Synchronous context-free grammars and optimal linear parsing strategies

Journal of Computer and System Sciences 81 (2015) 1333–1356

Contents lists available at ScienceDirect

Journal of Computer and System Sciences

www.elsevier.com/locate/jcss

Synchronous context-free grammars and optimal linear parsing strategies

Pierluigi Crescenzi [a], Daniel Gildea [b,*], Andrea Marino [c], Gianluca Rossi [d], Giorgio Satta [e]

[a] Dipartimento di Sistemi e Informatica, Università di Firenze, Viale Morgagni, 65, 50134 Firenze, Italy
[b] Computer Science Department, University of Rochester, Rochester, NY 14627, United States
[c] Dipartimento di Informatica, Università di Milano, Via Festa del Perdono 7, 20122 Milano, Italy
[d] Dipartimento di Matematica, Università di Roma Tor Vergata, Via Ricerca Scientifica 1, 00133 Roma, Italy
[e] Dipartimento di Ingegneria dell'Informazione, Università di Padova, Via Gradenigo 6/A, 35131 Padova, Italy

ARTICLE INFO

Article history:
Received 25 November 2013
Received in revised form 12 February 2015
Accepted 12 February 2015
Available online 15 April 2015

Keywords:
Formal languages
Parallel rewriting systems
Parsing

ABSTRACT

Synchronous Context-Free Grammars (SCFGs), also known as syntax-directed translation schemata [1,2], are unlike context-free grammars in that they do not have a binary normal form. In general, parsing with SCFGs takes space and time polynomial in the length of the input strings, but with the degree of the polynomial depending on the permutations of the SCFG rules. We consider linear parsing strategies, which add one nonterminal at a time. We show that for a given input permutation, the problems of finding the linear parsing strategy with the minimum space and time complexity are both NP-hard.

1. Introduction

Synchronous Context-Free Grammars (SCFGs) are widely used to model translational equivalence between strings in both the area of compilers for programming languages and, more recently, in the area of machine translation of natural languages. The formalism was first introduced by Lewis and Stearns [3] under the name of syntax-directed transduction grammars, and was later called syntax-directed translation schemata by Aho and Ullman [1,2]. The name SCFG, which we use in this article, was later introduced in the literature on computational linguistics, where the term "synchronous" refers to rewriting systems that generate strings in both a source and target language simultaneously [4–6]. In fact, SCFGs can be seen as a natural extension of the well-known rewriting formalism of Context-Free Grammars (CFGs). More precisely, while a CFG generates a set of strings, an SCFG generates a set of string pairs using essentially the same context-free rewriting mechanism, along with some special synchronization between the two derivations, as discussed below.

An SCFG is a string rewriting system based on synchronous rules. Informally, a **synchronous rule** is composed of two CFG rules along with a bijective pairing between all the occurrences of the nonterminal symbols in the right-hand side of the first rule and all the occurrences of the nonterminal symbols in the right-hand side of the second rule. There is no restriction on the terminal symbols appearing in the right-hand sides of the two CFG rules. Two nonterminal occurrences that are associated by the above bijection are called **linked** nonterminals. Linked nonterminals are not necessarily occurrences of the same nonterminal symbol. In what follows, we will often view a synchronous rule as a permutation of the nonterminal

* Corresponding author. Fax: +1 585 273 4556.
E-mail address: gildea@cs.rochester.edu (D. Gildea).

http://dx.doi.org/10.1016/j.jcss.2015.04.003

occurrences in the two right-hand sides, combined with some renaming of these occurrences and with some insertion and deletion of terminal symbols.

In an SCFG rewriting is restricted in two ways: the two CFG rule components in a synchronous rule must be applied simultaneously, and rewriting must take place at linked nonterminals. Other than that, the application of a synchronous rule is independent of the context, similarly to the CFG case. As a result, an SCFG generates a pair of strings by means of two context-free parse trees that have the same skeleton but differ by some reordering and renaming of the nonterminal children at each internal node, and by the insertion and the deletion of the terminal children of that node. Moreover, the projection of the generated string pairs on both dimensions are still context-free languages. Thus, the added generative power of an SCFG lies in its ability to model long-distance movement of phrase constituents in the translation from the source to the target language, through simple permutations implemented at the internal nodes in the generated trees, something that is not possible with models based on finite-state transducers.

Recently, SCFGs have received wide attention in the area of natural language processing, where several variants of SCFGs augmented with probabilities are currently used for translation between natural languages. This is due to the recent surge of interest in commercial systems for statistical machine translation working on large scale, real-world applications such as the translation of text documents from the world-wide web. However, from a theoretical perspective our knowledge of the parsing problem based on SCFGs and of several related tasks is quite limited, with many questions still left unanswered, as discussed below. This is rather surprising, in view of the fact that SCFGs are a very natural extension of the class of CFGs, for which the parsing problem has been extensively investigated and is well understood nowadays.

In the context of statistical machine translation, SCFGs are automatically induced from parallel corpora, that is, very large collections of source texts that come with target translations, and are usually enriched with annotations aligning source and target words [7,8]. Alternative translation models are currently in use in machine translation, such as word-to-word translation models [9] or phrase-based translation models [10], which are essentially finite-state models. However, it has been experimentally shown that the more powerful generative capacity of SCFGs achieves better accuracy than finite-state models in real-world machine translation applications [7].

The **recognition** (or membership) problem for SCFGs is defined as follows. Given as input an SCFG $\mathcal{G}$ and strings w_1 and w_2, we have to decide whether the pair w_1, w_2 can be generated by $\mathcal{G}$. The **parsing** problem for SCFGs (or synchronous parsing problem) is defined for the same input $\mathcal{G}$, w_1 and w_2, and produces as output some suitable representation of the set of all parse trees in $\mathcal{G}$ that generate w_1 and w_2. Finally, the **decoding** (or translation) problem for SCFGs requires as input an SCFG $\mathcal{G}$ and a single string w_1, and produces as output some suitable representation of the set of all parse trees in $\mathcal{G}$ that generate pairs of the form w_1, w_2, for some string w_2. In this paper we investigate the synchronous parsing problem, which is strictly related to the other two problems, as will be discussed in more detail in Section 5.

From the perspective of synchronous parsing, a crucial difference between CFGs and SCFGs is that SCFGs cannot always be **binarized**, that is, cast into a normal form with no more than two nonterminals on the right-hand side of each rule component in a synchronous rule. In fact, SCFGs form an infinite hierarchy, where grammars with at most r nonterminals on the right-hand side of a rule can generate sets of string pairs not achievable by grammars with the same quantity bounded by $r-1$, for each $r > 3$ [2]. Binarization is crucial to standard algorithms for CFG parsing, whether explicitly as a preprocessing transformation on the grammar, as for instance in the case of the Cocke–Kasami–Younger algorithm [11], or implicitly through the use of dotted rule symbols indicating which nonterminals have already been parsed, as in the case of Earley algorithm [12]. Unfortunately these techniques cannot be applied to SCFGs, because of the above restrictions on binarization, and the parsing problem for SCFGs seems significantly more complex than the parsing problem for CFGs, from a computational perspective. While parsing for CFGs can be solved in polynomial space and time by the above mentioned algorithms, it has been shown that parsing for SCFGs is NP-hard, when the grammar is considered as part of the input [13].

Despite this hardness result, when the SCFG is fixed, parsing can be performed in time polynomial in the length of the input strings using the bottom–up dynamic programming framework described in Section 2. The degree of the polynomial is determined by the maximum complexity of any rule in the grammar, because rules are parsed independently of one another. More precisely, the complexity of a given rule is $\mathcal{O}(n^{d(\pi,\sigma)})$, where n is the sentence length, and d is some function of π, the permutation associated with the rule, and σ, a parsing **strategy** for the rule. This leads us to consider the problem of finding the best strategy for a given rule, that is, finding the σ that minimizes $d(\pi,\sigma)$. We investigate the problem of finding the best **linear** parsing strategy for a given synchronous rule, that is, the way of collecting one after the other the linked nonterminals in a synchronous rule that results in the optimization of the space and time complexity for synchronous parsing. We show that this task is NP-hard. This solves an open problem that has been addressed in several previously published works; see for instance [14,15], and [16].

Relation with previous work The problem that we explore in this article is an instance of a **grammar factorization** problem. Factorization is the general method of breaking a grammar rule into a number of equivalent, smaller rules, usually for the purpose of finding efficient parsing algorithms. Our linear parsing strategies for SCFGs process two linked nonterminals from the original rule at each step. Each of these steps is equivalent to applying a binary rule in another rewriting system, which must be more general than SCFGs, since after all SCFGs cannot be binarized.

The general problem of grammar factorization has received a great deal of study recently in the field of computational linguistics, with the rise of statistical systems for natural language translation, as well as systems for parsing with monolingual grammars that are more powerful than CFGs [15,17–19]. Most work in this area addresses some subclasses of the very general rewriting framework known as Linear Context-Free Rewriting Systems (LCFRS) [20], which is equivalent to Multiple Context-Free Grammars [21], and which subsumes SCFGs and many other formalisms. Many algorithms have been proposed for efficiently factorizing subclasses of LCFRS, in order to optimize parsing under various criteria. Our result in this article is a hardness result, showing that such algorithms cannot extend to the widely used and theoretically important class of SCFGs. A related result has been presented by Crescenzi et al. [16], showing that optimal linear factorization for general LCFRS is NP-hard. Their reduction involves constructing LCFRS rules that are not valid as SCFG rules. Indeed, as already mentioned, SCFG rules can be viewed as permutations, and the special structure of these objects makes reductions less straightforward than in the case of LCFRS. This article therefore strengthens the result in [16], showing that even if we restrict ourselves to SCFGs, detection of optimal linear parsing strategies is still NP-hard.

2. Preliminaries

In this section we formally introduce the class of synchronous context-free grammars, along with the computational problem that we investigate in this article. We assume the reader to be familiar with basic definitions from formal language theory, and we only briefly summarize here the adopted notation.

For any positive integer n, we write $[n]$ to denote the set $\{1, \dots, n\}$, and for $n = 0$ we let $[n]$ be the empty set. We also write $[n]_0$ to denote the set $[n] \cup \{0\}$.

2.1. Synchronous context-free grammars

Let Σ be some finite alphabet. A string x over Σ is a finite ordered sequence of symbols from Σ. The length of x is written $|x|$; the empty string is denoted by ε, and we have $|\varepsilon| = 0$. We write Σ^* for the set of all strings over Σ, and $\Sigma^+ = \Sigma^* \setminus \{\varepsilon\}$. For strings $x, y \in \Sigma^*$, $x \cdot y$ denotes the concatenation of x and y, which we abbreviate as xy.

A **context-free grammar** (CFG for short) is a tuple $\mathcal{G} = (V_N, V_T, P, S)$, where V_N is a finite set of nonterminals, V_T is a finite set of terminals with $V_T \cap V_N = \emptyset$, $S \in V_N$ is a special symbol called the start symbol, and P is a finite set of rules having the form $A \to \alpha$, with $A \in V_N$ and $\alpha \in (V_T \cup V_N)^*$. The size of a CFG $\mathcal{G}$ is defined as $|\mathcal{G}| = \sum_{(A\to\gamma)\in P} |A\gamma|$.

The **derive** relation associated with a CFG $\mathcal{G}$ is written $\Rightarrow_{\mathcal{G}}$; we also use the reflexive and transitive closure of $\Rightarrow_{\mathcal{G}}$, written $\Rightarrow^*_{\mathcal{G}}$. The language (set of strings) derived in $\mathcal{G}$ is defined as $L(\mathcal{G}) = \{w \mid S \Rightarrow^*_{\mathcal{G}} w,\ w \in V_T^*\}$.

In what follows, we need to represent bijections between the occurrences of nonterminals in two strings over $V_N \cup V_T$. This can be realized by annotating nonterminals with **indices** from an infinite set. In this article, we draw indices from the set of positive natural numbers $\mathbb{N}$. We define $\mathcal{I}(V_N) = \{A^{\boxed{t}} \mid A \in V_N,\ t \in \mathbb{N}\}$ and $V_I = \mathcal{I}(V_N) \cup V_T$. For a string $\gamma \in V_I^*$, we write $\mathsf{index}(\gamma)$ to denote the set of all indices that appear in symbols in γ.

Two strings $\gamma_1, \gamma_2 \in V_I^*$ are **synchronous** if each index from $\mathbb{N}$ occurs at most once in γ_1 and at most once in γ_2, and $\mathsf{index}(\gamma_1) = \mathsf{index}(\gamma_2)$. Therefore γ_1, γ_2 have the general form

$$\gamma_1 = u_{1,0} A_{1,1}^{\boxed{t_1}} u_{1,1} A_{1,2}^{\boxed{t_2}} u_{1,2} \cdots u_{1,r-1} A_{1,r}^{\boxed{t_r}} u_{1,r}$$

$$\gamma_2 = u_{2,0} A_{2,1}^{\boxed{t_{\pi(1)}}} u_{2,1} A_{2,2}^{\boxed{t_{\pi(2)}}} u_{2,2} \cdots u_{2,r-1} A_{2,r}^{\boxed{t_{\pi(r)}}} u_{2,r}$$

where $r \geq 0$, $u_{1,i}, u_{2,i} \in V_T^*$ for each $i \in [r]_0$, $A_{1,i}^{\boxed{t_i}}, A_{2,i}^{\boxed{t_{\pi(i)}}} \in \mathcal{I}(V_N)$ for each $i \in [r]$, $t_i \neq t_j$ for $i \neq j$, and π is a permutation of the set $[r]$. Note that, under the above convention, nonterminals $A_{1,i}, A_{2,\pi^{-1}(i)}$ appear with the same index t_i, for each $i \in [r]$. In a pair of synchronous strings, two nonterminal occurrences with the same index are called **linked** nonterminals.

A **synchronous context-free grammar** (SCFG) is a tuple $\mathcal{G} = (V_N, V_T, P, S)$,[1] where V_N, V_T and S are defined as for CFGs, and P is a finite set of **synchronous rules**. Each synchronous rule has the form $[A_1 \to \alpha_1,\ A_2 \to \alpha_2]$, where $A_1, A_2 \in V_N$ and where $\alpha_1, \alpha_2 \in V_I^*$ are synchronous strings. We refer to $A_1 \to \alpha_1$ and $A_2 \to \alpha_2$, respectively, as the left and right **components** of the synchronous rule. Note that if we ignore the indices annotating the nonterminals in α_1 and α_2, then $A_1 \to \alpha_1$ and $A_2 \to \alpha_2$ are context-free rules.

Example 1. The list of synchronous rules reported below implicitly defines an SCFG. Symbols s_i are rule labels, to be used as references in later examples.

$$s_1 : [S \to A^{\boxed{1}}\ B^{\boxed{2}},\ S \to B^{\boxed{2}}\ A^{\boxed{1}}]$$

$$s_2 : [A \to aA^{\boxed{1}}b,\ A \to bA^{\boxed{1}}a]$$

$$s_3 : [A \to ab,\ A \to ba]$$

$$s_4 : [B \to cB^{\boxed{1}}d,\ B \to dB^{\boxed{1}}c]$$

$$s_5 : [B \to cd,\ B \to dc]$$

[1] We overload symbol $\mathcal{G}$. It will always be clear from the context whether $\mathcal{G}$ refers to a CFG or to an SCFG.

We now define the notion of derivation associated with an SCFG. In a derivation, we rewrite a pair of synchronous strings, always producing a new pair of synchronous strings. This is done in several steps, where, at each step, two linked nonterminals are rewritten by a synchronous rule. We use below the auxiliary notion of **reindexing**, which is an injective function f from $\mathbb{N}$ to $\mathbb{N}$. We extend f to V_I by letting $f(A^{\boxed{t}}) = A^{\boxed{f(t)}}$ for $A^{\boxed{t}} \in \mathcal{I}(V_N)$ and $f(a) = a$ for $a \in V_T$. We also extend f to strings in V_I^* by letting $f(\varepsilon) = \varepsilon$ and $f(X\gamma) = f(X)f(\gamma)$, for each $X \in V_I$ and $\gamma \in V_I^*$.

Let $\gamma_1, \gamma_2 \in V_I^*$ be two synchronous strings. The **derive** relation $[\gamma_1,\ \gamma_2] \Rightarrow_{\mathcal{G}} [\delta_1,\ \delta_2]$ holds whenever there exist an index t in $\mathsf{index}(\gamma_1) = \mathsf{index}(\gamma_2)$, a synchronous rule $s \in P$ of the form $[A_1 \to \alpha_1,\ A_2 \to \alpha_2]$, and some reindexing f such that

(i) $\mathsf{index}(f(\alpha_1)) \cap (\mathsf{index}(\gamma_1) \setminus \{t\}) = \emptyset$;
(ii) $\gamma_1 = \gamma_1' A_1^{\boxed{t}} \gamma_1''$, $\gamma_2 = \gamma_2' A_2^{\boxed{t}} \gamma_2''$; and
(iii) $\delta_1 = \gamma_1' f(\alpha_1) \gamma_1''$, $\delta_2 = \gamma_2' f(\alpha_2) \gamma_2''$.

