National University of Asuncion

Polytechnic School

Quantum Algorithms for Multiobjective Combinatorial Optimization

Thesis submitted by

GERARDO G. FOGEL

in partial fulfillment of the requirements for

Master in Computer Sciences

Advisors:

Benjamín Barán, D.Sc. & Marcos Villagra, Ph.D.

San Lorenzo - Paraguay August 2017

Quantum Algorithms for Multiobjective Combinatorial Optimization

GERARDO GABRIEL FOGEL LEZCANO

Advisors:

- Prof. D.Sc. **BENJAMIN BARAN**
- Prof. Ph.D **MARCOS VILLAGRA**

Submitted in partial fulfillment of the requirements for the degree of Master in Computer Science in the Polytechnic School at the National University of Asuncion, San Lorenzo - Paraguay, 2017.

San Lorenzo - Paraguay
August 2017

Quantum Algorithms for Multiobjective Combinatorial Optimization

Gerardo Gabriel Fogel Lezcano

Approved in 28 August 2017.

Prof. Marcos Villagra, PhD.

Prof. Benjamín Barán, D.Sc.

Prof. Sebastian Grillo, D.Sc.

Prof. Diego Pinto Roa, Dr.

Prof. Rolando Cuevas, D.Sc.

Fogel Lezcano, Gerardo Gabriel
Quantum Algorithms for Multiobjective Combinatorial Optimization / Gerardo Gabriel Fogel Lezcano, Advisor: Marcos Villagra, Co-Advisor: Benjamín Barán,
San Lorenzo: FPUNA, 2017.
Thesis degree (Master in Computer Sciences) — Polytechnic School, National University of Asuncion, 2017.

I would like to dedicate this thesis to my loving parents

"Champions are not made in gyms. Champions are made from something they have deep inside them - a desire, a dream, a vision. They have to have the skill, and the will. But the will must be stronger than the skill "

Muhammad Ali

PER ASPERA AD ASTRA

Quantum Algorithms for Multiobjective Combinatorial Optimization

Author: Gerardo Gabriel Fogel Lezcano

Advisors:
Prof. D.Sc. Benjamín Barán.
Prof. Ph.D Marcos Villagra.

ABSTRACT This thesis studies multiobjective optimization problems in the context of quantum computing. Quantum computing is a computational paradigm based on the laws of quantum physics as superposition, interference and entanglement. New quantum algorithms have emerged that proved to be more efficient than classical algorithms. Particularly, Grover's search algorithm can find a specific element out of a set of N elements with complexity $O(\sqrt{N})$. Applications of Grover's algorithm to optimization problems are currently being studied by other researchers, and in this thesis, a new adaptive search method based on Grover's algorithm applied to several biobjective optimization problems is introduced. This new algorithm is compared against one of the most cited multiobjective optimization algorithms known as NSGA-II. Experimental evidence suggests that the quantum optimization method proposed in this work is at least as effective as NSGA-II in average, considering an equal number of executions. The proposed quantum algorithm, however, only requires approximately the square root of the number of evaluations executed by NSGA-II. Also, two different types of oracles with regard to the proposed algorithm were considered and the experimental results have shown that one of this oracles has requiered less iterations for similar performance.

Key words : Multiobjective Optimization, Quantum Computing, Grover Search Algorithm.

Contents

List of Figures

List of Tables

List of Symbols

List of Acronyms

GAS	Grover Adaptative Search 9
MOGAS	Multiobjective Optimization Grover Adaptative Search 13
NSGA-II	Non-dominated Sorting Genetic Algorithm version two 15
RG	Randomly generated problems 15
ZDT	Test suites family functions 15
pop	Population number 17
gen	Generation number 17

List of Algorithms

Chapter 1

Introduction

1.1 Background

Quantum Computing is a computing paradigm based on the random nature and laws of quantum physics as superposition, entanglement and interference, and it has been a field of study in computer science since the 1980's [1]. It is strongly believed that quantum phenomena cannot be simulated efficiently using a classical computer [1]. In the middle of the 1990's, after the development of an efficient quantum algorithm for integer factorization by Shor [2], the idea of quantum computers became more relevant. In a similar way, another milestone was achieved with a quantum algorithm for search in unstructured databases developed by Grover [3]. This algorithm can find a specific marked element from a finite set of N elements with a computational complexity of order $O(\sqrt{N})$, instead of $O(N)$ for classical computers.

After Grover's search algorithm, several researchers proposed diverse methods based on Grover's algorithm applied to global optimization. Dürr and Høyer [4] presented a quantum algorithm for finding the minimum value of an objective function. Another relevant contribution comes from Baritompa, Bulger and Wood [5], who proposed an adaptive search method for minimization problems. Furthermore, Barán and Villagra [6] introduced the first quantum algorithm for multiobjetive combinatorial optimization based on a quantum adiabatic computer.

The design of quantum algorithms, that is a secuence of quantum operators, is nowadays one of the major problems in the quantum computing community. The strategies for constructing classical algorithms as divide

and conquer, procedural abstraction, etc, are not naturally adapted to the quantum paradigm. Strategies for designing quantum algorithms are phase amplification, order finding, phase estimation, to name a few. As an example of the applications of these strategies, Grover's algorithm uses the amplitude amplification technique, and Shor's algorithm relies in reductions to order finding and phase estimation. The use of very well known heuristics as branch-and-cut, backtracking, etc., in a quantum context is extremely difficult, and therefore it becomes necessary the study of different approaches to improve the efficiency of optimization algorithms.

1.2 Objectives

1.2.1 General Objective

The main objective of this work is to develop a quantum algorithm for multiobjective optimization problems.

1.2.2 Specific Objectives

To achieve the main goal of this thesis, the following specific objectives were identified:

- To implement a quantum adaptative search algorithm for multiobjective combinatorial optimization problems.
- To study different definitions of quantum oracle mechanisms associated to a search algorithm.
- To compare the proposed algorithm against NSGA-II, one of the most cited multiobjective optimization algorithms [8].

1.3 Main Contributions

In this thesis, we propose an implementation of Grover's algorithm to multiobjetive optimization problems. Two algorithms are proposed that can query the objective functions via so-called quantum oracles. For comparison purposes, two different oracles are studied. The first oracle "marks" non-dominated solutions with respect to a known feasible solution of the decision

space. The second oracle also "marks" non-dominated solutions as the first one but, the difference is that it marks non-comparable solutions too. Both oracles are implemented in an algorithm called MOGAS from *Multiobjective Optimization Grover Adaptive Search*, which is based on the Grover adaptive search algorithm of Baritompa, Bulger and Wood [5].

The experimental results of this work suggest that MOGAS (considering both types of oracles) was not only an effective approach for multiobjective optimization problems, but it was also efficient when compared against NSGA-II. In most of the studied cases, MOGAS obtained better or equal results in average for the same number of executions. It is important to note that in spite of the simple adaptive strategy used by MOGAS (considering both types of oracles), the results of this work present a remarkable performance over NSGA-II. Therefore, the experimental results show the efficiency of a simple quantum algorithm with respect to classical and more elaborate algorithms.

1.4 Outline of this Thesis

This thesis is structured as follow. First, in Chapter 2, a review of main ideas behind Grover's quantum search algorithm is explained. In Chapter 3, an explanation of Grover's adaptative search algorithm is presented. Chapter 4 introduces MOGAS, the novel proposal of this thesis. In Chapter 5, experimental results based on a biobjective optimization problems are studied. Finally, Chapter 6 concludes this thesis. In Appedix A, a brief overview of main ideas of quantum computing are summarized.

Chapter 2

Grover's Search Algorithm

In classical computation, finding a specific element out from a set of N elements requires up to N tries, i.e. the complexity is $O(N)$. In general, for unstructured search the complexity of finding a particular element is $O(N)$, which is tight [1].

Grover's search algorithm can find a specific element out from a set of N elements with complexity $O(\sqrt{N})$. This is possible, because it uses a quantum operator $\mathbf{G}$ known as the *Grover operator* which is constructed from an oracle operator $\mathbf{O_G}$ and a phase operator $\mathbf{W}$ (see Figure 2.1).

The number of iterations r necessary to find a desired item out of N alternatives is obtained from the following equation

$$r = \left\lfloor \frac{\pi}{4}\sqrt{N} \right\rfloor \approx \sqrt{N}, \tag{2.1}$$

which corresponds to a complexity $O(\sqrt{N})$ [3].

