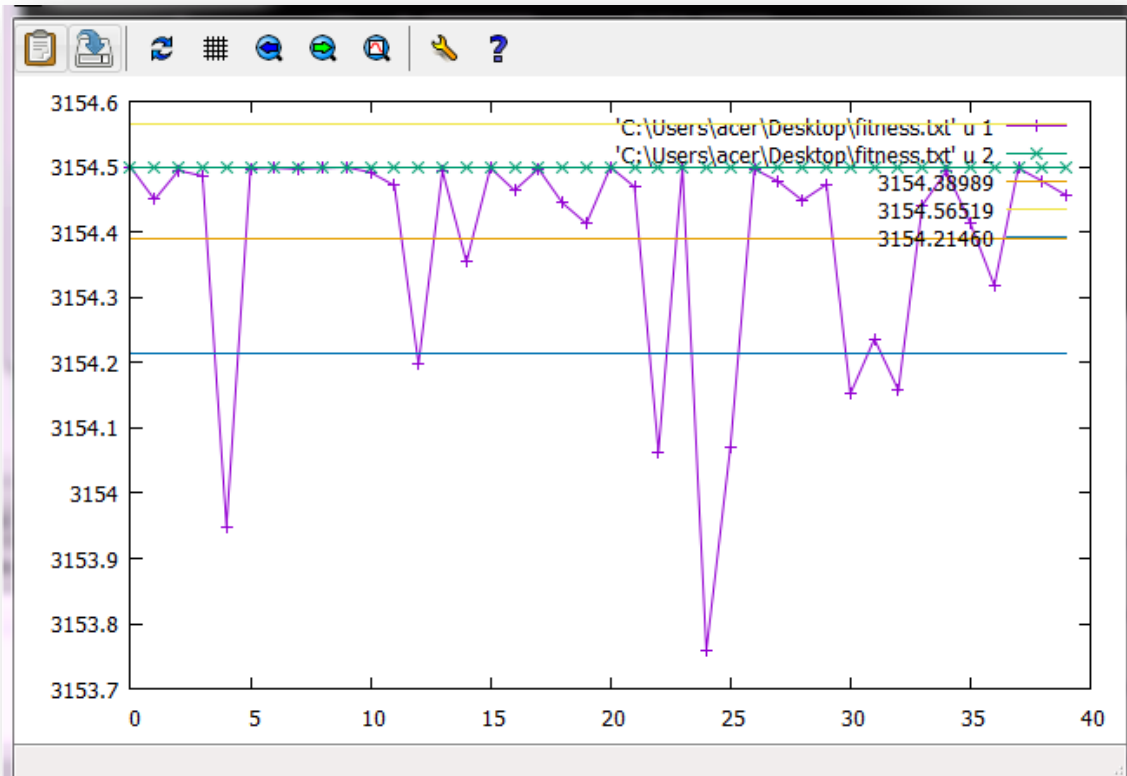


Figure 12 :<m=100 ; n=100 ; pcross =0.625 ; pmut = 0.0 ; maxgen=1000 ; exe = 40 >

