Evolving Hogg's Quantum Algorithm Using Linear-Tree GP

André Leier and Wolfgang Banzhaf

University of Dortmund, Dept. of Computer Science, Chair of Systems Analysis,
44221 Dortmund, Germany
{andre.leier, wolfgang.banzhaf}@cs.uni-dortmund.de

Abstract. Intermediate measurements in quantum circuits compare to conditional branchings in programming languages. Due to this, quantum circuits have a natural linear-tree structure. In this paper a Genetic Programming system based on linear-tree genome structures developed for the purpose of automatic quantum circuit design is introduced. It was applied to instances of the 1-SAT problem, resulting in evidently and "visibly" scalable quantum algorithms, which correspond to Hogg's quantum algorithm.

1 Introduction

In theory certain computational problems can be solved on a quantum computer with a lower complexity than possible on classical computers. Therefore, in view of its potential, design of new quantum algorithms is desirable, although no working quantum computer beyond experimental realizations has been built so far. Unfortunately, the development of quantum algorithms is very difficult, since they are highly non-intuitive and their simulation on conventional computers is very expensive.

The use of genetic programming to evolve quantum circuits is not a novel approach. It was elaborated first in 1997 by Williams and Gray [21]. Since then, various other papers [5,1,15,18,17,2,16,14,20] dealt with quantum computing as an application of genetic programming or genetic algorithms, respectively. The primary goal of most GP experiments, described in this context, was to demonstrate the feasibility of automatic quantum circuit design. Different GP schemes and representations of quantum algorithms were considered and tested on various problems.

The GP system described in this paper uses linear-tree structures and was build to achieve more "degrees of freedom" in the construction and evolution of quantum circuits compared to stricter linear GP schemes (like in [14,18]). A further goal was to evolve quantum algorithms for the k-SAT problem (only for $k = 1$ up to now). In [9,10] Hogg has already introduced quantum search algorithms for 1-SAT and highly constrained k-SAT. An experimental implementation of Hogg's 1-SAT algorithm for logical formulas in three variables is demonstrated in [13].

E. Cantú-Paz et al. (Eds.): GECCO 2003, LNCS 2723, pp. 390–400, 2003.

The following section briefly outlines some basics of quantum computing essential to understand the mathematical principles on which the simulation of quantum algorithms depends. Section 3 of this paper discusses previous work on automatic quantum circuit design. Section 4 describes the linear-tree GP scheme used here. The results of evolving quantum algorithms for the 1-SAT problem are presented in Sect. 5. The last section summarizes our results and draws conclusions.

2 Quantum Computing Basics

Quantum computing is the result of a link between quantum mechanics and information theory. It is computation based on quantum principles, that is quantum computers use coherent atomic-scale dynamics to store and to process information [19]. The basic unit of information is the qubit which, unlike a classical bit can exist in a superposition of the two classical states 0 and 1, i. e. with a certain probability p, resp. $1-p$, the qubit is in state 0, resp. 1. In the same way an n-qubit quantum register can be in a superposition of its 2^n classical states. The state of the quantum register is described by a 2^n-dimensional complex vector $(\alpha_0, \alpha_1, \ldots, \alpha_{2^n-1})^t$, where α_k is the probability amplitude corresponding to the classical state k. The probability for the quantum register being in state k is $|\alpha_k|^2$ and from the normalization condition of probability measures it follows $\sum_{k=0}^{2^n-1} |\alpha_k|^2 = 1$. It is common usage to write the classical states (the so-called computational basis states) in the 'ket' notation of quantum computing, as $|k\rangle = |a_{n-1}a_{n-2}\ldots a_0\rangle$, where $a_{n-1}a_{n-2}\ldots a_0$ is the binary representation of k. Thus, the general state of an n-qubit quantum computer can be written as $|\psi\rangle = \sum_{k=0}^{2^n-1} \alpha_k |k\rangle$.

