Producing Genomic Sequences after Genome Scaffolding with Ambiguous Paths: Complexity, Approximation and Lower Bounds

Tom Davot, Annie Chateau, Rodolphe Giroudeau, Mathias Weller, Dorine Tabary

Producing genomic sequences after genome scaffolding with ambiguous paths

Tom Davot[1], Annie Chateau[1], Rodolphe Giroudeau[1], Mathias Weller[2], Dorine Tabary[3]

[1] LIRMM - CNRS UMR 5506 - Montpellier, France
[2] CNRS, LIGM (UMR 8049), Champs-s/-Marne, France
[3] MIPS EA 2332- Mulhouse, France
{davot,chateau,rgirou}@lirmm.fr, mathias.weller@u-pem.fr, dorine.tabary@uha.fr

Abstract. Scaffolding is the final step in assembling Next Generation Sequencing data, in which pre-assembled contiguous regions ("contigs") are oriented and ordered using information that links them (for example, mapping of paired-end reads). As the genome of some species is highly repetitive, we allow placing some contigs multiple times, thereby generalizing established computational models for this problem. We study the subsequent problems induced by the translation of solutions of the model back to actual sequences, proposing models and analyzing the complexity of the resulting computational problems. We find both polynomial-time and $\mathcal{NP}$-hard special cases like planarity or bounded degree. Finally, we propose two polynomial-time approximation algorithms according to cut/weight score.

1 Introduction

Context and motivation. Genomic data are of major importance in numerous aspects of research and applications in biology and computational biology. Their production is massively encouraged by industrial and academic actors, who use them in various ways [26]. These data are produced by so-called sequencers, who output (typically millions of) *reads*, that is, tiny subsequences of DNA that need to be "assembled" in order to get the target genome. The genome, once stabilized, is stored in huge databases and is made available to the community for further analysis. A high-quality genome is of paramount importance to the accuracy of further methods (such as genome comparison, gene inference, studies on the order of given markers, etc). Thus, it is crucial to provide genomes as complete and error-free as possible.

Preprint version.
The final authenticated version is available online at https://doi.org/10.1007/s00453-021-00819-6.

Assembly and scaffolding steps. The operation consisting in merging reads together to produce longer sequences is called *genome assembly*. Many methods and tools propose to assemble Next Generation Sequencing (NGS) reads into genomes, metagenomes or transcriptomes, most of them modeling sequences through graphs (assembly graphs, k-mer graphs, A-Bruijn graphs, etc.) [5, 8, 21, 23, 24, 25, 27, 37]. Those tools are compared and evaluated through benchmarks of different origins [16, 34]. A recent state-of-the art about genome assembly has been compiled by Phillippy [29]. However, assembly software typically has trouble dealing with repetitive (parts of the) genomes [22, 32, 33] and, therefore, output a collection of "contiguous regions" (*contigs*), that is, large chunks of DNA covering most of the genome. Unfortunately, nearly all "known" genomes are in a thusly fragmented state; some mammalian genomes reach hundreds of contigs per chromosome [1]. To overcome this issue, an additional step, called *scaffolding*, intends to reduce the fragmentation using additional data (paired-end reads, long reads, phylogenetic information, etc.). To this end, scaffolding software computes the most likely order and relative orientation of these contigs along the genome and, if possible, fills gaps between them [10, 12, 14, 30]. Scaffolding methods are also mainly based on various models of graphs, representing the way additional data link contigs. However, as with reads, the target genome may contain multiple (inverted) copies of an entire contig, like the well-known and described large inverted repeat region in chloroplast genomes [20], and many scaffolders are incapable of handling these repeats. Recent techniques use third-generation sequencing data [6] to resolve these repeats, but it requires resequencing the large amount of available, highly fragmented genomes. A possible way to solve the problem without resequencing is to deduce multiplicities of contigs using external information (such as read-coverage) and take this multiplicity into account when scaffolding.

Linearization of the solution. Scaffolding leads to a set of paths and cycles in a scaffolding graph. When a repeated contig is involved in several paths corresponding to distinct parts of the genome, it is impossible to distinguish between the copies, and paths collapse into non-linear structures (see Figures 1 and 2, requiring some definitions of Section 2). This solution structure is informative *per se* and could be used as it comes, but it presents sequences non-linearly. However, the standard representation of scaffolds are linear sequences of nucleotides. Thus, we need to *linearize* the solution graph, that is, resolve the ambiguities arising from the indistinguishability among the copies of each repeated contig, *This is the main subject of this work*. It turns out that the most straight-forward linearization strategies may produce chimeric sequences, and we show that the ones avoiding chimeras in a parsimonious way are $\mathcal{NP}$-hard to compute (for reasonable scoring). In particular, our model is an edge-deletion problem (called Semi-Brutal Cut) concentrated on extremities of ambiguities in a "solution graph" whose structure influences the computational tractability of the problem (see Table 1 and Table 2 for a summary).

Table 1: Overview of complexity results for Semi-Brutal Cut.

Topologies	Type of cut	Complexity	
Complete bipartite	Cut score	Linear	Theorem 4
Complete			Theorem 5
Cobipartite			Theorem 6
Trees	All		Theorem 2
$\Delta \leq 2$			Theorem 3
$\mathcal{G}'$[1]		$\mathcal{NP}$-hard	Theorem 7
Supergraphs of $\mathcal{G}'$	Weight score		Corollary 2
Split graphs	Cut score		Theorem 8
		No $2^{o(n)}$ algorithm (under $\mathcal{ETH}$)	Corollary 3

1. Bipartite, planar and subcubic

Table 2: Overview of lower and upper bounds for Semi-Brutal Cut.

Topologies	Type of Cut	Hypothesis	Lower bound		Upper bound
$\mathcal{G}'$[1]	Cut score	$\mathcal{P} \neq \mathcal{NP}$	1.00009...	Theorem 9	4-approx Theorem 15
Subcubic			1.00041...	Theorem 12	
$\Delta \leq 4$			1.0069...	Theorem 14	
$\Delta \leq 5$			1.0128...		
$\Delta \leq 6$			1.0138...		
General, Split graphs			1.360...	Theorem 13	
		$\mathcal{UGC}$	$2-\epsilon$		
General	Weight score	$\mathcal{P} \neq \mathcal{NP}$	1.01887...	Theorem 10	2-approx Theorem 16
			1.01515...		
Subcubic			1.00017...	Theorem 11	

1. Bipartite, planar and subcubic

Organization of this article. The next section is devoted to definitions and the descriptions of problems related to scaffolding and linearization. In Section 3, the notion of unambiguous solutions is explained. In Section 4, we describe several reduction rules yielding a simplified version of the linearization problem. The polynomial cases are developed in Section 5 whereas the hardness cases are presented in Section 6. The non-approximability results are given in Section 7 and, in the last section, several polynomial-time approximation algorithms are developed.

2 Obtaining Sequences From Solution Graphs

We consider simple, loopless graphs. Let G be such a graph. We denote by $V(G)$ and $E(G)$ the set of vertices and edges of G, respectively (or V and E if no ambiguity occurs). The degree of a vertex v is the number of edges incident to v and is denoted by $\deg_G(v)$. The set of neighbors of v is denoted by $N(v)$. The maximum degree of G is $\Delta(G)$. Consider a set of contigs $\mathcal{C} = \{C_1, \ldots, C_n\}$ and a set of weighted links between contig extremities (obtained from paired-end

reads mapping). This can be represented by a graph G containing, for each contig C_i, vertices u_i and v_i representing the extremities of C_i, an edge u_iv_i representing the contig C_i (*contig edge*), and weighted links between contig extremities (*non-contig edges*). The contig edges form a perfect matching M^* in G. In the following of the paper, we refer to them as *matching edges*. The weight function ω is defined on non-matching edges and symbolizes, roughly, the amount of confidence that we have in the link. We call such a graph a *scaffold graph*. For the matching M^* and a vertex u, we define $M^*(u)$ as the unique vertex v with $uv \in M^*$. Slightly abusing notation, we sometimes consider graphs as sets of edges. Then, a path p is *alternating* with respect to a matching M^* if, for all vertices u of p, also $M^*(u)$ is a vertex of p (see Figure 3a for an example). Then, a linear (circular) chromosome in the target genome is reflected as an alternating path (cycle) in G. One might now ask for the most parsimonious way (that is, discarding as little weight as possible) of inferring a given number σ_p of linear (and σ_c of circular) chromosomes that, together, make up $\mathcal{C}$. This problem has been modeled as the following, computationally hard problem [7, 35].

Scaffolding (SCA)
Input: A scaffold graph (G, M^*, ω) and $\sigma_\mathrm{p}, \sigma_\mathrm{c}, k \in \mathbb{N}$.
Question: Is there some $S \subseteq E(G) \setminus M^*$ such that $S \cup M^*$ is a collection of $\leq \sigma_\mathrm{p}$ alternating paths and $\leq \sigma_\mathrm{c}$ alternating cycles and $\omega(S) \geq k$?

To work with multiplicities, we consider walks instead of paths. A length-ℓ *walk* in a graph G is a sequence $(u_0, u_1, \ldots, u_\ell)$ of vertices in $V(G)$ such that, for each two consecutive vertices u_i and u_{i+1} in the sequence, we have $u_iu_{i+1} \in E(G)$. The walk is called *closed* if $u_0 = u_\ell$ and it is called *alternating* with respect to a perfect matching M^* in G if (a) $u_iu_{i+1} \in M^*$ if and only if i is even, and (b) ℓ is even if and only if the walk is closed. We will consider walks as multisets of edges. For any multiset W, let $\chi_W(e)$ be the number of times that e occurs in W and let $\omega(W) := \sum_{e \in W} \chi_W(e)\omega(e)$. When working with multiplicities, each matching edge e of the scaffold graph has a *multiplicity* $m'(e)$. For matching edges, this can be read from the data as described in the introduction. Then, the scaffolding problem with multiplicities is the following:

Scaffolding with Multiplicities (MSCA)
Input: A scaffold graph (G, M^*, ω, m') and $\sigma_\mathrm{p}, \sigma_\mathrm{c}, k \in \mathbb{N}$.
Question: Is there a multiset S of $\leq \sigma_\mathrm{c}$ closed and $\leq \sigma_\mathrm{p}$ non-closed alternating walks in (G, M^*, ω) such that each $e \in M^*$ occurs exactly $m'(e)$ times in walks of S and $\omega(S) \geq k$?

Obtaining solutions for MSCA is not the topic of this work. Instead, we consider a solution for MSCA, that is, a multiset S of alternating walks in G such that each $e \in M^*$ occurs exactly $m'(e)$ times in walks of S. From S, we reconstruct a *solution graph*[4] $\mathrm{sol}(S) := (G^*, M^*, \omega, m)$ by "merging" all walks of

[4] Solution graphs differ from scaffold graphs in that the multiplicity function is defined on all edges and not just on matching edges.

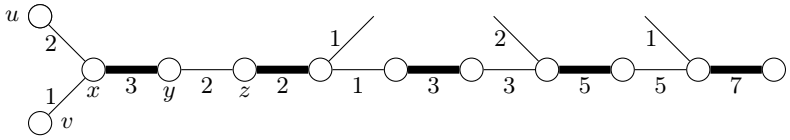


Fig. 1: Walks in a scaffold graph (not drawn) give a solution graph (drawn) with multiplicities. Matching edges are bold. The only ambiguous path is (x, y). It is ambiguous because it can be decomposed into $\{(\dots, u, x, y, z, \dots), (\dots, u, x, y, z, \dots), (\dots, v, x, y)\}$ or $\{(\dots, u, x, y), (\dots, u, x, y, z, \dots), (\dots, v, x, y, z, \dots)\}$. Removing all non-matching edges incident with x or all non-matching edges incident with y destroys all ambiguous paths.

S, that is, G^* contains exactly the edges e of G that occur in walks of S and $m(e) = \sum_{W \in S} \chi_W(e)$ is the number of their occurrences. Note that the function m is defined on every edges (contrary to m' which is only defined on the matching edges) and that for any matching edge e, we have $m(e) = m'(e)$. We also say that $\mathrm{sol}(S)$ is *made up of* S. This merge translates the fact that copies of repeated contigs cannot be distinguished using information from the scaffold graph. Any set of walks making up this solution graph is also a solution of Scaffolding with Multiplicities with the same optimal score. The solution graph is, in fact, a manner of representing all the optimal solutions. Any arbitrary choice between them could lead to chimeric scaffolds.[5] Indeed, the problem is that sol is not necessarily injective. For example, suppose that the edge xy in Figure 1 is used in three walks, two of which contain the vertex z. As x is incident to different non-matching edges, one of the three walks differs from the other two, but it cannot be determined whether or not it is the same walk that avoids z. See also Figure 2 for an example with sequences and Figure 3 for an example of a scaffold graph leading to a solution graph with ambiguous sequences. This notion is captured in the following definition.

