Automated Design of Quantum Circuits

Colin P. Williams and Alexander G. Gray

Jet Propulsion Laboratory
4800 Oak Grove Drive
Pasadena, CA 91109-8099
Colin.P.Williams@jpl.nasa.gov, Alexander.G.Gray@jpl.nasa.gov

Abstract. In order to design a quantum circuit that performs a desired quantum computation, it is necessary to find a decomposition of the unitary matrix that represents that computation in terms of a sequence of quantum gate operations. To date, such designs have either been found by hand or by exhaustive enumeration of all possible circuit topologies. In this paper we propose an automated approach to quantum circuit design using search heuristics based on principles abstracted from evolutionary genetics, i.e. using a *genetic programming* algorithm adapted specially for this problem. We demonstrate the method on the task of discovering quantum circuit designs for quantum teleportation. We show that to find a given known circuit design (one which was hand-crafted by a human), the method considers roughly an order of magnitude fewer designs than naive enumeration. In addition, the method finds novel circuit designs superior to those previously known.

1 Introduction: Quantum Circuit Design

1.1 Quantum Computation

Quantum computation is an emerging area of study, which considers the processing of *quantum* information, rather than the familiar classical information. The state of a quantum computer is defined as a superposition of qubits. A computation on such a computer is the unitary evolution of this state, i.e. the action of a unitary matrix operator U upon the state $|\Psi\rangle$. More detailed background on the framework of quantum information processing may be found in [12], [13], and [14].

1.2 Quantum Gates and Circuits

Much recent work has been devoted to the construction of unitary transformations from sequences of more primitive ones. Deutsch ([5]) introduced the notion that such simple unitary operators can be thought of as elementary gates performing logical operations, and more sophisticated operators can be thought of as circuits composed of gates, in analogy to the standard formalism for classical Boolean electrical circuits. This is sometimes called the network model of computation. Following the classical computation line of analysis, in which certain discrete entity. An important complication enters when we wish to allow topologies to have different sizes, i.e. numbers of gates, which we would prefer to leave unspecified when automating circuit design, leaving the algorithm to find the appropriate size. The second is the assignment of angle values within the gates, if applicable; when our gate selection set includes gates which are actually parametric families of gates, there are continuous parameters to be found.

The paper of DiVincenzo and Smolin ([6]) discussed numerical optimization for the discovery of parameters for two-qubit gates, within a fixed circuit topology, which lead to a desired unitary computation. They used this technique to show that certain gates of interest (the Toffoli gate and arbitrary three-qubit gates) could themselves be represented as circuits of two-qubit gates, by finding the necessary two-qubit gate parameters. In order to find the necessary circuit *topologies*, however, all possible topologies were tried. The focus of that paper was to show the *possibility* of decomposing particular computations into circuits of simpler gates; thus exhaustive enumeration was sufficient as a tool to prove the point. We are interested here in a practical and general method for efficiently finding correct circuit topologies for any given operator, in other words avoiding exhaustive enumeration. We return to the continuous aspect of the search problem later in Section 6.

3 Genetic Programming: A Set of Search Heuristics

3.1 Why Genetic Programming?

Our search problem makes a difficult demand on any search method we might think to employ. First, the search method must be amenable to problems in which it is difficult to characterize the structure of the solution space exactly. To clarify this point, consider that our formulation of the problem leaves the form of the target unitary transformation U completely unspecified; no deep knowledge of U's substructure, behavior, relationship to the gates used, or nature otherwise can be used to advantage to eliminate invalid possibilities in the search problem. This very general stance is appropriate for quantum circuit design since human techniques and intuitions about quantum circuits have not reached a mature stage yet; once specific classes of quantum circuits can be delineated, it may be fruitful to design search methods which take advantage of their extra constraints. Furthermore, the quantum circuit design problem is one in which it is difficult to evaluate the best next local move to make at any given point in the search; the entire solution must then be evaluated in order to evaluate the effect of a local change in a circuit candidate. Genetic programming is appropriate in this setting since it relies only on evaluations of entire circuits.

