Implementing Quantum Genetic Algorithms: A Solution Based on Grover's Algorithm

Mihai Udrescu
mudrescu@cs.upt.ro

Lucian Prodan
lprodan@cs.upt.ro

Mircea Vlăduţiu
mvlad@cs.upt.ro

Advanced Computing Systems and Architectures Laboratory
Computer Engineering Department – University "Politehnica" of Timişoara
2 V. Parvan Boulevard, Timişoara 300223, Romania
www.acsa.upt.ro

ABSTRACT

This paper presents a new methodology for running Genetic Algorithms on a Quantum Computer. To the best of our knowledge and according to reference [6] there are no feasible solutions for the implementation of the Quantum Genetic Algorithms (QGAs). We present a new perspective on how to build the corresponding QGA architecture. It turns out that the genetic strategy is not particularly helpful in our quantum computation approach; therefore our solution consists of designing a special-purpose oracle that will work with a modified version of an already known algorithm (maximum finding [1]), in order to reduce the QGAs to a Grover search. Quantum computation offers incentives for this approach, due to the fact that the qubit representation of the chromosome can encode the entire population as a superposition of basis-state values.

Categories and Subject Descriptors

I.1.2 [**Algorithms**]: [Analysis of Algorithms]

General Terms

Algorithms, Theory

Keywords

Quantum Computing, Genetic Algorithms

1. INTRODUCTION

By clearly identifying its most major problems and limitations, computer science has become mature [16]. The research community has put a lot of effort in the attempt to solve these problems, hence further pushing the computing frontiers; however, by using the means of what is now called *classical computation*, it seems that one can hardly expect more than marginal improvements, even for sophisticated approaches. In this context, inspiration was mainly found in biology and physics: bio-inspired computing [14] and quantum computing [16] are considered as possible solutions.

The optimism is fueled by theoretical and practical achievements. Genetic algorithms and evolvable hardware are already successfully used in a wide range of applications, spanning from image compression, robotics and other artificial intelligence related issues, to engineering problems as fault tolerance and reliability in critical environments [20]. Moreover, quantum computing seems to draw even more power from its exponential parallelism: Peter Shor has proven that a classical exponential problem (integer factorization) can be solved in polynomial time [24].

The above provided facts indicate that the merge between the two novel computing promises, namely genetic algorithms (GAs) and quantum computing (QC) would be natural and benefic [25]. Researchers already follow the path of so-called Quantum Evolutionary Programming (QEP) [6] with outstanding results [27]. For instance, the best approach for automated synthesis of quantum circuits uses genetic programming [13]. Also, quantum algorithm design can be approached by evolutionary means [28]. In fact, the majority of such applications addresses quantum computation design issues, regarding quantum algorithms and implementations [27]; they are all part of QEP's subarea called Quantum Inspired Genetic Algorithms (QIGAs) [6][15]. The other sub-area, called Quantum Genetic Algorithms (QGAs), tries to implement genetic algorithms in a quantum computation environment [6][22][23][26] in order to capitalize on the quantum computation exponential parallelism.

This paper proposes a new perspective on QGAs, by showing that the genetic algorithm strategy is essentially different in quantum computation: crossover and mutation are not required, because finding the best fitness can be reduced to Grover's algorithm [8] by means of designing a dedicated Oracle quantum circuit. The search space is entirely covered by the QGA because all individuals are encoded in a superposition state (at the same time), also fitness values generated for all individuals are encoded as a superposition of basis states (at the same time) in a quantum register. As opposed to classical GAs where the best individual fitness pair may not be available because the population is limited, in Quantum Computation the best individual is available,

CF'06, May 3–5, 2006, Ischia, Italy.

as it is a superposed basis state within the quantum state of the individual register. This creates incentives for implementing QGAs; as stated in [6] there is no coherent idea as of how to implement QGAs.

1.1 Motivation

The QGAs rely on qubit representations for the chromosomes and the use of quantum operators, in order to process them during the quest for the optimal solution of the search problem. In principle, this approach redefines the GA operators in quantum terms; these new operators will perform better because they exploit quantum parallelism [23]. It is claimed that, approaching specific applications this way will generate significant performance enhancement [9][10].

Because the chromosome is represented by qubits, just one quantum chromosome register would be able to store the *entire* possible population as a superposition of all the possible classical states. The function that evaluates the fitness of the initial population (which could also be the entire population) would take the *chromosome register* as input and the output would be stored in a *fitness register*. This means that a superposition of all the fitness values is stored, corresponding to the superposition of the individuals from the chromosome register.

The key observation that led us to this new perspective is the fact that if the best fitness value can be *marked* (i.e. by changing the phase of the corresponding basis state) without destroying the superposition of the registers, then Grover's iteration will find the solution. Therefore, all the quantum versions of GA operators, such as crossover or mutation, would not be required if we can figure out a way to mark the best fitness, inside the fitness superposition state.

1.2 Quantum evolutionary strategy

In this section we analyze how the evolutionary strategy is affected by the quantum computation features, according to our perspective over the QGAs. Our main reference would be the general, classical evolutionary computation systems, as described in reference [25] (pages 37-39). Figure 1 (a) presents the typical classical evolutionary system: *generation* of some individuals (usually in a random manner), followed by the *assessment-selection-variation* loop (which will eventually generate the solution).

The assessment process is based on computing fitness values, corresponding to the individuals in the successive populations. Because one cannot be sure that the solution is an individual from the current population, other individuals may be generated by sexual variation (crossover) or non-sexual variation (mutation).

In our quantum computation approach, the *variation* stage of the evolutionary strategy is no longer necessary. The strategy adopted for the evolutionary quantum computation is presented in Figure 1 b). The generation stage means that all the possible individuals (valid or non-valid) are generated as a basis-state superposition due to the fact that the qubits are used to represent the chromosome (i.e. individual encoding). The assessment is then applied on all the superposed individuals, therefore generating a fitness register, which consists of a superposition of all the fitness values. The selection is applied on all the superposed individuals, available within the quantum chromosome, by making use of Grover's algorithm. The fact that all the individuals are already available as superposed, renders the techniques used for generating new individuals as useless. Figure 1 (b) reflects this situation, by showing a simpler model; this is the reason why the algorithm created according to this model was called "Reduced Quantum Genetic Algorithm" (RQGA).

2. QUANTUM GENETIC ALGORITHMS

As part of Quantum Evolutionary Programming, QGAs have the ingredients of a substantial algorithmic speedup, due to the inherited properties from both QC and GAs. However, there still are questions as to how to implement a genetic algorithm on a quantum computer. The attempts made in this particular direction suggest there is room left for taking advantage of the massive quantum computation parallelism [23]. Moreover, as pointed out in [6], some questions were left open.

2.1 Running GAs in a Quantum Computation Environment

For the first time, the possibility (and the advantages) of the QGAs were indicated in [23]. The approach contains hard evidence for QGA speedup, but there still are some unanswered questions [6]. The proposed algorithm uses a number of m register pairs:

$$|\psi\rangle_i = |\phi\rangle_i^{individual} \otimes |\rho\rangle_i^{fitness} \tag{1}$$

where $i = 0..(m-1)$. The first (left, ϕ) register contains the individual, while the second (ρ) contains its corresponding fitness. Because we are dealing with quantum registers, both $|\phi\rangle$ and $|\rho\rangle$ can encode a superposition of exponentially many individuals and their corresponding superposed fitness values. Each time a new population (set of individuals) is generated in the *individual* register, the corresponding fitness is computed and stored in the *fitness* register.