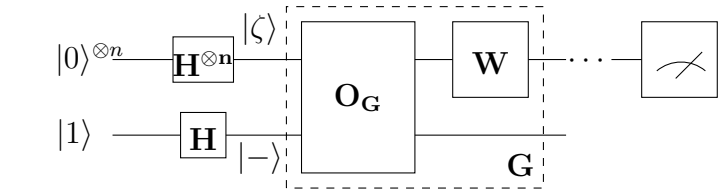
Grover's algorithm uses as inputs, n qubits $|0\rangle^{\otimes n}$, where $2^n = N$, and an ancilla qubit $|1\rangle$. The first input $|0\rangle^{\otimes n}$ is transformed to a superposition state using a n-fold Hadamard transformation $\mathbf{H}^{\otimes n}$,

$$|\zeta\rangle = \mathbf{H}^{\otimes n}|0\rangle^{\otimes n} = \frac{1}{\sqrt{N}} \sum_{x \in \{1,0\}^n} |x\rangle. \tag{2.2}$$

This superposition state represents a linear combination of all basis states. The second register is transformed using a Hadamard gate according to

$$\mathbf{H}|1\rangle = |-\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}}. \tag{2.3}$$

Grover's algorithm is based on the ability of an oracle to "mark" a desired solution, which is represented by one of the basis states. Given a superposition state, the marking process of an oracle is a change of the coefficient sign in the basis state which corresponds to a desired solution; such a marking process will only be possible if some interaction exists between the oracle operator and the ancilla register. After the marking process, the phase operator performs an increase of the absolute value of the amplitude associated to the solution state while decreasing amplitudes associated to the other non-solution states. This will happen at each iteration, and because of that, it allows a geometrical interpretation. The action of Grover's algorithm on an input state can be understood as a rotation from the all non-solution states toward a solution state, for each application of the Grover operator $\mathbf{WO_G}$ (see Figure 2.2) [1].



$$\mathbf{O_G} = \mathbf{I} - 2|x_0\rangle\langle x_0|, \quad \mathbf{W} = 2|\zeta\rangle\langle\zeta| - \mathbf{I}, \quad \mathbf{G} = \mathbf{W}\ \mathbf{O_G}$$

Figure 2.1: Quantum circuit for Grover's algorithm and the equivalent Dirac's notation of the oracle, phase and Grover operators.

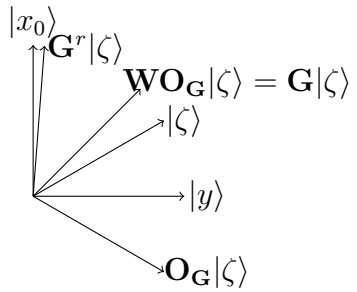


Figure 2.2: Geometrical interpretation of Grover's algorithm.

For example, consider finding a solution state x_0 from a set of $N = 8$ solutions. Then, the number of required iterations is given by

$$r = \left\lfloor \frac{\pi}{4}\sqrt{8} \right\rfloor = 2. \tag{2.4}$$

The quantum circuit for this particular example can be seen in Figure 2.3 with the time evolution of different states during a computation. The solution searched for this example corresponds to state $|7\rangle$. The state $|\zeta_0\rangle$ corresponds to a superposition of all initial states.

$$|\zeta_0\rangle = \frac{1}{2\sqrt{2}}|0\rangle + \frac{1}{2\sqrt{2}}|1\rangle + \frac{1}{2\sqrt{2}}|2\rangle + \cdots + \frac{1}{2\sqrt{2}}|7\rangle. \tag{2.5}$$

States $|\zeta_i\rangle$, with $i = 1, 2, 3, 4$ represent the states after the application of each quantum operator. Each quantum state $|\zeta_i\rangle$ can thus be explicitly computed

$$|\zeta_1\rangle = \frac{1}{2\sqrt{2}}|0\rangle + \frac{1}{2\sqrt{2}}|1\rangle + \frac{1}{2\sqrt{2}}|2\rangle + \cdots - \frac{1}{2\sqrt{2}}|7\rangle, \tag{2.6}$$

$$|\zeta_2\rangle = \frac{1}{4\sqrt{2}}|0\rangle + \frac{1}{4\sqrt{2}}|1\rangle + \frac{1}{4\sqrt{2}}|2\rangle + \cdots + \frac{5}{4\sqrt{2}}|7\rangle, \tag{2.7}$$

$$|\zeta_3\rangle = \frac{1}{4\sqrt{2}}|0\rangle + \frac{1}{4\sqrt{2}}|1\rangle + \frac{1}{4\sqrt{2}}|2\rangle + \cdots - \frac{5}{4\sqrt{2}}|7\rangle, \tag{2.8}$$

$$|\zeta_4\rangle = -\frac{1}{8\sqrt{2}}|0\rangle - \frac{1}{8\sqrt{2}}|1\rangle - \frac{1}{8\sqrt{2}}|2\rangle - \cdots + \frac{11}{8\sqrt{2}}|7\rangle. \tag{2.9}$$

After the application of two Grover's iterations $\mathbf{WO_G}$, the probabilities to observe/measure a solution state and the non-solution states are given by

$$P_{|7\rangle} = \left|\frac{11}{8\sqrt{2}}\right|^2 \approx 0.945 = 94.5\%, \tag{2.10}$$

$$P_{|x\rangle|x\notin\{x_0\}} = \left|-\frac{1}{8\sqrt{2}}\right|^2 \approx 0.0078 = 0.78\%. \tag{2.11}$$

Therefore, with high probability, it will be possible to observe the desired solution state.

Grover's algorithm can be generalized to find a set of l-solutions with r' iterations of the Grover operator $\mathbf{WO_G}$ and it is given by the following equation [1]:

$$r' = \left\lfloor \frac{\pi}{4}\sqrt{\frac{N}{l}} \right\rfloor \approx \sqrt{\frac{N}{l}}. \tag{2.12}$$

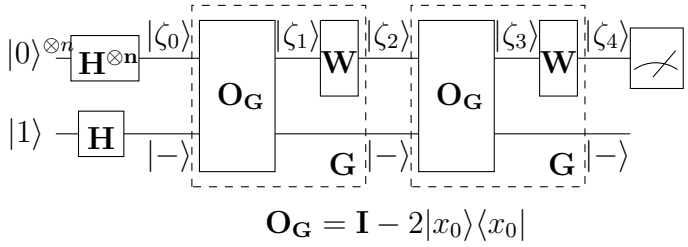


Figure 2.3: Example of the Grover's algorithm application for $N = 8$, when finding one possible solution.

A geometrical interpretation is also possible for this generalization. In this case, the rotation of the quantum state starts from all non-solution states toward the l-solution states. Therefore, after r' iterations, each one of the l-solution states can be observed with equal probability [1]. For example, if we desire to find two solution states x_0 and x_1 from a set of $N = 16$ solutions, then the number of iterations is given by

$$r' = \left\lfloor \frac{\pi}{4}\sqrt{\frac{16}{2}} \right\rfloor = 2. \tag{2.13}$$

Figure 2.4 shows the quantum circuit for $N = 16$ and how the quantum state changes in time. The desired solutions in this example correspond to states $|3\rangle$ and $|15\rangle$. The state $|\zeta_0\rangle$ corresponds to a superposition of all initial states.

$$|\zeta_0\rangle = \frac{1}{4}|0\rangle + \frac{1}{4}|1\rangle + \frac{1}{4}|2\rangle + \frac{1}{4}|3\rangle + \cdots + \frac{1}{4}|15\rangle. \tag{2.14}$$

States $|\zeta_i\rangle$, with $i = 1, 2, 3, 4$, are the states resulting after the application of each quantum operator.

$$|\zeta_1\rangle = \frac{1}{4}|0\rangle + \frac{1}{4}|1\rangle + \frac{1}{4}|2\rangle - \frac{1}{4}|3\rangle + \cdots - \frac{1}{4}|15\rangle, \tag{2.15}$$

$$|\zeta_2\rangle = \frac{1}{8}|0\rangle + \frac{1}{8}|1\rangle + \frac{1}{8}|2\rangle + \frac{5}{8}|3\rangle + \cdots + \frac{5}{8}|15\rangle, \tag{2.16}$$

$$|\zeta_3\rangle = \frac{1}{8}|0\rangle + \frac{1}{8}|1\rangle + \frac{1}{8}|2\rangle - \frac{5}{8}|3\rangle + \cdots - \frac{5}{8}|15\rangle, \tag{2.17}$$

$$|\zeta_4\rangle = -\frac{1}{16}|0\rangle - \frac{1}{16}|1\rangle - \frac{1}{16}|2\rangle + \frac{11}{16}|3\rangle - \cdots + \frac{11}{16}|15\rangle. \tag{2.18}$$

Following with the previous example, after the application of two Grover's iterations $\mathbf{WO_G}$, we obtain that the probabilities to observe/measure the solution states and the non-solution states are given by

$$P_{|3\rangle} = \left|\frac{11}{16}\right|^2 \approx 0.4726 = 47.26\%, \tag{2.19}$$

$$P_{|15\rangle} = \left|\frac{11}{16}\right|^2 \approx 0.4726 = 47.26\%, \text{ and} \tag{2.20}$$

$$P_{|x\rangle|x\notin\{x_0,x_1\}} = \left|-\frac{1}{16}\right|^2 \approx 0.0039 = 0.39\%. \tag{2.21}$$

With high probability it is possible to observe any of the two desired solution states.

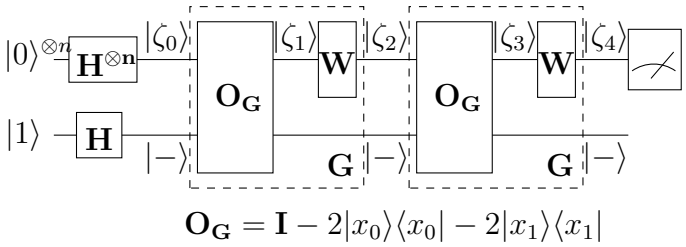


Figure 2.4: Example of the Grover's algorithm application for $N = 16$, when finding two possible solutions.