Second, it must be capable of considering solution structures of *variable length*. This is crucial if it is to have any hope of finding small designs; it must be given the latitude to explore solution candidates of different sizes. A particular set of search heuristics, the so-called *genetic programming* method [11], has the distinction of being the *only* search technique having the capability of searching over solutions of varying structure and size. Genetic programming is a type of *genetic algorithm* [8], which in turn is a type of stochastic hill-climbing ([]), or "go with the winners" algorithm ([1]), along with simulated annealing ([9]). Genetic programming is the kind of genetic algorithm which is concerned with non-fixed-length topological structures, rather than the simpler case of fixed-length solutions.

3.2 The Parts of Genetic Programming

Genetic programming is a simple set of search heuristics based loosely on the principles of evolutionary genetics. One of its most distinctive traits is that it is a *population-based* method, or one which maintains multiple solution candidates simultaneously, whose 'evolution' paths may interact with each other. In particular, they may trade substructures in an operation called *"crossover"*, in analogy to sexual reproduction. The method is heavily stochastic, sometimes performing random perturbations on solution candidates (*"mutations"*), and greedily selecting the current best solutions to continue pursuing via random sampling weighted by solution quality (*"fitness"*, *"survival of the fittest"*). A typical genetic programming algorithm has this form:

```
Initialize population with random solutions.

Until the stopping criterion has been reached,
   1. Evaluate the quality of each solution in the population.
   2. Sample from the population, weighted by solution quality, to form the
      'breeding pool'.
   3. For each member of this subset of the population, choose one of the
      following operations to perform on it:
      a. Mutation (choose with probability p(M))
      b. Crossover (choose with probability p(C); requires a partner)
```

Each iteration of the algorithm is called a "generation".

Because its directional guidance is based on evaluations of entire solutions, all that is necessary to apply the algorithm to a problem is a well-defined measure of solution quality; it is thus amenable to problems in which it is difficult to evaluate the best local move to make at each partial solution (such as the circuit design problem). The main power of the method, which distinguishes it from simple stochastic local perturbation, is in the crossover operation. If the problem is one in which we expect substructures to contain localized information, i.e. represent meaningful subsolutions (an analogy to subroutines of a program is useful here), then crossover has a hope of successfully transferring a subsolution to a different solution, perhaps increasing its overall quality. In the circuit design problem, it seems reasonable to expect that transferable subcircuits exist. Crossover is also the main mechanism for obtaining topology candidates of different sizes.

4 A Genetic Programming Algorithm for Quantum Circuit Design

For this investigation we designed a genetic programming algorithm tailored specifically for the problem of quantum circuit design.

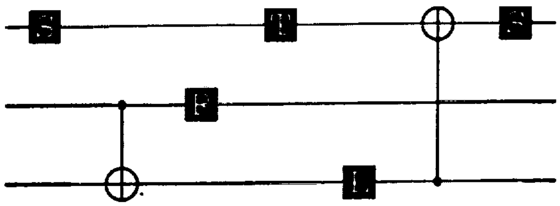

Fig. 1. An example circuit.

4.1 Representation

Circuit Representation. An anonymous quantum circuit is shown in Figure 1 as an example of the representation we use. It is represented as the following nested list data structure, which encodes with each circuit element, its name, parameters if any, and embedding (the wires to which it is connected, followed by the number of wires in the circuit: three in this case):

$$\left\{\begin{array}{c}
\left\{\begin{pmatrix} i & 0 \\ 0 & 1 \end{pmatrix}, params[], \{1;3\}\right\}, \\
\left\{\begin{pmatrix} 1&0&0&0 \\ 0&1&0&0 \\ 0&0&0&1 \\ 0&0&1&0 \end{pmatrix}, params[], \{2,3;3\}\right\}, \\
\left\{\begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix}, params[], \{2;3\}\right\}, \\
\left\{\begin{pmatrix} -1 & 0 \\ 0 & -i \end{pmatrix}, params[], \{1;3\}\right\}, \\
\left\{\begin{pmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix}, params[], \{3;3\}\right\}, \\
\left\{\begin{pmatrix} 1&0&0&0 \\ 0&1&0&0 \\ 0&0&0&1 \\ 0&0&1&0 \end{pmatrix}, params[], \{3,1;3\}\right\}, \\
\left\{\begin{pmatrix} i & 0 \\ 0 & 1 \end{pmatrix}, params[], \{1;3\}\right\}
\end{array}\right\} \tag{1}$$

Gate Selection Set. The algorithm chooses gates from a prespecified selection set. These gates may have unspecified continuous angle parameters associated with them, which must be adjusted by the search algorithm. The gates may also be fixed, or parameterless, gates. In a general setting where little is known about the target transformation, it is sensible to select the gate set such that it forms a universal gate set. It may also be sensible to choose an *overcomplete* set, one which includes a number of gates beyond a computation-universal core subset. This may be useful for obtaining more compact representations, yet may be more costly than having a smaller number of gate types, depending on the technological practicalities of quantum hardware manufacture which hold at the time of the design. An *undercomplete* set may make sense when some known properties of the target computation allow it.