The quantum circuit model of computation describes quantum algorithms as a sequence of unitary – and therefore reversible – transformations (plus some non-unitary measurement operators), also called quantum gates, which are applied successively to an initialized quantum state. Usually this state to an n-qubit quantum circuit is $|0\rangle^{\otimes n}$. A unitary transformation operating on n qubits is a $2^n \times 2^n$ matrix U, with $U^\dagger U = I$. Each quantum gate is entirely determined by it's gate type, the qubits, it is acting on, and a certain number of real-valued (angle) parameters. Figure 1 shows some basic gate types working on one or two qubits. Similar to the universality property of classical gates, small sets of quantum gates are sufficient to compute any unitary transformation to arbitrary accuracy. For example, single qubit and $CNOT$ gates are universal for quantum computation, just as H, $CNOT$, $Phase[\pi/4]$ and $Phase[\pi/2]$ are. In order to be applicable to an n-qubit quantum computer (with a 2^n-dimensional state vector) quantum gates operating on less than n qubits have to be adapted to higher dimensions. For example, let U be an arbitrary single-qubit gate applied to qubit q of an n-qubit register. Then the entire n-qubit transformation is composed of the tensor product

$$\underbrace{I \otimes \ldots \otimes I}_{n-(q+1)} \otimes U \otimes \underbrace{I \ldots \otimes I}_{q}$$

$$H = 1/\sqrt{2}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \qquad Phase[\phi] = \begin{pmatrix} 1 & 0 \\ 0 & e^{\phi} \end{pmatrix} \qquad CNOT = \begin{pmatrix} 1&0&0&0 \\ 0&1&0&0 \\ 0&0&0&1 \\ 0&0&1&0 \end{pmatrix}$$

$$Rx[\phi] = \begin{pmatrix} \cos\phi & i\sin\phi \\ i\sin\phi & \cos\phi \end{pmatrix} \qquad Ry[\phi] = \begin{pmatrix} \cos\phi & \sin\phi \\ -\sin\phi & \cos\phi \end{pmatrix} \qquad Rz[\phi] = \begin{pmatrix} e^{-i\phi} & 0 \\ 0 & e^{i\phi} \end{pmatrix}$$

Fig. 1. Some basic unitary 1- and 2-qubit transformations: Hadamard-gate H, a $Phase$-gate with angle parameter ϕ, a $CNOT$-gate, some rotation gates $Rx[\phi], Ry[\phi], Rz[\phi]$ with rotation angle ϕ.

Calculating the new quantum state requires 2^{n-1} matrix-vector-multiplications of the 2×2 matrix U. It is easy to see, that the costs of simulating quantum circuits on conventional computers grow exponentially with the number of qubits.

Input gates sometimes known as oracles enable the encoding of problem instances. They may change from instance to instance of a given problem, while the "surrounding" quantum algorithm remains unchanged. Consequently, a proper quantum algorithm solving the problem has to achieve the correct outputs for all oracles representing problem instances. In quantum algorithms like Grover's [6] or Deutsch's [3,4], oracle gates are permutation matrices computing Boolean functions (Fig. 2, left matrix). Hogg's quantum algorithm for k-SAT [9,10] uses a special diagonal matrix, encoding the number of conflicts in assignment s, i. e. the number of false clauses for assignment s in the given logical formula at position (s, s) (Fig. 2, right matrix).

$$\begin{pmatrix} 1&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0 \\ 0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&0&1&0 \end{pmatrix} \qquad \begin{pmatrix} 1&0&0&0&0&0&0&0 \\ 0&i&0&0&0&0&0&0 \\ 0&0&i&0&0&0&0&0 \\ 0&0&0&-1&0&0&0&0 \\ 0&0&0&0&i&0&0&0 \\ 0&0&0&0&0&-1&0&0 \\ 0&0&0&0&0&0&-1&0 \\ 0&0&0&0&0&0&0&-i \end{pmatrix}$$

Fig. 2. Examples for oracle matrices. Left matrix: implementation of the AND function of two inputs. The right-most qubit is flipped, if the two other qubits are '1'. This gate is also called a $CCNOT$. Right matrix: a diagonal matrix with coefficients $(i^{c(000)}, \ldots, i^{c(111)})$, where $c(s)$ is the number of conflicts of assignment s in the formula $\bar{v}_1 \wedge \bar{v}_2 \wedge \bar{v}_3$. For example, the assignment ($v_1 =$ true, $v_2 =$ false, $v_3 =$ true) makes two clauses false, i. e. $c(101) = 2$ and $i^2 = -1$.