Chapter 3

Grover's Adaptative Search Scheme

3.1 Grover Adaptative Search (GAS)

Grover's algorithm is generally used as a search method to find a set of desired solutions from a set of possible solutions. Baritompa, Bulger and Wood [5] presented an application of Grover's algorithm to global optimization problems, which they call *Grover Adaptative Search (GAS)*. Basically, GAS is based on Grover's search with an "adaptive" oracle operator in a minimization context of the objective function, where adaptive means that the oracle can remember answers to previous oracle queries. The oracle operator marks all the solutions from a set, below a certain threshold value y given by

$$g(x) = \begin{cases} 1, & \text{if } f(x) < y \\ 0, & \text{if } f(x) \geq y \end{cases}, \tag{3.1}$$

where x is a possible solution in the decision space and $f(x)$ is the value of the objective function (in this case, the value of the objective function of a current known solution y). The oracle marks a solution x if and only if the boolean function $g(x) = 1$ [5].

The algorithm requires two extra parameters, a known solution x_k and the iteration count r_k. The iteration count is a number computed from the number of solutions that are better than the currently known solution. Initially, the algorithm randomly chooses a feasible solution from the decision space which becomes the known solution; however, the number of solutions

which are better than the known solution is unknown and a sequence of iteration counts is required to perform the search until an improved solution is found. Furthermore, when the algorithm finds a better solution, it becomes the new known solution. The new known solution is then used as a new threshold for the next iteration of GAS and the sequence of iteration counts must be computed again. In this way, GAS can find improved solutions in an adaptive search framework [5].

The pseudocode of the algorithm introduced by Baritompa, Bulger and Wood [5] is shown below, where the parameter k represents the search process count.

Algorithm 1 Grover Adaptative Search

1: Randomly choose x from the decision space.
2: Set $x_1 \leftarrow x$
3: Set $y_1 \leftarrow f(x_1)$.
4: For $k = 1, 2, \ldots$ until termination condition is met, do:
(a) Perform a Grover search of r_k iterations on $f(x) = y$ with threshold y_k, denoting the outputs as x and y.
(b) If $y < y_k$ set $x_{k+1} \leftarrow x$ and $y_{k+1} \leftarrow y$; otherwise, set $x_{k+1} \leftarrow x_k$ and $y_{k+1} \leftarrow y_k$.

3.2 Dürr and Høyer's Algorithm

Dürr and Høyer presented an algorithm based on Grover's method [4]. In their proposal, they introduced a random strategy for the selection of the iteration count based on a random selection of a number from a set of integer numbers, which differentiates it from GAS algorithm. This set starts with $\{0\}$ as the only element. When the search is unsuccessful in finding a better solution, the set increases its elements to a maximum of $\{0, \ldots, \lceil m - 1 \rceil\}$ at each search step, until a solution better than the current known solution is found. In this way, the set incorporates more integer numbers as elements and the probability of selecting the right iteration counts for a successful search increases.

The parameter m is updated at each step by $\min\{\lambda^i m, \sqrt{N}\}$, where λ is an incrementing factor given as a parameter, i represents the count of the

previous unsuccessful search steps and $N = 2^n$ is the number of total elements from the decision space based on the number of qubits n. Therefore, m is not allowed to exceed $\sqrt{N}$, which is the optimal iteration number to find a specific element from a set of N elements. As was described before in Section 3.1, if a successful search occurs, then a new solution takes place as the new threshold and the search process restarts.

The pseudocode of Dürr and Høyer's algorithm is presented bellow. This corresponds to an interpretation that has been described by Baritompa, Bulger and Wood [5] which is based on the proposed GAS algorithm, where the parameter k represents the search process count.

Algorithm 2 Dürr and Høyer's Algorithm

1: Randomly choose x from the decision space.
2: Set $x_1 \leftarrow x$
3: Set $y_1 \leftarrow f(x_1)$.
4: Set $m \leftarrow 1$.
5: Choose a value for the parameter λ (8/7 is suggested).
6: For $k = 1, 2, \ldots$ until termination condition is met, do:
(a) Choose a random rotation count r_k uniformly from $\{0, \ldots, \lceil m - 1 \rceil\}$.
(b) Perform a Grover search of r_k iterations on $f(x)$ with threshold y_k, and denote the outputs by x and y.
(c) If $y < y_k$ set $x_{k+1} \leftarrow x$, $y_{k+1} \leftarrow y$ and $m \leftarrow 1$; otherwise, set $x_{k+1} \leftarrow x_k$, $y_{k+1} \leftarrow y_k$ and $m \leftarrow \min\{\lambda m, \sqrt{N}\}$.

Chapter 4

Grover's Adaptive Search for Multiobjetive Optimization

4.1 Multiobjective Optimization

The goal of a multiobjective optimization problem is to optimize several objectives (at least two) at the same time. These objectives are frequently in conflict, and therefore, there may exists several "optimal" solutions. The set of optimal solutions is known as a Pareto-optimal set, where solutions provide the best compromise relations between the objective functions considering the entire feasible decision space [7] [8].

The feasible decision space is the set of all feasible solutions, which are compared against each other by means of the *Pareto dominance relation*. Indeed, the relation makes possible to determine if a solution is dominated or not by another solution. One solution Y is dominated by another solution Y', denoted by $Y' \prec Y$, if Y' is better or equal in every objective function and strictly better in at least one objective function. Thus, a non-dominated solution is Pareto-optimal if there is no solution that dominates it. The set of all non-dominated solutions corresponds to the Pareto-optimal set and its mapping to the objective space is known as the Pareto Front. Furthermore, a solution Y is said to be non-comparable with respect to a solution Y'' and it is denoted $Y \sim Y''$ if neither Y dominates Y'' ($Y \not\prec Y''$) nor Y'' dominates Y ($Y'' \not\prec Y$) [7].

4.2 Multiobjective Grover Adaptive Search (MOGAS)

In this thesis, a new adaptative search algorithm based on the Dürr and Høyer's algorithm is proposed and is named as Multiobjetive Grover Adaptive Search (MOGAS). This proposed new quantum algorithm is used for the first time in multiobjective optimization problems. MOGAS uses two different oracle operators based on the Pareto dominance relation. The first oracle marks all the non-dominanted solutions with respect to a known (current) solution. The second oracle marks all the non-dominated and non-comparable solutions. These oracles are based on the boolean functions

$$h_1(x) = \begin{cases} 1, & \text{if } \mathbf{F}(x) \prec \mathbf{Y} \\ 0, & \text{otherwise} \end{cases}, \tag{4.1}$$

$$h_2(x) = \begin{cases} 1, & \text{if } \mathbf{F}(x) \prec \mathbf{Y} \vee \mathbf{F}(x) \sim \mathbf{Y} \\ 0, & \text{otherwise} \end{cases}, \tag{4.2}$$

where x is a feasible solution of the decision space, $\mathbf{F}(x)$ is a vector where each element represents the value of the objective function with respect to solution x, and $\mathbf{Y}$ is a general vector where each element is the value of each objective function for the current known solution.

Then, the first oracle marks a non-dominated solution if and only if the boolean function $h_1(x) = 1$ and the second oracle marks a non-dominated or non-comparable solution if and only if $h_2(x) = 1$.

The pseudocode of the proposed MOGAS algorithm, where the parameter k represents the search process count, is presented bellow:

Algorithm 3 MOGAS Algorithm

1: Randomly choose x from the decision space.
2: Set $S \leftarrow \{x_1 \leftarrow x\}$
3: Set $\mathbf{Y_1} \leftarrow \mathbf{F}(x_1)$.
4: Set $m \leftarrow 1$.
5: Choose a value for the parameter λ (8/7 is suggested).
6: For $k = 1, 2, \ldots$ until termination condition is met, do:
(a) Choose a random rotation count r_k uniformly from $\{0, \ldots, \lceil m - 1 \rceil\}$.
(b) Perform a Grover search of r_k iterations on $\mathbf{F}(x)$ with threshold $\mathbf{Y_k}$, and denote the outputs by x and $\mathbf{Y}$.
(c) If $\mathbf{Y} \not\prec \mathbf{Y_k}$ set $x_{k+1} \leftarrow x_k$, $\mathbf{Y_{k+1}} \leftarrow \mathbf{Y_k}$ and $m \leftarrow \min\{\lambda m, \sqrt{N}\}$.
Otherwise, set $m \leftarrow 1$, $x_{k+1} \leftarrow x$, $\mathbf{Y_{k+1}} \leftarrow \mathbf{Y}$ and with respect to all elements of the set S, where $j = 1, \ldots, |S|$, do:
If $\exists\ x_j \in S : \mathbf{F}(x) \prec \mathbf{F}(x_j)$, then, set $S \leftarrow S - \{x_j\}$ and finally set $S \leftarrow S \cup \{x\}$.
7: Set $PF \leftarrow \{\mathbf{F}(x_j) : j = 1, \ldots, |S|\}$, $\forall\ x_j \in S$.