Of course, if the fitness register is measured then, due to the correlation of these registers [16], the result is only one of the superposed values; the individuals that give the measured fitness will remain as superposed in the individual register. Fitness register measurement is a crucial element in developing QGAs [23]. For the general expression of the pair register (N-qubit for the individual register and M-qubit for the fitness register) given in Equation 2, the measurement of the second register ($|y\rangle$) will have r as result with the probability from Equation 3.

$$|\psi\rangle_i = \sum_{x=0}^{2^N-1} \sum_{y=0}^{2^M-1} c_{x,y}|x,y\rangle, \text{ with } \sum_{x=0}^{2^N-1} \sum_{y=0}^{2^M-1} |c_{x,y}|^2 = 1 \tag{2}$$

$$P(r) = \sum_{x=0}^{2^N-1} |c_{x,r}|^2 \tag{3}$$

The post-measurement state of the pair registers will be:

$$|\psi_r\rangle_i = \frac{1}{\sqrt{P(r)}} \sum_{x=0}^{2^N-1} c_{x,r}|x,r\rangle \tag{4}$$

Due to the fact that an individual cannot have more than one fitness, it is obvious that, if individual u has a fitness value v, then $c_{u,y} = 0$ for all $y \neq v$.

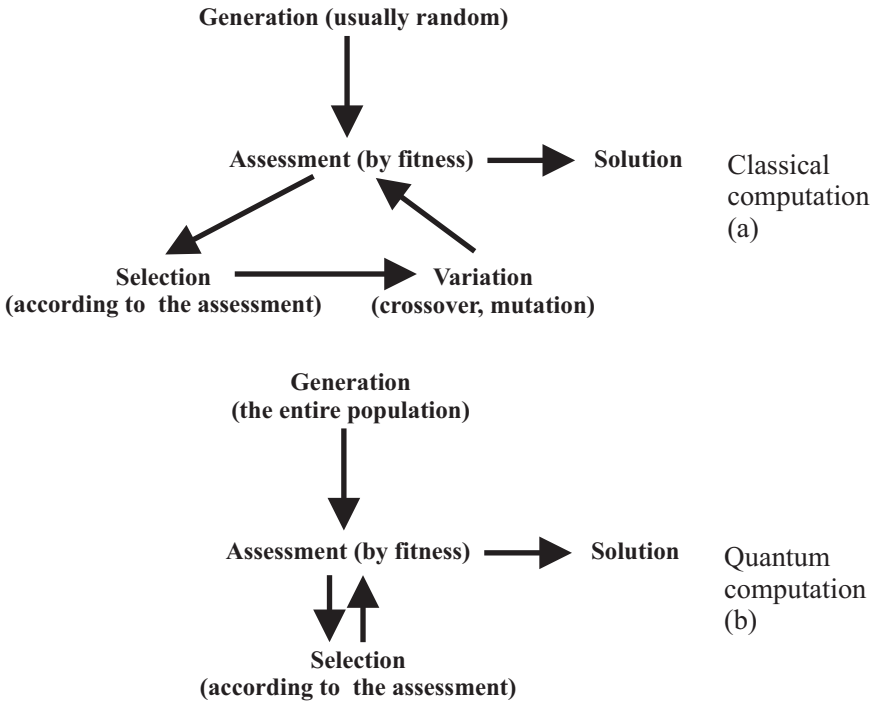

Figure 1: A comparison between the classical evolutionary strategy and the quantum genetic algorithm approach that is presented in this paper.

The QGA, as described in [23], is presented in the following pseudo code:

Genetic Algorithm Running on a Quantum Computer (QGA)

1. For $i := 1$ to m prepare $|\phi\rangle_i^{individual}$ as superpositions of individuals and compute the corresponding fitness pair register $|\rho\rangle_i^{fitness}$ (the outcome will be a state representing a superposition of fitness values).
2. Measure all fitness registers.
3. Repeat
   (a) Selection according to the m measured fitness values.
   (b) Crossover and mutation are employed in order to prepare a new population (setting the m individual registers).
   (c) For the new population, the corresponding fitness values will be computed and then stored in the fitness registers.
   (d) Measure all fitness registers.

   Until the condition for termination is satisfied.

Reference [6] provides analysis and critique for the above presented algorithm. The identified advantages of using QGAs over the classical GAs, which are drawn from the quantum computational features, are:

- Due to the superposition of individuals (i.e. basis states) that is stored in the individual register, the building block [6] could be crossed not by just one individual, but by a superposition of exponentially many individuals. Thus, the selection of a new population is made with the contribution of many building blocks for subsequent populations.
- In quantum computation *true random numbers* can be generated. It was proven that a GA with a true random number generator will outperform a pseudo-random solution, which is the only possibility in classical computation [22].

The questions that remain open are:

a) How would the crossover operator in quantum computation be built?

b) How would the fitness function on a quantum computer be implemented?

c) How could the correlation be maintained between the individual register (superposed) basis states and the (superposed) fitness values from the fitness register?

Although the advantages appear to be substantial, one can easily argue that the power of quantum computation is not sufficiently used by this approach. However, some of the open questions have been addressed in reference [6]. Giraldi *et al.* developed a mathematical formalism in order to avoid misinterpretations regarding question c). The question b) is also addressed by defining quantum genetic operators. The proposed formalism establishes the necessary correlation between the fitness and the individual registers, which – it is argued – cannot be accomplished with the QGA construction provided in [23]. Question a) still remains open according to reference [6].

2.2 Mathematical Formalism

The QGA formalism uses m quantum register pairs (N-qubit individual register and M-qubit fitness register, as presented in Section 2.1). Also, in order to achieve proper correlation between the individual and its fitness value, the fitness

function must be chosen so that it is a "quantum function" as defined by [17], hence a pseudo-classical operator with a corresponding Boolean function: $f : \{0,1\}^N \rightarrow \{0,1\}^M$, $U_f : |x\rangle \otimes |0\rangle \rightarrow |x\rangle \otimes |f(x)\rangle$ if $|x\rangle$ is a basis state.

When acting on a superposition, the unitary operator corresponding to function f will dictate the following mapping:

$$U_f : \sum_{x=0}^{2^N-1} a_x|x\rangle\otimes|0\rangle \rightarrow \sum_{x=0}^{2^N-1} a_x|x\rangle\otimes|f(x)\rangle = \sum_{x=0}^{2^N-1} a_x|x, f(x)\rangle \quad (5)$$

An important aspect regarding the pseudo-classical Boolean functions is that they are universal (i.e. any computational function can be represented in such a form), and are easily implemented as gate networks [2]. In fact, due to their universality, Boolean functions form the backbone of the classical computation's circuit model.

The QGA algorithm, after adopting Giraldi's formalism can be rewritten as in the pseudo-code from below.

