Information and Computation 261 (2018) 265–280

Contents lists available at ScienceDirect

Information and Computation

www.elsevier.com/locate/yinco

Block interpolation: A framework for tight exponential-time counting complexity

Radu Curticapean [1]

Institute for Computer Science and Control, Hungarian Academy of Sciences, MTA SZTAKI, Kende u. 13–17, Budapest, Hungary

ARTICLE INFO

Article history:
Received in revised form 8 April 2016
Available online 8 February 2018

Keywords:
Exponential-time hypothesis
Counting complexity
Permanent
Matching polynomial
Independent set polynomial
Tutte polynomial

ABSTRACT

We devise a framework for proving tight lower bounds under the counting exponential-time hypothesis #ETH introduced by Dell et al. (2014) [18]. Our framework allows us to convert classical #P-hardness results for counting problems into tight lower bounds under #ETH, thus ruling out algorithms with running time $2^{o(n)}$ on graphs with n vertices and $O(n)$ edges. As exemplary applications of this framework, we obtain tight lower bounds under #ETH for the evaluation of the zero-one permanent, the matching polynomial, and the Tutte polynomial on all non-easy points except for one line. This remaining line was settled very recently by Brand et al. (2016) [24].

1. Introduction

Counting complexity is a classical subfield of complexity theory, launched by Valiant's seminal paper [2] that introduced the class #P and proved #P-hardness of the zero-one permanent, a problem that may equivalently be considered as asking for the number of perfect matchings in a bipartite graph. This initial breakthrough spawned an ongoing research program that systematically studies the complexity of computational counting problems, and many results in this area can be organized as dichotomy results. Such results show that, among problems that can be expressed in certain rich frameworks, each problem is either polynomial-time solvable or #P-hard. Moreover, these results often give criteria for deciding which side of the dichotomy a given problem occupies. For instance, a full dichotomy was shown for the problems of counting solutions to constraint-satisfaction problems [3,4], and similar results are known for large subclasses of so-called Holant problems [5,6], and for the evaluation of graph polynomials such as the Tutte polynomial [7] and the cover polynomial [8,9].

Over the course of the counting complexity program, it became clear that most interesting counting problems are #P-hard, and that the class of polynomial-time solvable problems is rather limited, nevertheless containing some surprising examples, such as counting perfect matchings in planar graphs, counting spanning trees, and problems amenable to holographic algorithms [10]. To attack the large body of hard problems, several relaxations were studied, such as approximate counting [11–13], counting modulo fixed numbers [14,15], and counting on restricted graph classes, such as planar and/or 3-regular graphs [16,17].

In this paper, we follow an avenue of relaxations recently introduced by Dell et al. [18] and consider the possibility of sub-exponential exact algorithms for counting problems. More precisely, we rule out such algorithms for various counting problems under popular complexity-theoretic assumptions. For instance, we can clearly count perfect matchings on m-edge

E-mail address: radu.curticapean@gmail.com.

[1] Supported by ERC Starting Grant PARAMTIGHT (No. 280152) and ERC Consolidator Grant SYSTEMATICGRAPH (No. 725978). Part of this work was done at Saarland University and appeared in the author's PhD thesis [1]. Another part was done while visiting the Simons Institute for the Theory of Computing.

https://doi.org/10.1016/j.ic.2018.02.008
0890-5401/

graphs in time $2^{O(m)}$ by brute-force, but is there a chance of obtaining a running time of $2^{o(m)}$? An unconditional negative answer would imply the separation of FP and #P, so our results need to rely upon additional hardness assumptions: We build upon the *exponential-time hypothesis* #ETH, introduced in [18], which we may consider for now as the hypothesis that the satisfying assignments to 3-CNF formulas φ on n variables cannot be counted in time $2^{o(n)}$. This hypothesis is trivially implied by the better-known and widely-believed decision version ETH, introduced in [19,20], which assumes the same lower bound for *deciding* the satisfiability of φ.

Dell et al. [18] were able to prove almost-tight lower bounds under #ETH for a variety of counting problems: For instance, they could rule out algorithms with running time $2^{o(n/\log n)}$ for the zero-one permanent on graphs with n vertices and $O(n)$ edges. Similar lower bounds were shown for counting vertex covers, and for most points of the Tutte polynomial.