The operation of MOGAS is based on the oracle operator. Using the oracle, MOGAS can find a non-dominated solution with respect to a known solution. In this way, the algorithm can reach the Pareto-optimal set by finding new non-dominated solutions at each iteration. Therefore, with the search process it is possible to incorporate a new element in the Pareto-optimal set or replace some old elements from it each time a non-dominated solution is found.

Chapter 5

Experimental Results

A general purpose quantum computer seems to currently be out of reach at the present. Nevertheless, the basic ideas of quantum algorithms can be fully explored using linear algebra, and therefore, computational performances of quantum algorithms are possible by executing linear algebra operations.

To verify the effectiveness of the proposed algorithm, we have tested it by means of simulations against one of the most cited optimization algorithms for multiobjective problems, the Non-dominated Sorting Genetic Algorithm - version two [7] [8] known as NSGA-II. The tests were made considering some biobjective problems based on the well known ZDT test suite [9] and on randomly generated instances. The randomly generated problems (RG) consist of a random selection of numbers from a set of integer numbers between 1 and 1000 for each of the two objective functions. Then, three different suites of this type of random instances were established for testing. With respect to the ZDT test suite, the ZDT1, ZDT3 and ZDT4 were selected considering two objective functions. For each of these functions, a total of twenty decision variables ($\mathbf{m_d} = 20$) were used. Each decision variable is a random real number from the interval $[0, 1]$.

The decision space for each instance consist in a set of $1024 = 2^{10}$ points. The amount of points is based on the number of qubits ($n = 10$) selected for the proposed MOGAS algorithm. Since the problem has two objective functions ($\mathbf{f_1}, \mathbf{f_2}$) that should be minimized, the vector dimension (for $\mathbf{F}(x)$ and $\mathbf{Y}$) is $p = 2$. Table 5.1 presents the main characteristics of the considered test suites.

Table 5.1: Test suites used for the experiments.

Function	$\mathbf{m_d}$	$\mathbf{x_i}, \mathbf{x_j}$ $i, j = 1, \ldots, 2^{10}$	$\mathbf{f_1}$	$\mathbf{f_2}$
$\mathrm{RG}_{1,2,3}$	–	$x_i \in [1, 10^3]$ $x_j \in [1, 10^3]$ $x_i, x_j \in \mathbb{N}$	x_i	x_j
ZDT1	20	$x_i \in [0, 1]$	x_i	$g_1(x_i)\lfloor 1 - \sqrt{x_i/g_1(x_i)}\rfloor$, $g_1(x_i) = 1 + 9\frac{(\sum_{k=2}^{m} x_{i_k})}{(m-1)}$
ZDT3	20	$x_i \in [0, 1]$	x_i	$g_3(x_i)\lfloor 1 - \sqrt{x_i/g_3(x_i)} - \frac{x_i}{g_3(x_i)} sin(10\pi x_i)\rfloor$, $g_3(x_i) = 1 + 9\frac{(\sum_{k=2}^{m} x_{i_k})}{(m-1)}$
ZDT4	20	$x_i \in [0, 1]$	x_i	$g_4(x_i)\lfloor 1 - \sqrt{x_i/g_4(x_i)}\rfloor$, $g_4(x_i) = 1 + 10(m-1) + \sum_{k=2}^{m}(x_{i_k}^2 - 10cos(4\pi x_{i_k}))$

The testing procedure was based on ten executions of both algorithms, that is, MOGAS (considering the two different types of oracles) and NSGA-II, over all test suites. At each execution, the termination criteria was to complete two hundred generations (with a population size equal to fifty, what means $200 * 50 = 10000$ evaluations) for NSGA-II and a total of four hundred algorithm consultations for MOGAS, where an algorithm consultation is exactly one application of Grover's algorithm with r_k iterations on $\mathbf{F}(x)$ considering a threshold $\mathbf{Y_k}$, and denoting the outputs by x and its evaluation $\mathbf{Y} = \mathbf{F}(x)$ respectively.

The hypervolume was used as the metric for the comparison of the results, considering that it is the most used comparison metric in multiobjective optimization [8]. The hypervolume is an indicator used in the multiobjective optimization of evolutionary algorithms to evaluate the performance of the search. It is based on a function that maps the set of Pareto-optimal to a scalar with respect to a reference point.

In tables 5.2, 5.3 and 5.4, the obtained experimental results from the testing procedure are presented considering the hypervolume.

The tables are composed of six columns that correspond to each test suite and a column for the order of execution. In these six columns, the result of the hypervolume metric in percentage for each execution is given. In this way, each row summarizes the experimental results for every test suite with respect to a specific execution order denoted in the left column. Also, in the last row, an average of these ten executions for all test suites is presented.

Tables 5.2 and 5.3 correspond to results obtained for MOGAS using h_1 and h_2 respectively. Table 5.4 corresponds to results obtained using NSGA-II with a population size equal to fifty.

From the experimental results obtained, MOGAS presents similar results compared to NSGA-II with a population size of fifty with respect to RG instances; in most cases, however, MOGAS delivers better or equal results. Nevertheless, considering the structured ZDT test suites and compared to NSGA-II results, only MOGAS based on the boolean function h_1 as oracle presents equal or better results, whereas MOGAS based on the boolean function h_2 as oracle presents nearly equal results but not equal or better results.

Nevertheless, considering the algorithm consultations of MOGAS as a single evaluation of the objective function, the results present an important fact to note: MOGAS used only four hundred evaluations of the objective function vector $\mathbf{F}(x)$, whereas NSGA-II (with a population size of fifty) used 10000 (pop$*$gen$= 50 * 200$) evaluations of the same vector to deliver similar results.

Tables 5.5, 5.6 and 5.7 summarize the average results of both MOGAS algorithms and NSGA-II, considering objective function evaluations. These tables are composed of six columns that correspond to each test suite and a column for the number of evaluations. In these six columns, the average results of the hypervolume metric in percentage corresponding to ten executions are presented. In this way, each row summarizes the average results for every test suite with respect to a specific number of evaluations given in the

left column.

The obtained experimental results are presented in figures 5.1 to 5.6 as the performance in the hypervolume metric (in percentage) versus the number of evaluations of the objective function vector.

Considering the average number of iterations of the Grover operator needed for MOGAS using both oracles, the presented experimental results reveal that MOGAS using h_2 as oracle, in most cases, uses less iterations compared to MOGAS using h_1 as oracle.

Certainly, the oracle based on h_2 finds more solutions from the decision space. Therefore, the probability to change the threshold at every consultation performed increases. This way, the parameter m is set to one more often and the iteration number chosen corresponds to a lower number. Thus, the total number of iterations for MOGAS using h_2 is smaller than the one using oracle based on h_1.

Tables 5.8 to 5.13 summarize the average results of the number of iterations used by MOGAS, considering the number of times the Grover operator is invoked. These tables have two columns that correspond to each different type of oracle and a column for the number of evaluations. In these two columns, the average results of the number of iterations corresponding to ten executions are presented. In this way, each row summarizes the average result for both oracles with respect to a specific number of evaluations presented in the left column.

Table 5.2: Results of the testing procedure - MOGAS, with the hypervolume measured in percentage (After 400 consultations and the oracle based on the boolean function h_1).

Test suites / # Executions	RG_1 [%]	RG_2 [%]	RG_3 [%]	ZDT1 [%]	ZDT3 [%]	ZDT4 [%]
1	98.4	98.4	98.8	51.4	57	61.2
2	99	98.4	99.1	52.8	57.3	60.2
3	99	98.6	98.6	52.7	58.1	60.5
4	99	98.7	99.1	52.1	58.2	60.7
5	98.9	98.9	99.1	52.7	58.2	60.5
6	99.1	98.5	98.7	52.4	58.3	60.7
7	98.9	98.5	99	52.9	57.1	61.3
8	99.1	98.7	99.1	53.1	56.5	58.8
9	98.9	98.7	99.1	52.3	58.2	59.7
10	98.9	98.7	98.4	53.2	57.7	58.9
Average	99	99	99	53	58	60

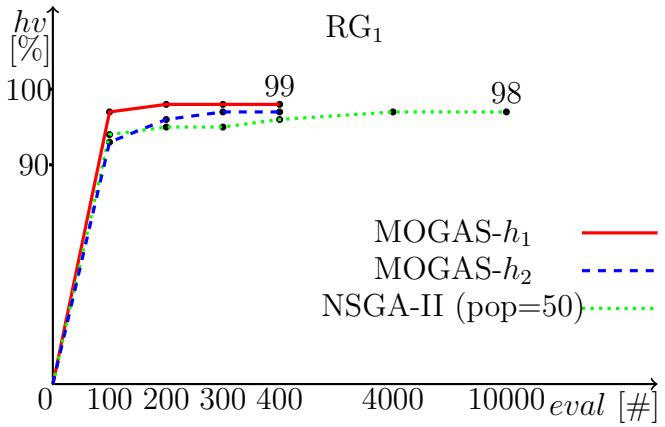


Figure 5.1: Graphs of the hypervolume metric in percentage (hv) versus the number of evaluations of the objective function vector ($eval$) made by each algorithm (MOGAS and NSGA-II) with respect to the RG_1 suite test.

