Journal of Computer and System Sciences 80 (2014) 796–805

Contents lists available at ScienceDirect

Journal of Computer and System Sciences

www.elsevier.com/locate/jcss

Finding the smallest binarization of a CFG is NP-hard

Carlos Gómez-Rodríguez [*]

Departamento de Computación, Universidade da Coruña, Spain

ARTICLE INFO

Article history:
Received 10 June 2012
Received in revised form 25 September 2013
Accepted 19 December 2013
Available online 2 January 2014

Keywords:
Natural language processing
Complexity
Binarization
Grammars
Context-free grammar
Parsing

ABSTRACT

Grammar binarization is the process and result of transforming a grammar to an equivalent form whose rules contain at most two symbols in their right-hand side. Binarization is used, explicitly or implicity, by a wide range of parsers for context-free grammars and other grammatical formalisms. Non-trivial grammars can be binarized in multiple ways, but in order to optimize the parser's computational cost, it is convenient to choose a binarization that is as small as possible. While several authors have explored heuristics to obtain compact binarizations, none of them guarantee that the resulting grammar has minimum size. However, to our knowledge, no hardness results for this problem have been published. In this article, we address this issue and prove that the problem of finding a minimum binarization of a given context-free grammar is NP-hard, by reduction from vertex cover. We also provide a lower bound on the approximability of this problem.

1. Introduction

The process of grammar binarization, by which a grammar is transformed into an equivalent binary form by splitting its larger rules into sets of binary rules, is essential to perform efficient natural language parsing with context-free grammars (CFGs) and other grammatical formalisms.

The most widely known and discussed application of binarization is probably the preprocessing step for the CKY parsing algorithm for CFGs [1,2], which needs to use a binary grammar in order to achieve $O(n^3)$ time complexity.[1] However, most if not all the well-known CFG parsing algorithms use binarization in one way or another: although algorithms such as the Earley [5], Left-Corner [6], Graham–Harrison–Ruzzo [7] or Head-Corner [8] parsers do not perform it explicitly as a preprocessing step, they binarize rules implicitly using dotted items. In fact, some of these parsers (such as the Left-Corner [6] and Head-Corner [8] parsers) can be seen as implementations of the same underlying parser with different binarization strategies.

Binarization is also used to achieve efficient parsing for more powerful grammatical formalisms, such as conjunctive grammars [9], synchronous context-free grammars [10], linear context-free rewriting systems [11], coupled context-free grammars [12] or regular tree grammars [13]. While in this article we will only deal with CFGs, the hardness result presented in this article is applicable to all these formalisms as they contain CFGs as a particular case. Note that, in the case of synchronous CFGs and linear context-free rewriting systems with bounded fan-out, this is arguably not very useful because these formalisms do not guarantee that a binarization of any size even *exists* for a given grammar [10,14]. However, it can

[*] Correspondence to: Facultade de Informática, Campus de Elviña, s/n, 15071, A Coruña, Spain. Fax: +34 981 167160.
E-mail address: cgomezr@udc.es.
URL: http://www.grupolys.org/~cgomezr.

[1] While CKY is traditionally described for grammars in Chomsky Normal Form [3], which impose some extra conditions apart from being binary (e.g. disallowing rules with a single nonterminal in the right-hand side), it is straightforward to define it in a more general way that can work efficiently with any binary grammar [4].

0022-0000/$ – see front matter © 2013 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.jcss.2013.12.003

still be applied to subclasses of these formalisms that do guarantee this property, such as well-nested linear context-free rewriting systems [15] or binarizable synchronous CFGs [10].

A CFG can be binarized in different ways, depending on which pairs of symbols are chosen to be grouped to form binary rules. In particular, the number of possible binarizations for a single $(n+1)$-ary CFG production rule is the nth Catalan number, $C_n = \frac{1}{n+1}\binom{2n}{n}$ [16]. This naturally raises the question of which binarization should be used to obtain the best parsing efficiency. Since grammar size is an important factor in the efficiency of natural language parsers [17], an obvious approach is to binarize in a way that tries to collapse pairs of nonterminal symbols that are frequent in grammar rules, so as to obtain a binary grammar as compact as possible [18]. Song et al. [16] provide empirical evidence that using compact grammars greatly improves the efficiency of a CKY parser with respect to naive binarization, and gain further improvements by using corpus statistics to predict the amount of useless items (items not leading to a complete parse) that will be generated by different binarizations.

