DIMACS Technical Report 2001-41
November 2001

1.375-Approximation Algorithm for Sorting by Reversals

by

Piotr Berman [1]
Dept. of Computer Science
and Engineering
The Pennsylvania State University

Sridhar Hannenhalli [2]
Celera Genomics Co.
45 W Gude Dr.
Rockville, MD 20350

Marek Karpinski [3]
Dept. of Computer Science
University of Bonn

[1]Research done in part while visiting Dept. of Computer Science, University of Bonn. Work partially supported by NSF grant CCR-9700053, NIH grant 9R01HG02238-12 and DFG grant Bo 56/157-1. E-mail `berman@cse.psu.edu`

[2]E-mail `Sridhar.Hannenhalli@celera.com`

[3]Research done in part while vising Dept. of Computer Science, Princeton University. Work partially supported by DFG grants, DIMACS, and IST grant 14036 (RAND-APX). E-mail `marek@cs.uni-bonn.de`

ABSTRACT

Analysis of genomes evolving by inversions leads to a general combinatorial problem of *Sorting by Reversals*, MIN-SBR, the problem of sorting a permutation by a minimum number of reversals. This combinatorial problem has a long history, and a number of other motivations. It was studied in a great detail recently in computational molecular biology. Following a series of preliminary results, Hannenhalli and Pevzner developed the first exact polynomial time algorithm for the problem of sorting signed permutations by reversals, and a polynomial time algorithm for a special case of unsigned permutations. The best known approximation algorithm for MIN-SBR, due to Christie, gives a performance ratio of 1.5. In this paper, by exploiting the polynomial time algorithm for sorting signed permutations and by developing a new approximation algorithm for maximum cycle decomposition of breakpoint graphs, we improve the performance ratio for MIN-SBR to 1.375. Besides its biological and combinatorial importance, better approximation algorithms for MIN-SBR have become particularly challenging recently because this problem has been proven to NP-hard by Caprara, and MAX-SNP hard by Berman and Karpinski, excluding thus an existence of a polynomial time approximation scheme (PTAS) for that problem.

1 Introduction

A *reversal* $\rho = \rho(i,j)$ applied to a permutation $\pi = \pi_1 \dots \pi_{i-1}\pi_i \dots \pi_j\pi_{j+1} \dots \pi_n$ reverses the order of elements $\pi_i \dots \pi_j$ and transforms π into permutation $\pi \cdot \rho = \pi_1 . . . \pi_{i-1}\pi_j . . . \pi_i\pi_{j+1} . . . \pi_n$. *Reversal distance* $d(\pi, \sigma)$ is defined as the minimum number of reversals $\rho_1, \dots, \rho_t$ needed to transform π into the permutation σ, i.e., $\pi \cdot \rho_1 \cdots \rho_t = \sigma$. Let $id = 12..n$ be the identity permutation, then $d(\pi, \sigma) = d(\pi \cdot \sigma^{-1}, id)$. The problem of computing the reversal distance for given two permutation is equivalent to the problem of *Sorting by reversal*, MIN-SRB, where for a given π we compute $d(\pi, id)$. This problem received a lot of attention because it models *global genome rearrangements*. The importance of computational methods to analyze genome rearrangements was first recognized by Sankoff et al. [SCA90]. See Sankoff et al. [SLA92], Hannenhalli et al. [HCK95] and Bafna and Pevzner [BP95a] for applications of MIN-SBR to analyze genome rearrangements. Similar combinatorial problems were investigated by Gates and Papadimitriou [GP79], Amato *et al.* [ABIR89] and Cohen and Blum [CB95].

Biologists derive gene orders either by sequencing entire genomes or by comparative physical mapping. Sequencing provides information about directions of genes and allows one to represent a genome by a *signed* permutation (Kececioglu and Sankoff [KS93]). Most of currently available experimental data on gene orders are based on comparative physical maps. Physical maps usually do not provide information about directions of genes and, therefore lead to representation of a genome as an *unsigned* permutation π.

Kececioglu and Sankoff [KS93] gave the first algorithm with a proven performance guarantee for MIN-SBR by giving a 2-approximation algorithm and conjectured that the problem is NP-hard. They were first to exploit the link between the reversal distance and the number of *breakpoints* in a permutation. Since Sorting by Reversals as well as other genome rearrangements problems were believed to be NP-hard, most of the efforts in analyzing gene orders were directed towards approximation algorithms. Bafna and Pevzner [BP93] improved the performance ratio to 1.75 for unsigned permutations and 1.5 for signed permutations. Hannenhalli and Pevzner [HP95] found however an exact polynomial algorithm for sorting *signed* permutations by reversals, a problem which also was believed to be NP-hard (see [BH96] and [KST97] for faster algorithms). However, MIN-SBR, the problem of sorting an unsigned permutation, was shown to be NP-hard by Caprara [C97] thus proving the conjecture. Later, this problem was also shown to be MAX-SNP hard by Berman and Karpinski [BK99], while Christie [Ch98] improved the performance ratio for MIN-SBR to 1.5.

In this paper, by exploiting a polynomial time algorithm for sorting a signed permutation by reversals, and by developing a new approximation algorithm for maximum cycle decomposition of breakpoint graphs, we design a 1.375-approximation algorithm for sorting by reversals. This improvement over 1.5 ratio of Christie is obtained here by a different method and a substantially more complicated algorithm.

Kececioglu and Sankoff [KS94], Kececioglu and Gusfield [KG95], Hannenhalli and Pevzner [HP95] and [HP96], Kececioglu and Ravi [KR95], and Bafna and Pevzner [BP95b] provide other computational studies of genome rearrangements and Pevzner and Waterman [PW95] gave a survey of combinatorial problems motivated by genome rearrangements.

Bafna and Pevzner [BP93] revealed important links between the breakpoint graph of a permutation and the reversal distance. In particular, they showed a strong correspondence between the maximum cycle decomposition of the breakpoint graph of the permutation and its reversal distance. Moreover, for all known biological instances, it was observed that the maximum cycle decomposition is sufficient to estimate the reversal distance exactly. Although, in general, the maximum cycle decomposition does not suffice to compute the reversal distance precisely, it does suffice to compute the reversal distance approximately with a guaranteed performance.

This paper is organized as follows. In Section 2 we reduce the approximating of MIN-SBR to GEDSAC, a problem of finding a sufficiently large disjoint set of alternating cycles in the breakpoint graph of a given permutation. In Section 3 we reduce GEDSAC to GEIS, a problem of finding a sufficiently large independent set in a special variety of graphs. Finally, Section 4 describes an algorithm for GEIS.

2 From MIN-SBR to Alternating Cycles

Solving MIN-SBR problem directly does not seem feasible, because as yet it is not known how to evaluate individual reversals, and sequences of reversals form exponentially large searching space. Fortunately, Hannenhalli and Pevzner found a reduction of this problem to the one of finding an optimal decomposition of a certain graph with two edge colors. Without going into details yet, we will have two goals: finding a maximally large family of edge-disjoint cycles, while in the same time minimizing the number of so-called hurdles that this family of cycles defines.

At first the new task does not appear to be any easier to solve. However, as we shall see, because we only want to approximate the optimal solution, we will be able to simplify the task dramatically. To begin with, we will be searching for cycles that consists of at most 6 segments. Moreover, we will be able to eliminate the explicit counting of the hurdles altogether.

We start from precise definitions and then proceed with an amortized analysis that will reveal the relative importance of various cycles and hurdles from the point of view of approximation. Importantly, we will show that we can neglect the existence of certain classes of objects, and eliminate another class of objects by applying certain kinds of greedy choices. We will conclude this Section with an algorithm for approximating MIN-SBR that uses as a subroutine the algorithm for certain simpler problem which is provided in the remaining sections.

2.1 Definitions and graph-theoretic background

Bafna and Pevzner [BP93], Hannenhalli and Pevzner [HP95, HP96] have described how to reduce the MIN-SBR to a purely graph-theoretic problem, Maximum Decomposition into Alternating Cycles, or MDAC in short. In this section, we will paraphrase the results in

[HP96], where they describe an exact algorithm for MIN-SBR problem that is polynomial in certain cases that are important in estimating evolutionary distances.

We use $[i, j]$ to denote the set of integers $\{i, i+1, \ldots, j\}$. A permutation π is a 1-1 mapping of $[1, n]$ onto itself, and π_i is the value or π for an argument i. We extend π to one extra argument by setting $\pi_0 = 0$. To avoid modulo notation, we will assume $\pi_i = \pi_{i+n+1}$ for every integer i.

A *breakpoint graph of* π, G_π, has a node set $[0, n]$ and two sets of edges:

$$\textit{breaks} = \{\{\pi_i, \pi_{i+1}\} : \ i \in [0, n]\};$$
$$\textit{chords} = \{\{i, i+1\} : \ i \in [0, n]\}.$$

If a chord happens to be a break, we count it as a separate object, and say that this is a *short chord*. For that reason, our edge sets can be actually multisets. An *alternating cycle*, AC for short, is a connected set of edges C with the following property: if a node belongs to i breaks of C, than it also belongs to i chords of C. A *decomposition into alternating cycles*, DAC for short, is a partition of the edges of G_π into ACs.

A DAC $\mathcal{C}$ of G_π can be represented by the following *consecutive* relation: edges e and e' are consecutive edges on a cycle; here cycle is identified with its traversal. In turn, this relation uniquely determines a *spin* of π, a signed permutation π' such that $\pi'_i = \pm\pi_i$ (see [BP93, HP96]); a sequence of reversals that sorts π' obviously sorts π as well, and any sequence of reversals that sorts π sorts one of its spins. Because we can find an optimum reversal sequence for a spin of π in polynomial time [HP95], the seatch for an optimal reversal sequence for π is equivalent to the search for an optimal spin of π, and, in turn, the search for an optimal DAC of G_π.

A given cycle decomposition $\mathcal{C}$ defines a set of *hurdles* (defined later). The following theorem of Hannenhalli and Pevzner [HP96] is crucial:

Theorem 1 *Given a cycle decomposition of the breakpoint graph of* π*, there exists a polynomial time algorithm that finds a sequence of* $n - c + h + f$ *reversals that sorts permutation* π*, where* c *is the number of cycles in the decomposition,* h *is the number of hurdles and* $f \in [0, 1]$*. Moreover, the minimum length sequence can be computed in that fashion.*

The above theorem is actually a joint corollary of Lemma 3.1 and Theorem 2.1 of [HP96]. Because we are interested in an approximation algorithm, we will ignore the small term f. Our goal in this section is to show how to handle the minimization of h so we will later maximize c in a separate problem. To define *hurdles*, we need some more definitions.

We will use the following geometric representation of G_π: the nodes $\pi_0, \ldots, \pi_n$ are placed counter-clockwise on a circle $\mathbf{C}$, each break $\{\pi_i, \pi_{i+1}\}$ is a $\mathbf{C}$-arc segment, and a chord $\{i, i+1\}$ is the line segment that joins points i and $i+1$. Note that in this representations numbers are viewed as node names, moreover, if $\pi_{i+1} = \pi_i \pm 1$, then the break $\{\pi_i, \pi_{i+1}\}$ and the short chord $\{\pi_i, \pi_{i+1}\}$ are indeed two different objects. To avoid confusion, we will apply the word *chord* exclusively to the representations of the chords of G_π, while a *chordal segment* is any line segment that connects two (representations of) nodes of G_π.

If two chords e_0 and e_1 intersect in the interior of $\mathbf{C}$, they form an *interleaving pair.*

A *chord component* is a connected component of the graph $<$ *chords, interleaving pairs* $>$. We will assume that there is more than one chord component; otherwise we will have a trivial case for the algorithm of this section.

We define the *area of a chord component* C, denoted by $A(C)$, as follows: for each chord $e \in C$ (viewed as a line segment) we remove the endpoints, then we take the union of these chords, and finally we take the smallest convex set that contains that union.

Observation 1 *$A(C)$ is a convex polygon and its set of nodes is the the set of endpoints of the chords of C. Moreover, the chords of C subdivide $A(C)$ into convex polygons that either have the entire boundary covered by the chords of C, or the entire boundary with the exception of a single segment.*

This observation leads to the next one:

Observation 2 *$A(C)$ cannot intersect a chord e if $e \notin C$.*

To see that, consider a chord e that shares a point, say x, with $A(C)$; if x does not lie on one of the chords of C, it must belong to the interior of one of the convex subdivisions of $A(C)$ that is surrounded on all sides, except one, by segments of chords from C. Since e extends from x in two directions up to the circle $\mathbf{C}$; in at least one of these two directions e intersects one of these surrounding segments, and hence $e \in C$.

A *crescent* $Cr(i,j)$ is an area bounded by counter-clockwise arc from i to j and the chordal segment $\{i,j\}$. If the counterclockwise listing of the nodes of $A(C)$ is $i_0, i_1, \ldots, i_k = i_0$, then $interior(\mathbf{C}) - A(C)$ is a disjoint union $Cr(i_0, i_1), \ldots, Cr(i_{k-1}, i_k)$; we call them the *neighbor crescents* of C.

The relative positions of different chord components are described in the next observation.

Observation 3 *$A(C')$ is a subset of one of the neighbor-crescents of C for each chord component $C' \neq C$.*

To see that, note that a chord $e \notin C$ must be contained in one of the neighbor-crescents of C, as it is disjoint with $A(C)$. By definition, chords contained in different neighbor-crescents cannot interleave (intersect in the interior of C), so the entire chord component of e most be contained in a single neighbor-crescent.

Lemma 1 *If $Cr(i,j)$ is a neighbor-crescent of a chord component C, and there exists a chord contained in that crescent, then the chords contained in $Cr(i,j)$, together with the breaks on the $\mathbf{C}$-arc extending from i to j, form an AC.*

Moreover, $\{i,j\}$ is a chord only if it is a short chord, in which case the AC of $Cr(i,j)$ consists of exactly one chord and exactly one break.

Proof. We need to show that every node on the **C**-arc from i to j belongs to the same number of breaks of this arc and the chords contained in $Cr(i,j)$. This is obvious for nodes different than i and j: they belong to breaks of this arc, and to two chords that are at least partially contained in $Cr(i,j)$, and we have observed that if a chord is partially contained in a neighbor-crescent, then it is completerly contained.

