Independent set reconfiguration: general and RNA-focused parameterized algorithms

Théo Boury, Laurent Bulteau, Bertrand Marchand, Yann Ponty

Independent set reconfiguration: general and RNA-focused parameterized algorithms

Théo Boury[1,3], Laurent Bulteau[2], Bertrand Marchand[1,2] and Yann Ponty[1]

[1]LIX, Ecole Polytechnique, Palaiseau,France.
[2]LIGM, Université Gustave Eiffel, Marne-la-vallée,France.
[3]ENS Lyon, Lyon, France.

Contributing authors: theo.boury@ens-lyon.fr; laurent.bulteau@u-pem.fr; bertrand.marchand@lix.polytechnique.fr; yann.ponty@lix.polytechnique.fr;

Abstract

In this paper, we study the Independent Set (IS) reconfiguration problem in graphs, and its applications to RNA kinetics. An IS reconfiguration is a scenario transforming an IS $\boldsymbol{L}$ into another IS $\boldsymbol{R}$, inserting/removing one vertex at a time while keeping the cardinalities of intermediate sets as large as possible. We focus on the *bipartite* variant where only start and end vertices are allowed in intermediate ISs. Our motivation is an application to the *RNA energy barrier*, a classic hard problem from bioinformatics, which asks, given two RNA structures given as input, whether there exists a reconfiguration pathway connecting them and staying below an energy threshold. A natural parameter for this problem would be the difference between the initial IS size and the threshold (*barrier*). We first show the para-NP hardness of the problem with respect to this parameter. We then investigate two new parameters, the *cardinality range* $\boldsymbol{\rho}$ and a measure of *arboricity* $\boldsymbol{\Phi}$. $\boldsymbol{\rho}$ denotes the maximum allowed size difference between an IS along the reconfiguration and a maximum IS, while $\boldsymbol{\Phi}$ is a measure of the amount of "branching" in the two input RNA structures. We show that bipartite IS reconfiguration is XP for $\boldsymbol{\rho}$ in the general case, and XP for $\boldsymbol{\Phi}$ in the sub-case of bipartite graphs stemming from RNA instances. We give two different routes yielding XP algorithms for $\boldsymbol{\rho}$: The first is a direct $\boldsymbol{O(n^2)}$-space, $\boldsymbol{O(n^{2\rho+2.5})}$-time algorithm based on a separation lemma; The second builds on a parameterized equivalence

with the directed pathwidth problem, leading to a $O(n^{\rho+1})$-space, $O(n^{\rho+2})$-time algorithm for the reconfiguration problem through an adaptation of a prior result by Tamaki [1]. This equivalence is an interesting result in its own right, connecting a reconfiguration problem (which is essentially a *connectivity* problem within a *reconfiguration network*) with a *structural* parameter for an auxiliary graph. For Φ, our $O(n^{\Phi+1})$-algorithm stems from seeing the problem as an instance of *minimum cumulative-cost scheduling*, and relies on a *merging* procedure that might be of independent interest. These results improve upon a partial $O(n^{2\rho+2.5})$-algorithm that only applied to the RNA case. We also demonstrate their practicality of these algorithms through a benchmark on small random RNA instances.

Keywords: reconfiguration problems - parameterized algorithms - RNA bioinformatics - directed pathwidth - minimal cumulative cost scheduling

1 Introduction

Reconfiguration problems. Reconfiguration problems informally ask whether there exists, between two *configurations* of a system, a *reconfiguration pathway* entirely composed of *legal* intermediate configurations, connected by legal *moves*. In a thoroughly studied sub-category of these problems, configurations correspond to *feasible solutions* of some *optimization problem*, and a feasible solution is legal when its quality is higher than a specified threshold.

Examples of optimization problems for which reconfiguration versions have been studied include Dominating Set, Vertex Cover, Shortest Path or Independent Set, which is our focus in this article. Associated complexities range from polynomial (see [2] for examples) to NP-complete (for bipartite independent set reconfiguration [3]), and even **PSPACE**-complete for many of them [3, 4]. Such computational hardness motivates the study of these problems under the lens of *parametrized complexity* [4–7], in the hope of identifying tractable sub-regimes. Typical parameters considered by these studies focus on the value of the *quality threshold* (typically a *solution size* bound) defining legal configurations and the length of the reconfiguration sequences.