Definition 1. *Let A be an alternating path between u and v or an alternating cycle in a solution graph. If all edges of A have the same multiplicity μ (that is, $m(e) = \mu$ for all $e \in A$), then A is called* μ-uniform *(or simply* uniform *if μ is unknown). Further,*

1. *if A is an alternating μ-uniform cycle and $\mu > 1$, or*
2. *if A is an alternating μ-uniform u-v-path and each of u and v is incident with a non-matching edge of multiplicity strictly less than μ,*

then A is called ambiguous.

An example of ambiguous path is depicted in Figure 1. Roughly speaking, the problem is that there are many ways of pairing up sequences on each end of

[5] A sequences is called *chimeric* if it does not occur in the target genome, but is made up of chunks picked from different chromosomes or regions of the genome.

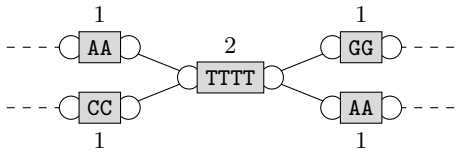


Fig. 2: A schema illustrating solution ambiguity: from the solution graph alone, we cannot tell whether the target genome contains (1) `AATTTTGG` and `CCTTTAA` or (2) `AATTTTAA` and `CCTTTTGG`. As methods "ignore" and "clever" choose one of the two, they may produce wrong sequences. Method "brutal" removes all four edges incident with the matching edges `TTTT` and "semi-brutal" removes either the left or the right pair of edges. Note that this problem does not go away if we require the solution to be represented as a collection of walks, as this collection will just represent one of the arbitrary choices.

ambiguous paths and that the number of cycles is undefined in ambiguous cycles. Interestingly, ambiguous paths and cycles are enough to characterize ambiguity of solution graphs (proof in Section 3).

Theorem 1. *Let G^* be a solution graph. Then, G^* is made up of a unique multiset of alternating walks if and only if G^* does not contain ambiguous paths or cycles.*

For biological applications, the representation as solution graph is not satisfying. Instead, it is necessary to translate the solution into sequences. However, each solution S corresponds to a different collection of sequences which, without additional external knowledge, are equally likely from a biological point of view. For a solution graph G^*, we let $\text{sol}^{-1}(G^*)$ denotes the set of multisets S of walks with $\text{sol}(S) = G^*$. Theorem 1 states that $|\text{sol}^{-1}(S)| = 1$ if and only if G^* does not contain ambiguous paths or cycles. However, if the solution graph does contain ambiguous paths, we propose the following strategies for its translation into sequences.

Ignore. Choose an arbitrary multiset of walks making up G^*. In this case, we preserve the weight of the solution, but there is no way to distinguish between the elements of $\text{sol}^{-1}(G^*)$ and the arbitrary choice could lead to an erroneous solution, biologically speaking. Indeed, there is a risk to produce a chimeric sequence, and this strategy has to be put aside in a bioinformatic context.

Clever. Choose walks that optimize some criterion (i.e. N50[6]). This strategy consists in finding, among all solutions of maximal weight in $\text{sol}^{-1}(G^*)$, one which maximizes this global criterion. Again, this strategy induces a risk to produce chimeric sequences, and we will not consider it any further.

Brutal. Isolate ambiguous paths by removing all non-matching edges incident to their extremities. Remove one non-matching edge in each ambiguous cycle to transform it into a uniform path.

[6] N50 is a statistical measure on contig lengths: given a set of contigs, the N50 is defined as the sequence length of the shortest contig at 50% of the total genome length.

Semi-brutal. Choose a proper set of endpoints of ambiguous path and remove all non-matching edges incident to it. Remove one non-matching edge in each ambiguous cycle to transform it into a uniform path.

We will focus on methods "brutal" and "semi-brutal" as the other methods may produce chimeric sequences (See Figure 2). However, since we remove edges, the final scaffold may not have maximum weight among all uniquely linearizable solutions, and we will discuss this point. Method "brutal" can be executed in polynomial time, but it may decrease the weight of the solution drastically. Method "semi-brutal" forces us to make a choice each time we encounter an ambiguous path, and we might want to choose "wisely", that is, destroy ambiguous paths in a way that optimizes a scoring. Let v be either an extremity of an ambiguous path or a vertex of an ambiguous cycle, we sometimes say "to cut v", by which we mean removing all non-matching edges incident to it, and in that case v is denoted as a *cut*. Thus, the following problem arises:

Semi-Brutal Cut (SBC)
Input: A solution graph (G^*, M^*, ω, m) and some $k \in \mathbb{N}$.
Question: Is there a set X of cuts of G^* which destroys all ambiguous paths and the score of X is at most k?

In Section 4, we show how to simplify the problem statement. Notice that separating Semi-Brutal Cut from Scaffolding with Multiplicities is necessary in order to avoid the production of a chimeric sequence, as explained in Figure 3. Several possible scoring functions seem sensible to optimize:

Cut score. Pay one per cut: $\text{score}(X) := |X|$.

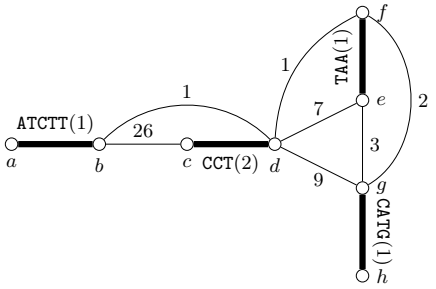
Path score. Pay one for each multiplicity that is cut:
$\text{score}(X) := \sum\{m(uv) \mid uv \in E(G^*) \setminus M^* \wedge \{u, v\} \cap X \neq \varnothing\}$.

Weight score. Pay the total weight of edges that are cut:
$\text{score}(X) := \sum\{m(uv) \cdot \omega(uv) \mid uv \in E(G^*) \setminus M^* \wedge \{u, v\} \cap X \neq \varnothing\}$.

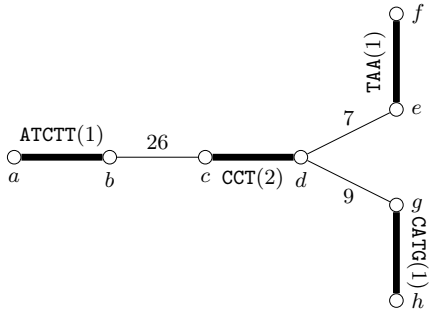
Note that, from the perspective of computational complexity, the path score is a special case of the weight score, since we can just set $\omega(e) = 1$ for all edges e. Thus, when saying "both scores" we refer to cut and weight score. Unfortunately, it turns out that all these variants are $\mathcal{NP}$-hard (see Section 6).

3 Unambiguous Solutions

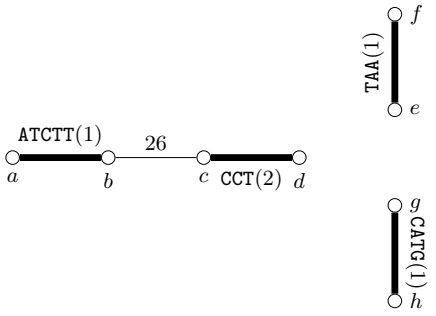
We show in this section that the solution graph G^* has to be free of ambiguous paths and cycles in order to be uniquely deconstructable into walks that make up G^*. To this end, we present a reduction rule whose application does not change unique deconstructability (indeed, it does not change $|\text{sol}^{-1}(G^*)|$).



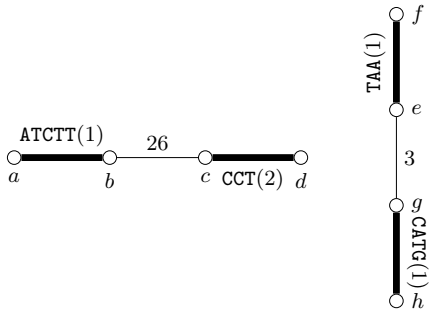
(a) Scaffold graph. This graph illustrates relationship between four contigs, figured by bold edges ab, cd, ef and gh. Labels on these edges show the sequence of the contigs, and their mutliplicity (in parenthesis). Edge cd, whose sequence is `CCT`, has multiplicity two. Other contigs have multiplicity one. Links between contigs are labeled by their weight.



(b) Solution graph after solving Scaffolding with Multiplicities. The solution graph is obtained as a solution for the MSCA instance asking for two open walks with total weight ≥ 42. In the solution graph, the contig of multiplicity two labeled `CCT` constitutes an ambiguous path, yielding two possible sets of sequences $\{$`ATCCT..CCT..TAA`, `CCT..CATG`$\}$ and $\{$`ATCCT..CCT..CATG`, `CCT..TAA`$\}$.



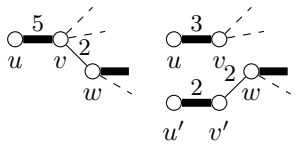
(c) Linearization using Semi-Brutal Cut. Brutal cut would provide a set of four independent sequences of total weight zero (the initial set of contigs), whereas Semi-Brutal Cut with weight-score provides a unique set of four sequences $\{$`ATCCT..CCT`, `CCT`, `TAA`, `CATG`$\}$, and weight 26 (minimal weight-score 16). After solving successively MSCA (with $\sigma_p = 2$ and $\sigma_c = 0$) and SBC, the solution is compatible with the initial hypothesis. The only ambiguous path is the matching edge cd and the cut vertex is d.



(d) Direct linearization from the scaffold graph. Directly searching two maximum weighted alternating paths such that the solution graph does not contain ambiguity yields a chimeric sequence `TTA..CATG` (note that the sequence of (f, e) is the reverse complement of the sequence of (e, f)) corresponding to (f, e, g, h).

Fig. 3: Example for a hypothetical genome consisting of the chromosomes `ATCTT..CCT..TAA` and `CCT..CATG`: a scaffold graph (Figure 3a), a solution graph (Figure 3b), scaffolds after solving Semi-Brutal Cut (Figure 3c), and a direct linearization leading to chimeric solution (Figure 3d).

Rule 1 *Let $(u, \ldots, v)$ be a μ-uniform alternating path in G^* such that $\deg_{G^*}(u) = 1$. Let vw be a non-matching edge. Then, create the $m(vw)$-uniform alternating path $(u', \ldots, v')$, create the edge $v'w$ with multiplicity $m(vw)$, remove vw, and decrease the multiplicity of $(u, \ldots, v)$ by $m(vw)$.*



We prove the correctness of this rule, that is, the input solution graph can be uniquely deconstructed if and only if the output solution graph can.

