Journal of Computer and System Sciences 82 (2016) 1144–1160

Contents lists available at ScienceDirect

Journal of Computer and System Sciences

www.elsevier.com/locate/jcss

Approximately counting locally-optimal structures ☆

Leslie Ann Goldberg [a], Rob Gysel [b], John Lapinskas [a,*]

[a] Department of Computer Science, University of Oxford, Parks Road, OX1 3QD, UK
[b] Department of Computer Science, University of California, 2063 Kemper Hall, One Shields Avenue, Davis, CA 95616-8562, United States

ARTICLE INFO

Article history:
Received 12 May 2015
Accepted 6 April 2016
Available online 14 April 2016

Keywords:
Complexity
Approximate counting
Bipartite graphs
Maximal independent sets
Minimal separators

ABSTRACT

In general, constructing a locally-optimal structure is a little harder than constructing an arbitrary structure, but significantly easier than constructing a globally-optimal structure. A similar situation arises in listing. In counting, most problems are #P-complete, but in approximate counting we observe an interesting reversal of the pattern. Assuming that #BIS is not equivalent to #SAT under AP-reductions, we show that counting maximal independent sets in bipartite graphs is harder than counting maximum independent sets. Motivated by this, we show that various counting problems involving minimal separators are #SAT-hard to approximate. These problems have applications for constructing triangulations and phylogenetic trees.

1. Introduction

A *locally-optimal* structure is a combinatorial structure that cannot be improved by certain (greedy) local moves, even though it may not be globally optimal. An example is a maximal independent set in a graph. It is trivial to construct an independent set in a graph (for example, the singleton set containing any vertex is an independent set). It is easy to construct a maximal independent set (the greedy algorithm can do this). However, it is NP-hard to construct a globally-optimal independent set, which in this case means a maximum independent set. In the setting in which we work, this situation is typical. Constructing a locally-optimal structure is somewhat more difficult than constructing an arbitrary structure, and constructing a globally-optimal structure is more difficult than constructing a locally-optimal structure. For example, in bipartite graphs, it is trivial to construct an independent set, easy to (greedily) construct a maximal independent set, and more difficult to construct a maximum independent set (even though this can be done in polynomial time). This general phenomenon has been well-studied. In 1987, Johnson, Papadimitriou and Yannakakis [22] defined the complexity class PLS (for "polynomial-time local search") that captures local optimisation problems where one iteration of the local search algorithm takes polynomial time. As the authors point out, practically all empirical evidence leads to the conclusion that finding locally-optimal solutions is much easier than solving NP-hard problems, and this is supported by complexity-theoretic evidence, since a problem in PLS cannot be NP-hard unless NP=co-NP. An example that illustrates this point is the graph partitioning problem. For this problem it is trivial to find a valid partition, and it is NP-hard to find a globally-optimal

☆ The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007–2013) ERC grant agreement no. 334828. The paper reflects only the authors' views and not the views of the ERC or the European Commission. The European Union is not liable for any use that may be made of the information contained therein. The research leading to these results has also received funding from the National Science Foundation, grant number IIS-1219278. A preliminary version [16] of this paper appears in the proceedings of ICALP 2015.

* Corresponding author.
E-mail addresses: leslie.goldberg@cs.ox.ac.uk (L.A. Goldberg), rsgysel@ucdavis.edu (R. Gysel), john.lapinskas@cs.ox.ac.uk (J. Lapinskas).

http://dx.doi.org/10.1016/j.jcss.2016.04.001

(minimum weight) partition but Schäffer and Yannakakis [27] showed that finding a locally-optimal solution (with respect to a particular swapping-dynamics) is PLS-complete, so is presumably of intermediate complexity.

For listing combinatorial structures, a similar pattern emerges. By self-reducibility, there is a nearly-trivial polynomial-space polynomial-delay algorithm for listing the independent sets of a graph [15]. A polynomial-space polynomial-delay algorithm for listing the *maximal* independent sets exists, due to Tsukiyama et al. [31], but it is more complicated. On the other hand, there is no polynomial-space polynomial-delay algorithm for listing the *maximum* independent sets unless P=NP. There is a polynomial-space polynomial-delay algorithm for listing the maximum independent sets of a bipartite graph [23], but this is substantially more complicated than any of the previous algorithms.

When we move from constructing and listing to counting, these differences become obscured because nearly everything is #P-complete. For example, counting independent sets, maximal independent sets, and maximum independent sets of a graph are all #P-complete problems, even if the graph is bipartite [32]. Furthermore, even *approximately* counting independent sets, maximal independent sets, and maximum independent sets of a graph are all #P-complete with respect to approximation-preserving reductions [10].