Quantum information processing is useless without readout (measurement). When the state of a quantum computer is measured in the computational basis, result 'k' occurs with probability $|\alpha_k|^2$. By measurement the superposition collapses to $|k\rangle$. A partial measurement of a single qubit is a projection into the subspace, which corresponds to the measured qubit. The probability p of measuring a single qubit q with result '0' ('1') is the sum of the probabilities for all basis states with qubit $q = 0$ ($q = 1$). The post-measurement state is just the superposition of these basis states, re-normalized by the factor $1/\sqrt{p}$. For example, measuring the first (right-most) qubit of $|\psi\rangle = \alpha_0|00\rangle + \alpha_1|01\rangle + \alpha_2|10\rangle + \alpha_3|11\rangle$ gives '1' with probability $|\alpha_1|^2 + |\alpha_3|^2$, leaving the post-measurement state $|\psi'\rangle = 1/\sqrt{|\alpha_1|^2 + |\alpha_3|^2}(\alpha_1|01\rangle + \alpha_3|11\rangle)$.

According to the *quantum principle of deferred measurement*, "measurements can always be moved from an intermediate stage of a quantum circuit to the end of the circuit" [12]. Of course, such a shift has to be compensated by some other changes in the quantum circuit.

Note, that quantum measurements are irreversible operators, though it is usual to call these operators measurement *gates*. To get a deeper insight into quantum computing and quantum algorithms the following references might be of interest to the reader: [12],[7],[8].

3 Previous Work in Automatic Quantum Circuit Design

Williams and Gray focus in [21] on demonstrating a GP-based search heuristic more efficient than the exhaustive enumeration strategy which finds a correct decomposition of a given unitary matrix U into a sequence of simple quantum gate operations. In contrast, however, to subsequent GP schemes for the evolution of quantum circuits, a unitary operator solving the given problem had to be known in advance.

Extensive investigations concerning the evolution of quantum algorithms were done by Spector et al. [15,18,17,1,2]. In [18] they presented three different GP schemes for quantum circuit evolution: the standard tree-based GP (TGP) and both stack-based and stackless linear genome GP (SBLGP/SLLGP). These were applied to evolve algorithms for Deutsch's two-bit early promise problem, using TGP, the scaling majority-on problem, using TGP as well, the quantum four-item database search problem, using SBLGP, and the two-bit-AND-OR problem, using SLLGP. Better-than-classical algorithms could be evolved for all but the scaling majority-on problem.

Without doing a thorough comparison Spector et al. pointed out some pros and cons of the three GP schemes: The tree structure of individuals in TGP simplifies the evolution of scalable quantum circuits, as it seems to be predestined for "adaptive determination of program size and shape" [18]. A disadvantage of the tree representation are its higher costs in time, space and complexity. Furthermore, possible return-value/side-effect interactions may make evolution more complicated for TGP. The linear representation in SBLGP/SLLGP seems to be better suited for evolution, because the quantum algorithms are itself se-

quential (in accordance with the principle of deferred measurement). Moreover, the genetic operators in linear GP are simpler to implement and memory requirements are clearly reduced compared to TGP. The return-value/side-effect interaction is eliminated in SBGL, since the algorithm-building functions do not return any values. Overall, Spector et al. stated that, applied to their problems, results appeared to emerge more quickly with SBLGP than with TGP. If scalability of the quantum algorithms would be not so important, the SLLGP approach should be preferred.

In [17] and [2] a modified SLLGP system was applied to the 2-bit-AND-OR problem, evolving an improved quantum algorithm. The new system is steady-state rather than generational as its predecessor system, supports true variable-length genomes and enables distributed evolution on a workstation cluster. Expensive genetic operators allow for "local hill-climbing search [...] integrated into the genetic search process". For fitness evaluation the GP system uses a standardized lexicographic fitness function consisting of four fitness components: the number of fitness cases on which the quantum program "failed" (MISSES), the number of expected oracle-gates in the quantum circuit (EXPECTED-QUERIES), the maximum probability over all fitness cases of getting the wrong result (MAX-ERROR) and the number of gates (NUM-GATES).

Another interesting GP scheme is presented in [14] and its function is demonstrated by generating quantum circuits for the production of two to five maximally entangled qubits. In this scheme gates are represented by a gate type and by bit-strings coding the qubit operands and gate parameters. Qubit operands and parameters have to be interpreted corresponding to the gate type. Assigning a further binary key to each gate type the gate representation is completely based on bit strings, where appropriate genetic operators can be applied to.