Proof. Let G_1^* be a solution graph and G_2^* be the solution resulting of the application of Rule 1 in G_1^*. Consider the function τ mapping multisets $\mathcal{W}$ of walks making up G_1^* to multisets of walks in G_2^*. It works by replacing $(u, \ldots, v, w)$ (or $(w, v, \ldots, u)$) in $m(vw)$ walks by $(u', \ldots, v', w)$ (or $(w, v', \ldots, u')$). Clearly, no two different multisets for G_1^* map to the same multiset for G_2^* and, thus, τ is injective. To show that τ is surjective, suppose that there is a multiset $\mathcal{W}'$ of walks making up G_2^* that is not in the image of τ. Note that any walk W' of $\mathcal{W}'$ containing $(u', \ldots, v')$ also contains $(u', \ldots, v', w)$ (or $(w, v', \ldots, u')$) as a sub-walk, as for any edge e in $(u', \ldots, v'), m(e) = m(v'w)$ and no walk starts with a non-matching edge. Thus, replacing $(u', \ldots, v', w)$ (or $(w, v', \ldots, u')$) by $(u, \ldots, v, w)$ (or $(w, v, \ldots, u)$) in all walks of W' yields a multiset W of walks making up G_1^* and $\tau(W) = W'$. Thus, τ is a bijection implying that the number of different multisets of walks making up G_1^* is equal to the number of multisets making up G_2^*. □

Theorem 1. *Let G^* be a solution graph. Then, G^* is made up of a unique multiset of alternating walks if and only if G^* does not contain ambiguous paths or cycles.*

Proof. "⇒": Suppose that G^* contains an ambiguous cycle c. Let $\mu' > 1 \in \mathbb{N}$ such that c is a μ'-uniform alternating cycle. For each $k \in \mathbb{N}$, let c^k be the closed alternating walk which passes k times across the edges of c. The two multisets of walks $\{c^{\mu'}\}$ and $\{c^1, c^{\mu'-1}\}$ make up c, contradicting the uniqueness of such a multiset. Suppose that G^* contains an ambiguous path $p = (v, w, \ldots, x, y)$ and let $\mathcal{W}$ be a multiset of walks that make up G^*. Let $m \in \mathbb{N}$ such that p is μ-uniform and let uv and yz be non-matching edges incident to v and y, respectively, whose respective multiplicities are strictly less than m. Thus, $\mu > 1$. Note that no walk W of $\mathcal{W}$ starts or ends with an inner edge $e \in M^*$ of p, since otherwise, e is incident with a non-matching edge of p that is traversed strictly less than μ times, as no walk of $\mathcal{W}$ starts or ends with a non-matching edge. Thus, each time a walk of $\mathcal{W}$ traverses vw, it also traverses p. Consider the graph $G_{\mathcal{W}}^*$ on the vertex set $\{(W_1, W_2) \mid (W_1, p, W_2) \in \mathcal{W}\}$ and $G_{\mathcal{W}}^*$ contains an edge $\{(W_1, W_2), (W_1', W_2')\}$ if and only if $W_1 = W_1'$ ("blue edge") or $W_2 = W_2'$ ("red edge"). Note that sub-walks can be empty and no edge is blue and red at the same time, as otherwise, its endpoints are equal (but there are no self-loops in $G_{\mathcal{W}}^*$). Also note that the blue edges form a transitive subgraph of $G_{\mathcal{W}}^*$ and, by symmetry, so do the red edges. Since the multiplicity of uv in G^* is non-zero and different from that of vw, we

know that $G^*_{\mathcal{W}}$ does not entirely consist of blue edges and, by symmetry, the same can be said for red edges. Thus, $G^*_{\mathcal{W}}$ is not a clique and, therefore, there are pairs (W_1, W_2) and (W'_1, W'_2) such that (a) $W = (W_1, p, W_2)$ and $W' = (W'_1, p, W'_2)$ are (not necessarily distinct) walks in $\mathcal{W}$ and (b) $W_1 \neq W'_1$ and $W_2 \neq W'_2$. If $W \neq W'$, then the result of removing W and W' from $\mathcal{W}$ and inserting (W_1, p, W'_2) and (W'_1, p, W_2) is another multiset of walks making up G^*. Thus, $W' = W = (X_1, p, X_2, p, X_3)$ for walks X_1, X_2, X_3 in G^*. But then, the result of removing W from $\mathcal{W}$ and inserting (X_1, p, X_3) and the closed walk consisting of p and X_2 is another multiset of walks making up G^*. In both cases, $\mathcal{W}$ is not unique.

"$\Leftarrow$": Let G^* be free of ambiguous paths or cycles. We suppose that Rule 1 is applied on G^*. If G^* is empty, then G^* has a unique multiset of walks making it up. If G^* contains a uniform alternating cycle c, then since c is not ambiguous, c is 1-uniform. Hence, the unique multiset making up c is $\{c\}$. Otherwise, let $\mu \in \mathbb{N}$ and let $p = (u, \ldots, v)$ be a maximal, μ-uniform, alternating path in G^* (as p may consist of a single edge and G^* is not empty, p exists). Note that all inner vertices of p have degree two in G^* and suppose without loss of generality that $\deg_{GS}(u) \leq \deg_{G^*}(v)$. If $\deg_{G^*}(u) = 1$ and $\deg_{G^*}(v) \geq 2$, then Rule 1 applies. Thus, suppose $\deg_{G^*}(u) > 1$ and $\deg_{G^*}(v) > 1$. Then, by maximality of p, both u and v are incident to a non-matching edge with multiplicity strictly less than m and, thus, p is ambiguous, contradicting the assumption that G^* is free of ambiguous paths. Hence, $\deg_{G^*}(u) = 1$ and $\deg_{G^*}(v) = 1$, and p is isolated. The multiset consisting of μ $(u, \ldots, v)$-walks is the unique multiset making up p. □

4 Reduction rules

In this section, we present a set of reduction rules that simplify instances of Semi-Brutal Cut. First, let us deal with some trivial cases to remove them from consideration.

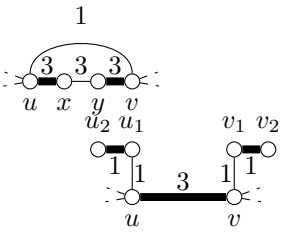
Rule 2 *Let c be an isolated, μ-uniform cycle in (G^*, M^*, ω, m). If $\mu = 1$, then remove c. Otherwise, cut a vertex incident to the lightest non-matching edge of c.*

Rule 3 *Remove all isolated, uniform, alternating paths from (G^*, M^*, ω, m).*

Rule 4 *Let $uv \in M^*$ be a matching edge that does not occur in ambiguous paths and let u and v have degree at least two. Then, remove uv, add new vertices u' and v' and add the matching edges uv' and vu' with multiplicity $m(uv)$.*

Correctness of Rule 4 follows immediately from the fact that no ambiguous path is changed, created or destroyed by applying the rule. Furthermore, since both u' and v' have degree one in the result G' of applying the rule, all solutions X for G' avoid them and, since all scoring functions only depend on the non-matching edges incident to the solution, all solutions maintain their scores.

Rule 5 *Let $\mu \in \mathbb{N}$ and let $p = (u, \dots, v)$ be a μ-uniform, alternating path in G^*. If uv is a non-matching edge of G^*, then create two matching edges u_1u_2 and v_1v_2 with multiplicity $m(uv)$, add the non-matching edges uu_1 and vv_1 with weight 0 and multiplicity $m(uv)$, and remove uv (and, for the weight score, decrease k by $\omega(uv)$). In any case, remove all inner vertices of p and create a matching edge uv with multiplicity μ.*



Proof (Correctness of Rule 5). First, since no inner vertex of p can be cut, replacing p by a single matching edge does not change the score (cut- or weight-) of a solution. It remains to show correctness for the case that uv exists in G^*. Then $m(uv) < \mu$ since the input graph does not contain isolated cycles and, thus, p is ambiguous. Thus, any solution X for G^* contains u or v. In the output graph, p is still ambiguous and either u_1u or v_1v must be removed in any solution. Further, u_1 and v_1 can be replaced by u and v, respectively, in any solution for the output graph. Thus, X is a solution in the input graph if and only if it is a solution in the output graph. X has the same cut score in both input and output graph. Under the weight score, $score(X)$ decreases by $\omega(uv)$. □

Finally, note that all reduction rules can be applied in linear time. Further, it turns out that all matching edges of G^* either occur in ambiguous paths or are incident with a degree-one vertex. In the latter case, we call the matching edge *clean*.

Proposition 1. *Let (G^*, M^*, ω, m) be reduced with respect to the presented reduction rules. Then, it is free of ambiguous paths if and only if all its edges are clean.*

Proof. "⇒": To show the contraposition, let uv be an edge that is not clean, that is, u has a neighbor $x \neq v$ and v has a neighbor $r \neq u$. If $m(ux), m(vr) < m(uv)$, then uv is an ambiguous path, proving the claim. Otherwise, since $m(vx), m(ur) \leq m(uv)$ by definition of solution graph, $m(vx) = m(uv)$ or $m(ur) = m(uv)$. By symmetry, suppose that $m(vx) = m(uv)$ and let $y := M^*(x)$. By definition of solution graph, $m(xy) \geq m(vx)$. If $m(xy) = m(vx)$, then $p = (y, x, v, u)$ is an $m(uv)$-uniform, alternating path in G^*, contradicting reducedness with respect to Rule 5. Thus, suppose $m(xy) > m(vx)$. But then, any ambiguous path containing uv must end at v and, since v is not incident to an edge of multiplicity strictly less than $m(uv)$, we know that uv is not contained in any ambiguous paths, contradicting reducedness with respect to Rule 4.

"⇐": To show the contraposition, let $p = (u, \dots, v)$ be an ambiguous path in G^* and note that both u and v are incident to a non-matching edge. But then, clearly, $uM^*(u)$ cannot be clean in G^*. □

Given Proposition 1, the multiplicity function is no longer important since the presence of a vertex of degree one in a matching edge suffices to determine if the matching edge is ambiguous. Semi-Brutal Cut can be now described as follows.

Semi-Brutal Cut (SBC)
Input: A solution graph (G^*, M^*, ω) and some $k \in \mathbb{N}$.
Question: Is there a set X of cuts of G^* which makes all the matching edge clean and the score of X is at most k?

5 Polynomial cases

In the following, we consider special solution graphs for which Semi-Brutal Cut can be solved in polynomial time for all of the presented scoring functions. Recall that the goal is to clean all matching edges in G^* (see Proposition 1).

5.1 Sparse graphs

We first show that SBC is polynomial for both scores into some classes of sparse graphs. First, we consider the class of trees. We suppose that the tree G^* is rooted in an extremity of an ambiguous edge. Under the weight score, we can thus formulate the following dynamic program.

Let x be a vertex, T_x is the subgraph induced by the subtree rooted at x and $M^*(x)$. For any vertex x, a table entry $c(x)$ represents the minimum score of a solution below x in which x has degree one in T_x and $\bar{c}(x)$ represents the minimum score of a solution below x in which $M^*(x)$ has degree one in T_x. For convenience, we set $\omega(e) = 0$ for all matching edge e. If x is a leaf of G^* then, clearly, $c(x) = \bar{c}(x) = 0$. For any non-leaf x, we set

$$c(x) = \sum_{y \in Children(x)} min(\bar{c}(y), c(y)) + \omega(xy)$$

$$\bar{c}(x) = \sum_{y \in Children(x)} \begin{cases} c(y) & \text{if } y = M^*(x) \\ min(\bar{c}(y), c(y) + \omega(xy)) & \text{otherwise} \end{cases}$$

Lemma 1. *Those costs $c(x)$ and $\bar{c}(x)$ represent respectively the minimum weight score of a semi-brutal cut in the subtree rooted at x when x or $M^*(x)$ has degree one in T_x.*

Notice that it may hapend that both extremities of a matching edge have degree one in an optimal solution, and in this case, we have $c(x) = \bar{c}(x)$.

Proof. We prove it by induction on the subtree's height. Let x be any vertex in the tree. Let $h(x)$ denote the height of the subtree rooted at x. When $h(x) = 0$, x is a leaf and a solution of Semi-Brutal Cut in T_x consists in cutting nothing, thus the cost is zero. Suppose now that for any vertex x' with height $h(x') < h(x)$, $c(x')$ and $\bar{c}(x')$ satisfy the lemma's property. We prove that:

1. any solution X of Semi-Brutal Cut in T_x has $score(X) \geq c(x)$ if x has degree one in T_x after applying X, or $score(X) \geq \bar{c}(x)$ if $M^*(x)$ has degree one in T_x, and

2. there exists a solution S_x reaching the costs $c(x)$ and $\bar{c}(x)$.

1. Let X be a solution of Semi-Brutal Cut in T_x. We denote by X_y the restriction of X to T_y, for any children y of x. X_y is trivially a solution of Semi-Brutal Cut in T_y, whose height is strictly less than $h(x)$. By induction hypothesis, $score(X_y) \geq c(y)$ if all non-matching edges incident to y in T_y are removed in X_y, or $score(X_y) \geq \bar{c}(y)$ otherwise.
   - Suppose that all non-matching edges incident to x in T_x are removed in X. Thus, the weight score of X contains the total weight of these non-matching edges plus the score of any subsolution X_y:

     $$score(X) = \sum_{y \in Children(x)\}} score(X_y) + \omega(xy)$$

     Using the previous equation and induction hypothesis, we have $score(X) \geq c(x)$.
   - Suppose that all non-matching edges incident to $M^*(x)$ in T_x are removed after applying X. For any child y, the weight score of X contains the score of X_y plus $\omega(xy)$ if y belongs to X_y:

     $$score(X) = \sum_{y \in Children(x)} \begin{cases} score(X_y) + \omega(xy) & \text{if } y \in X_y \\ score(X_y) & \text{otherwise} \end{cases}$$

     We distinguish $M^*(x)$ amongst children of x.
     - If $M^*(x)$ is in $Children(x)$, then necessarily all incident edges to it has to be removed. In this case, $score(X_{M^*(x)}) \geq c(M^*(x))$, by induction hypothesis.
     - For any other children y of x, either y has degree one in X_y, yielding a cost $c(y)+\omega(xy)$, or $M^*(y)$ has degree one in X_y, yielding a cost $\bar{c}(y)$.