We also write $[\gamma_1,\ \gamma_2] \Rightarrow_{\mathcal{G}}^{s} [\delta_1,\ \delta_2]$ to explicitly indicate that the derive relation holds through rule s.

Note that δ_1, δ_2 above are guaranteed to be synchronous strings, because α_1 and α_2 are synchronous strings and because of condition (i) above. Note also that, for a given pair $[\gamma_1,\ \gamma_2]$ of synchronous strings, an index t and a synchronous rule as above, there may be infinitely many choices of a reindexing f such that the above constraints are satisfied. However, all essential results about SCFGs are independent of the specific choice of reindexing, and therefore we will not further discuss this issue here.

A derivation in $\mathcal{G}$ is a sequence $\sigma = s_1 s_2 \cdots s_d$, $d \geq 0$, of synchronous rules $s_i \in P$, $i \in [d]$, with $\sigma = \varepsilon$ for $d = 0$, satisfying the following property. For some pairs of synchronous strings $[\gamma_{1,i},\ \gamma_{2,i}]$, $i \in [d]_0$, we have $[\gamma_{1,i-1},\ \gamma_{2,i-1}] \Rightarrow_{\mathcal{G}}^{s_i} [\gamma_{1,i},\ \gamma_{2,i}]$ for each $i \in [d]$. We always implicitly assume some canonical form for derivations in $\mathcal{G}$, by demanding for instance that each step rewrites a pair of linked nonterminal occurrences of which the first is leftmost in the left component. When we want to focus on the specific synchronous strings being derived, we also write derivations in the form $[\gamma_{1,0},\ \gamma_{2,0}] \Rightarrow_{\mathcal{G}}^{\sigma} [\gamma_{1,d},\ \gamma_{2,d}]$, and we write $[\gamma_{1,0},\ \gamma_{2,0}] \Rightarrow_{\mathcal{G}}^{*} [\gamma_{1,d},\ \gamma_{2,d}]$ when σ is not further specified. The **translation** generated by an SCFG $\mathcal{G}$ is defined as

$$T(\mathcal{G}) = \{[w_1,\ w_2] \mid [S^{\boxed{1}},\ S^{\boxed{1}}] \Rightarrow_{\mathcal{G}}^{*} [w_1,\ w_2],\ w_1, w_2 \in V_T^*\}.$$

Example 2. Consider the SCFG $\mathcal{G}$ from Example 1. The following is a (canonical) derivation in $\mathcal{G}$

$$\begin{aligned}
[S^{\boxed{1}},\ S^{\boxed{1}}] &\Rightarrow_{\mathcal{G}}^{s_1} [A^{\boxed{1}} B^{\boxed{2}}, B^{\boxed{2}} A^{\boxed{1}}] \\
&\Rightarrow_{\mathcal{G}}^{s_2} [aA^{\boxed{3}} bB^{\boxed{2}}, B^{\boxed{2}} bA^{\boxed{3}} a] \\
&\Rightarrow_{\mathcal{G}}^{s_2} [aaA^{\boxed{4}} bbB^{\boxed{2}}, B^{\boxed{2}} bbA^{\boxed{4}} aa] \\
&\Rightarrow_{\mathcal{G}}^{s_3} [aaabbbB^{\boxed{2}}, B^{\boxed{2}} bbbaaa] \\
&\Rightarrow_{\mathcal{G}}^{s_4} [aaabbbcB^{\boxed{5}} d, dB^{\boxed{5}} cbbbaaa] \\
&\Rightarrow_{\mathcal{G}}^{s_5} [aaabbbccdd, ddccbbbaaa]
\end{aligned}$$

It is not difficult to see that $T(\mathcal{G}) = \{[a^p b^p c^q d^q,\ d^q c^q b^p a^p] \mid p, q \geq 1\}$.

We conclude this section with a remark. Our definition of SCFG is essentially the same as the definition of the syntax-directed transduction grammars in [3] and the syntax-directed translation schemata in [1,2], as already mentioned in the introduction. The only difference is that in a synchronous rule $[A_1 \to \alpha_1,\ A_2 \to \alpha_2]$ we allow A_1, A_2 to be different nonterminals, while in the above formalisms we always have $A_1 = A_2$. Although our generalization does not add to the weak generative power of the model, that is, the class of translations generated by the two models are the same, it does increase its strong generative capacity, that is, the parse tree mappings defined by syntax-directed translation schemata are a proper subset of the parse tree mappings defined by SCFGs. As a consequence of this fact, when the definitions of the two models are enriched with probabilities, SCFGs can define certain parse tree distributions that cannot be captured by syntax-directed translation schemata, as argued in [13]. The above generalization has been adopted in several translation models for natural language.

2.2. Parsing strategies for SCFGs

Recognition and parsing algorithms for SCFGs are commonly used in the area of statistical machine translation. Despite the fact that the underlying problems are NP-hard, it has been experimentally shown that the typology of synchronous rules that we encounter in real world applications can be processed efficiently, for most of the cases, if we adopt the appropriate parsing strategy, as already discussed in Section 1. We are thus interested in the problem of finding optimal parsing strategies for synchronous rules, under some specific parsing framework.

Standard parsing algorithms for SCFGs exploit dynamic programming techniques, and are derived as a generalization of the well-known Cocke–Kasami–Younger algorithm for word recognition based on CFGs [11,22], which essentially uses a bottom–up strategy. All these algorithms are based on the representation described below. For a string $w = a_1 \cdots a_n$, $n \geq 1$, and for integers $i, j \in [n]_0$ with $i < j$, we write $w[i, j]$ to denote the substring $a_{i+1} \cdots a_j$. Assume we are given the input pair $[w_1, w_2]$. To simplify the discussion, we focus on a sample synchronous rule containing only occurrences of nonterminal symbols

$$\begin{aligned} s: [&A_1 \to A_{1,1}^{\boxed{1}} A_{1,2}^{\boxed{2}} A_{1,3}^{\boxed{3}} A_{1,4}^{\boxed{4}} A_{1,5}^{\boxed{5}} A_{1,6}^{\boxed{6}}, \\ &A_2 \to A_{2,1}^{\boxed{6}} A_{2,2}^{\boxed{1}} A_{2,3}^{\boxed{4}} A_{2,4}^{\boxed{2}} A_{2,5}^{\boxed{5}} A_{2,6}^{\boxed{3}}], \end{aligned} \tag{1}$$

Synchronous rule s can be associated with the permutation π of the set [6] identified by the sequence 614253, which is visualized in Fig. 1a. Recall that, for each $k \in [6]$, nonterminals $A_{1,k}, A_{2,\pi^{-1}(k)}$ are linked in rule s.

Assume that, for each $k \in [6]$, we have already parsed all possible occurrences of the linked nonterminals $A_{1,k}, A_{2,\pi^{-1}(k)}$ over $[w_1, w_2]$. If we want to parse the linked nonterminals A_1, A_2 in the left-hand side of rule s, we need to check whether some of the derivations for the linked nonterminals $A_{1,k}, A_{2,\pi^{-1}(k)}$ can be combined, according to the permutation associated to s, to provide a parse of two contiguous substrings of $[w_1, w_2]$. Formally, we need to explore the search space of all integers $i_{1,0} \leq i_{1,1} \leq \cdots \leq i_{1,6}$, $i_{1,k} \in [|w_1|]_0$, and $i_{2,0} \leq i_{2,1} \leq \cdots \leq i_{2,6}$, $i_{2,k} \in [|w_2|]_0$, and check whether, for some choice of these integers and for each $k \in [6]$, we have

$$[A_{1,k}^{\boxed{1}}, A_{2,\pi^{-1}(k)}^{\boxed{1}}] \Rightarrow_{\mathcal{G}}^{*} [w_1[i_{1,k-1}, i_{1,k}],\ w_2[i_{2,\pi^{-1}(k)-1}, i_{2,\pi^{-1}(k)}]]. \tag{2}$$

Informally, when condition (2) holds we say that linked nonterminals $A_{1,k}, A_{2,\pi^{-1}(k)}$ span substrings $w_1[i_{1,k-1}, i_{1,k}]$ and $w_2[i_{2,\pi^{-1}(k)-1}, i_{2,\pi^{-1}(k)}]$ of the input.

For reasons of computational efficiency, it is advantageous to break the parsing of synchronous rule s into several steps, which collectively determine a **strategy**. In this article, we restrict ourselves to strategies which add linked nonterminals one pair at each step, according to some fixed total ordering, which we call a **linear** parsing strategy. The result of the partial analyses obtained at each step is represented by means of a data structure which we call a **state**. To provide a concrete example, let us choose the linear parsing strategy σ of gathering all the A_{1k}'s on the first component of rule s from left to right. At the first step, we then collect linked nonterminals $A_{1,1}, A_{2,\pi^{-1}(1)} = A_{2,2}$ and construct the partial analysis represented by the state $\langle(s,\sigma,1), (i_{1,0}, i_{1,1}), (i_{2,1}, i_{2,2})\rangle$, meaning that $A_{1,1}$ spans substring $w_1[i_{1,0}, j_{1,1}]$ and $A_{2,2}$ spans substring $w_2[i_{2,1}, j_{2,2}]$. The first element in the state, $(s,\sigma,1)$, indicates that this state is generated from synchronous rule s after the first combination step, assuming our current strategy σ. We refer to this first element as the **type** of the state.

At the second step we add to our partial analysis the linked nonterminals $A_{1,2}, A_{2,4}$, as shown in Fig. 1b. We construct a new state $\langle(s,\sigma,2),\ (i_{1,0}, i_{1,2}),\ (i_{2,1}, i_{2,2}),\ (i_{2,3}, i_{2,4})\rangle$, meaning that $A_{1,1}, A_{1,2}$ together span $w_1[i_{1,0}, i_{1,2}]$, $A_{2,2}$ spans $w_2[i_{2,1}, i_{2,2}]$ and $A_{2,4}$ spans $w_2[i_{2,3}, i_{2,4}]$. Note that the integer $i_{1,1}$ is dropped from the description of the state, since it will not be referenced by any further step based on the associated partial analysis. After adding the third pair of linked nonterminals $A_{1,3}, A_{2,6}$, we create state $\langle(s,\sigma,3),\ (i_{1,0}, i_{1,3}),\ (i_{2,1}, i_{2,2}),\ (i_{2,3}, i_{2,4}),\ (i_{2,5}, i_{2,6})\rangle$, spanning four separate substrings of the input, as shown in Fig. 1c. After adding the linked nonterminals $A_{1,4}, A_{2,3}$, we have that the span of $A_{2,3}$ fills in the gap between the spans of the previously parsed nonterminals $A_{2,2}$ and $A_{2,4}$, as shown in Fig. 1d. We can then collapse these three spans into a single string, obtaining a new state $\langle(s,\sigma,4),\ (i_{1,0}, i_{1,4}),\ (i_{2,1}, i_{2,4}),\ (i_{2,5}, i_{2,6})\rangle$ which spans three separate substrings of the input. Finally, at the next two steps states of type $(s,\sigma,5)$ and $(s,\sigma,6)$ can be constructed, each spanning two substrings only.

We refer below to the number of substrings spanned by a state as the **fan-out** of the state (this notion will be formally defined later). The above example shows that the fan-out of each state depends on the parsing strategy that we are adopting.

Bottom–up dynamic programming algorithms for the parsing problem for SCFGs are designed on the basis of the above state representation for partial analyses. These algorithms store in some appropriate data structure the states that have already been constructed, and then retrieve and combine states in order to construct new states. Let n be the maximum length between the input strings w_1 and w_2. Because a state with fan-out f may have $\mathcal{O}(n^{2f})$ instantiations, fan-out provides a way of bounding the space complexity of our algorithm. When we use linear parsing strategies, fan-out is also relevant in assessing upper bounds on time complexity. Consider the basic step of adding the $(k+1)$-th pair of linked nonterminals to a state of type (s,σ,k) having fan-out f. As before, we have $\mathcal{O}(n^{2f})$ instantiations for states of type (s,σ,k). We also have $\mathcal{O}(n^4)$ possible instantiations for the span of the $(k+1)$-th pair, since any pair of linked nonterminals spans exactly two substrings. However, the $(k+1)$-th pair might share some of its boundaries with the boundaries of the state of type (s,σ,k), depending on the permutation associated with the synchronous rule s. If we define $\delta(s,\sigma,k)$ as the number of independent boundaries in the $(k+1)$-th pair, with $0 \leq \delta(s,\sigma,k) \leq 4$, we have that all executions of the above step can be carried out in time $\mathcal{O}(n^{2f+\delta(s,\sigma,k)})$.

If we want to optimize the space or the time complexity of a dynamic programming algorithm cast in the above framework, we need to search for a parsing strategy that minimizes the maximum fan-out of its states, or else a strategy that

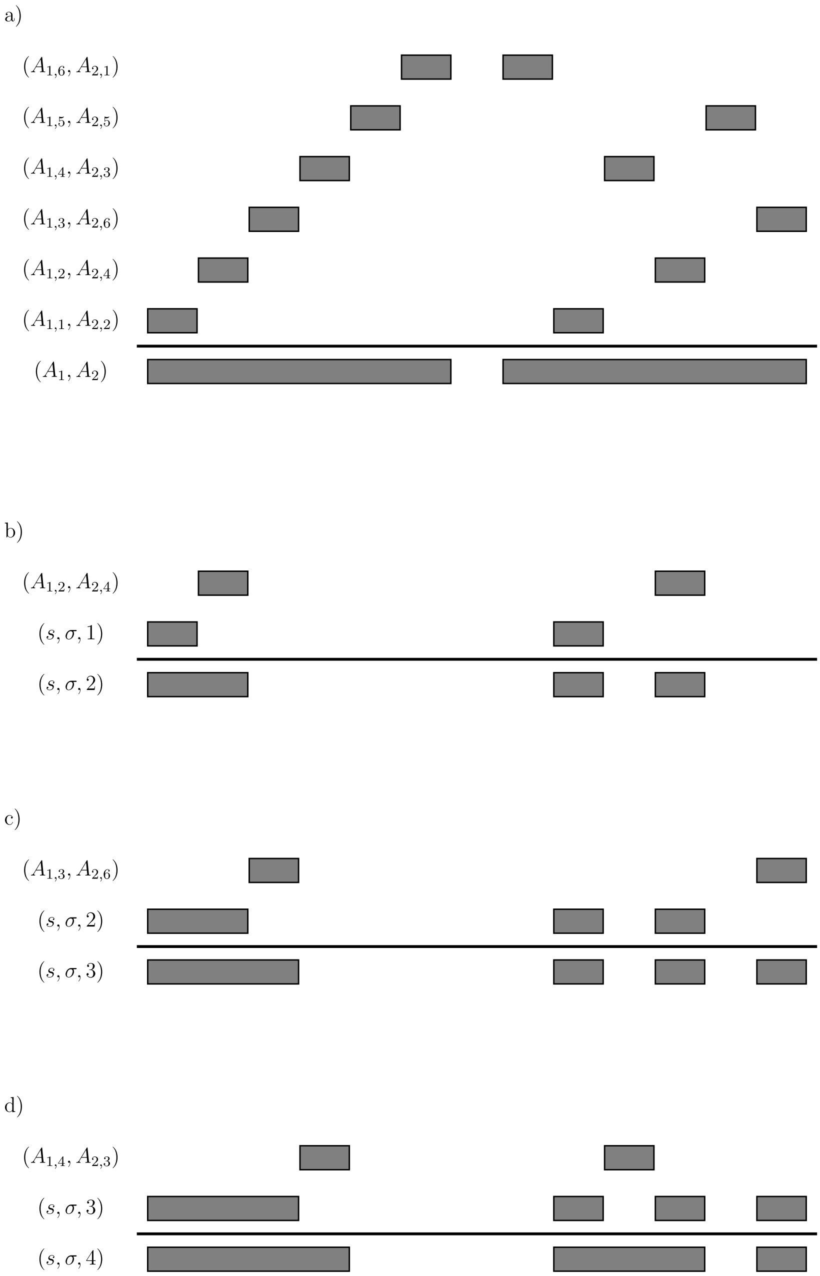

Fig. 1. a): combining spans to parse the SCFG rule s of Eq. (1). b), c) and d): the first three steps in a linear parsing strategy for this rule.

minimizes the maximum value of the sum of the fan-out and the $\delta()$ function. This needs to be done for each individual synchronous rule in the grammar. In our running example, the critical step is provided by state type $(s, \sigma, 3)$ with fan-out 4, leading to space complexity of $\mathcal{O}(n^8)$. Furthermore, the combination of state type $(s, \sigma, 2)$ (fan-out 3) with linked pair $A_{1,3}, A_{2,6}$ ($\delta(s, \sigma, 3) = 3$) leads to time complexity of $\mathcal{O}(n^9)$. However, we can switch to a different strategy σ', by collecting linked nonterminal pairs in s in the order given by the left components $A_{1,4}, A_{1,5}, A_{1,2}, A_{1,3}, A_{1,1}, A_{1,6}$. According to this new strategy, states of types $(s, \sigma', 2)$ and $(s, \sigma', 3)$ both have fan-out three, while every other state type has fan-out two. This leads to space complexity of $\mathcal{O}(n^6)$ for rule s. It is not difficult to see that this strategy is also space optimal for rule s, on the basis of the observation that any grouping of two linked nonterminals $A_{1,k}, A_{2,\pi^{-1}(k)}$ and $A_{1,k'}, A_{2,\pi^{-1}(k')}$ with $k, k' \in [6]$ and $k \neq k'$, has a fan-out of at least three. As for the time complexity, the critical step is the combination of state type $(s, \sigma', 2)$ (fan-out 3) with linked pair $A_{1,2}, A_{2,4}$ ($\delta(s, \sigma', 3) = 2$), leading to time complexity of $\mathcal{O}(n^8)$ for this strategy. It is not difficult to verify that σ' is also a time optimal strategy.