1.1. Hardness via polynomial interpolation

The lower bounds in [18] are obtained via polynomial interpolation, one of the most prominent techniques for non-parsimonious reductions between counting problems [21,7,16,17,22,23,18]. To illustrate this technique, and for the purposes of further exposition, let us reduce counting *perfect matchings* to counting *matchings* (that are not necessarily perfect), using a standard argument similar to [16]. In the following, let G be a graph with n vertices. We wish to obtain the number of perfect matchings in G by querying an oracle for counting matchings in arbitrary graphs.

Step 1 – Set up interpolation: For $k \in \mathbb{N}$, let m_k denote the number of matchings with exactly k unmatched vertices in G. In particular, m_0 is equal to the number of perfect matchings in G. For an indeterminate x, define a polynomial μ via

$$\mu(x) = \sum_{k=0}^{n} m_k \cdot x^k \tag{1}$$

and observe that its degree is n. Hence, we could use Lagrange interpolation to recover all its coefficients if we were given the evaluations of μ at $n+1$ distinct input points. In particular, this would give us the constant coefficient m_0, which counts the number of perfect matchings in G.

Step 2 – Evaluate the polynomial with gadgets: We can evaluate $\mu(t)$ at points $t \in \mathbb{N} \setminus \{0\}$ by a reduction to counting matchings: For $t \in \mathbb{N}$ with $t \geq 1$, define a graph G_t from G by adding, for each vertex $v \in V(G)$, a *gadget* that consists of an independent set of $t-1$ fresh vertices together with edges from all of these vertices to v. Then it can be checked that $\mu(t)$ is equal to the number of matchings in G_t: Each matching in G with exactly k unmatched vertices can be extended to t^k matchings in G_t by including up to one gadget edge at each unmatched vertex.

In summary, by evaluating the polynomial $\mu(t)$ for all $t \in \{1, \ldots, n+1\}$ via gadgets and an oracle for counting matchings in G_t, we can use Lagrange interpolation to obtain m_0. This gives a polynomial-time Turing reduction from counting perfect matchings to counting matchings, transferring the #P-hardness of the former problem to the latter.

Furthermore, the above argument can also be used to derive a lower bound for counting matchings, which is however far from being tight: If the running time for counting perfect matchings on n-vertex graphs has a lower bound of $2^{\Omega(n)}$, then only a $2^{\Omega(\sqrt{n})}$ lower bound for counting matchings follows from the above argument, since the reduction incurs a quadratic blowup. This is because G_{n+1} has a gadget of size $O(n)$ at each vertex, and thus $O(n^2)$ vertices in total.

Following the same outline as above, but using more sophisticated gadgets with $O(\log^c n)$ vertices, similar reductions for various problems were obtained in [23,22,18], implying $2^{\Omega(n/\log^c n)}$ lower bounds for these problems, which are however still not tight. In particular, these reductions share the somewhat unsatisfying commonality that they "leak" hardness: Tight lower bounds for the source problems of computing specific hard coefficients in a polynomial became less tight over the course of the reduction.

1.2. The limits of interpolation

Let us say that a reduction is *gadget-interpolation-based* if it proceeds along the two steps sketched above: First encode a hard problem into the coefficients of a polynomial p, then find gadgets that can be "locally" placed at vertices or edges so as to evaluate $p(\xi)$ at sufficiently many points ξ. Finally use Lagrange interpolation to recover p from these evaluations. As remarked before, this is a well-trodden route for #P-hardness proofs. However, when embarking on this route to prove lower bounds under #ETH, we run into the following obstacles:

1. Gadget-interpolation-based reductions typically yield polynomials p of degree $n = |V(G)|$, hence require $n+1$ evaluations of p at *distinct* points, and thus in turn require $n+1$ *distinct* gadgets to be placed at vertices of G. But since there are only finitely many simple graphs on $O(1)$ vertices, the size of such gadgets must necessarily grow as some unbounded function $\alpha(n)$. Thus, any gadget-interpolation-based reduction can only yield $2^{\Omega(n/\alpha(n))}$ time lower bounds for some unbounded function $\alpha \in \omega(1)$, but such bounds are typically not tight.
2. Additionally, such reductions issue only polynomially many queries to the target problem. This is required for the setting of #P-hardness, but it is nonessential in exponential-time complexity: To obtain a lower bound of $2^{\Omega(n)}$, we might as