Proceedings of the Federated Conference on Computer Science and Information Systems pp. 421–428 DOI: 10.15439/2017F259 ISSN 2300-5963 ACSIS, Vol. 11

Comparison of two types of Quantum Oracles based on Grover's Adaptative Search Algorithm for Multiobjective Optimization Problems

Gerardo G. Fogel
National University of Asuncion
Asuncion, Paraguay
Email: gerardofogel@gmail.com

Benjamín Barán
National University of Asuncion
Asuncion, Paraguay
Email: bbaran@pol.una.py

Marcos Villagra
National University of Asuncion
Asuncion, Paraguay
Email: mvillagra@pol.una.py

***Abstract*—Quantum Computing is a field of study in computer science based on the laws of quantum physics. Quantum computing is an attractive subject considering that quantum algorithms proved to be more efficient than classical algorithms and the advent of large-scale quantum computation. In particular, Grover's search algorithm is a quantum algorithm that is asymptotically faster than any classical search algorithm and it is relevant for the design of fast optimization algorithms. This article proposes two algorithms based on Grover's adaptative search for biobjective optimization problems where access to the objective functions is given via two different quantum oracles. The proposed algorithms, considering both types of oracles, are compared against NSGA-II, a highly cited multiobjective optimization evolutionary algorithm. Experimental evidence suggests that the quantum optimization methods proposed in this work are at least as effective as NSGA-II in average, considering an equal number of executions. Experimental results showed which oracle required less iterations for similar effectiveness.**

I. Introduction

Quantum Computing is a field of study in computer science since the 1980's. It is based on the laws of quantum physics as superposition, entanglement and interference, which cannot be efficiently simulated by classical computers [1]. In the middle of the 1990's, after the development of an efficient quantum algorithm for integer factorization [2], the idea of quantum computers became more relevant, considering that the quantum algorithms proved to be asymptotically faster over classical algorithms. 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 $O(N)$ required by 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.

In this paper, we propose an application 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 from 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 implement 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 the proposed MOGAS algorithm (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 strategies 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 simple quantum algorithms with respect to classical algorithms.

This paper is organized as follows. In Section 2, a brief introduction to Grover's algorithm is given. In Section 3, an application of Grover's search algorithm to optimization problems and the algorithm of Dürr and Høyer is explained. Section 4 reviews basic definitions of multiobjective optimization. In Section 5 the proposed algorithm MOGAS is presented and Section 6 shows the experimental results and some discussions. Finally, Section 7 concludes the paper.

II. Grover's Search Algorithm

In this section we briefly explain Grover's algorithm, which is an integral part of the proposed algorithm of this work. For details refer to the book by Nielsen and Chuang [1].

The fundamental element of information in a quantum computer is the *quantum bit* or qubit. These qubits may be in a superposition state of classical states one and zero, that is, a linear combination of zeros and ones with complex coefficients (or amplitudes). Qubits are represented by basis vector states $|1\rangle$ and $|0\rangle$, usually referred to as the *computational basis*[1]. In quantum computing, quantum states are described using the linear algebra of Hilbert's spaces, and therefore, they are represented using vectors over a complex number field [1].

In classical computation, finding a specific element out from a set of N elements requires N tries; that is, the complexity of finding a particular element is $O(N)$, which is tight [1].

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

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

$$r = \left\lfloor \frac{\pi}{4}\sqrt{N} \right\rfloor \approx \sqrt{N}, \tag{1}$$

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

The input to Grover's algorithm is a set of 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 an n-fold Hadamard transformation $\mathbf{H}^{\otimes n}$,

$$|\zeta\rangle = \mathbf{H}^{\otimes n}|0\rangle^{\otimes n} = \frac{1}{\sqrt{N}} \sum_{x \in \{1,0\}^n} |x\rangle. \tag{2}$$

A superposition of basis states is a particular case of linear combination where the square moduli of the complex coefficients (amplitudes) must sum to one. The second register is transformed using a Hadamard gate according to

$$\mathbf{H}|1\rangle = |-\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}}. \tag{3}$$

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 sign of the coefficient 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 is possible to observe/measure the desired solution state with high probability [1].

[1]The ket notation $|\cdot\rangle$ is simply a notation for a column vector of a vector space.

III. Dürr and Høyer's Algorithm

Grover's algorithm is generally used as a search method to find a set of desired solutions from a set of possible solutions. However, Dürr and Høyer presented an algorithm based on Grover's method [4] for optimization. Their algorithm finds an element of minimum value inside an array of N elements using at most $O(\sqrt{N})$ queries to the oracle.