2.3. Fan-out and optimization of parsing

What we have informally shown in the previous section is that, under the outlined framework based on state representations for partial analyses, we can exploit the properties of the specific permutation of a given synchronous rule to reduce the maximum fan-out of states, and hence improve the space and time complexity of our parsing algorithms. In this section, we provide formal definitions of these concepts, and introduce the computational problem that is investigated in this article.

Let s be a synchronous rule with $r > 2$ linked nonterminals, and let π_s be the permutation representing s. A **linear parsing strategy** for s is defined as a permutation σ_s of the set $[r]$. The intended meaning of σ_s is that, when parsing the rule s, the pair of linked nonterminals $A_{1,\sigma(k)}, A_{2,\pi^{-1}(\sigma(k))}$ is collected at the k-th step, for each $k \in [r]$, as shown in Fig. 1.

Let us consider state type (s, σ_s, k), $k \in [r]$, defined as in Section 2.2. We define the count of **internal boundaries** for (s, σ_s, k) as

$$\begin{aligned}\mathbf{ib}(\pi_s, \sigma_s, k) = & \left|\left\{h : \sigma_s^{-1}(h) \le k \wedge \sigma_s^{-1}(h+1) > k\right\}\right| \\ & + \left|\left\{h : \sigma_s^{-1}(h) > k \wedge \sigma_s^{-1}(h+1) \le k\right\}\right| \\ & + \left|\left\{h : \sigma_s^{-1}(\pi_s^{-1}(h)) \le k \wedge \sigma_s^{-1}(\pi_s^{-1}(h+1)) > i\right\}\right| \\ & + \left|\left\{h : \sigma_s^{-1}(\pi_s^{-1}(h)) > k \wedge \sigma_s^{-1}(\pi_s^{-1}(h+1)) \le k\right\}\right| .\end{aligned} \tag{3}$$

In the definition above, the term $\left|\left\{h : \sigma_s^{-1}(h) \le k \wedge \sigma_s^{-1}(h+1) > k\right\}\right|$ counts the number of nonterminals $A_{1,i}$ that have already been collected at step k and such that nonterminal $A_{1,i+1}$ has not yet been collected. Informally, this term counts the nonterminals in the right-hand side of the first CFG rule component of s that represent right internal boundaries of the span of a state of type (s, σ_s, k). The second term in the definition counts the number of nonterminals in the right-hand side of the same rule component that represent left internal boundaries. Similarly, the remaining two terms count right and left internal boundary nonterminals, respectively, in the right-hand side of the second CFG rule component of s.

For state type (s, σ_s, k), $k \in [r]$, we also define the count of **external boundaries** as

$$\begin{aligned}\mathbf{eb}(\pi_s, \sigma_s, k) = & I(\sigma_s^{-1}(1) \le k) + I(\sigma_s^{-1}(n) \le k) \\ & + I(\sigma_s^{-1}(\pi_s^{-1}(1)) \le k) + I(\sigma_s^{-1}(\pi_s^{-1}(n)) \le k) .\end{aligned} \tag{4}$$

The indicator functions $I()$ count the number of nonterminals that are placed at the left and right ends of the right-hand sides of the two rule components and that have already been collected at step k. Informally, the sum of these functions counts the nonterminals that represent external boundaries of the span of state type (s, σ_s, k).

Finally, the **fan-out** of state type (s, σ_s, k) is defined as

$$\mathbf{fo}(\pi_s, \sigma_s, k) = \frac{1}{2}(\mathbf{ib}(\pi_s, \sigma_s, k) + \mathbf{eb}(\pi_s, \sigma_s, k)) . \tag{5}$$

Dividing the total number of boundaries by two gives the number of substrings spanned by the state type (s, σ_s, k). Observe that the fan-out at step k is a function of both the permutation π_s associated with the SCFG rule s, and the linear parsing strategy σ_s.

As discussed in Section 2.2, the fan-out at step k gives space and time bounds on the parsing algorithm relative to that step and parsing strategy σ_s. Thus the complexity of the parsing algorithm relative to synchronous rule s depends on the fan-out at the most complex step of σ_s. We wish to find, for an input synchronous rule s with associated permutation π_s, the linear parsing strategy that minimizes quantity

$$\min_{\sigma} \max_{k \in [r]} \mathbf{fo}(\pi_s, \sigma, k) , \tag{6}$$

where σ ranges over all possible linear parsing strategies for s. Our main result in this article is that this minimization problem is NP-hard. This is shown by first proving that the optimization of the $\mathbf{ib}(\pi_s, \sigma_s, k)$ component of the fan-out is NP-hard, in the next section, and then by extending the result to the whole fan-out in a successive section.

3. Permutation multigraphs and cutwidth

With the goal of showing that the minimization problem in (6) is NP-hard, in this section we investigate the minimization problem for the $\mathbf{ib}(\pi_s, \sigma_s, k)$ component of the fan-out, defined in (3). More precisely, given as input a synchronous rule s with $r > 2$ nonterminals and with associated permutation π_s, we investigate a decision problem associated with the computation of the quantity

$$\min_{\sigma} \max_{k \in [r]} \mathbf{ib}(\pi_s, \sigma, k) , \tag{7}$$

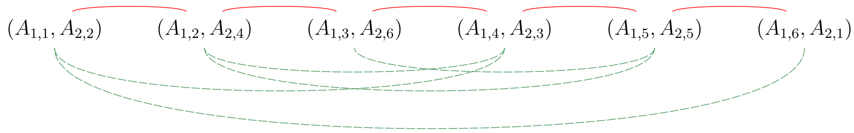

Fig. 2. The permutation multigraph corresponding to the SCFG rule s of Eq. (1). In this figure and all subsequent figures, green edges are shown with dashed lines.

where σ ranges over all possible linear parsing strategies for s. We do this by introducing a multigraph representation for the synchronous rule s, and by studying the so-called cutwidth problem for such a multigraph.

3.1. Permutation multigraphs

Our strategy for proving the NP-hardness of the optimization problem in (7) will be to reduce to the problem of finding the cutwidth of a certain class of multigraphs, which represent the relevant structure of the input synchronous rule. In this section we introduce this class of multigraphs, and discuss its relation with synchronous rules. We denote undirected multigraphs as pairs $G = (V, E)$, with set of nodes V and multiset of edges E.

A **permutation multigraph** is a multigraph $G = (V, A \uplus B)$ such that both $P_A = (V, A)$ and $P_B = (V, B)$ are Hamiltonian paths, and $\uplus$ is the merge operation defined for multisets. In the following, the edges in A will be called **red**, the edges in B will be called **green**.

A permutation multigraph $G = (V, A \uplus B)$ can be thought of as encoding some permutation: if we identify nodes in V with integers in $[|V|]$ according to their position on path A, the order of vertices along path B defines a permutation of the set $[|V|]$. We can therefore use a permutation multigraph to encode the permutation associated with a given synchronous rule. More precisely, let s be a synchronous rule of the form

$$\begin{aligned} s:\ [&A_1 \rightarrow u_{1,0} A_{1,1}^{\boxed{1}} u_{1,1} \cdots u_{1,r-1} A_{1,r}^{\boxed{r}} u_{1,r}, \\ &A_2 \rightarrow u_{2,0} A_{2,1}^{\boxed{\pi_s(1)}} u_{2,1} \cdots u_{2,r-1} B_{2,r}^{\boxed{\pi_s(r)}} u_{2,r}]\,, \end{aligned} \tag{8}$$

where $r \geq 2$, $u_{1,i}, u_{2,i} \in V_T^*$ for each $i \in [r]_0$ and π_s is a permutation of the set $[r]$. We associate with s the permutation multigraph $G_s = (V_s, E_{s,A} \uplus E_{s,B})$ defined as

- $V_s = \{(A_{1,i}, A_{2,\pi_s^{-1}(i)}) : i \in [r]\}$;
- $E_{s,A} = \{((A_{1,i}, A_{2,\pi_s^{-1}(i)}), (A_{1,j}, A_{2,\pi_s^{-1}(j)})) : i, j \in [r] \wedge |i - j| = 1\}$;
- $E_{s,B} = \{((A_{1,\pi_s(i)}, A_{2,i}), (A_{1,\pi_s(j)}, A_{2,j})) : i, j \in [r] \wedge |i - j| = 1\}$.

To see that G_s is a permutation multigraph, observe that G_s is the superposition of the following two Hamiltonian paths

- $\langle (A_{1,1}, A_{2,\pi_s^{-1}(1)}), ((A_{1,2}, A_{2,\pi_s^{-1}(2)}), \ldots, (A_{1,r-1}, A_{2,\pi_s^{-1}(r-1)}), (A_{1,r}, A_{2,\pi_s^{-1}(r)}) \rangle$;
- $\langle (A_{1,\pi_s(1)}, A_{2,1}), (A_{1,\pi_s(2)}, A_{2,2}), \ldots, (A_{1,\pi_s(r-1)}, A_{2,r-1}), (A_{1,\pi_s(r)}, A_{2,r}) \rangle$.

An example permutation multigraph is shown in Fig. 2, with one Hamiltonian path shown above and one below the vertices.

We shall now discuss a mathematical relation between internal boundary counts for states associated with linear parsing strategies for the synchronous rule s and width values for the permutation multigraph G_s. We first recall the definition of the width and cutwidth of a graph and a multigraph. Let $G = (V, E)$ be an undirected (multi)graph such that $|V| = n > 1$. A **linear arrangement** of G is a bijective mapping ν from V to $[n]$. We call **positions** the integer values of ν. For any $i \in [n-1]$, the **width** of G at i with respect to ν, denoted by $\mathbf{wd}(G, \nu, i)$, is defined as $|\{(u, v) \in E : \nu(u) \leq i < \nu(v)\}|$. In the case of a multigraph, the size of the previous set should be computed taking into account multiple occurrences. Informally, $\mathbf{wd}(G, \nu, i)$ is the number of distinct edges crossing over the gap between positions i and $i+1$ in the linear arrangement ν. To simplify the notation below, we also let $\mathbf{wd}(G, \nu, n) = 0$. The **cutwidth** of G is then defined as

$$\mathbf{cw}(G) = \min_{\nu} \max_{i \in [n]} \mathbf{wd}(G, \nu, i)\,,$$

where ν ranges over all possible linear arrangements of G. The cutwidth of the multigraph of Fig. 2 is six, which is achieved between $(A_{1,3}, A_{2,6})$ and $(A_{1,4}, A_{2,3})$ in the linear arrangement shown.

Let us now consider synchronous rule s in (8) and the associated permutation π_s, and let σ_s be some linear parsing strategy defined for s. The **linear arrangement associated with** σ_s is the linear arrangement ν_s for permutation multigraph $G_s = (V_s, E_s)$ defined as follows. For each $i \in [r]$, $\nu_s((A_{1,i}, A_{2,\pi_s^{-1}(i)})) = k$ if and only if $\sigma_s(k) = (A_{1,i}, A_{2,\pi_s^{-1}(i)})$. The following relation motivates our investigation of the cutwidth problem for permutation multigraphs in the remaining part of this section.

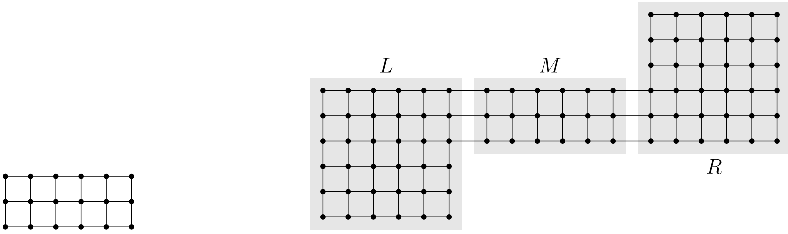

Fig. 3. The $\Gamma[3,6]$ grid (left), whose cutwidth is 4, and the composed grid $\Sigma[6,6,3,6]$ (right) with grids L, M and R shaded.

Lemma 1. *Let s be a synchronous rule with $r > 2$ linked nonterminals, and let σ_s be a linear parsing strategy for s. Let π_s and G_s be the permutation and the permutation multigraph, respectively, associated with s, and let ν_s be the linear arrangement for G_s associated with σ_s. For every $i \in [r]$ we have*

$$\mathbf{wd}(G_s, \nu_s, i) = \mathbf{ib}(\pi_s, \sigma_s, i) .$$

Proof. The lemma follows from the definition of the internal boundary function in (3) and the definition of the permutation multigraph. The first two terms in (3) count edges from the set $E_{s,A}$ crossing the gap at position i in linear arrangement ν_s of G_s, while the second two terms in (3) count edges from the set $E_{s,B}$. □

Note that Lemma 1 directly implies the relation $\mathbf{cw}(G_s) = \min_\sigma \max_{i\in[r]} \mathbf{ib}(\pi_s, \sigma, k)$.

In the rest of the present section we investigate the **permutation multigraph cutwidth** problem, or `PMCW` for short. An instance of `PMCW` consists of a permutation multigraph G and an integer k, and we have to decide whether $\mathbf{cw}(G) \le k$. We show that the `PMCW` problem is NP-complete. We reduce from the minimum bisection width problem, or `MBW` for short. The `MBW` problem consists of deciding whether, given a graph G and an integer k, there is a partition of the nodes of G into two equal size subsets V_1 and V_2, such that the number of edges with one endpoint in V_1 and one endpoint in V_2 is not greater than k. It is known that the `MBW` problem is NP-complete even when restricted to cubic graphs (graphs where every vertex has three edges) with no multi-edges and no self-loops [23]. We use this variant of the `MBW` problem in our reduction. Our proof that the `PMCW` problem is NP-complete is a modification of the proof reported in [24, Theorem 4.1, p. 434], showing that the problem of deciding whether an undirected graph has (modified) cutwidth not greater than a given integer is NP-complete for graphs with maximum vertex degree of three.

3.2. Construction of permutation multigraph G'

Throughout the rest of this section, we let $G = (V, E)$ be a cubic graph where $V = \{v_1, \dots, v_n\}$ is the set of its vertices. Note that $n > 1$ must be even. We also let k be an arbitrary positive integer. We construct a permutation multigraph G' and an integer k' such that $\langle G, k\rangle$ is a positive instance of `MBW` if and only if $\langle G', k'\rangle$ is a positive instance of `PMCW` (this statement will be shown in Sections 3.3 and 3.4).

Let H and W be positive integers. We will make use of a grid gadget $X = \Gamma[H, W]$ with H rows and W columns; for an example, see the left part of Fig. 3. More precisely, for any $h \in [H]$ and for any $w \in [W]$, the grid X includes a node $x^{h,w}$. Moreover, for any $h \in [H]$ and for any $w \in [W-1]$, there is an edge $\left(x^{h,w}, x^{h,w+1}\right)$, and, for any $h \in [H-1]$ and for any $w \in [W]$, there is an edge $\left(x^{h,w}, x^{h+1,w}\right)$. It is known that, for any H and W greater than 2, $\mathbf{cw}(X) = \min\{H+1, W+1\}$ [25].