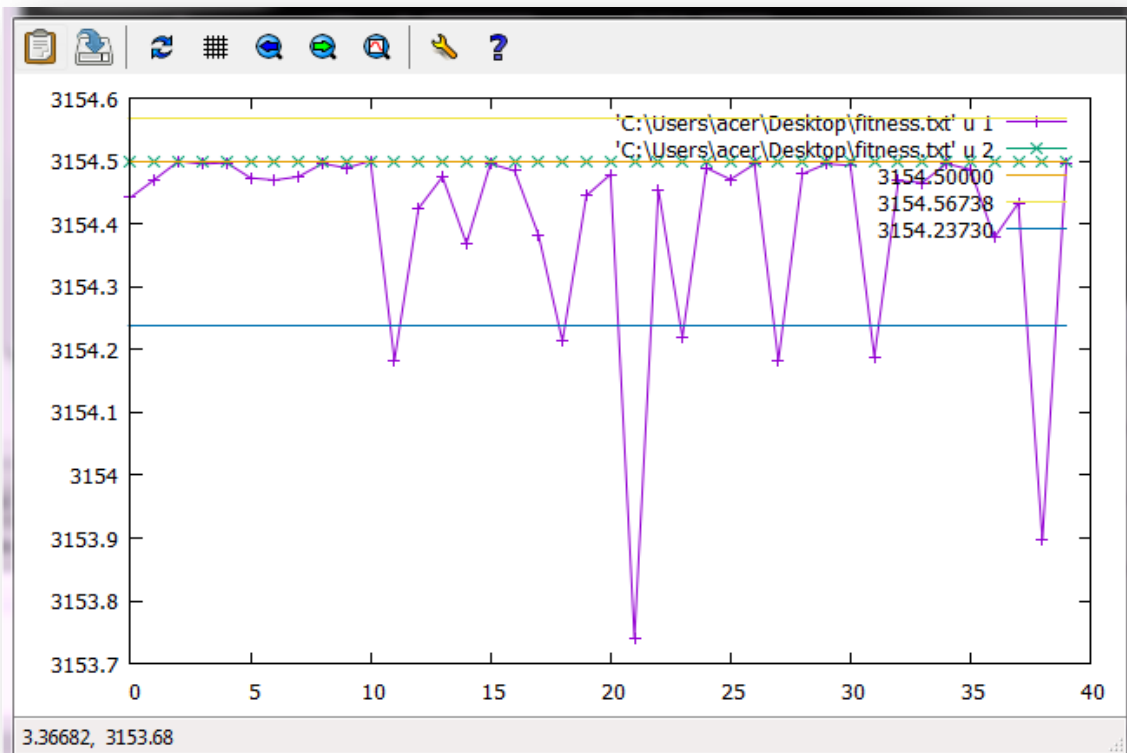


Figure 13 :<m=100 ; n=100 ; pcross =0.75 ; pmut = 0.0 ; maxgen=1000 ; exe = 40 >

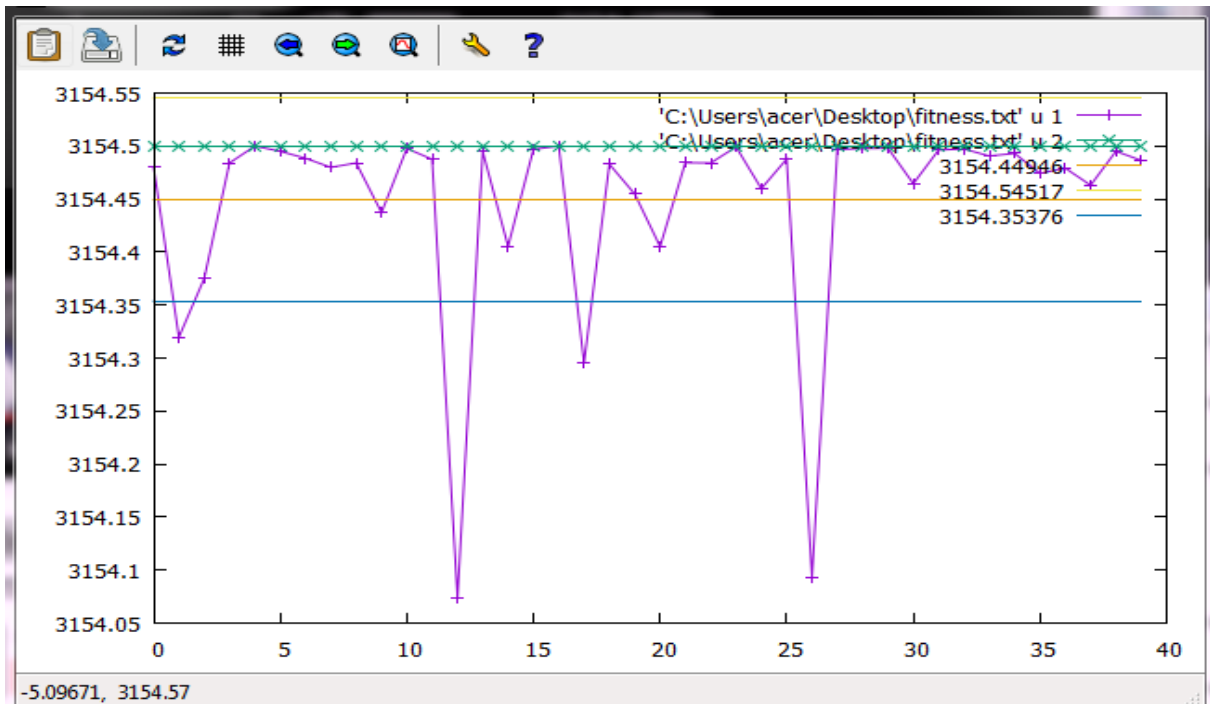


Figure 14:<m=100 ; n=100 ; pcross =0.875 ; pmul = 0.0 ; maxgen=1000 ; exe = 40 >

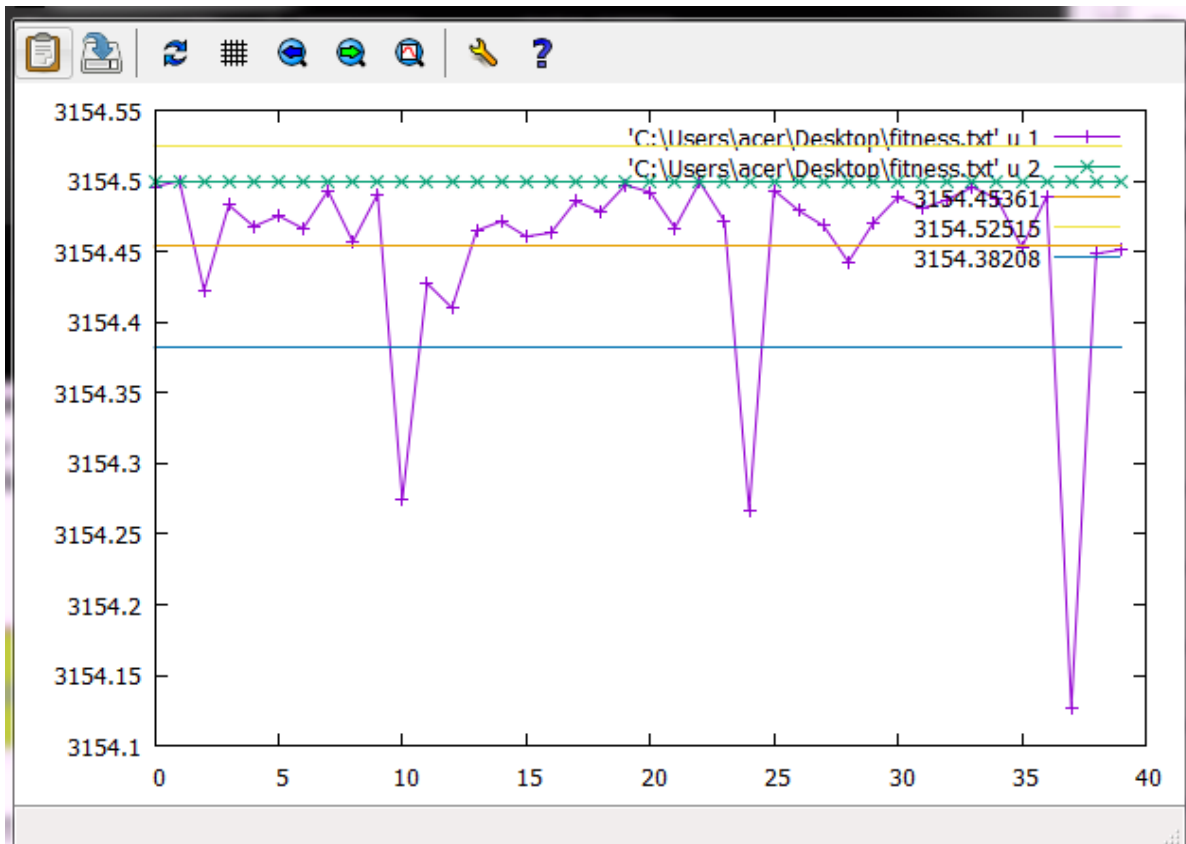


Figure 15:<m=100 ; n=100 ; pcross =1.0 ; pmul = 0.0 ; maxgen=1000 ; exe = 40 >

6. Statics analyze of interference (crossover) experiments:

To have a clear idea about all data presented in the previous graphs we precedeto simple statistics analyses, where we focus on the maximum solution achieved so far by each experiment, the average value of solutions realized and finally the standard deviation of solutions found. The extracted data are regrouped in the table below, and represented over the three adequate graphs.

pcross	pmut	capacity	maxfitness	avgfitness	segmafitness
0	0,0	3154,5000	3154,36230	3149,13721	7,20881081
0,05	0,0	3154,5000	3154,47949	3149,91016	1,93285191
0,125	0,0	3154,5000	3154,49780	3149,59229	0,89003757
0,25	0,0	3154,5000	3154,49976	3149,22119	0,352199465
0,375	0,0	3154,5000	3154,49854	3149,20605	0,532612264
0,5	0,0	3154,5000	3154,49976	3149,36597	0,214299604
0,625	0,0	3154,5000	3154,49951	3149,38989	0,175307825
0,75	0,0	3154,5000	3154,50000	3149,40234	0,165061265
0,875	0,0	3154,5000	3154,50000	3149,44946	0,0957758799
1	0,0	3154,5000	3154,50000	3194,45361	0,0714417398

Table 1:table of crossover experminets

4.1. Maximum solution achieved:

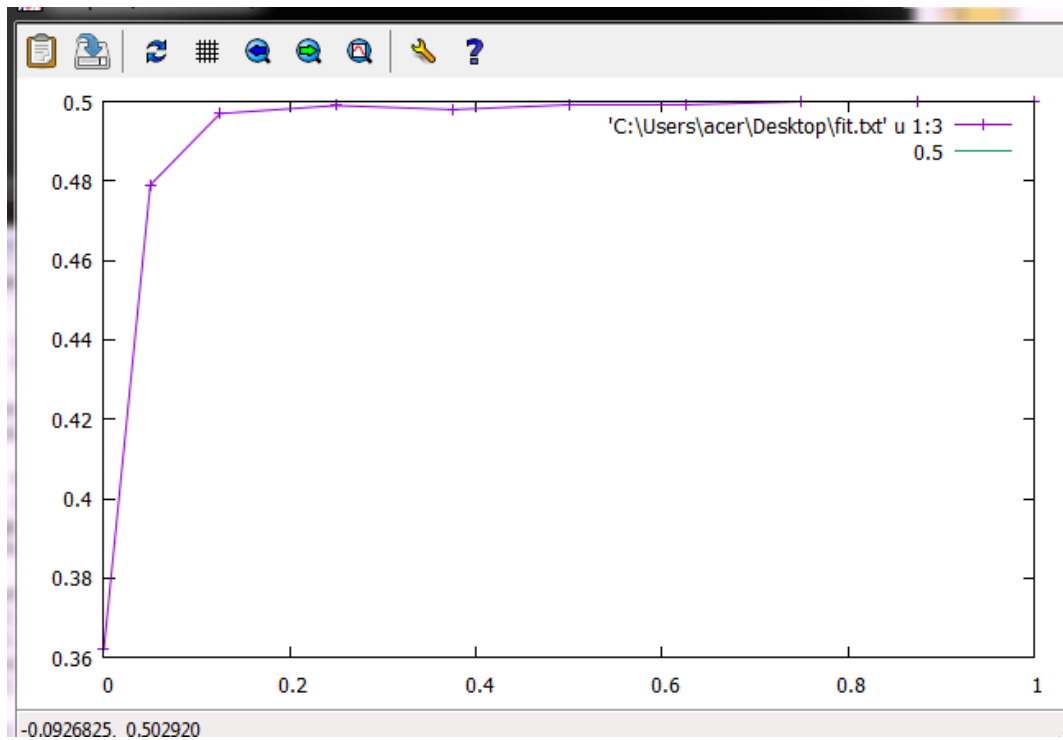


Figure 16:<maximum fitness versus crossover probability >

This graph shows a very interested property, increasing the crossover probability from **0.0 to 1.0** with step **0.1**, pushes the algorithm to reach the maximum value of profit. Where, it is attained since probability crossover value of ***pcross* = 0.75** and all after. So, ***pcross* = 0.75** might be adopted for further investigation.

$$\lim_{pcross \rightarrow 1} maxfitness = Solution$$

4.2. Average value of solutions found and standard deviation:

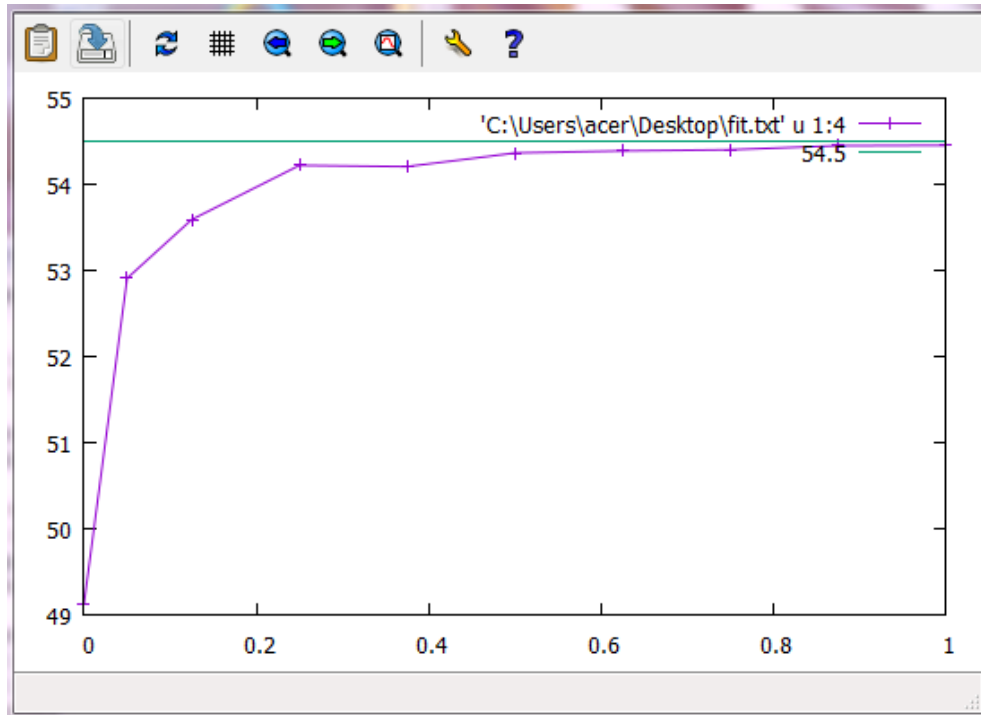


Figure 17:< average fitness versus crossover probability >

The graph presenting the average value behaves like the one of the maximum values attained by the algorithms for each value of ***pcross***, which one again confirms the convergence aspect of the algorithm ***GIQA***, to reach the maximum value. This is agreed well with the behavior of the standard deviation which tends decreased speedily towards **zero** when the crossover probability ***pcroo*** increases.

