Quantum pattern matching fast on average

Ashley Montanaro
Department of Computer Science, University of Bristol, UK.

Technical version: `arXiv:1408.1816`

One of the most fundamental tasks in computer science is pattern matching: finding some desired data (the *pattern*) within a larger data set (the *text*). This problem has been of interest for decades, both in its own right and as part of more complicated questions in text processing, bioinformatics and image processing.

Here we consider the d-dimensional pattern matching problem, for arbitrary $d = O(1)$. Two examples of this problem are shown in Figure 1 below. We are given access to a text T and a pattern P over an alphabet Σ with $|\Sigma| = q$. Our task is to find an instance of P within T, if such an instance exists. That is, writing $[n] := \{0, \dots, n-1\}$ and thinking of T and P as functions $T : [n]^d \to \Sigma$, $P : [m]^d \to \Sigma$, we are required to output $s \in [n-m]^d$ such that $T(s+x) = P(x)$ for all $x \in [m]^d$, if such an s exists; otherwise, we should output "not found". We call any function of the form $S : [k]^d \to \Sigma$ a *string*, and think of strings interchangeably as functions or $k \times \cdots \times k$ arrays of elements of Σ.

The classical KMP algorithm of Knuth, Morris and Pratt [7] from 1977 solves the pattern matching problem for $d = 1$ in time $\Theta(n+m)$ in the worst case. This is clearly optimal, as every classical pattern-matching algorithm which is correct on all inputs must inspect every character of the pattern and the text. However, significantly improved runtimes can be achieved for more typical inputs. Consider a model where each character of the text is chosen at random from Σ, and the pattern is either uniformly random too (in which case, if it is long enough, it will not match the text with high probability), or is chosen to be a random substring of the text. A simple algorithm was given by Knuth [7, Section 8] which runs in time $O(n(\log_q m)/m + m)$ with high probability on such random inputs, while still preserving efficient worst-case behaviour. Observe that this runtime is substantially sublinear for large m, but never better than $O(\sqrt{n \log n})$.

Shortly after this algorithm was developed, Yao proved an $\Omega((n/m) \log_q m)$ lower bound for the 1-dimensional matching problem, for random text and pattern [11]. The bound extends to give a $\Omega((n/m)^d (\log_q m))$ lower bound for the d-dimensional problem [6]. More recently, an algorithm which runs in time $O((n/m)^d \log_q m + m^d)$ for the general d-dimensional problem, for random text and pattern, was given by Kärkkäinen and Ukkonen [6]. This is thus optimal up to the $O(m^d)$ term, which corresponds to preprocessing time for the pattern.

A quantum pattern-matching algorithm for the 1-dimensional case has been presented by Ramesh and Vinay [10]. The algorithm runs in time $\widetilde{O}(\sqrt{n} + \sqrt{m})$ and hence achieves a square-root speedup over the best possible classical algorithm's worst-case complexity. However, the sublinear classical results mentioned above raise the following question: could there be a quantum pattern-matching algorithm which significantly outperforms its classical counterparts on average-case inputs which are more likely to occur in practice?

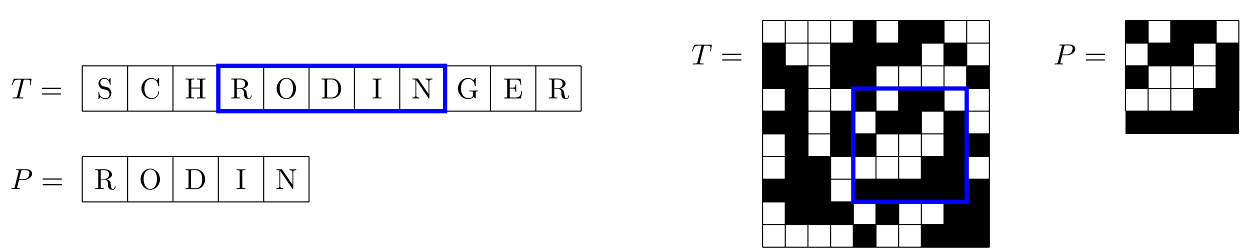

Figure 1: Examples of 1D and 2D pattern matching problems, with matches highlighted.

Statement of results

We give a quantum algorithm which, for most instances of the d-dimensional pattern matching problem, is super-polynomially faster than the best possible classical algorithm.

Theorem 1. *Assume $m = \omega(\log n)$. Let $T : [n]^d \to \Sigma$ be picked uniformly at random. Let $P : [m]^d \to \Sigma$ be picked either (a) by choosing an arbitrary $m \times \cdots \times m$ substring of T, or (b) by choosing each element of P uniformly at random from Σ. Then there is a quantum algorithm which runs in time $\widetilde{O}((n/m)^{d/2} 2^{O(d^{3/2}\sqrt{\log m})})$ and determines which is the case. In case (a), the algorithm also outputs the position at which P matches T. The algorithm fails with probability $O(1/n^d)$, taken over both the choice of T and P, and the algorithm's internal randomness.*

Any classical bounded-error algorithm for the same problem must make $\widetilde{\Omega}((n/m)^d + n^{d/2})$ queries to T and P in total.

