Comparison of two types of Quantum Oracles based on Grover’s Adaptable 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 adaptive 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 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.


In this paper, we propose an application of Grover’s algorithm to multiobjective 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 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 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 \(|0\rangle\) and \(|1\rangle\), usually referred to as the computational basis. 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 \(G\) known as the Grover operator. The Grover operator is constructed from an oracle operator \(O_G\) and a phase operator \(W\).

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

\[
r = \left\lceil \frac{\pi}{4} \sqrt{N} \right\rceil \approx \sqrt{N},
\]

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 \(H^{\otimes n}\),

\[
|\zeta\rangle = H^{\otimes n}|0\rangle^{\otimes n} = \frac{1}{\sqrt{N}} \sum_{x \in \{0,1\}^n} |x\rangle.
\]

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

\[
H|1\rangle = |\zeta\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}}.
\]

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.
interpretation that has been described by Baritompa, Bulger and Wood [5], where the parameter $k$ represents the search process count.

Algorithm 1 Dürr and Høyers’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 $\lambda$ (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 $\{0, \ldots, [m - 1]\}$.
   (b) Perform a Grover search of $r_k$ iterations on $f(x)$ with threshold $y_1$, 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 exist 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 $Y' \succeq Y$, 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 $Y''$ (i.e., $Y' \neq Y''$) nor $Y''$ dominates $Y$ (i.e., $Y' \neq Y$) [7].

V. MULTIOBJECTIVE GROVER ADAPTIVE SEARCH (MOGAS)

In this work, a new adaptive search algorithm based on the heuristic of Dürr and Høyers is proposed named Multiobjective Grover Adaptive Search (MOGAS). MOGAS uses two different oracle operators based on the Pareto dominance relation. The first oracle marks all the non-dominated and non-comparable solutions. These oracles are based on the boolean functions

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

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

where $x$ is a feasible solution of the decision space, $F(x)$ is a vector where each element represents the value of the objective function with respect to solution $x$, and $Y$ 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

1: Randomly choose $x$ from the decision space.
2: Set $S \leftarrow \{x_1 \leftarrow x\}$.
3: Set $Y_1 \leftarrow F(x_1)$.
4: Set $m \leftarrow 1$.
5: Choose a value for the parameter $\lambda$ (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 $\{0, \ldots, [m - 1]\}$.
   (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 $Y \neq Y_k$ set $x_{k+1} \leftarrow x_k$, $Y_{k+1} \leftarrow Y_k$ and $m \leftarrow \min\{\lambda m, \sqrt{N}\}$.
   Otherwise, set $m \leftarrow 1$, $x_{k+1} \leftarrow x$, $Y_{k+1} \leftarrow Y$ and with respect to all elements of the set $S$, where $j = 1, \ldots, |S|$, do:
   \[ \exists x_j \in S : F(x) \prec F(x_j) \text{, then, set } \] 
   \[ S \leftarrow S \setminus \{x_j\} \text{ and finally set } S \leftarrow S \cup \{x\}. \]
7: Set $PF \leftarrow \{F(x_j) : j = 1, \ldots, |S|\}, \forall x_j \in S$.

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.
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, three different suites of this type of random instances were selected for the proposed MOGAS. Where the algorithm consultation is exactly to a threshold (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 \( r_0 \) iterations on \( F(x) \) considering a threshold \( Y_k \), and denoting the outputs by \( x \) and \( Y \) respectively.

The hyphervolume 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 hyphervolume 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 hyphervolume.

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 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 $F(x)$, 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 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 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 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.
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