The endpoints of the arc, i and j, each belong to exactly one break of the arc, so we need to show that they are contained in exactly one chord contained in $Cr(i,j)$. This follows from the fact that all the chords form a single simple cycle, exactly like the breaks. Because there exists chords both inside $Cr(i,j)$ and outside, the cords contained inside form a collection of (node disjoint) simple paths. Our previous arguments show that i and j are the only possible endpoints of these paths, and therefore there exists exactly one such path, from i to j.

In particular, if $\{i,j\}$ is a chord, it forms a one-edge path of chords from i to j. Since there exists only one such path, there are no other chords contained in the crescent $Cr(i,j)$, and this $\{i,j\}$ is a short chord that forms a cycle together with the break (arc) $\{i,j\}$.

❑

The last lemma characterizes neighbor-crescents of C that contain some chords. We say that other neighbor-crescents are *empty*. Obviously, if $Cr(i,j)$ is an empty neigbor crescent, the **C**-arc from i to j forms a single break, we say that that break is *associated* with C. We define *edge component* of C as the set consisting of the chords of C and the breaks associated with C.

Lemma 2 *Edge components of chord components form a DAC decomposition.*

Proof. By the definition, edge components are pairwise disjoint and together they contain all the edges. Thus it suffices to show that each edge component is an AC.

To see that, observe that we can form the edge component of C by starting with the set of all edges — chords and breaks — and then removing, one by one, the ACs of non-empty neighbor cycles. From the definition, if we remove AC from an AC, what remains is an AC as well.

❑

Given a fixed DAC $\mathcal{D}$ of the edge set of G_π into cycles, we define *a cycle component* as a connected component of the graph where the nodes are the edges (chords and breaks) and a pair of edges is connected if either 1) they are in the same edge component, or 2) they belong to the same cycle from $\mathcal{D}$.

Observation 4 *The union of chords of a cycle components, including their endpoints, forms a connected set.*

Proof. By the definitions of edge components and chord components the union of chords that belong to one edge component is connected even without including their endpoints.

Now consider two consecutive edges on a cycle, say a break $\{\pi_{i-1}, \pi_i\} \in C_1$ and a chord $\{\pi_i, \pi_i + 1\} \in C_2$, where C_1 and C_2 are edge components. Then chord $\{\pi_i, \pi_i - 1\}$ belongs to C_1, so after including the point π_i the union of the chords of C_1 and C_2 is connected.

❑

This observation allows us to apply Observations 1, 2, 3, and Lemma 1 to chord set of a cycle component in the same way as to a chord component. In particular, for a chord component C we can define $A(C) = A(C \cap \text{chords})$ and the neighbor-crescents. If C, C_1, C_2 are cycle components and $A(C_1)$ and $A(C_2)$ are contained in two different neighbor-crescents of C, then we say that C *separates* C_1 and C_2.

A cycle is *oriented* if it contains a chord $\{i, i+1\}$ and in each of the crescents $Cr(i, i+1)$, $Cr(i+1, i)$ it contains a break incident to $\{i, i+1\}$. A cycle component is oriented if it contains an oriented cycle, or if it consists of two edges only. Note that a cycle of two edges is always a separate singleton cycle component—this is the case when its chord is short.

A *hurdle* is an unoriented cycle component that does not separate two other unoriented cycle components.

2.2 Breaking cycle components into edge components

Our cost is the number of breaks minus the number of ACs plus the number of hurdles. To minimize the cost, we need to maximize the number of ACs in our DAC, while simultaneously minimizing the number of hurdles. We would like to separate those two tasks as much as we can.

In our quest for a small number of reversals, we will first maximize the number of cycles found in each edge component. We will show soon that by restricting ourselves to ACs contained in a single edge component, we are not decreasing the number of ACs in a decomposition, and sometimes we can even increase that number. However, this restriction may increase the number of hurdles. The amortized analysis introduced in the next subsection shows how to take care of these extra hurdles. In at account the increase in the number of cycles that may result from our restriction.

Let us consider now an optimum DAC $\mathcal{C}$ and a neighbor-crescent $Cr(i, j)$ of a chord component X. By Lemma 1 we can partition all edges into two ACs A and B, where A is the set of edges contained in $Cr(i, j)$. We will modify $\mathcal{C}$ so that every cycle $C \in \mathcal{C}$ will satisfy $C \subset A$ or $C \subset B$.

By definition, i and j are the only nodes that belong simultaneously to edges of A and B. Consider $C \in \mathcal{C}$ such that $C \cap A \neq \emptyset$ and $C \cap B \neq \emptyset$. Because C is connected, without loss of generality we may assume that both $C \cap A$ and $C \cap B$ contain at least one edge that contains i.

We will distinguish now between several cases.

Odd case. Suppose that i belongs to exactly one edge of $C \cap A$. Because every node in $Cr(i, j)$ other than i and j must belong to an even number of edges of C, this means that j also belongs to exactly one edge of $C \cap A$. Note that i belongs to two other edges that in turn belong to another cycle of $D \in \mathcal{C}$, and D has exactly the same properties: i.e. for $\mathbf{C} \in \{C, D\}$, $\mathbf{A} \in \{A, B\}$ and $\mathbf{i} \in \{i, j\}$ there exists exactly one edge of $\mathbf{A} \cap \mathbf{C}$ that is incident to $\mathbf{i}$. We consider two subcases.

Odd Group case. For some $k > 2$ there exists a cycle of nodes $i = i_0, i_1 = j, \ldots, i_k = i_0$ and a sequence of chord components $X_0, \ldots, X_{k_1}$ such that $Cr(i_l, i_{l+1})$ is a neighbor-crescent of X_l. Let Y_l be an AC formed from the edge component of X_l and other edge components that are not contained in $Cr(i_l, i_{l+1})$. By applying above argument inductively, one can see that for $\mathbf{C} \in \{C, D\}$, $\mathbf{A} \in \{Y_0, \ldots, Y_k\}$ and $\mathbf{i} \in \{i_0, \ldots, i_k\}$ there exists exactly one edge of $\mathbf{A} \cap \mathbf{C}$ that is incident to $\mathbf{i}$.

Thus we can replace cycles C and D with k cycles of the form $(C \cup D) \cap Y_l$. We associate $k - 2$, the resulting increase in the number of cycles with the convex polygon bounded by the polyline $(i_0, i_1, \ldots i_k = i_0)$.

Odd Pair case. Same as the Odd Group case, but for $k = 2$, so we have no assured increase in the number of cycles.

Even case. When the premise of the Odd Case does not hold, then both $C \cap A$ and $C \cap B$ is an AC (or a union of ACs, because such an edge set may be disconnected). Therefore we can replace C with these intersections and the modified DAC surely has more cycles. We can associate 1 with node i as the lower bound on the increase in the number of cycles.

2.3 Amortized analysis

Throughout the paper, we will use potential analysis to assure that we obtain the promised approximation ratio. For every unit of the cost of the optimum solution we can place $11/8$ of the potential units, and for every unit of the cost of our solution, we place -1 of the potential units. We deliver a desired solution if the sum of the placed potential units is non-negative. At many stages of the analysis, we add and subtract the potential units in various parts of our structure; such a move is valid if we assure that the sum of additions does not exceed the sum of subtractions.

Each break contributes 1 both to the cost of the optimum solution and the cost of the solution obtained by our algorithm. Thus we can place $11/8 - 1 = 3/8$ on each break.

Each AC of the optimum solution contributes -1 to the optimal cost, so we can place $-11/8$ on this cycle. However, when the ACs of the optimum solution span more than one edge component, we break them as described in the previous subsection.

After the break-up, the maximal number of cycles misrepresents the true number in the optimum, to account for that we place the corrective amounts of units. Initially, we place them as follows: in Odd Group case with k chord components we place $11/8(k-2)$ on the polygon that separates these components, in Odd Pair case that increases the number of cycles we place $11/8$ on the chord separating the two components, and in Even Case we place $11/8$ on a separating node. Once the break-up is complete, edge components incident to objects with corrective units share those units evenly. The least possible share occurs for Odd Group case with $k = 3$ and it equals to $11/24$.

Because of the break-up, we lost the account of the hurdles in the optimum solution, so we do not take them into account, this can only decrease the total balance of the potential. At each edge component that we estimate to be a hurdle we place -1 unit. The estimation method does have to be correct, the only requirement is that we will have at least as many estimated hurdles as we have the actual ones. In particular, we will estimate every edge component with at least 5 breaks to be a hurdle.

2.3.1 Small edge components

We first analyze the case when $\mathcal{C}$ contains only a few edges, in which case our algorithm can find a maximum DAC of this edge component. As a result, on every cycle from the modified optimum solution we may place 1. We will establish the situations when such a maximum DAC of $\mathcal{C}$ is not good enough to assure our approximation ratio, *i.e.* when the resulting balance of the potential in $\mathcal{C}$ is negative when we estimate that $\mathcal{C}$ is a hurdle.

An AC with i breaks will be called an i-cycle. On each cycle of $\mathcal{C}$ we have put $-1\frac{1}{8}$ and 1 for the balance of $-\ {}^3/_8$; because each break has a balance of ${}^3/_8$, the balance of an i-cycle C is ${}^3/_8(i-1) = {}^3/_8\, w(C)$.

The overall balance of $\mathcal{C}$ equals the sum of its cycle balances ${}^3/_8\, w(\mathcal{C})$. From this sum we subtract 1 when we estimate $\mathcal{C}$ to be a hurdle.

Observe that a 1-cycle forms an edge component by itself (let us call it a em 1-component), and this component cannot be a hurdle, because this cycle is oriented. Thus the balance of such a component is 0. Later we may assume that $\mathcal{C}$ contains no 1-cycles.

Consider that $\mathcal{C}$ consists of k cycles $C_1, \cdots, C_k$. When we estimate $\mathcal{C}$ to be a hurdle, its balance is ${}^3/_8\, w(\mathcal{C}) - 1$, so it is negative only if $w(\mathcal{C}) \le 2$. Clearly, it suffices to consider the following cases:

(a) $k = 1, w(C_1) = 1$, balance equals $-\ {}^5/_8$;

(b) $k = 1,\ w(C_1) = 2$, balance equals $-\ {}^1/_4$;

(c) $k = 2, w(C_1) = w(c_2) = 1$, balance equals $-\ {}^1/_4$.

Case (a): $\mathcal{C}$ consists of a single 2-cycle C. Then the two chords of C must interleave (they form an edge component), it is easy to see that in this case each of two chords of C has one break of C contained in each of its two sides. We conclude that C (and $\mathcal{C}$) is oriented, so it cannot be a hurdle, and we do not have to subtract 1.

Case (c): $\mathcal{C}$ consists of two 2-cycles. If one of the 2-cycles is oriented, the entire $\mathcal{C}$ is oriented and we do not estimate it to be a hurdle.

Fig. 2.3.1[1] shows that is these two cycles have two nodes in common, there exists an alternative decomposition which changes $\mathcal{C}$ into two oriented cycle components therefore we can assume that the two 2-cycles of $\mathcal{C}$ share at most one node.

One can show that in this case we can give $\mathcal{C}$ a part of positive potential created when we broke cycle components of the optimum solution into edge components. Suppose that $\mathcal{C}$ is also cycle component in the optimum solution — then this is an unoriented component of

[1] In our figures, we place the nodes on a straight line, so that the breaks become short straight segments and the chords become arcs.

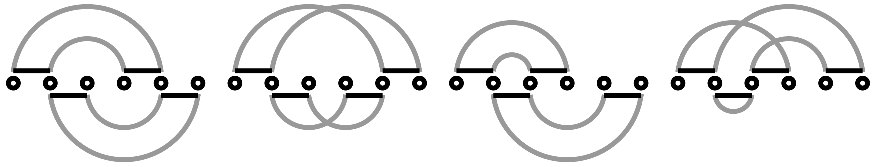


Figure 1: Examples of an unoriented cycle component of weight 2 and of oriented cycle components formed from the same edge sets.

the optimum solution, so we do not attribute creating a hurdle in our DAC to $\mathcal{C}$. Otherwise, consider the last component breaking step involving $\mathcal{C}$, if this was not an Odd Pair case, $\mathcal{C}$ has received at least $^{11}\!/_{24}$ corrective units and its total potential balance is $^{11}\!/_{24} - {}^{1}\!/_{4} > 0$. Now consider Odd Pair Case. $\mathcal{C}$ was intersected by two ACs from the optimum solution, say C and D, and we replaced them by another two cycles, one being $E = (C \cup D) \cap \mathcal{C}$. A brief inspection shows that in this case we have $E = \mathcal{C}$, and thus we can further decompose D into two cycles, which provides a corrective term of $^{11}\!/_{8}$ and thus the total balance of $\mathcal{C}$ is positive.

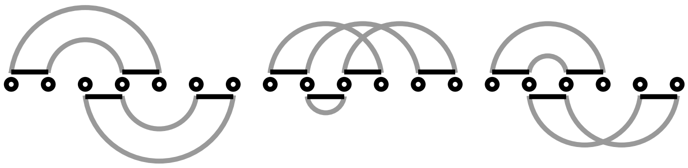


Figure 2: Unoriented cycle components of weight 2. In case (b), if one of the outer sides of the component contains only one 1-component, then we can turn these two components into one oriented component of weight 2

We can formulate our rule for estimating hurdles: (i) if $w(\mathcal{C}) > 2$, we estimate $\mathcal{C}$ to be a hurdle (and we count them as unoriented). Apart from that, we estimate hurdles according to the definition.

If estimated hurdle falls in case (b) or (c) and we can show that either it is also an cycle component of the optimum solution, or it receives corrective units of potential when we account for the increased number of cycles, then the balance of $\mathcal{C}$ is positive. We have just shown that case (c) is always in this situation.

Case (c): if the balance of $\mathcal{C}$ is negative, it is a hurdles and it does not receive any corrective units.

Component $\mathcal{C}$ has three neigbor crescents and at most one neighbor contains an estimated unoriented component.