4 The Linear-Tree GP Scheme

The steady-state GP system described here is a linear-tree GP scheme, introduced first in [11]. The structure of the individuals consists of linear program segments, which are sequences of *unitary* quantum gates, and branchings, caused by single qubit measurement gates. Depending on the measurement result ('0' or '1'), the corresponding (linear) program branch, the '0'- or '1'-branch, is executed. Since measurement results occur with certain probabilities, usually both branches have to be evaluated. Therefore, the quantum gates in the '0'- and '1'-branch have to be applied to their respective post-measurement states. From the branching probabilities the probabilities for each final quantum state can be calculated.

In this way linear-tree GP naturally supports the use of measurements as an intermediate step in quantum circuits. Measurement gates can be employed to conditionally control subsequent quantum gates, like an "if-then-else"-construct in a programming language. Although the principle of deferred measurement suggests the use of purely sequential individual structures, the linear-tree structure may simplify legibility and interpretation of quantum algorithms.

The maximum number of possible branches is set by a global system parameter; without using any measurement gates the GP system becomes very similar to the modified SLLGP version in [17]. From there, we adopted the idea of using fitness components with certain weights: MISSES, MAX-ERROR and TOTAL-ERROR (the summed error over all fitness cases) are used in this way. A penalty function based on NUM-GATES and a global system parameter is used to increase slightly the fitness value for any existing gate in the quantum circuit. In order to restrict the evolution, in particular at the beginning of a GP run, fitness evaluation of an individual is aborted if the number of MISSES exceeds a certain value, set by another global system parameter. The bitlength of gate parameters (interpreted as a fraction of 2π) was fixed to 12 bits which restricts angle resolution. This corresponds to current precisions for NMR experiments. The genetic operators used here are RANDOM-INSERTION, RANDOM-DELETION and RANDOM-ALTERATION, each referred to a single quantum gate, plus LINEAR-XOVER and TREE-XOVER. A GP run terminates when the number of tournaments exceeds a given value (in our experiments, 500000 tournaments) or the fitness of a new best individual under-runs a given threshold.

It should be emphasized that the GP system is not designed to *directly* evolve scalable quantum circuits. Rather, by scalability we mean that the algorithm does not only work on n but also on n+1 qubits. At least for the 1-SAT problem, scalability of the solutions became "visible", as is shown below.

5 Evolving Quantum Circuits for 1-SAT

The 1-SAT problem for n variables, solved by classical heuristics in $O(n)$ steps, can be solved even faster on a quantum computer. Hogg's quantum algorithm, presented in [9,10], finds a solution in a single search step, using a clever input matrix (see Sect. 2 and Fig. 2). Let R denote this input matrix, with $R_{ss} = i^{c(s)}$ where $c(s)$ is the number of conflicts in the assignment s of a given logical 1-SAT formula in n variables. Thus, the problem description is entirely encoded in this input matrix. Furthermore, let be U the matrix defined by $U_{rs} = 2^{-n/2}(-1)^{d(r,s)}$, where $d(r,s)$ is the Hamming distance between r and s. Then the entire algorithm is the sequential application of Hadamard gates applied to n qubits ($H^{\otimes n}$) initially in state $|0\rangle$, R and U. It can be proven, that the final quantum state is the (equally weighted) superposition of all assignments s with $c(s) = 0$ conflicts.[1] A final measurement will lead, with equal probability, to one of the 2^{n-m} solutions, where m denotes the number of clauses in the 1-SAT formula.

We applied our GP system on problem instances of $n = 2..4$ variables. The number of fitness cases (the number of formulas) is $\sum_{k=1}^{n} \binom{n}{k} 2^k$ in total. Each fitness case consists of an input state (always $|0\rangle^{\otimes n}$), an input matrix for the formula and the desired output. For example,

[1] For all 1-SAT (and also maximally constrained 2-SAT) problems Hogg's algorithm finds a solution with probability one. Thus, an incorrect result definitely indicates the problem is not soluble [9].

$$(|00\rangle, \begin{pmatrix} 1\,0\,0\,0 \\ 0\;i\;0\,0 \\ 0\,0\,1\,0 \\ 0\,0\,0\;i \end{pmatrix}, |-0\rangle)$$

is the fitness case for the 1-SAT formula $\bar{v}_2$ in two variables v_1, v_2. Here, the '$-$' in $|-0\rangle$ denotes a "don't care", since only the rightmost qubit is essential to the solutions $\{v_1 = \text{true/false}, v_2 = \text{false}\}$. That means, an equally weighted superposition of all solutions is not required.