Baritompa, Bulger and Wood [5] presented an application of Grover's algorithm for global optimization, 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. The oracle operator marks all the solutions from a set below a certain threshold value y given by

$$g(x) = \begin{cases} 1, & \text{if } f(x) < y \\ 0, & \text{if } f(x) \geq y \end{cases}, \tag{4}$$

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 currently known solution and an iteration count. This iteration count is a value 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 that are better than this last solution is unknown and an iteration count is required to perform the search. This is due to the black box nature of the oracle [5].

When the algorithm finds a better solution, it becomes the new known solution. This 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].

Dürr and Høyer introduced a strategy for the selection of the iteration count based on a random selection of a number from a set of integer numbers. This set starts with $\{0\}$ as the only element. When the search is unsuccessful in finding a better solution, the algorithm adds more 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. Thus, the probability of selecting the right iteration count for a successful search increases.

The value of m is updated at each step by $\min\{\lambda^i m, \sqrt{N}\}$, where λ is 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.

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

Algorithm 1 Dürr and Høyer's Algorithm

Randomly choose x from the decision space.
Set $x_1 \leftarrow x$.
Set $y_1 \leftarrow f(x_1)$.
Set $m \leftarrow 1$.
Choose a value for the parameter λ (8/7 is suggested).
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 $m \leftarrow \min\{\lambda m, \sqrt{N}\}$.

IV. Multiobjective Optimization

The goal of a multiobjective optimization problem is to optimize several objectives (at least two) at the same time. The 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 a 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].

V. Multiobjective Grover Adaptive Search (MOGAS)

In this work, a new adaptative search algorithm based on the heuristic of Dürr and Høyer is proposed named *Multiobjetive Grover Adaptive Search (MOGAS)*. 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

$$h_1(x) = \begin{cases} 1, & \text{if } \mathbf{F}(x) \prec \mathbf{Y} \\ 0, & \text{otherwise} \end{cases}, \tag{5}$$

$$h_2(x) = \begin{cases} 1, & \text{if } \mathbf{F}(x) \prec \mathbf{Y} \vee \mathbf{F}(x) \sim \mathbf{Y} \\ 0, & \text{otherwise} \end{cases}, \tag{6}$$

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 vector where each element is the value of each objective function for the current known solution.

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

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

Algorithm 2 MOGAS Algorithm

Randomly choose x from the decision space.
Set $S \leftarrow \{x_1 \leftarrow x\}$
Set $\mathbf{Y_1} \leftarrow \mathbf{F}(x_1)$.
Set $m \leftarrow 1$.
Choose a value for the parameter λ (8/7 is suggested).
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 $S \leftarrow S \cup \{x\}$.
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. Then, using any of the presented oracles, h_1 or h_2, 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 proposed search process it is possible to incorporate a new element into the Pareto-optimal set or replace some old elements from it each time a non-dominated solution is found.

TABLE I
TEST SUITES USED FOR THE EXPERIMENTS.

Function	m	x_i, x_j $i,j = 1,\ldots,2^{10}$	f_1	f_2
$RG_{1,2,3}$	–	$x_i \in [1, 10^3]$ $x_j \in [1, 10^3]$ $x_i, x_j \in \mathbb{N}$	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	$g_4(x_i)\lfloor 1 - \sqrt{x_i/g_4(x_i)}\rfloor$, $g_4(x_i) = 1 + 10(m-1) + \sum_{k=2}^{m}(x_{i_k}^2 - 10cos(4\pi x_{i_k}))$

TABLE II
RESULTS OF THE TESTING PROCEDURE - MOGAS (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

TABLE III
RESULTS OF THE TESTING PROCEDURE - MOGAS (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

VI. EXPERIMENTAL RESULTS

Currently, a general purpose quantum computer has not been implemented. 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 [9].

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 [10] 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 were used and to each of these decision variables a random real number from the interval $[0, 1]$ was assigned.

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 that should be minimized, the vector dimension (for $\mathbf{F}(x)$ and $\mathbf{Y}$) is $p = 2$. Table I presents the main characteristics of the considered test suites.

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) for NSGA-II and a total of four hundred algorithm consultations for MOGAS. Where the algorithm consultation is exactly to a performed Grover search with regard to r_k iterations on $\mathbf{F}(x)$ considering a threshold $\mathbf{Y_k}$, and denoting the outputs by x and $\mathbf{Y}$ 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, which was proposed by Zitzler and Thiele [11]. It is based on a function that maps the set of Pareto-optimal to a scalar with respect to a reference point. In tables II, III and IV, 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.