$$\lim_{pcross \rightarrow 1} avgfitness \approx Solution$$

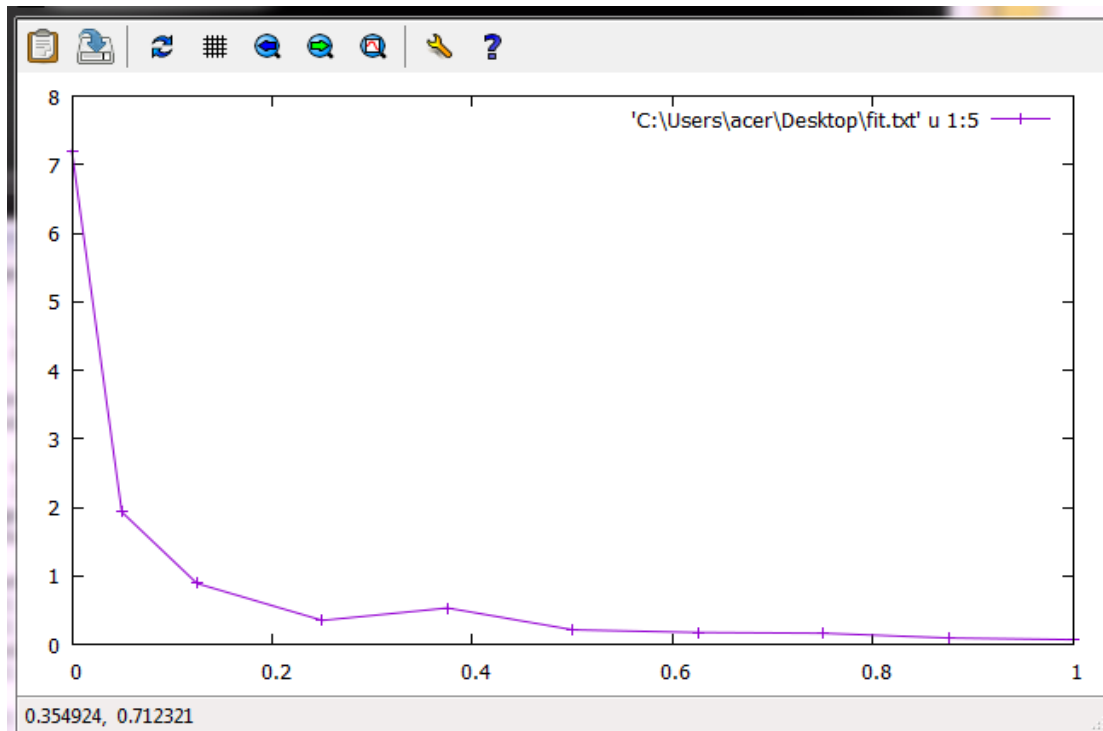


Figure 18:<average fitness versus crossover probability >

All this study, and analyze are done over 40 executions to have results and interpretation statistically consistent and valid, because, we don't have to forget that our algorithm **GIQA** is fundamentally a random algorithm.

$$\lim_{pcross \rightarrow 1} \sigma \approx 0$$

5. The effect of interference (crossover operation):

The study of the effect of interference on proposed algorithm performance, measured the reaction of the algorithm to the augmentation of the amount of population engaged in the interference, which had reveals, a sensible reaction towards high performance, we've obtained that choosing ***pcross* = 0.75** is much sufficient to set much benefit of the algorithm.

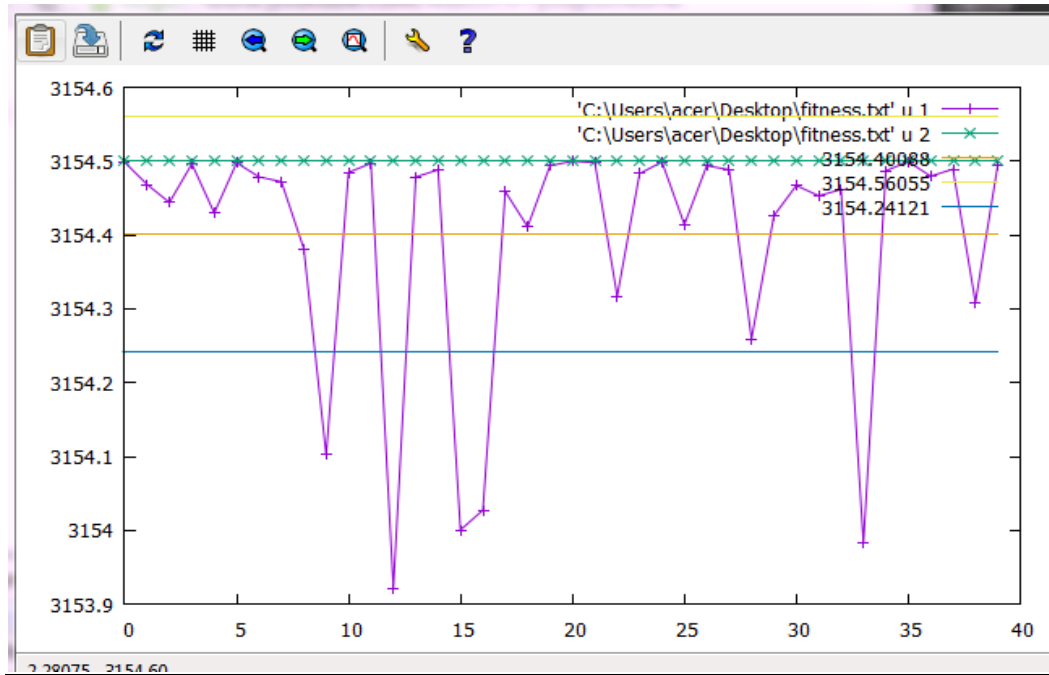


Figure 19:<m=100 ; n=100 ; pcross =0.75 ; pmut = 0.02 ; maxgen=1000 ; exe = 40 >

6. Analyze the effect the phase shifting reflected by mutation operation on the algorithm performance:

We are interesting in studying the effect of mutation on the algorithm performance. To do this, we fix the crossover probability at **0.75** and we vary the value of mutating probability from **0.0** to **0.1** by step of **0.01** to have the chance to see the fitness's variation over this interval $[0, 0.1]$. Of course, over 40 executions, to get credible results. The experiments data are catch and represented in the following graphs.

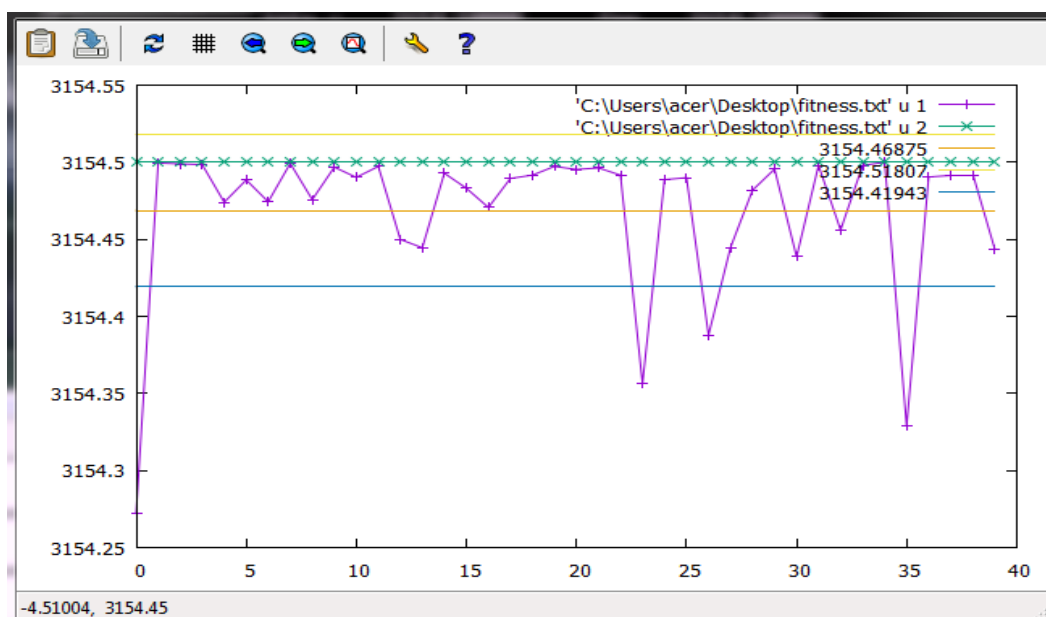


Figure 20: <m=100 ; n=100 ; pcross =0.75 ; pmul = 0.04 ; maxgen=1000 ; exe = 40 >