     Hence, using the previous equation and induction hypothesis, we have $score(X) \geq \bar{c}(x)$.
2. Now we show that is possible to build two solutions X_x and $\bar{X}_x$ of Semi-Brutal Cut in T_x with weight $c(x)$ and $\bar{c}(x)$, respectively. Considering a child y of x, we denote by X_y a solution of Semi-Brutal Cut in T_y, where all non-matching edge incident to y in T_y are remoed, with weight score $c(y)$, and $\bar{X}_y$ a solution of Semi-Brutal Cut in T_y, where all non-matching edge incident to $M^*(y)$ are removed in T_y, with weight score $\bar{c}(y)$. Such solutions do exist, by induction hypothesis.
   - We define the set X_x by:

     $$X_x = \{x\} \cup \bigcup_{y \in Children(x)} \begin{cases} X_y & \text{if } c(y) < \bar{c}(y) \\ \bar{X}_y & \text{otherwise.} \end{cases}$$

     Since X_y and $\bar{X}_y$ are solutions of Semi-Brutal Cut in T_y, they clean all ambiguous edges below y. Removing all non-matching edges incident to x cleans the ambiguous edge $xM^*(x)$ in T_x. Thus, X_x is a solution of Semi-Brutal Cut in T_x, with weight score $c(x)$.

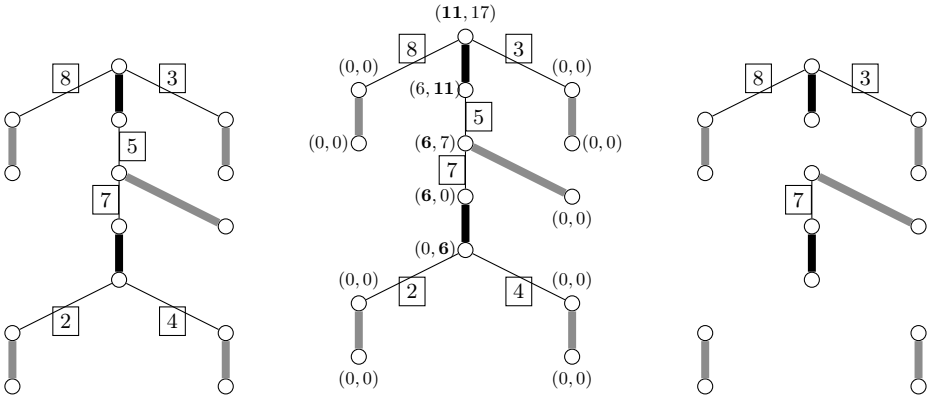


Fig. 4: Example of an application of the dynamic programming algorithm with matching edges (bold, gray if already clean) and weights (numbers in boxes). **Left**: the input solution graph G^*. **Middle**: costs $(\bar{c}, c)$ after the bottom-up step. Bold figures indicate the backtracking path. For example, the minimal cost at the root is a non-cutting cost 11, which comes from the cutting cost 11 of its matching-child and non-cutting costs or cutting costs of non-matching children. **Right**: resulting solution graph after the backtracking step, which is made up of six paths.

– We define the set $\bar{X}_x$ by:

$$\bar{X}_x = \bigcup_{y \in Children(x)} \begin{cases} X_y & \text{if } c(y) + \omega(xy) \leq \bar{c}(y) \text{ or } y = M^*(x) \\ \bar{X}_y & \text{otherwise} \end{cases}$$

Since the X_y and $\bar{X}_y$ are solutions of Semi-Brutal Cut in T_y, they clean all ambiguous edges below y. If $M^*(x)$ is below x, a solution removing all non-matching edges incident to $M^*(x)$ in $T_{M^*(x)}$ cleans $xM^*(x)$. For any other children of x, either y belongs to X_x or $M^*(y)$ has degree one, thus X_x cleans $yM^*(y)$ in T_x. Thus $\bar{X}_x$ is a solution of Semi-Brutal Cut in T_x, with weight score $\bar{c}$.

□

Corollary 1. *Using a bottom-up step computing these costs, setting the score of the root as the minimum between $c(r)$ and $\bar{c}(r)$ and backtracking those costs to decide which vertices should be cut leads to an optimal solution in linear time and space.*

An example of the application of this dynamic program can be found in Figure 4. While presented here for the weight score, we remark that this dynamic program can be modified to work for the cut score. For that we add a third table entry representing the fact x is not cut and all neighbors of x except $M^*(x)$ are in the solution. Hence, the formulation of the dynamic program is the following with

$n(x) = 0$ if x is a leaf of G^*.

$$c(x) = \sum_{y \in Children(x)} min(\bar{c}(y), c(y), n(y)) + 1$$

$$\bar{c}(x) = \sum_{y \in Children(x)} \begin{cases} min(c(y), n(y)) & \text{if } y = M^*(x) \\ min(\bar{c}(y), c(y)) & \text{otherwise} \end{cases}$$

$$n(x) = \sum_{y \in Children(x)} \begin{cases} \bar{c}(y) & \text{if } y = M^*(x) \\ c(y) & \text{otherwise} \end{cases}$$

Since we can easily adapt the proof of Lemma 1 to the cut score formulation, we let the reader check the correctness of this dynamic program.

Theorem 2. *On trees,* Semi-Brutal Cut *can be solved in linear time and space for both scoring functions.*

As a side note, we remark that Semi-Brutal Cut can be solved in linear time if $\Delta(G^*) = 2$. To this end, we just need to check the two possibilities of removing every second non-matching edge in every cycle. Since each cycle can be worked on individually and independently, this can be done in linear time. What remains can be solved in linear time with Theorem 2.

Theorem 3. Semi-Brutal Cut *can be solved in linear time on a collection of paths and cycles ($\Delta(G^*) = 2$) under both scores.*

5.2 Dense graphs

In some classes of dense graphs, we can show that SBC is polynomial under the cut score. Concerning the weight score, Corollary 2 states that SBC is $\mathcal{NP}$-complete for most dense classes (Section 6.1). For the following proofs, note that any graph can be solved with $|M^*|$ cuts by simply cutting an arbitrary extremity of all matching edges.

Theorem 4. Semi-Brutal Cut *can be solved in linear time for cut score on complete bipartite graphs.*

Proof. Note that, if G^* is bipartite, both cells of the partition have equal size since M^* is a bijection between the two. Let $K_{n,n}$ be a complete bipartite graph (with $n := |M^*|$ and suppose that $n \geq 2$ as, otherwise, matching edges are already clean. Then, it is sufficient to cut all but one vertex of any of the two cells of the bipartition to turn all matching edges clean. To show that $n - 1$ cuts are also necessary, assume that there is a solution X with cut score $n - 2$. Since there are n matching edges in G^*, there are two matching edges uv and xy that do not intersect X. Since G^* is complete bipartite, (u, v, x, y) forms an alternating cycle in G^*, so neither uv nor xy are clean. □

Theorem 5. Semi-Brutal Cut *can be solved in linear time under cut score on complete graphs.*

Proof. If any solution does not contain a cut in a matching edge uv, then either all neighbors of u or all neighbors v are cut, which implies a solution with a cut score of $|V(G^*)| - 2 > |M^*|$. Hence, $|M^*|$ cuts are necessary. □

Theorem 6. Semi-Brutal Cut *can be solved in linear time under cut score on co-bipartite*[7] *graphs.*

Proof. In this proof, suppose that $|M^*| > 2$ as, otherwise, the claim trivially holds. Let (V_1, V_2) denote a bipartition of the vertices of G^* into two cliques. First, assume that (G^*, M^*, ω) has a solution X with $|X| = |M^*| - 2$ cuts. Then, there are matching edges uv and xy that avoid X. If either V_1 or V_2 intersects $uvxy$ in at least three vertices, say u, v, and x, then uv is not clean. Thus, $uvxy$ intersects both cells in exactly two vertices. If one cell contains ux and the other vy, then $uvxy$ induces a cycle, and neither uv nor xy are clean. Thus, without loss of generality, let $uv \subseteq V_1$ and $xy \subseteq V_2$. But then, all other vertices have to be cut, implying $|X| \geq 2(|M^*| - 2) > |M^*| - 2$.

Since we know that no solution X with $|X| \leq |M^*| - 2$ exists and a solution X with $|X| = |M^*|$ is trivial, we just have to check if G^* contains a matching edge uv such that we can cut the vertices of $N(u) - v$ instead of cutting u or v. We show that G^* can be solved with $|M^*| - 1$ cuts if and only if there is a matching edge uv with $|N(u)| \leq |M^*|$ and there are no matching edges $xy \subseteq N(u)$. Since this can be checked in linear time, the theorem follows.

"⇒": If there is a solution X for G^* with $|X| = |M^*| - 1$, then there is a matching edge uv avoiding X and all other matching edges intersect X in exactly one extremity. By symmetry, let $N(u) - v \subseteq X$, implying $|N(u)| \leq |M^*|$ and, as each matching edge except uv contains only a single cut, no matching edge xy is included in X and, thus, in $N(u)$.

"⇐": Let Q be a set containing an arbitrary extremity of each $xy \in M^*$ with $xy \cap N(u) = \varnothing$ and let $X := Q \cup N(u) - v$. Then, $|Q| = |M^*| - |N(u)|$ and $|X| = |Q| + |N(u)| - 1 = |M^*| - 1$. Towards a contradiction, assume that X is not a solution, that is, some matching edge $xy \in M^*$ is not clean. Then, $xy = uv$ since all other matching edges contain a cut. But uv is clean since $N(u) - v \subseteq X$. □

6 Computational Hardness

6.1 Hardness in Sparse Graphs

While Semi-Brutal Cut is known to be $\mathcal{NP}$-complete for both cut and weight score [36], we extend the cut-score hardness to planar, bipartite, subcubic graphs.

Theorem 7. Semi-Brutal Cut *is $\mathcal{NP}$-complete under both scores, even if the graph is planar, bipartite and subcubic.*

[7] A graph is *co-bipartite* if its vertices can be partitioned into two cliques.

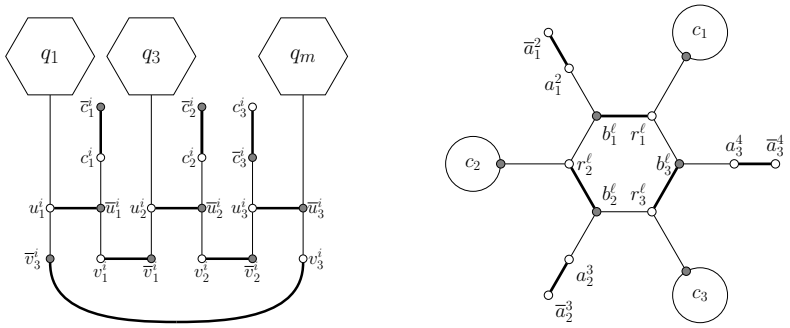


Fig. 5: Matching edges are bold. **Left:** variable gadget c_{x_i} linked to the clause gadgets q_1, q_3 and q_m, $m \notin \{1,3\}$, where x_i occurs positively in C_1 and C_3 and negatively in C_m. **Right:** clause gadget corresponding to the clause $C_\ell = (x_1 \vee \overline{x_2} \vee x_3)$.

To this end, we reduce the classic $\mathcal{NP}$-complete 3-SAT [13] problem to SBC.

3-Satisfiability (3-SAT)
Input: A boolean formula φ in conjunctive normal form where each clause contains exactly three literals.
Question: Is there a satisfying assignment β for φ?