Directed pathwidth. *Directed pathwidth*, originally defined in [8] and attributed to Robertson, Seymour and Thomas, represents a natural extension of the notions of pathwidth and path decompositions to directed graphs. Like its undirected restriction, it may alternatively be defined in terms of *graph searching* [9], *path decompositions* [10, 11] or *vertex separation number* [1, 12]. An intuitive formulation can be stated as the search for a visit order of the directed graph, using as few active vertices as possible at each step, and such that no vertex may be deactivated until all its in-neighbors have been activated. Although an FPT algorithm is known for the undirected pathwidth [13], it remains open whether computing the directed pathwidth admits a FPT

algorithm. XP algorithms [1, 12] are known, and have been implemented in practice [14, 15].

RNA energy barrier. RNAs are single-stranded biomolecules, which fold onto themselves into 2D and 3D structures through the pairing of nucleotides along their sequence [16]. Thermodynamics then favors low-energy structures, and the RNA energy barrier problem asks, given two structures, whether there exists a re-folding pathway connecting them that does not go through unlikely high-energy intermediate states [17, 18]. Interestingly, the problem falls under the wide umbrella of reconfiguration problems described above, namely the reconfiguration of solutions of optimization problems (here, energy minimization). An important specificity of the problem is that the probability of a refolding pathway depends on the energy difference between intermediate states and the *starting point* rather than the absolute energy value. Another aspect is that since some pairings of the initial structure may impede the formation of new pairings for the target structure, it induces a notion of *precedence constraints*, and may therefore also be treated as a *scheduling* problem, as carried out in [19, 20].

Problem statement. In our work, we focus on independent set reconfigurations where only vertices from the start or end ISs (L and R) are allowed within intermediate ISs. This amounts to considering the induced subgraph $G[L \cup R]$, bipartite by construction. We write $\alpha(G)$ for the size of a maximum independent set of G (recall that $\alpha(G)$ can be computed in polynomial time on bipartite graphs).

Bipartite Independent Set Reconfiguration (BISR)
Input: Bipartite graph $G = (V, E)$ with partition $V = L \cup R$; integer ρ
Parameter: ρ
Output: True if there exists a sequence $I_0 \cdots I_\ell$ of independent sets of G such that

- $I_0 = L$ and $I_\ell = R$;
- $|I_i| \geq \alpha(G) - \rho, \forall i \in [0, \ell]$;
- $|I_i \triangle I_{i+1}| = 1, \forall i \in [0, \ell - 1]$.

False otherwise.

The vertices of L and R will typically respectively be called "start" and "end" vertices. We further assume that when a bipartite graph G is given, a specification of which side is L (the "start" set) and which is R (the "end" set) is given. When we explicitly need to state which sets are the "start" and "end" sets, we will write $G_{L \to R}$.

Figure 1 shows an example of an instance of BISR and a possible reconfiguration pathway. We introduce the *cardinality range* (or simply *range*) $\rho = \max_{1 \leq i \leq \ell} \alpha(G) - |I_i|$ as a natural parameter for this problem, since it measures

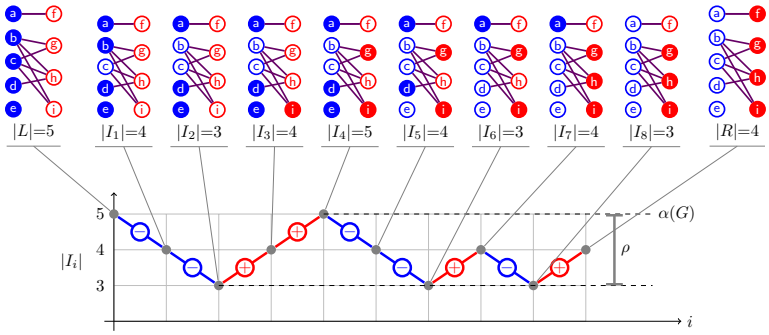


Fig. 1 Example of a bipartite independent set reconfiguration from L (blue) to R (red). Selected vertices at each step have a filled background. All intermediate ISs have size at least 3, and the optimal IS has size 5, so this scenario has a range of 2; it can easily be verified that it is optimal.

a distance to optimality. As mentioned above, another natural parameter for RNA kinetics is the *barrier*, denoted k, and defined as $k = \max_{1 \leq i \leq \ell} |L| - |I_i|$. Intuitively, k measures the size difference from the starting point rather than from an "absolute" optimum. Note that $k = \rho - (\alpha(G) - |L|)$, so one has $0 \leq k \leq \rho$. Both parameters are obviously similar for instances where L is close to being a maximum independent set, which is generally the case in RNA applications, but in theory the range ρ can be arbitrarily larger than the barrier k.