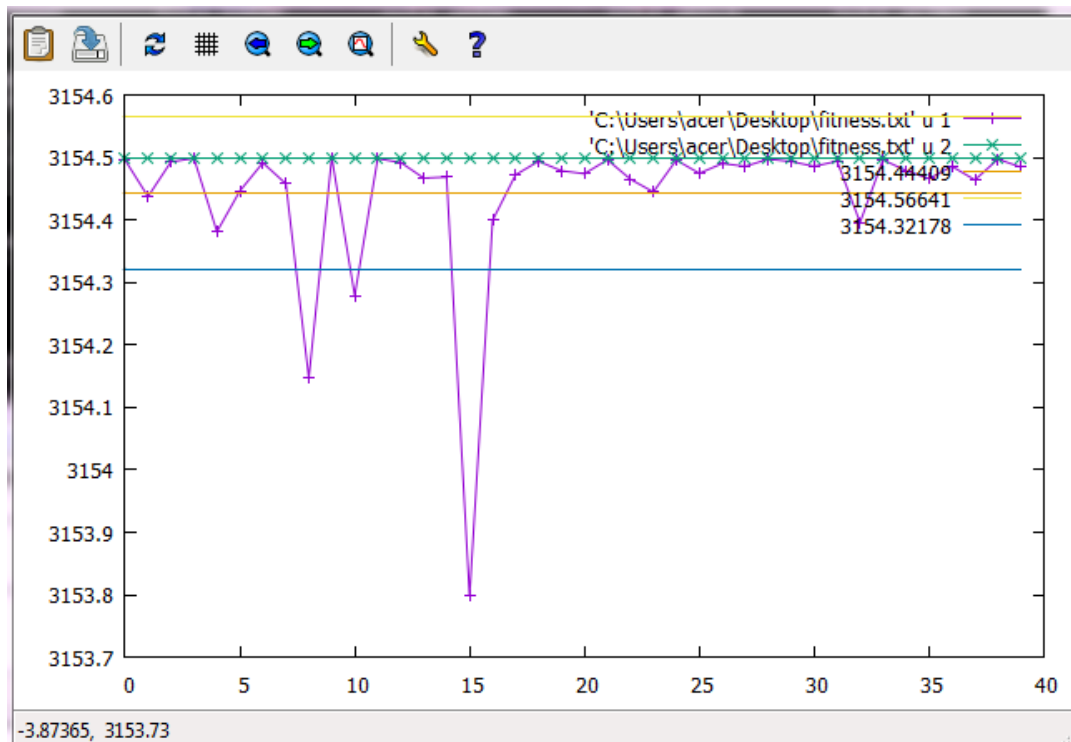


Figure 21: <m=100 ; n=100 ; pcross =0.75 ; pmul = 0.05 ; maxgen=1000 ; exe = 40 >

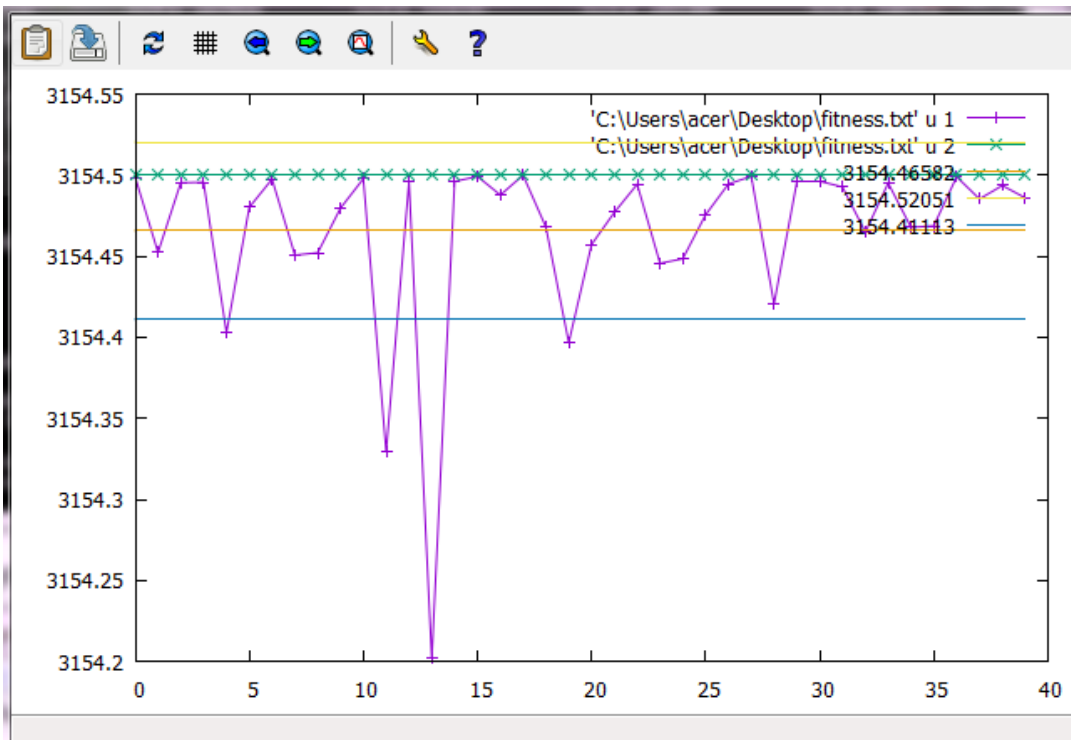


Figure 22: <m=100 ; n=100 ; pcross =0.75 ; pmul = 0.03 ; maxgen=1000 ; exe = 40 >

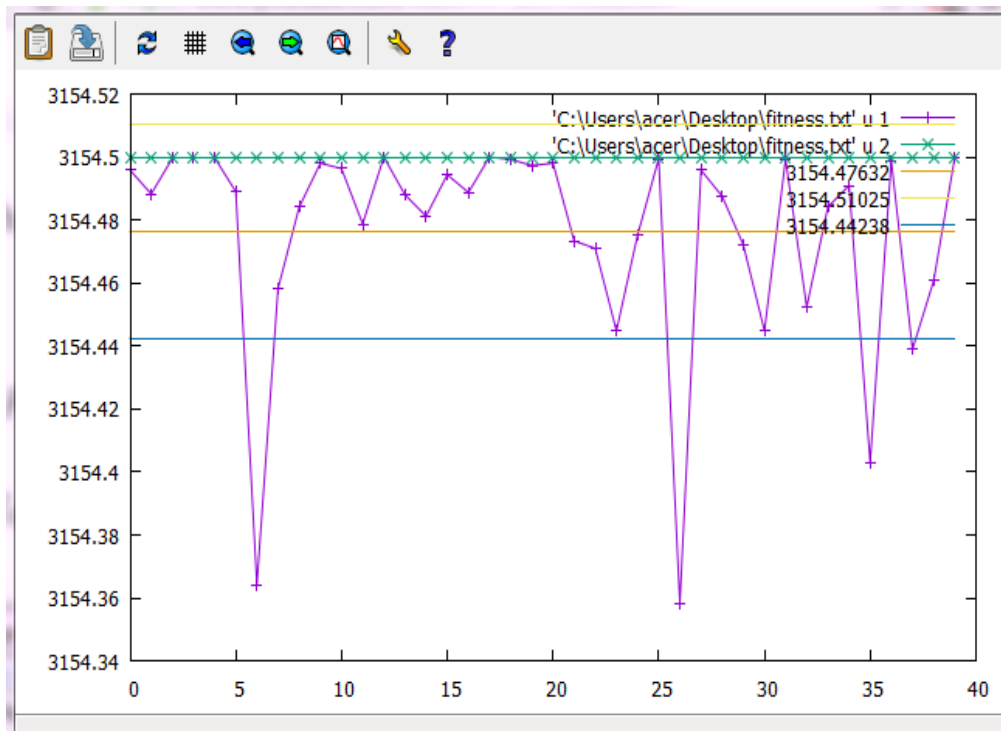


Figure 23:<m=100 ; n=100 ; pcross =0.75 ; pmult = 0.06 ; maxgen=1000 ; exe = 40 >