4.2 Evaluation

Solution Quality Measure. To evaluate the quality of a circuit candidate, we compare its matrix form S to the target matrix U using the objective function

$$f(S,U)=\sum_{i=1}^{2^N}\sum_{j=1}^{2^N}|U_{ij}-S_{ij}|, S, U \in U(2^N) \quad (2)$$

This is similar to the objective function used in [6]:

$$f(S,U)=\sum_{i=1}^{2^N}\sum_{j=1}^{2^N}|U_{ij}-S_{ij}|^2, S, U \in U(2^N) \quad (3)$$

We call f the fitness or the *discrepancy*; our goal is to find circuits which minimize the discrepancy between the circuits in our population and the target. When $f = 0$, we have found a circuit which implements U exactly. Otherwise, we have found an approximation to U.

We regard the most sensible evaluation measure as an open question. A paper by Knill [10] considers several measures, many of which are not practically computable, since they take into account all possible states on which the operator may act. One requirement of the measure chosen is that it yields a minimum (maximum) when $S = U$; this property is true of all of Knill's measures. There is a degree of arbitrariness in specifying the proper qualitative behavior of the metric when S differs from U.

While a measure such as f allows the discovery of approximate circuits in a well-defined way, in this paper we focus only upon unitary operations which we can represent *exactly*.

4.3 Selection

Selection is the choosing of a subset from the population to modify in some way. Sampling is weighted by a factor derived from a circuit candidate's discrepancy score, in the way described below, and is performed at the beginning of each generation.

A Ranking-based Scheme. Rather than translate the discrepancy score of a circuit into its selection probability such that the latter is directly proportional to the score, we instead first order the circuits according to their discrepancies, then determine selection probabilities based directly on the resulting rankings. This procedure has the effect of desensitizing the process with respect to the exact discrepancy distribution, which tends to exhibit extreme ratios between the best candidates and the worst ones; we would like to deemphasize such differences in order to avoid complete domination of the selection process by a few candidates too early in the evolution, which corresponds to entrapment in a local optimum.

Selection Probability Distribution. The circuits are ranked from 1 to N, the number of circuits in the population, 1 denoting the best. Probabilities are defined with which to select members of the population for breeding (i.e. crossover), mutation, and other operations which yield modified solution candidates. We desire a functional form yielding probabilities of selection which decrease as the ranking increases (i.e. gets worse), choosing a quadratic form as a compromise between a form yielding a very weak selection effect (which makes the algorithm closer to a purely random search) such a linear decrease, and a form yielding a very aggressive selection effect (making the algorithm more 'greedy', or susceptible to short-term gains which might cause it to become trapped in a local optimum), such as an exponential decrease.

The probability $P(r)$ of selecting the circuit having ranking r is then ar^2+br+c for some a, b, and c. To determine some values for these variables we set up some constraints, namely that $P(r)$ is a true probability, i.e. $\sum_{r=1}^{N} ar^2+br+c=1$, that the lowest ranked member is never picked, i.e. $aN^2+bN+c=0$, and that the derivative of the probability goes to zero as r goes to N, guaranteeing that the probability function is monotonic decreasing. This set of equations yields values of a, b, and c such that

$$P(r)=\frac{6N}{1-3N+2N^2}r^2+\frac{6}{N(1-3N+2N^2)}r-\frac{12}{1-3N+2N^2}. \qquad (4)$$

To derive the new generation's population from the last generation's members, selection from the described probability distribution is performed N times with replacement; note that the population size stays constant and that on average circuits are multiply represented in the next generation a number of times proportional to their fitness. This process yields the *parents* which are fit enough to draw upon for the various modifications (i.e. search operations) that follow.