We total the balance of $\mathcal{C}$ and its neigbor crescents that contain oriented components only. If even one of these components, say $\mathcal{D}$, has $w(\mathcal{D}) > 0$, then the sum of balances is at least $3/8 - 1/4 > 0$. IN this case we say that $\mathcal{C}$ is *rich*. Thus we can assume that the "oriented" neighbor-crescents contain only 1-components. If an oriented neighbor-crescent contains exactly one such component, we change the solution as shown in Fig. 2.3.1bc. If this change turns another component, say $\mathcal{D}$, into a hurdle, than clearly $\mathcal{D}$ is rich.

We are left with the case when each oriented neighbor-crescent contains multiple 1-components. In this case there exists an optimum solution where no cycle overlaps both $\mathcal{C}$ and one of this 1-components (this fact follows from the discussion of the Odd Group case and Even Case). If all three neighbor-crescents are oriented, we actually get an optimal solution. Thus were can additionally assume that $\mathcal{C}$ has one unoriented neighbor-crescent $Cr(i,j)$ which forms Odd Pair case with $\mathcal{C}$, hence the chordal segment $\{i,j\}$ separates the area of $\mathcal{C}$ from the area of another edge component, say $\mathcal{D}$. We say that $\mathcal{C}$ is a child of $\mathcal{D}$.

If $w(\mathcal{D}) = 1$, then $\mathcal{D}$ has but one child, and we add the balances of $\mathcal{C}$ and $\mathcal{D}$. If $w(\mathcal{D}) = 2$, then $\mathcal{D}$ has at most 4 sides, thus at most 3 children, the balance of $\mathcal{D}$ is $3/4$ and each child has balance $-1/4$, so again, we add the balance of $\mathcal{D}$ to the balances of its children. Thus it remains to consider the case when $w(\mathcal{D}) > 2$ and thus $\mathcal{D}$ is unconditionally estimated to be a hurdle.

Within that parent component $\mathcal{D}$ we must have a plausible single cycle E that was created in $\mathcal{D}$ during the breaking step of $Cr(i,j)$.

We introduce the following terminology for such a situation: cycle E is a *absorber* and the cycle/component $\mathcal{C}$ is a *little hurdle*. We say that absorber E absorbs little hurdle $\mathcal{C}$.

2.3.2 Catalogue of cycles with negative potential

We transfer the negative potential of the little hurdles to the respective absorbers. Now the entire potential is contained in the cycles contained in the chord components that are estimated to be hurdles. Our task is to find enough ACs and to absorb enough little hurdles to create the nonnegative balance of the potential. Obviously, we can ignore the cycles of the optimum solution that have non-negative potential. Therefore we need to establish which ACs have negative potential.

Consider an i-cycle from the optimum solution. If it is not an absorber, then its potential is $3/8(i-1)-1$, so for $i = 2,3,4,5$ this potential is $-5/8$, $-2/8$, $1/8$ and $4/8$ respectively. Thus only 2-cycles and 3-cycles have negative potential.

Now consider a (i,j)-*absorber* which we define to be an $(i+j)$-cycle that absorbs j little hurdles. Its balance is $3/8(i+j-1) - 1/4 j - 1 = 3/8(i-1) + 1/8 j - 1$. Note that this covers the case of ordinary i-cycles that will be considered as $(0,i)$-absorbers.

Since (i,j)-absorber of the optimum solution has potential balance $3/8(i-1) + 1/8 j - 1$, the absorbers with negative potential are (1,j)-absorbers for $j \leq 7$, (2,j)-absorbers for $j \leq 4$ and (3,1)-absorbers. As we will show, only (2,1)- and (3,1)-absorbers actually exist.

Consider absorber C of j little hurdles $H_1, \ldots, H_j$ there exists a DAC of $C \cup H_1 \cup \ldots H_j$ into $j+1$ ACs, each of them intersecting both C and one or more of the little hurdles. We say that this is a decomposition into *good* ACs.

Consider a good AC D that intersects a little hurdle H. One can see that $D \cap H$ must be a path of three edges, which we will call a *long segment*. The edges of C shall be called *short segments*.

The variety of possible absorbers is restricted by the following two lemmas.

Lemma 3 *A good cycle must contains both kinds of short segments, i.e. at least one break and at least one chord.*

Proof. Consider a little hurdle that is contain in the crescent $Cr(k,l)$ of the absorber. One cans see that at every node inside $Cr(k,l)$ there are two incident chords that are contained in $Cr(k,l)$, and that at k and l there is exactly one; thus chords form a simple path from k to l. The same holds for the breaks. Thus a long segments that go from k to l can be replaced by a path within the crescent that contains only chords (or only breaks).

Suppose now that there exists a good cycle where each short segment is a chord. Then the long segments can be replaced by the path that consists of chords only, which yields a cycle of chords. Because not all nodes lie on this cycle, this is a contradiction. The same holds if all short segments are breaks.

❑

Observation 5 *A good AC with k long segments contains at least $k+2$ short segments.*

Indeed, the number of segments has to be even, and the long segments cannot be consecutive, as they are never incident to each other. Thus were there only k short segments, every stretch of short segments would have but one edge, thus all these edges would be of the same kind.

¿From that we immediately deduce

Observation 6 *An absorber of j hurdles contains at least $2j+1$ breaks.*

Indeed, such an absorber C together with its little hurdles can be decomposed into $j+1$ good cycles that together have $2j$ long segments, thus according to the previous observation these good cycles must contain at least $4j+2$ short segments, and since half of these short segments are breaks, $2j+1$ breaks.

Now we can show

Observation 7 *$(1,j)$-absorbers exist only for $j=0$ and $(2,j)$-absorbers exist only for $j \leq 1$.*

Indeed, for $j \geq 1$ a $(1,j)$-absorber would have $j+1 < 2j+1$ breaks, and for $j \geq 2$ a $(2,j)$-absorber would have $j+2 < 2j+1$ breaks, a contradiction.

It is worthwile to note that besides the fact that the variety of possible absorbers is quite restricted, the shapes of (2,1)-absorbers and (3,1)-absorbers are quite restricted as well, as we can see in Fig. 2.3.2.

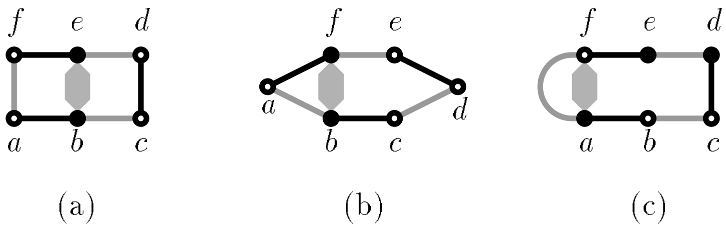


Figure 3: Conceivable (2,1) absorbers, the hexagons indicate the position of the little hurdle, edge colors indicate chords and breaks. Only (a) can exist, (b) and (c) cannot be decomposed into 2 good cycles.

2.3.3 When DAC of a large edge component is good enough

DAC found by our algorithm is good enough if it assures that the balance of the component in question is nonnegative. In a large component, we will require that the sum of balances of ACs of the optimum solutions and ACs found by our algorithm is at least 1, so we can create a hurdle and still have a nonnegative potential.

We define function ϕ such that if C is an AC from the optimum solution, then $-\phi(C)$ is the potential that C contributes to the overall balance of its edge component. We can view a regular i-cycle as a $(i,0)$-absorber, in the previous subsection we calculated that

$$\phi(C) = 1 - {}^{3}\!/_{8}\, i + {}^{1}\!/_{8}\, j \text{ for an } (i,j)\text{-absorber } C.$$

In turn, when our algorithm finds an (i,j)-absorber C for its solution, this decreases the resulting cost by $1+j$: 1 for the AC in the decomposition and j for the decrease in the estimate on the number of hurdles in its solution. Thus we can define

$$\gamma(C) = 1 + j.$$

Consequently, if $\mathcal{I}$ is a DAC of the edge component under discussion that is found by our algorithm, and $\mathcal{I}^*$ is an optimal DAC of that component, the condition by sufficiently good $\mathcal{I}$ is

$$\gamma(\mathcal{I}) \geq \phi(\mathcal{I}^*) + 1$$

.

Let $n_{i,j}$ be the number of (i,j) absorbers in the optimum DAC. We can rewrite the above condition as follows:

$$\gamma(\mathcal{I}) \geq 1 + {}^{5}\!/_{8}\, n_{2,0} + {}^{2}\!/_{8}\, n_{3,0} + {}^{4}\!/_{8}\, n_{2,1} + {}^{1}\!/_{8}\, n_{3,1}.$$

Finding DAC $\mathcal{I}$ for a given edge component with the above property is Good Decomposition of a Component problem, or GDoaC for short.

2.3.4 When a non-overlapping collection of ACs is good enough

We will solve GDoaC problem usining somewhat simpler GEDSAC problem:

given: an edge component $\mathcal{C}$;

find: a *good* edge disjoint set of ACs, where a set $\mathcal{I}$ is good if it satisfies

$$g(\mathcal{I}) \geq {}^5/_8\, n_{2,0} + {}^2/_8\, n_{3,0} + {}^4/_8\, n_{2,1} + {}^1/_8\, n_{3,1}.$$

A solution $\mathcal{I}$ of GEDSAC may provide a solution of GDoaC: if $\mathcal{I}$ does not cover all the edges of the considered edge component $\mathcal{C}$, then the remaining edges form an extra AC, and thus we are getting a DAC that solves GDoaC. So it remains to consider when $\mathcal{I}$ covers all the edges of $\mathcal{C}$, so it is a DAC, and yet fails to be a good decomposition. Assuming that the component contains m breaks, we have $\gamma(\mathcal{I}) \geq {}^1/_3\, m$ (this inequality is tight if all ACs in $\mathcal{I}$ are 3-cycles). On the other hand, $\phi(I^*) \leq {}^5/_{16}\, m$ (this inequality is tight if all ACs in $\mathcal{I}^*$ are 2-cycles). Thus

$$\gamma(\mathcal{I}) \geq ({}^1/_3 - {}^5/_{16})m + \phi(\mathcal{I}^*) = {}^1/_{48}\, m + \phi(\mathcal{I}^*).$$

We conclude that if an edge component has at least 48 breaks, then a solution of GEDSAC immediately yields of solution of GDoaC, and as we have already established, for $2 < m < 48$ breaks it suffices to solve GDoaC problem exactly.

2.4 Algorithm

We can summarize this section in the form of an algorithm for MIN-SBR problem.

1. Input permutation π.
2. Form graph G_π.
3. Decompose G_π into edge components, establish which constitute small hurdles.
4. For each large edge component
   (i) Establish (2,1)- and (3,1)-absorbers.
   (ii) If the component has fewer than 48 breaks, solve exactly GDoaC problem, else
   (iii) Solve GEDSAC problem and add the cycle formed from edges not covered by the solution.
5. Combine DACs of edge component into a single DAC of G_π, re-partition the absorbers and their little hurdles to decrease the number of hurdles.
6. Use this DAC to define a spin π' of π and apply the algorithm of Hannenhalli and Pevzner [HP95] to π'.

3 Simplifying the input of GEDSAC

Let $\mathcal{V}_{i,j}$ be the set of (i,j)-absorbers, $\mathcal{V}_2 = \mathcal{V}_{2,0} \cup \mathcal{V}_{2,1}$, $\mathcal{V}_3 = \mathcal{V}_{3,0} \cup \mathcal{V}_{3,1}$, $\mathcal{V} = \mathcal{V}_2 \cup \mathcal{V}_3$ and let $\mathcal{E}$ be the set of edge overlapping pairs of cycles from $\mathcal{V}$. Then a solution of GEDSAC is an independent set in $\mathcal{G} =< \mathcal{V}, \mathcal{E} >$. Thus GEDSAC is a kind of maximum independent set problem.

In general, we can easily approximate an maximum independent set in two situations: there is a bound on the number of neighbors that a node may have, or there is a bound on the number of independent neighbors. In this section we will describe an algorithm that simplifies $\mathcal{G}$ so that later independent set techniques will be easier to apply.

In the remainder of the paper, we will use the following set-theoretic notation. We use $N(a)$ to denote the set of neighbors of node a, $N(a,B)$ to denote $N(a) \cap B$, and $N(A,B)$ to denote $\cup_{a\in A} N(a,B)$. We also use $G[U]$ to denote a subgraph of G that is induced by the node set U.

Given a subset U of $\mathcal{V}$ we we define $\iota(U)$ to be the maximum value of $\phi(\mathcal{I})$ where $\mathcal{I} \subset U$ is an independent set. If $\mathcal{I}^*$ is an independent set and $\phi(\mathcal{I}^*) = \iota(G)$, we say that $\mathcal{I}^*$ is an ϕ-MIS (a Maximum weighted Independent Set when ϕ is the weight function).

A graph $< V, E, \chi >$ is a *abstract breakpoint graph* if $\chi : E \to \{b, g\}$ is an edge coloring function (we choose our colors to be black and gray) and the following two conditions are satisfied:

1. $< V, \chi^{-1}(c) >$ is a collection of simple paths for $c = g, b$;
2. every node has the same number of neighbors in $< V, \chi^{-1}(b) >$ and in $< V, \chi^{-1}(g) >$.

Moreover, we have a set H of disjoint pairs of nodes from V that have degree 2, these pairs are the *abstract little hurdles.*

It is easy to see that an edge component, as introduced in the previous section, forms an abstract breakpoint graph if we refer to the breaks as the black edges and to the chords as the gray edges. Note that now we do not have the case when the same node pair is a gray edge and a black edge (it is unnecessary, because it such edge/pair of edges forms a separate edge component).

We define alternating cycles, i-cycles and DAC for abstract breakpoint graphs in the same manner as before. We distinguish a set $\mathcal{V}_{i,1}$ of $(i+1)$-cycles as abstract $(i,1)$-absorbers, these cycles must contain a pair of nodes that belong to an abstract little hurdle, and are separated by a path of 3 edges. We use $\mathcal{V}_{i,1}$ to denote the set of (abstract) $(i,1)$-absorbers.

We define the overlap graph $< \mathcal{G}, \mathcal{E} >$ as before, however, we change the functions γ and ϕ by multiplying them with 8, so they will have integer values. In terms of the new terminology, the task of GEDSAC is to find an independent set $\mathcal{I}$ such that $\gamma(\mathcal{I}) \geq \iota(\mathcal{V})$.