The purpose of this paper is to highlight an interesting situation that arises in approximate counting where, contrary to the situations that we have just discussed, approximately counting locally-optimal structures is apparently more difficult than counting globally-optimal structures.

In order to explain the result, we first briefly summarise what is known about the complexity of approximate counting within #P. This will be explained in more detail in Section 2. There are three relevant complexity classes — the class containing problems which admit a fully-polynomial randomised approximation scheme (FPRAS), the class $\#\mathrm{RH}\Pi_1$, and #P itself. Dyer et al. [10] showed that #BIS, the problem of counting independent sets in a bipartite graph, is complete for $\#\mathrm{RH}\Pi_1$ with respect to approximation-preserving (AP) reductions and that #IS, the problem of counting independent sets in a (general) graph is #P-complete with respect to AP-reductions. It is generally believed that the $\#\mathrm{RH}\Pi_1$-complete problems are not FPRASable, but that they are of intermediate complexity, and are not as difficult to approximate as the problems which are #P-complete with respect to AP-reductions. Many problems have subsequently been shown to be $\#\mathrm{RH}\Pi_1$-complete and #P-complete with respect to AP-reductions. More examples will be given in Section 2.

We can now describe the interesting situation which emerges with respect to independent sets in bipartite graphs. Dyer et al. [10] showed that approximately counting independent sets and approximately counting *maximum* independent sets are both $\#\mathrm{RH}\Pi_1$-complete with respect to AP-reductions. Thus, the pattern outlined above would suggest that approximately counting *maximal* independent sets in bipartite graphs ought to also be $\#\mathrm{RH}\Pi_1$-complete. However, we show (Theorem 1, below) that approximately counting *maximal* independent sets in bipartite graphs is actually #P-complete with respect to AP-reductions. Thus, either $\#\mathrm{RH}\Pi_1$ and #P are equivalent in approximation complexity (contrary to the picture that has been emerging in earlier papers), or this is a scenario where approximately counting locally-optimal structures is actually more difficult than approximately counting globally-optimal ones.

Motivated by the difficulty of approximately counting maximal independent sets in bipartite graphs, we also study the problem of approximately counting other locally-optimal structures that arise in algorithmic applications. First, the problem of counting the *minimal separators* of a graph arises in diverse applications from triangulation theory to phylogeny construction in computational biology. A minimal separator is a particular type of vertex separator. Definitions are given in Section 1.1. Algorithmic applications arise because fixed-parameter-tractable algorithms are known whose running time is polynomial in the number of minimal separators of a graph. These algorithms were originally developed by Bouchitté and Todinca [5,6] (and improved in [11]) to exactly solve the so-called *treewidth* and *minimum-fill* problems. The former problem, finding the exact treewidth of a graph, is widely studied due to its applicability to a number of other NP-complete problems [4]. The technique has recently been generalized [14] to cover problems including *treecost* [2] and *treelength* [26]. The algorithm can also be used to find a minimum-width *tree-decomposition* of a graph, a key data structure that is used to solve a variety of NP-complete problems in polynomial time when the width of the tree-decomposition is fixed [4]. In recent years, much research has been dedicated to exact-exponential algorithms for treewidth [3], the fastest of which [12] has running time closely connected to the number of minimal separators in the graph. Indeed, there exist polynomials p_L and p_U such that if the graph has n vertices and M minimal separators, then the running time is at least $p_L(n)M$ and at most $p_U(n)M^2$.

Bouchitté and Todinca's approach has also recently been applied to solve the *perfect phylogeny problem* and two of its variants [21]. In this problem, the input is a set of phylogenetic characters, each of which may be viewed as a partition of a subset of *species*. The goal is to find a phylogenetic tree such that every character is *convex* on that tree — that is, the parts of each partition form connected subtrees that do not overlap. Such a tree is called a *perfect phylogeny*.

In all of these applications, it would be useful to count the minimal vertex separators of a graph, since this would give an a priori bound on the running time of the algorithms. Thus, we consider the difficulty of this problem, whose complexity was previously unresolved, even in terms of exact computation. Theorem 2 shows that the problem of counting minimal separators is #P-complete, both with respect to Turing reductions (for exact computation) and with respect to AP-reductions. Thus, this problem is as difficult to approximate as any problem in #P.

Motivated by applications to treewidth [11] and phylogeny [20,21], we also consider various heuristic approximations to the minimal separator problem. The number of inclusion-minimal separators is a natural choice for a lower bound on the number of minimal separators. Conversely, the number of (s,t)-minimal separators, taken over all vertices s and t, is a natural choice for an upper bound on the number of minimal separators. Theorem 2 shows that both of these bounds