Table 1 gives some parameter settings for GP runs applied to the 1-SAT problem.

Table 1. Parameter settings for the 1-SAT problem with $n = 4$. *)After evolving solutions for $n = 2$ and $n = 3$, intermediate measurements seemed to be irrelevant for searching 1-SAT quantum algorithms, since at least the evolved solutions did not use them. Without intermediate measurements (gate type M), which constitute the tree structure of quantum circuits, tree crossover is not applicable. In GP runs for $n = 2, 3$ the maximum number of measurements was limited by the number of qubits.

Population Size	5000
Tournament Size	16
Basic Gate Types	H,Rx,Ry,Rz,C^kNOT,M
Max. Number of Gates	15
Max. Number of Measurments	0*)
Number of Input Gates	1
Mutation Rate	1
Crossover (XO) Rate	0.1
Linear XO Probability	1*)
Deletion Probability	0.3
Insertion Probability	0.3
Alteration Probability	0.4

For the two-, three- and four-variable 1-SAT problem 100 GP runs were done recording the best evolved quantum algorithm of each run. Finally the over-all best quantum algorithm was determined. For each problem instance our GP system evolved solutions (Figs. 3 and 4) that are essentially identical to Hogg's algorithm. This can be seen at a glance, when noting that $U = Rx[3/4\pi]^{\otimes n}$.[2]

The differences in fitness values of the best algorithms of each GP run, were negligible, though they differed in length and structure, i. e. in the arrangement of gate-types. Most quantum algorithms did not make use of intermediate measurements. Details of the performance and convergence of averaged fitness values over all GP runs can be seen in the three graphs of Fig. 5.

[2] Note, that U is equal to $Rx[3/4\pi]^{\otimes n}$ up to a global phase factor, which of course has no influence on the final measurement results.

```
Misses:     0
Max. Error:     8.7062e-05
Total Error:    0.0015671
Oracle Number:  1
Gate Number:    10
Fitness Value:  0.00025009

Individual:
H 0
H 1
H 2
INP
RX 6.1083 0
RX 2.6001 0
RX 3.0818 0
RX 2.3577 1
RX 2.3562 2
RZ 0.4019 1
```

Fig. 3. Extract from the GP system output: After 100 runs this individual was the best evolved solution to 1-SAT with three variables. Here, INP denotes the specific input matrix R.

```
                                    H 0
                    H 0             H 1
H 0                 H 1             H 2
H 1                 H 2             H 3
INP                 INP             INP
Rx[3/4 Pi] 0        Rx[3/4 Pi] 0    Rx[3/4 Pi] 0
Rx[3/4 Pi] 1        Rx[3/4 Pi] 1    Rx[3/4 Pi] 1
                    Rx[3/4 Pi] 2    Rx[3/4 Pi] 2
                                    Rx[3/4 Pi] 3
```

Fig. 4. The three best, slightly hand-tuned quantum algorithms to 1-SAT with $n = 2, 3, 4$ (from left to right) after 100 evolutionary runs each. Postprocessing was used to eliminate introns, i. e. gates which have no influence on the quantum algorithm or the final measurement results respectively, and to combine two or more rotation gates of the same sort into one single gate. Here, the angle parameters are stated more precisely in fractions of π. INP denotes the input gate R as specified in the text. Without knowledge of Hogg's quantum algorithm, there would be strong evidence for the scalability of this evolved algorithm.

Further GP runs with different parameter settings hinted at strong parameter dependencies. For example, an adequate limitation of the maximum number of gates leads rapidly to good quantum algorithms. In contrast, stronger limitations (somewhat above the length of the best evolved quantum algorithm) made convergence of the evolutionary process more difficult. We experimented also