For positive integers H_l, W_l, H_m, and W_m with $H_l > H_m$, we will also exploit a **composed grid** $\Sigma[H_l, W_l, H_m, W_m]$, which is formed by combining two grid gadgets $L = \Gamma[H_l, W_l]$ and $R = \Gamma[H_l, W_l]$ with a grid gadget $M = \Gamma[H_m, W_m]$, as shown in the right part of Fig. 3. The nodes of L, R, and M will be denoted as $l^{h,w}$, $r^{h,w}$, and $m^{h,w}$, respectively. Besides the edges of the three grids L, R, and M, for any $h \in [H_m]$, the composed grid $\Sigma[H_l, W_l, H_m, W_m]$ also includes the edges $\left(l^{h,W_l}, m^{h,1}\right)$ and $\left(m^{h,W_m}, r^{H_l-H_m+h,1}\right)$.

The target graph G' consists of several grid gadgets, as shown in Fig. 4. More specifically, it has one grid $G_i = \Gamma[2n^4+1, 6n^4]$, $i \in [n]$, for each of the n nodes of the source cubic graph G. The nodes of G_i will be denoted as $g_i^{h,w}$. In addition, G' has a composed grid $S = \Sigma[3n^4+1, 12n^4, 2n^4+1, 8n^4+1]$. For each grid G_i, $i \in [n]$, we add to G' a sheaf of $4n^2$ edges connecting distinct nodes in G_i to $4n^2$ distinct nodes of M, as will be explained in detail below. In addition, for each edge $(v_i, v_j) \in E$ with $i < j$, we add to G' two edges, each edge connecting a node in G_i with a node in G_j. The choice of all of the above connections will be done in a way that guarantees that G' is a permutation multigraph. The large grids L and R and the wide sheaves connecting each G_i to M are designed to force half of the G_i grids to fall on either side of M in any layout of optimal cutwidth. This in turn guarantees that we have a layout of low cutwidth only if the source graph G has a bisection of small width.

Before providing a mathematical specification of G', we informally summarize the organization of the edges of G' connecting the M and G_i grids. When scanning the columns of M from left to right, we will have a first column that has no connection to any of the G_i components, followed by a first block of $4n^2$ columns with connections to the G_1 component,

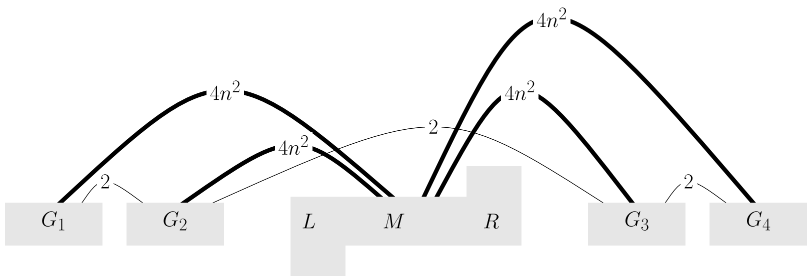

Fig. 4. Overview of the graph G' constructed from a graph G consisting of four vertices v_1, v_2, v_3, v_4, and three edges (v_1, v_2), (v_2, v_3), (v_3, v_4).

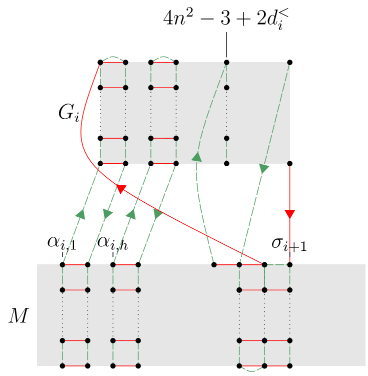

Fig. 5. Block 1 is composed by all columns of G_i connected with M as in the pattern displayed by the two green columns at the left of position $4n^2 - 3 + 2d_i^<$. Block 3 is composed by all columns of G_i starting with the column at position $4n^2 - 3 + 2d_i^<$ and extending to the right.

followed by a second block of $4n^2$ columns with connections to G_2, and so on up to the n-th block of $4n^2$ columns with connections to G_n. The remaining columns of M do not have any connection with the G_i components.

Looking at one of the grids G_i, the columns are organized into three blocks, when scanning from left to right.

- **Block 1** (left portion of G_i in Fig. 5): The first column has two edges connecting to the M component, one from its top vertex and one from its bottom vertex, and the remaining columns in the block each have a single edge connecting to M from the column's bottom vertex. This block extends from the column with index 1 to column $4n^2 - 4 - 2d_i^>$, where $d_i^>$ denotes the number of edges of G of the form (v_i, v_j) with $j > i$, that is, the number of "forward" neighbors of i. The block therefore contains a total of $4n^2 - 3 - 2d_i^>$ edges connecting G_i to M.
- **Block 2** (Fig. 6): This block represents the edges from the source graph G. For each edge (v_i, v_j) of G such that $i < j$, we have two columns each having a single edge connecting to M from the column's bottom vertex, and a single edge connecting to the grid G_j from the column's top vertex (such as the column labeled $\gamma_{i,h}$ in Fig. 6). This is followed by two columns for each edge (v_i, v_j) in G such that $j < i$, with each column having its bottom vertex connected to the grid G_j and no connection to M (such as the column labeled $\gamma'_{i,h}$ in Fig. 6). Block 2 extends from column $4n^2 - 3 - 2d_i^>$ to column $4n^2 - 4 + 2d_i^<$, where $d_i^<$ denotes the number of edges of G of the form (v_i, v_j) with $j < i$, that is, the number of "backward" neighbors of i. The number of edges connecting G_i to M is $2d_i^>$, and the number of edges connecting G_i to grids G_j with $j \neq i$ is $2d_i^> + 2d_i^< = 6$, where the equality follows from the fact that G is a cubic graph.
- **Block 3** (right portion of G_i in Fig. 5): The first column has the top vertex connected to M. All remaining columns of G_i do not have connections with M, with the exception of the rightmost column, which has two edges connecting its top and bottom vertices to M. This block extends from the column with index $4n^2 - 3 + 2d_i^<$ to the column with index $6n^4$, and the block contains 3 edges connecting G_i to M.

Altogether, the above blocks provide a total number of edges connecting G_i and M equal to $(4n^2 - 3 - 2d_i^>) + 2d_i^> + 3 = 4n^2$, as anticipated.

To define the edges in each of these three blocks, we need to introduce some auxiliary notation. In what follows, for every $i \in [n]$, we denote by $N_i^<$ the set of "backward" neighbors of v_i, that is, $N_i^< = \{j : (v_i, v_j) \in E \wedge j \in [i-1]\}$. Similarly, the set of "forward" neighbors of v_i is the set $N_i^> = \{j : (v_i, v_j) \in E \wedge j \in [n] \setminus [i]\}$. We then have $d_i^< = |N_i^<|$ and $d_i^> = |N_i^>|$. As already observed, we must have $d_i^> + d_i^< = 3$ since G is a cubic graph. For every $i \in [n]$ and $j \in [i]$ we also define $d_i^{<,j} = |N_i^< \cap [j-1]|$; in words, $d_i^{<,j}$ is the number of backward neighbors of v_i having index strictly smaller than j. Note that $d_i^{<,i} = d_i^<$ for every $i \in [n]$.

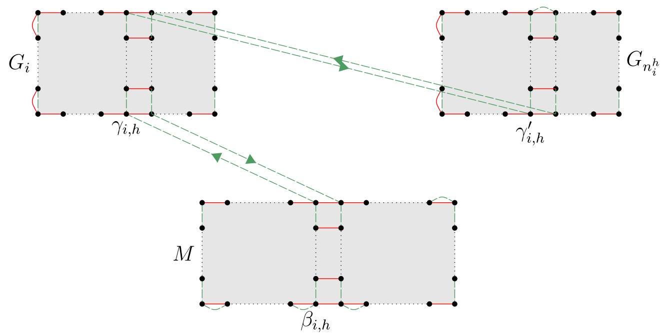

Fig. 6. Block 2 is composed by all columns of G_i representing connections between grids of G' that correspond to edges of the source graph G.

For every $i \in [n]$ and $h \in [d_i^>]$, we denote by n_i^h the index of the h-th forward neighbor of v_i from left to right: formally, n_i^h is the value j such that $j \in N_i^>$ and $|[j-1] \cap N_i^>| = h-1$. Finally, for every $i \in [n+1]$, we define $\sigma_i = 4n^2(i-1)+1$. This quantity will be used in the construction of G' below as an offset when locating the index of the next available column, from left to right, in the M component. For instance, we have $\sigma_1 = 1$ since in M the first block with $4n^2$ connections to G_1 starts at column 2, as already anticipated.

We are now ready to specify precisely each of the three blocks of edges connecting each G_i to the other grids.

Block 1 For any $h \in \left[2n^2 - 2 - d_i^>\right]$, G' includes the edges (see Fig. 5)

$$\left(m^{1,\alpha_{i,h}}, g_i^{2n^4+1,2h-1}\right), \left(g_i^{2n^4+1,2h}, m^{1,\alpha_{i,h}+1}\right),$$

where $\alpha_{i,h} = \sigma_i + (2h-1)$. In addition to the above edges, there is one edge connecting node $g_i^{1,1}$ with M that is associated with Block 1. However, in order to simplify the presentation, it is more convenient to list this edge under Block 3 below.

Block 2 We now add to G' the edges that are derived from the original graph G. For every $i \in [n]$ and for every $h \in \left[d_i^>\right]$, that is, for any forward edge $(v_i, v_{n_i^h})$ in G, G' includes the four edges (see Fig. 6)

$$\left(m^{1,\beta_{i,h}}, g_i^{2n^4+1,\gamma_{i,h}}\right), \left(g_i^{1,\gamma_{i,h}}, g_{n_i^h}^{2n^4+1,\gamma'_{i,h}}\right), \left(g_{n_i^h}^{2n^4+1,\gamma'_{i,h}+1}, g_i^{1,\gamma_{i,h}+1}\right), \left(g_i^{2n^4+1,\gamma_{i,h}+1}, m^{1,\beta_{i,h}+1}\right),$$

where $\beta_{i,h} = \sigma_i + 4n^2 - 4 - 2d_i^> + (2h-1)$, $\gamma_{i,h} = 4n^2 - 4 - 2d_i^> + (2h-1)$, and $\gamma'_{i,h} = 4n^2 - 3 + 2d^{<,i}_{n_i^h}$. Observe that $\beta_{i,1} = \alpha_{i,2n^2-2-d_i^>} + 1$, so that the two runs of edges defined by Blocks 1 and 2, connecting grid G_i to grid M, are one next to the other. We have thus added to G', for any edge (v_i, v_j) of G, two edges connecting the two grids G_i and G_j, and two edges connecting grid G_i to grid M.

Combining Blocks 1 and 2 together, we have added a total of $4n^2 - 4$ edges from grid G_i to grid M, for every $i \in [n]$.

Block 3 Finally, G' includes the four edges (see Fig. 5)

$$\left(m^{1,\sigma_{i+1}-3}, g_i^{1,4n^2-3+2d_i^<}\right), \left(g_i^{1,6n^4}, m^{1,\sigma_{i+1}-2}\right), \left(m^{1,\sigma_{i+1}-1}, g_i^{1,1}\right), \left(g_i^{2n^4+1,6n^4}, m^{1,\sigma_{i+1}}\right).$$

Combining all three blocks, we have a total of $4n^2$ edges connecting G_i to M. We further note that each vertex of G_i has at most one edge connecting to vertices outside G_i; this property will play an important role later in our proofs.

So far we have specified G' as if it were a (plain) graph; however, G' is a permutation multigraph. The coloring of the edges of G', that is, the definition of the two edge sets A and B, is specified below by describing the Hamiltonian path of red edges and the Hamiltonian path of green edges. Some of the edges specified above are included in both the red and green paths; these are double edges in the multigraph G'.

Red path The red path is schematically represented in Fig. 7. The path starts from $l^{3n^4+1,1}$ (that is, the bottom left corner of L) and travels horizontally through the n^4 bottom lines of L, alternating the left-to-right and the right-to-left directions and moving upward, until it reaches node $l^{2n^4+1,1}$ coming from previous node $l^{2n^4+2,1}$. Then the path continues traveling horizontally through the three components L, M and R, once again alternating the horizontal directions. The path eventually reaches the node $m^{1,1}$ coming from previous node $l^{1,12n^4}$, since $2n^4+1$ is always odd. At this point, the red path continues horizontally, from left to right, until it arrives at some node $m^{1,\sigma_{i+1}-1}$, $i \in [n]$; let us call x_i such a node. Observe that x_i is

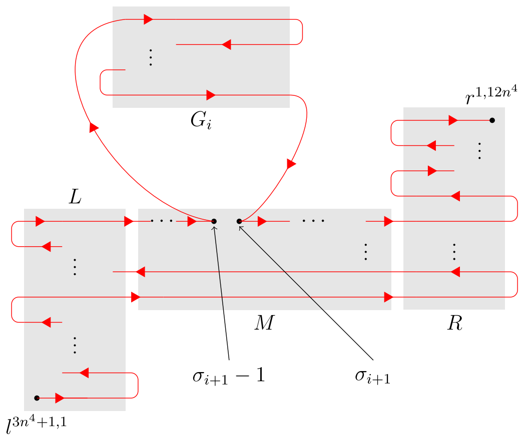

Fig. 7. The red path across graph G'. The displayed pattern that connects grid M with grid G_i is repeated for every $i \in [n]$.

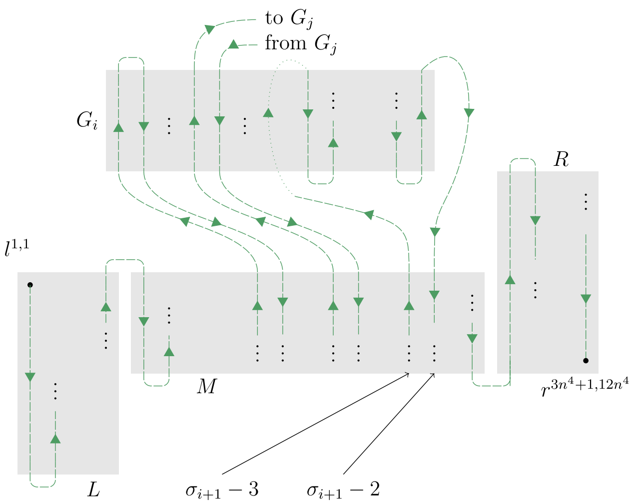

Fig. 8. The green path across graph G'. Grid M is connected with grid G_i, $i \in [n]$, through three different patterns, displayed in the figure, one for each of the three blocks of G_i.

the penultimate node in the first row of M, from left to right, that is connected with a node in the leftmost column of G_i (see Fig. 5).

Next, the path moves one step forward from x_i to the leftmost column of G_i, reaching node $g_i^{1,1}$. Now, the path starts traveling horizontally from the first to the last row of G_i, alternating the left-to-right direction with the right-to-left direction, until it arrives at $g_i^{2n^4+1,6n^4}$. Afterward, the red path returns to M by reaching the node to the right of x_i. The path then continues horizontally from left to right, repeating the process of visiting the components G_i for $i \in [n]$, as described above, eventually reaching node $r^{n^4+1,1}$ from previous node $m^{1,8n^4+1}$. The path can now visit the remaining nodes of R by traveling horizontally and alternating the left-to-right and the right-to-left directions, until it reaches node $r^{1,12n^4}$, where the path stops.

Green path The green path is schematically represented in Fig. 8. The path starts from $l^{1,1}$ (that is, the top left corner of L). It travels vertically, alternating the top-to-bottom direction with the bottom-to-top direction and moving rightward, until it reaches node $l^{1,12n^4}$ from previous node $l^{2,12n^4}$. The path then moves to $m^{1,1}$, travels vertically down to $m^{2n^4+1,1}$, moves one step to the right to node $m^{2n^4+1,2}$ and again travels vertically up to $m^{1,2}$. From now on, as soon as there is an edge exiting M and reaching some yet unvisited node in the last row of some grid G_i, the green path follows such an edge and travels vertically through the current column in G_i, until it reaches a node x_i in the first row. We need to distinguish two cases for x_i.

- If x_i has no edge exiting G_i, then the green path makes a step to the vertex to the right of x_i. This means that we are in Block 1 of G_i.

- On the other hand, if x_i has an edge exiting G_i, then the green path follows this edge thus reaching a node in the last row of a grid G_j, for some $j > i$ (as in Fig. 5). This means that we are visiting the first part of Block 2 of G_i, where edges (v_i, v_j) in the source graph G with $j > i$ are encoded by our construction. The path then travels vertically through two columns in G_j, until it reaches the node in the bottom row of the second column, and, from there, it returns to G_i at the vertex to the right of x_i.