3.1 Simplifications of $\mathcal{V}_2$

The goal of this section is to reduce GEDSAC to the case when $\mathcal{G}_2 = \mathcal{G}[\mathcal{V}_2]$ is a graph of degree 4.

We will simplify the overlap graph by *selections* and *eliminations.*

A selection is to to pick a small independent set $\mathcal{I}_0$ and define $\mathcal{V}' = \mathcal{V} - \mathcal{I}_0 - N(\mathcal{I}_0, \mathcal{V})$; afterwards we will find an independent set $\mathcal{I}$ in $\mathcal{G}[\mathcal{V}']$ and return $\mathcal{I}_0 \cup \mathcal{I}$. A selection is valid if $\gamma(\mathcal{I}_0) \geq \iota(\mathcal{V}) - \iota(\mathcal{V}')$.

An elimination removes a set $\mathcal{S}$ from $\mathcal{V}$ and is valid if $\iota(\mathcal{V} - S) = \iota(\mathcal{V})$.

When we prove validity of a selection or an elimination, we may apply the following game. The opponent chooses $\mathcal{I}^*$, an independent set in $\mathcal{G}$. We modify $\mathcal{I}^*$ so that $\phi(\mathcal{I}^*)$ does not decrease, and then we show $\mathcal{J}^*$ such that $\mathcal{I}^* - \mathcal{J}^*$ is an independent set in the reduced $\mathcal{G}$. Our score is $\gamma(\mathcal{I}_0)$ (or 0 in elimination case). The opponent's score is $\phi(\mathcal{J}^*)$.

3.1.1 Simplifying $\mathcal{V}_{2.1}$

Consider a cycle $u \in \mathcal{V}_{2,1}$, we may assume that u is $(a, b, c, d, e, f \circlearrowleft)$ from Fig. 2.3.2a. The simplest selection rule is

Single selection rule: if $\iota(N(u, \mathcal{V})) \leq \gamma(u)$, make the selection of $\{u\}$.

In our case, $\gamma(u) = 16$.

We prove the following

Lemma 4 *After applying Single selection rule, each $u \in \mathcal{V}_{2,1}$ has at most 4 neighbors in $\mathcal{V}_2$.*

Proof. Cycle u has two node that have degree 2 that form an abstract little hurdle, namely b and e. Suppose that another node on this cycle has degree 2, say c. Then an overlap of u with another AC is one of the paths (a, b, c, d), (d, e, f) and (a, f), or a union of these paths. Clearly, in this case $\iota(N(u, \mathcal{V})| \leq 5|N(u, \mathcal{V})| \leq 15 < 16 = \gamma(u)$. Thus in $\mathcal{V}$ after applying Single selection rule, then only b and e have degree 2.

More precisely, we may assume that u has 4 non-overlapping neighbors in $\mathcal{V}$, and that these neighbors have the following overlaps with u: (a, b, c), (c, d), (d, e, f) and (a, f), let us call these neighbors $v_{a,c}$, $v_{c,d}$, $v_{d,f}$ and $v_{a,f}$.

Now suppose that there exist a gray edge (a, d). This edge must belong to $v_{c,d}$ and to $v_{a,c}$, a contradiction because these two ACs were assumed not to overlap. Non-existence of other diagonal edges follows from symmetric arguments.

Consequently no AC from $\mathcal{V}_2$ can overlap u on two of the paths (a, b, c), (c, d), (d, e, f) and (a, f). A 2-cycle would have to have a 4th edge forming a diagonal, and diagonals do not exist. A 3-cycle from $\mathcal{V}_{2,1}$ would contain either b or e as its third degree 2 node, and this is not possible either.

At this point, we can conceive one possibility for u to have more than 4 neighbors in $\mathcal{V}_2$: two of them have the same overlap with u. Suppose that (a, b, c) is this overlap. Then none of the two neighbors can be a (2,1)-absorber, because b would be its third node of degree 2. Thus we got that 2-cycles, $(a, b, c, g \circlearrowleft)$ and $(a, b, c, h \circlearrowleft))$. Because a is incident to two gray edges only, $g = h$, so this is one and the same cycle.

Suppose that (a, f) is this overlap. If one of the neighbors is a (2,1)-absorber, it must have the form $(a, b', c', d', e', f \circlearrowleft)$ where b' and d' are dgree 2 nodes, in this case both neighbors must share the path (c', b, a, f, e', d') and thus they are identical. If both neighbors are 2-cycles, the argument is similar.

❑

3.1.2 Double tangles

Simplification of $\mathcal{V}_{2,0}$ requires the use of several rules. In this subsection we will show the validity of the first one.

> **Double tangle selection rule:** If G contains an induced subgraph as shown in Fig. 3.1.2, make selection of $\mathcal{I}_0 = \{(a, \underline{b}, \underline{a}, b \circlearrowleft), (b, \underline{c}, \underline{b}, c \circlearrowleft), (c, \underline{d}, \underline{c}, d \circlearrowleft)\}$.

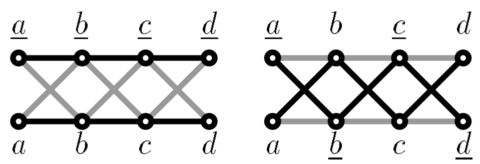


Figure 4: A double tangle, note two possible representations.

Before we prove the validity of Single selection rule, we will prove a lemma that makes this proof simpler and which is also used later.

Lemma 5 *Assume that (i) $\mathcal{I}^* \subset \mathcal{V}$ is an independent set that maximizes $\phi(\mathcal{I})$, (ii) a 2-cycle u contains no nodes of degree 2 and (iii) u is contained in a union of two elements of $\mathcal{I}^*$. Then there exists an independent set $\mathcal{I}' \subset \mathcal{V}$ that also maximizes $\phi(\mathcal{I})$ such that $u \in \mathcal{I}'$.*

Proof. Assume that $u \subset v_0 \cup v_1$ (here we view the cycles as edge sets). We can modify $\mathcal{I}^*$ by replacing v_0 and v_1 with u and $u_1 = (v_0 \cup v_1) - u$. Because the value of ϕ is a linear combination of the numbers of cycles, breaks and absorbed little hurdles, we have $\phi(\{v_0, v_1\}) = \phi(\{u, u_1\})$ unless the number of absorbed little hurdles changes in the process. Obviously, we worry only in this number decreases, which means that $\{v_0, v_1\}$ contains more absorbers than $\{u_1\}$.

Consider the case when $v_0 \in \mathcal{V}_{3,1}$ and $v_1 \in \mathcal{V}_{2,0}$ and to show that $u_1 \in \mathcal{V}_{3,1}$. Clearly, u_1 is a 4-cycle, and we need to show only that u_1 absorbs the little hurdle of v_0. Let h be the abstract little hurdle absorbed by v_0; none of the nodes of h belongs to u because u has no nodes of degree 2, thus both of them are in $v_0 - u$. As a result, one of the paths that connects h inside u_1 is contained in v_0 and therefore it has a correct length. We conclude that u_1 absorbs h.

The case when $v_0 \in \mathcal{V}_{2,1}$ and $v_1 \in \mathcal{V}_{3,0}$ is similar (we need to observe that $u \cap v_0$ is a single edge). When $v_0 \in \mathcal{V}_{2,1}$ and $v_1 \in \mathcal{V}_{3,1}$ then $\phi(u) = \phi(\{v_0, v_1\})$ so $\mathcal{I}' = \mathcal{I}^* - \{v_0, v_1\} cup \{u\}$. The case $v_0, v_1 \in \mathcal{V}_{2,1}$ is not possible because an intersection of u with an element of $\mathcal{V}_{2,1}$ can have one edge only.

❑

When we apply Lemma 5, we say that cycle u is *forced*.

Lemma 6 *Double tangle selection rule is valid.*

Proof. Let $v_0 = (a, \underline{b}, \underline{a}, b\circlearrowleft)$, $v_1 = (b, \underline{c}, \underline{b}, c\circlearrowleft)$, $v_2 = (c, \underline{d}, \underline{c}, d\circlearrowleft)$ and $I_0 = v_0 \cup v_1 \cup v_2$. Let $\mathcal{J}^*$ be the set of those cycles in $\mathcal{I}^*$ that overlap I_0, *i.e.* $\mathcal{J}^* = N(\mathcal{I}_0, \mathcal{I}^*)$. It suffices to show that $\phi(\mathcal{J}^*) \leq 24 = \gamma(\mathcal{I}_0)$.

First we show that $|\mathcal{J}^*| \leq 5$. Note that $|I_0| = 12$. An intersection of an AC u with I_0 is either u itself, or a path between two *contact* nodes $a, \underline{a}, d, \underline{d}$. One can see that such an intersection has at least 2 edges, and the that there are only 4 paths of 2 edges. Thus $4(|\mathcal{J}^*| - 4) \leq 12 - 4 \times 2$, implying $|\mathcal{J}^*| \leq 5$.

Since $|\mathcal{J}^*| \leq 5$, the only possibility for $\phi(J^*) > 24$ is when $|\mathcal{J}^*| = 5$ and $\mathcal{J}^* \subset \mathcal{V}_{2,0}$.

One can see that a 2-cycle that contain edge (b, c) must be contained in I_0: the only conceivable 2-cycle not contained in I_0 is $(\underline{a}, b, c, \underline{d}\circlearrowleft)$, but it would imply that $(\underline{a}, \underline{d})$ is a black edge and thus we have a black cycle, a contradiction. Thus the intersection of $u \in N(I_0, \mathcal{V}_{2,0})$ with I_0 that contains (b, c) must have 4 edges. If we another intersection of 4 edges, $2|\mathcal{J}^*| + 4 \leq 12$ and $|\mathcal{J}^*| \leq 4$.

Because we can apply the same argument to all edges of v_1, this cycle must be contained in $\mathcal{I}_0$.

We are left with 8 edges of $\mathcal{I}_0$ that belong to exactly 4 intersections, thus $(a, b, \underline{a})$ and $(\underline{a}, \underline{b}, a)$ are among these intersections. By Lemma 5, this implies that we can force v_0, and, by symmetry, we can do the same with v_2. Thus we can reduced the problem to the case when $\mathcal{J}^* = \mathcal{I}_0$.

❑

3.1.3 Tangles

Our subsequent two rules deal with smaller induced subgraphs called tangles. In this subsection we will formulate these rules and show their validity. We assume that the elimination rule can be applied only when selection rules cannot be.

Tangle selection rule: If G contains an induced subgraph as shown in Fig. 3.1.3, and node a has degree 2, then make selection of $\mathcal{I}_0 = \{(a, \underline{b}, \underline{a}, b\circlearrowleft), (b, \underline{c}, \underline{b}, c\circlearrowleft)\}$.

Tangle elimination rule: If G contains an induced subgraph as shown in Fig. 3.1.3, then remove the cycles $(a, \underline{b}, \underline{a}, b\circlearrowleft)$ and $(b, \underline{c}, \underline{b}, c\circlearrowleft)\}$ from $\mathcal{V}$.

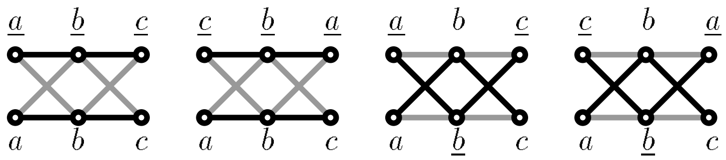


Figure 5: A tangle, note four possible representations.

Let I_0 be the edge set of the tangle. As before, $\mathcal{J}^*$ is the set of the ACs from $\mathcal{I}^*$ that overlap I_0. We start from the observation that any intersection of an AC with I_0 forms consists of one or of the following two alternating paths: $(a, b, \underline{c})$, $(a, b, \underline{a})$, $(a, \underline{b}, \underline{c})$, $(a, \underline{b}, \underline{a})$, $(\underline{a}, \underline{b}, c)$, $(\underline{a}, \underline{b}, a)$, $(c, \underline{b}, \underline{c})$, $(\underline{c}, \underline{b}, c)$.

Let u_0 and u_1 be the cycles selected by the Tangle selection rule. If a is a node of degree 2, than any AC that contains a must contain two of the above paths, one that starts with (a, b) and another that starts with $a, \underline{b}$. Thus $|\mathcal{J}^*| \leq 3$ and $\gamma(\{u_0, u_1\}) = 16 > 15 \geq \phi(\mathcal{J}^*)$, which shows that the Tangle selection rule is valid.

It is easy to see that if two different tangles are not edge disjoint, then together they form a double tangle, thus this is never the case when we apply Tangle elimination rule, so it cannot happen that different applications of this rule remove different cycles from the same tangle. It remains to show that we can modify $\mathcal{I}^*$ without decreasing $\phi(\mathcal{I}^*)$ in such a way that neither u_0 nor u_1 belongs to $\mathcal{I}^*$.

Let $v_0 = (a, b, \underline{c}, \underline{b}\circlearrowleft)$ and $v_1 = (\underline{a}, \underline{b}, c, a\circlearrowleft))$ be the 2-cycles contained in I_0 that are still in $\mathcal{V}$ after the application of the Tangle elimination rule. Suppose first that $\mathcal{I}^*$ contains both

u_0 and u_1, then we can force v_0 and v_1. Finally to consider the case when $\mathcal{I}^*$ contains exactly one of the removed cycles, say u_0. Then u_1 is contained in the union of two cycles of $\mathcal{I}^*$ and because we did not apply Tangle selection rule, none of the nodes of u_1 has degree 2. Consequently, we can force u_1, and later we can again force v_0 and v_1.