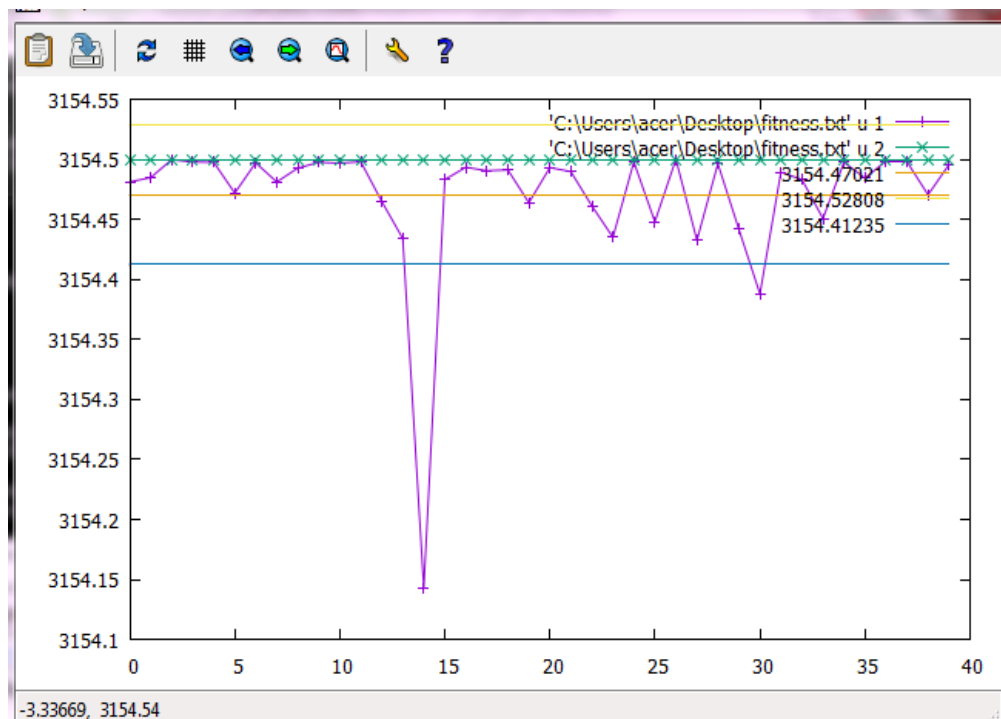


Figure 24:<m=100 ; n=100 ; pcross =0.75 ; pmult = 0.07 ; maxgen=1000 ; exe = 40 >

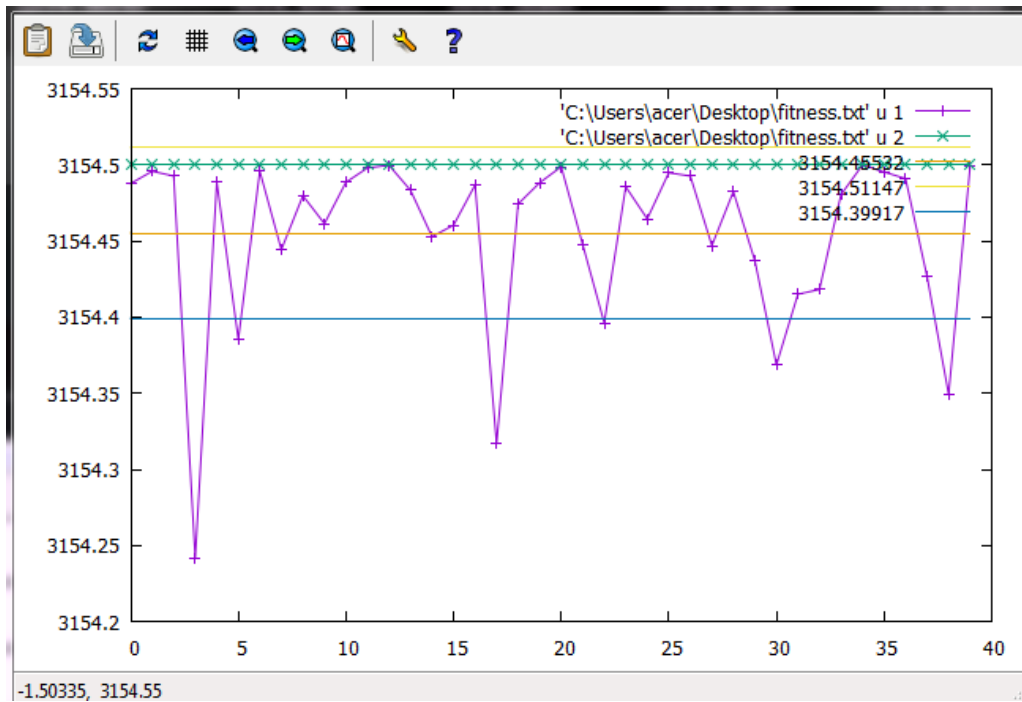


Figure 25:<m=100 ; n=100 ; pcross =0.75 ; pmult = 0.08 ; maxgen=1000 ; exe = 40 >

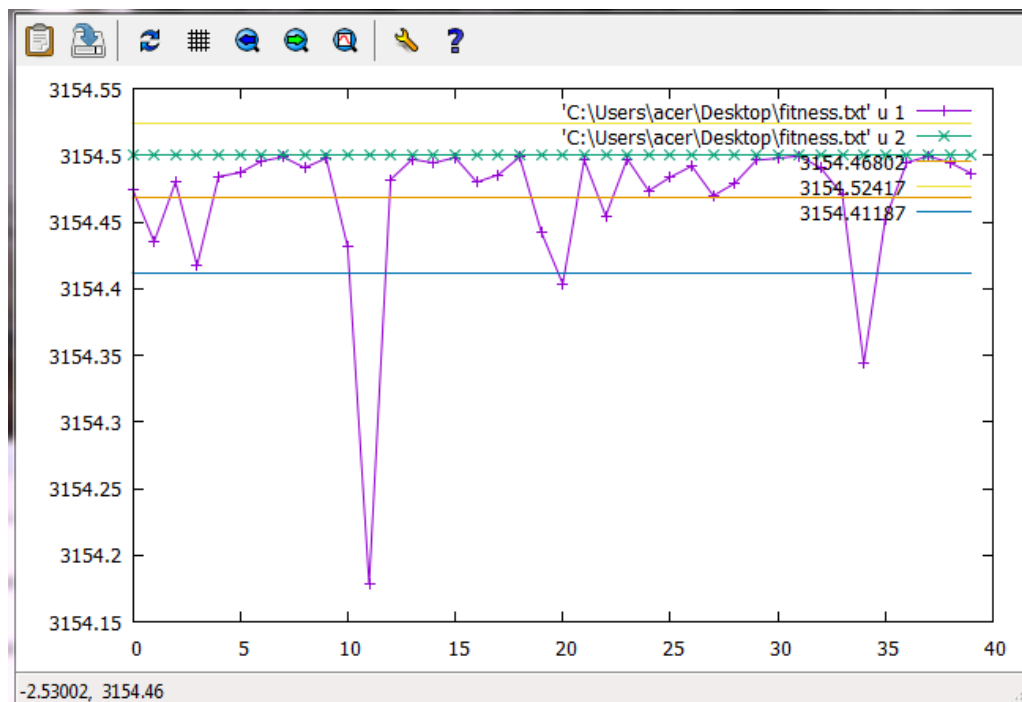


Figure 26:<m=100 ; n=100 ; pcross =0.75 ; pmult = 0.09 ; maxgen=1000 ; exe = 40 >

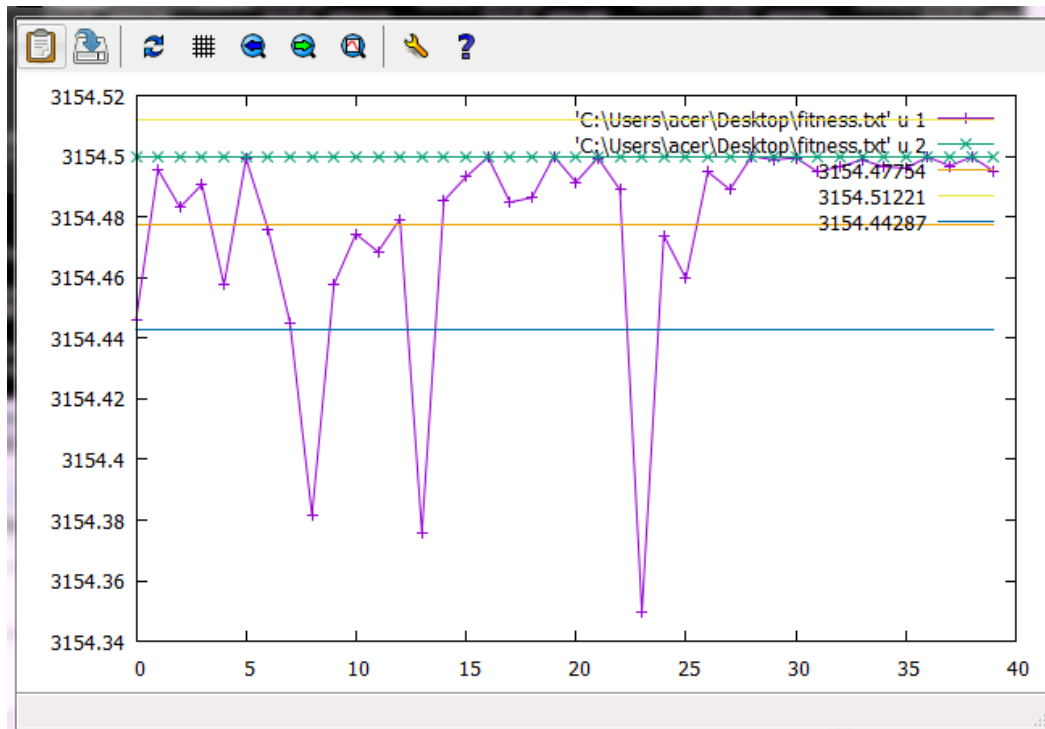


Figure 27:<m=100 ; n=100 ; pcross =0.75 ; pmut = 0.15 ; maxgen=1000 ; exe = 40 >

7. Statics analyses of phase shifting (mutation)experiments:

we adopt the same statics analyze done for the crossover operation, and the data are collected in the table below followed by the adequate graphs for each of maximum solution, average fitness, and the standard deviation.