The $\widetilde{O}$, $\widetilde{\Omega}$ notation suppresses factors logarithmic in m and n. The time complexity is stated in the standard quantum circuit model, assuming that a query to T or P uses time $O(1)$. We can think of T and P as either easily evaluated oracle functions in the query complexity model, or data stored in an efficiently accessible quantum random-access memory [4].

Observe that, for any fixed d, $2^{O(d^{3/2}\sqrt{\log m})} = o(m^\epsilon)$ for any $\epsilon > 0$. When m is large, Theorem 1 thus demonstrates a super-polynomial separation between quantum and classical complexity (when m is small, e.g. $O(\log n)$, straightforward use of Grover's algorithm is faster). For example, when $m = \Omega(n)$, we get a quantum algorithm running in time $\widetilde{O}(2^{O(d^{3/2}\sqrt{\log n})})$, as opposed to the best classical complexity of $\widetilde{\Omega}(n^{d/2})$. The omitted constants in the $O(d^{3/2}\sqrt{\log m})$ term in the exponent are not unreasonably high. For $d = 1$, for example, the algorithm's runtime is $\widetilde{O}(\sqrt{n/m}\, 2^{2.68\ldots\sqrt{\log_2 m}})$. Theorem 1 is a rare example of a super-polynomial average-case separation between quantum and classical computation for a natural problem and a natural distribution on the input. An exponential average-case separation was previously proven [2] for a related problem (an oracular hidden shift problem over $\mathbb{Z}_2^n$), but that problem is arguably less natural than pattern matching.

Theorem 1 is based on a more general pattern matching result, which holds for non-random patterns and texts. In order to state this result more formally, we need some notation. For any string $S : [n]^d \to \Sigma$, we define a new string $S^{\rhd k} : [n-k+1]^d \to \Sigma^{k^d}$, where $S^{\rhd k}(x_1, \ldots, x_d)$ is equal to the size $k \times \cdots \times k$ substring of S beginning at position $x_1, \ldots, x_d$. Formally, for any $f : [n]^d \to \Sigma$, $s \in [n]^d$, $k \in [n-s]$, let $f_{s,k} : [k]^d \to \Sigma$ be defined by $f_{s,k}(z_1, \ldots, z_d) = f(s_1 + z_1, \ldots, s_n + z_n)$. Then

$$S^{\rhd k}(x_1, \ldots, x_d) = S_{s,k}.$$

An example of this operation is shown in Figure 2.

Define the *m-injectivity length* of S, $\upsilon(S, m)$, to be the minimal k such that $S^{\rhd k}_{s,m}$ is injective for all $s \in [n-m]^d$. Thus $\upsilon(S, m) \le \nu$ if every $m \times \cdots \times m$ substring of $S^{\rhd \nu}$ is injective. For any S, $1 \le \upsilon(S, m) \le m$. Finally, define $\upsilon(S) := \upsilon(S, n)$; $\upsilon(S)$ is the minimal k such that $S^{\rhd k}$ is injective. Then the most general result we have is as follows:

Theorem 2. *Fix $d = O(1)$. Let $T : [n]^d \to \Sigma$ and $P : [m]^d \to \Sigma$ satisfy $\upsilon(T, m), \upsilon(P) \le \nu \le m/2$, for some ν. Further assume that, for every offset s such that P does not match T at that offset, the fraction of positions $x \in [m]^d$ where $P(x) \ne T(x + s)$ is at least γ. Then there is a bounded-error quantum algorithm which outputs $s \in [m]^d$ such that P matches T at offset s, if such an s exists; otherwise, the algorithm outputs "not found". The algorithm makes*

$$O\left(\left(\frac{n \log^2 m\, 2^{\sqrt{(2\log_2 3) d \log_2 m}}}{m}\right)^{d/2} \left(\nu \log m\, 2^{\sqrt{(2\log_2 3) d \log_2 m}} + \frac{1}{\sqrt{\gamma}}\right)\right)$$

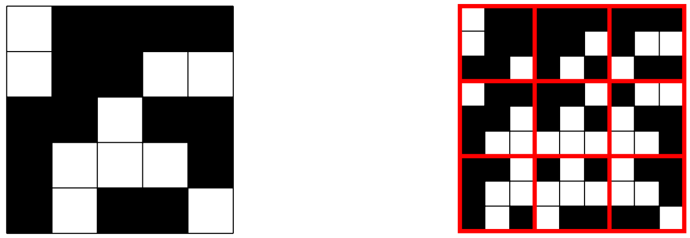

Figure 2: Converting a non-injective 2D string S into an injective string $S^{\triangleright 3}$.

queries to each of T and P. The runtime is the same up to a polylog(m) *factor.*

Theorem 2 may appear somewhat hard to digest. The intuition is that the algorithm is efficient, i.e. has runtime close to $O((n/m)^{d/2})$, when: the strings formed by concatenating all short substrings of both T and P are injective; and offsets where there is no match can be efficiently tested and discarded. The algorithm can thus be seen as achieving a speedup in a scenario somewhat similar to that considered in the field of property testing [9], where it has the promise that each potential match is either actually a match, or is far from being a match.