From the vertex to the right of x_i, the green path continues vertically down in the current column of G_i. Upon reaching the bottom vertex of this column, the path exits G_i and comes back to some node in the top row of M. Such node is placed in the column of M adjacent at the right to the last column of M that the green path had visited before its exit to G_i. Then the green path proceeds downward, along the current column of M, it moves to the next column at the right, and alternates its direction to reach the node in the first row of M. The above process is then iterated, until all the columns in Blocks 1 and 2 of the current grid G_i have been visited.

When the green path reaches node $m^{1,\sigma_{i+1}-3}$, it exits M and reaches the top node in the column of G_i with index $4n^2 - 3 + 2d_i^<$ (see Fig. 5), entering for the first time Block 3 of the current grid G_i. We remark that this step represents a switch in the construction of the green path, in the following sense. Blocks 1 and 2 of G_i are visited by the green path in such a way that odd-indexed columns are visited bottom-to-top and even-indexed columns are visited top-to-bottom. On the other hand, when Block 3 of G_i is entered, we revert the previous pattern in such a way that odd-indexed columns are visited top-to-bottom and even-indexed columns are visited bottom-to-top.

Once Block 3 of G_i is entered, the green path travels vertically through its columns, by alternating direction and moving rightward, never leaving G_i at its intermediate nodes. In this way the path eventually reaches the node $g_i^{1,6n^4}$, at which point it can return to M, reaching the topmost node in the column with index $\sigma_{i+1} - 2$ (see again Fig. 5).

At this point the columns with indices $\sigma_{i+1} - 2$, $\sigma_{i+1} - 1$, and σ_{i+1} are visited vertically, alternating the top-to-bottom direction with the bottom-to-top direction and moving rightward. After this step, the green path is located at the bottom of column $\sigma_{i+1} + 1$, coming from the bottom of column σ_{i+1}, and it moves upward to the first line of M, where the path is ready to start visiting the next grid G_{i+1}. This is done by iterating all of the process described above.

When all of the grids G_i have been visited, there are no more edges exiting M. The path then continues vertically, alternating the top-to-bottom direction with the bottom-to-top direction and moving rightward, until it reaches node $m^{2n^4+1,8n^4+1}$, which forms the lower-right corner of M. The path can now move one step to the right to node $r^{3n^4+1,1}$, and visit the remaining nodes of R by traveling vertically and alternating the vertical direction. In the end, the green path ends at node $r^{3n^4+1,12n^4}$.

Double edges Some edges are included in both the red path and the green path. These are double edges in G', and count double when computing the width function. Double edges occur on the first and the last columns of the various component grids G_i, L, M, and R, where the red path crosses from one row to the next, and on the upper and lower rows of these grids, where the green path crosses from one column to the next. There are no duplicate edges in the interior of any grid. There are also no duplicate edges between any two grids, with the only exceptions of the points where the green path connects L to M, and the point where the green path connects M to R.

To be used later, we need to establish exact values for the cutwidth of the multigraph grids G_i, L, M, and R. In order to do so, we define a multigraph grid X_m, which consists of the (regular) grid $X = \Gamma[H, W]$ with the following double edges

- from $x^{1,i}$ to $x^{1,i+1}$, for any i odd in $[W - 1]$
- from $x^{H,i}$ to $x^{H,i+1}$, for any i even in $[W - 1]$
- from $x^{i,1}$ to $x^{i+1,1}$, for any i odd in $[H - 1]$
- from $x^{i,W}$ to $x^{i+1,W}$, for any i even in $[H - 1]$.

The set of edges of the grid X is contained in the set of edges of the multigraph grid X_m. Hence the cutwidth of X_m is at least $\min\{H + 1, W + 1\}$, which is the cutwidth of X as reported at the beginning of Section 3.2. Without loss of generality, let us assume $H \le W$, and let us consider the linear arrangement ν of X_m such that, for any $x^{i,j}$, $\nu(x^{i,j}) = (j - 1)H + i$ if j is even, and $(j - 1)H + H - i$ otherwise. It is easy to verify that ν induces a maximum width equal to $H + 1$. This proves that the cutwidth of the multigraph grid X_m is still $\min\{H + 1, W + 1\}$.

Observe that the multigraph grid R which we use in G' has a set of edges which is a proper subset of the set of edges of X_m, for appropriate values of H and W. The subset relation follows from the fact that R does not have double edges at the portion of its leftmost column that connects with M. Similarly, the multigraph grid L has a set of edges which is a proper subset of the set of edges of an upside-down instance of X_m. Thus both L and R have cutwidth $\min\{H + 1, W + 1\}$. Consider now grid M. The set of its edges is a proper subset of the edges of an upside-down instance of X_m. This follows from the fact that M does not have double edges at its leftmost and rightmost columns, where it connects with L and R, respectively. Furthermore, the green path through M sometimes leaves the first row of M to connect to some grid G_i, as depicted in Fig. 8. Thus we can claim a cutwidth of $\min\{H + 1, W + 1\}$ for this grid as well. It is easy to verify that each of the green paths through grids L, M, and R corresponds to a linear arrangement that realizes the maximum width of $\min\{H + 1, W + 1\}$ for these grids.

Finally, each grid G_i can be split into two parts. The first part consists of what we have called Blocks 1 and 2, and the second part consists of Block 3. The first part is a grid with a proper subset of the edges of an upside-down instance of X_m, for appropriate values of H and W. This is so because the green path at Block 2 repeatedly leaves G_i to connect to three other grids G_j, $j \neq i$, irrespective of whether these connections are backward or forward in the source graph G. The second part of G_i is an instance of X_m, for appropriate values of H and W. The difference between these two parts is due to the fact that, when moving from Block 2 to Block 3, the green path switches to a "reversed" pattern, as already observed in this section. An optimal linear arrangement for G_i can be defined by following the green path within each column of this grid, and moving from one column to the next in a left to right order. The maximum width in the first part of G_i, with the exception of the last column, is then $\min\{H+1, W+1\}$, and this is also the maximum width in the second part, with the exception of the first column. It is not difficult to verify that, even for the positions corresponding to the two adjacent columns above, this arrangement induces a maximum width of $\min\{H+1, W+1\}$. We have thus found that all of the grid components in G' have cutwidth of $\min\{H+1, W+1\}$.

We conclude our construction by setting $k' = 3n^4 + 2n^3 + 2k + 2$ in the target instance $\langle G', k' \rangle$ of the PMCW problem.

3.3. *MBW to PMCW*

We show here that if G admits a partition of its nodes into two equal size subsets, inducing a cut of size at most k, then G' admits a linear arrangement ν' whose maximum width is at most k'.

Lemma 2. *If $\langle G, k \rangle$ is a positive instance of MBW then $\langle G', k' \rangle$ is a positive instance of PMCW.*

Proof. We specify a linear arrangement of G' having width no greater than k' at each position. We arrange the vertices within each grid component G_i, as well as within grid components R, M, and L, to be contiguous to one another. Within each grid component, we arrange the vertices in column-major order proceeding through the columns from left to right; within each column, we place vertices in the order specified by the green path of G'. This is the same linear arrangement for each multigraph grid that has been presented in the paragraph "Double Edges", at the end of Section 3.2. Disregarding edges that are not internal to the grid itself, this results in a maximal width of $H + 1$ for each individual grid, as already discussed.

We concatenate the linear arrangements for the grid components in a manner corresponding to a solution of the MBW problem given by $\langle G, k \rangle$. To this end, let V_1 and V_2 be the sets in a partition of the vertices in G such that $|V_1| = |V_2|$ and at most k edges in G have one endpoint in V_1 and the other endpoint in V_2.

Our linear arrangement begins with the grid components G_i for all i such that $v_i \in V_1$, in any order, then concatenates components L, M, and R, and finally adds G_i for all i such that $v_i \in V_2$, in any order. Each position in the linear arrangement within component L has at most $3n^4 + 2$ edges internal to L, since the height of L's grid is $3n^4 + 1$. In addition, each position within L has $4n^2$ edges connecting M to each of the $\frac{n}{2}$ components G_i to the left of L, for a total of $2n^3$ edges. Finally, each position within L has at most $2k$ edges connecting components G_i and G_j for i, j such that $v_i \in V_1$, $v_j \in V_2$ and $(v_i, v_j) \in E$. Thus, the total width at each position within L is at most $3n^4 + 2 + 2n^3 + 2k = k'$. The same analysis applies to each position within R.

At all other positions in the linear arrangement, we have smaller width. This is because, for positions within each G_i, we have at most $2n^4 + 2$ edges from the grid G_i itself, at most $2n^3$ edges from M to any G_j standing on the same side as G_i with respect to M, and no more than $3n$ edges from some G_j to some G_h, since the source graph G is cubic and each connection between two vertices in G corresponds to two arcs connecting the associated grids in G'. For positions within M, we have at most $2n^4 + 2$ edges internal to M, at most $4n^3$ edges from M to some G_i, and $2k < 3n$ edges from some G_i to some G_j. Finally, positions between grid components have at most $2n^3$ edges from M to some G_i, at most $3n$ edges from some G_i to some G_j, and, in the case of positions between M and either L or R, $2n^4 + 1$ edges connecting M to either L or R. Thus, all positions outside grids L and R have a width bound of $2n^4 + \mathcal{O}(n^3)$.

We conclude that the maximum width of the linear arrangement is that of the L and R components, $3n^4 + 2 + 2n^3 + 2k = k'$. Then $\langle G', k' \rangle$ is a positive instance of PMCW. □

3.4. *PMCW to MBW*

In this section we shall prove that if G' admits a linear arrangement ν' whose maximum width is at most k', then our source graph G admits a partition into two equal size subsets of nodes inducing a cut of size at most k. To this aim, we need to develop several intermediate results. Informally, our strategy is to investigate the family of linear arrangements for G' having maximum width bounded by $k' + n^2$. We show that, in these arrangements, two important properties hold for the grid components L, M, R and G_i, $i \in [n]$, of G', described in what follows.

- The first property states that, for each grid component, a subset of its nodes must appear all in a row in the linear arrangement. We call such a subset the *kernel* of the grid. In other words, nodes from different kernels cannot be intermixed, and each linear arrangement induces a total order among the kernels. In addition, the kernels of the grids L,

M, R must appear one after the other in the total order, and the kernels of the grids G_i can only be placed to the left or to the right of the kernels of L, M, R. We therefore call L, M, R the middle grids.

- We illustrate the second property by means of an example. Consider one of the middle grids, say L. Assume that, under our liner arrangement, there is a grid X with kernel to the left of L's kernel and a grid Y with kernel to the right of L's kernel. Assume also some edge e of G', connecting a node x from X with a node y from Y. If x and y are in the kernels of their respective grids, edge e must cross over L's kernel, contributing one unit to the width of G' at each gap i within L's kernel. If x and y are not in the kernels of their respective grids, it is possible to "misplace" one of these two nodes, say x, moving it to the opposite side with respect to L's kernel, in such a way that e no longer contributes to the width at i. The second property states that, if we do this, we will bring new edges, internal to grid X, into the count of width at i. This means that, if our goal is the one of optimizing the width at i, we will have no gain in misplacing node x or node y.

With the two properties above, we can then show that exactly $\frac{n}{2}$ of the G_i grids must be placed to the left of the middle grids L, M, R, and all of the remaining G_i grids must be placed to the right of the middle grids, which eventually leads to the fact that if $\langle G', k'\rangle$ is a positive instance of PMCW then $\langle G, k\rangle$ is a positive instance of MBW.

We start with some preliminary results, needed to prove the first property above. Let V_1 and V_2 be sets of nodes from some graph with $V_1 \cap V_2 = \emptyset$, and let E be the set of edges of the graph. We write $\delta(V_1, V_2) = |\{(u, v) : (u, v) \in E \wedge u \in V_1 \wedge v \in V_2\}|$.

Lemma 3. *For any grid $X = \Gamma[H, W]$ with $W \geq 2H + 1$ and for any partition of its nodes in two sets V_1 and V_2 with $|V_1| \geq H^2$ and $|V_2| \geq H^2$, we have $\delta(V_1, V_2) \geq H$.*

Proof. We distinguish the following three cases.

(i) For each h with $1 \leq h \leq H$ there exist $w_{h,1}$ and $w_{h,2}$ with $1 \leq w_{h,1}, w_{h,2} \leq W$ such that $x^{h,w_{h,1}} \in V_1$ and $x^{h,w_{h,2}} \in V_2$. This implies that, for each row of the grid, there exists at least one edge connecting one node in V_1 to one node in V_2. Hence, $\delta(V_1, V_2) \geq H$.

(ii) There exists h with $1 \leq h \leq H$ such that, for any w with $1 \leq w \leq W$, $x^{h,w} \in V_1$. In this case, for each w with $1 \leq w \leq W$, either there exists h_w with $1 \leq h_w \leq H$ such that $x^{h_w,w} \in V_2$ (and, hence, the w-th column of X contributes to $\delta(V_1, V_2)$ by at least one unit) or else, for all h with $1 \leq h \leq H$, $x^{h,w} \in V_1$. This latter case can happen at most $\left\lfloor \frac{|V_1|}{H} \right\rfloor$ times: this implies that the former case happens at least $W - \left\lfloor \frac{|V_1|}{H} \right\rfloor$ times. Hence,

$$\delta(V_1, V_2) \geq W - \left\lfloor \frac{|V_1|}{H} \right\rfloor \geq \frac{WH - |V_1|}{H} = \frac{|V_2|}{H} \geq H,$$

where the last inequality is due to the fact that $|V_2| \geq H^2$.

(iii) There exists h with $1 \leq h \leq H$ such that, for any w with $1 \leq w \leq W$, $x^{h,w} \in V_2$. We can deal with this case similarly to the previous one.

The lemma thus follows. □

Corollary 1. *For any grid $X = \Gamma[H, W]$ with $W \geq 2H + 1$, for any linear arrangement ν of X, and for any i with $H^2 \leq i \leq HW - H^2$, $\mathbf{wd}(X, \nu, i) \geq H$.*

Proof. The result follows by observing that, for any i with $H^2 \leq i \leq HW - H^2$, we can define a partition of the nodes of the grid by including in V_1 all the nodes x such that $\nu(x) \leq i$ and by including in V_2 all the other nodes. Since this partition satisfies the hypothesis of the previous lemma, we have that $\mathbf{wd}(X, \nu, i) \geq H$. □

Let ν be an arbitrary linear arrangement for the nodes of G'. We denote by ν_i (respectively, ν_L, ν_M, and ν_R) the linear arrangement of G_i (respectively, L, M, and R) induced by ν. Moreover, for any node x of G_i (respectively, L, M, and R) and the associated position $p = \nu(x)$ under ν, we denote by $p_i = \nu_i(x)$ (respectively, $p_L = \nu_L(x)$, $p_M = \nu_M(x)$, and $p_R = \nu_R(x)$) the corresponding position of x under ν_i (respectively, ν_L, ν_M, and ν_R).

We now introduce the notion of kernel, which plays a major role in the development of our proofs below. Consider any linear arrangement ν of G' and any of the grids G_i. The **kernel** $K_i^{(\nu)}$ relative to ν and G_i is a set of positions p of the nodes of G' under ν such that $\left(2n^4+1\right)^2 \leq p_i \leq \left(6n^4\right)\left(2n^4+1\right) - \left(2n^4+1\right)^2$. Corollary 1 implies that for any $p \in K_i^{(\nu)}$, $\mathbf{wd}(G_i, \nu_i, p_i) \geq 2n^4 + 1$.

Similarly, we define the kernel $K_L^{(\nu)}$ (respectively, $K_R^{(\nu)}$) as the set of positions p of the nodes of G' under ν such that $\left(3n^4+1\right)^2 \leq p_L, p_R \leq \left(12n^4\right)\left(3n^4+1\right) - \left(3n^4+1\right)^2$. Again, Corollary 1 implies that for any $p \in K_L^{(\nu)}$ (respectively, $p \in K_R^{(\nu)}$), we have $\mathbf{wd}(L, \nu_L, p_L) \geq 3n^4 + 1$ (respectively, $\mathbf{wd}(R, \nu_R, p_R) \geq 3n^4 + 1$). We define the kernel $K_M^{(\nu)}$ as the set of positions p

of the nodes of G' under ν such that $\left(2n^4+1\right)^2 \le p_M \le \left(2n^4+1\right)\left(8n^4+1\right) - \left(2n^4+1\right)^2$. Corollary 1 implies that for any $p \in K_M^{(\nu)}$ we have $\mathbf{wd}(M, \nu_M, p_M) \ge 2n^4+1$.