To finish the activity of this generation, each parent is replaced by a new circuit resulting from an operation performed on it; the operation to be performed on each circuit is chosen from a discrete probability distribution determined by the user of the algorithm.

4.4 Search Operators

Mutation. Mutation is the random perturbation of a single gate, chosen uniformly at random from the gates within the operand circuit. In the case of fixed gates, i.e. gates without parameters which can vary, the selected gate's embedding is changed by uniformly randomly selecting new connecting lines to replace the old ones.

Substitution. Substitution is similar to mutation, but is the replacement of an existing gate chosen uniformly randomly from the gates within the operand circuit, with another one selected from the gate selection set uniformly randomly. Though replacement can be achieved through an appropriate insertion-deletion pair of operations, described below, its inclusion as a separate operation allows its probability of occurence to be more explicitly controlled.

Crossover. The circuit resulting from the crossover, or mating, operation is obtained by considering two parent circuits, A and B. A split point is chosen uniformly randomly somewhere along each of the two parent circuits. The circuit resulting from crossover has the first part of the circuit A attached to the second part of the circuit B, or the first part of the circuit B attached to the second part of the circuit A, each with probability 0.5. Note that crossover allows the size of the resulting circuit to change from that of either A or B.

Transposition. Transposition is an operation obtained by generalizing crossover; its result is also defined by considering two parents A and B. A subcircuit is first defined by the selection of beginning and end points in parent A. The beginning point is chosen uniformly randomly along the length of A, and the end point is chosen uniformly randomly from the region between the that point and the end of A. The resulting circuit is found by inserting the subcircuit at a uniformly randomly chosen point along the length of parent circuit B. This also allows the size of the resulting circuit to change from that of either A or B.

Insertion. Insertion is similar to transposition, except that only one parent need be considered; a randomly constructed sequence of gates is inserted at a random point in the parent, resulting in a larger circuit. The beginning and end points of a subcircuit of the parent are chosen as described for the transposition operator, only so that the length of this subcircuit can be used as the length of the random gate sequence to be inserted. This sequence is constructed by choosing uniformly randomly from the gate selection set the described number of gates.

Deletion. Deletion is the inverse of insertion, in that a random subcircuit is chosen from within the parent; this sequence is deleted from the parent, resulting in a smaller circuit.

5 Experimental Results: Quantum Teleportation Circuits

Quantum teleportation has been identified as an important and interesting application of nonlocal effects in quantum mechanics [3]. Brassard has presented a circuit for the 'send' and 'receive' halves of quantum teleportation in [4]. This circuit is compact, requiring only 4 gates in the 'send' subcircuit and 6 in the 'receive' subcircuit. It is shown in Figure 2. The gate definitions can be found in the example circuit shown in 1 and Figure 1.

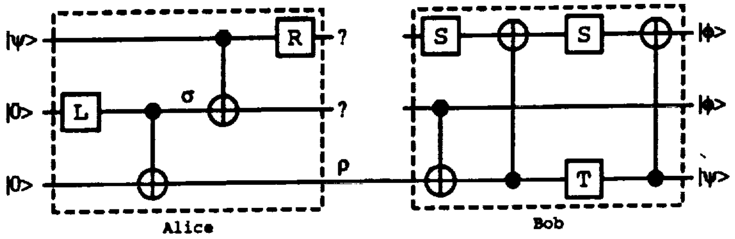

Fig. 2. The quantum teleportation circuit - 'send' and 'receive' parts.

We chose to demonstrate the search algorithm on the computation matrix generated by this circuit, primarily for its general interestingness. Its small size gives the advantage of tractability in the algorithm experimentation phase. Also, because we start with a circuit to obtain the target unitary transform, we know that a compact circuit implementation exists for the problem. We can analyze the computational resources our search method requires to reproduce the hand-designed circuit. As discussed in Section 4.2, using a problem for which an exact circuit representation is known to exist for the gate selection set used avoids the need to consider the appropriateness of the particular fitness measure being used to score inexact circuits.

5.1 The 'Send' Circuit

The algorithm was given the send circuit's computation matrix and a gate selection set consisting of L, R, and XOR. 10 runs were performed, each requiring a different number of generations to find a correct circuit, as follows: 9, 26, 16, 10, 31, 11, 20, 55, 36, 50. 26.4 generations were required on average.