Table 5.3: Results of the testing procedure - MOGAS, with the hypervolume measured in percentage (After 400 consultations and the oracle based on the boolean function h_2).

Test suites / # Executions	RG_1 [%]	RG_2 [%]	RG_3 [%]	ZDT1 [%]	ZDT3 [%]	ZDT4 [%]
1	98	98.7	99	51	55.4	57.4
2	96.6	98.4	99.1	49	56.4	59.9
3	98.7	98.7	99.1	50.7	53.2	60.4
4	97.3	98.2	99.2	50.4	53.8	61.1
5	98.7	98.7	98.7	50.9	55.4	55.1
6	99.2	98.9	96.7	50.7	54.2	58
7	98.4	98.7	99	49.7	52.7	59.8
8	98.8	98.2	98.6	49	52	59.1
9	98.1	98.8	97.6	47.7	52.3	61.3
10	97	98.6	99.2	49.1	53.2	58.9
Average	98	99	99	50	54	59

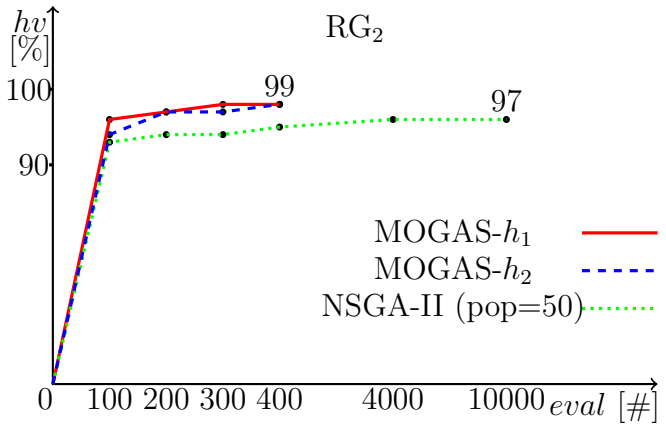


Figure 5.2: Graphs of the hypervolume metric in percentage (hv) versus the number of evaluations of the objective function vector ($eval$) made by each algorithm (MOGAS and NSGA-II) with respect to the RG_2 suite test.

Table 5.4: Results of the testing procedure - NSGA-II, with the hypervolume measured in percentage (After 200 generations and a population size equal to 50).

Test suites / # Executions	RG_1 [%]	RG_2 [%]	RG_3 [%]	ZDT1 [%]	ZDT3 [%]	ZDT4 [%]
1	98.1	97.3	98.4	52.1	55.7	60.2
2	99	96.8	97.7	51.2	56.4	60.1
3	97.8	98.6	97.5	51.1	56.9	60.1
4	97.1	98.1	99.1	51.9	55.6	59.6
5	97.5	97.1	98.3	51.9	58.4	60.4
6	98.2	96.6	98.5	52.7	56.6	59.7
7	97.9	98.2	98.8	53.2	57.6	60.7
8	97.6	97.7	98.8	51.5	57.2	60.6
9	97.8	96.1	98.8	51.9	55.9	60
10	98.7	97.3	98.5	52.8	58	59.6
Average	98	97	98	52	57	60

Table 5.5: Average Results of the testing procedure - MOGAS, with the hypervolume measured in percentage (From 100 to 400 evaluations and the oracle based on the boolean function h_1).

Test suites / # Evaluations	RG_1 [%]	RG_2 [%]	RG_3 [%]	ZDT1 [%]	ZDT3 [%]	ZDT4 [%]
100	97.7	97.4	98.2	49.2	54.8	57.9
200	98.5	98.2	98.6	51.4	56.8	58.9
300	98.9	98.5	98.8	52.3	57.5	59.7
400	98.9	98.6	98.9	52.5	57.7	60.2

Table 5.6: Average Results of the testing procedure - MOGAS, with the hypervolume measured in percentage (From 100 to 400 evaluations and the oracle based on the boolean function h_2).

Test suites / # Evaluations	RG_1 [%]	RG_2 [%]	RG_3 [%]	ZDT1 [%]	ZDT3 [%]	ZDT4 [%]
100	94.2	95.1	92.9	44.7	47	55.1
200	97	97.7	97	47.6	50.2	57.4
300	97.9	97.7	98.2	48.7	52.7	58.4
400	98.1	98.6	98.6	49.8	53.9	59.1

Table 5.7: Average Results of the testing procedure - NSGA-II, with the hypervolume measured in percentage (From 100 to 10000 evaluations corresponding to a population size equal to 50).

Test suites / # Evaluations	RG_1 [%]	RG_2 [%]	RG_3 [%]	ZDT1 [%]	ZDT3 [%]	ZDT4 [%]
100	94.6	94.5	95.6	47.4	51	54.9
200	95.9	95.1	96.2	49	51.9	56.8
300	96.5	95.3	96.5	49.4	53	57.3
400	96.5	95.9	96.8	49.9	53.4	57.7
4000	97.7	97.2	98.1	51.5	56.5	60
10000	98	97.4	98.4	52	56.8	60.1

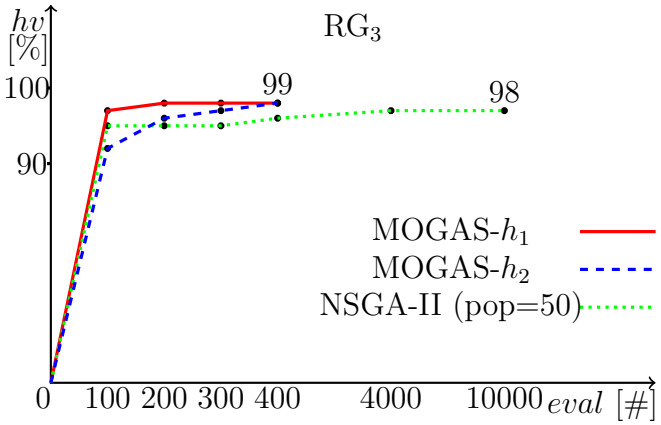


Figure 5.3: Graphs of the hypervolume metric in percentage (hv) versus the number of evaluations of the objective function vector ($eval$) made by each algorithm (MOGAS and NSGA-II) with respect to the RG_3 suite test.

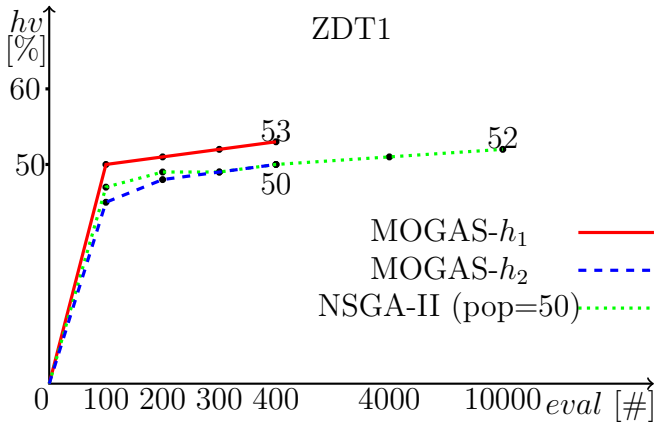


Figure 5.4: Graphs of the hypervolume metric in percentage (hv) versus the number of evaluations of the objective function vector ($eval$) made by each algorithm (MOGAS and NSGA-II) with respect to the ZDT1 suite test.

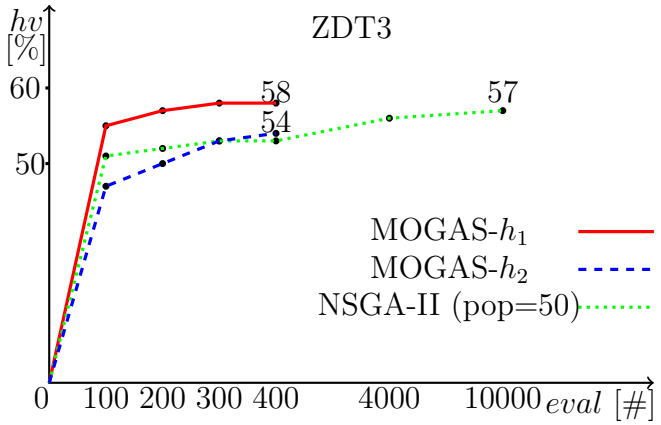


Figure 5.5: Graphs of the hypervolume metric in percentage (hv) versus the number of evaluations of the objective function vector ($eval$) made by each algorithm (MOGAS and NSGA-II) with respect to the ZDT3 suite test.