pcross	pmut	capacity	maxfitness	avgfitness	segmafitness
0,75	0,00	3154,5000	3154,36230	3154,40600	0,13300
0,75	0,01	3154,5000	3154,47949	3154,39500	0,17300
0,75	0,02	3154,5000	3154,49780	3154,40000	0,15900
0,75	0,03	3154,5000	3154,49976	3154,46500	5,40000
0,75	0,04	3154,5000	3154,49854	3154,46799	4,90000
0,75	0,05	3154,5000	3154,49976	3154,44399	0,12200
0,75	0,06	3154,5000	3154,49951	3154,47599	3,30000
0,75	0,07	3154,5000	3154,50000	3154,46999	5,70000
0,75	0,08	3154,5000	3154,50000	3154,45499	5,60000
0,75	0,09	3154,5000	3154,50000	3154,46799	5,60000
0,75	0,10	3154,5000	3154,50000	3154,48799	1,30000

Table 2 : table of mutation experiments

7.1. Maximum solution achieved:

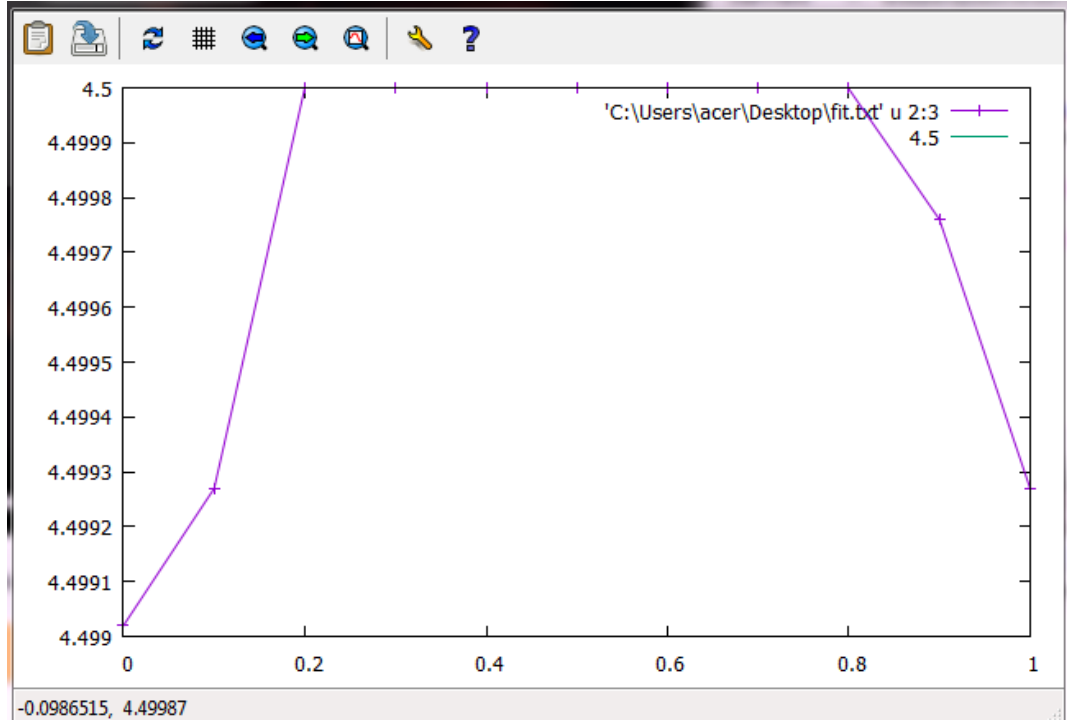


Figure 28:<m=100 ; n=100 ; pcross =0.75 ; pmut =0; maxgen=1000 ; exe = 40 >

8. Fluctuation:

The effect of fluctuation which is perfectly modeled by the mutation operation, the effect of increasing mutation probability is clear on the capacity of the proposed algorithm to reach the maximum value of profits(cap). Which is attained at the value $pmut = 0.02$, and saves it until the value $pmut = 0.09$ where it decreases slightly. So, we can take the value 0.03 for the mutation probability (fluctuation) to be much confident that the maximum value of the profit (cap) is covered.

$$\begin{cases} p_{croo} = 0.75 \\ p_{mut} = 0.03 \end{cases} \rightarrow \text{maxfit} = \text{maxprofit}(\text{capacity})$$

8.1. The fitness average value and standard deviation:

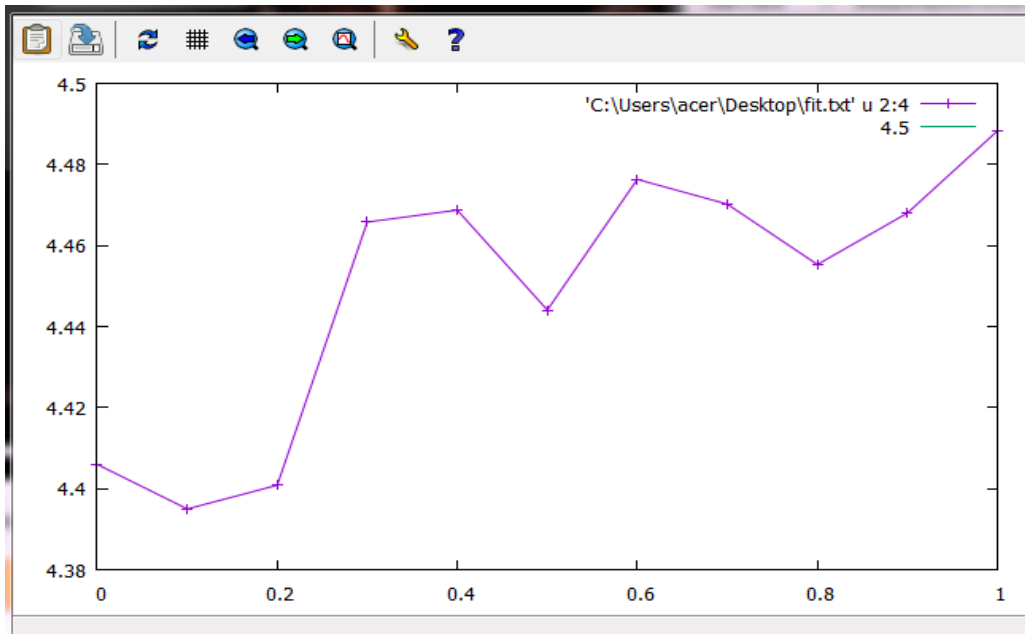


Figure 29:<m=100 ; n=100 ; pcross =0.75 ; pmut = 0 ; maxgen=1000 ; exe = 40 >

The fitness average value is globally increasing towards the maximum fitness, with certain variation. This fact is well supported by the globally decreasing of the standard deviation. We can observe that the mutation probability value $pmut = 0.03$ is a good choice.