Observe that, for any $i \in [n]$, we have $|K_i^{(\nu)}| = (2n^4+1)(6n^4) - 2((2n^4+1)^2)+1 \ge 3n^8$ for n sufficiently large. Furthermore, for n sufficiently large, we have $|K_L^{(\nu)}| = |K_R^{(\nu)}| = (3n^4+1)(12n^4) - 2((3n^4+1)^2)+1 \ge 17n^8$, and $|K_M^{(\nu)}| = (2n^4+1) \times (8n^4+1) - 2(2n^4+1)^2 + 1 \ge 7n^8$.

Recall that in our construction in Section 3.2 we have set $k' = 3n^4 + 2n^3 + 2k + 2$. From now on, we denote by ν' any linear arrangement of G' having maximum width at most $k' + n^2$. For any two sets of positive integers A and B, we will write $A < B$ if each element of A is smaller than every element in B.

Lemma 4. *Let ν' be a linear arrangement of G' having maximum width at most $k'+n^2$, and let $\mathcal{K}^{(\nu')} = \{K_L^{(\nu')}, K_M^{(\nu')}, K_R^{(\nu')}\} \cup \{K_i^{(\nu')} : i \in [n]\}$. For any pair of kernels $K', K'' \in \mathcal{K}^{(\nu')}$ with $K' \ne K''$, either $K' < K''$ or $K'' < K'$.*

Proof. We first consider the kernels in $\{K_i^{(\nu')} : i \in [n]\}$. Let $p, p' \in K_i^{(\nu')}$ be two positions such that $p_i = p'_i - 1$. Assume that there exists a position $q \in K_j^{(\nu')}$, $j \ne i$, such that $p < q < p'$. We know (by Corollary 1 and definition of kernel) that $\mathbf{wd}(G_i, \nu'_i, p_i) \ge 2n^4+1$ and $\mathbf{wd}(G_j, \nu'_j, q_j) \ge 2n^4+1$. Since G_i and G_j have disjoint edge sets, and since in between p and q there is no position associated with a node from G_i, we conclude that $\mathbf{wd}(G', \nu', q) \ge 4n^4 + 2 > k' + n^2$, for n sufficiently large. This is in contrast with our assumption about the linear arrangement ν'.

Essentially the same argument can be used when we consider all of the kernels in $\mathcal{K}^{(\nu')}$. □

Intuitively, the above lemma states that in any linear arrangement ν' of G' with maximum width at most $k'+n^2$, the kernels of the grid components of G' cannot overlap one with the other. As a consequence, ν' induces an ordering of the nodes of the source graph G which is determined by the positions of the corresponding kernels.

Lemma 5. *Let ν' be a linear arrangement of G' having maximum width at most $k'+n^2$. Then either $K_L^{(\nu')} < K_M^{(\nu')} < K_R^{(\nu')}$ or $K_R^{(\nu')} < K_M^{(\nu')} < K_L^{(\nu')}$.*

Proof. Assuming $K_L^{(\nu')} < K_R^{(\nu')}$, we show below that, under ν', kernel $K_R^{(\nu')}$ cannot be placed in between kernels $K_L^{(\nu')}$ and $K_M^{(\nu')}$. Essentially the same argument can be used to show that kernel $K_L^{(\nu')}$ cannot be placed in between kernels $K_M^{(\nu')}$ and $K_R^{(\nu')}$.

Assume that we have $K_L^{(\nu')} < K_R^{(\nu')} < K_M^{(\nu')}$. Since the number of nodes of L which lie to the left of $K_R^{(\nu')}$ is at least equal to $17n^8$, and since at most $n^4(12n^4)$ nodes of L can belong to its last n^4 rows, we have that at least $5n^8$ nodes of the first $2n^4+1$ rows of L lie to the left of $K_R^{(\nu')}$. On the other hand, since at least $7n^8$ nodes of M belong to $K_M^{(\nu')}$, we have that at least $7n^8$ nodes of M lie to the right of $K_R^{(\nu')}$.

Let us now consider the grid $X = \Gamma[2n^4+1, 12n^4+8n^4+1]$ composed by the $(2n^4+1)$ upper rows of L and all of the rows of M. We apply Lemma 3 to X. If we define V_1 (respectively, V_2) as the set of nodes of X contained in $K_L^{(\nu')}$ (respectively, $K_M^{(\nu')}$), we have that both $|V_1|$ and $|V_2|$ are greater than $(2n^4+1)^2$. Then we have that at least $2n^4+1$ edges internal to X cross over all positions (gaps) of $K_R^{(\nu')}$. From the definition of kernels, there are at least $3n^4+1$ edges internal to $K_R^{(\nu')}$ crossing over each position of $K_R^{(\nu')}$. Adding these together, we have at least $5n^4+2$ edges at each position of $K_R^{(\nu')}$, which is greater than $k'+n^2$ (for n sufficiently large).

The case of $K_R^{(\nu')} < K_L^{(\nu')}$ can be dealt with in a very similar way and the lemma thus follows. □

In the following, without loss of generality, we will always assume that $K_L^{(\nu')} < K_M^{(\nu')} < K_R^{(\nu')}$. By applying essentially the same argument from the proof of Lemma 5, we can show that $K_i^{(\nu')}$ cannot lie between $K_L^{(\nu')}$ and $K_M^{(\nu')}$ or between $K_M^{(\nu')}$ and $K_R^{(\nu')}$, which implies the following result.

Lemma 6. *Let ν' be a linear arrangement of G' having maximum width at most $k'+n^2$. For any $i \in [n]$, either $K_i^{(\nu')} < K_L^{(\nu')}$ or $K_i^{(\nu')} > K_R^{(\nu')}$.*

So far we have seen that kernels always appear in some total order in the linear arrangements we are interested in, and with the kernels of grids L, M and R all in a row. We move on now with a second property of the family of linear arrangements we are looking at. As already described above, this property states that, if our goal is the one of optimizing the width at certain gaps, then misplacing nodes that are not in a kernel does not result in any gain. We first provide two results about general grids, and then come back to G' and our linear arrangements.

Lemma 7. *Let $X = \Gamma[H, W]$ and let S be a set of nodes of X such that $|S| \le W(H - e - 2)$ with $e \ge 0$ and there exists w with $1 \le w \le W$ such that, for any h with $1 \le h \le H$, $x^{h,w} \in S$ (in other words S contains an entire column of the grid). Then, $\delta(S, \overline{S})$ contains at least $e + 2$ edges in distinct rows, where $\overline{S}$ denotes the set of nodes of the grid which do not belong to S.*

Proof. For each h with $1 \le h \le H$, either there exists w with $1 \le w < W$ such that $(x^{h,w} \in S \wedge x^{h,w+1} \in \overline{S}) \vee (x^{h,w} \in \overline{S} \wedge x^{h,w+1} \in S)$ (in this case, the row contributes at least by one horizontal edge to $\delta(S, \overline{S})$), or, for any w with $1 \le w \le W$, $x^{h,w} \in S$. This latter case, however, can happen at most $\left\lfloor \frac{|S|}{W} \right\rfloor$ times. Since $|S| \le W(H - e - 2)$, we have that the first case happens at least $e + 2$ times, thus proving the lemma. $\square$

Lemma 8. *Let $X = \Gamma[H, W]$ and let S be a set of nodes of X such that $|S| \le W(H - |F| - 2)$, where F is a subset of the set of nodes of the first row or of the last row which belong to S. Then, $\delta(S, \overline{S})$ contains at least $|F|$ edges not included in the first row or in the last row.*

Proof. For each w with $1 \le w \le W$ such that $x^{1,w} \in F \vee x^{H,w} \in F$, either there exists h with $1 \le h < H$ such that $(x^{h,w} \in S \wedge x^{h+1,w} \in \overline{S}) \vee (x^{h,w} \in \overline{S} \wedge x^{h+1,w} \in S)$ (in this case, the column contributes at least by one vertical edge to $\delta(S, \overline{S})$), or, for any h with $1 \le h \le H$, $x^{h,w} \in S$ (that is, S includes the entire w-th column). If this latter case happens at least once, then we can apply the previous lemma with $e = |F|$, thus obtaining that $\delta(S, \overline{S})$ contains at least $|F| + 2$ horizontal edges on distinct rows, which implies that $\delta(S, \overline{S})$ contains at least $|F|$ edges not included in the first row or in the last row. Otherwise, $\delta(S, \overline{S})$ contains at least $|F|$ vertical edges: indeed, if $x^{1,w} \in S$ and $x^{H,w} \notin S$ or vice versa, then at least one vertical edge of the w-th column is in $\delta(S, \overline{S})$, otherwise at least two vertical edges of this column are in $\delta(S, \overline{S})$ (since, in this case, we have both to exit from S and to enter again in S). $\square$

We need to introduce some additional notation. From now on, we denote by l^* the first gap from left to right occurring between two vertices of $K_L^{(\nu')}$, and we denote by r^* the first gap from left to right occurring between two vertices of $K_R^{(\nu')}$. For any $i \in [n]$, we define the value α_i as follows. If $K_i^{(\nu')} > K_L^{(\nu')}$, α_i is the number of nodes of the first or of the last row of G_i whose position under ν' is smaller than l^* and which are endpoints of an edge exiting G_i. Otherwise, α_i is the number of nodes of the first or of the last row of G_i whose position is greater than l^* and which are endpoints of an edge exiting G_i. Similarly, we denote by α_M the number of nodes of the first row of M whose position is smaller than l^* and which are endpoints of an edge exiting M.

Lemma 9. *For any linear arrangement ν' of G' having maximum width at most $k' + n^2$ and for any $i \in [n]$, there exist at least α_i distinct edges within G_i which cross over l^*.*

Proof. First observe that $\alpha_i \le 4n^2 + 6$ because there are $4n^2$ edges connecting G_i to M and 6 edges connecting G_i to other grids G_j. We only study the case in which $K_i^{(\nu')} > K_L^{(\nu')}$, since the other case can be proved in the same way.

Let V_i be the vertex set of G_i. Let $\mathcal{P}(G_i) = \{p : (\nu')^{-1}(p) \in V_i\}$ and let $S_i = \{p : (\nu')^{-1}(p) \in V_i \wedge p < l^*\}$ (clearly, $|S_i| \ge \alpha_i$). Since $|\mathcal{P}(G_i)| = 12n^8 + 6n^4 \le 13n^8$ and $|K_i^{(\nu')}| \ge 3n^8$, and since S_i is a subset of $\mathcal{P}(G_i) - K_i^{(\nu')}$, we are guaranteed that $|S_i| \le 10n^8$. Because $10n^8 \le (6n^4)(2n^4 + 1 - \alpha_i - 2)$ (assuming $n \ge 4$) we have the precondition $|S_i| \le (6n^4)(2n^4 + 1 - \alpha_i - 2)$ that we need in order to apply Lemma 8. Lemma 8 implies that there exist at least $|\alpha_i|$ distinct edges connecting S_i to $\overline{S_i}$ (that is, edges within G_i): these edges clearly cross over l^*. $\square$

Similarly, we can prove the following result.

Lemma 10. *For any linear arrangement ν' of G' having maximum width at most $k' + n^2$, there exist at least α_M distinct edges within M which cross over l^*.*

From now on, let $\kappa_l^{(\nu')}$ be the number of kernels $K_i^{(\nu')}$ such that $K_i^{(\nu')} < K_L^{(\nu')}$ and let $\kappa_r^{(\nu')}$ be the number of kernels $K_i^{(\nu')}$ such that $K_i^{(\nu')} > K_R^{(\nu')}$. Let also $\tau^{(\nu')}$ denote the number of edges (v_i, v_j) in G such that $K_i^{(\nu')} < K_L^{(\nu')}$ and $K_j^{(\nu')} > K_L^{(\nu')}$.

Lemma 11. *Let ν' be a linear arrangement of G' having maximum width at most $k' + n^2$. There exist at least $\kappa_l^{(\nu')} \cdot (4n^2) + 2\tau^{(\nu')}$ distinct edges which cross over l^*, not including edges internal to L or R.*

Proof. We define $I_L^{(\nu')}$ as the set of integers $i \in [n]$ such that $K_i^{(\nu')} < K_L^{(\nu')}$. Thus we have $|I_L^{(\nu')}| = \kappa_l^{(\nu')}$. We also define $J_L^{(\nu')}$ as the set of integers $j \in [n]$ such that $K_j^{(\nu')} > K_L^{(\nu')}$ and there exists $i \in I_L^{(\nu')}$ with $(v_i, v_j) \in E$, where E is the set of edges of G.

For each $i \in I_L^{(\nu')}$, let us consider the $4n^2$ distinct edges connecting G_i to M, along with each pair of edges connecting G_i to each grid G_j such that $K_j^{(\nu')} > K_L^{(\nu')}$ and $(v_i, v_j) \in E$. Let also $\mathcal{E}$ be the set of all these edges, for every $i \in I_L^{(\nu')}$. Thus we have $|\mathcal{E}| = \kappa_l \cdot (4n^2) + 2\tau^{(\nu')}$.

Consider now an arbitrary edge $e \in \mathcal{E}$. Let x be one of the two endpoints of e, and assume that x belongs to some grid X among the $n+3$ grid components of G'. We say that x is **misplaced** if, under ν', the kernel of X is placed at some side with respect to l^* and x is placed at the opposite side. From the definition of $\mathcal{E}$, it is easy to see that if none of the endpoints of e are misplaced, or else if both of the endpoints of e are misplaced, then e must cross over l^*. On the other hand, if exactly one of the endpoints of e is misplaced, then e does not cross over l^*.

Consider then the set of all the misplaced endpoints of some edge in $\mathcal{E}$. By construction of G', these endpoints are distinct and belong to the first row or to the last row of some grid component of G'. Furthermore, the edges in $\mathcal{E}$ are all single rather than multiple edges, as already observed in Section 3.2. By definition of α_M and α_i, $i \in [n]$, we have that $\alpha_M + \sum_{i \in (I_L^{(\nu')} \cup J_L^{(\nu')})} \alpha_i$ is greater than or equal to the number of all the misplaced endpoints of some edge in $\mathcal{E}$, and from the above observations we have that the latter number is in turn greater than or equal to the number of edges in $\mathcal{E}$ which do not cross over l^*. By Lemmas 9 and 10, it follows that, among the edges within M and among the edges within the components G_i, $i \in (I_L^{(\nu')} \cup J_L^{(\nu')})$, there exist $\alpha_M + \sum_{i \in (I_L^{(\nu')} \cup J_L^{(\nu')})} \alpha_i$ distinct edges which cross over l^*. This quantity plus the number of edges in $\mathcal{E}$ which cross over l^* gives us the desired result. □

We are now ready to show the inverse relation of the statement in Lemma 2. In what follows we focus our attention on linear arrangements of G' having maximum width at most k'. The reason why all of the previous lemmas in this section have been stated for linear arrangements with maximum width at most $k' + n^2$ is because in Section 4 we need to refer to this extended class.

Lemma 12. *If $\langle G', k' \rangle$ is a positive instance of* PMCW *then $\langle G, k \rangle$ is a positive instance of* MBW.

Proof. Let ν' be a linear arrangement of G' having maximum width bounded by $k' = 3n^4 + 2n^3 + 2k + 2$, and consider quantity $\mathbf{wd}(G', \nu', l^*)$. From Lemma 11 there are at least $\kappa_l^{(\nu')} \cdot (4n^2) + 2\tau^{(\nu')}$ distinct edges which cross over l^*, not including edges internal to the grids L or R. In addition, recall that there are at least $3n^4 + 1$ edges internal to L that are crossing over l^*. This is because of Corollary 1 and because of the way we have defined kernels. If $\kappa_l^{(\nu')} > \frac{n}{2}$, the number of edges contributing to $\mathbf{wd}(G', \nu', l^*)$ would be at least $3n^4 + 1 + (\frac{n}{2} + 1) \cdot 4n^2 = 3n^4 + 2n^3 + 4n^2 + 1 > 3n^4 + 2n^3 + 2k + 2 = k'$, for sufficiently large values of n, where the inequality follows from the fact that k is bounded by the number of edges in G, which is $\frac{3n}{2}$. This is against our assumptions on ν'. Thus we must conclude that $\kappa_l^{(\nu')} \le \frac{n}{2}$. Similarly, we can prove that $\kappa_r^{(\nu')}$ cannot be greater than $\frac{n}{2}$. Hence, we have that $\kappa_l^{(\nu')} = \kappa_r^{(\nu')} = \frac{n}{2}$.

Using the above fact in Lemma 11, we have that the number of edges external to L and R crossing over l^* is at least $2n^3 + 2\tau^{(\nu')}$. Including the edges internal to L gives at least $3n^4 + 2n^3 + 2\tau^{(\nu')} + 1$ edges crossing over l^*. Since the width of l^* is at most $3n^4 + 2n^3 + 2k + 2$, it also follows that $\tau^{(\nu')} \le k + \frac{1}{2}$. This means that the number of edges (v_i, v_j) in G such that $K_i^{(\nu')} < K_L^{(\nu')}$ and $K_j^{(\nu')} > K_L^{(\nu')}$ is at most k. Hence, by partitioning the nodes of G according to the position of their corresponding kernels under ν', we have an equal size subset partition whose cut is at most k. □

3.5. Cutwidth and internal boundaries

We can now present the main results of Section 3.