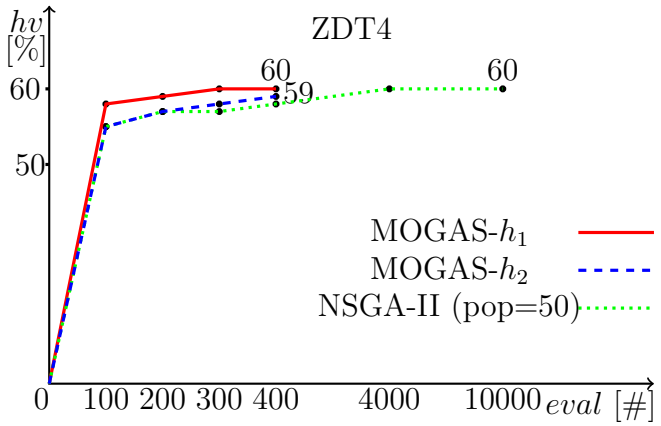


Figure 5.6: Graphs of the hypervolume metric in percentage (hv) versus the number of evaluations of the objective function vector ($eval$) made by each algorithm (MOGAS and NSGA-II) with respect to the ZDT4 suite test.

Table 5.8: Average iteration numbers used on the RG1 (From 100 to 400 evaluations).

Oracle Types / # Evaluations	MOGAS-h_1 [#]	MOGAS-h_2 [#]
100	815	352
200	2162	1299
300	3588	2277
400	5149	3474

Table 5.9: Average iteration numbers used on the RG2 (From 100 to 400 evaluations).

Oracle Types / # Evaluations	MOGAS-h_1 [#]	MOGAS-h_2 [#]
100	748	343
200	2078	991
300	3483	2373
400	4975	3485

Table 5.10: Average iteration numbers used on the RG3 (From 100 to 400 evaluations).

Oracle Types / # Evaluations	MOGAS-h_1 [#]	MOGAS-h_2 [#]
100	838	349
200	1952	888
300	3344	1947
400	4848	3274

Table 5.11: Average iteration numbers used on the ZDT1 (From 100 to 400 evaluations).

Oracle Types / # Evaluations	MOGAS-h_1 [#]	MOGAS-h_2 [#]
100	219	280
200	602	801
300	1182	1517
400	2094	2385

Table 5.12: Average iteration numbers used on the ZDT3 (From 100 to 400 evaluations).

Oracle Types / # Evaluations	MOGAS-h_1 [#]	MOGAS-h_2 [#]
100	255	259
200	863	668
300	1635	1319
400	2571	2247

Table 5.13: Average iteration numbers used on the ZDT4 (From 100 to 400 evaluations).

Oracle Types / # Evaluations	MOGAS-h_1 [#]	MOGAS-h_2 [#]
100	407	410
200	1260	1255
300	2676	2273
400	3858	3474

Chapter 6

Conclusions

This thesis compared two different types of oracles used in a quantum algorithm for multiobjective optimization problems. The presented multiobjective quantum algorithm, called MOGAS, is a natural extension of previous quantum algorithms for single-objective optimization based on Grover's search method. The experimental results of this work suggests that MOGAS (considering both types of oracles) was not only an effective approach for multiobjective optimization problems, but it was also efficient as was observed when MOGAS was compared against NSGA-II, which is one of the most cited multiobjective optimization algorithms [8]. In most of the studied cases, MOGAS obtained better or equal results in average after comparing it against NSGA-II for the same number of executions especially with respect to the oracle based on the boolean function h_1; in regard of h_2, the results presented in this work are almost equal compared to NSGA-II when considering hypervolume.

On the other hand, considering that MOGAS is a quantum black-box optimization algorithm, it is important to note that MOGAS only required around the square root of the number of objective function evaluations performed by NSGA-II, that is certainly very close to theoretically findings for search problems denoted by [3].

In spite of the simple adaptive strategy used by MOGAS (considering both types of oracles), the experimental results of this work present a remarkable improvement over NSGA-II. Therefore, the presented experimental results show the efficiency of a simple quantum algorithm with respect to a classical more elaborated algorithm.

Another interesting fact to note is the difference between the number of

iterations used by MOGAS. The oracle based on the boolean function h_2, in most cases, employed a smaller number of iterations than the one using h_1. Hence, h_2 is more efficient than h_1, which represents a saving in the number of queries to the quantum oracle.

For future research, it is interesting to study other different definitions of oracles for multiobjective problems. It is also very important to lay some theoretical foundations that can show the convergence of MOGAS to the set of Pareto-optimal solutions.

Bibliography

[1] Nielsen, M. A. and Chuang, I. L., *Quantum computation and quantum information*, Cambridge university press, 2010.

[2] Shor, P. W., *Algorithms for quantum computation: Discrete logarithms and factoring*, In Foundations of Computer Science, 1994 Proceedings, 35th Annual Symposium on (pp. 124-134). IEEE, 1994.

[3] Grover, L. K., *A fast quantum mechanical algorithm for database search*, In Proceedings of the twenty-eighth annual ACM symposium on Theory of computing (pp. 212-219), ACM, 1996.

[4] Dürr, C. and Høyer, P., *A quantum algorithm for finding the minimum*, arXiv preprint quant-ph/9607014, 1996.

[5] Baritompa, W. P., Bulger, D. W., and Wood, G. R., *Grover's quantum algorithm applied to global optimization*, SIAM Journal on Optimization, 15(4), 1170-1184, 2005.

[6] Barán, B. and Villagra, M., *Multiobjective Optimization in a Quantum Adiabatic Computer*. In Proceedings of the 42nd Latin American Conference on Informatics (CLEI), Symposium on Theory of Computation, ENTCS 329, pp.27–38, Valparaiso-Chile, 2016.

[7] von Lücken, C., Barán, B. and Brizuela, C., *A survey on multi-objective evolutionary algorithms for many-objective problems*, Computational Optimization and Applications, 58(3), 707-756, 2014.

[8] Riquelme, N., Baran, B., and von Lücken, C., *Performance metrics in multi-objective optimization*, Computing Conference (CLEI), 2015 Latin American. IEEE, 2015.

[9] Chase, N., et al., *A benchmark study of multi-objective optimization methods*, BMK-3021, Rev 6, 2009.

[10] Rao, S. S. and Rao, S. S., *Engineering optimization: theory and practice*, John Wiley and Sons, 2009.

Appendix A

Quantum Computation

Building a working quantum computer is one of the grand challenges of the 21st century. A quantum computer exploits quantum phenomena in order to rapidly solve complex computational problems in nature. It is not believed to help in solving efficiently **NP**-complete problems [1], nevertheless, it does gives a speed-up in the solution of several computational problems.

In this chapter, a review of the basic mathematical concepts of quantum computation is given. In Section A.1 the basic unit of quantum information is defined and in section A.2 we explain the basic buildings blocks for quantum computation. For a complete introduction to quantum computation, we refer the interested reader to the book by Nielsen and Chuang [1].

A.1 Quantum Bits and Registers

A.1.1 The Qubit

A *classical* bit holds two states: 1 or 0. To extend this notion to quantum bits, or just qubits, define it as a vector on a Hilbert space. We denote vectors using the Dirac notation, i.e., $|\psi\rangle$ is a vector in some vector space, and $\langle\psi|$ represents its dual in the dual vector space of functionals.

Definition A.1.1. Let $\mathcal{H} = span\{|0\rangle, |1\rangle\}$ be a Hilbert space equipped with a ℓ_2-norm. A qubit is a vector $|\psi\rangle \in \mathcal{H}$ over the complex field defined as

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle\,,$$

where $|\alpha|^2 + |\beta|^2 = 1$.

The set $\{|0\rangle, |1\rangle\}$ is known as the *computational basis* which is an orthonormal basis for $\mathcal{H}$. These vectors can be written in matrix notation as

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad \text{and} \quad |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix},$$

which is a canonical basis for a 2 dimensional space.

A qubit could be considered as a 1-bit register holding the two states $|0\rangle$ and $|1\rangle$ at the same time. In general it is known as a *superposition state*. Contrarily to a classical bit, which could be queried obtaining a deterministic answer (either 0 or 1), when we query a qubit we get a probabilistic answer. This answer would be 0 with probability $|\alpha|^2$ or 1 with probability $|\beta|^2$.

A.1.2 Registers

In order to generalize a system to n qubits we need to introduce tensor vector spaces.

Definition A.1.2. Let $\mathcal{H} = span\{|x\rangle : x \in \{0,1\}^n\}$ be a Hilbert space with a ℓ_2-norm. An n qubit state is a vector $|\psi\rangle \in \mathcal{H}$ defined as

$$|\psi\rangle = \sum_{x \in \{0,1\}^n} \alpha_x |x\rangle,$$

where $1 = \sum_x |\alpha_x|^2$, and $|x\rangle = |x_1\rangle \otimes \cdots \otimes |x_n\rangle$ with each $|x_i\rangle \in span\{|0\rangle, |1\rangle\}$.

Observe that the number of possible basis states for a n-qubit register is 2^n and then it scales exponentially faster. This seems to store an exponential amount of information. However, by the Holevo bound [1], in order to recover information faithfullly, n is the maximum number of bits one can store in an n-qubit register.

A.1.3 Operations

Algorithms for quantum computers are built as a sequence of quantum operations (or gates, in analogy to classical circuits) acting on qubit registers. These quantum operations are essentially unitary operations defined over some Hilbert space.