3.1.4 Possible sets of neighbors of a 2-cycle in $\mathcal{G}[\mathcal{V}_2]$

Now we will inspect possible sets of neighbors inside $\mathcal{G}[CV_2]$. in $\mathcal{G}[\mathcal{V}_{2,0}]$. A neighbor can share one edge, and then we will call it a straight neighbor at this edge, or two adjacent edges, and then we will call it a corner neighbor at this corner (the node where the common edges meet). Fig. 3.1.4a shows the straight neighbor of a 2-cycle $u = (a, b, c, d\circlearrowleft)$ at edge (a, b) and the corner neighbor at c. We start with

Observation 8 *A 2-cycle can have at most one straight neighbor at each edge and at most one corner neighbor at each corner.*

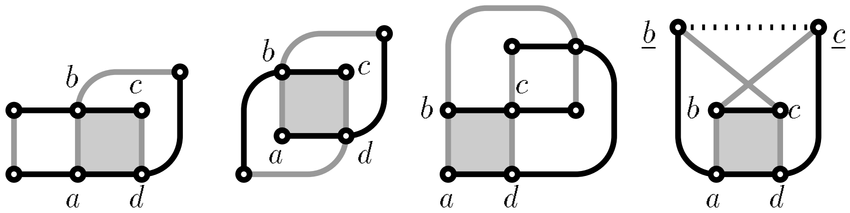


Figure 6: Various neighbor sets of 2-cycle $(a, b, c, d\circlearrowleft)$.

As it suffices to discuss eight possible neighbors, we can denote the corner neighbors with v_a, v_b, v_c, v_d and the straight neighbors $v_{ab}, v_{bc}, v_{c,d}, v_{ad}$. We will show that not all combinations of these neighbors are possible. First we add the following selection rule.

> **Double selection rule:** if $w \cup x$ is connected, w and x do not overlap and $\iota(N(\{w, x\}, \mathcal{V})) \leq \gamma(\{w, x\})$, make the selection of $\{w, x\}$.

The next two observations restrict the possible neighbors if one of the neighbors is an absorber.

Observation 9 *If v_a is a (2,1)-absorber, then v_b, v_c, v_d, v_{ab} and v_{ad} do not exist.*

Proof. ¿From the only possible form of a (2,1)-absorber (see Fig. 2.3.2), one can see that a must be one of the nodes of the abstract little hurdle, so it has degree 2. The existence of v_{ab}, v_{ad}, as well as the corner neighbors v_b, v_d would require a to have some other neighbor besides b and d. It remains to discuss v_c. We will show that if v_c exists, we can apply the Double selection rule to v_a and v_c. Fig. 3.1.4 illustrates two possible situations. An AC

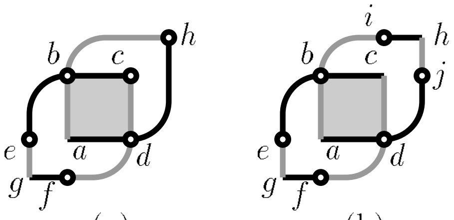


Figure 7: Various corner neighbors of $(a, b, c, d \circlearrowleft)$, nodes of degree 2 have the little circles missing.

containing a black edge incident to b must also overlap a gray edge incident to that node, and an AC containing a black edge incident to d must also contain a gray edge adjacent to that node.

Let $\mathcal{I}_0 = \{v_a, v_c\}$ and $\mathcal{J}^* = N(\mathcal{I}_0, \mathcal{I}^*)$. In case (a) this means that an AC overlapping $\mathcal{I}_0$ must contain one of the following 4 segments: (e, g, f), $b, a, d)$, (b, h) and (d, h), hence $\phi(\mathcal{J}^*) \leq 5|\mathcal{J}^*| \leq 20 < 24 = \gamma(\mathcal{I}_0)$. In case (b) this means that an AC overlapping $\mathcal{I}_0$ must contain one of the following 4 segments: $(e, g, f), (b, a, d), (b, i)$, (d, j) and (i, h, j), hence $\phi(\mathcal{J}^*) \leq 5|\mathcal{J}^*| \leq 25 < 32 = \gamma(\mathcal{I}_0)$.

❑

Observation 10 *If v_{ab} is a (2,1)-absorber then there are no corner neighbors v_a and v_b.*

This observation follows from the fact that v_a or v_b would share three edges with the absorber v_{ab}, which would imply a diagonal edge of the absorber, and that means that v_{ab} is eliminated by the Single selection rule.

In the next three observations it will be sufficient to consider exclusively neighbors of u that belong to $\mathcal{V}_{2,0}$, as the cases involving $\mathcal{V}_{2,1}$ are already discussed.

To make Observation 11, suppose that u has corner neighbors at to corners forming a diagonal pair, *e.g.,* v_a and v_c as in Fig. 3.1.4b. One can see that the the edges of v_a and v_c form a tangle. Because we have performed Tangle elimination, and we assume that u is still a node of $\mathcal{G}[\mathcal{V}_{2,0}]$, we can conclude that v_a and v_c were both eliminated and they are not neighbors of u anymore. Thus

Observation 11 *If a 2-cycle has a corner neighbor at some corner, it does not have one at the diagonal corner.*

Observation 12 is similar. Suppose that u has straight neighbors at two consecutive edges, and a corner neighbor adjacent to the same pair of edges. One can see (comp. Fig. 3.1.4c) that in this case the edges of these neighbors form a tangle. As a result, if $\mathcal{G}$ contains the corner neighbor, it does contains neither of the straight neighbors, and vice versa. Thus

Observation 12 *If a 2-cycle has a neighbor at some corner, it does not have a straight neighbor at one of the two edges adjacent to this corner.*

Now suppose that u has corner neighbors at b and c, $(a,b,c,\underline{b}\circlearrowleft)$ and $(b,c,d,\underline{c}\circlearrowleft)$ respectively (see Fig. 3.1.4d). Then, if we add a straight neighbor at $\{b,c\}$ we will close black cycle $(\underline{b},a,d,\underline{c}\circlearrowleft)$ which is not possible in a breakpoint graph. Thus

Observation 13 *If a 2-cycle has corner neighbors at two adjacent corners, it does not have a straight neighbor at the edge that joins these corners.*

While it is still possible for a 2-cycle u to have five neighbors in $\mathcal{G}[\mathcal{V}_{2,0}]$, we can restrict this situation to one case only. First, it is necessary that u has at least one corner neighbor, because there can be only four straight ones (Observation 8). However, if there is a corner neighbor, there are at most three straight neighbors (Observation 12), so it is necessary that u has at least two corner neighbors. However, a corner neighbor excludes another corner neighbor at the diagonal corner (Observation 11), so the only way to have at least two corner neighbors is to have exactly two, at adjacent corners, say v_a and v_b. Because there is no v_{ab} (Observation 13), the other straight neighbors, v_{bc}, $v_{c,d}$ and v_{ad} must exist. Fig. 3.1.4a shows this situation.

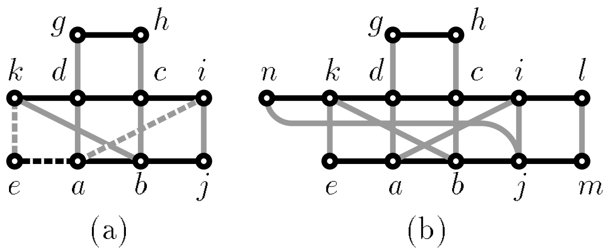


Figure 8: 2-cycle $(a,b,c,d\circlearrowleft)$ with five neighbors. Solid lines indicate possible commitments.

Note that v_{cd} can be a (2,1)-absorber, but v_a, v_b, v_{bc} and v_{ad} cannot, because each of them has a corner neighbor.

3.1.5 Eliminating 2-cycles with five neighbors

To eliminate a 2-cycle with five neighbors, we asume that it is $u = (a,b,c,d\circlearrowleft)$ and that $N(u,\mathcal{V}_2) = \{v_a, v_b, v_{ad}, v_{bc}, v_{cd}\}$. We will formulate three selection rules in terms of this notation. The first one is the following:

> **Fig. 3.1.4 absorber selection rule:** if $v_{cd} \in \mathcal{V}_{2,1}$, select $\mathcal{I}_0 = \{v_a, v_{bc}, v_{cd}\}$. selection of $\{w,x,y\}$.

To argue the validity of this rule, we can assume that $v_{cd} = (c, d, g', g, h, h' \circlearrowleft)$ where $\{g', h'\}$ is the abstract little hurdle of (2,1)-absorber v_{cd}. Consider a non-overlapping set $\mathcal{J} \subset N(\mathcal{I}_0, \mathcal{V})$. Observe that every element of $\mathcal{J}$ must contain either (g, h) or an edge of the path (b, k, d, c, i, j). For example, if an AC contains (b, j), then it either contains (b, k) — and we are done, or (b, c). In the later case this AC must contain one of the edges of (d, c, i). We conclude that $|\mathcal{J}| \leq 6$. Thus Fig. 3.1.4 absorber selection is valid: $\gamma(\mathcal{I}_0) = 32 > 6{\times}5 \geq \phi(\mathcal{J})$.

If $v_{cd} \in \mathcal{V}_{2,0}$, we have $\gamma(\mathcal{I}_0) = 24$. We could also apply Fig. 3.1.4 absorber selection rule, and start the argument in the same way, but $24 < 30$ and the argument may fail. Nevertheless, we will establish that there is only one case when such a rule would fail, and the will lead to the formulation of two valid rules. In other words, we want to find out when for every non-overlapping set $\mathcal{I}^*$ such that $\phi(\mathcal{I}^*)$ is maximal we have $\phi(\mathcal{J}^*) \geq 25$ for $\mathcal{J}^* = N(\mathcal{I}_0, \mathcal{V})$.

Suppose that there exists a black edge (j, h). Then we cannot have a black edge (e, g) as it would close a black cycle. We will assume that the black edge (e, g) does not exists, otherwise we would swap node names $a \leftrightarrow b$, $e \leftrightarrow j$, etc.

In our case analysis we use a notion of a *portal*, a node incident both to edges from I_0 and to other edges, and of a *crossing*, a pair of edges that are incident to the same portal, with different colors, one in I_0 and one not. A crossing is *used* if it is contained in a cycle from $\mathcal{I}^*$.

Case 1: $|\mathcal{J}^*| = 6$.

Case 1.1: all 6 cycles of $\mathcal{J}^*$ overlap I_0 but none is contained in I_0.

In this case at least 6 cycles contain two crossings each for the total of 12 crossings; because each portal is in the middle of exactly two crossing and there are 6 portals—nodes a, k, g, h, i and j—every crossing must be used. Consider edge (a, i), it is contained in crossings (a, i, c) and (b, a, i), thus a cycle from $\mathcal{J}^*$ contains path (b, a, i, c) and we can force cycle $v_b = (b, a, i, c \circlearrowleft)$. Now consider crossing (e, a, d); if the cycle of $\mathcal{J}^*$ that contains (e, a, d) contains (e, a, d, k), another cycle of $\mathcal{J}^*$ contains path (g, d, c, h), which means that we can force cycle $v_{cd} = (g, d, c, h \circlearrowleft))$ and thus exit Case 1. On the other hand, if the cycle containing (e, a, d) contains (e, a, d, c), we can force $u = (a, b, c, d \circlearrowleft)$ and exit Case 1.1 in this way.

Case 1.2: exactly 5 cycles from $\mathcal{J}^*$ overlap I_0 but are not contained in I_0. Then they leave only one portal to be used by the cycle that is contained in I_0 (the sixth element of $\mathcal{J}^*$), and because we must use b, c, d and one other node, cycle $u = (a, b, c, d \circlearrowleft)$ is the only possibility. Since we eliminated crossings centered at portal a, the remaining 10 crossings must be used. The cycle of $\mathcal{J}^*$ containing crossing (a, i, c) must contain path (e, a, i, c, h) and it cannot be a 2-cycle (otherwise $e = h$ and we have a gray cycle). The cycle of $\mathcal{J}^*$ that contains crossing (e, k, d) must contain path (e, k, d, g) and it cannot be a 2-cycle either, because $\{e, g\}$ is not a black edge. Because we have found two 3-cycles in $\mathcal{J}^*$, $\phi(\mathcal{J}^*) \leq 24$.

Case 2: $|\mathcal{J}^*| = 5$. Then $\phi(\mathcal{J}^*) \geq 25$ only if $\mathcal{J}^* \subset \mathcal{V}_{2,0}$.

Case 2.1: $u \in \mathcal{J}^*$. Then the cycle containing (g, d) also contains path (g, d, k), and because (g, e) is not a black edge, it cannot contain (g, d, k, e) and be a 2-cycle; thus it continues as (g, d, k, b, j). If this is a 2-cycle, we have $g = j$, and hence gray cycle $(j, i, a, d \circlearrowleft)$, a

contradiction.
Case 2.2: $u \notin \mathcal{J}^*$. Because a 2-cycle containing (c, d) belongs to $\mathcal{J}^*$, we must have $v_{cd} \in \mathcal{J}^*$. $\mathcal{J}^*$ also contains a 2-cycle with edge (a, b). If this is v_a, then edges of v_{bc} must belong to three cycles, impossible. Thus $v_b \in \mathcal{J}^*$, and $\{a, d\}$ must be contained in v_{ad}. The remaining two 2-cycles must contain paths (f, b, j) and (i, j), let us call them w and x, and their existence implies the configuration from Fig. 3.1.4b.

Our conclusion is that the following rule is valid:

> **Fig. 3.1.4a selection rule:** if configuration from Fig. 3.1.4a is not a part of a configuration from Fig. 3.1.4b, select $\mathcal{I}_0 = \{v_a, v_{bc}, v_{cd}\}$.

To complete the elimination of 2-cycles with 5 neighbors in $\mathcal{V}_2$ it remains to show the validity of our final rule

> **Fig. 3.1.4b selection rule:** if there exists a configuration from Fig. 3.1.4b, select $\mathcal{I}_1 = \{v_{ad}, v_b, v_{cd}, w, x\}$.

We have $\gamma(\mathcal{I}_1) = 40$, so to show that $\phi(\mathcal{J}^*) \leq \gamma(\mathcal{I}_1)$ it suffices to prove that $|\mathcal{J}^*| \leq 8$. Let I_1 be the edge set of $\mathcal{I}_1$. We have two cases.
Case i: at least six cycles of $\mathcal{J}^*$ are not contained in I_1. Because I_1 has six portals, as in case **1.1**, there must be exactly six such cycles, and the remaining cycles can use only edges between the inner nodes, a, b, j and k, d, c, i. There are at most two of them, because they must overlap path (a, b, j).
Case ii: exactly $m \leq 5$ cycles of $\mathcal{J}^*$ are not contained in I_1. Then the cycles of $\mathcal{J}^*$ that are contained in I_1 together have at most $20 - m$ edges, hence there are at most $\lfloor (20 - m)/4 \rfloor$ of them. Therefore $|\mathcal{J}^*| \leq m + 5 - \lceil m/4 \rceil \leq 8$.

