Discrete simulations of continuous-time query algorithms that are efficient with respect to queries, gates and space*

Dominic Berry† Richard Cleve†,‡ Sevag Gharibian†

1 Statement of result

The continuous-time query model [3] can be thought of as a variant of the stardard query model, but where arbitrarily small λ-fractional queries to the data $x_1 x_2 \ldots x_N \in \{0,1\}^N$, of the form $|j\rangle \mapsto e^{i\pi\lambda x_j}|j\rangle$, can be made at cost only λ. In the limit as $\lambda \to 0$, such algorithms become continuous-time Hamiltonian evolution processes.

We show that any continuous-time quantum query algorithm whose total query time is T and whose driving Hamiltonian is implementable with G 1- and 2-qubit gates (in a sense defined below) can be simulated by a discrete-query quantum algorithm using the following resources:

- $O(T \log T / \log\log T)$ queries
- $O(T\, G\, \mathrm{polylog}\, T)$ 1- and 2-qubit gates
- $O(\mathrm{polylog}\, T)$ qubits of space.

This extends a previous result [2] where the query cost is the same, but where the orders of the second and third resource costs are at least $T^2 \mathrm{polylog}\, T$ and $T \mathrm{polylog}\, T$ respectively. The present result can also be compared with the recent result [5] where the query cost is superior to ours, $O(T)$ (which is asymptotically optimal), but whose methodology does not (as far as we know) yield an efficient gate construction from an efficiently implementable driving Hamiltonian.

Informally, a driving Hamiltonian $H(t)$ is *implementable with G gates* if the following unitary operation U_H can be simulated with G gates. U_H acts on three registers: a *start time*, a *finish time*, and a *state*. For any start and finish times $t_s, t_f \in [0,T]$ and any state $|\psi\rangle$, $U_H|t_s\rangle|t_f\rangle|\psi\rangle = |t_s\rangle|t_f\rangle|\psi'\rangle$, where $|\psi'\rangle$ is the state that results when state $|\psi\rangle$ evolves under H from time t_s to time t_f. (A technically complete definition, that specifies matters of precision, is omitted from this abstract for reasons of space.)

2 Significance to quantum computation

These new bounds are useful in circumstances where abstract black-box query algorithms are translated into concrete algorithms with subroutines substituted for the black-box queries. In these circumstances, what matters most is the total gate complexity, which can be large if the cost of the operations performed between the queries is large—even if the number of queries is small. An implication of our bound is that, whenever the implementation cost of the driving Hamiltonian is small, the total gate complexity is not much more than the query complexity times the cost of implementing each query.

For specific cases, such as the continuous-time quantum algorithm in [4] for AND-OR tree evaluation, a very efficient discrete-time simulation is known [1]. The contribution of our approach is that it is *systematic*, yeilding a gate-efficient discrete-query algorithm from *any* continuous-time query algorithm where the driving Hamiltonian can be efficiently implemented.

*Research supported by Canada's NSERC, CIFAR, MITACS and the U.S. ARO.

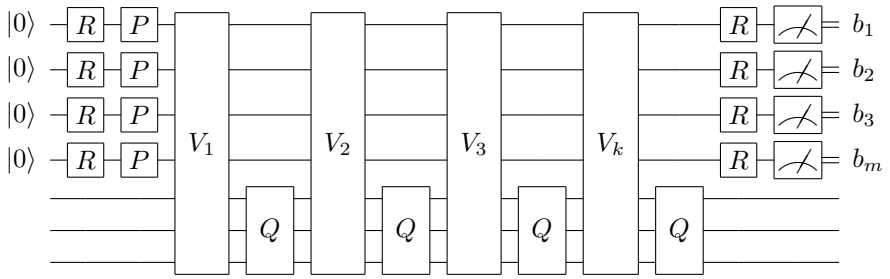
†David R. Cheriton School of Computer Science and Institute for Quantum Computing, University of Waterloo.

‡Perimeter Institute for Theoretical Physics.

3 Sketch of technical contribution

In a nutshell, our result is obtained by simulating the construction in [2], but by representing some of the qubits in a highly compressed form. This compressed form was known by the authors of [2], but it was not known that all of the steps can be carried out within the compressed form—especially the *measurement of control qubits*.