1-qubit Operations

Maybe the simplest operation is the NOT operator. Given a qubit $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, it interchanges the amplitudes in the following way

$$NOT(\alpha|0\rangle + \beta|1\rangle) = \beta|0\rangle + \alpha|1\rangle .$$

This operation is easily defined as

$$\begin{aligned} NOT &\equiv |0\rangle\langle 1| + |1\rangle\langle 0| \\ &= \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}. \end{aligned}$$

Another important quantum gate is the Hadamard operation denoted with H. It is basically a rotation operation and agrees exactly with a Fourier transform on a 2-dimensional space. It acts in the following way

$$H|i\rangle = \frac{1}{\sqrt{2}}(|0\rangle + (-1)^i|1\rangle).$$

This operation is defined as

$$\begin{aligned} H &\equiv \frac{1}{\sqrt{2}} \sum_{i,j\in\{0,1\}} (-1)^{i\cdot j}|i\rangle\langle j| \\ &= \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}. \end{aligned}$$

2-qubits Operations

A very popular generalization of the NOT operation is CNOT, which stands for controlled-NOT. This gate flips a target qubit if and only if another qubit known as control-bit is set to 1, and it is denoted by $CNOT$. Its actions are

$$\begin{aligned} CNOT : |00\rangle &\longrightarrow |00\rangle \\ |01\rangle &\longrightarrow |01\rangle \\ |10\rangle &\longrightarrow |11\rangle \\ |11\rangle &\longrightarrow |10\rangle . \end{aligned}$$

Here, the first qubit controls the fliping of the second qubit. It resembles exactly a conditional statement for quantum computation. Formally it is defined as

$$\begin{aligned} CNOT &\equiv \sum_{i,j\in\{0,1\}} |0\rangle\langle 0| \otimes |j\rangle\langle j| + |1\rangle\langle 1| \otimes |1-j\rangle\langle j| \\ &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \end{aligned}$$

A second very important 2-quibts operation is SWAP. This operation interchanges the state of two different qubits

$$SWAP\,|i,j\rangle = |j,i\rangle\,.$$

The operation is formally defined as

$$\begin{aligned} SWAP &\equiv \sum_{i,j\in\{0,1\}} |i\rangle\langle j| \otimes |j\rangle\langle i| \\ &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \end{aligned}$$

n-qubits Operations

It is easy to see that for an n-qubit system, the number of rows and columns in the matrix representation of the operators grow exponentially faster. To define operations on n-qubits, the braket notation is a convenient tool.

A generalization of the Hadamard gate to n qubits is known as the Walsh-Hadamard transform denoted as W. It can be written as

$$W \equiv \frac{1}{\sqrt{2^n}} \sum_{x,y\in\{0,1\}^n} (-1)^{x\cdot y} |x\rangle\langle y|\,.$$

Further generalizations of other operations like CNOT, controlled-SWAP, etc. is straightforward [1].

A.2 Measurements and Observables

Evolution in a closed quantum system is unitary. The previous section presented different types of unitary operators. However, to read-out the result of a quantum computation we need to be able to measure the state of the system. In this section we will see a basic introduction to the process of measurement of a quantum system.

First we start by defining an *observable*. In simple terms, this is the dynamic variable we want to measure, e.g., velocity, energy, spin, etc. Normally, in quantum computation we want to know if a qubit is in state $|0\rangle$ or $|1\rangle$. Formally, an observable is a Hermitian operator that acts on the Hilbert space of the system whose eigenvalues and eigenvectors correspond to the values and states of the dynamic variable.

To be able to measure an observable we need to make a *measurement*. Formally, a measurement is a set of linear operators $\{M_m\}$ that acts on the Hilbert space of the system being observed. The index m refers to the outcome of the measurement.

Say that the system is in state $|\psi\rangle$. When we measure it, the probability that m occurs is

$$P(m) = \langle\psi| M_m^\dagger M_m |\psi\rangle ,$$

and the state of the system after measurement is

$$\frac{M_m |\psi\rangle}{\sqrt{P(m)}}.$$

In this thesis, we deal with a type of measurements called *projective measurements*. We want each m to be an eigenvalue of an observable. Given an obervable O, its spectral decomposition is

$$O = \sum_m m P_m,$$

where P_m is a projection onto the subspace with eigenvalue m. Thus, the probability of getting m is

$$P(m) = \langle\psi| P_m |\psi\rangle ,$$

and the corresponding state of the system is

$$\frac{P_m |\psi\rangle}{\sqrt{P(m)}}.$$

Resumen Extendido en Español

Introducción

La optimización se aplica frecuentemente en disciplinas tan diversas como la ingeniería, la economía, la administración de recursos y procesos, la industria, las políticas gubernamentales, entre otras; principalmente avocada a resolver problemas concernientes a la toma de decisiones respecto a una función objetivo (criterios de diseño, costos, riesgos financieros, utilidades, etc.) a fin de encontrar un conjunto de valores óptimos basados en variables de decisión con las que en principio sería posible minimizar o maximizar la función objetivo considerada, atendiendo a que la misma se encuentra sujeta a ciertas restricciones bajo determinadas circunstancias [10].

Desde la década de los 80 del siglo pasado ha surgido una nueva disciplina de estudio, la computación cuántica, nacida a partir de las ideas de Paul Benioff, Richard Feynman, Yuri Manin y otros, quienes notaron la incapacidad básica de emplear a los computadores clásicos en las simulaciones de ciertos fenómenos cuánticos. Esto motivó la idea de construir computadores más eficientes cuyo funcionamiento se basa en las mismas leyes del mundo cuántico, incorporando fenómenos tales como la superposición, la interferencia y el entrelazamiento cuántico [1].

Sin embargo, aún con la idea concebida de dichos computadores cuánticos, no fue hasta mediados de los 90 que Peter Shor [2] descubrió un algoritmo cuántico que teóricamente permite la factorización de números enteros en tiempo polinomial y con ello demostró la eficiencia de los algoritmos cuánticos y su enorme ventaja potencial sobre los algoritmos clásicos. Otro avance considerable surgió con el desarrollo del algoritmo cuántico de búsqueda por Grover [3], con el cual se reduce la complejidad computacional de la búsqueda de un orden de $O(N)$ para computadores clásicos, a un orden de $O(\sqrt{N})$ para computadores cuánticos, con lo que se obtiene una mejora considerable.

Fue a partir del algoritmo de búsqueda de Grover que varios investigadores propusieron diversos métodos de aplicar dicho algoritmo a la optimización global de funciones y en general a la búsqueda de una solución óptima. Dürr y Høyer [4], presentaron un algoritmo cuántico basado en el algoritmo de Grover aplicado al problema de búsqueda del mínimo de una función objetivo.

Por otra parte, Baritompa, Bulger y Wood [5], propusieron un método basado en el algoritmo de Grover como base para implementar un algoritmo de optimización global adaptativo. Mientras que Barán y Villagra [6], introdujeron el primer algoritmo cuántico para optimización combinatoria multiobjetivo basado en un computador cuántico adiabático.

En el presente trabajo se expone una alternativa de aplicación del algoritmo de Grover basado en un algoritmo de búsqueda adaptativa para problemas multiobjetivos que llamaremos *MOGAS*, del inglés *Multiobjective Grover Adaptive Search*. Dicho algoritmo basa su funcionamiento considerando dos tipos diferentes de óraculos. El primero de estos oráculos marca todas las soluciones no dominadas en un contexto de muchos objetivos contradictorios a partir de un umbral establecido. El segundo oráculo marca las soluciones no dominadas así como también las soluciones no comparables a partir de un umbral establecido. De esta forma, esta tesis propone un nuevo algoritmo aplicado a una búsqueda adaptativa para problemas multiobjetivos que posibilite la obtención de una nueva solución no dominada. Esta nueva propuesta, apoyada en resultados experimentales sugiere que MOGAS en promedio es similarmente eficiente como el algoritmo evolutivo NSGA-II. Sin embargo, utilizando menos evaluaciones de las funciones objetivos.

Computación Cuántica

Para empezar se ha establecido una diferencia fundamental entre bits clásicos y bits cuánticos o qubits. Respecto a los bit clásicos, estos son considerados estados bases de la computación clásica constituyéndose en la forma de representar las informaciones para llevar a cabo los procesos de lectura/escritura de datos, la decodificación de instrucciones, las operaciones aritméticas entre otras alternativas. En general, se componen de unos (1) y ceros (0) referenciados físicamente por lo general a altos (1) y bajos (0) niveles de voltaje [1].

Los qubits se fundamentan en fenómenos cuánticos y se definen como un estado en superposición. Un estado en superposición es una combinación lineal de los elementos que componen la base computacional. Sin embargo,

a diferencia de los bits clásicos, cuando un qubit es medido, el qubit colapsa al estado $|0\rangle$ o $|1\rangle$ con cierta probabilidad [1]. De modo a dar un ejemplo concreto de la física de un qubit, es importante pensar por ejemplo en dos diferentes polarizaciones posibles de un fotón y podemos considerar una de las direcciones de polarización como el estado cuántico $|1\rangle$ y a su dirección perpendicular como el estado cuántico $|0\rangle$. La polarización del fotón puede encontrarse en un estado de superposición, con una amplitud normalizada asociada a cada estado cuya magnitud determina la probabilidad de que alguna de las polarizaciones sea observada, lo cual ocurre solo cuando se realiza una medición sobre el fotón. En dicho instante, el estado de superposición colapsa en uno de los dos estados posibles [1].