Genetic Algorithm Running on a Quantum Computer (QGA) with proper formalism

1. For $i := 1$ to m set the individual-fitness pair registers as $|\psi\rangle_i^1 = \frac{1}{\sqrt{n}} \sum_{u=0}^{n-1} |u\rangle_i^{ind} \otimes |0\rangle_i^{fit}$ (a superposition of n individuals with $0 \leq n \leq 2^N$).
2. Compute the fitness values corresponding to the individual superposition, by applying a unitary transformation $U_{f_{fit}}$ (corresponding to pseudo-classical Boolean operator $f_{fit} : \{0,1\}^N \rightarrow \{0,1\}^M$). For $i := 1$ to m do $|\psi\rangle_i^2 = U_{f_{fit}}|\psi\rangle_i^1 = \frac{1}{\sqrt{n}} \sum_{u=0}^{n-1} |u\rangle_i^{ind} \otimes |f_{fit}(u)\rangle_i^{fit}$.
3. For $i := 1$ to m measure the fitness registers, obtaining the post-measurement states (we suppose that $|y\rangle_i$ is obtained by measurement): $|\psi\rangle_i^3 = \frac{1}{\sqrt{k_i}} \sum_{v\in\{0,1,\ldots,n-1\}} |v\rangle_i^{ind} \otimes |y\rangle_i^{fit}$ with k_i values in $\{0,\ldots,n-1\}$ to satisfy $f_{fit}(v) = y$.
4. Repeat
   a. Selection according to the m measured fitness values $|y\rangle_i$.
   b. Crossover and mutation are employed in order to prepare a new population (setting the m individual registers $|u\rangle_i^{ind}$).
   c. For the new population, the corresponding fitness values will be computed and then stored in the fitness registers($|f_{fit}(u)\rangle_i^{fit}$).
   d. Measure all fitness registers

   Until the condition for termination is satisfied.

Besides the necessary formalism, reference [6] also provides some insight regarding the implementation of the genetic operators in the quantum computational environment. These considerations lead towards two main implementation problems:

α) the number of all valid individuals is not always a power of 2, which is the total number of basis states;

β) crossover implementation is a difficult task, and requires a thorough investigation, including quantum computation architectural aspects [18].

3. THE SOLUTION: A NEW APPROACH

An observation concerning the individual-fitness quantum register pair is that all the possible valid individuals (n) can be encoded in the same quantum state superposition, which has a total of 2^N possible basis states ($n \leq 2^N$). If a method of measuring the highest fitness value from the fitness register can be figured out, then the measurement of the individual register will yield that corresponding individual (or one of them, if several have the same highest fitness value).

Approaching the QGAs in this manner renders some genetic operators as no longer necessary, as long as finding the maximum has an efficient solution. This effectively leads to solving problem β.

Because the individual is encoded on N qubits, we have a total of 2^N basis states which can participate in the superposition. It is possible that not all of these basis states will encode valid individuals (problem α); the proposed method relies on defining some constrains regarding the fitness function and the fitness value format, without losing the generality of the solution. We will consider the fitness function as a Boolean pseudo-classical unitary operator U_f (characterized by $f : \{0,1\}^N \rightarrow \{0,1\}^M$) which can be also applied to non-valid individuals. The fitness value space $\{0,1\}^M$ can be split, so that a distinct subspace is allocated to the fitness values corresponding to valid individuals and another distinct subspace corresponds only to non-valid individuals. This enables us to concentrate only on processing states that correspond to valid individuals (Section 3.2 further elaborates on this particular aspect).

The method of finding the highest fitness value is inspired from efficient quantum algorithms for finding the maximum [1][5]. Finding the best fitness value is equivalent to marking the highest classical state that is superposed in the fitness register state or, in other words, the highest basis state with non-zero amplitude. Basically, the proposed methodology relies on reducing the highest fitness value problem to Grover's algorithm. In order to do so, special oracle and fitness value format are defined. Section 3.1 presents the quantum algorithm for finding the maximum [1], Section 3.2 presents details for oracle implementation and fitness register structure, while Section 3.1.2 provides our adaptation of the algorithm in order to find the best value in the fitness register.

3.1 Computing the Maximum

This subsection analyzes the available quantum methodologies for finding the maximum, and provides a modified version of the original algorithm [1][5], in order to meet the specific QGA demands.

3.1.1 The Initial Algorithm

The quantum algorithms for minimum/maximum finding [1][5] are inspired from the classical "bubble sort" algorithm, but their complexity in quantum version [3] is $\mathcal{O}(\sqrt{n})$.

Such an algorithm takes an unsorted table of m elements as input, in order to return the index of the maximum value element. By adopting the formalism from [1], we have a pool $P[i]$ of m elements ($i = 0..(m-1)$) which will be processed in order to obtain the index k of the maximum element ($P[k]$). In order to meet our demands, Grover's algorithm uses a specially designed oracle that "marks" all the basis states that are greater than some given value $P[j]$ (within the pool):

$$O_j(i) = \begin{cases} 1 & \text{if } P[i] > P[j] \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

We notice that the oracle works with the pool indices as parameters. Also, there is no indication as to how the pool is represented. The oracle just "knows" the answer to the following question: "is $P[i]$ bigger than $P[j]$?" Therefore, the resulting algorithm will have the form of the following pseudo code:

Quantum Algorithm for finding the maximum from an unsorted table of m elements

1. Initialize $k := random\ \ number$; $0 \leq k \leq m-1$ as the starting index of this search;
2. Repeat $\mathcal{O}(\sqrt{m})$ times
   a. Set two quantum registers as $|\psi\rangle = \frac{1}{\sqrt{m}} \sum_{i=0}^{m-1} |i\rangle|k\rangle$; the first register is a superposition of all indices;
   b. Use Grover's algorithm for finding marked states from the first register (i.e. those which make $O_k(i) = 1$);
   c. Measure the first register. The outcome will be one of the basis states which are indices for values $> P[k]$. Let the measurement result be x. Make $k := x$;
3. Return k as result. It is the index of the maximum.

The complexity analysis performed in [1] reveals the fact that this algorithm will find the index of the maximum in $13.6\sqrt{m}$ steps, with an error rate smaller than $\frac{1}{2}$.

3.1.2 The Modified Algorithm

In the initial algorithm, a quantum form for the pool of elements is not necessary. However, for the specific QGA-related purposes, we need to maintain the correlation between the individual register (corresponding to the indices) and the fitness register (corresponding to the fitness values). Therefore, a maximum finding algorithm – that is usable in the desired, genetic algorithm context – must have a quantum (i.e. basis state superposition) state for representing the values, which in turn has to be correlated appropriately with the quantum register representing the indices.