Theorem 1. *The problem* PMCW *is NP-complete.*

Proof. Let G be a cubic graph with $n > 1$ vertices, and let $k > 0$ be some integer. From Lemma 2 and from Lemma 12, we have that $\langle G, k \rangle$ is a positive instance of MBW if and only if $\langle G', k' \rangle$ is a positive instance of PMCW. This relation shows that an algorithm for PMCW could be used to solve MBW, and thus PMCW is NP-hard.

To conclude the proof, we observe that the problem PMCW is in NP because a linear arrangement of a graph can be guessed in polynomial time and its maximum width can be computed in polynomial time as well. □

We can now deal with a decision problem associated with the problem of finding a linear parsing strategy for a synchronous rule that minimizes the number of internal boundaries, defined in (3).

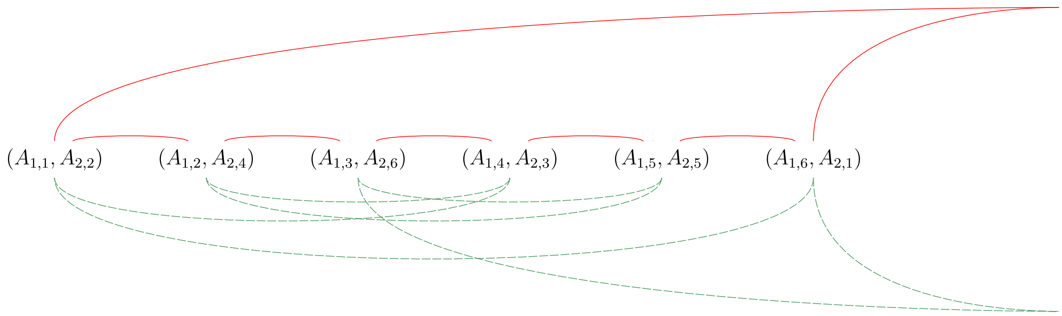

Fig. 9. Extended edges for the permutation multigraph corresponding to the SCFG rule s of Eq. (1).

Theorem 2. *Let s be a synchronous rule with r nonterminals and with associated permutation π_s, and let k be some positive integer. The problem of deciding whether*

$$\min_{\sigma} \max_{i \in [r]} \mathbf{ib}(\pi_s, \sigma, i) \le k$$

is NP-complete.

Proof. We have already observed that the relation $\mathbf{cw}(G_s) = \min_\sigma \max_{i \in [r]} \mathbf{ib}(\pi_s, \sigma, k)$ directly follows from Lemma 1. The statement then follows from Theorem 1. $\square$

4. Relating permutation multigraphs to SCFGs

As discussed in Section 2, our main goal is finding efficient ways of parsing synchronous context-free rules. In this section, we use our results on permutation multigraphs to prove NP-hardness for optimizing both space complexity and time complexity of linear parsing strategies for SCFG rules. We begin by examining space complexity, and then generalize the argument to prove our result on time complexity.

4.1. Space complexity

Optimizing the space complexity of a parsing strategy is equivalent to minimizing the maximum that the fan-out function achieves across the steps of the parsing strategy (6). According to the definition of the fan-out function (5), fan-out consists of the two terms **ib** and **eb**, accounting for the internal and the external boundaries, respectively, realized at a given step by a linear parsing strategy. Let s be a synchronous rule with G_s the associated permutation multigraph. We have already seen in Lemma 1 a relation between the **ib** term and the width function for G_s. In order to make precise the equivalence between the fan-out problem and the cutwidth problem for G_s, we must now account for the **eb** term.

Let b_R and e_R be the first and the last vertices of the red path in G_s, and let b_G and e_G be the first and the last vertices of the green path. We collectively refer to these vertices as the **endpoints** of G_s, and we define $V_e = \{b_R, e_R, b_G, e_G\}$. Let σ_s be some linear parsing strategy for rule s, and let ν_s be the corresponding linear arrangement for G_s, as defined in Section 3.1. Informally, we observe that under σ_s the number of external boundaries at a given step i is the number of vertices from the set V_e that have been seen to the left of the current position i under ν_s (including i itself). Using definition (5) and Lemma 1, this suggests that we can represent the fan-out at i as the width of an *augmented* permutation multigraph containing special edges from the vertices in V_e, where the special edges always extend past the right end of any linear arrangement. We introduce below some mathematical definitions that formalize this idea.

Assume G_s has n nodes and set of edges E. We define the **extended width** at position $i \in [n] - 1$ to be

$$\mathbf{ewd}(G_s, \nu_s, i) = \mathbf{wd}(G_s, \nu_s, i) + \sum_{v \in V_e} I(\nu_s(v) \le i) \,. \tag{9}$$

The contribution of the endpoints to the extended width can be visualized as counting, at each position in the linear arrangement, an additional set of edges running from the endpoint of the red and green paths all the way to the right end of the linear arrangement, as shown in Fig. 9. We will refer to these additional edges as **extended edges**. To simplify the notation below, we also let $\mathbf{ewd}(G_s, \nu_s, n) = 4$.

Let π_s be the permutation associated with synchronous rule s. Observe that the first term in (9) corresponds to the number of internal boundaries $\mathbf{ib}(\pi_s, \sigma_s, i)$, by Lemma 1, and the second term counts the number of external boundaries $\mathbf{eb}(\pi_s, \sigma_s, i)$. We can then write, for each $i \in [n]$

$$\mathbf{fo}(\pi_s, \sigma_s, i) = \frac{1}{2}\mathbf{ewd}(G_s, \nu_s, i) \,, \tag{10}$$

which will be used below to assess the complexity of the fan-out problem. Finally, we define the **extended cutwidth** of G_s as

$$\mathbf{ecw}(G_s) = \min_{\nu} \max_{i \in [n]} \mathbf{ewd}(G_s, \nu, i) . \tag{11}$$

From (10) and (11) we see that the extended cutwidth of G_s is related to the optimal computational complexity that we can achieve when parsing synchronous rule s with the techniques described in Section 2.2. With such motivation, we investigate below a decision problem related to the computation of the extended cutwidth of a permutation multigraph.

From now on, we assume that $\langle G, k\rangle$ is an instance of MBW, where G is a cubic graph with n vertices. We also assume that G' and k' are constructed from G and k as in Section 3.2.

Lemma 13. *If $\langle G, k\rangle$ is a positive instance of MBW, then $\mathbf{ecw}(G') \leq k' + 2$.*

Proof. Under the assumption that $\langle G, k\rangle$ is a positive instance of MBW, consider the linear arrangement ν used in the proof of Lemma 2 to show that $\langle G', k'\rangle$ is a positive instance of PMCW. We already know that the maximum (regular) width of ν is at most k'. With the exception of the first column of grid L and the last column of grid R, the extended width under ν at positions within L and R is two greater than the (regular) width, because the vertices b_R and b_G are both to the left, while e_R and e_G are to the right. For positions in the first column of L, the extended width is one greater than the width, because only b_G is to the left, as depicted in Fig. 8.

The critical point is the last column of R. We observe that the edges connecting vertices in the grid components G_i and M contribute to the extended width at positions within R always in the same amount. We thus focus our analysis on the only edges that are internal to R. Recall that e_R is the topmost vertex in the last column of R. We let i be the position of e_R under ν. At position $i-1$ the contribution to the extended width of the edges internal to R consists of $3n^4+1$ red edges and one green edge; see again Fig. 8. At the next position i, one red edge and one green edge internal to R are lost. However, these two edges are replaced by one new red edge from R and one new extended edge impinging on vertex e_R. Thus the extended width at positions $i-1$ and i must be the same. For all of the next positions corresponding to vertices in the last column of R, the contribution to the extended width of the edges internal to R always decreases.

From the above observations, we conclude that the extended width at positions within grid components L and R is bounded by $k'+2 = 3n^4 + 2n^3 + 2k + 4$. As already observed in the proof of Lemma 2, the width at all of the remaining positions for ν is lower by $n^4 + \mathcal{O}(n^3)$, and this must also be the case for the extended width, since this quantity exceeds the (regular) width by at most four. The existence of linear arrangement ν thus implies $\mathbf{ecw}(G') \leq k' + 2$. □

Let ν' be a linear arrangement of G' having maximum width at most $k' + n^2$. Then ν' satisfies the hypotheses of all of the lemmas in Section 3.4 constraining the linear arrangement of the kernels of the grid components of G'. However, contrary to the case of the regular cutwidth, the extended cutwidth is not invariant to a reversal of a linear arrangement. This is so because the extended edges always end up at a position to the right of the right end of any linear arrangement. For this reason we can no longer assume that in ν' we have $L < R$. Let then X_L and X_R be the leftmost and the rightmost, respectively, of L and R under ν'. Let also e^* be the rightmost of l^* and r^*.

We already know from the first part of the proof of Lemma 12 that the number of kernels to the left of $K^{(\nu')}_{X_L}$ is $\frac{n}{2}$, and this is also the number of kernels to the right of $K^{(\nu')}_{X_R}$. Using this fact, the following result can easily be shown using the same argument presented in the proof of Lemma 11. As in Section 3.4, let $\tau^{(\nu')}$ denote the number of edges (v_i, v_j) in G such that $K^{(\nu')}_i < K^{(\nu')}_{X_L}$ and $K^{(\nu')}_j > K^{(\nu')}_{X_R}$.

Lemma 14. *Let ν' be a linear arrangement of G' having maximum width at most $k' + n^2$. There are at least $\frac{n}{2} \cdot (4n^2) + 2\tau^{(\nu')}$ distinct edges which cross over e^*, not including edges internal to the grids L or R.*

The proof of the next lemma uses arguments very similar to those already exploited in the proof of Lemma 9 and in the proof of Lemma 12.

Lemma 15. *If $\mathbf{ecw}(G') \leq k' + 2$, then $\langle G, k\rangle$ is a positive instance of MBW.*

Proof. Let ν' be a linear arrangement of G' having maximum extended width bounded by $k' + 2 = 3n^4 + 2n^3 + 2k + 4$. Since the maximum (regular) width of G' is at most its maximum extended width, ν' satisfies the hypotheses of all of the lemmas in Section 3.4 constraining the arrangement of the kernels and the misplaced nodes from the grid components of G'.

From Lemma 14 there are at least $2n^3 + 2\tau^{(\nu')}$ distinct edges which cross over e^*, not including edges internal to the grids L or R. In addition, there are at least $3n^4 + 1$ edges internal to X_R that are crossing over e^*. Finally, consider the two endpoints of either the red or green path appearing in the first and in the last lines of X_L, and let α_e be the number of such endpoints that have been misplaced to the right of e^* under ν'. Note that we have $0 \leq \alpha_e \leq 2$.

Let $\mathcal{P}(X_L) = \{p : (\nu')^{-1}(p) \in X_L\}$ and let $S_{X_L} = \{p : (\nu')^{-1}(p) \in X_L \wedge p > e^*\}$. Because $|\mathcal{P}(X_L)| = 36n^8 + 12n^4$ and $|K_{X_L}| \geq 17n^8$, we know that $|S_{X_L}| \leq |\mathcal{P}(X_L) - K_{L'}| \leq 20n^8$. Therefore $|S_{X_L}| \leq (12n^4)(3n^4 + 1 - \alpha_e - 2)$ and we can apply Lemma 8 with $X = X_L$ and $S = S_{X_L}$. From Lemma 8, there are at least α_e edges internal to X_L that cross over e^*. Along with the $2 - \alpha_e$ extended edges departing from X_L, this accounts for two additional edges that cross over e^*.

Combining all of the above contributions, the total number of edges crossing over e^* is at least $3n^4 + 2n^3 + 2\tau^{(\nu')} + 3$. Since the extended width of e^* is at most $3n^4 + 2n^3 + 2k + 4$, it also follows that $\tau^{(\nu')} \leq k + \frac{1}{2}$. This means that the number of edges (v_i, v_j) in G such that $K_i^{(\nu')} < K_L^{(\nu')}$ and $K_j^{(\nu')} > K_L^{(\nu')}$ is at most k. Hence, by partitioning the nodes of G according to the position of their corresponding kernels under ν', we have an equal size subset partition whose cut is at most k. □

Theorem 3. *Let G be a permutation multigraph and let k be a positive integer. The problem of deciding whether* $\mathbf{ecw}(G) \leq k$ *is NP-complete.*

Proof. By Lemmas 13 and 15, an algorithm that decides whether $\mathbf{ecw}(G) \leq k$ can be used to solve MBW. Thus the problem in the statement of the theorem is NP-hard. The problem is in NP because the maximum extended width of a linear arrangement can be verified in polynomial time. □

Theorem 4. *Let s be a synchronous rule with r nonterminals and with associated permutation π_s, and let k be a positive integer. The problem of deciding whether*

$$\min_{\sigma} \max_{i \in [n]} \mathbf{fo}(\pi_s, \sigma, i) \leq k$$

is NP-complete, where σ ranges over all linear parsing strategies for s.

Proof. The NP-hardness part directly follows from Theorem 3, along with (10) and (11). The problem is also in NP, because the maximum value of the fan-out for a guessed linear parsing strategy can be computed in polynomial time. □

We have already discussed how fan-out is directly related to the space complexity of the implementation of a linear parsing strategy. From Theorem 4 we then conclude that optimization of the space complexity of linear parsing for SCFGs is NP-hard.

4.2. Time complexity

We now turn to the optimization of the time complexity of linear parsing for SCFGs. It turns out that at each step of a linear parsing strategy, the time complexity is related to a variant of the notion of width, called modified width, computed for the corresponding position of a permutation graph. With this motivation, we investigate below the modified width and some extensions of this notion, and we derive our main result in a way which parallels what we have already done in Section 4.1.

We start with some additional notation. Let $G = (V, E)$ be an undirected (multi)graph such that $|V| = n > 1$, and let ν be some linear arrangement of G. For any $i \in [n]$, the **modified width** of G at i with respect to ν, written $\mathbf{mwd}(G, \nu, i)$, is defined as $|\{(u, v) \in E : \nu(u) < i < \nu(v)\}|$. Informally, $\mathbf{mwd}(G, \nu, i)$ is the number of distinct edges crossing over the vertex at position i-th in the linear arrangement ν. Again in case of multigraphs the size of the previous set should be computed taking into account multiple edge occurrences. The following result is a corollary to Lemma 3.

Corollary 2. *For any grid $X = \Gamma[H, W]$ with $W \geq 2H + 1$, for any linear arrangement ν of X, and for any i with $H^2 < i \leq HW - H^2$,* $\mathbf{mwd}(X, \nu, i) \geq H - 2$.

Proof. Modified width at the vertex in position i can be related to the (regular) width of the gaps before and after position i, and the degree $\Delta(\nu^{-1}(i))$ of the vertex at position i

$$\mathbf{mwd}(X, \nu, i) = \frac{1}{2}(\mathbf{wd}(X, \nu, i - 1) + \mathbf{wd}(X, \nu, i) - \Delta(\nu^{-1}(i)))\,.$$

For each i with $H^2 < i \leq HW - H^2$, we can use Corollary 1 and write

$$\mathbf{mwd}(X, \nu, i) \geq H - \frac{1}{2}\Delta(\nu^{-1}(i))\,.$$

Because vertices in a grid have degree at most four, we have $\mathbf{mwd}(X, \nu, i) \geq H - 2$. □

Let s be a synchronous rule and let G_s be the associated permutation multigraph. Let also ν_s be a linear arrangement for G_s. We define the **extended modified width** at $i \in [n]$ as

$$\mathbf{emwd}(G_s, \nu_s, i) = \mathbf{wd}(G_s, \nu_s, i) + \sum_{v \in V_e} I(\nu_s(v) \leq i) \, . \tag{12}$$

Again, the contribution of the endpoints of G_s to the extended modified width can be visualized as counting, at each position in the linear arrangement, an additional set of edges running from the endpoint of the red and green paths all the way to the right end of the linear arrangement, as shown in Fig. 9. The **extended modified cutwidth** of G_s is

$$\mathbf{emcw}(G_s) = \min_{\nu} \max_{i \in [n]} \mathbf{emwd}(G_s, \nu, i) \, ,$$

where ν ranges over all possible linear arrangements for G_s.

From now on, we assume that $\langle G, k \rangle$ is an instance of MBW, where G is a cubic graph with n vertices. We also assume that G' and k' are constructed from G and k as in Section 3.2.