Construction 1 *Let φ be an instance of* 3-SAT *with n variables $x_1, \ldots, x_n$ and m clauses $C_1, \ldots, C_m$. For each variable x_i, let ψ_i be the list of indices ℓ such that C_ℓ contains x_i and $|\psi_i|$ is the number of occurrences of x_i in φ. We construct the following solution graph (G^*, M^*, ω) with a proper 2-coloring of G^* (see Figure 5).*

- *For each x_i, we construct a cycle c_i on the vertex set $\bigcup_{j \leq |\psi_i|} \{u^i_j, \overline{u}^i_j, v^i_j, \overline{v}^i_j\}$ such that, for all $j \leq |\psi_i|$,*
  - *$u^i_j \overline{u}^i_j, v^i_j \overline{v}^i_j \in M^*$, and*
  - *the vertices u^i_j and v^i_j are blue and the vertices $\overline{u}^i_j$ and $\overline{v}^i_j$ are red.*
- *For each C_ℓ, we construct an alternating 6-cycle q_ℓ on the vertex set $\bigcup_{j \leq 3} \{r^\ell_j, b^\ell_j\}$ such that, for all $j \leq 3$, $\{r^\ell_j, b^\ell_j\} \in M^*$, and r^ℓ_j is red and b^ℓ_j is blue.*
- *For each clause C_ℓ and each $j \leq 3$, let x_i be the j^{th} literal of C_ℓ and let t be such that C_ℓ is the t^{th} clause in which x_i occurs. Then,*
  - *create two singles matching edges $a^i_t \overline{a}^i_t$ and $c^i_t \overline{c}^i_t$, where a^i_t and c^i_t are blue and $\overline{a}^i_t$ and $\overline{c}^i_t$ are red,*
  - *if x_i is a positive literal, introduce the edges $r^\ell_j u^i_t$, $b^\ell_j \overline{a}^i_t$ and $\overline{u}^i_t c^i_t$, and*
  - *if x_i is a negative literal, introduce the edges $b^\ell_j \overline{u}^i_t$, $r^\ell_j a^\ell_j$ and $u^i_t \overline{c}^i_t$.*
- *Each non-matching edge has weight one, except the edges $\overline{u}^i_t c^i_t$ and $u^i_t \overline{c}^i_t$ which have weight zero.*

Note that each matching edge except the $a^\ell_i \overline{a}^\ell_i$ and $c^i_t \overline{c}^i_t$ is ambiguous. Clearly, Construction 1 can be carried out in polynomial time. Further, the resulting

graph G^* is bipartite and $\Delta(G^*) = 3$. We first use Construction 1 on a restricted subcase of 3-SAT defined below.

MONOTONE 3-SATISFIABILITY (MONOTONE 3-SAT)
Input: A boolean formula φ in conjunctive normal form where each clause contains exactly three positive literals or three negative literals.
Question: Is there a satisfying assignment β for φ?

In order to prove Theorem 7, we use the following properties of Construction 1, yielding a "canonical" set of cuts, if the input formula is monotone.

Lemma 2. *Let $X \subseteq V(G^*)$ be a set of cuts cleaning all ambiguous edges in (G^*, M^*, ω), let c_i be a variable gadget and let q_ℓ be a clause gadget. Let $s = 1$ under the cut score and $s = 2$ under the weight score. We suppose that we start by cutting the vertices in the variable gadgets, and then we cut the vertices in the clause gadgets. There is a set X' of cuts with $score(X') \leq score(X)$ that also cleans all ambiguous edges and*
(a) $score(X' \cap V(c_i)) \geq s|\psi_i|$ and $score(X' \cap V(q_\ell)) \geq 2$,
(b) if $score(X' \cap V(c_i)) = s|\psi_i|$, then $X' \cap V(c_i)$ is either $\bigcup_{j \leq |\psi_i|}\{u^i_j\}$ or $\bigcup_{j \leq |\psi_i|}\{\overline{u}^i_j\}$ (in X', cuts are only on positive sides or only on negative sides),
(c) $score(X' \cap V(q_\ell)) = 2$ if and only if X' contains a vertex adjacent to q_ℓ (the score is two in a clause gadget iff it has been isolated by a cut in an adjacent variable gadget, meaning that the variable satisfies the clause).

Proof. (a): For each $j \leq |\psi_i|$, we need to remove two edges to clean the ambiguous edges $\{u^i_j, \overline{u}^i_j\}$, which can be done only by cutting at least one vertex among $\{\overline{v}^i_{j-1}, u^i_j, \overline{u}^i_j, v^i_j\}$. Thus, we need to remove at least $2|\psi_i|$ edges with at least $|\psi_i|$ cuts, that is $score(X' \cap V(c_i)) \geq s|\psi_j|$. In the clause q_ℓ, we need to remove at least two edges in the inner cycles, which can be done by cutting at least two vertices. Thus, we have $score(X' \cap V(q_\ell)) \geq 2$.
(b): Note that cutting all vertices in either $\bigcup_{j \leq |\psi_i|}\{u^i_j\}$ or $\bigcup_{j \leq |\psi_i|}\{\overline{u}^i_j\}$ suffices to remove all ambiguous path in c_i. In that case, we have $score(X' \cap V(c_i)) = s|\psi_i|$. If X contains some $\overline{u}^i_j$ and does not contain $\overline{u}^i_{j+1}$ for some j, then we need a extra cut to linearize $\{v^i_j, \overline{v}^i_j\}$ (and analogously for u^i_j) which will increase the score by one. Hence, if $|X \cap V(c_i)| = s|\psi_i|$, we can suppose that X contains either $\bigcup_{j \leq |\psi_i|}\{u^i_j\}$ or $\bigcup_{j \leq |\psi_i|}\{\overline{u}^i_j\}$. If X contains a cut in some v^i_j or some $\overline{v}^i_j$, then since the edge $\{v^i_j, \overline{v}^i_j\}$ is already clean by a cut in $\{\overline{u}^i_j, u^i_{j+1}\}$, we can can remove the cut in X'.

(c): We need to remove at least two non-zero weighted edges from the inner cycle of C_ℓ. Suppose that all literals of C_ℓ occurs positively. Suppose by symmetry that $\{b_1^\ell, b_2^\ell\} \in X'$. If the leaving edge incident to r_3^ℓ is cut, then all ambiguous edges of C_ℓ are destroyed. Otherwise, we need to remove one more non-zero weighted edge from q_ℓ which must add another cut (see Figure 6). □

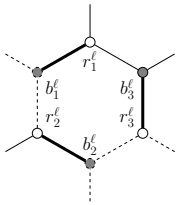


Fig. 6: A cut of size two in q_ℓ when one incident edge to q_ℓ is cut. Dashed edges and vertices are part of the cut.

We are now able to prove Theorem 7.

Proof. of Theorem 7

Recall that Monotone 3-SAT remains $\mathcal{NP}$-complete if the input formula is planar [2] and, in this case, since each gadget is planar and the edges between the clause gadget and the variable gadget can be placed in any order on the gadgets, the graph produced by Construction 1 can also be assumed to be planar. Since, clearly, Semi-Brutal Cut $\in \mathcal{NP}$, it remains to show that Construction 1 is correct, that is φ is satisfiable if and only if the solution graph (G^*, M^*, ω) resulting from Construction 1 can be linearized with a score of $(3s+2)m$.

"⇒": Let β be a satisfying assignment for φ. Then, for each variable x_i and for all $j \leq |\psi_i|$, we cut the vertices u_j^i if $\beta(x_i) = 1$ and the vertices $\overline{u}_j^i$ otherwise. As β is satisfying, this removes at least one edge adjacent to each clause gadget. Thus, according to Lemma 2(c), we can clean the matching edges in each clause gadget q_j with a score of two. Since we also cut either the vertices u_j^i or the vertices $\overline{u}_j^i$ for each vertex gadget, we conclude that all matching edges of the result are clean, and we have a score of $2m + \sum_i s|\psi_i| = (2+3s)m$.

"⇐": Let $X \subseteq V$ be the set of vertices such that cutting each vertex of X destroys all ambiguous paths in (G^*, M^*, ω) and $score(X) = (3s+2)m$. According to Lemma 2(a), each variable gadget has a score of $s|\psi_i|$ and each clause gadget has a score of two. Moreover, by Lemma 2(b), for each variable gadget c_i, we can suppose that $X \cap V(c_i)$ equals $\bigcup_{j \leq |\psi_i|} \{u_j^i\}$ or $\bigcup_{j \leq |\psi_i|} \{\overline{u}_j^i\}$. In the former case, we set $\beta(x_i) = 1$ and, in the latter, we set $\beta(x_i) = 0$. To show that β satisfies φ, assume that there is a clause C_ℓ that is not satisfied by β. Then, none of the edges incident to q_ℓ is cut which, by Lemma 2(c), contradicts the fact that the score of q_ℓ is equal to two. □

6.2 Hardness in Dense Graphs

We can see that if we add some zero weighted non-matching edges on a solution graph, it does not change the weight score of an optimal solution. This observation leads to the following result.

Corollary 2. *Let $\mathcal{G}$ be a class of graphs such that, for any planar, subcubic, bipartite graph G, $\mathcal{G}$ contains a supergraph of G. Then,* Semi-Brutal Cut *is $\mathcal{NP}$-complete on $\mathcal{G}$ under the weight score.*

Concerning the cut score, in some classes of dense graphs, SBC can be solved in polynomial time (see Section 5.2). However, we show in this part an example of a class of dense graphs where computing an optimal solution for SBC is $\mathcal{NP}$-hard under the cut score. A graph G is a *split graph* if we can partition its vertices into two sets I and C inducing an independent set and a clique, respectively. We show that SBC is hard to compute in split graphs by doing a reduction from the well-known Vertex Cover problem, defined below.

Vertex Cover (VC)
Input: An undirected graph G and a number $k \in \mathbb{N}$.
Question: Is there a $V' \subseteq V(G)$ with $|V'| \leq k$ intersecting all $e \in E(G)$?

Construction 2 *Let G be an instance of* Vertex Cover*, we suppose that G is connected. We construct the following solution graph G^* as follows:*

1. *for each vertex v of G, construct a matching edge v_1v_2,*
2. *for each edge uv of G, add the non-matching edges v_1u_2 and u_1v_2, and*
3. *for each pair of vertices (u, v), add the edge u_1v_1.*

The set of v_1 (resp. v_2) vertices form a clique (resp. is independent). Thus, G^* is a split graph. Note that all matching edges are ambiguous.

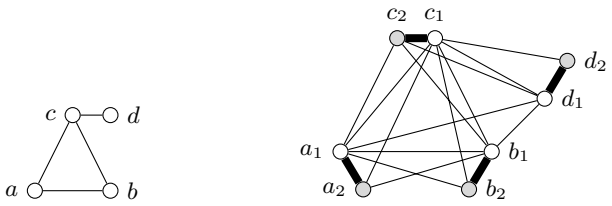


Fig. 7: Construction 2 transforms left instance into right instance, where gray vertices form an independent set and white vertices form a clique.

Theorem 8. Semi-Brutal Cut *is $\mathcal{NP}$-hard under the cut score on split graphs.*

Proof. We show that G has a size-k vertex cover if and only if using k cuts suffices to clean all matching edges in G^*.

"$\Rightarrow$": Let V' be a vertex cover of G. For each vertex $v \in V'$, cut the vertex v_1 in G^*. Suppose that there is a matching edge v_1v_2 that is not clean. There is an edge v_2u_1 that is not removed by a cut. Then, neither of the two vertices u and v belong to V' and the edge uv is not covered in G, contradicting the fact that V' is a vertex cover.

"$\Leftarrow$": Let X be a solution of SBC under the cut score for G^*. For each $v \in V(G)$, suppose that $v_2 \notin X$, since otherwise, $X' = (X \setminus \{v_2\}) \cup \{v_1\}$ is also a solution and $|X'| \leq |X|$. For each vertex $v_1 \in X$, add the vertex v in the vertex

cover of G. If there is an edge uv that is not covered, then $\{u_1, u_2, v_1, v_2\} \cup X = \varnothing$, and since (u_1, u_2, v_1, v_2) is a cycle, the matching edges u_1u_2 and v_1v_2 are not clean. □

Recall that Vertex Cover cannot be solved in $2^{o(n)}$ time unless ETH[8] fails [17]. Since Construction 2 is linear on vertices and edges, we obtain the following result.

Corollary 3. *There is no algorithm solving* Semi-Brutal Cut *with cut score in* $2^{o(n)}$ *in split graphs.*

7 Non-Approximability

In this section, we prove approximation lower bounds for Semi-Brutal Cut. First recall the definition of *L-reduction* between two hard problems Π and Π', described by Papadimitriou and Yannakakis [28]. This reduction consists of polynomial-time computable functions f and g such that, for each instance x of Π, $f(x)$ is an instance of Π' and for each feasible solution y' for $f(x)$, $g(y')$ is a feasible solution for x. Moreover, let $\Pi'' \in \{\Pi, \Pi'\}$, we denote by $OPT_{\Pi''}$ the value of an optimal solution of Π'' and by $val_{\Pi''}(y'')$ the value of a solution y'' of an instance of Π''. There are constants $\alpha, \beta > 0$ such that:

1. $OPT_{\Pi'}(f(x)) \leq \alpha OPT_{\Pi}(x)$ and
2. $|val_{\Pi}(g(y')) - OPT_{\Pi}(x)| \leq \beta |val_{\Pi'}(y') - OPT_{\Pi'}(f(x))|$.

7.1 Reduction from Max 3-SAT(4)

In the following, we present an *L-reduction* from the classical problem Max 3-SAT(4) to Semi-Brutal Cut under the cut score.