This means that the oracle will operate on the values register, where its input data is available. Hence, the oracle expression from Equation 6 will be modified accordingly:

$$\tilde{O}_y(x) = \begin{cases} 1 & \text{if } x > y \\ 0 & \text{otherwise} \end{cases} \quad (7)$$

In Equation 7, $x, y \in \mathbb{N}$ and are encoded by the superposed basis states from the values register. Also, because a number of $s \in \mathcal{O}(\sqrt{m})$ steps have to be run in order to complete the algorithm, it is required that s index-value quantum register pairs be prepared. Each register pair, except the last one used, generates a partial maximum search solution. The modified quantum maximum finding algorithm is presented in the following pseudocode:

Quantum Algorithm for finding the maximum from an unsorted table of m elements, which is represented as a quantum state

1. Initialize $k := random integer$ with $0 \leq k \leq m-1$; $max := P[k]$;
2. For $j := 0$ to $s-1$ set the pair registers as $|\psi\rangle_j^1 = \frac{1}{\sqrt{m}} \sum_{i=0}^{m-1} |i\rangle_j^{index} \otimes |0\rangle_j^{value}$;
3. For $j := 0$ to $s-1$ set the value corresponding to the index $|\psi\rangle_j^2 = P|\psi\rangle_j^1 = \frac{1}{\sqrt{m}} \sum_{i=0}^{m-1} |i\rangle_j^{index} \otimes |P[i]\rangle_j^{value}$;
4. For $j := 0$ to $s-1$ loop
   (a) Apply the oracle on the value register $\tilde{O}_{max}(P[i])$. Therefore, if $|P[i]\rangle_j^{value} > max$ then the corresponding basis states are marked;
   (b) Use Grover's algorithm for finding marked states in the value register after applying the oracle. As pointed out in reference [16], we find one of the marked basis states $|p\rangle = |P[i]\rangle_j^{value}$, with $P[i] - max > 0$;
   (c) $max := p$;
5. Having the highest value in the $|\bullet\rangle_{s-1}^{value}$ register, we measure the $|\bullet\rangle_{s-1}^{index}$ register in order to obtain the corresponding individual (or one of the corresponding individuals).

3.2 The Oracle

The oracle implementation must be made so that the problems mentioned in reference [6], namely α) and β) from Subsection 2.2, are dealt with. This subsection presents the envisaged solutions.

3.2.1 Problem α: The Solution

In order to deal with problem α), we have to adopt a constraint, which does not restrict the generality of the fitness functions. We consider the ordinary fitness function f_{fit} (which applies only on the valid individuals) $f_{fit} : \{0,1\}^N \rightarrow \{0,1\}^M$, which is Boolean (and therefore universal), with a straightforward correspondence to the unitary representation $U_{f_{fit}}$ [16][17]. The modified fitness function will accept invalid individuals (see Appendix A for details) as argument, and the returned values will belong to distinct areas, corresponding to valid or invalid individuals. This can be achieved by defining $f_{fit}^{mod} : \{0,1\}^N \rightarrow \{0,1\}^{M+1}$ as:

$$f_{fit}^{mod}(x) \in \begin{cases} 0 \times \{0,1\}^M & \text{if } x \text{ is a non-valid individual} \\ 1 \times \{0,1\}^M & \text{if } x \text{ is a valid individual} \end{cases} \quad (8)$$

The fitness values are encoded by the qubits in a modified fitness register, which has a $(M+1)$-qubit size. The valid individuals always produce fitness values with the most significant qubit being '1'; a '0' value for the most significant qubit in the fitness register indicates the correspondence to a non-valid individual, as presented in Figure 2 (where the quantum state matrix representation is used).

3.2.2 Solving Problem β: Building the Appropriate Oracle

Our approach avoids solving problem β directly. Instead, it concerns the definition of an appropriate (i.e. application specific) oracle, starting from Equation 7 and the algorithm from Subsection 3.1.2.

We propose a solution that uses two's complement number representation [19] for marking the states that have a value greater than a given $l \in \mathbb{N}, l > 0$. As a consequence, the fitness register will have the form from Figure 3.

The oracle processes all the fitness register qubits except the most significant one (v), which indicates if the value represented by the other qubits belongs to a valid individual or not. All the value qubits ($f_M \ldots f_0$) from the fitness register encode two's complement positive integers as fitness values. The oracle adds $-(l+1)$ to the fitness register, therefore the

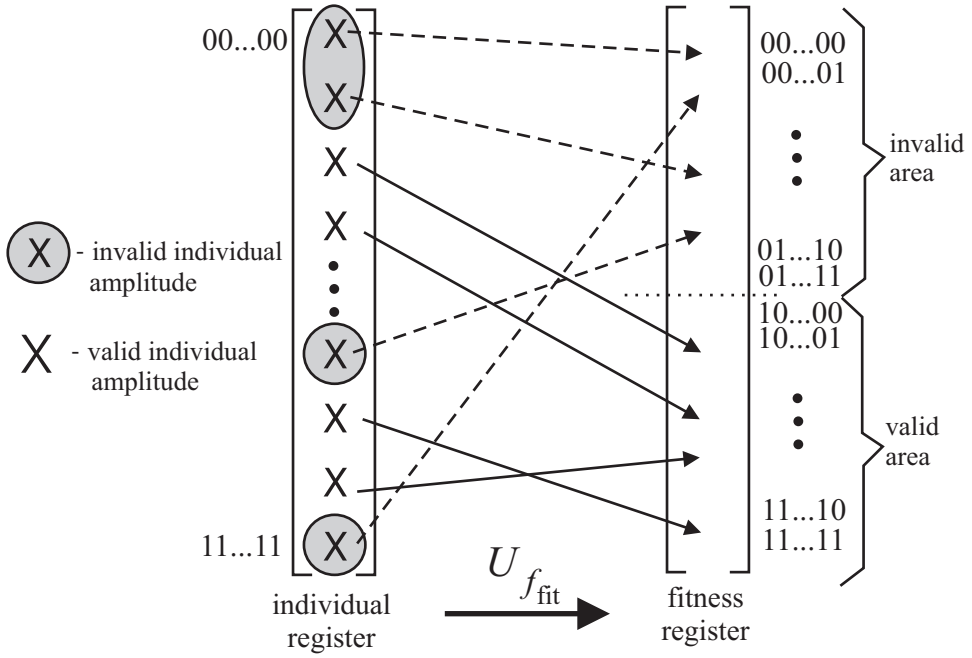

Figure 2: The basics of fitness function construction: when it is applied to valid individuals it produces a value in the valid area (upper half: $|10\ldots00\rangle\ldots|11\ldots11\rangle$) of the fitness register, whereas when applied to invalid individuals, the corresponding values in the fitness register will always be in the invalid area (lower half: $|00\ldots00\rangle\ldots|01\ldots11\rangle$).

basis states (from the state output by the quantum adder [30]) greater than l will always have $f'_M = 0$ (see the oracle implementation from Figure 4.)

For the solution given in Figure 4 we used 2 negation gates (denoted with 'x') and one XOR gate [2] [2][16], in order to change the phase of the corresponding superposed basis states. The architectures for the quantum arithmetic circuits, the adder/subtractor for our particular case, are presented in references [7] and [30]. After marking the corresponding basis states (by shifting their amplitudes), their value is restored by adding $l+1$. Only the qubits containing the result of the arithmetic function ($f''_0 \ldots f''_M$) are used by the Grover iteration circuit [8][16] in order to find one of the marked basis states.

The Grover iterations that are actually applied to the $f''_M \ldots f''_0$ register comply with the Grover algorithm version defined by reference [4], in order to find one of the marked solutions, without any a priori knowledge about the number of solutions.

Although the oracle uses two's complement addition (which means that we will have to change the fitness values in the superposition), the correlation between the individual and the fitness registers is not destroyed, because the addition is a pseudo-classical permutation function [17][30]. The Grover iteration will find as a marked basis state $|p\rangle = |f''_M \ldots f''_0\rangle$, with $p \in \mathbb{N}$, $f''_M, \ldots, f''_0 \in \{0,1\}$ which is given by $|p\rangle = -|q\rangle$ for $|q\rangle = |f_M \ldots f_0\rangle$, with $f_M, \ldots, f_0 \in \{0,1\} = \mathbb{B}$.