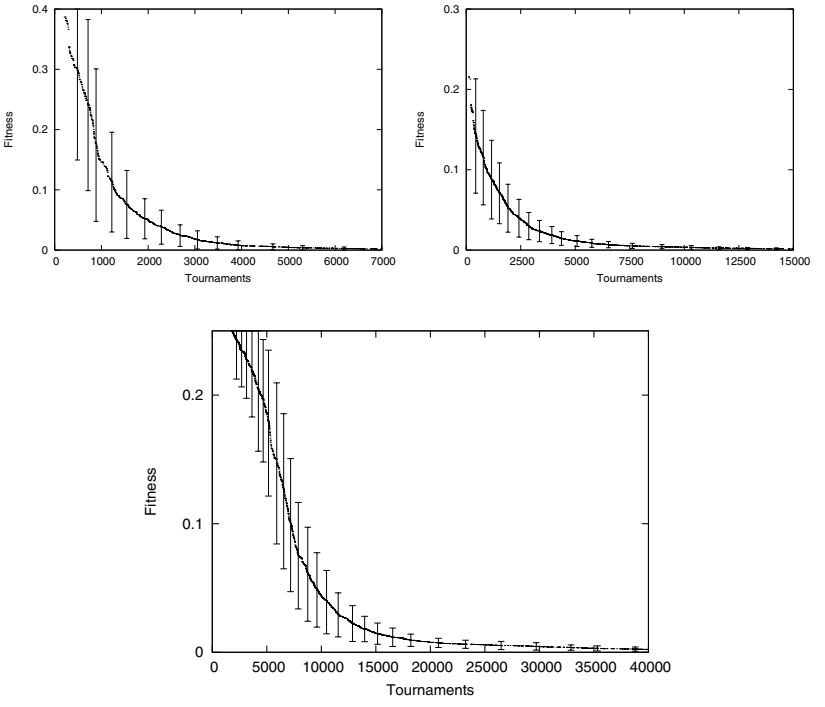

Fig. 5. Three graphs illustrating the course of 100 evolutionary runs for quantum algorithms for the two-, three- and four-variable 1-SAT problem. Errorbars show the standard deviation for the averaged fitness values of the 100 best evolved quantum algorithms after a certain number of tournaments. The dotted line marks averaged fitness values. Convergence of the evolution is obvious.

with different gate sets. Unfortunately, for larger gate sets "visible" scalability was not detectable. GP runs on input gates implementing a logical 1-SAT formula as a permutation matrix, which is a usual problem representation in other quantum algorithms, did not lead to acceptable results, i. e. quantum circuits with zero error probability. This may be explained with the additional problem-specific information (the number of conflicts for each assignment) encoded in the matrix R. The construction of Hogg's input representation from some other representation matrices does not need to be hard for GP at all, but it may require some more ancillary qubits to work. Note, however, that due to the small number of runs with these parameter settings the results do not have statistical evidence.

6 Conclusions

The problems of evolving novel quantum algorithms are evident. Quantum algorithms can be simulated in acceptable time only for very few qubits without excessive computer power. Moreover, the number of evaluations per individual to calculate its fitness are given by the number of fitness-cases usually increases exponentially or even super-exponentially. As a direct consequence, automatic quantum circuit design seems to be feasible only for problems with sufficiently small instances (in the number of required qubits). Thus the examination of scalability becomes a very important topic and has to be considered with special emphasis in the future.

Furthermore, as Hogg's k-SAT quantum algorithm shows, a cleverly designed input matrix is crucial for the outcome of a GP-based evolution. For the 1-SAT problem, the additional tree structure in the linear-tree GP scheme did not take noticeable effect, probably because of the simplicity of the problem solutions.

Perhaps, genetic programming and quantum computing will have a brighter common future, as soon as quantum programs do not have to be simulated on classical computers, but can be tested on true quantum computers.

Acknowledgement. This work is supported by a grant from the Deutsche Forschungsgemeinschaft (DFG). We thank C. Richter and R. Stadelhofer for numerous discussions and helpful comments.

References

[1] H. Barnum, H. Bernstein, and L. Spector, *Better-than-classical circuits for OR and AND/OR found using genetic programming*, 1999, LANL e-preprint quant-ph/9907056.

[2] H. Barnum, H. Bernstein, and L. Spector, *Quantum circuits for OR and AND of ORs*, J. Phys. A: Math. Gen., 33 (2000), pp. 8047–8057.

[3] D. Deutsch, *Quantum theory, the Church-Turing principle and the universal quantum computer*, Proc. R. Soc. London A, 400 (1985), pp. 97–117.