TABLE IV
RESULTS OF THE TESTING PROCEDURE - NSGA-II (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 V
AVERAGE RESULTS OF THE TESTING PROCEDURE - MOGAS (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

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 II and III correspond to results obtained for MOGAS using h_1 and h_2 respectively. Table IV 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 problems; 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 V, VI and VII summarize the average results of both

TABLE VI
AVERAGE RESULTS OF THE TESTING PROCEDURE - MOGAS (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 VII
AVERAGE RESULTS OF THE TESTING PROCEDURE - NSGA-II (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

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 1 to 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 marks 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 when compared to the oracle based on h_1.

Tables VIII to XIII 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,

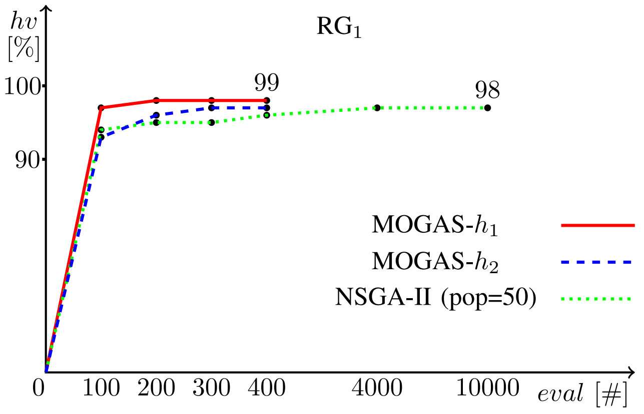

Fig. 1. Graphs of the hypervolume metric in percentage (hv) versus the number of evaluations of the objective function vector ($eval$) made by each algorithm (MOGAS and NSGA-II) with respect to the RG_1 suite test.

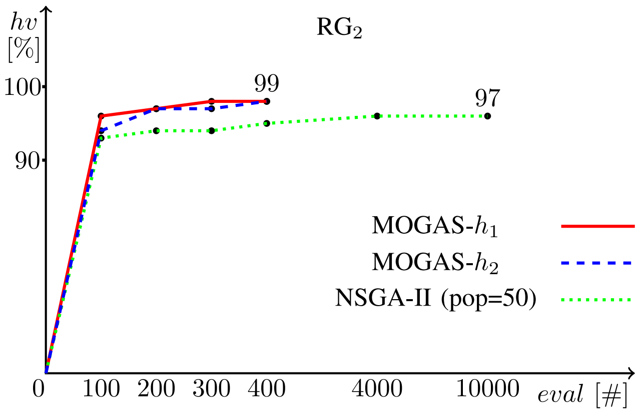

Fig. 2. Graphs of the hypervolume metric in percentage (hv) versus the number of evaluations of the objective function vector ($eval$) made by each algorithm (MOGAS and NSGA-II) with respect to the RG_2 suite test.

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 VIII
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 IX
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 X
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 XI
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 XII
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 XIII
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

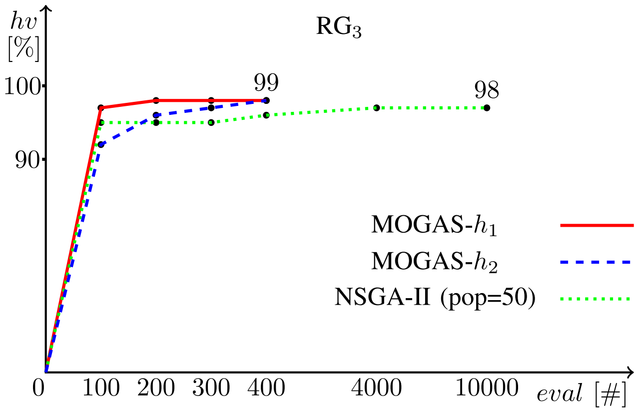

Fig. 3. Graphs of the hypervolume metric in percentage (hv) versus the number of evaluations of the objective function vector ($eval$) made by each algorithm (MOGAS and NSGA-II) with respect to the RG_3 suite test.