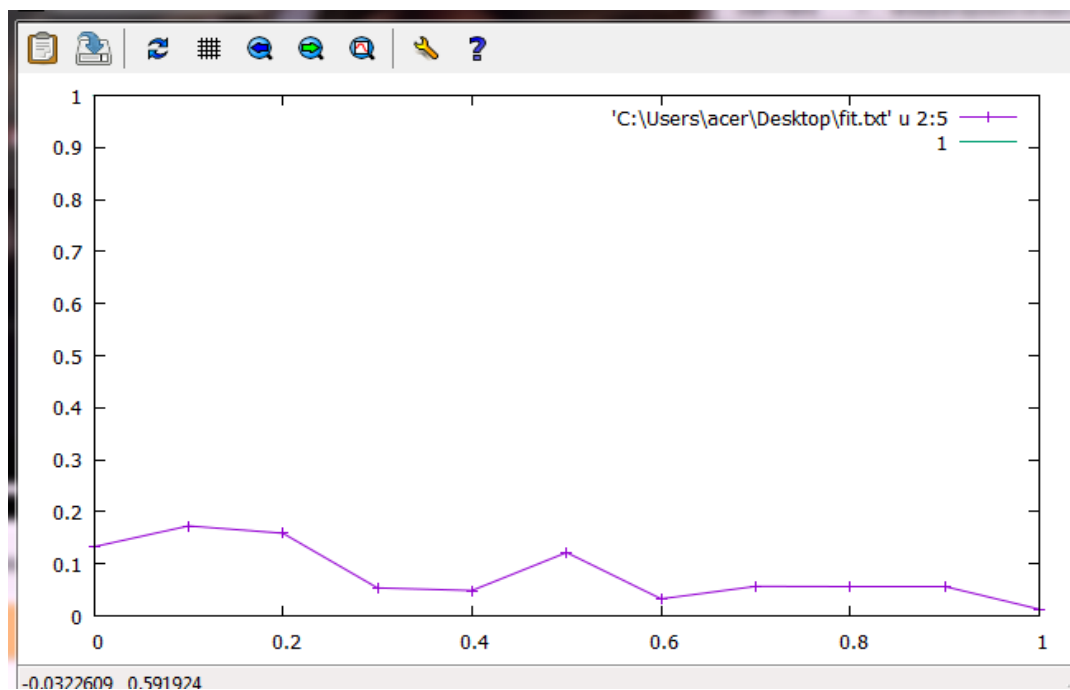


Figure 30:<m=100 ; n=100 ; pcross =0.75 ; pmut = 0 ; maxgen=1000 ; exe = 40;>

8.2. The behavior of GIQA over 1000 iteration:

We launch the algorithm for one execution, over 1000 iteration, to see its behavior in founding and reaching high solution, or as we claimed the exact solution. The graph below presents this behavior.

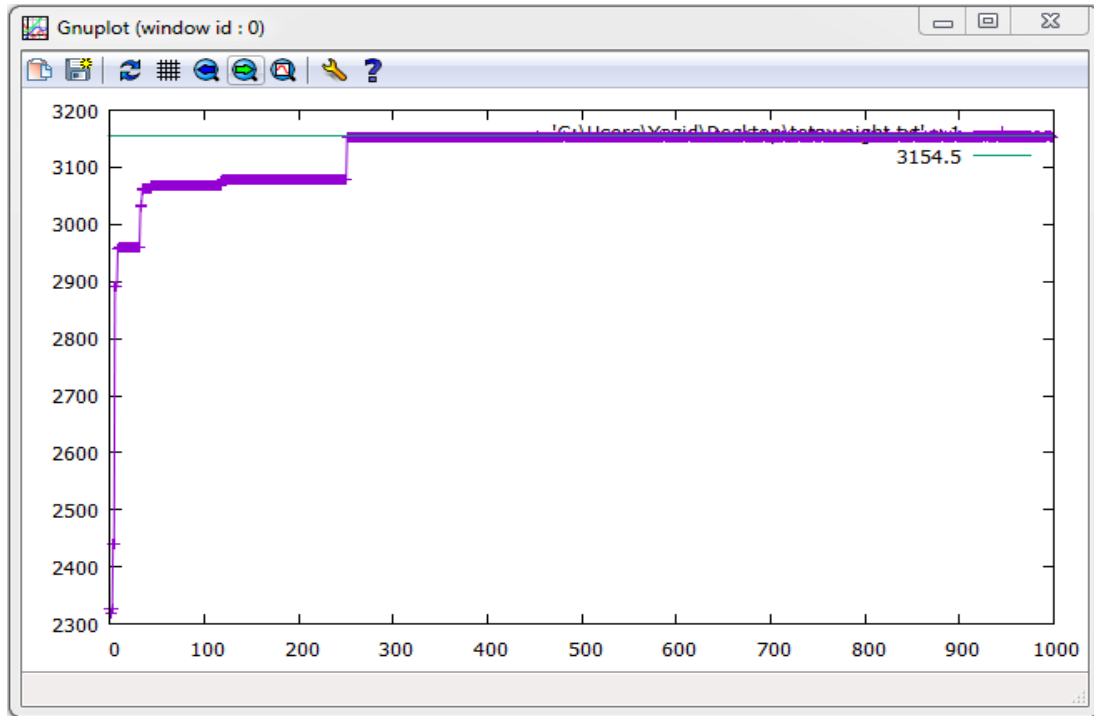


Figure 31 :<m=100 ; n=100 ; pcross =0.75 ; pmut = 0.03 ; maxgen=1000 ; exe = 1;(tetweight)>

The algorithm shows four zones of behavior changing; **[0, 30]** in this interval the algorithm presents a sudden speedily increasing behavior to reach a very high point not far from the solution, after that it stabilizes until the iteration **50** where another speedy step takes it more high and it stabilizes again, a little step up is made at the **125** iteration and stands stable, however at **250** iteration, it jumps to be really very close to the solution, and get fluctuated so closely to the solution, that its representation on the graph seems to make a line.

9. Qbit behavior :

We turn our attention toward the behavior of a single Qbit in Qregister, we focus on the Qregister representing the three best particles, the first, the second, and the third ones. Our goal is to get a nice idea about the variation of the θ – *angle* of the best Qbits.

The three Qbits present a strong variation of θ , this is more violent for the first best Qbit and the strength gets decreased a little bit for the second, and after that the third one. So,

teta variation is more strong when we approach the top Qbit, we can deduce that this strong behavior of their Qbits that's give particles potential to found more best solutions.

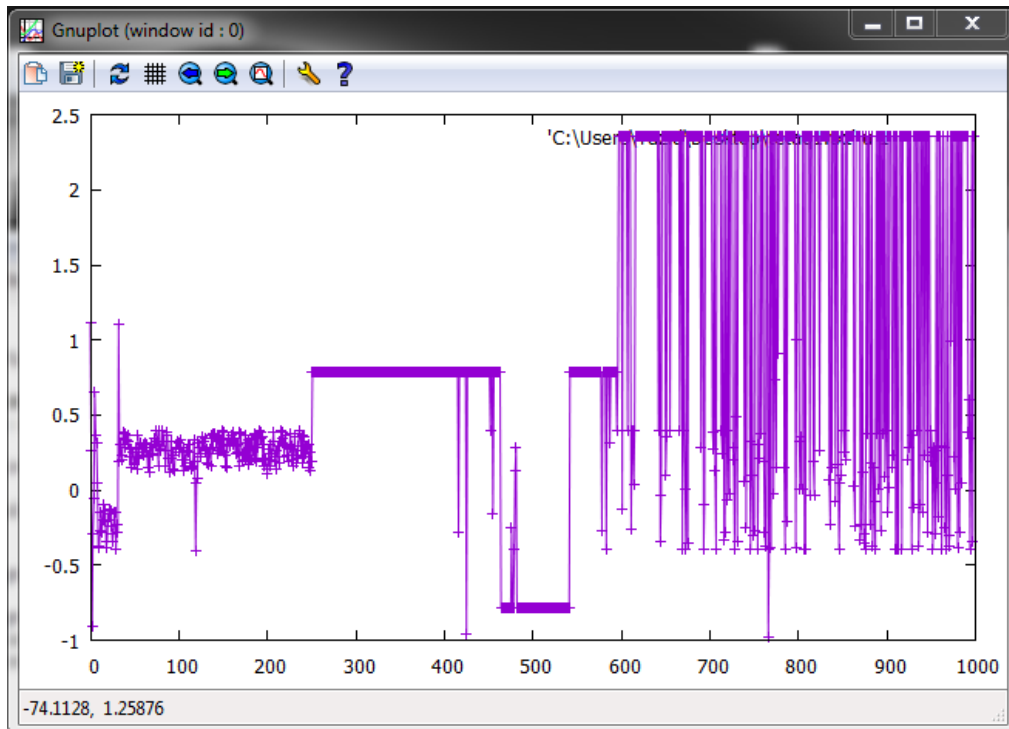


Figure 32:<m=100 ; n=100 ; pcross=0.75 ; pmul = 0.03 ; maxgen=1000 ; exe = 1;(tetaq1)>