The construction in [2] begins with a fractional query algorithm with total query cost T. This is partioned into segments corresponding to time intervals of the form $[t_0, t_0 + 1/4]$, and with m ($\geq T$) fractional queries of size $1/4m$ in each such interval. In [2] it is shown that each such segment is simulated by a circuit of the form



where $k \in O(\log(T/\varepsilon))$ whose gates are as follows. On the first m qubits (that we refer to as the *control* qubits),

$$P = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}, \qquad R = \begin{pmatrix} \cos\phi & \sin\phi \\ \sin\phi & -\cos\phi \end{pmatrix} \qquad \text{with } \phi \approx 1/\sqrt{8m}. \tag{1}$$

The gates labelled Q are the (full) queries. The gates $V_1, \ldots, V_k$ are the unitaries corresponding to evolving the driving Hamiltonian for various time intervals specified by the control qubits: V_1 for the time interval from t_0 to the relative position of the first 1 in the control qubits; V_2 for the time interval dilineated by the relative positions of the first and second 1s in the control qubits; and so on. The simulation is successful if $b_1 = \cdots = b_m = 0$, which occurs with probability at least 3/4. ([2] shows how to cope with unsuccessful instances.)

The state of the control registers $(\sin\phi|0\rangle + \cos\phi|1\rangle)^{\otimes m}$ is highly compressible in that most of its amplitude is concentrated on basis states with low Hamming weight. A natural compressed representation of this state is in terms of the positions of the 1s in binary. For example, $|0000100010000100\rangle$ can be represented as $|5\rangle|9\rangle|14\rangle$. To simplify the exposition here, we assume that m is polynomial in T/ε so that the positions can be represented using $k \in O(\log(T/\varepsilon))$ qubits; however, the methodology works even if m is much larger (in that case, it suffices to represent the positions of the 1s approximately, within precision polynomial in T/ε).

The stages of the above circuit can be simulated with the control qubits in compressed form as follows.

Initialization of the control qubits ($R^{\otimes m}|0^m\rangle$): The *encoded* qubits are initially in state $|0^k\rangle^{\otimes k}$. This is first mapped to a state that is approximately

$$\left(\sum_{s_1=0}^{2^k-1} \sin\phi(\cos\phi)^{s_1}|s_1\rangle\right)\left(\sum_{s_2=0}^{2^k-1} \sin\phi(\cos\phi)^{s_2}|s_2\rangle\right)\ldots\left(\sum_{s_k=0}^{2^k-1} \sin\phi(\cos\phi)^{s_k}|s_k\rangle\right). \tag{2}$$

Then a series of additions are performed: add the first k-qubit register to the second; add the second to the third, etc. Intuitively, the reason why this works is a quantum analogue of the fact that the random variable "position of the next 1" in a binomial distribution follows an exponential distribution.

Queries and driving operations: Our definition of driving Hamiltonian implementation cost fits perfectly in this context. In the compressed representation, V_1 is the implementation of the driving Hamiltonian with t_s hardwired to 0 and t_f controlled by the first k-qubit register. V_2 is the implementation of U_H with t_s controlled by the first k-qubit register and t_f controlled by the second k-qubit register, and so on.

Measurement of the control qubits ($R^{\otimes m}$ and measure): What remains is to perform the final measurement. This should logically correspond to what happens if the state is uncompressed to m qubits and then, for each qubit, an R gate is applied and it is measured in the computational basis. However, this cannot be *literally* implemented this way, because it would increase the gate and space usage to that in [2]; our task is to *logically* perform this, but without uncompressing. In the compressed form, the measured registers would contain the *addresses* of the 1s in the final measurement (with high probability, there are $O(\log(T/\varepsilon))$ of them).

Our first observation is that we can perform an *incomplete* measurement that captures a seemingly small part of what we are seeking: we can cause the state to either collapse to $|0^m\rangle$ or to the subspace that is the orthogonal complement of this state—and with the correct probabilities. This is achieved by performing the inverse of the initialization unitary and then the measurement that distinguishes between state $|0^k\rangle^{\otimes k}$ and all other states of these registers (but in this case remaining in coherent superposition of these other states).