Techniques

Theorem 2 is ultimately based around the use of a quantum algorithm for finding hidden shifts in injective functions $f : \mathbb{Z}_{2^n}^d \to \Sigma$. The algorithm is a variant of algorithms of Kuperberg [8]. Kuperberg's work described several algorithms: two for finding hidden shifts in injective functions $f : \mathbb{Z}_{2^n} \to \Sigma$, and one for finding hidden shifts in general abelian groups. The algorithm given here achieves essentially the same asymptotic complexity as the best algorithm given in [8], and appears somewhat simpler to analyse. In particular, we include a full proof of its correctness and complexity.

To use the algorithm, we first make the pattern and text injective. This is similar to the "injectivisation" idea used by Gharibi [3] in the context of quantum algorithms for abelian hidden shift problems, but here we need a slightly different notion, as used by Knuth [7], to ensure we preserve matching after injectivisation. We then apply the hidden shift algorithm by guessing an offset where the pattern matches the text. If our guess is fairly close, then the algorithm succeeds in finding the actual offset where the pattern matches. This guessing process is then wrapped within the use of the bounded-error variant of Grover's search algorithm [5] to obtain the final result. Theorem 1 is then derived by simply calculating the quantities ν, γ that occur in Theorem 2 for random strings.

Kuperberg showed in [8] that, based on a similar idea of guessing offsets, his algorithms gave a super-polynomial quantum speedup for the task of finding an injective pattern of length m, promised to be hidden in an injective text of length $2m$. The contribution here is thus to generalise this idea to arbitrary dimensions $d > 1$, to remove the restriction on the length of the text, and to relax the injectivity constraint. We also modify the promise that the pattern is guaranteed to be contained in the text to the promise that any non-matches can be tested efficiently. Observe that a constraint of this form is required if one seeks a runtime which is $o(m^{d/2})$. Imagine we are told an offset at which the pattern is claimed to match the text. If we have no lower bound on the number of positions at which it does not match the text if the claim is false, then verifying this claimed match would require $\Omega(m^{d/2})$ quantum queries in the worst case [1].

References

[1] C. Bennett, E. Bernstein, G. Brassard, and U. Vazirani. Strengths and weaknesses of quantum computing. *SIAM J. Comput.*, 26(5):1510–1523, 1997. `quant-ph/9701001`.

[2] D. Gavinsky, M. Roetteler, and J. Roland. Quantum algorithm for the Boolean hidden shift problem. In *Proc. 17th International Computing & Combinatorics Conference (COCOON'11)*, pages 158–167, 2011. `arXiv:1103.3017`.

[3] M. Gharibi. Reduction from non-injective hidden shift problem to injective hidden shift problem. *Quantum Inf. Comput.*, 13(3&4):221–230, 2013. `arXiv:1207.4537`.

[4] V. Giovannetti, S. Lloyd, and L. Maccone. Quantum random access memory. *Phys. Rev. Lett.*, 100:160501, 2008. `arXiv:0708.1879`.

[5] P. Høyer, M. Mosca, and R. de Wolf. Quantum search on bounded-error inputs. In *Proc. 30th International Conference on Automata, Languages and Programming (ICALP'03)*, pages 291–299, 2003. `quant-ph/0304052`.

[6] J. Kärkkäinen and E. Ukkonen. Two and higher dimensional pattern matching in optimal expected time. In *Proc. 5th ACM-SIAM Symp. Discrete Algorithms*, pages 715–723, 1994.

[7] D. Knuth, J. Morris, Jr., and V. Pratt. Fast pattern matching in strings. *SIAM J. Comput.*, 6(2):323–350, 1977.

[8] G. Kuperberg. A subexponential-time quantum algorithm for the dihedral hidden subgroup problem. *SIAM J. Comput.*, 35(1):170–188, 2005. `quant-ph/0302112`.

[9] A. Montanaro and R. de Wolf. Quantum property testing, 2013. `arXiv:1310.2035`.

[10] H. Ramesh and V. Vinay. String matching in $\widetilde{O}(\sqrt{n}+\sqrt{m})$ quantum time. *Journal of Discrete Algorithms*, 1:103–110, 2003. `quant-ph/0011049`.

[11] A. Yao. The complexity of pattern matching for a random string. *SIAM J. Comput.*, 8(3):368–387, 1979.