We can summarize this section with the following theorem.

Theorem 2 *Given $\mathcal{V}_{2,0}, \mathcal{V}_{2,1}, \mathcal{V}_{3,0}$ and $\mathcal{V}_{3,1}$ defining an instance of GEDSAC problem, we can in linear time find an equivalent instance for which $\mathcal{G}[\mathcal{V}_2]$ is a graph of degree 4.*

3.1.6 Simplification of $\mathcal{G}[\mathcal{V}_{3,0}]$

Suppose that a 3-cycle w contains three edges of some 2-cycle u. Then given a non-ovelapping set $\mathcal{I}^*$ such that $w \in \mathcal{I}^*$, we can insert u and remove w, plus possibly another AC, say x, that overlaps the fourth edge of u. If x does not exists or $x \notin \mathcal{V}_{2,0}$, then $\phi(\mathcal{I}^*)$ increases. Because we do not have to consider cycles that do not belong to some $\mathcal{I}^*$ with maximal $\phi(\mathcal{I}^*)$, the following rule is valid:

> **3-cycle elimination rule:** if $w \in \mathcal{V}_{3,0}$, $u \in \mathcal{V}_{2,0}$, w and u share three edges and $N(u, \mathcal{V}_2) \subset N(w, \mathcal{V}_2)$, then remove w from $\mathcal{V}$.

3.2 GEIS — Good enough independent set problem

Our goal is to form a more abstract problem in which we do not know that a selected set of objects consists of cycles and that the dependency edges are cycle overlaps. The input to this problem will be a graph G with the following *basic properties*:

1. set of nodes V has four parts, $V_{2,0}, V_{2,1}, V_{3,0}, V_{3,1}$ (where $V_i = V_{i,0} \cup V_{i,1}$);
2. $G[V_2]$ is a graph of degree 4;
3. if S is an independent set, $u \in V_i$ and $S \subset N(u, V)$ then $|S| \leq 2i$;
4. if $u \in V_{i,j}$, then $\gamma(u) = 8(j+1)$ and $\phi(u) = 8 - 3(i-1) - j$.

The goal is to find an independent set I such that for every other independent set I^* we have $\gamma(I) \geq \phi(I^*)$.

Unfortunately, we did not succeed in that goal, and we need to require that an *abstract overlap graph* satisfies all the basic properties and two additional properties that are more complex. In this section we will formulate this property and then we will show that the overlap graphs resulting from the applications of the selection and elimination rules of the previous section indeed satisfy this property,

We start from the following definition:

Definition 1 *Assume that $I \subset V$ is an independent set.*

- *$mix(I, I^*) = \{\{u, w\} \in E : \ u \in I \cap V_{2,0}$ and $w \in I^* \cap V_{3,0}\}$;*
- *A pair $(u, \underline{u})$ is dangerous for I if for some I^* with maximal $\phi(I^*)$, and, under this restriction, with minimal $mix(I, I^*)$ we have*

  1. $u \in I \cap V_{2,0}$, $\underline{u} \in I^* \cap V_{2,0}$;
  2. $N(\underline{u}, I) = \{u\}$;
  3. $|N(\underline{u}, V_2)| = 4$; and
  4. $|N(u, I^* \cap V_{3,0})| = 3$;

The motivation to consider the danerous pairs is the following: the amortized analysis of our algorithm for GEIS breaks down when there exist a dangerous pair for the analyzed solution.

We say that a pair $(u, \underline{u})$ is *flipped* if we change I into $I' = I - \{u\} \cup \{\underline{u}\}$. We can always flip a dangerous pair, because condition 2 says that u is the only neighbor of $\underline{u}$ in I. We say that a flip is safe if $(\underline{u}, u)$ is not dangerous for I'. Note that a flip can be safe even if $(u, \underline{u})$ is not dangerous for I. We need to show that (a) we have a condition that a dangerous pair must satisfy, and (b) if $(u, \underline{u})$ satisfies this condition for I then $(\underline{u}, u)$ does not satisfy this condition for I'. In other words, our flips will destroy all dangerous pairs, without decreasing $\gamma(I)$.

Our condition will consist of two three parts..

We say that $(u, \underline{u})$ is *awkward* for I if $u \in I \cap V_2$, $\underline{u} \in V_2 - I$ and $(u, \underline{u})$ satisfies properties 2 and 3 of a dangerous pair. Clearly, a dangerous pair must be awkward, so if $(\underline{u}, u)$ is not awkward for I', the flip of $(u, \underline{u})$ is safe. This happens if $|N(u, V_2)| < 4$.

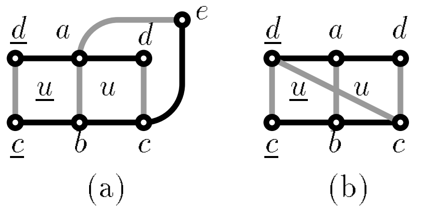


Figure 9: Corner neighbors in a dangerous pair.

If $(u, \underline{u})$ is awkward and $|N(u, V_2)| = 4$, then we say that A is a *trouble* for $(u, \underline{u})$ if $A \subset N(u, V)$, A is independent, $A \cap V_2 = \{\underline{u}\}$ and $|A \cap V_{3,0}| = 3$. Set A is a plausible $N(u, V) \cap I^*)$ from the definition of a dangerous pair.

However, it may happen that it is impossible that $A = N(u, V) \cap I^*)$ for an independent set with maximal $\phi(I^*)$ and minimal $mix((I, I^*)$. Thus we say that an independent set B is a solution for trouble A if $B \cup N(B, V) \subset A \cup N(A, V)$ and either $\phi(B) > \phi(A)$ or $\phi(B) \geq \phi(A)$ and $mix(I, B) < mix(I, A)$.

If $A = N(u, V) \cap I^*$ than $I^* - A \cup B$ is also an independent set with maximal $\phi(I^*)$, but $mix(I, I^* - A \cup B) < mix(I, I^*)$ and thus A does not corroborate the fact that $(u, \underline{u})$ is dangerous. We can conclude that

Observation 14 *Pair $(u, \underline{u})$ is dangerous for I only if it is awkward for I and it has a trouble with no solution.*

If a pair satisfies the above condition, we say that it is *troublesome* for I. Now we can phrase the fifth desired property of the inputs to GEIS problem:

5. If I is an independent set and $(u, \underline{u})$ is troublesome for I, then $(\underline{u}, u)$ is not troublesome for $I - \{u\} \cup \{\underline{u}\}$.

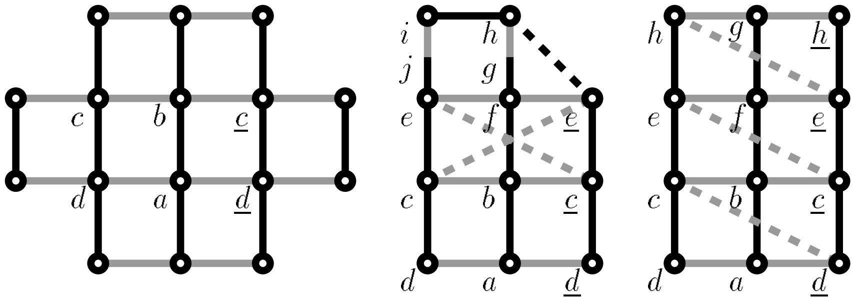
We can postulate that the GEIS input satisfies property 5 because we can prove the following theorem.

Lemma 7 *After applications of the rules of selection and elimination the overlap graph of an abstract breakpoint graph satisfies properties 1-5.*

Proof. Conditions 1-4 are obvious, so we will be proving only condition 5. Consider a troublesome pair $(u, \underline{u})$.

If $|N(u, \mathcal{V}_2)| < 4$, then $(\underline{u}, u)$ is not awkward, and as such, not troublesome. Therefore we may assume that u has 4 neighbors in V_2. We will also assume that $u = (a, b, c, d \circlearrowleft)$, $\underline{u} = (a, b, \underline{c}, \underline{d} \circlearrowleft)$. The trouble for $(u, \underline{u})$ is an independent set of 4 neighbors of u, thus each of these neighbors overlaps u on exactly one edge. We may assume that $A = \{\underline{u}, w_{ad}, w_{bc}, w_{cd}\}$, where each w is a 3-cycle that overlaps u on the indicated edge.

The form of the neighbors of u in V_2 is also restricted, as we show below.



(a) (b) (c)

Figure 10: Neighbors of a troublesome pair $(u, \underline{u})$.

Lemma 8 *If* $(u, \underline{u})$ *is troublesome for* $\mathcal{I}$*, then* u *has no neighbors in* $\mathcal{V}_{2,1}$ *and no corner neighbors in* $\mathcal{V}_{2,0}$.

Proof. Supose first that u has a neighbor in $\mathcal{V}_{2,1}$, say v. If v and u share two edges, say (b, c, d), then node c has degree 2 and that precludes the existence of w_{bc} and w_{cd}. If v and u share one edge, say (b, c), then w_{bc} and v share at least 5 edges, not possible.

Now suppose that u has a corner neighbor at d, for some e this 2-cycle equals $v = (a, d, c, e \circlearrowleft)$ (see Fig. 3.2a). Then w_{ad} contains path (d, a, e) and w_{cd} contains (d, c, e). This means that $B = \{\underline{u}, w_{bc}, v\}$ uses only the cycle edges of A, while $\phi(B) = 12 > 11 = \phi(11)$; thus B is a solution to trouble A, a contradiction.

Finally suppose that u has a corner neighbor at b. One can see that this 2-cycle must contain the path $(c, b, a, \underline{d})$ so it equals $v = (c, b, a, \underline{d} \circlearrowleft)$ (see Fig. 3.2b). Then w_{bc} must contain the path $(b, c, \underline{d})$ while $\underline{u}$ contains $(b, a, \underline{d})$. In this case $B = \{v, w = \underline{u} \cup w_{bc} - v, w_{ad}, w_{cd}\}$ is a solution to trouble A: it uses the same set of cycle edges, $\phi(B) = \phi(A)$ and $mix(B, I) = mix(A, I) - \{\{w_{bc}, u\}\}$.

❑

Because each neighbor that u has in $\mathcal{V}_2$ shares exactly one edge with u, we can use u_{ad}, u_{bc}, u_{cd} to denote these neighbors (while $u_{ab} = \underline{u}$).

If $(\underline{u}, u)$ is also troublesome, we would be able to define, by analogy, the neighbors of $\underline{u}$ denoted $\underline{u}_{a\underline{d}}, \underline{u}_{b\underline{c}}, \underline{u}_{\underline{cd}}, \underline{w}_{a\underline{d}}, \underline{w}_{b\underline{c}}$ and $\underline{w}_{\underline{cd}}$. Fig. 3.2a without the dashed edges shows u and $\underline{u}$ together with their neighbors in $\mathcal{V}_{2,0}$.

Consider 3-cycle w_{bc}, it clearly shares three edges with $u_{bc} = (b, c, e, f \circlearrowleft)$ and because it is not a subject of 3-cycle elimination rule, there must exist a cycle $v \in \mathcal{V}_2$ that does not overlap w_{bc} but uses cycle edge (e, f).

Suppose that $v \in \mathcal{V}_{2,1}$, as in Fig. 3.2b where $v = (e, f, g, h, i, j \circlearrowleft)$, where $\{g, j\}$ is a little hurdle. Because $w_{b\underline{c}}$ is not a subject of 3-cycle elimination rule, there must exist 2-cycle $(\underline{e}, f, g, h \circlearrowleft)$, thus $(\underline{e}, h)$ is a black edge. Cycle $\underline{w}_{b\underline{c}}$ must contain path $(\underline{e}, \underline{c}, b, f, e)$. If it contains also (e, j) then it contains path $(\underline{e}, \underline{c}, b, f, e, j, i)$ but that means that $\underline{e} = j$, a contradiction, because edge $(\underline{e}, f)$ becomes a diagonal of v, and that makes v a subject of Single selection rule. Thus has to use edge (e, c), so its 6th edge is gray edge $(c, \underline{e})$.

Consider w_{bc}: it contain path $(e, c, b, f, \underline{e})$, if it also contain $(\underline{e}, h)$ then its 6th edge is (e, h), a diagonal of v, a contradiction. Thus w_{bc} is closed with two edge path $(\underline{e}, \underline{c}, e)$, so $(\underline{c}, e)$ is a gray edge. Contradiction: we have obtained a gray cycle $(c, b, \underline{c}, e, f, \underline{e} \circlearrowleft)$.

It remains to consider the case when the cycles that assure that neither w_{bc} nor $\underline{w}_{b\underline{c}}$ is subject of 3-cycle elimination are both 2-cycles, as in Fig. 3.2c. We again consider the possibilities for w_{bc} and $\underline{w}_{b\underline{c}}$; the former must use a gray edge from e to $\underline{h}$ or $\underline{c}$, and the latter must use a gray edge from $\underline{e}$ to h or c. To avoid a gray cycle, we must have a pair of gray edges as in Fig. 3.2c or a symmetric one. Therefore we can assume two gray edges $(h, \underline{e})$ and $(e, \underline{c})$, and thus a new 2-cycle $x = (e, \underline{c}, \underline{e}, h \circlearrowleft)$. This cycle shares 3 edges with $\underline{w}_{b\underline{c}}$. Because $\underline{w}_{b\underline{c}}$ is not a subject of 3-cycle elimination, there must exist a -cycle y that uses $(e, \underline{c})$, the single edge of $x - \underline{w}_{b\underline{c}}$ and non-overlapping with $\underline{w}_{b\underline{c}}$. One can see that y must contain gray edge $(c, \underline{d})$.

We have a contradiction, because $\underline{w}_{c\underline{d}}$ must contain path $(h, e, \underline{c}, \underline{d}, c, d)$, which implies that (d, h) is a gray edge and that $(d, h, \underline{e}, f, e, \underline{c}, b, c, \underline{d}, a \circlearrowleft)$ is a gray cycle.