Our results. We first prove that in general, the barrier k may not yield any interesting parameterized algorithm, since BISR is Para-NP-hard for this parameter.

We thus focus on two other parameterizations, the *range* ρ, as defined above and illustrated in Figure 1, and the *arboricity* Φ in the case of RNA instances, as illustrated on Figure 7.

For the range ρ, we prove that BISR is in XP by providing two distinct algorithmic strategies to tackle it. Our first algorithmic strategy stems from a parameterized equivalence we draw between BISR parameterized by ρ and the problem of computing the directed pathwidth of directed graphs. Within this equivalence, ρ maps exactly to the directed pathwidth. This allows to apply XP algorithms for Directed Pathwidth to BISR while retaining their complexity, such as the $O(n^{\rho+2})$-time, $O(n^{\rho+1})$-space algorithm from Tamaki [1] (with $n = |V|$). This equivalence between directed pathwidth and bipartite independent set reconfiguration is itself an interesting result, as it connects a *structural* problem, whose parameterized complexity is open, with a reconfiguration problem of the kind that is routinely studied in parameterized complexity [4–7].

The other algorithmic strategy for BISR parameterized by ρ is more direct, and runs with a time complexity of $O(n^{2\rho}\sqrt{n}m)$ ($m = |E|$) but using only $O(n^2)$ space. It relies on a separation lemma involving, if it exists, a *mixed*

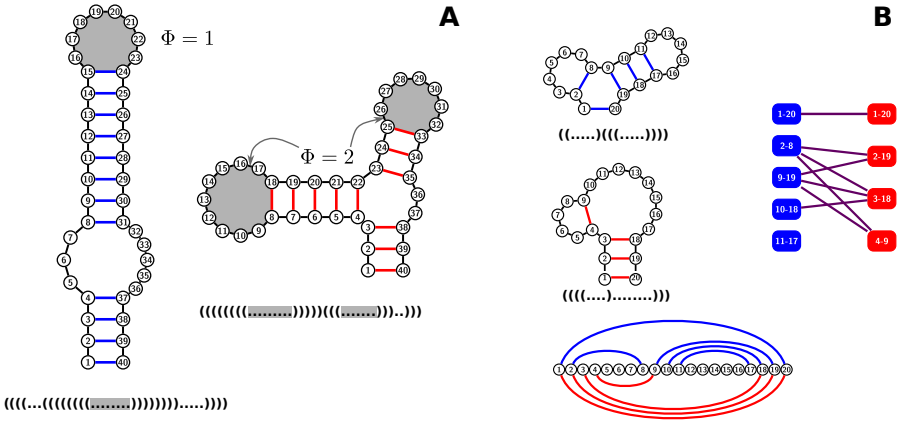


Fig. 2 (A) Example of two RNA structures, and the corresponding value for the arboricity Φ. Within this work, we consider only "conflict-free" secondary structures that can be seen as well-parenthesized strings. Φ is then the number of "terminal" pairs of matching parentheses, highlighted by gray shading. (B) Example of a conflict bipartite graph associated to two input secondary structures. The arcs (base-pairs) of both structures are the vertices of the graph, and two vertices are in conflict if the corresponding arcs are not nested in one another. By "RNA instances of BISR", we mean instances of BISR in which the bipartite graph is such a conflict graph of two RNA structures.

maximum independent set of G containing at least one vertex from both parts of the graph. In the specific case of bipartite graphs arising from RNA reconfiguration, we improve the run-time of the subroutine computing a mixed MIS to $O(n^2)$ (rather than $O(\sqrt{nm})$), with a dynamic programming approach.