The algorithm listed below is inspired from the quantum maximum algorithm from Section 3.1. The initial max value must obey the $2^{M+1} \leq max \leq 2^{M+2}-1$ requirement, so that the search for the highest fitness value will take place only in the valid fitness area. We have a number of $m \in \mathcal{O}(\sqrt{n})$ (due to the complexity analysis provided in [1]) pair registers (individual-fitness), where the individual register is on N qubits, and the fitness register on $M+2$ qubits. Also, it can be said that m "quantum selection steps" are required by this algorithm.

Reduced Quantum Genetic Algorithm

1. For $i := 0$ to $m-1$ set the pair registers as $|\psi\rangle_i^1 = \frac{1}{\sqrt{2^N}} \sum_{u=0}^{2^N-1} |u\rangle_i^{ind} \otimes |0\rangle_i^{fit}$;
2. For $i := 0$ to $m-1$ compute the unitary operation corresponding to fitness computation $|\psi\rangle_i^2 = U_{f_{fit}} |\psi\rangle_i^1 = \frac{1}{\sqrt{2^N}} \sum_{u=0}^{2^N-1} |u\rangle_i^{ind} \otimes |f_{fit}(u)\rangle_i^{fit}$;
3. $max := randominteger$, so that $2^{M+1} \leq max \leq 2^{M+2}-1$;
4. For $i := 0$ to $m-1$ loop
   (a) Apply the oracle $\tilde{O}_{max}\left(f_{fit}(u)\right)$. Therefore, if $|f_{fit}(u)\rangle_i^{fit} > max$ then the corresponding $|f_{fit}(u)\rangle_i^{fit}$ basis states are marked;
   (b) Use Grover iterations for finding marked states in the fitness register after applying the oracle. We find one of the marked basis states $|p\rangle = |f_{fit}(u)\rangle_i^{fit}$, with $f_{fit}(u)\, max \geq 0$;
   (c) $max := p$;
5. Having the highest fitness value in the $|\bullet\rangle_{m-1}^{fit}$ register, we measure the $|\bullet\rangle_{m-1}^{ind}$ register in order to obtain the corresponding individual (or one of the corresponding individuals, if there is more than one solution).

The measurement of the individual corresponding to the best fitness value is possible due to the fact that the correlation with the first (individual) register was not destroyed

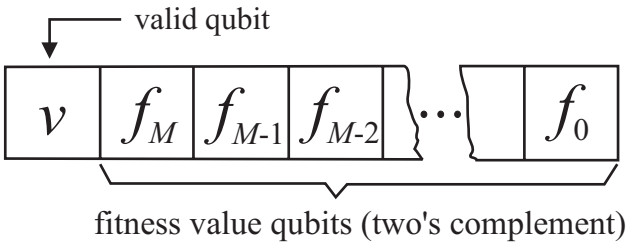

Figure 3: The format of the fitness register, for the oracle implementation based on a two's complement approach.

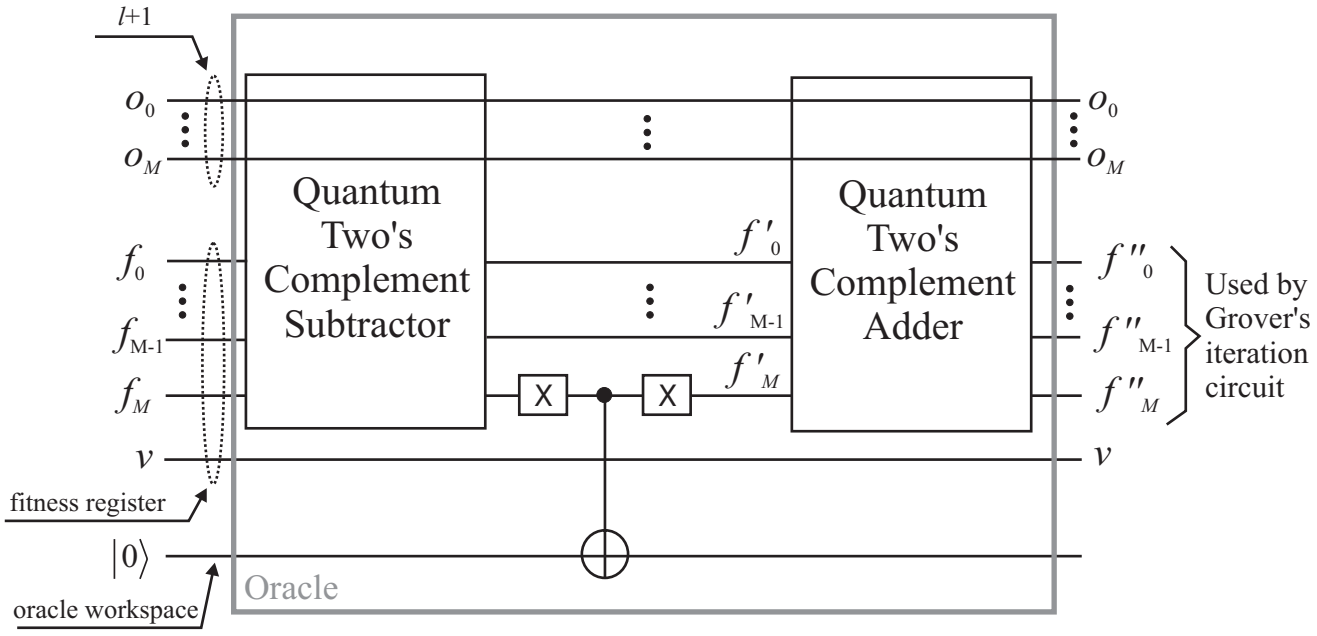

Figure 4: Oracle implementation for a fitness register having the structure from Figure 3.

(only controlled not and Hadamard gates were used [16][17]). Grover's algorithm, interpreted as prescribed by reference [4], is a method of augmenting the amplitude of oracle marked basis states, and it works even if the number of such states is not known in advance.

3.2.3 Complexity Assessment

The fact that the complexity of the original quantum maximum finding algorithm is $\mathcal{O}(\sqrt{n})$ Grover iterations [1] and our proposed algorithm is an adaptation that does not increase the number of steps, may not be enough to achieve an efficient algorithm design. The search space for the modified version of maximum finding has a $n = 2^N$ size, and therefore a logarithmic search mechanism must be refined. The solution can be a search based on Hogg's algorithms [11] or a better (logarithmic) version of the maximum finding procedure. The drawback concerning the usage of Hogg's algorithms is that we do not have reliable complexity assessments even if it is claimed that these algorithms are efficient [12].

However, in our approach, as well as for the quantum maximum finding algorithm, the initial state (that is processed with Grover iterations) is an equally weighted superposition, therefore none of them will require extra Grover iteration steps in order to properly augment their amplitude.

4. CONCLUSIONS

This paper has described a new methodology for running Genetic Algorithms on a Quantum Computer, called *Reduced Quantum Genetic Algorithm* or *RQGA*. This method takes advantage of the quantum computation features, so that all the possible chromosome binary representations can be encoded in just one individual quantum register. This register is correlated with its pair (fitness) register, which contains a superposition of all corresponding fitness values. Due to quantum mechanical properties, measuring the highest fitness value in the fitness register leads to a post-measurement state of the corresponding individual register that contains superposed basis state(s) encoding the individual(s) with the highest fitness.