❑

The final property is a bit simpler. Let $A \subset V_2$ be an independent set. We define set K to be an A-*butterfly* with center $u \in A$ if there exist $L \subset N(u, V)$ such that $L \cup A - \{u\}$ is independent, $|H| \geq 3$, $K = H \cap V_{3,0}$ and $|K| \geq 2$. A-butterfly K touches A-butterfly K' a times if there are a edges between elements of K and K'. A set of butterflies independent of no two elements touch each other.

6. If K is an A-butterfly, and $\mathcal{K}$ is an independent set of butterflies, K may touch elements of $\mathcal{K}$ at most $5|K|$ times.

Lemma 9 *The overlap graph of an overlap graph of an abstract breakpoint graph satisfies property 6.*

Proof. Each element of an A-butterly K with center u is a cycle of 6 edges, of which 1 or 2 belong to u – otherwise K would not be a subset of an independent set from $N(u, V)$ with at least 3 elements. Moreover, at most one butterfly element shares 2 edges with u.

One can see that K has at most 15 edges that do not belong to u. If K touches another A-butterly K', then they share one of these edges. This claim is obvious if u is not a center of K'. Otherwise both K and K' contain at least 4 out of 8 edges that are incident to u but do not belong to u. The proof can be completed with a simple case analysis.

❑

We can conclude this section with the following theorem.

Theorem 3 *After applications of the rules of selection and elimination the overlap graph of an abstract breakpoint graph satisfies properties 1-6.*

4 Small improvements and complements

4.1 The method

In this section we will describe 5 algorithms for various versions of Independent Set problem. The first 4 form a sequence, as each invokes the previous as a subroutine, and the last algorithm in that sequence, called Main, finds a solution to GEIS problem, or, to be a bit more precise, almost a solution, as the condition that GEIS solution must satisfy may be still unfulfilled. However, this "almost solution" will have properties that allow The fifth algorithm, which we call Postprocessing, to improve it so that GEIS condition is assured.

The reason for splitting our algorithm into Main and Postprocessing is that all five algorithms have a similar nature: apply a single set of rules as long as possible to improve the tentative solution, and then perform the analysis under the assumption that none of the rules apply. Main and Postprocessing use different sets of rules, and even have different sets of objects among which the search for the solution is conducted.

We introduce new graph notation. If X is a set of nodes, $X(i)$ is the set of those elements of X that have exactly i neighbors (or, degree equal i), $X(\leq i) = \bigcup_{j=0}^{i} X(j)$. Each of the first three problems will have a fixed weight function ϕ, $\iota(X)$ is the maximum value of $\phi(I)$ for an independent subset of X, I^* is an independent set such that $|I^*| = \iota(V)$ and the goal is to find an independent set J such $|J| \geq \phi(V)$.

The common outline of our algorithms is the following. Initially, we use a number *preprocessing rules* that replace the given graph with a smaller one; once not preprocessing rule can be applied, we form an empty candidate set J. Afterwards, we perform, as long as possible, two kinds of operations that increase the size of the candidate: *small improvements* and *complement improvements*. More formally,

1. A preprocessing rule r has condition ψ, instance translation σ_r and solution translation τ_r. If the condition $\psi_r(p)$ holds for some parameter p, we replace G with $G' = \sigma_r(p, G)$ and later look for a solution in G'. Once we select such a solution J, we return an independent set of G equal to $\tau_r(p, J)$.

2. If we are given a list of preprocessing rules, we always apply the first possible rule. This way the condition of a rule tacitly assumes that the conditions of the previous rules do not hold.

3. A possible small improvement is an independent set X of size at most k, where k is a constant. Applying X means that J is replaced with $J - N(X, J) \cup X$. X is an improvement if applying X increases the size of J. We may also specify an objective function h; in such a case X is an improvement if applying X increases $h(J)$.

4. To attempt a complement improvement, we are finding an independent set J' in the graph $G[V - J]$ using the *complement algorithm* $\mathcal{A}$. If $|J'| > |J|$, we apply this improvement by replacing J with J'. Before we attempt a complement improvement,

we make sure that no small improvements apply. Thus if we attempt a complement improvement and it cannot be applied, the algorithm terminates.

4.2 Algorithm 1

The first algorithm will be used in the graphs where $V = V(\leq 3)$. We define $\phi(u) = {}^1/_2$ if $u \in V(3)$, otherwise $\phi(u) = 1$.

This algorithm was studied already by Halldórsson and Yoshikara [HY99] who proved 9/7 approximation ratio. Unfortunately, it is not clear how to adapt their analysis for our purpose.

We use four preprocessing reductions, simplicial, branchy, 2-greedy, and greedy.

The condition $\psi_s(u)$ of the simplicial reduction is that the set $N(u, V)$ is a clique; $\sigma_s(u, G) = G[V - N(u, V) - \{u\}]$ and $\tau_s(u, J) = J \cup \{u\}$.

The condition $\psi_b(u, v, w, x)$ of the branchy reduction holds if $\{v, w\} \subset V(2)$ and (u, v, w, x) is a path. We form $\sigma_b(u, v, w, x, G)$ by adding the edge $\{u, x\}$ to $G[V - \{v, w\}]$; in turn, $\tau_b(u, v, w, x, J)$ equals $J \cup \{u\}$ if $J \cup \{u\}$ is independent and $J \cup \{v\}$ otherwise.

The condition of 2-greedy reduction, $\psi_{g2}(u)$, holds if $u \in V(2)$, and the condition of the greedy reduction, $\psi_g(u)$, holds if $u \in V$. The instance and solution translations of these reductions are the same as for the simplicial reduction.

The validity of the first three rules is easy to see. When we can apply the greedy rule, we have $V = V(3)$, thus a maximum independent set has size at most $|V|/2$ and $\iota(G) \leq |V|/4$. Because each preprocessing step decreases $|V|$ by at most 4, we must obtain an independent set of size at least $|V|/4$.

4.3 Algorithm 2

The second algorithm will be used in the graphs where $V = V(3) \cup V(4)$. We define $\phi(u) = {}^2/_3$ if $u \in V(3)$ and $\phi(u) = {}^1/_3$ if $u \in V(4)$.

We have no preprocessing in this algorithm. We use small improvements with size bound 2 and objective function ϕ.

As a complement algorithm we use Algorithm 1. We can do it because when there are no small improvements, every node in $V - J$ has a neighbor in J, as a result in $G[V - J]$ a node has at most 3 neighbors.

Consider now independent sets J computed by Algorithm 2 and J^*. We define $C = J \cap J^*$, $A = J - J^*$ and $B = J^* - J$. Moreover, we define B^1 as the set of these nodes of B that have exactly one neighbor in J and $B^2 = B - B^1$.

On each node of J we put potential 6, and on each node u of J^* we put potential $-6\phi(u)$ (-4 if $u \in J^*(3)$ and -2 if $u \in J^*(4)$). It suffices to prove that the total potential is not negative.

Because we cannot apply small improvements anymore, each node in $V - J$ has a neighbor in J, and thus at most three neighbors in $V - J$. Moreover, if $u \in B^2$, then u has at least two neighbors in J, and thus at most two neighbors in $V - J$. A node in $B^1(3)$ also has at most two neighbors in $V - J$. Thus we can guarantee that the complement algorithm can find a set of size at least $|B^2 \cup B^1(3)| + |B^1(4)|/2 \leq |J|$, where the inequality is implied by the fact that we cannot apply the complement algorithm anymore. Thus we do not increase the total potential when we increase it by 1 for each node in $B^1(4)$, by 2 for each node in $B^2 \cup B^1(3)$ and decrease by 2 for each node in J.

Later, for brevity, we will say that we increase [the potential of] a node or decrease [the potential of] a node.

Next, for every edge $\{u, v\}$ such that $u \in A$ and $v \in B$ we decrease u by 1, and decrease v by 1. The table below shows the potentials at the beginning, and after each of the two changes.

$A(3)$	$A(4)$	$B^1(3)$	$B^1(4)$	$B^2(3)$	$B^2(4)$	$C(3)$	$C(4)$
6	6	-4	-2	-4	-2	$6-4$	$6-2$
4	4	-2	-1	-2	0	0	2
1	0	-1	0	0	2	0	2

Note that the only nodes with negative values are in $B^1(3)$, and this value is -1. To finish the argument, we define $n : B^1(3) \to A$ such that $N(u, J) = \{n(u)\}$. Because we do not have small improvements with objective function ϕ, $\phi(n(u)) \geq \phi(u)$, thus $n(u) \in A(3)$ and $n(u)$ has potential 1. Moreover, because we do not have improvements of size 2, the function n is 1-1. We can conclude that the total potential is non-negative.

4.4 Algorithm 3

Like Algorithm 1, Algorithm 3 will be used in the graphs where $V = V(\leq 3)$. We define $\phi(u) = {}^2/_3$ if $u \in V(3)$, otherwise $\phi(u) = 1$.

The preprocessing uses four reductions: simplicial, branchy, almost-greedy and make-$V(4)$. When we can apply neither the simplicial reduction, nor the branchy one, $V = V(2) \cup V(3)$ and $V(2)$ is an independent set. To describe the remaining two reductions, define E' to be the set of edges incident to $V(2)$, and consider a connected component of (V, E'), say C. Observe that $|C(3)| \leq |C(2)| + 1$.

The almost greedy reduction has condition $\psi_{ag}(C)$ that holds if $|C(2)| > 1$; $\sigma_{ag}(C, G) = G[V - C]$ and $\tau_{ag}(C, J) = J \cup C(2)$.

The make-$V(4)$ has condition $\psi_{m4}(C)$ holds if $C(2) \neq \emptyset$. Because the other reduction do not apply, $|C(2)| = 1$, $C(3) = 2$ and $C(3)$ is independent. We form $\sigma_{m4}(C, G)$ from $G[V - C]$ by adding a new node n, and edges that connect n with nodes in $N(C, V - C)$. We define $\tau_{m4}(C, J)$ to be $J - \{n\} \cup C(3)$ if $n \in J$ and $J \cup C(2)$ otherwise. Note that usually the new node will belong to $V(4)$, and this motivates the name of this reduction.

Once no preprocessing reduction is applicable, we apply Algorithm 2. The validity of the above reductions is easy to show and we leave it to the reader.

Small improvements:

1. of size at most 8 that increase $|J_2|$;
2. of size at most 9 that do not change $|J_2|$ but increase $|J_3|$;
3. of size 1 that change neither $|J_2|$ nor $|J_3|$ but increase $\gamma(J)$;
4. safe flips (as described in 3.2).

Complement algorithm: run Algorithm 3 in $G[V_2 - J]$, take the result if it increases J_2.

Figure 11: Algorithm Main.

4.5 Algorithm Main

In this section we describe the main part of our algorithm that solves GEIS problem. Our input is an abstract interleaving graph G, i.e. a graph with node set V partitioned into four parts $V_{2,0}, V_{2,1}, V_{3,0}$ and $V_{3,1}$ with properties 1-6 described in 3.2. We will analyze Main in a similar manner as Algorithm 2, i.e. we will be eliminating negative potential from the graph until it remains only in few cases of nature that is easy to determined. The elimination of this remaining negative potential is the problem solved by Postprocessing.

We introduce the following notation: if X is a node set, $X_{i,j} = X \cap V_{i,j}$ and $X_i = X_{i,0} \cup X_{i,1}$. Algorithm Main has no pre-processing and is described in Fig. 11.

Assume that algorithm Main terminated and that J^* is an independent set that maximizes $\phi(J^*)$ and minimizes $mix(J, J^*)$. We place potential $\gamma(u)$ on each $u \in J$ and $-\phi(u)$ on each $u \in J^*$. Our task is to have non-negative sum of values. We will be increasing and decreasing the values of the nodes until we will be left with small negative sum $-d$, where d is our deficit. The structure of this deficit will allow us to increase $\gamma(J)$ by at least $d/8$ using a *postprocessing algorithm*.

We define A, B and C in terms of J and J^* as in the analysis of Algorithm 2. We will also use the following notation:

$B_2^1 = \{u \in B_2 : \ |N(u, A_2)| = 1\}$ and $A_2^1 = N(B_2^1, A_2)$;
$B_2^{1,4} = \{u \in B_2^1 : \ |N(u, B_2)| = 4\}$ and $B_2^{1,3} = B_2^1 - B_2^{1,4}$;
if $N(u, A_2) = \{v\}$ for $u \in B_2^1$, then $n(u) = v$;
$B_2^2 = B_2 - B_2^1$ and $A_2^2 = A_2 - A_2^1$;
$B_2^{2,i} = \{u \in B_2^2 : \ |N(u, A - A_{2,0}^1)| = i\}$;
if $N(u, A_2^2) = \{v\}$ for $u \in B_{2,1}^2$, then $n(u) = v$;
$B_3^1 = \{u \in B_3 : \ |N(u, A)| = 1\}$ and $B_3^2 = B_3 - B^1$.

$a(u) = |N(u, A_3)|$; $b(u) = |N(u, B)|$;

We observe first that in $G[V_2 - J]$ Algorithm 3 can find an independent set J' of size at least $|B_2^2 \cup B_2^{1,3}| + 2/3|B_2^{1,4}|$. Because $|J'| \le |J_2|$, we do not increase the total potential when we increase each node in $B_2^2 \cup B_{2,3}^1$ by 3, increase each node in $B_{2,4}^1$ by 2, and decrease each node in $A_2 \cup C_2$ by 3. The table below shows lower estimates of the potential of nodes before and after in this redistribution.

$v \in \dots$	$B_{2,0}^{1,4}$	$B_{2,1}^{1,4}$	$B_2^{1,3} \cup B_2^2$	C_2	$A_{2,0}$	$A_{2,1}$
before	-5	-4	-5	$-5+8$	8	16
after	-3	-2	-2	$-5+5$	5	13

Observe that we can remove from consideration C_2, because the total potential of C_2 is non-negative. Moreover, $u \in A_{2,1}$ has 13, at most 4 neighbors in B, and each such neighbors has at least -3; thus we can remove from consideration $A_{2,1}$ and $N(A_{2,1}, B)$. Later, A_2 will denote $A_{2,0}$.