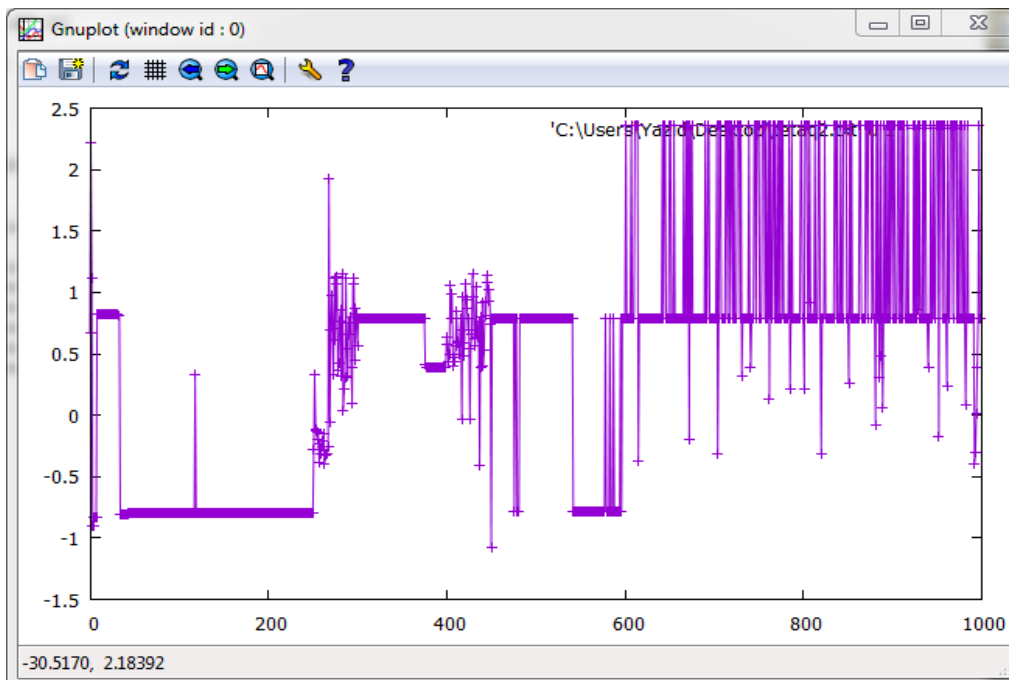


Figure 33:<m=100 ; n=100 ; pcross=0.75 ; pmul = 0.03 ; maxgen=1000 ; exe = 1;(tetaq2)>

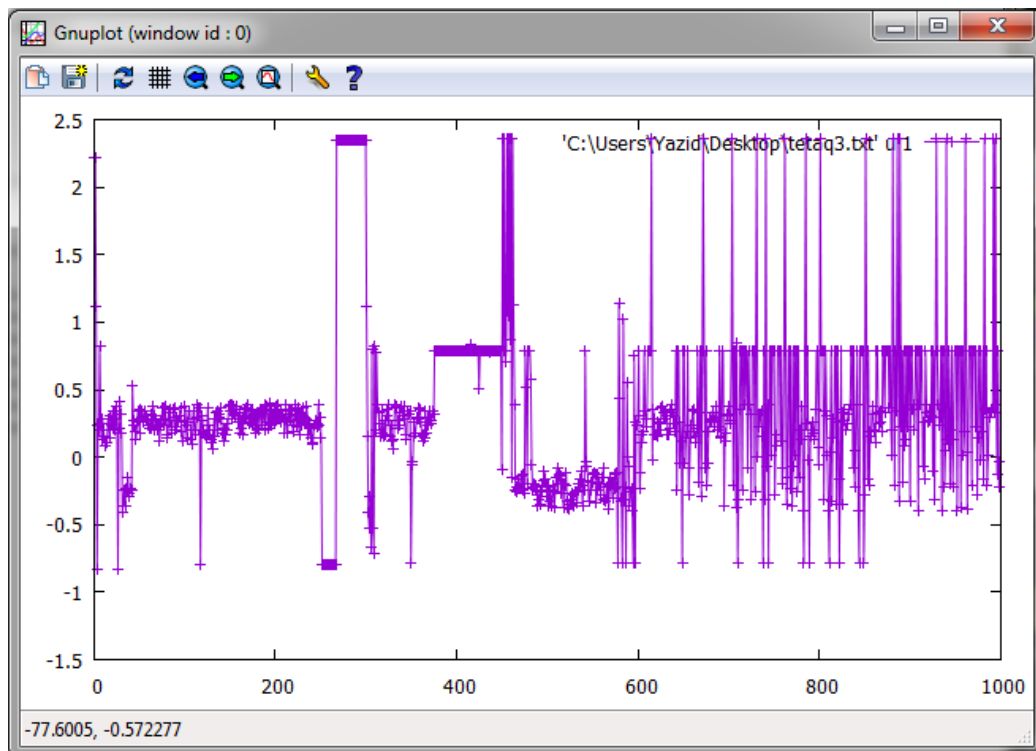


Figure 34:<m=100 ; n=100 ; pcross =0.75 ; pmut = 0.03 ; maxgen=1000 ; exe = 1;(tetaq3)>

10.Conclusion:

Genetic Inspired Quantum Algorithm (**GIQA**) was deeply analyzed and the results show a very high performance getting us to think that it solves the knapsack problem.

General Conclusion

In this work we have been interested to the combinatorial problems, where we have propose a novel evolutionary quantum inspired algorithm to attempt solve this challenging class of problems, at least with a good accuracy.

Our idea is inspired from the superposition principle of quantum mechanics, which contains all the secret and weirdness of this fundamental theory. Interference and phase shifting were the two implication of superposition principle that we have modeled to catch and benefit the proposed algorithm with the computational power of quantum computation. Genetic Algorithm was the natural evolutionary technique choice to this task, because its mechanism fit well. The interference phenomenon is well modeled by the recombination (crossover) operator applied on the phenotype rather than its genotype. Also, the phase shifting is perfectly modeled by the mutation operation.

We have investigate our propose algorithm named Genetic Inspired Quantum Algorithm to study its performance. We have chosen the Knapsack NP-Hard problem as a model to evaluate and valid our proposed algorithm. So, several experiments were done, with a statistics vision, were the algorithm was executed several time (40 executions) to have a consistent and credible results, knowing that our algorithm is a random one.

We have recorded a great accuracy of our solution compared to the expected solution of the problem. The algorithm reached the expected exact solution with a sensible percentage; so , we have pushed our investigation to more understand its behavior and find the adequate parameters to adjust them to increase the percentage of finding the exact solution. Our goal was successfully attained with high accuracy.

Our further researches will focus on extending our proposed algorithm genetic Quantum Inspired Algorithm **GIQA** to the General model of Knapsack problem and the other NP-Hard problem. We hope also adapt **GIQA** to the multi-objective NP-Hard problems.

Bibliographie :

[1] : Holland, J. H.: *Adaptation in natural and artificial systems*. Ann Arbor: The University of Michigan Press, 1975.

[2]: Rechenberg, I.: *Evolutionsstrategie - Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Stuttgart: Frommann-Holzboog, 1973.

[3]: Schwefel, H.-P.: *Numerical optimization of computer models*. Chichester: Wiley & Sons, 1981.

[4]. Fogel, L. J., Owens, A. J. and Walsh, M. J.: *Artificial Intelligence through Simulated Evolution*. New York: John Wiley, 1966.

[5]:[6]:[8][11] IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, VOL. 6, NO. 6, DECEMBER 2002 580 Quantum-Inspired Evolutionary Algorithm for a Class of Combinatorial Optimization.

[12]:Memory Méthodes de résolution de problèmes difficiles académiques .2013

[13] "FORTRAN". American Heritage Dictionary of the English Language (5 ed.). The Free Dictionary. 2011. Retrieved 2016-02-08.

[14] John Backus. "The history of FORTRAN I, II and III" (PDF). Softwarepreservation.org. Retrieved 19 November 2014.

[15] Eugene Loh (18 June 2010). "The Ideal HPC Programming Language". Queue(Association of Computing Machines)

[18] Singh, Madhusudan. "gnuplotfortran - the Fortran 95 interface to gnuplot". gnuplotfortran.sourceforge.net. Retrieved 10 May 2016.

webGraphe:

[7][10] : <http://www.geatbx.com/docu/alginde-01.html#TopOfPage> consulted 20/05/2016

[9] : <http://slideplayer.com/slide/7285101> consulted 19/05/2016

[16] <https://en.wikipedia.org/wiki/Code::Blocks> consulted 26/05/2016

[17] <https://en.wikipedia.org/wiki/Gnuplot> consulted 26/05/2016

[19] http://linuxgazette.net/168/misc/lg/2_cent_tip_piping_to_gnu_plot_from_c.html consulted 26/05/2016

Abstract

The quantum-inspired evolutionary algorithm is new area of intelligent computation. In this work we propose a new quantum inspired genetic algorithm, to solve a class of combinatorial optimization problems. our QIGA is applied to the 0-1 Knapsack problem to evaluation and validation.

Key words: QIGA, Quantum Computing , Genetic Algorithms, Knapsack Problem

Abstrait

Les algorithmes évolutionnaire inspiré du quantique est un nouveau domaine de la computation intelligente. Dans notre travail, nous avons proposé un nouvel algorithme génétique inspiré du quantique, pour la résolution d'une class de problèmes d'optimisation combinatoire. notre QIGA est appliqué sur le problème du 0-1 Knapsack pour l'évaluation et la validation.

Mots clés: QIGA, computation quantique , algorithmes évolutionnaire, problème de knapsack

ملخص

الخوارزمية التطورية المستوحات من ميكانيك الكم هي حقل جديد للحاسوبية الذكية . في عملنا هذا , اقترحنا خوارزمية جينية مستوحات من ميكانيك الكم جديدة ,لحل قسم من مشاكل التحسينات التوافقية . الخوارزمية المقترحة طبقت على مشكلة حقيبة الظهر 0-1 من أجل التقييم والتحقق .

الكلمات المفتاحية : الخوارزمية الجينية المستوحات من الكم , الحوسبة الكمية,الخوارزميات التطورية , مشكل حقيبة الظهر .