Max 3-SAT(4)
Input: A boolean formula φ in exact 3-CNF where every variable occurs in four clauses.
Task: Find an assignment that satisfies a maximum number of clauses.

Our goal is to reuse Construction 1 to reduce Max 3-SAT(4), such that each unsatisfied clause in ϕ causes an additional cut in G^*. Indeed, if there is no optimal solution with a score of $5m$ in G^* (that is, if φ can not be satisfied), then we can spend an "extra" cut per unsatisfied clause to solve G^*. The inverse, however, does not hold if there is a variable x_i that occurs two times positively and two times negatively. Indeed, by cutting five vertices in C_i, we may be able to satisfy the four clauses where x_i occurs (see Figure 8). Thus, in the following, we modify Construction 1 slightly.

[8] The (widely believed) "Exponential Time Hypothesis" (ETH) states that the boolean satisfyability problem (SAT) cannot be solved in $2^{o(n)}$ time, where n is the number of variables of the input formula.

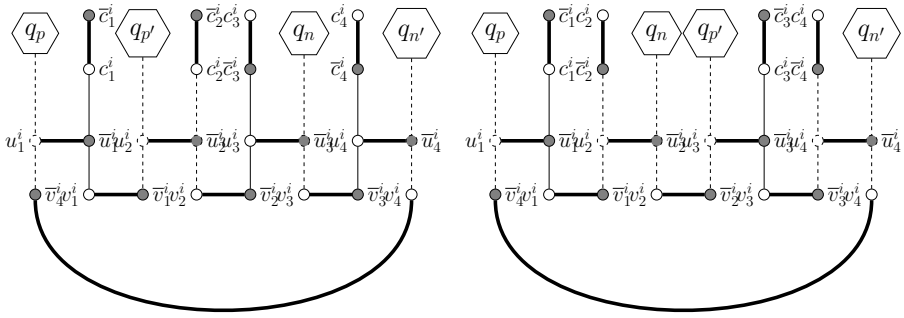


Fig. 8: Matching edges are bold. Example of variable gadget r_i which occurs two times positively and two times negatively in Construction 1 (**Left**) and Construction 3 (**Right**). The cut-vertices are dashed. We can see that we need to add a cut in Construction 3 in order to remove all the edges leaving the gadget.

Construction 3 *We reuse Construction 1 and change some variable gadgets. Let x_i be a variable which two times positively and two times negatively. Before building the gadget c_i, we modify the clauses order in ψ_i by interleaving positive and negative clauses. Other variable gadgets remain unchanged.*

The resulting graph G^* is bipartite and subcubic. An example of a variable gadget defined in Construction 3 is given in Figure 8. Notice that, if we do not take in account the weight on the edges, all clauses are symmetric. Thus, the properties (a) and (c) of Lemma 2 hold. We can add the following property:

Lemma 3. *Let $X \subseteq V(G^*)$ be an optimal set of cuts that cleans all matching edges in (G^*, M^*, ω), let c_i be a variable gadget. There is a set X' of cuts with $score(X') = score(X)$ that also clean all matching edges, and $X' \cap V(c_i)$ is either $\bigcup_{j \leq |\psi_i|} \{u_j^i\}$ or $\bigcup_{j \leq |\psi_i|} \{\overline{u}_j^i\}$.*

Proof. Recall that X covers the edges of M^* and, by Lemma 2(a), $score(X \cap V(c_i)) \geq |\psi_i|$. By symmetry, suppose that x_i occurs mostly positively in φ. If x_i occurs four times positively, then replacing $X \cap V(c_i)$ by $\bigcup_{j \leq |\psi_i|} \{u_j^i\}$ in X yields a solution X' as sought. Thus, suppose that x_i occurs three times positively. Let C_ℓ be the clause where x_i occurs negatively and let z denote the neighbor of $\overline{u}_j^i$ in c_ℓ. If $score(X \cap V(c_i)) > |\psi_i|$, then replacing $X \cap c_i$ by $\bigcup_{j \leq |\psi_i|} \{u_j^i\}$ plus z yields a solution X' as sought. Finally, if $score(X \cap V(c_i)) = |\psi_i|$, then X already corresponds to X' as, otherwise, some ambiguous edge $v_j^i \overline{v}_j^i$ is not clean.

Suppose now that x_i occurs two times positively and two times negatively. Note that one cut in $r_{i'}$ is not enough to clean all ambiguous edges and cutting either the vertices $\{u_1^{i'}, u_2^{i'}\}$ or the vertices $\{\overline{u}_1^{i'}, \overline{u}_2^{i'}\}$ cleans all matching edges in the variable gadget. Further if X cuts $\{v_1^{i'}, v_2^{i'}\}$ or $\{\overline{v}_1^{i'}, \overline{v}_2^{i'}\}$, then we can instead cut $\{u_1^{i'}, u_2^{i'}\}$ or $\{\overline{u}_1^{i'}, \overline{u}_2^{i'}\}$, respectively, without creating ambiguous edges. Suppose without loss of generality that $\{u_1^{i'}, u_2^{i'}\} \subseteq X$. Suppose further that there is

some $\overline{u} \in X \cap V(r_{i'}) \setminus \{u_1^{i'}, u_2^{i'}\}$. Then, there is some clause gadget q_n linked to $\overline{u}$ since, otherwise, $X \setminus \{\overline{u}\}$ is also a solution, contradicting optimality of X. Since all matching edges of $r_{i'}$ are already clean, the cut can only remove the edge between $\overline{u}$ and q_n. Let z be the neighbor of $\overline{u}$ in q_n. In X, the two non-matching edges incident to z must be removed, otherwise it contradicts the optimality of X. Thus, we can replace $\overline{u}$ by its neighbor in q_n without changing the score of X. By swapping the one or two cuts in $X \cap V(r_{i'}) \setminus \{u_1^{i'}, u_2^{i'}\}$, we obtain $X' \cap V(r_j) = \{u_1^{i'}, u_2^{i'}\}$. □

Theorem 9. *It is NP-hard to approximate* Semi-Brutal Cut *to any factor better than* $1 + \frac{7(\epsilon_4 - 1)}{41 \cdot \epsilon_4}$ *under the cut score, even on subcubic bipartite graphs.*

Proof. First, note that it is $\mathcal{NP}$-hard to approximate Max 3-SAT(4) to any factor $\epsilon_4 \leq 1.00052$, unless $\mathcal{P} = \mathcal{NP}$ [4]. Recall that in an optimal solution of Max 3-SAT(4), at least $7/8$ of the clauses are satisfied [15], yielding

$$OPT(\varphi) \geq 7m/8. \tag{1}$$

To show that Construction 3 constitutes an L-reduction, let f be a function transforming any instance φ of Max 3-SAT(4) into an instance I of Semi-Brutal Cut as above, let X be a feasible solution for I corresponding to the properties of Lemma 2(a), Lemma 2(c) and Lemma 3, and let g be the function that transforms X into an assignment as constructed in the proof of Theorem 7: each variable x_i is set to true if X cuts u_j^i for all j, and false, otherwise. By Lemma 3, for each clause gadget q_ℓ without an adjacent vertex in X, the "extra" cut occurs in q_ℓ. Hence, we can linearize I with one more cut for each of the at most $m/8$ unsatisfied clauses in φ. Thus,

$$OPT(I) \leq 5m + m/8 \overset{(1)}{\leq} 41/7 OPT(\varphi) \tag{2}$$

An important obstacle to overcome (and reason why Construction 1 is not enough for Theorem 9) is that an approximate solution to SBC might spend extra cuts in variable gadgets in order to "change the assignment" of a variable x_i mid-way. However, since each variable occurs at most four times, this only happens for variables that occur two times positively and two times negatively. Now, with our modification to Construction 1 and by Lemma 3, we can observe that each extra cut occurs in an unsatisfied clause gadget. Thus, the number of satisfied clauses of φ and the clause gadgets in which we have to spend extra cuts add up to m. Hence,

$$6m = val(g(X)) + val(X) = OPT(I) + OPT(\varphi) \tag{3}$$

Thus, we constructed an L-reduction with $\alpha = 41/7$, $\beta = 1$ and, since $\epsilon_4 \cdot val(g(X)) \leq OPT(\varphi)$, we conclude

$$\begin{aligned} val(X) &\overset{(3)}{=} OPT(I) + OPT(\varphi) - val(g(X)) \\ &\geq OPT(I) + (1 - 1/\epsilon_4) \cdot OPT(I) \\ &\overset{(2)}{\geq} (1 + \frac{7(\epsilon_4 - 1)}{41 \cdot \epsilon_4}) \cdot OPT(I) \end{aligned}$$

□

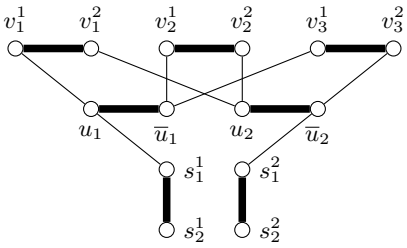


Fig. 9: The graph produced by Construction 4 and on input $\varphi = (x_1 \vee x_2) \wedge (\neg x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2)$. Matching edges are bold and all non-matching edges have weight one.

Note that, by losing the bipartition property, we can use Construction 3 to show that it is hard to approximate SBC on subcubic graphs to any factor better than $^{(7(\epsilon_4-1))}/_{(65\cdot\epsilon_4)}) \approx 1.000056$ using [31]. However, we show in the next subsection how to obtain a better lower bound under the weight score for such graphs.

7.2 Reduction from MAX 2-SAT

We now present an L-reductions from the classical problem MAX 2-SAT to Semi-Brutal Cut under both scores.

MAX 2-SAT (Max 2-SAT)
Input: A boolean formula φ in conjunctive normal form where each clause C_i contains exactly two variables.
Question: Find an assignment maximizing the number of satisfied clauses.

Let φ be an instance of MAX 2-SAT with n variables $x_1, \ldots, x_n$ and m clauses $C_1, \ldots, C_m$. For each variable x_i, let ψ_i be the list of indices ℓ such that C_ℓ contains x_i. Let (G^*, M^*, ω) be a solution graph and u be a vertex of G^*, we denote by $\omega(u)$ the sum of the weight of the non-matching edges incident to u.

Construction 4 *Let φ be an instance of* MAX 2-SAT. *We construct the following solution graph (G^*, M^*, ω).*

1. *For each x_i, construct a matching edge $u_i\overline{u}_i$ (*variable edge*).*
2. *For each clause C_j, construct a matching edge $v_j^1 v_j^2$ (*clause edge*).*
3. *For each clause C_j, let x_k be the t^{th} variable of the clause. If x_k occurs positively in the clause, then add the edge $v_j^t u_k$ and $\omega(v_j^t u_k) = 1$. Otherwise, add the edge $v_j^t \overline{u}_k$ with $\omega(v_j^t \overline{u}_k) = 1$.*
4. *Finally, for each variable matching edge $u_i\overline{u}_i$, add a matching edge $s_1^i s_2^i$. If $\omega(u_i) < \omega(\overline{u}_i)$, add an edge $s_1^i u_1$ with $\omega(s_1^i u_1) = \omega(\overline{u}_i) - \omega(u_i)$. If $\omega(u_i) > \omega(\overline{u}_i)$, add an edge $s_1^i \overline{u}_1$ with $\omega(s_1^i \overline{u}_1) = \omega(u_i) - \omega(\overline{u}_i)$.*

We can suppose that no variable occurs exclusively positively or exclusively negatively in the formula, thus each matching edge except the $s_1^i s_2^i$ is ambiguous.

An example of a graph produced by Construction 4 is given in Figure 9. A *normalized* solution is a solution that contains exactly one cut per ambiguous edge. The following lemma shows that we can transform any solution into a normalized solution with the same weight score.

Lemma 4. *Let X be a solution of a solution graph (G^*, M^*, ω). There is a normalized solution X' with $score(X') \leq score(X)$ under the weight score.*

Proof. Since X is a solution of SBC, after removing all non-matching edges incident to a cut, all ambiguous edges are clean. We construct X' by choosing one degree-one vertex per ambiguous edge. Clearly, X' is a solution and since each edge removed by X' is also removed by X, we have $score(X') \leq score(X)$.

In this, we suppose that each solution is normalized under the weight score. If a cut in a clause edge $v_j^1 v_j^2$ is adjacent to a cut in a variable edge, then we say that the clause edge $v_j^1 v_j^2$ is *satisfied*. If no extremity of a clause edge $v_j^1 v_j^2$ is adjacent to a cut in a variable edge, we say that the clause edge $v_j^1 v_j^2$ is *unsatisfied*.

