QIC710/CS667/CO681/PH767/AM871 Quantum Information Processing

Classical Lower Bound for Simon's Problem

Richard Cleve, Fall 2011

We prove that any classical (possibly probabilistic) algorithm for Simon's problem that succeeds with probability at least 3/4 must make $\Omega(\sqrt{2^n})$ queries. The proof uses some standard techniques that arise in computational complexity; however, this account assumes no prior background in the area.

The first part of the proof is to "play the adversary" by coming up with a way of generating an instance of f that will be hard for any algorithm. Note that picking some *fixed* f will not work very well. A fixed f has a fixed r associated with it and the first two queries of the algorithm *could* be 0^n and r, which would reveal r to the algorithm after only two queries. Rather, we shall *randomly* generate instances of f. First, we pick r at random, uniformly from $\{0,1\}^n - 0^n$ (we exclude the all-zero string to avoid a special case, and make the proof technically easier to write). Picking r does not fully specify f but it partitions $\{0,1\}^n$ into 2^{n-1} colliding pairs of the form $\{x, x \oplus r\}$, for which $f(x) = f(x \oplus r)$ will occur. Let us also specify a representative element form each colliding pair, say, the smallest element of $\{x, x \oplus r\}$ in the lexicographic order. Let T be the set of all such representatives: $T = \{s : s = \min\{x, x \oplus r\}$ for some $x \in \{0,1\}^n\}$. Then we can define f in terms of a random one-to-one function $\phi : T \to \{0,1\}^n$ uniformly over all the $2^n(2^n - 1)(2^n - 2) \cdots (2^n - 2^{n-1} + 1)$ possibilities. The definition of f can then be taken as

$$f(x) = \begin{cases} \phi(x) & \text{if } x \in T \\ \phi(x \oplus r) & \text{if } x \notin T. \end{cases}$$

We shall prove that no classical probabilistic algorithm can succeed with probability 3/4 on such instances unless it makes a very large number of queries.

The next part of the proof is to show that, with respect to the above distribution among inputs, we need only consider *deterministic* algorithms (by which we mean ones that make no probabilistic choices). The idea is that any probabilistic algorithm is just a probability distribution over all the deterministic algorithms, so its success probability p is the average of the success probabilities of all the deterministic algorithms (where the average is weighted by the probabilities). At least one deterministic algorithm must have success probability $\geq p$ (otherwise the average would be less than p). Therefore we need only consider deterministic algorithms.

Next, consider some deterministic algorithm and the first query that it makes: $(x_1, y_1) \in \{0,1\}^n \times \{0,1\}^n$, where x_1 is the input to the query and y_1 is the output of the query. The result of this will just be a uniformly random element of $\{0,1\}^n$, independent of r. Therefore the first query by itself contains absolutely no information about r.

Now consider the second query (x_2, y_2) (without loss of generality, we can assume that the inputs to all queries are different; otherwise, the redundant queries could be eliminated from the algorithm). There are two possibilities: $x_1 \oplus x_2 = r$ (collision) or $x_1 \oplus x_2 \neq r$ (no collision). In the first case, we will have $y_1 = y_2$ and so the algorithm can deduce that $r = x_1 \oplus x_2$. But the first case arises with probability only $\frac{1}{2^n-1}$. With probability $1 - \frac{1}{2^n-1}$, we are in the second case, and all that the algorithm deduces about r is that $r \neq x_1 \oplus x_2$ (it has ruled out just one possibility among $2^n - 1$).

We continue our analysis of the process by induction on the number of queries. Suppose that $k-1$ queries, $(x_1, y_1), \ldots, (x_{k-1}, y_{k-1})$ have been made without any collisions so far. (No collision so far means that, for all $1 \leq i < j \leq k-1$, $y_i \neq y_j$.) Then all that has been deduced about r is that it is not $x_i \oplus x_j$ for all $1 \leq i < j \leq k-1$. In other words, up to $(k-1)(k-2)/2$ possibilities for r have been eliminated. When the next query (x_k, y_k) is made, the number of potential collisions arising from it are at most $k-1$ (there are $k-1$ previously made queries to collide with). Therefore, the probability of a collision at query k is at most

$$\frac{k-1}{2^n - 1 - (k-1)(k-2)/2} \leq \frac{2k}{2^{n+1} - k^2}. \tag{1}$$

Since the collision probability bound in Eq. (1) holds all k, the probability of a collision occurring somewhere among m queries is at most the sum of the right side of Eq. (1) with k varying from 1 to m:

$$\sum_{k=1}^{m} \frac{2k}{2^{n+1} - k^2} \leq \sum_{k=1}^{m} \frac{2m}{2^{n+1} - m^2} \leq \frac{2m^2}{2^{n+1} - m^2}. \tag{2}$$

If this quantity is to be at least 3/4 then

$$\frac{2m^2}{2^{n+1} - m^2} \geq \frac{3}{4}. \tag{3}$$

It is an easy exercise to solve for m in the above inequality, yielding

$$m \geq \sqrt{\tfrac{6}{11} 2^n}, \tag{4}$$

which gives the desired bound.

Actually, there is a slight technicality remaining. We have shown that $\sqrt{(6/11)2^n}$ queries are necessary *to attain a collision* with probability 3/4; whereas the algorithm is not technically required to make queries that include a collision. Rather, the algorithm is just required to deduce r, and it is conceivable that an algorithm could deduce r some other way without a collision occurring. But any algorithm that deduces r can be modified so that it makes one additional query that collides with a previous one. Hence, we have a slightly smaller lower bound of $\sqrt{(6/11)2^n} - 1$, but this is still $\Omega(\sqrt{2^m})$.