[4] D. Deutsch and R. Jozsa, *Rapid solution of problems by quantum computation*, Proc. R. Soc. London A, 439 (1992), pp. 553–558.

[5] Y. Ge, L. Watson, and E. Collins, *Genetic algorithms for optimization on a quantum computer*, in Proceedings of the 1st International Conference on Unconventional Models of Computation (UMC), C. Calude, J. Casti, and M. Dinneen, eds., DMTCS, Auckland, New Zealand, Jan. 1998, Springer, Singapur, pp. 218–227.

[6] L. Grover, *A fast quantum mechanical algorithm for database search*, in Proceedings of the 28th Annual ACM Symposium on Theory of Computing (STOC), ACM, ed., Philadelphia, Penn., USA, May 1996, ACM Press, New York, pp. 212–219, LANL e-preprint quant-ph/9605043.

[7] J. Gruska, *Quantum Computing*, McGraw-Hill, London, 1999.

[8] M. Hirvensalo, *Quantum Computing*, Natural Computing Series, Springer-Verlag, 2001.

[9] T. Hogg, *Highly structured searches with quantum computers*, Phys. Rev. Lett., 80 (1998), pp. 2473–2476.

[10] T. Hogg, *Solving highly constrained search problems with quantum computers*, J. Artificial Intelligence Res., 10 (1999), pp. 39–66.

[11] W. Kantschik and W. Banzhaf, *Linear-tree GP and its comparison with other GP structures*, in Proceedings of the 4th European Conference on Genetic Programming (EUROGP), J. Miller, M. Tomassini, P. Lanzi, C. Ryan, A. Tettamanzi, and W. Langdon, eds., vol. 2038 of LNCS, Lake Como, Italy, Apr. 2001, Springer, Berlin, pp. 302–312.

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

[13] X. Peng, X. Zhu, X. Fang, M. Feng, M. Liu, and K. Gao, *Experimental implementation of Hogg's algorithm on a three-quantum-bit NMR quantum computer*, Phys. Rev. A, 65 (2002).

[14] B. Rubinstein, *Evolving quantum circuits using genetic programming*, in Proceedings of the 2001 Congress on Evolutionary Computation, IEEE, ed., Seoul, Korea, May 2001, IEEE Computer Society Press, Silver Spring, MD, USA, pp. 114–151. The first version of this paper already appeared in 1999.

[15] L. Spector, *Quantum computation - a tutorial*, in GECCO-99: Proceedings of the Genetic and Evolutionary Computation Conference, W. Banzhaf, J. Daida, A. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. Smith, eds., Orlando, Florida, USA, Jul. 1999, Morgan Kaufmann Publishers, San Francisco, pp. 170–197.

[16] L. Spector, *The evolution of arbitrary computational processes*, IEEE Intelligent Systems, (2000), pp. 80–83.

[17] L. Spector, H. Barnum, H. Bernstein, and N. Swamy, *Finding a better-than-classical quantum AND/OR algorithm using genetic programming*, in Proceedings of the 1999 Congress on Evolutionary Computation, P. Angeline, Z. Michalewicz, M. Schoenauer, X. Yao, and A. Zalzala, eds., Washington DC, USA, Jul. 1999, IEEE Computer Society Press, Silver Spring, MD, USA, pp. 2239–2246.

[18] L. Spector, H. Barnum, H. Bernstein, and N. Swamy, *Quantum Computing Applications of Genetic Programming*, in Advances in Genetic Programming, L. Spector, U.-M. O'Reilly, W. Langdon, and P. Angeline, eds., vol. 3, MIT Press, Cambridge, MA, USA, 1999, pp. 135–160.

[19] A. Steane, *Quantum computation*, Reports on Progress in Physics, 61 (1998), pp. 117–173, LANL e-preprint quant-ph/9708022.

[20] A. Surkan and A. Khuskivadze, *Evolution of quantum algorithms for computer of reversible operators*, in Proceedings of the 2002 NASA/DoD Conference on Evolvable Hardware (EH), IEEE, ed., Alexandria, Virginia, USA, Jul. 2002, IEEE Computer Society Press, Silver Spring, MD, USA, pp. 186–187.

[21] C. Williams and A. Gray, *Automated Design of Quantum Circuits*, in Explorations in Quantum Computing, C. Williams and S. Clearwater, eds., Springer, New York, 1997, pp. 113–125.