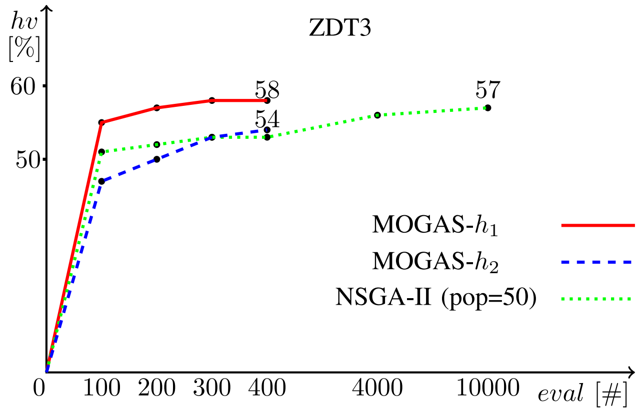

Fig. 5. Graphs of the hypervolume metric in percentage (hv) versus the number of evaluations of the objective function vector ($eval$) made by each algorithm (MOGAS and NSGA-II) with respect to the ZDT3 suite test.

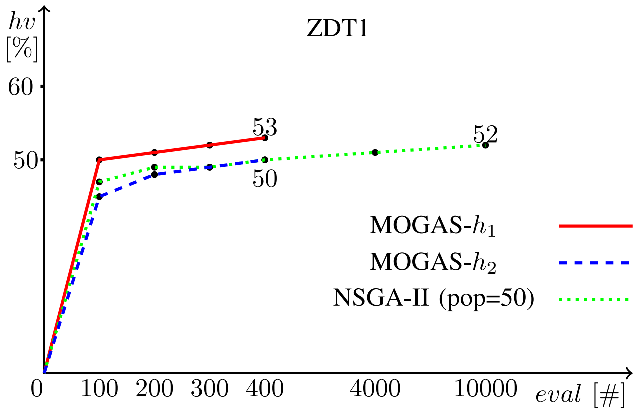

Fig. 4. Graphs of the hypervolume metric in percentage (hv) versus the number of evaluations of the objective function vector ($eval$) made by each algorithm (MOGAS and NSGA-II) with respect to the ZDT1 suite test.

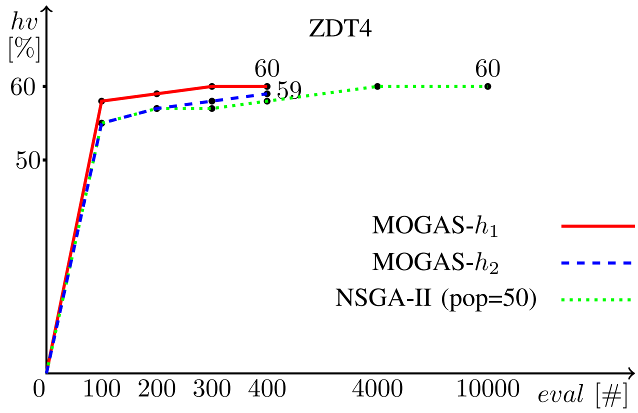

Fig. 6. Graphs of the hypervolume metric in percentage (hv) versus the number of evaluations of the objective function vector ($eval$) made by each algorithm (MOGAS and NSGA-II) with respect to the ZDT4 suite test.

VII. Concluding Remarks

This work 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.

In spite of the simple adaptive strategy used by MOGAS (considering both types of oracles), the experimental results of this work present a remarkable performance 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.

ACKNOWLEDGMENT

The authors acknowledge support from Conacyt grant 14-POS-008.

REFERENCES

[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. doi:10.1109/SFCS.1994.365700

[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. doi:10.1145/237814.237866

[4] Dürr, C. and Høyer, P., *A quantum algorithm for finding the minimum*, arXiv preprint quant-ph/9607014, 1996. doi:10.1.1.57.2796

[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. doi:10.1137/040605072

[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. doi:10.1016/j.entcs.2016.12.003

[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. doi:10.1007/s10589-014-9644-1

[8] Riquelme, N., Baran, B., and von Lücken, C., *Performance metrics in multi-objective optimization*, Computing Conference (CLEI), 2015 Latin American. IEEE, 2015. doi:10.1109/CLEI.2015.7360024

[9] Lipton, R. J., and Regan, K. W. *Quantum Algorithms Via Linear Algebra*, MIT Press, 2014.

[10] Chase, N., et al., *A benchmark study of multi-objective optimization methods*, BMK-3021, Rev 6, 2009. doi:10.1.1.520.1343

[11] E. Zitzler and L. Thiele., *Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach*. IEEE Transactions on Evolutionary Computation, 3(4):257-271, 1999. doi:10.1109/4235.797969