Definition 2. *Let φ be a* MAX 2-SAT *instance, let (G^*, M^*, ω) be the graph produced by Construction 4, and let X be a normalized solution for it. An assignment S for φ* corresponds *to X if, for all matching edges $u_i\overline{u}_i$, we have $u_i \in X \Rightarrow S(x_i) = 1$ and $\overline{u}_i \in X \Rightarrow S(x_i) = 0$.*

Lemma 5. *Let X be a normalized solution for (G^*, M^*, ω), produced by Construction 4 and let S be its corresponding assignment. Let m' be the number of unsatisfied clauses in S. There is a solution X' such that $score(X') = 2m + m' \leq score(X)$ and S is the corresponding assignment of X'.*

Proof. Suppose there is a clause edge $v_j^1 v_j^2$ that is neither satisfied nor unsatisfied. Thus, there is a cut vertex adjacent to $v_j^1 v_j^2$ that is not adjacent to the cut vertex of $v_j^1 v_j^2$. Suppose that the cut vertex in $v_j^1 v_j^2$ is v_j^1. We can take $X' = X \cup \{v_j^2\} - v_j^1$. Since the edge incident to v_j^2 is already removed, we have $score(X') \leq score(X)$. S is the corresponding assignment of X' since we do not change the cuts in the variable edges. Hence, we can suppose that X' does not contain any clause edge that is neither satisfied nor unsatisfied.

Let $u_i\overline{u}_i$ be a variable edge. We have $\omega(u_i) = \omega(\overline{u}_i) = |\psi_i|$. Thus, the sum of the weight removed by the cuts in the variable edges is equal to $\sum_{i \leq n} |\psi_i| = 2m$. Let $v_j^1 v_j^2$ be a clause edge. If $v_j^1 v_j^2$ is satisfied, then its cut does not increase the weight of X' since the non-matching edge incident to this cut is already removed by the cut in the variable edge. If $v_j^1 v_j^2$ is unsatisfied, then the cut in $\{v_j^1, v_j^2\}$ increases by one the weight of X'. Since the sum of weight removed by the cuts in the unsatisfied clause-edges correspond to the number of unsatisfied clause, we have $score(X') = 2m + m'$. □

Theorem 10. Semi-Brutal Cut *cannot be approximated to any factor better than $1 + \frac{\epsilon_2 - 1}{3 \cdot \epsilon_2}$ under the weight score.*

We have $\epsilon_2 \approx 1.06$, if the Unique Game Conjecture is true [19] and $\epsilon_2 = 22/21$, if $\mathcal{P} \neq \mathcal{NP}$ [15].

Proof. First, we can see that a random assignment satisfies each clause with probability $3/4$ and hence it is not hard to find an assignment that satisfies $3m/4$ clauses, yielding

$$OPT(\varphi) \geq 3m/4. \tag{4}$$

Similarly to proof of Theorem 9, we show that Construction 4 constitutes an L-reduction. Let f be a function transforming any instance φ of MAX 2-SAT into an instance I of Semi-Brutal Cut as above. Let X be a feasible normalized solution for I corresponding to the property of Lemma 5. And let g be the function that construct the assignment of φ which corresponds to X. Hence,

$$OPT(I) \leq 2m + m/4 \overset{(4)}{\leq} 3 \cdot OPT(\varphi), \tag{5}$$

and by Lemma 5, we have

$$3m = val(g(X)) + val(X) = OPT(I) + OPT(\varphi). \tag{6}$$

Thus, we constructed an L-reduction with $\alpha = 3$ and $\beta = 1$. Since $\epsilon_2 \cdot val(g(X)) < OPT(\varphi)$, we conclude

$$val(X) \geq (1 + \frac{\epsilon_2 - 1}{3 \cdot \epsilon_2}) \cdot OPT(I) \qquad \square$$

MAX 2-SAT(3) is restricted subproblem of MAX 2-SAT where the number of occurrences of the variable is bounded by three. Berman and Karpinski [3] show that MAX 2-SAT(3) cannot be approximated to a factor better than $\epsilon_2' \leq 2012/2011$ unless $\mathcal{P} = \mathcal{NP}$. In that subproblem, the maximum degree of the graph provided by Construction 4 is restricted to three. Using the same arguments as for Theorem 10, we obtain:

Theorem 11. *It is $\mathcal{NP}$-hard to approximate* Semi-Brutal Cut *within any ratio better than $(1 + \frac{\epsilon_2' - 1}{3 \cdot \epsilon_2'})$ under the weight score, even on subcubic graphs.*

Samewise, we can use the inapproximability result on MAX 2-SAT(3) to build a L-reduction to Semi-Brutal Cut under the cut score on subcubic graphs.

Theorem 12. *It is $\mathcal{NP}$-hard to approximate* Semi-Brutal Cut *within any ratio better than $(1 + \frac{9(\epsilon_2' - 1)}{11 \cdot \epsilon_2'})$ under the cut score, even on subcubic graphs.*

Proof. Let (G^*, M^*, ω) be a solution graph produced by Construction 4 and let X be an optimal solution in (G^*, M^*, ω) for SBC under the cut score. Let $u_i\overline{u}_i$ be a variable edge, suppose by symmetry that x_i occurs two times positively and one time negatively. Suppose that $u_i\overline{u}_i$ does not contain a cut. If the neighbor $v_{j'}^{t'}$ of $\overline{u}_i$ belongs to X, then we can swap $v_{j'}^{t'}$ and $\overline{u}_i$ in X. Otherwise, the two neighbors v_j^t and $v_{j'}^{t'}$ belong to X and then the set of cuts $X' = X \cup \{u_i\} \setminus \{v_j^t, v_{j'}^{t'}\}$ is also a solution of SBC and $|X'| < |X|$, contradicting the optimality of X. Thus, each variable edge contains at least one cut. Each cut in a variable edge can clean

up to two clause edges. If a variable edge contains two cuts, then one of these cuts only serves to clean one clause edge $v_j^1 v_j^2$, and then we can transfer this cut into $v_j^1 v_j^2$. Hence, we can suppose that each variable edge contains exactly one cut. Then, the number of cuts in the clause edges corresponds to the number of unsatisfied clauses in the corresponding assignment of X. Note that, if a set of cuts is not optimal, it is easy to transform it into another set with a better cut-score and such that each variable edge contains exactly one cut. The number of variables in a MAX 2-SAT(3) is equal to $2m/3$. Let f denote the function that transforms any instance φ of MAX 2-SAT(3) into an instance of Semi-Brutal Cut as in Construction 4. Let X be a solution of SBC in I that contains exactly one cut per variable edge and let g be the function that construct the assignment of φ which corresponds to X. We have,

$$OPT(I) \leq 2m/3 + m/4 \overset{(4)}{\leq} 11/9 \cdot OPT(\varphi), \tag{7}$$

and

$$\frac{5m}{3} = val(g(X)) + val(X) = OPT(I) + OPT(\varphi). \tag{8}$$

We constructed an L-reduction with $\alpha = 11/9$ and $\beta = 1$ and we obtain:

$$val(X) \geq (1 + \frac{9(\epsilon_2' - 1)}{11 \cdot \epsilon_2'}) \cdot OPT(I) \qquad \square$$

7.3 Strict Reduction from Vertex Cover

Strict-reduction is the simplest type of approximation-preserving reduction [9]. In a strict reduction, the approximation ratio $\rho_{\Pi'}$ of a solution y to an instance $f(x)$ of a problem Π' must be at most as good as the approximation ratio ρ_Π of the corresponding solution $g(y)$ to instance x of problem Π. In other words:

$$\rho_{\Pi'}(f(x), y) \leq \rho_\Pi(x, g(y)).$$

The proof of Theorem 8 shows that Construction 2 is a strict reduction from Vertex Cover, which leads to the following result.

Theorem 13. *If* Vertex Cover *can not be approximated to a ratio better than ρ, then neither can* Semi-Brutal Cut *on split graphs under the cut score.*

In order to find an inapproximability result on the general case for SBC under the cut score, we reduce Vertex Cover problem, defined in Section 6.2 and we use the following construction.

Construction 5 *Let G be an instance of VC, create a solution graph G^* as follows: for each $v \in V(G)$, add a new path $vv^1v^2v^3$ and set $vv^1, v^2v^3 \in M^*$. Call the resulting graph G^* and note that $E(G^*) \supseteq E(G)$ and the ambiguous edges of G^* are exactly the edges vv^1 and $\Delta(G^*) = \Delta(G) + 1$.*

Theorem 14. *If* Vertex Cover *can not be approximated to a ratio better than ρ on graphs with bounded degree Δ, then neither can* Semi-Brutal Cut *on graph with bounded degree $\Delta + 1$ under the cut score.*

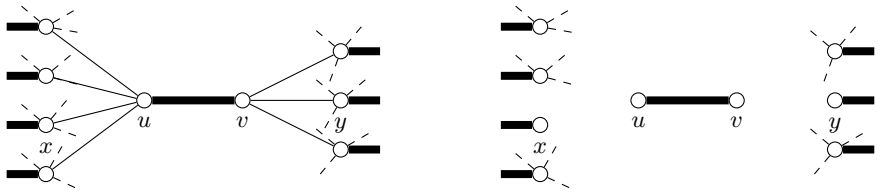


Fig. 10: A forbidden path $xuvy$ (left) and the result of cutting all its vertices (right).

Proof. We show that G has a size-k vertex cover if and only if using k cuts suffices to clean all matching edges in G^*.

"$\Rightarrow$": Let V' be a vertex-cover of G. Then, cutting all vertices of V' in G^* leaves no edge of $E(G)$. The remaining graph is a collection of alternating paths of length three and, thus, all matching edges are clean.

"$\Leftarrow$": Let X be a solution of SBC under the cut score for G^*. Let $Y := \{v \mid \{v, v^1, v^2, v^3\} \cap X \neq \varnothing\}$ and note that $|Y| \leq |X|$. Now, if Y is not a vertex cover of G, then there is an edge $uv \in E(G)$ such that $Y \cap uv = \varnothing$. Then, none of $\{u^3, u^2, u^1, u, v, v^1, v^2, v^3\}$ is cut, implying that neither uu^1 nor vv^1 is clean, contradicting the fact that X is a solution of SBC.

Hence, Construction 5 is a strict reduction, transferring non-approximability results of Vertex Cover to Semi-Brutal Cut under the cut score. □

Vertex Cover is also non-approximable within a factor of 1.3606 under $\mathcal{NP} \neq \mathcal{P}$ [11] and within a factor $2-\epsilon, \epsilon > 0$ under $\mathcal{UGC}$ [18]. Let G be an instance of VC, the maximum degree of the graph produced by Construction 5 is equal to $\Delta(G)+1$. Berman and Karpinski [3] show that if the instance of VC has a maximum degree three, four or five, then VC can note be approximated to a ratio better than $^{145}/_{144}, ^{79}/_{78}, ^{74}/_{73}$, respectively. Thus, this results hold for SBC under the cut score, for solution graphs with maximum degree four, five and six, respectively.

8 Approximable Cases

In this section, we propose one greedy approximation algorithm for each score.

Cut score: Our approximation algorithm works similarly to the well-known classical 2-approximation for Vertex Cover that just returns the extremities of any maximal matching. Contrary to Vertex Cover, our forbidden structures are not edges, but ambiguous edges. Thus, we have to consider length-four paths containing an ambiguous edge, and we will cut all four of their vertices. In the following, we call a path $xuvy$ *forbidden* if xu and vy are non-matching edges and uv is an ambiguous edge (see Figure 10).

Lemma 6. *Let Q be a maximal packing of vertex-disjoint forbidden paths in (G^*, M^*, ω), let X be any solution for SBC under the cut score on (G^*, M^*, ω). Then, (a) cutting all vertices of Q cleans all ambiguous edges in G^* and (b) $X \cap p \neq \varnothing$ for all $p \in Q$.*

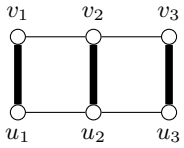


Fig. 11: Tightness of the approximation ratio for the cut-score greedy algorithm. Matching edges are bold. The approximation algorithm for the cut score provides a solution $\{v_1, v_2, u_2, u_3\}$ whereas an optimal solution is $\{v_2\}$.

Proof. (a): Let H be the result of cutting all vertices of Q in G^*. Towards a contradiction, assume that H contains an ambiguous edge uv. By definition, there are two non-matching edges xu and vy in H. But then, the path $xuvy$ is a forbidden path, contradicting the maximality of Q.