Therefore, the initial problem (See section 2.1) is reduced to finding the best fitness value, without destroying the individual-fitness register correlation. This objective is met by adapting an existing quantum algorithm for finding the maximum [1][5]. Without loosing the generality of the solution, the adaptation requires that a specific structure be adopted for the fitness register, and a special oracle be defined by employing two's complement integer representation. As a result, the problem of finding the highest fitness value can be solved by Grover's algorithm without employing variation genetic operators. Therefore, it can be concluded that the search strategy itself is different for RQGAs in comparison with the classical GAs: the special Oracle and the Grover iterations perform successive *quantum selection steps*. However, with all the genetic interpretation, the conclusion is that in quantum computation the original genetic search strategy becomes unnecessary.

By using the complexity analysis from [1](performed for the quantum maximum finding algorithm), and by noticing the fact that our algorithm is built around a modified version of maximum finding (without changing the number of required Grover iterations), we concluded that the Reduced Quantum Genetic Algorithms have the potential of being efficient if the maximum finding procedure can be refined to a logarithmic complexity or replaced by one of Hogg's search algorithms. This consequence has the potential of broadening the area of computational problems where the quantum solutions outperform the classical ones, and can also act as counterbalance to the rising skepticism with regard to the effectiveness of Grover's search applications [31]. At the same time, implementing RQGAs creates incentives for strengthening the connection between classical evolvable hardware and quantum programmable gate arrays [21], with potential results in adaptive system design and fault tolerance [29].

5. REFERENCES

[1] A. Ahuja and S. Kapoor. A quantum algorithm for finding the maximum. *ArXiv:quant-ph/9911082*, 1999.

[2] A. Barenco, C. H. Bennett, R. Cleve, D. P. Vincenzo, N. Margolus, P. Shor, T. Sleator, J. Smolin, and H. Weinfurter. Elementary gates for quantum computation. *Phys. Rev.*, A(52):3457–3467, 1995.

[3] E. Bernstein and U. Vazirani. Quantum complexity theory. *SIAM J. Computing*, 26(5):141173, 1997.

[4] M. Boyer, G. Brassard, P. Hoyer, and A. Tapp. Tight bounds on quantum searching. *Fortsch. Phys - Prog. Phys.*, 46(4-5):493–505, 1998.

[5] C. Durr and P. Hoyer. A quantum algorithm for finding the minimum. *ArXiv:quant-ph/9607014*, 1996.

[6] G. Giraldi, R. Portugal, and R. Thess. Genetic algorithms and quantum computation. *ArXiv:cs.NE/0403003*, 2004.

[7] P. Gossett. Quantum carry-save arithmetic. *quant-ph/9808061*, 1998.

[8] L. Grover. Quantum mechanics helps in searching for a needle in a haystack. *Phys. Rev. Lett.*, 79:325–328, 1997.

[9] K.-H. Han and J.-H. Kim. Genetic quantum algorithm and its application to combinatorial optimization problem. In *Proc. of the 2000 Congress on Evolutionary Computation*. citeseer.nj.nec.com/han00genetic.html, 2000.

[10] K.-H. Han and J.-H. Kim. Quantum-inspired evolutionary algorithm for a class of combinatorial optimization. *IEEE Transactions on Evolutionary Computation*, 6(6):580–593, 2002.

[11] T. Hogg. Highly structured searches with quantum computers. *Phys. Rev. Lett.*, 80:2473–2476, 1998.

[12] A. Leier and W. Banzhaf. Evolving hogg's quantum algorithm using linear-tree gp. In *Proc. Genetic and Evolutionary Computation Conference (GECCO)*, pages 390–400, 2003.

[13] M. Lukac, M. Perkowski, H. Goi, M. Pivtoraiko, H.-Y. Chung, K. Chung, H. Jeech, K. Byung-Guk, and K. Yong-Duk. Evolutionary approach to quantum and reversible circuits synthesis. *Artificial Intelligence Review*, 20(3-4):361–417, 2003.

[14] D. Mange, M. Sipper, A. Stauffer, and G. Tempesti. Toward robust integrated circuits: The embryonics approach. *Proc. IEEE*, 88(4):516–541, 2000.

[15] A. Narayanan and M. Moore. Quantum-inspired genetic algorithms. In *Proc. International Conference on Evolutionary Computation (ICEC-96)*, pages 61–66. IEEE, 1996.

[16] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, Cambridge, 2000.

[17] B. Omer. *Quantum programming in QCL*. Technical Report, Institute of Information Systems, Technical University of Vienna, Vienna, 2000.

[18] M. Oskin, F. Chong, and I. Chuang. A practical architecture for reliable quantum computers. *IEEE Computer*, 35(1):79–87, 2002.

[19] B. Parhami. *Computer Arithmetic. Algorithms and Hardware Designs*. Oxford University Press, Oxford, 2000.

[20] L. Prodan, M. Udrescu, and M. Vlăduţiu. Self-repairing embryonic memory arrays. In *Proc. IEEE NASA/DoD Conference on Evolvable Hardware, Seattle*, pages 130–137, 2004.

[21] L. Prodan, M. Udrescu, and M. Vlăduţiu. Survivability of embryonic memories: Analysis and design principles. In *Proc. IEEE NASA/DoD Conference on Evolvable Hardware (EH'05)*, pages 280–289, 2005.

[22] B. Rylander, T. Soule, and J. Foster. Computational complexity, genetic programming, and implications. In *Proc. 4th EuroGP*, pages 348–360, 2001.

[23] B. Rylander, T. Soule, J. Foster, and J. Alves-Foss. Quantum evolutionary programming. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 1005–1011, 2001.

[24] P. W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *Proc. 35th Symposium on Foundations of Computer Science*, pages 124–134, 1994.

[25] L. Spector. *Automatic Quantum Computer Programming: A Genetic Programming Approach*. Kluwer Academic Publishers, Boston, 2004.

[26] L. Spector, H. Barnum, and H. Bernstein. Genetic programming for quantum computers. In *Genetic Programming 1998: Proceedings of the Third Annual Conference, Madison, Wisconsin*, pages 365–373, 1998.

[27] L. Spector, H. Barnum, H. Bernstein, and N. Swamy. Quantum computing applications of genetic programming. *Advances in Genetic Programming*, 3(7):135–160, 1998.

[28] L. Spector, H. Barnum, H. Bernstein, and N. Swamy. Finding a better-than-classical quantum and/or algorithm using genetic programming. In *Proceedings of 1999 Congress of Evolutionary Computation, Piscataway, NJ*, pages 2239–2246. IEEE, 1999.

[29] M. Udrescu, L. Prodan, and M. Vlăduţiu. Improving quantum circuit dependability with reconfigurable quantum gate arrays. In *Proceedings 2nd ACM Conference on Computing Frontiers*, pages 133–144. Ischia, Italy, May 2005.

[30] V. Vedral, A. Barenco, and A. Ekert. uantum networks for elementary arithmetic operations. *quant-ph/9511018*, 1996.

[31] G. Viamontes, I. Markov, and J. P. Hayes. Is quantum search practical? In *International Workshop on Logic and Synthesis*, pages 478–485, 2004.

APPENDIX

A. RQGA EXAMPLE

In this section we will examine an example of how the genetic algorithms will run according to the algorithm described in Section 3.2.2 (Reduced Quantum Genetic Algorithm). Therefore we will take into consideration a typical problem for the genetic approach.