In each case a circuit was found implementing the given computation exactly; although most were different from the original human-designed circuit, all had 4 gates and included at least one each of the L, R, and XOR gates (thus none was necessarily any better than the original circuit). The variance of the number of generations required to find a zero-discrepancy circuit is large, owing to the heavily stochastic nature of the algorithm.

A population size of 100 circuit candidates was used. This is the number of circuit solutions which must be evaluated upon each generation of the algorithm.

Thus, on average, about 2,640 circuits are evaluated for this problem before an answer is found.

By comparison to exhaustive enumeration, the number of possible circuit topologies for this problem, *knowing the number of gates to consider in advance*, can be simply computed as follows: With 3 circuit lines, there are 3 ways to embed the L gate, 3 ways to embed the R gate, and or $\binom{3}{2} = 6$ ways to embed the XOR gate, yielding $3+3+6=12$ different choices for each gate possibility. If we fix the topology size we consider to 4 gates, there are $12^4 = 20,736$ different possible topologies to consider for this problem, using a naive exhaustive approach. Since our search method actually considers circuits of many different sizes, a fair comparison would have to take into account every size class of circuit up to some fairly high number. Our method considered circuits at least as large as 13 gates; note that there are $12^{13} > 10^{14}$ circuits having 13 gates!

We note here that this number does not take into account symmetries and other structure in this search problem, several of which are considered in [6]. Even accounting for these effective reductions of the search space, the computational advantage of a stochastic approach such as the one proposed is still quite significant. Our method may be also be able to take advantage of such information for even greater search efficiency.

Figure 3 shows a typical plot of the average circuit discrepancy over the population at each generation for this problem. The dots on the lower portion of the graph indicate the discrepancy of the best circuit(s) in the population at each generation.

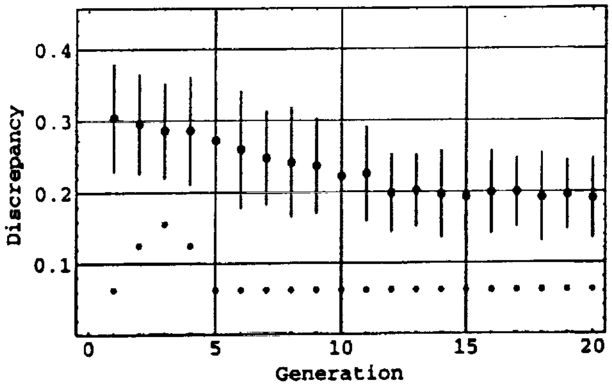

Fig. 3. Typical evolution plot.

5.2 The 'Receive' Circuit

Experiments with the 'receive' part of the circuit demonstrate a further advantage of this approach to automated circuit design beyond achieving a significant savings in time and computational resources. The flexibility and generality of our approach allows the human user to select a gate set of interest and see whether interesting circuits using those gates are found by the search technique. This type of automated search has the potential to find circuits which are difficult for even resourceful and expert human circuit designers to find. This is true especially when a large number of gates is involved; however this small but practical circuit example illustrates that even modest combinatorial problems are very difficult to find optimal answers for, when unaided by computer methods.

Rather than the original set of gates used in [4] for this circuit, consisting of S, T, and XOR, the genetic programming algorithm was given the gate selection set used above, consisting of L, R, and XOR. One of the resulting exact circuits is shown in Figure 4. Comparing this to the original 'receive' part of the human-designed circuit shown in Figure 2, it is clear that the new circuit is smaller (4 gates versus 6), and that the overall teleportation circuit is more elegant since it requires only 3 types of gates, L, R, and XOR, rather than 5 now that S and T are no longer needed.

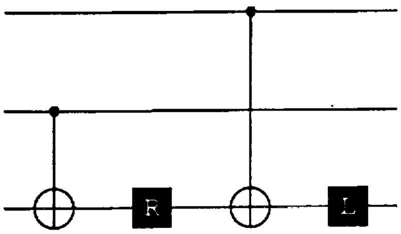

Fig. 4. An efficient circuit found by the search method.