While several authors have recently tackled the problem of searching for small binarizations for a given grammar [18,16,19]; no efficient algorithm has been described that will solve the problem of finding the smallest size binarization for a given CFG (the *minimum binarization problem*). Instead, the mentioned papers use greedy algorithms to find a reasonably small binarization, without guarantee of optimality. An experiment by DeNero et al. [19] for transducer grammars showed this strategy to output a grammar more than seven times smaller than naive right-branching binarization, but still more than twice the size of the optimal grammar obtained using an integer linear programming solver (which, reportedly, could only find optimal solutions for very small grammars).

To the best of our knowledge, in spite of the absence of an efficient algorithm to compute the optimal binarization for a given CFG; and even though growing use of mildly context-sensitive formalisms such as synchronous context-free grammars, linear context-free rewriting systems or range concatenation grammars has led to a surge of interest in theoretical results about binarization in the last few years [11,10,15,20]; no hardness results are known for the minimum binarization problem.

In this article, we fill that gap in the literature by proving that the minimum binarization problem is NP-hard. We do so by reduction from the vertex cover problem, inspired by the proof for the (related, but not equivalent) smallest grammar problem by Charikar et al. [21]. We also provide an inapproximability result, showing that it is NP-hard to approximate the minimum binarization problem within any factor smaller than 2575/2574.

Related work As already mentioned, the problem of finding a binarization as small as possible for a given grammar has been considered in several recent papers in the computational linguistics literature [18,16,19]; but we know of no published hardness results for the minimum binarization problem.

On the other hand, the related problem of finding the minimum (not necessarily binary) context-free grammar that defines a given single-string language, which is relevant in data compression [22], is known to be NP-hard. While this problem (called the "smallest grammar problem" [21]) is not equivalent to the minimum binarization problem, they do have obvious similarities, and the hardness proof for the minimum binarization problem in Section 3 has been inspired by the proof for the smallest grammar problem given by Charikar et al. [21].

In particular, the main differences between the two problems are the following:

1. The smallest grammar problem is concerned with a language with a single string (or, equivalently, it starts from a grammar with a single rule), whereas the minimum binarization problem can have any context-free grammar as input.
2. In relation to the previous point, the smallest grammar problem is not limited to transformations that preserve the structure of parse trees: it only requires so-called *weak equivalence*, i.e., that the input and output grammars generate the same set of strings. On the other hand, as will be explained in detail in Section 2, the minimum binarization problem requires that the output grammar be a factorization of the input grammar, i.e., a transformation of the input obtained by splitting its rules into smaller rules. This is so that the parse trees under the original grammar can be recovered from parses obtained with the binarized one, and the string and tree probabilities under the original grammar carry over to the binarized one if a probabilistic model is used.
3. In the minimum binarization problem, size is minimized among binary grammars only, while in the smallest grammar problem any context-free grammar is allowed. Note that this difference is relevant because the smallest binary CFG for a given string or language may be different to (and larger than) the smallest unrestricted CFG for that string: for example, we can generate the language $\{abc\}$ with a non-binary grammar of size 3 (the one with a single rule $S \rightarrow abc$, with three symbols on its right-hand side) but the smallest binary grammars for that same language have size 4 (one of them would have the rules $S \rightarrow Ac$, $A \rightarrow ab$, for a total of four symbols in their right-hand sides).

While the first two differences would not prevent us from carrying over the core of the hardness proof from the smallest grammar problem to the minimum binarization problem, the third one does, since the proof by Charikar et al. [21] relies on the construction of a grammar whose optimized version has non-binary rules. It is also not obvious how to attempt a proof by reduction of the smallest grammar problem to the minimum binarization problem, since we do not know whether the smallest binary grammar need always be a binarization of the smallest grammar, and even supposing that this proposition holds, we would still need to find a polynomial way of finding the optimal set of pairs of nonterminals to collapse (or of pairs of binary rules to merge into larger rules) to obtain the smallest grammar from the smallest binarization.

Thus, rather than attempting such a reduction, we will prove that the minimum binarization problem is NP-hard by using a new proof by reduction from vertex cover, inspired on the one in [21] (which was, in turn, inspired by arguments