Problem: A person has a backpack with a maximum capacity of 10 kilograms. There are 4 items with the following characteristics: item I_1 having 7 kg and a value of 40 \$, item I_2 (4 kg, 100 \$), item I_3 (2 kg, 50 \$), and item I_4 (3 kg, 30 \$). Which items would this person load in his backpack so that in order to maximize their overall value?

Solution: The classical genetic approach will have as chromosome a 4-bit code, where each '1' bit means that the corresponding item is present in the backpack. For instance, a 1001 code indicates that the backpack contains items I_1 and I_4. In the quantum version, the chromosome encoding will have all the 4-qubit classical values (see Table 1) as superposed basis states, which represent valid and invalid (i.e. the items will add up to more than 10 kg) individuals.

$$|\psi\rangle_i^1 = \frac{1}{4}\sum_{u=0}^{15}|u\rangle_i^{ind}\otimes|0\rangle_i^{fit} = \\ = \frac{1}{4}\begin{pmatrix} |0000\rangle+|0001\rangle \\ +|0010\rangle+|0011\rangle \\ +|0100\rangle+|0101\rangle \\ +|0110\rangle+|0111\rangle \\ +|1000\rangle+|1001\rangle \\ +|1010\rangle+|1011\rangle \\ +|1100\rangle+|1101\rangle \\ +|1110\rangle+|1111\rangle \end{pmatrix}\otimes|0000000000\rangle \quad (9)$$

I_1	I_2	I_3	I_4	Value [\$]	mass [kg]	Validity
0	0	0	0	0	0	valid
0	0	0	1	30	3	valid
0	0	1	0	50	2	valid
0	0	1	1	80	5	valid
0	1	0	0	100	4	valid
0	1	0	1	130	7	valid
0	1	1	0	150	6	valid
0	1	1	1	180	9	valid
1	0	0	0	40	7	valid
1	0	0	1	70	10	valid
1	0	1	0	90	9	valid
1	0	1	1	120	12	invalid
1	1	0	0	140	11	invalid
1	1	0	1	170	14	invalid
1	1	1	0	190	13	invalid
1	1	1	1	220	16	invalid

Table 1: All the chromosome binary combinations, valid and invalid, with the corresponding fitness values.

The fitness function that we will apply, would be an arithmetic function according to the requirements formulated in Section 2.2. The fitness formula, given in Equation 10, contains two variables, the chromosome value *val* and the

chromosome mass m; also two constants are used, the total added value of all items ($val_t = 220$) and the maximum allowed package mass ($m_{max} = 10$). For any chromosome x, the fitness function is given by:

$$\begin{aligned} f_{fit}(x) &= val(x) - (val_t + 1) \times (m(x) \operatorname{div} m_{max}) \\ &= val(x) - 221 \times (m(x) \operatorname{div} 10) \end{aligned} \tag{10}$$

Applying the fitness function over the individual registers $|\phi\rangle_i^{ind}$ means that we use the $U_{f_{fit}}$ basis state permutation, obtaining the state presented in Equation 11.

$$|\psi\rangle_i^2 = |\phi\rangle_i^{ind} \otimes |\rho\rangle_i^{fit} = \frac{1}{4} \begin{pmatrix} |0000\rangle \otimes |1000000000\rangle \\ +|0001\rangle \otimes |1000011110\rangle \\ +|0010\rangle \otimes |1000110010\rangle \\ +|0011\rangle \otimes |1001010000\rangle \\ +|0100\rangle \otimes |1001100100\rangle \\ +|0101\rangle \otimes |1010000010\rangle \\ +|0110\rangle \otimes |1010010110\rangle \\ +|0111\rangle \otimes |1010110100\rangle \\ +|1000\rangle \otimes |1000101000\rangle \\ +|1001\rangle \otimes |1001000110\rangle \\ +|1010\rangle \otimes |1001011010\rangle \\ +|1011\rangle \otimes |0110011011\rangle \\ +|1100\rangle \otimes |0110101111\rangle \\ +|1101\rangle \otimes |0111001101\rangle \\ +|1110\rangle \otimes |0111100001\rangle \\ +|1111\rangle \otimes |0111111111\rangle \end{pmatrix} \tag{11}$$

The fitness values of the invalid individuals are negative numbers (all least significant 9 bits of the fitness values represent 2's complement numbers), with the most significant bit being dedicated to indicating the validity of the corresponding chromosome (0 means invalid, 1 indicates a valid individual), see Section 3.2.2 for details.

The next step of the Reduced Quantum Genetic Algorithm is to apply the oracle over the first pair of individual-fitness registers. According to the Reduced Quantum Genetic Algorithm, we have to get a random value for variable max. Suppose that the yielded value for max is 84, then the state of the $|\psi\rangle_0$ register at this point is given in Equation 12 and 13: $|\psi\rangle_0^3$ is the pair registers state after applying the subtractor and phase-shift part of the oracle, while $|\psi\rangle_0^4$ is the state obtained after applying the entire oracle, including the adder part (see Figure 4 from Section 3.2.2). One observation linked to the details presented in Section 3.2.2 is that the phase shift (i.e. amplitude a_i becomes $-a_i$) is triggered by a '0' value of the 2nd bit from the left in the fitness register.

$$|\psi\rangle_0^3 = |\phi\rangle_0^{ind} \otimes |\rho\rangle_0^{fit} = \frac{1}{4} \begin{pmatrix} |0000\rangle \otimes |1110101011\rangle \\ +|0001\rangle \otimes |1111001001\rangle \\ +|0010\rangle \otimes |1111011101\rangle \\ +|0011\rangle \otimes |1111111011\rangle \\ -|0100\rangle \otimes |1000001111\rangle \\ -|0101\rangle \otimes |1000101101\rangle \\ -|0110\rangle \otimes |1001000001\rangle \\ -|0111\rangle \otimes |1001011111\rangle \\ +|1000\rangle \otimes |1111010011\rangle \\ +|1001\rangle \otimes |1111110001\rangle \\ -|1010\rangle \otimes |1000000101\rangle \\ +|1011\rangle \otimes |1101000110\rangle \\ +|1100\rangle \otimes |1101011010\rangle \\ +|1101\rangle \otimes |1101111000\rangle \\ +|1110\rangle \otimes |1110001100\rangle \\ +|1111\rangle \otimes |1110101010\rangle \end{pmatrix} \tag{12}$$

$$|\psi\rangle_0^4 = |\phi\rangle_0^{ind} \otimes |\rho\rangle_0^{fit} = \frac{1}{4} \begin{pmatrix} |0000\rangle \otimes |1000000000\rangle \\ +|0001\rangle \otimes |1000011110\rangle \\ +|0010\rangle \otimes |1000110010\rangle \\ +|0011\rangle \otimes |1001010000\rangle \\ -|0100\rangle \otimes |1001100100\rangle \\ -|0101\rangle \otimes |1010000010\rangle \\ -|0110\rangle \otimes |1010010110\rangle \\ -|0111\rangle \otimes |1010110100\rangle \\ +|1000\rangle \otimes |1000101000\rangle \\ +|1001\rangle \otimes |1001000110\rangle \\ -|1010\rangle \otimes |1001011010\rangle \\ +|1011\rangle \otimes |0110011011\rangle \\ +|1100\rangle \otimes |0110101111\rangle \\ +|1101\rangle \otimes |0111001101\rangle \\ +|1110\rangle \otimes |0111100001\rangle \\ +|1111\rangle \otimes |0111111111\rangle \end{pmatrix} \tag{13}$$