(b): Let H be the result of cutting all vertices of X in G^*. Let $xuvy \in Q$ be a forbidden path in (G^*, M^*, ω) and assume towards a contradiction that $X \cap xuvy = \varnothing$. Then, none of the edges of $xuvy$ are removed when cutting the vertices of X, that is, $xuvy$ survives in H. Then, however, uv is ambiguous in H, contradicting X being a solution for (G^*, M^*, ω). □

With Lemma 6, we can show that any maximal packing of forbidden paths constitutes a 4-approximation for Semi-Brutal Cut under the cut score.

Theorem 15. *A 4-approximate solution to* Semi-Brutal Cut *under the cut score can be computed in linear time. This ratio is tight.*

Proof. A packing of forbidden paths in (G^*, M^*, ω) can be computed by scanning all matching edges uv and, if uv is ambiguous, then $xuvy$ is a forbidden path for any non-matching edges xu and vy. By removing x, u, v, and y from G^*, we make sure that the resulting packing is vertex-disjoint. Thus, such a packing can be produced in linear time.

Let Q be any maximal vertex-disjoint packing of forbidden paths in (G^*, M^*, ω). By Lemma 6(a), the vertices of Q form a solution for SBC. To show that this solution is 4-approximate, consider any optimal solution X for (G^*, M^*, ω). By Lemma 6(b), X intersects each path in Q. Since the paths in Q are mutually vertex disjoint and each of them contains exactly four vertices, we conclude that Q contains at most four times as many vertices as X. The ratio is tight, as shown by Figure 11. □

Weight score: Let (G^*, M^*, ω) be a solution graph and let $X \subseteq E \setminus M^*$ be a set of non-matching edges. For a vertex v, we let $\omega_X(v)$ denote the sum of the weights of all non-matching edges incident with v that are not in X. More formally, we define $\omega_X(v) := \sum_{e \in E \setminus (M^* \cup X)} \omega(e) \cdot \chi_e(v)$, where $\chi_e(v) := |e \cap \{x\}|$ is the characteristic function of e. The principle of our algorithm is to successively visit each ambiguous edge and cut the edges incident to the extremity with the lowest value of w_S, where S contains all previously cut edges.

Algorithm 1: Greedy Algorithm for the weight score

Data: A solution graph (G^*, M^*, ω).
Result: A set $X \subseteq E \setminus M^*$ whose removal cleans all matching edges.

```
X ← ∅ ;
A ← list of extremities of ambiguous edges;
while A ≠ ∅ do
    u ← argmin_{x∈A} ω_X(x);
    remove the two extremities of the ambiguous edge containing u from A;
    add all non-matching edges incident with u to X;
return X;
```

Fig. 12: Tightness of the approximation ratio for the weight-score greedy algorithm. Edges are bold ($\in M^*$), solid ($\in X_{opt}$) or dashed ($\in X$) and all edges have weight one. Thus, $\omega(X) = 2$ and $\omega(X_{opt}) = 1$.

Theorem 16. *In $\mathcal{O}((|V| + |E|) \log |V|)$ time, Algorithm 1 computes a solution for* Semi-Brutal Cut *under the weight score with an approximation ratio of 2 and this ratio is tight.*

Proof. Since each time some extremities are removed from A, the ambiguous edge they belonged to has been cleaned, there are no more ambiguous edge remaining when $A = \varnothing$. Thus, the set X that is returned is indeed a solution. Let X_{opt} be an optimal solution. Let uv denote the ambiguous edge of G^* considered in step i of Algorithm 1, let X_i be the set of edges added to X in step i. If X_{opt} contains all non-matching edges incident to u, then let Q_i contains them. Otherwise, X_{opt} contains all non-matching edges incident to v, and we let Q_i contain those. Then, $\omega(X_i) \leq \omega(Q_i)$ for all i and, thus, $\omega(X) \leq \sum_i \omega(Q_i)$. Further, $\bigcup_i Q_i = X_{opt}$ and, since each edge of G^* occurs in at most two sets Q_i, we conclude $\sum_i \omega(Q_i) \leq 2\omega(X_{opt})$. The claimed approximation factor of two follows and, by Figure 12, it is tight.

Concerning the running time, the list of ambiguous edges is build in $\mathcal{O}(|E| + |V|)$ with a depth-first search algorithm. The sorting of this list can be done in $\mathcal{O}(|V| \log |V|)$. Maintaining the sorting of the list at each cut yields a $\mathcal{O}((|V| + |E|) \log |V|)$. □

Corollary 4. Semi-Brutal Cut *is $\mathcal{APX}$-complete under both scores.*

9 Conclusion

In this paper, we present complexity and approximation results obtained for a theoretical problem occurring in modern-day production of genomic sequences. We consider two variants of the problem, depending on the optimality criterion (number of cuts vs. weight of cuts). We show that the complexity of both variants

depend heavily on the input topology. To this end, we explore the demarcation line between polynomial-time computability and $\mathcal{NP}$-hard cases for sparse and dense classes of graphs. Finally, we present simple constant-factor approximation algorithms for both optimization goals. Interesting open questions include the existence and design of efficient FPT algorithms, and/or kernel techniques.

Acknowledgement: This work was partially supported by the Région Occitanie.

References

[1] Y. Anselmetti, V. Berry, C. Chauve, A. Chateau, E. Tannier, and S. Bérard. Ancestral gene synteny reconstruction improves extant species scaffolding. *BMC Genomics*, 16(10):S11, 2015.

[2] Mark De Berg and Amirali Khosravi. Optimal binary space partitions for segments in the plane. *Int. J. Comput. Geometry Appl.*, 22(3):187–206, 2012.

[3] P. Berman and M. Karpinski. On some tighter inapproximability results (extended abstract). In *Proceedings of the 26th International Colloquium on Automata, Languages and Programming*, pages 200–209, 1999.

[4] P. Berman, M. Karpinski, and A. D. Scott. Approximation hardness and satisfiability of bounded occurrence instances of SAT. *Electronic Colloquium on Computational Complexity (ECCC)*, 10(022), 2003.

[5] J. N. Burton, A. Adey, R. P. Patwardhan, R. Qiu, J. O. Kitzman, and J. Shendure. Chromosome-scale scaffolding of de novo genome assemblies based on chromatin interactions. *Nature biotechnology*, November 2013. ISSN 1546-1696.

[6] M. D. Cao, S. H. Nguyen, D. Ganesamoorthy, A. G. Elliott, M. A. Cooper, and L. J. M. Coin. Scaffolding and completing genome assemblies in real-time with nanopore sequencing. *Nature Communications*, 8:14515, February 2017.

[7] A. Chateau and R. Giroudeau. A complexity and approximation framework for the maximization scaffolding problem. *Theoretical Computer Science*, 595:92–106, 2015.

[8] R. Chikhi and G. Rizk. Space-efficient and exact de bruijn graph representation based on a bloom filter. *Algorithms for Molecular Biology*, 8:22, 2013.

[9] P. Crescenzi. A short guide to approximation preserving reductions. In *Proceedings of the Twelfth Annual IEEE Conference on Computational Complexity, Ulm, Germany, June 24-27, 1997*, pages 262–273, 1997.

[10] A. Dayarian, T. P. Michael, and A. M. Sengupta. SOPRA: Scaffolding algorithm for paired reads via statistical optimization. *BMC Bioinformatics*, 11:345, 2010.

[11] I. Dinur and S. Safra. On the hardness of approximation minimum vertex cover. *Annals of Mathematics*, 162(1):439–485, 2005.

[12] N. Donmez and M. L. Brudno. SCARPA: scaffolding reads with practical algorithms. *Bioinformatics*, 29(4):428–434, 2013.

[13] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.

[14] A. A. Gritsenko, J. F. Nijkamp, M. J. T. Reinders, and D. de Ridder. GRASS: a generic algorithm for scaffolding next-generation sequencing assemblies. *Bioinformatics*, 28(11):1429–1437, 2012.

[15] J. Håstad. Some optimal inapproximability results. *J. ACM*, 48(4):798–859, 2001.
[16] M. Hunt, C. Newbold, M. Berriman, and T. Otto. A comprehensive evaluation of assembly scaffolding tools. *Genome Biol*, 15(3):42, 2014.
[17] R. Impagliazzo, R. Paturi, and F. Zane. Which Problems Have Strongly Exponential Complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001.
[18] S. Khot and O. Regev. Vertex cover might be hard to approximate to within 2-epsilon. *J. Comput. Syst. Sci.*, 74(3):335–349, 2008.
[19] S. Khot, G. Kindler, E. Mossel, and R. O'Donnell. Optimal inapproximability results for MAX-CUT and other 2-variable csps? *SIAM J. Comput.*, 37(1):319–357, 2007.
[20] R. Kolodner and K. K. Tewari. Inverted repeats in chloroplast DNA from higher plants*. *Proceedings of the National Academy of Sciences of the United States of America*, 76(1):41–45, 1979.
[21] S. Koren, T. J. Treangen, and M. Pop. Bambus 2: Scaffolding metagenomes. *Bioinformatics*, 27(21):2964–2971, 2011.
[22] E. Lerat. Identifying repeats and transposable elements in sequenced genomes: how to find your way through the dense forest of programs. *Heredity*, 104(6):520–533, June 2010.
[23] I. Mandric and A. Zelikovsky. ScaffMatch: scaffolding algorithm based on maximum weight matching. *Bioinformatics*, 31(16):2632–2638, 04 2015. ISSN 1367-4803.
[24] I. Mandric, J. Lindsay, I. I. Măndoiu, and A. Zelikovsky. Scaffolding algorithms. In I. Măndoiu and A. Zelikovsky, editors, *Computational Methods for Next Generation Sequencing Data Analysis*, chapter 5, pages 107–132. Wiley, 2016.
[25] J. R. Miller, S. Koren, and G. Sutton. Assembly algorithms for next-generation sequencing data. *Genomics*, 95(6):315–327, 2010.
[26] Marcos Morey, Ana Fernández-Marmiesse, Daisy Castiñeiras, José M. Fraga, María L. Couce, and José A. Cocho. A glimpse into past, present, and future dna sequencing. *Molecular Genetics and Metabolism*, 110(1):3 – 24, 2013. ISSN 1096-7192. Special Issue: Diagnosis.
[27] Y. Mostovoy, M. Levy-Sakin, J. Lam, E. T. Lam, A. R. Hastie, P. Marks, J. Lee, C. Chu, C. Lin, Z. Dzakula, H. Cao, S. A. Schlebusch, K. Giorda, M. Schnall-Levin, J. D. Wall, and P.-Y. Kwok. A hybrid approach for de novo human genome sequence assembly and phasing. *Nat Meth*, 13(7):587–590, July 2016.
[28] C. H. Papadimitriou and M. Yannakakis. Optimization, approximation, and complexity classes. *Journal of Computer and System Sciences*, 43(3):425–440, 1991.
[29] Adam M. Phillippy. New advances in sequence assembly. *Genome Research*, 27 (5), May 2017.
[30] K. Sahlin, F. Vezzi, B. Nystedt, J. Lundeberg, and L. Arvestad. BESST - efficient scaffolding of large fragmented assemblies. *BMC Bioinformatics*, 15(1):281, 2014.
[31] Dorine Tabary, Tom Davot, Mathias Weller, Annie Chateau, and Rodolphe Giroudeau. New results about the linearization of scaffolds sharing repeated contigs. In *Combinatorial Optimization and Applications - 12th International Conference, COCOA 2018, Atlanta, GA, USA, December 15-17, 2018, Proceedings*, pages 94–107, 2018.
[32] H. Tang. Genome assembly, rearrangement, and repeats. *Chemical Reviews*, 107 (8):3391–3406, 2007.
[33] T. J. Treangen and S. L. Salzberg. Repetitive DNA and next-generation sequencing: computational challenges and solutions. *Nat Rev Genet*, 13(1):36–46, January 2012.

[34] F. Vezzi, G. Narzisi, and B. Mishra. Reevaluating assembly evaluations with feature response curves: GAGE and assemblathons. *PLoS ONE*, 7(12):52210, 2012.

[35] M. Weller, A. Chateau, and R. Giroudeau. Exact approaches for scaffolding. *BMC Bioinformatics*, 16(Suppl 14):S2, 2015.

[36] M. Weller, A. Chateau, and R. Giroudeau. On the linearization of scaffolds sharing repeated contigs. In *Proc. 11th COCOA'17*, pages 509–517, 2017.

[37] D. R. Zerbino and E. Birney. Velvet: algorithms for de novo short read assembly using de Bruijn graphs. *Genome Res*, 18(5):821–829, 2008.