Lemma 16. *If $\langle G, k \rangle$ is a positive instance of MBW, then $\mathbf{emcw}(G') \leq k'$.*

Proof. Consider the linear arrangement used in Lemma 13 to show that $\mathbf{ecw}(G') \leq k' + 2$. The critical points in that linear arrangement are all within the L and R components. At all positions in these components, each vertex has two edges extending to its right and two edges extending to its left (one red and one green in each case). Thus, at these positions, the extended modified cutwidth is less than the extended cutwidth by two. □

Lemma 17. *If $\mathbf{emcw}(G') \leq k'$, then $\langle G, k \rangle$ is a positive instance of MBW.*

Proof. Assume a linear arrangement ν' for G' having maximum extended modified width bounded by k'. The maximum (non-extended) modified width of G' must be smaller than or equal to k', and the maximum (non-extended, non-modified) width of G' must be smaller than or equal to $k' + 4$, because the maximum degree of vertices in G' is four. Since the maximum width of G' under ν' is bounded by $k' + n^2$, we apply Lemma 14 and conclude that there are at least $2n^3 + 2\tau^{(\nu')}$ distinct edges which cross over the vertex to the left of the gap e^*, not including edges internal to the grids L or R.

By Corollary 2, the number of edges internal to X_R which cross over the vertex to the left of the gap e^* is at least $3n^4 - 1$. Finally, consider the two endpoints of either the red or green path appearing in the first and in the last lines of X_L, and let α_e be the number of such endpoints that have been misplaced to the right of e^* under ν', with $0 \leq \alpha_e \leq 2$. We apply the same argument as in the proof of Lemma 15, and conclude that there are at least α_e edges internal to X_L that cross over the vertex to the left of the gap e^*. Along with the $2 - \alpha_e$ extended edges departing from X_L, this accounts for two additional edges that cross over the vertex to the left of the gap e^*.

Combining all of the above contributions, the total number of edges crossing over the vertex to the left of the gap e^* is at least $3n^4 + 2n^3 + 2\tau^{(\nu')} + 1$. Since the extended modified width of the vertex to the left of the gap e^* is at most $k' = 3n^4 + 2n^3 + 2k + 2$, it also follows that $\tau^{(\nu')} \leq k + \frac{1}{2}$. This means that the number of edges (v_i, v_j) in G such that $K_i^{(\nu')} < K_{X_L}^{(\nu')}$ and $K_j^{(\nu')} > K_{X_R}^{(\nu')}$ is at most k. Hence, by partitioning the nodes of G according to the position of their corresponding kernels under ν', we have an equal size subset partition whose cut is at most k. □

Theorem 5. *Let G be a permutation multigraph and let k be a positive integer. The problem of deciding whether $\mathbf{emcw}(G) \leq k$ is NP-complete.*

Proof. By Lemmas 17 and 16, an algorithm that decides whether $\mathbf{emcw}(G) \leq k$ can be used to solve MBW. Thus deciding whether $\mathbf{emcw}(G) \leq k$ is NP-hard. The problem is in NP because the maximum extended modified width of a linear arrangement can be verified in polynomial time. □

We now relate the notion of extended modified width for a permutation multigraph and the time complexity of a parsing algorithm using a linear strategy. Let s be a synchronous rule with r nonterminals having the form in (8), and let G_s be the permutation multigraph associated with s. Let also σ_s be some linear parsing strategy defined for s, and let ν_s be the linear arrangement associated with σ_s, as defined in Section 3.1. Recall from Section 2.2 that the family of parsing algorithms we investigate in this article use parsing states to represent the boundaries (internal and external) that delimit the substrings of the input that have been parsed at some step, following our strategy σ_s.

For some i with $i \in [r-1]$, let us consider some parsing state with state type (s, σ_s, i). In the next parsing step $i+1$, we move to a new state with state type $(s, \sigma_s, i+1)$ by adding to our partial analyses the $(i+1)$-th pair of nonterminals from the right-hand side of s, defined according to σ_s. As already observed in Section 2.2, this operation involves some updates to the sequence of boundaries of our old state of type (s, σ_s, i). More precisely, the new state is constructed from the old state by removing a number $\delta_{i+1}^{(-)}$ of boundaries, and by adding a number $\delta_{i+1}^{(+)}$ of new boundaries. From the definition of G_s, we know that $\delta^{(-)}_{i+1}$ is the number of backward edges at vertex $i+1$, and $\delta^{(+)}_{i+1}$ is the number of forward edges at vertex $i+1$, where backward, forward, and vertex $i+1$ are all defined relative to the linear arrangement ν_s and include the extended edges. We also have $\delta^{(-)}_{i+1} + \delta^{(+)}_{i+1} = \Delta(\nu_s^{-1}(i+1))$, where $\Delta(\nu^{-1}(i+1))$ is the degree of the vertex at position $i+1$.

The total number of boundaries t_{i+1} involved in the parsing step $i+1$ is then the number of boundaries for state type (s, σ_s, i), which includes the $\delta^{(-)}_{i+1}$ boundaries that need to be removed at such step, plus the new boundaries $\delta^{(+)}_{i+1}$. We already know that $\mathbf{ewd}(G_s, \nu_s, i)$ is the number of boundaries for (s, σ_s, i). We can then write

$$\begin{aligned} t_{i+1} &= \mathbf{ewd}(G_s, \nu_s, i) + \delta^{(+)}_{i+1} \\ &= \mathbf{ewd}(G_s, \nu_s, i) - \delta^{(-)}_{i+1} + \delta^{(-)}_{i+1} + \delta^{(+)}_{i+1} \\ &= \mathbf{emwd}(G_s, \nu_s, i+1) + \Delta(\nu_s^{-1}(i+1)) \\ &= \mathbf{emwd}(G_s, \nu_s, i+1) + 4\,. \end{aligned}$$

That is, the total number of boundaries involved in a parsing step is the number of boundaries that are not affected by the step, which correspond to edges passing over a vertex in the linear arrangement, $\mathbf{emwd}(G_s, \nu_s, i+1)$, plus the number of boundaries opened or closed by adding the new nonterminal, which is the vertex's degree $\Delta(\nu_s^{-1}(i+1))$.

Let w_1 and w_2 be the input strings in our synchronous parsing problem, and let n be the maximum between the lengths of w_1 and w_2. Observe that there may be $\mathcal{O}(n^{t_{i+1}})$ different instantiations of parsing step $i+1$ in our algorithm. In order to optimize the time complexity of our algorithm, relative to synchronous rule s, we then need to choose a linear arrangement that achieves maximum extended modified width of $\mathbf{emcw}(G_s)$. From Theorem 5, we then conclude that optimization of the time complexity of linear parsing for SCFGs is NP-hard.

5. Discussion

In this section, we discuss the implications of our results for machine translation. Synchronous parsing is the problem of finding a suitable representation of the derivations of a string pair consisting of one string from each language in the translation. In the context of statistical machine translation, this problem arises when we wish to analyze string pairs consisting of known parallel text in, say, English and Chinese, for the purposes of counting how often each SCFG rule is used and estimating its probability. Thus, synchronous parsing corresponds to the training phase of a statistical machine translation system. Our results show that it is NP-hard to find the linear synchronous parsing strategy with the lowest space complexity or the lowest time complexity. This indicates that learning complex SCFG rules from parallel text is a hard problem. Practical systems can, however, avoid this problem by imposing a limit on the number of nonterminals in a rule, or by using heuristics to find good parsing strategies [15].

A separate, but closely related, problem arises when translating new Chinese sentences into English, a problem known as decoding. A simple decoding algorithm consists of parsing the Chinese string with the Chinese side of the SCFG, and simply reading the English translation off of the English side of each rule used. This can be accomplished in time $\mathcal{O}(n^3)$ using the CYK parsing algorithm for (monolingual) context-free grammars, since we use only one side of the SCFG.

More generally, we may wish to compute not only the single highest-scoring translation, but a compact representation of all English translations of the Chinese string. Just as the chart constructed during monolingual parsing can be viewed as a non-recursive CFG generating all analyses of a string, we can parse the Chinese string with the Chinese side of the SCFG, retain the resulting chart, and use the English sides of the rules as a non-recursive CFG generating all possible English translations of the Chinese string. However, in general, the rules cannot be binarized in this construction, since the Chinese and English side of each rule are intertwined. This means that the resulting non-recursive CFG has size greater than $\mathcal{O}(n^3)$, with the exponent depending on the maximum length of the SCFG rules. One way to reduce the exponent is to factor each SCFG rule into a sequence of steps, as in our linear SCFG parsing strategies.

Machine translation systems do, in fact, require such a representation of possible translations, rather than simply taking the single best translation according to the SCFG. This is because the score from the SCFG is combined with a score from an English N-gram language model in order to bias the output English string toward hypotheses with a high prior probability, that is, strings that look like valid English sentences. In order to incorporate scores from an English N-gram language model of order m, we extend the dynamic programming algorithm to include in the state of each hypothesis the first and last $m-1$ words of each contiguous segment of the English sentence. Thus, the number of contiguous segments in English, which is the fan-out of the parsing strategy on the English side, again enters into the complexity [15]. Given a parsing strategy with fan-out f_c on the Chinese side and fan-out f_e on the English side, the space complexity of the dynamic programming table is $\mathcal{O}(n^{f_c} V^{2 f_e (m-1)})$, where n is the length of the Chinese input string, V is the size of the English vocabulary, and m is the order of N-gram language model. Under the standard assumption that each Chinese word has a constant number of possible English translations, this is equivalent to $\mathcal{O}(n^{f_c + 2 f_e (m-1)})$. Thus, our NP-hardness result for the space complexity of linear strategies for synchronous parsing also applies to the space complexity of linear strategies for decoding with an integrated language model.

Similarly, the time complexity of language-model-integrated decoding is related to the time complexity of synchronous parsing through the order m of the N-gram language model. In synchronous parsing, the time complexity of a step combining of state of type (s, σ, k) and a nonterminal $(A_{1,k+1}, A_{2,\pi^{-1}(k+1)})$ to produce a state of type $(s, \sigma, k+1)$ is $\mathcal{O}(n^{a+b+c})$, where a is the number of boundaries in states of type (s, σ, k), b the number in nonterminal $(A_{1,k+1}, A_{2,\pi^{-1}(k+1)})$, and c the number in type $(s, \sigma, k+1)$. If we rewrite a as $a_c + a_e$, where a_c is the number of boundaries in Chinese and a_e is the number of boundaries in English, then the exponent for the complexity of synchronous parsing is:

$$a_e + b_e + c_e + a_c + b_e + c_e$$

and the exponent for language-model-integrated decoding is:

$$(m-1)(a_e + b_e + c_e) + a_c + b_e + c_e$$

Note that these two expressions coincide in the case where $m = 2$. Since we proved that optimizing the time complexity of linear synchronous parsing strategies is NP-complete, our result also applies to the more general problem of optimizing time complexity of language-model-integrated decoding for language models of general order m.

Open problems This article presents the first NP-hardness result regarding parsing strategies for SCFGs. However, there is a more general version of the problem whose complexity is still open. In this article, we have restricted ourselves to linear parsing strategies, that is, strategies that add one nonterminal at a time to the subset of right hand side nonterminals recognized so far. In general, parsing strategies may group right hand side nonterminals hierarchically into a tree. For some permutations, hierarchical parsing strategies for SCFG rules can be more efficient than linear parsing strategies [15]. Whether the time complexity of hierarchical parsing strategies is NP-hard is not known even for the more general class of LCFRS. An efficient algorithm for minimizing the time complexity of hierarchical strategies for LCFRS would imply an improved approximation algorithm for the well-studied graph-theoretic problem of treewidth [26]. Minimizing fan-out of hierarchical strategies, on the other hand, is trivial, for both LCFRS and SCFG. This is because the strategy of combining all right hand side nonterminals in one step (that is, forming a hierarchy of only one level) is optimal in terms of fan-out, despite its high time complexity.

References

[1] A.V. Aho, J.D. Ullman, Syntax directed translations and the pushdown assembler, J. Comput. Syst. Sci. 3 (1969) 37–56.
[2] A.V. Aho, J.D. Ullman, The Theory of Parsing, Translation, and Compiling, vol. 1, Prentice-Hall, Englewood Cliffs, NJ, 1972.
[3] P.M. Lewis II, R.E. Stearns, Syntax-directed transduction, J. ACM 15 (3) (1968) 465–488.
[4] S. Shieber, Y. Schabes, Synchronous tree-adjoining grammars, in: Proceedings of the 13th International Conference on Computational Linguistics, COLING-90, Helsinki, vol. III, 1990, pp. 253–258.
[5] Y. Schabes, S.M. Shieber, An alternative conception of tree-adjoining derivation, Comput. Linguist. 20 (1994) 91–124.
[6] D. Chiang, Evaluation of grammar formalisms for applications to natural language processing and biological sequence analysis, Ph.D. thesis, University of Pennsylvania, 2004.
[7] D. Chiang, Hierarchical phrase-based translation, Comput. Linguist. 33 (2) (2007) 201–228.
[8] M. Galley, M. Hopkins, K. Knight, D. Marcu, What's in a translation rule?, in: Proceedings of the 2004 Meeting of the North American Chapter of the Association for Computational Linguistics, NAACL-04, Boston, 2004, pp. 273–280.
[9] P.F. Brown, S.A. Della Pietra, V.J. Della Pietra, R.L. Mercer, The mathematics of statistical machine translation: parameter estimation, Comput. Linguist. 19 (2) (1993) 263–311.
[10] P. Koehn, F.J. Och, D. Marcu, Statistical phrase-based translation, in: Proceedings of the 2003 Meeting of the North American Chapter of the Association for Computational Linguistics, NAACL-03, Edmonton, Alberta, 2003, pp. 48–54.
[11] D.H. Younger, Recognition and parsing of context-free languages in time n^3, Inf. Control 10 (1967) 189–208.
[12] J. Earley, An efficient context-free parsing algorithm, Commun. ACM 6 (8) (1970) 451–455.
[13] G. Satta, E. Peserico, Some computational complexity results for synchronous context-free grammars, in: Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing, HLT/EMNLP, Vancouver, Canada, 2005, pp. 803–810.
[14] D. Gildea, D. Štefankovič, Worst-case synchronous grammar rules, in: Proceedings of the 2007 Meeting of the North American Chapter of the Association for Computational Linguistics, NAACL-07, 2007, pp. 147–154.
[15] L. Huang, H. Zhang, D. Gildea, K. Knight, Binarization of synchronous context-free grammars, Comput. Linguist. 35 (4) (2009) 559–595.
[16] P. Crescenzi, D. Gildea, A. Marino, G. Rossi, G. Satta, Optimal head-driven parsing complexity for linear context-free rewriting systems, in: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, ACL-11, 2011, pp. 450–459.
[17] C. Gómez-Rodríguez, M. Kuhlmann, G. Satta, D. Weir, Optimal reduction of rule length in Linear context-free rewriting systems, in: Proceedings of the 2009 Meeting of the North American Chapter of the Association for Computational Linguistics, NAACL-09, 2009, pp. 539–547.
[18] B. Sagot, G. Satta, Optimal rank reduction for linear context-free rewriting systems with fan-out two, in: Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, Uppsala, Sweden, 2010, pp. 525–533.
[19] C. Gómez-Rodríguez, M. Kuhlmann, G. Satta, Efficient parsing of well-nested linear context-free rewriting systems, in: Human Language Technologies: the 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Los Angeles, California, 2010, pp. 276–284.
[20] K. Vijay-Shankar, D.L. Weir, A.K. Joshi, Characterizing structural descriptions produced by various grammatical formalisms, in: Proceedings of the 25th Annual Conference of the Association for Computational Linguistics, ACL-87, Stanford, CA, 1987, pp. 104–111.
[21] H. Seki, T. Matsumura, M. Fujii, T. Kasami, On multiple context-free grammars, Theor. Comput. Sci. 88 (1991) 191–229.
[22] J.E. Hopcroft, J.D. Ullman, Introduction to Automata Theory, Languages, and Computation, Addison–Wesley, Reading, MA, 1979.
[23] T. Bui, S. Chaudhuri, T. Leighton, M. Sipser, Graph bisection algorithms with good average case behavior, Combinatorica 7 (1987) 171–191.
[24] F. Makedon, C. Papadimitriou, I. Sudborough, Topological bandwidth, SIAM J. Algebr. Discrete Methods 6 (1985) 418–444.
[25] J.D.P. Rolim, O. Sýkora, I. Vrto, Optimal cutwidths and bisection widths of 2- and 3-dimensional meshes, in: 21st International Workshop on Graph-Theoretic Concepts in Computer Science, 1995, pp. 252–264.
[26] D. Gildea, Grammar factorization by tree decomposition, Comput. Linguist. 37 (1) (2011) 231–248.