After applying the Grover algorithm over the fitness register (rightmost 10 qubits) of $|\psi\rangle_0^4$, we will get state $|\psi\rangle_0^5$, as presented in Equation 14, where the amplitudes $a_0, a_1, a_2, a_3, a_8, a_9, a_{11}, a_{12}, \ldots a_{15} \approx 0$ and $|a_4|^2 + |a_5|^2 + |a_6|^2 + |a_7|^2 + |a_{10}|^2 \approx 1$.

$$|\psi\rangle_0^5 = |\phi\rangle_0^{ind} \otimes |\rho\rangle_0^{fit} = \begin{pmatrix} a_0|0000\rangle \otimes |1000000000\rangle \\ +a_1|0001\rangle \otimes |1000011110\rangle \\ +a_2|0010\rangle \otimes |1000110010\rangle \\ +a_3|0011\rangle \otimes |1001010000\rangle \\ +a_4|0100\rangle \otimes |1001100100\rangle \\ +a_5|0101\rangle \otimes |1010000010\rangle \\ +a_6|0110\rangle \otimes |1010010110\rangle \\ +a_7|0111\rangle \otimes |1010110100\rangle \\ +a_8|1000\rangle \otimes |1000101000\rangle \\ +a_9|1001\rangle \otimes |1001000110\rangle \\ +a_{10}|1010\rangle \otimes |1001011010\rangle \\ +a_{11}|1011\rangle \otimes |0110011011\rangle \\ +a_{12}|1100\rangle \otimes |0110101111\rangle \\ +a_{13}|1101\rangle \otimes |0111001101\rangle \\ +a_{14}|1110\rangle \otimes |0111100001\rangle \\ +a_{15}|1111\rangle \otimes |0111111111\rangle \end{pmatrix} \tag{14}$$

Therefore, if the fitness register of $|\psi\rangle_0$ is measured after applying Grover iterations, then we will get (with a high probability) one of the following basis states (of the right-

most 10 qubits of $|\psi\rangle_0^5$): $|1001100100\rangle, |1010000010\rangle, |1010010110\rangle, |1010110100\rangle$. Suppose that $|1010000010\rangle$ (+ 130 if we convert this value in decimal) is measured. In the individual register we will have $|0101\rangle$; also the new $max := 130$.

The next algorithm iteration will involve the next individual-fitness pair registers ($|\psi\rangle_1 = |\rho\rangle_1^{ind} \otimes |\phi\rangle_1^{fit}$), by subsequently setting states $|\psi\rangle_1^3$ (oracle – subtractor and phase-shift), $|\psi\rangle_1^4$ (oracle – adder), and $|\psi\rangle_1^5$ (Grover iterations):

$$|\psi\rangle_1^3 = |\phi\rangle_1^{ind} \otimes |\rho\rangle_1^{fit} = \frac{1}{4} \begin{pmatrix} |0000\rangle \otimes |1101111101\rangle \\ +|0001\rangle \otimes |1110011011\rangle \\ +|0010\rangle \otimes |1110101111\rangle \\ +|0011\rangle \otimes |1111001101\rangle \\ +|0100\rangle \otimes |1111100001\rangle \\ +|0101\rangle \otimes |1111111111\rangle \\ -|0110\rangle \otimes |1000010011\rangle \\ -|0111\rangle \otimes |1000110001\rangle \\ +|1000\rangle \otimes |1110100101\rangle \\ +|1001\rangle \otimes |1111000011\rangle \\ +|1010\rangle \otimes |1111010111\rangle \\ +|1011\rangle \otimes |0100011000\rangle \\ +|1100\rangle \otimes |0100101100\rangle \\ +|1101\rangle \otimes |0101001010\rangle \\ +|1110\rangle \otimes |0101011110\rangle \\ +|1111\rangle \otimes |0101111100\rangle \end{pmatrix} \tag{15}$$

$$|\psi\rangle_1^4 = |\phi\rangle_1^{ind} \otimes |\rho\rangle_1^{fit} = \frac{1}{4} \begin{pmatrix} |0000\rangle \otimes |1000000000\rangle \\ +|0001\rangle \otimes |1000011110\rangle \\ +|0010\rangle \otimes |1000110010\rangle \\ +|0011\rangle \otimes |1001010000\rangle \\ +|0100\rangle \otimes |1001100100\rangle \\ +|0101\rangle \otimes |1010000010\rangle \\ -|0110\rangle \otimes |1010010110\rangle \\ -|0111\rangle \otimes |1010110100\rangle \\ +|1000\rangle \otimes |1000101000\rangle \\ +|1001\rangle \otimes |1001000110\rangle \\ +|1010\rangle \otimes |1001011010\rangle \\ +|1011\rangle \otimes |0110011011\rangle \\ +|1100\rangle \otimes |0110101111\rangle \\ +|1101\rangle \otimes |0111001101\rangle \\ +|1110\rangle \otimes |0111100001\rangle \\ +|1111\rangle \otimes |0111111111\rangle \end{pmatrix} \tag{16}$$

$$|\psi\rangle_1^5 = |\phi\rangle_1^{ind} \otimes |\rho\rangle_1^{fit} = \begin{pmatrix} a_0|0000\rangle \otimes |1000000000\rangle \\ +a_1|0001\rangle \otimes |1000011110\rangle \\ +a_2|0010\rangle \otimes |1000110010\rangle \\ +a_3|0011\rangle \otimes |1001010000\rangle \\ +a_4|0100\rangle \otimes |1001100100\rangle \\ +a_5|0101\rangle \otimes |1010000010\rangle \\ +a_6|0110\rangle \otimes |1010010110\rangle \\ +a_7|0111\rangle \otimes |1010110100\rangle \\ +a_8|1000\rangle \otimes |1000101000\rangle \\ +a_9|1001\rangle \otimes |1001000110\rangle \\ +a_{10}|1010\rangle \otimes |1001011010\rangle \\ +a_{11}|1011\rangle \otimes |0110011011\rangle \\ +a_{12}|1100\rangle \otimes |0110101111\rangle \\ +a_{13}|1101\rangle \otimes |0111001101\rangle \\ +a_{14}|1110\rangle \otimes |0111100001\rangle \\ +a_{15}|1111\rangle \otimes |0111111111\rangle \end{pmatrix} \tag{17}$$

After applying the Grover algorithm over the fitness register (rightmost 10 qubits) of $|\phi\rangle_1^{fit}$ from $|\psi\rangle_1^4$, we will get state $|\psi\rangle_1^5$, as presented in Equation 17, where the amplitudes $a_0, a_1, a_2, a_3, a_4, a_5, a_8, a_9, \ldots, a_{15} \approx 0$ and $|a_6|^2 + |a_7|^2 \approx 1$.

Therefore, if we measure the fitness register of $|\psi\rangle_1^5$ after applying Grover iterations, then we will measure (with a high probability) one of the following basis states (of the rightmost 10 qubits of $|\psi\rangle_1^5$): $|1010010110\rangle, |1010110100\rangle$. Suppose that we measure $|1010110100\rangle$ (+ 180 in decimal). In the individual register we will have $|0111\rangle$, which is also the solution for our problem.