En la computación cuántica, los estados cuánticos se apoyan en el álgebra lineal de los espacios de Hilbert y por lo tanto son representados por medio de vectores sobre el campo de los números complejos. Así como en la computación clásica donde los datos están constituidos por cadenas de bits, identificamos una secuencia de estados cuánticos utilizando el producto tensorial entre vectores [1].

Por otra parte, mientras que en la computación clásica la aplicación de circuitos lógicos compuestos por una secuencia de compuertas lógicas equivaldría a un algoritmo clásico, en la computación cuántica los circuitos cuánticos se basan en operadores cuánticos o transformaciones lineales. La condición primordial de que los circuitos cuánticos sean del tipo unitario (el inverso de un operador cuántico es la transpuesta conjugada o adjunto de dicho operador) es debido a la característica reversible de todo sistema cuántico. Con una secuencia de dichos operadores cuánticos se construye un algoritmo cuántico [1].

Algoritmo de búsqueda de Grover

A partir de un conjunto de N elementos, desde el punto de vista de la computación clásica, hallar un elemento específico tomaría en el peor de los casos N intentos por lo que corresponde a una complejidad $O(N)$ [1]; siendo esta cota óptima en el caso clásico.

El algoritmo de búsqueda de Grover permite encontrar un elemento específico de un conjunto de N elementos con una complejidad $O(\sqrt{N})$. Esto es así porque se basa en la aplicación iterativa del operador de Grover $\mathbf{G}$, el cual a su vez se compone conjuntamente de un operador oráculo $\mathbf{O_G}$ y un

operador de desplazamiento de fase $\mathbf{W}$.

La cantidad de iteraciones r necesarias se obtiene a partir de la siguiente ecuación

$$r = \left\lfloor \frac{\pi}{4}\sqrt{N} \right\rfloor \approx \sqrt{N},$$

de allí la complejidad $O(\sqrt{N})$ [3].

El algoritmo de Grover emplea dos estados cuánticos, uno compuesto por n qubits $|0\rangle^{\otimes n}$, donde $2^n = N$, y otro compuesto por un qubit $|1\rangle$. El primero es transformado en un estado de superposición por medio de n transformadas Hadamard $\mathbf{H}^{\otimes n}$, donde [1]

$$|\psi\rangle = \mathbf{H}^{\otimes n}|0\rangle^{\otimes n} = \frac{1}{\sqrt{N}} \sum_{x\in\{1,0\}^n} |x\rangle.$$

Esto representa una combinación lineal equiprobable de todos los estados de la base. El segundo registro es transformado por la transformada Hadamard para cambiar a otra base computacional acorde a [1]

$$|-\rangle = \mathbf{H}|1\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}}.$$

El funcionamiento del algoritmo de Grover se basa en un oráculo que marca la solución buscada, representada por uno de los estados que componen el estado de superposición. Cabe destacar que el proceso de marcación se basa en el cambio de signo del estado deseado, solamente posible gracias a la interacción del mencionado operador oráculo y el segundo registro. Seguidamente, el operador de desplazamiento de fase permite el incremento de la amplitud asociada al estado solución mientras decrementa las amplitudes asociadas a todos los demás estados no solución. Esto sucede con cada iteración y es por ello que el algoritmo posibilita una interpretación geométrica relativa a la rotación de todos los estados que no son soluciones hacia el estado solución por cada aplicación del operador de Grover $\mathbf{WO_G}$ [1]. Otra importante versatilidad que ofrece el algoritmo de Grover es una generalización al hallar un conjunto de l-soluciones con la salvedad que las iteraciones r' requeridas del operador de Grover $\mathbf{WO_G}$ está dada por la siguiente ecuación [1]

$$r' = \left\lfloor \frac{\pi}{4}\sqrt{\frac{N}{l}} \right\rfloor \approx \sqrt{\frac{N}{l}}.$$

Nuevamente se aplica la interpretación geométrica antes mencionada, con la salvedad de que se trata de una rotación relativa de la combinación lineal de los estados que no son soluciones hacia la combinación lineal de los l-estados soluciones. Por lo tanto, tras las r' iteraciones, cada uno de los l-estados soluciones son equiprobables de ser observados [1].

Algoritmo Adaptativo de Grover Propuesto

El algoritmo de búsqueda adaptativa para problemas multiobjetivos propuesto, llamado *MOGAS* por sus siglas en inglés, está basado en la heurística propuesta por Dürr y Høyer [4]. Sin embargo, el algoritmo MOGAS utiliza dos tipos diferentes de oráculos cuyos principios de funcionamientos se basan en la relación de dominancia Pareto.

El primero de los mencionados oráculos, marca todas las soluciones no dominadas con respecto a una solución conocida, la cuál es establecida como umbral para cada búsqueda a realizarse. De la misma manera, el segundo oráculo marca todas las soluciones no dominadas con respecto a una solución conocida, al igual que el oráculo anteriormente mencionado. No obstante, éste segundo oráculo marca además las soluciones no comparables con respecto a la solución conocida establecida como umbral.

De esta manera, la operación del MOGAS está basada en la utilización de dichos oráculos. Con ellos, el MOGAS permite encontrar una nueva solución no dominada con respecto a una solución ya conocida. Así, el algoritmo propuesto posibilita la obtención de las soluciones que conforman el conjunto Pareto en cada iteración. Por lo tanto, con el proceso de búsqueda propuesto es posible incorporar una nueva solución no dominada o reemplazar una o varias soluciones ya encontradas del conjunto Pareto por otra que la domine.

Resultados Experimentales

Para verificar la efectividad del algoritmo propuesto, se ha llevado a cabo pruebas por medio de simulaciones en comparación con uno de los algoritmos evolutivos más citado para problemas multiobjetivos, conocido como NSGA-II [8]. Las pruebas fueron hechas considerando algunos problemas biobjetivos basados en las bien conocidas series de pruebas ZDT [9] y otras series de pruebas aleatoriamente generadas.

Estas series de pruebas aleatoriamente generadas (RG) consisten en la selección aleatoria de números a partir de un conjunto de números naturales que pertenecen al intervalo $[1, 1000]$ para cada una de las funciones objetivo. Con respecto a las series de pruebas ZDT, las series ZDT1, ZDT3 y ZDT4 fueron seleccionadas considerando dos funciones objetivos. Para cada función objetivo, se utilizaron veinte variables de decisión cuyos valores han sido asignados aleatoriamente a partir del intervalo $[0, 1]$ de números reales.

Las pruebas se basaron en diez ejecuciones con ambos algoritmos, esto es, el MOGAS (considerando los dos tipos diferentes de oráculos) y el NSGA-II, con relación a todas las series de pruebas mencionadas. En cada ejecución, el criterio de terminación fue completar doscientas generaciones (con una población de cincuenta) establecida para el NSGA-II y un total de cuatrocientas consultas al MOGAS, donde cada consulta equivale al proceso de búsqueda efectuada por el MOGAS con respecto a un valor de iteración dado.

La métrica seleccionada para comparar los resultados experimentales fue el hipervolumen, atendiendo a que es una de las métricas más empleadas en problemas multiobjetivos [8]. El hipervolumen es un indicador que se utiliza para evaluar el desempeño de la búsqueda en la optimización multiobjetivo de algoritmos evolutivos. Dicho indicador está basado en una función que mapea el conjunto Pareto-óptimo a un escalar con relación a un punto de referencia.

A partir de los resultados experimentales, se puede observar que el algoritmo MOGAS presenta resultados similares comparado con el NSGA-II (con una población de cincuenta) respecto a la calidad de la solución considerando el valor del hipervolumen. Con respecto a las series de pruebas aleatorias RG, en la mayoría de los casos MOGAS ofrece un mejor o similar resultado. No obstante, considerando las series de pruebas estructuradas ZDT se observó que el MOGAS aplicando el primero de los oráculos presenta resultados similares o mejores. Mientras que con el segundo de los oráculos presenta resultados próximos pero no similares o mejores.

Sin embargo, atendiendo al número de consultas realzadas por el MOGAS, considerándola como una evaluación de la función objetivo, los resultados presentan un importante hecho a remarcar: MOGAS solo emplea un total de cuatrocientas evaluaciones del vector función objetivo, mientras que el NSGA-II realiza un total de diez mil evaluaciones (pop$*$gen$=50 * 200$) sobre el mismo vector para la obtención de resultados similares. A su vez, los resultados experimentales revelan que el MOGAS empleando el segundo de

los oráculos en la mayoría de los casos requiere menos iteraciones comparado con el primero de los oráculos.

En conclusión, el algoritmo MOGAS propuesto ofrece similares resultados comparado con el NSGA-II considerando el valor del hipervolumen como métrica. No obstante, cabe resaltar que similares resultados son obtenidos por MOGAS luego de un total de 400 evaluaciones con respecto a 10000 evaluaciones de las funciones objetivos hechas por el NSGA-II. Lo que sugiere que MOGAS solo requiere aproximadamente la raiz cuadrado del número de evaluaciones requeridas por NSGA-II, siendo dicho valor muy próximo a lo determinado teóricamente por [3].