Theoretical Computer Science 634 (2016) 35–46

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

The complexity of counting locally maximal satisfying assignments of Boolean CSPs ☆

Leslie Ann Goldberg [a], Mark Jerrum [b,*]

[a] Department of Computer Science, University of Oxford, UK
[b] School of Mathematical Sciences, Queen Mary, University of London, UK

ARTICLE INFO

Article history:
Received 21 September 2015
Received in revised form 9 March 2016
Accepted 5 April 2016
Available online 11 April 2016
Communicated by V.Th. Paschos

Keywords:
Constraint satisfaction problem
Computational complexity of counting problems
Approximate computation

ABSTRACT

We investigate the computational complexity of the problem of counting the locally maximal satisfying assignments of a Constraint Satisfaction Problem (CSP) over the Boolean domain $\{0, 1\}$. A satisfying assignment is *locally maximal* if any new assignment which is obtained from it by changing a 0 to a 1 is unsatisfying. For each constraint language Γ, #LocalMaxCSP(Γ) denotes the problem of counting the locally maximal satisfying assignments, given an input CSP with constraints in Γ. We give a complexity dichotomy for the problem of *exactly* counting the locally maximal satisfying assignments and a complexity trichotomy for the problem of *approximately* counting them. Relative to the problem #CSP(Γ), which is the problem of counting *all* satisfying assignments, the locally maximal version can sometimes be easier but never harder. This finding contrasts with the recent discovery that approximately counting locally maximal independent sets in a bipartite graph is harder (under the usual complexity-theoretic assumptions) than counting all independent sets.

1. Introduction

A Boolean Constraint Satisfaction Problem (CSP) is a generalised satisfiability problem. An instance of a Boolean CSP is a set of variables together with a collection of constraints that enforce certain relationships between the variables. These constraints are chosen from an agreed finite set ("language") Γ of relations of various arities on the Boolean domain $\{0, 1\}$. The study of the computational complexity of Boolean CSPs has a long history, starting with Schaefer, who described the complexity of the basic decision problem: is a given Boolean CSP instance satisfiable? The computational complexity of the satisfiability problem depends, of course, on the constraint language Γ, becoming potentially harder as Γ becomes larger and more expressive. Schaefer showed [15] that, depending on Γ, the satisfiability problem is either polynomial-time solvable or NP-complete, and he provided a precise characterisation of this dichotomy.

The problem #CSP(Γ) is the problem of determining the number of satisfying assignments of a CSP instance with constraint language Γ. A dichotomy for this counting problem was given by Creignou and Hermann [1].

☆ 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.

* Corresponding author.
E-mail address: m.jerrum@qmul.ac.uk (M. Jerrum).

http://dx.doi.org/10.1016/j.tcs.2016.04.008
0304-3975/

Theorem 1. *(See Creignou, Hermann [1].) Let Γ be a constraint language with domain $\{0,1\}$. The problem $\#\mathsf{CSP}(\Gamma)$ is in FP if every relation in Γ is affine. Otherwise, $\#\mathsf{CSP}(\Gamma)$ is $\#\mathsf{P}$-complete.*

A relation is *affine* if it is expressible as the set of solutions to a system of linear equations over the two-element field $\mathbb{F}_2$. The constraint language Γ is said to be affine if and only if every constraint in Γ is affine. Thus, Theorem 1 shows (assuming $\mathsf{FP} \neq \#\mathsf{P}$) that $\#\mathsf{CSP}(\Gamma)$ is tractable if and only if the set of satisfying assignments can be expressed as the set of solutions to a system of linear equations.

Since most constraint languages Γ lead to intractable counting problems, it is natural to consider the complexity of *approximately* counting the satisfying assignments of a CSP. Dyer, Goldberg, Greenhill and Jerrum [7] used approximation-preserving reductions (AP-reductions) between counting problems to explore the complexity of approximately computing solutions. They identified three equivalence classes of interreducible counting problems within $\#\mathsf{P}$: (i) problems that have a polynomial-time approximation algorithm or "FPRAS", (ii) problems that are equivalent to $\#\mathsf{BIS}$ under AP-reductions, and (iii) problems that are equivalent to $\#\mathsf{SAT}$ under AP-reductions. Here, $\#\mathsf{BIS}$ is the problem of counting independent sets in a bipartite graph and $\#\mathsf{SAT}$ is the problem of counting the satisfying assignments of a Boolean formula in CNF. Dyer, Goldberg and Jerrum [8] show that all Boolean counting CSPs can be classified using these classes.

Theorem 2. *(See [8, Theorem 3].) Let Γ be a constraint language with domain $\{0,1\}$. If every relation in Γ is affine then $\#\mathsf{CSP}(\Gamma)$ is in FP. Otherwise if every relation in Γ is in IM_2 then $\#\mathsf{CSP}(\Gamma) \equiv_{\mathsf{AP}} \#\mathsf{BIS}$. Otherwise $\#\mathsf{CSP}(\Gamma) \equiv_{\mathsf{AP}} \#\mathsf{SAT}$.*

In the statement of Theorem 2, $\equiv_{\mathsf{AP}}$ is the equivalence relation "interreducible via approximation-preserving reductions". In order to define IM_2, we must first define the binary implication relation $\mathsf{Implies} = \{(0,0),(0,1),(1,1)\}$. Then IM_2 is the set of relations that can be expressed using conjunctions of these implications, together with unary constraints. The precise definition of IM_2 is given in Section 2, along with precise definitions of the other concepts that appear in this introduction.