In our second redistribution we inspect every edge $\{u, v\}$ such that $u \in A$ and $v \in B$; we decrease u by 1, and increase v by the same amount.

Elements of $B_2^2 \cup B_3^2$ start the second redistribution with potential -2 and gain at least 2, so the potential becomes nonnegative. Elements of $B_{3,1}$ start the second redistribution with -1 and gain at least 1, so they also get nonegative potential.

The table below provides remaining lower estimates of potential values.

$B_{3,0}^1$	$B_2^{1,3} \cup B_{2,1}^{1,4}$	$B_{2,0}^{1,4}$	A_2	$A_{3,0}$	$A_{3,1}$
-2	-2	-3	5	8	16
-1	$-1 + a(u)$	$-2 + a(u)$	$5 - b(u)$	2	10

Before we rearrange the potential again, we need two lemmas.

Lemma 10 *Function n is an injection from $B_2^1 \cup B_{2,1}^2$ to A_2.*

Proof. Assume that $u \ne v$ and $n(u) = n(v) = x$. If $u, v \in B_2^1$, then $N(\{u, v\}, J) = \{x\}$, so we could apply $\{u, v\}$ to increase J, impossible after the termination of the algorithm. If $u \in B_2^1$ and $v \in B_{2,1}^2$, then $n(u) \in A_2^1$ and $n(v) \in A_2^2$, so $n(u) \ne n(v)$. If $u, v \in B_{2,1}^2$, then for $z = u, v$ we define $A(z) = N(z, A_2^1)$, and because n is a bijection between B_2^1 and A_2^1, we can also define $B(z) = n^{-1}(A(z))$. We can apply $\{u, v\} \cup B(u) \cup B(v)$ to increase $|J_2|$, because $|N(\{u, v\} \cup B(u) \cup B(v), J_2)| = |\{x\} \cup n(B(u) \cup n(B(v))| < |\{u, v\} \cup n(B(u) \cup n(B(v))|$, and again, this is impossible.

❑

Lemma 11 $B_2^{2,0} = \emptyset$.

Proof. Suppose $u \in B_2^{2,0}$. We can define $A(u)$ and $B(u)$ as in the proof of Lemma 10. We can apply $\{u\} \cup B(u)$ to increase J_2, because $|\{u\} \cup B(u)| = 1 + |B(u)| = 1 + |A(u)| = 1 + |N(\{u\} \cup B(u), J_2)|$.

❑

Our goal is to eliminate the negative potential except for the following situation: $u \in B^1_{3,0}$ has potential -1 and its sole neighbor in A_2 has potential 1. We define sets $A_- = A^1_2$ and $B^i_- = N(A_-, B^i_2)$ for $i = 1, 2$. Our third redistribution is a series of rules.
Rule 1: $u \in B^1_-$ and the potential of u is nonnegative; remove u from B^1_- and $n(u)$ from A_-.
Rule 2: $u \in B^2_-$, $|N(u, A - A_-)| \geq 2$ and $v \in N(u, A_-)$; undo the second operation for $\{v, u\}$, *i.e.* add 1 to v and subtract 1 from u, then remove u from B^2_-.
Rule 3: $u \in B^2_-$ and $|N(u, A - A_-)| < 2$ (by Lemma 11 $N(u, A - A_-) = \{n(u)\}$) and $N(n(u), B^1_{3,0}) = \varnothing$; subtract 1 from the potential of $n(u)$ and add 1 to the potential of u, then perform the actions of Rule 2.
Rule 4: $u \in B^2_-$; define $A_2(u) = N(u, A_-) \cup \{n(u)\}$, $B_2(u) = n^{-1}(A_2(u))$ and $B_3(u) = N(A(u), B^1_3)$, change the potential of all nodes in these three sets to 0, remove $A_2(u)$ from A_- and $B_i(u)$ from B^i_- for $i = 2, 3$..

While the validy if the first three rules is obvious, for Rule 4 we need a proof.

Lemma 12 *Rule 4 is valid.*

Proof. We need to show that the sum of the potential of $A_2(u) \cup B_2(u) \cup B_3(u)$ is nonnegative. Define $A_3(u) = N(B_2(u), A_3)$, let $a = |A_3(u)|$, $b = |B_3(u)|$ and $c = |A_2(u)|$. By Lemma 10, $|B_2(u)| = c$. Clearly, $c \leq 4$ and, because we have applied Rules 1 and 2, $a < c$. Suppose that $b > a$, then we can find a subset $B' \subset B_3(u)$ such that $|B'| = a + 1$ and the set $B_2(u) \cup B'$ forms an improvement that does not change $|J_2|$, increases $|J_3|$ and has size 8 at most, a contradiction because Main has terminated.

Note that after the first operation that redistributed the potential, the potential of $B_2(u)$ has at least $-3c + 1$, in the second operation this potential increased by $2c - 1 + a$, the potential of $A_2(u)$ after the second operation is at least c and the potential of $B_3(u)$ is at least $-b \geq -a$. Therefore the potentials of $A_2(u) \cup B_2(u) \cup B_3(u)$ is nonnegative.

❑

Observe that once we cannot apply rules 1-4 anymore, $B^2_- = \varnothing$. Below, for $v \in A_-$ let $B^3(u)$ be the set of neighbors of v in B_3 that took a unit of the potential from v, and have not returned it in while we have applied one of the rules.
Rule 5: $v \in A_-$, $|B^3(v)| \leq 2$, We can use the definitions, actions and reasoning as for Rule 4, except that $A_2(v) = \{v\}$.
Rule 6: $v \in A_-$, $u \in B^3(v) \cap B_{3,1}$ and $N(u, A - A_-) \neq \varnothing$; undo the second operation for $\{v, u\}$, *i.e.* add 1 to v and subtract 1 from u, then remove u from B_{3-}.
Rule 7: $v \in u, A_-$, $u \in B^3(v) \cap B_{3,1}$ and $N(u, A - A_-) = \varnothing$; We can use identical definitions, actions and reasoning as for Rule 4.

Lemma 13 *Once we cannot apply any of the Rules 1-7, $A_- = \emptyset$.*

Proof. The only configuration with negative potential that is neither eliminated by Rules 1-7 nor allowed be the claim is $v \in A_-$, $|B^3(v)| = 3$ and the potential of $u = n^{-1}(v)$ is -2. This implies that (v, u) is a dangerous pair for J, and thus our algorithm would perform a safe flip. Importantly, this safe flip decreases neither $|J_2|$ nor $|J_3|$, because u has exactly one neighbor in A. By the property 5 of abstract overlap graphs safe flips can liquidate the existence of dangerous pairs.

❑

4.6 Postprocessing algorithm

We can add the following two rules to the third potential redistribution:
Rule 8: $u \in B_3^1$ has negative potential, $N(u, A) = \{v\}$ and $v \in A_3$; take one unit of the potential from v and give it to u.

An application of Rule 8 results in u having potential 0. This rule is valid if the potential of v does not drop below 0. This is not possible, because after the second potential redistribution v has potential 2, and if would subtract from this potential twice, then v would have two neighbors in B_3^1, say u_0 and u_1, and $\{u_0, u_1\}$ would form an improvement.
Rule 9: $v \in A_2^2$ has potential 1 and at most one $u \in N(v, B_3)$ has potential -1; move 1 unit of potential from v to u.

Now the only nodes with negative potential form a set $B' \subset B_3$ each $u \in B'$ has a neighbor v with potential at least 1, let A' be the set of these neighbors. Because of Rule 10, a node $v \in A'$ has at least two neighbors in B', as a result, we can partition B' into a set of A'-butterflies.

We fist show that the following rule is valid:

> **Butterfly selection rule.** If $u \in J_2$, K is a butterfly with center u, $|K| \geq 3$ and $N(K, J) = \{u\}$, then change J into $J - \{u\} \cup K$.

One can see that the Butterfly selection rule is valid. If $|K| = 4$, we gain 24 potential units, and our butterfly K touches the "correct" butterflies at most 20 times, thus we can add 1 potential unit to each $u \in N(K, B')$. If $|K| = 3$, we gain 16 units and we need to add 1 potential unit to at most 15 elements of "correct" butterflies.

After applying Butterfly selection rule, we are left with 2-element J_2-butterflies only. Each such butterfly, together with its center, has potential -1, and selecting a 2-element butterfly yields 8 potential units. By Property 6, a butterfly can touch at most 10 other butterflies. However, we need to find an independent set of butterflies of size at most $1/8$ times the optimum. We do it as follows.

Form a graph of all J_2-butterflies, where $\{b, c\}$ is an edge if butterfly b touches butterfly c.
Apply improvements of size 2 to obtain set $\mathcal{B}$ of butterflies.

We can give -2 potential to each butterfly from the optimum solution and 11 units to each butterly found by this algorithm. Then we redistribute the potential according to each edge between the optimal solution $\mathcal{B}^*$ and our solution $\mathcal{B}$, so the potential of our butterflies remains at least 1, and the only butterflies from $\mathcal{B}^*$ that have negative potential have only one neighbor in $\mathcal{B}$, so their potential is -1. It is easy to see that if a butterfly from B touches two butterflies from $\mathcal{B}$, and that would create an improvement of size 2. Thus the overall potential is nonnegative which means that $|\mathcal{B}| \geq |\mathcal{B}^*|/5.5$.

We finish Postprocessing by applying all butterflies from set $\mathcal{B}$. to the solution that resulted from Main and then Butterfly selection rule. We can conclude that

Theorem 4 *There exists a polynomial time algorithm that solves GEIS problem.*

and by combining all the theorems together we get our main result.

Theorem 5 *There exists a polynomial time approximation algorithm for MIN-SBR problem with approximation ratio* $^{11}\!/_{8}$.

References

[ABIR89] N. Amato, M. Blum, S. Irani and R. Rubinfeld, *Reversing Trains: a turn of the century sorting problem*, J. of Algorithms **10**:413-428, 1989.

[BP93] V. Bafna and P. Pevzner, *Genome rearrangements and sorting by reversals*, Proc. of 34th IEEE FOCS, 148-157, 1993; also in SIAM J. on Computing **25**:272-289, 1996.

[BP95a] V. Bafna and P. Pevzner, *Sorting by reversals: Genome rearrangements in plant organelles and evolutionary history*, Mol. Biol. and Evol. **12**:239-246, 1995.

[BP95b] V. Bafna and P. Pevzner, *Sorting by transpositions*, Proc. of 6th ACM-SIAM SODA, 614-623, 1995.

[BF94] P. Berman and M. Fürer, *Approximating maximum independent set in bounded degree graphs*, Proc. of 5th ACM-SIAM SODA, 365-377, 1994.

[BH96] P. Berman and S. Hannenhalli, *Fast Sorting by Reversals*, Proc. of 7th CPM, 168-185, 1996.

[BK99] P. Berman and M. Karpinski, *On some tighter inapproximability results*, Proc. of 26th ICALP, LNCS **1644**:200-209, Springer-Verlag, Berlin, 1999.

[C97] A. Caprara, *Sorting by Reversals is difficult*, Proc. of 1st ACM RECOMB, 75-83, 1997, to appear in SIAM J. of Discr. Math. 2001.

[C99] A. Caprara, *Formulations and hardness of Multiple Sorting by Reversals*, Proc. of 3rd ACM RECOMB, 84-93, 1999.

[Ch98] D. A. Christie, *A 3/2 Approximation algorithm for sorting by reversals*, Proc. of 9th ACM-SIAM SODA, 244-252, 1998.

[CB95] D. Cohen and M. Blum, *On the problem of Sorting Burnt Pancakes*, Discrete Appl. Math. **61**:105-125, 1995.

[GP79] W.H. Gates and C.H. Papadimitriou, *Bounds for sorting by prefix reversals*, Discr. Math. **27**:47-57, 1979.

[HCK95] S. Hannenhalli, C. Chappey, E. Koonin and P. Pevzner, *Genome sequence comparison and scenarios for gene rearrangements: A test case*, Genomics **30**:299-311, 1995.

[HP95] S. Hannenhalli and P. Pevzner, *Transforming cabbage into turnip*, Proc. of 27th ACM STOC 1995, 178-189.

[HP96] S. Hannenhalli and P. Pevzner, *To cut... or not to cut*, Proc. of 7th ACM-SIAM SODA 1996, 304-313.

[HY99] M. M. Halldórsson and K. Yoshikara, *Greedy approximation of independent sets in low degree graphs*, Proc. of 6th ISAAC, 1996.

[KST97] H. Kaplan, R. Shamir and R.E. Tarjan, *Faster and simpler algorithm for sorting signed permutations by reversals*, Proc. of 8th ACM-SIAM SODA, 178-187, 1997.

[KG95] J. Kececioglu and D. Gusfield, *Reconstructing a history of recombinations from a set of sequences*, Proc. of 5th ACM-SIAM SODA, 471-480, 1995.

[KR95] J. Kececioglu and R. Ravi, *On mice and men: evolutionary distances between genomes under translocation*, Proc. of 6th ACM-SIAM SODA, 604-613, 1995.

[KS93] J. Kececioglu and D. Sankoff, *Exact and approximation algorithms for the inversion distance between two permutations*, Algorithmica **13**:180-210, 1995.

[KS94] J. Kececioglu and D. Sankoff, *Efficient bounds for oriented chromosome inversion distance*, CMP'94, LNCS **807**:307-325, Springer-Verlag, Berlin, 1994.

[PW95] P. Pevzner and M. S. Waterman, *Open combinatorial problems in computational molecular biology* Proc. of 3rd Israel Symp. on Theory of Computing and Systems, 158-163, IEEE Computer Society Press, 1995.

[SCA90] D. Sankoff, R. Cedergen and Y. Abel, *Genomic divergence through gene rearrangement*, in *Molecular Evolution: Computer Analysis of Protein and Nucleic Acid Sequences*, chapter 26, 428-238, Academic Press, 1990.

[SLA92] D. Sankoff, G. Leduc, N. Antoine, B. Paquin, B. F. Lang and R. Cedergen, *Gene order comparisons for phylogenetic inference: Evolution of the mitochondrial genome*, Proc. Natl. Acad. Sci. USA, **89**:6575-6579, 1992.