6 Discussion

6.1 Genetic Programming Search as a Tool

At the moment, genetic programming's ability to work with structures of varying sizes makes it the only tool available. Its other primary strength is its effectiveness for opaque problems, where search moves are difficult to evaluate without considering their effect on the entire solution. Rather disappointingly, however, the method's search heuristics are not well-understood formally. For example, issues of convergence, estimated run-time, optimal parameter settings, and behavior dependence on problem context remain empirical issues. Aldous and Vazirani provide one way in which to understand genetic algorithms in general, placing them with simulated annealing in the class of "go with the winners" algorithms ([1]). However, this framework addresses only the 'survival of the fittest' aspect of genetic algorithms, not the effect of the crossover operation, which is one of the hallmarks of genetic algorithms. While much has been written about genetic algorithms, most analyses have been empirical rather than formal. Genetic *programming*, dealing with variable-length structures, is also surely subsumed by some more general model which can be understood formally – unfortunately this has not yet arrived.

On the positive side, its flexible framework allows the practitioner to plug in his or her own heuristics, encoding any prior knowledge of the problem the user may have (for example, regarding the size of the circuit or the types of gates to use). The specifiable gate selection set allows the specification of only the gates available to the user.

6.2 Extension to Continuous Case

The proposed search method can be extended to allow the inclusion of continuous, or parametrized, gates in the gate selection set, as opposed to the fixed gates used in these experiments. This capability requires necessitates greater computational effort since an optimization must be performed to tune the continuous gate parameters of each circuit candidate such that the discrepancy is minimized given the circuit's discrete topology. However, the ability to incorporate continuous gates holds the promise of more compact circuit solutions, as well as better circuit approximations where necessary. Experiments elucidating this approach, as well as several other potentially powerful extensions, will be described in future reports.

7 Conclusions

In this paper we have formalized the problem of automated quantum circuit design as a search problem. We proceeded to propose a search method tailored for this problem. We then demonstrated its usefulness by showing that it is computationally more efficient than naive enumeration. Finally, we demonstrated that it is capable of discovering useful circuits even when the number of gates considered is small, as exemplified by a novel circuit found by our algorithm for quantum teleportation.

7.1 Acknowledgements

The research described in this paper was performed at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration and was supported in part by the Center for Integrated Space Microsystems under task number 277-3ROUO-0 and by the JPL Autonomy Program under task number 234-8AX24-0.

References

1. D. Aldous and U. Vazirani, ????.

2. A. Barenco, 1995. Elementary gates for quantum computation. *Physical Review A*, ????????, pp. ???????. http://xxx.lanl.gov/find/quant-ph/1/Barenco/0/2/0/95/1/0
3. C. H. Bennett, G. Brassard, C. Crepeau, R. Jozsa, A. Peres, and W. Wootters, 1993. Teleporting an unknown quantum state via dual classical and Einstein-Podolsky-Rosen channels. *Physical Review Letters*, **70**, pp. 1895-1899.
4. G. Brassard, 1996. Teleportation as a quantum computation. In T. Toffoli, M. Biafore, and J. Leao (eds.), *Proceedings of the Fourth Workshop on Physics and Computation*, pp. 48-50.
5. D. Deutsch, 1989. Quantum computational networks. In *Proceedings of the Royal Society of London* A**425**, p. 73.
6. D. DiVincenzo and J. Smolin, 1994. Results on two-bit gate design for quantum computers. *IEEE ?????*, ?????, pp. ??????.
7. D. DiVincenzo, 1995. Two-bit gates are universal for quantum computation. *Physical Review A*, **51**, pp. 1015-1022.
8. J. H. Holland, 1975. *Adaptation in Natural and Artificial Systems.* Ann Arbor: University of Michigan Press.
9. S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, 1983. Optimization by Simulated Annealing. *Science*, **220**, pp. 671-680.
10. E. Knill, 1995. Approximation by quantum circuits. LANL Report LAUR-95-2225.
11. J. R. Koza, 1992. *Genetic Programming.* Cambridge: MIT Press.
12. S. Lloyd, 1993. A potentially realizable quantum computer. *Science*, **261**, pp. 1569-1571.
13. A. Steane, 1997. Quantum computing. Review for *Reports on Progress in Physics*, to appear. http://xxx.lanl.gov/find/quant-ph/1/Steane/0/2/0/97/1/0
14. C. P. Williams and S.H. Clearwater, 1998. *Explorations in Quantum Complexity.* Santa Clara: TELOS/Springer-Verlag. (includes CD-ROM.)