There are many other questions that one can ask about Boolean CSPs aside from deciding satisfiability and counting the satisfying assignments. Here, we study the complexity of counting and approximately counting the number of locally maximal satisfying assignments of a CSP instance. A satisfying assignment is *locally maximal* if any new assignment which is obtained from it by changing a single 0 to a 1 is unsatisfying. So local maximality is with respect to the set of 1's in the satisfying assignment. Also, it is with respect to local changes — changing a single 0 to a 1. Other notions of maximality are discussed in Section 4.

Goldberg, Gysel and Lapinskas [9] show (assuming that $\#\mathsf{BIS}$ is not equivalent to $\#\mathsf{SAT}$ under AP reductions) that counting locally maximal structures can be harder than counting all structures. In particular, Theorem 1 of [9] shows that counting the locally maximal independent sets in a bipartite graph is equivalent to $\#\mathsf{SAT}$ under AP-reductions. Obviously, counting *all* independent sets in a bipartite graph is exactly the problem $\#\mathsf{BIS}$, which is presumed to be easier. Thus, Goldberg, Gysel and Lapinskas have found an example of a (restricted) Boolean counting CSP (namely, $\#\mathsf{BIS}$) where approximately counting locally maximal satisfying assignments is apparently harder than approximately counting all satisfying assignments. However, $\#\mathsf{BIS}$ isn't exactly a Boolean counting CSP — rather, it is a Boolean counting CSP with a restriction (bipartiteness) on the problem instance. This prompted us to investigate the complexity of approximating the number of satisfying assignments of unrestricted Boolean CSPs. In particular, we study $\#\mathsf{LocalMaxCSP}(\Gamma)$, the problem of counting locally maximal satisfying assignments of an instance of a Boolean CSP with constraint language Γ.

Given the phenomenon displayed by $\#\mathsf{BIS}$, one might expect to find constraint languages Γ that exhibit a jump upwards in computational complexity when passing from $\#\mathsf{CSP}(\Gamma)$ to $\#\mathsf{LocalMaxCSP}(\Gamma)$, but we determine that this does not in fact occur. The reverse may occur: counting locally maximal vertex covers in a graph is trivial (there is just one), but counting all vertex covers is equivalent under AP-reducibility to $\#\mathsf{SAT}$. It turns out that this trivial phenomenon, which occurs when the property in question is monotone increasing, is essentially the only difference between $\#\mathsf{CSP}(\Gamma)$ and $\#\mathsf{LocalMaxCSP}(\Gamma)$.

Our first result (Theorem 3) presents a dichotomy for the complexity of exactly solving $\#\mathsf{LocalMaxCSP}(\Gamma)$ for all Boolean constraint languages Γ. In most cases, $\#\mathsf{LocalMaxCSP}(\Gamma)$ is equivalent in complexity to $\#\mathsf{CSP}(\Gamma)$. However, if Γ is *essentially monotone* (the proper generalisation of the vertex cover property) then $\#\mathsf{LocalMaxCSP}(\Gamma)$ is in FP. Our second result (Theorem 4) presents a trichotomy for the complexity of *approximately* solving $\#\mathsf{LocalMaxCSP}(\Gamma)$. Once again, in most cases, $\#\mathsf{LocalMaxCSP}(\Gamma)$ is equivalent with respect to AP-reductions to $\#\mathsf{CSP}(\Gamma)$. The only exceptional case is the one that we have already seen — if Γ is *essentially monotone* then $\#\mathsf{LocalMaxCSP}(\Gamma)$ is in FP.

The result leaves us with a paradox. There is a very direct reduction from $\#\mathsf{BIS}$ to $\#\mathsf{CSP}(\{\mathsf{Implies}\})$ that is "parsimonious", i.e., preserves the number of solutions. So $\#\mathsf{BIS}$ is AP-reducible to $\#\mathsf{CSP}(\{\mathsf{Implies}\})$ (which can also be seen from Theorem 2). How can it be, then, that the complexity of approximately counting locally maximal independent sets in bipartite graphs jumps upwards from the complexity of approximately counting all independent sets, whereas Theorem 4 tells us that $\#\mathsf{LocalMaxCSP}(\{\mathsf{Implies}\})$ remains AP-equivalent to $\#\mathsf{CSP}(\{\mathsf{Implies}\})$?

The resolution of the paradox is as follows. Suppose $G = (U, V, E)$ is an instance of $\#\mathsf{BIS}$, i.e., a graph with bipartition $U \cup V$ and edge set $E \subseteq U \times V$. The parsimonious reduction from $\#\mathsf{BIS}$ to $\#\mathsf{CSP}(\{\mathsf{Implies}\})$ simply interprets the vertices $U \cup V$ of instance G as Boolean variables, and each edge $(u, v) \in E$ as a constraint $\mathsf{Implies}(u, v)$. Then there is an obvious bijection between independent sets in G and satisfying assignments of the constructed instance of $\#\mathsf{CSP}(\{\mathsf{Implies}\})$, but it involves interpreting 0 and 1 in different ways on the opposite sides of the bipartition: on U, 1 means "in the independent set" while on V, 1 means "out of the independent set". Of course, local maximality is not preserved by this change in