Our idea is to complete the measurement by applying the above procedure recursively, on the two halves of the logical string—where we are making use of the fact that the underlying operation that we are simulating has a tensor product structure. We first explain what is being implemented in terms of the logical (uncompressed) data and then how to implement it on the compressed data. The first step is to apply the incomplete measurement on the m-qubit string. If the outcome is $|0^m\rangle$ then we halt with that outcome; otherwise, we consider the two halves of the string and apply the same procedure. That is, we perform the procedure on the first $m/2$ qubits that distinguishes them from $|0^{m/2}\rangle$ and then the same procedure again on the second $m/2$ qubits. Whenever a collapse to a $|0^r\rangle$ state occurs, a branch of the measurement proccess ends; otherwise, we keep refining. Note that, since the Hamming weight of the outcome is, with high probability, bounded by $k \in O(\log(T/\varepsilon))$, at every level of depth in the recursion there are at most k active branches.

Now we explain how to perform the above logical procedure on the compressed data. The main problem is that, in compressed form, it is not clear how to extract the left half and the right half of an m-qubit string without uncompressing the string. Our idea here is to use the following *graduated compression scheme*.

Graduated compression scheme: We first consider strings of Hamming weight up to 1, where the most compressed form is to either store a special symbol "null", which indicates that the string is all zeroes, or to store the location of the 1 in binary. For example, for $x = 0000000001000000$, the compressed representation is 1001. In intermediate compressed forms, the location of the 1 is represented partly by the placement of information and partly by a binary string. Here is a series of intermediate compressed forms for $x = 0000000001000000$:

$$\begin{array}{rcl}
C_4(x) & = & (\quad 1001 \quad) \\
C_3(x) & = & (\quad \text{null}, \quad 001 \quad) \\
C_2(x) & = & (\quad \text{null}, \quad \text{null}, \quad 01, \quad \text{null} \quad) \\
C_1(x) & = & (\ \text{null}, \ \text{null}, \ \text{null}, \ \text{null}, \ 1, \ \text{null}, \ \text{null}, \ \text{null}\ ) \\
x & = & (\ 0, \ 0, \ 0, \ 0, \ 0, \ 0, \ 0, \ 0, \ 0, \ 1, \ 0, \ 0, \ 0, \ 0, \ 0, \ 0\ ).
\end{array}$$

In $C_3(x)$, placement indicates that the 1 is in the second half, and similarly for $C_2(x)$ and $C_1(x)$.

It is straightforward to extend this compression scheme for Hamming weight up to k, by concatenating k copies of it (one for each potential 1). It is easy to transform $C_j(x)$ to $C_{j-1}(x)$, though the length of the encoding can approximately double by doing so. The advantage of encoding $C_{j-1}(x)$ is that it contains the two halves of the logical string in separate registers, each half in a compressed form. This is exactly what is required to perform the logical operation described above. Whenever a logical outcome $|0^r\rangle$ is obtained, it need not be represented as a quantum state. Hence, at most $k^2 \in O(\log^2(T/\varepsilon))$ qubits are active during the process.

References

[1] A. Ambainis, A. M. Childs, B. W. Reichardt, R. Špalek, and S. Zhang, Any AND-OR formula of size N can be evaluated in time $N^{1/2+o(1)}$ on a quantum computer, In *Proc. 48th IEEE Symposium on Foundations of Computer Science*, pp. 363–372 (2007).

[2] R. Cleve, D. Gottesman, M. Mosca, R. Somma, and D. Yonge-Mallo, Efficient discrete-time simulations of continuous-time quantum query algorithms, In *Proc. 41st ACM Symposium on Theory of Computing*, pp. 409–416 (2009).

[3] E. Farhi and S. Gutmann, Analog analogue of a digital quantum computation, *Physical Review A*, **57**:2403–2406 (1998).

[4] E. Farhi, J. Goldstone, and S. Gutmann, A quantum algorithm for the Hamiltonian NAND tree, *Theory of Computing*, **4**(8):169–190 (2008).

[5] T. Lee, R. Mittal, B. W. Reichardt, R. Spalek, and M. Szegedy, Quantum query complexity of state conversion, To appear in *Proc. 52nd IEEE Symposium on Foundations of Computer Science* (2011). arXiv:1011.3020