As for the arboricity Φ, we also show membership in XP of BISR restricted to RNA instances, through dynamic programming over the sub-trees of $T(S)$ for one of the input structures S (see Figure 7).

We present benchmark results for all algorithms, on random instances of general bipartite graphs as well as instances of the RNA Energy Barrier problem. The approach based on directed pathwidth yields reasonable solving times for RNA strings of length up to ~ 150 nucleotides.

Techniques. The equivalence between BISR parameterized by ρ and directed pathwidth is obtained by defining a directed graph from a *maximum matching* of the input bipartite graph. Our more "direct" algorithm for the ρ parameterization is an example of the "bounded search-tree" technique [21], enabled by a *separation lemma*.

As for the arboricity parameterization, the XP algorithm we present involves seeing the problem in terms of *cumulative cost-optimal scheduling* [22]. In particular, we develop a *merge* procedure for combining optimal solutions on disjoint graphs into an optimal solution for the union of the graphs. We believe this merge procedure and its associated concepts (canonical solutions, preferability criteria) can be of independent interest.

Outline. To start with, Section 2 presents some previously known results related to BISR, and some notations and definitions we will use throughout the article. Then, Section 3 shows that BISR is equivalent to the computation of *directed pathwidth* in directed graphs. Section 4 presents the separation lemma and merge procedure on which our direct XP algorithm in ρ and our XP algorithm in Φ are based. The related concepts of canonical schedule and preferability between schedules are also introduced in this section. Section 5 and Section 6 build on the technical results of Section 4 to present our direct XP algorithm parameterized by ρ and our algorithm XP in Φ in the RNA case. To finish, Section 7 explains some optimizations specific to RNA reconfiguration instances, and presents our numerical results.

2 Preliminaries

2.1 State of the art

Computational hardness. Bipartite Independent Set Reconfiguration was proven NP-complete in [3], through the equivalent k-Vertex Cover Reconfiguration problem. Formulated in terms of RNAs, and restricted to secondary structures (i.e. the subset of bipartite graphs that can be obtained in RNA reconfiguration instances), it was independently proven NP-hard in [17]. To the authors' knowledge, its parameterized complexity remains open.

Independent set reconfiguration in an unrestricted setting (allowing vertices which are outside from the start or end independent sets, i.e. in possibly non-bipartite graphs) when parameterized by the minimum allowed size of intermediate sets has been proven W[1]-hard [4, 5], and fixed-parameter tractable for planar graphs or graphs of bounded degree [6]. Whether this more general problem is in XP for this parameter remains open. We note that in this setting, parameter ρ seems slightly less relevant since it involves computing a maximal independent set in a general graph (i.e. testing if there exists a reconfiguration from $\emptyset$ to $\emptyset$ with range ρ is equivalent to deciding if $\alpha(G) \geq \rho$).

Heuristics. Given the great practical importance of the RNA Energy barrier problem in Bioinformatics (BISR in the RNA case), several heuristics [23, 24] have been developped for it. In this paper, we assess the potential of parameterized algorithmics for the development of efficient *exact* algorithms for the problem, starting with a simple energy model (corresponding to Bipartite Independent Set Reconfiguration).

Exact algorithms. As for algorithms for BISR, the closest precedent is an algorithm by Thachuk et al. [18]. It is restricted to RNA secondary structure conflict graphs, and additionally to conflict graphs for which both parts L and R are *maximum independent sets* of G. In this restricted setting, although it is not stated as such, [18] provides an XP algorithm with respect to the barrier parameter k which then coincides with the range parameter ρ that we introduce. In this paper, we extend this line of study by showing the

Para-NP-hardness of BISR for k in the general setting. We further show that generalizing k into ρ allows to retain membership in XP.

Recent unpublished work [19, 20] describe polynomial-time algorithm for restricted input versions of BISR. More precisely, within RNA instances of BISR, they tackle the $\Phi = 1$ case, called "bipartite permutation graphs" in [20] and "convex bipartite partial orders" in [19]. The specialization of XP algorithm we describe for Φ yields a $O(n^3)$ algorithm in that case, improving over the $O(n^6 \log n)$ of [20].

2.2 Preliminary results

Restriction to the monotonous case. A reconfiguration pathway for BIPARTITE INDEPENDENT SET RECONFIGURATION is called *monotonous* or *direct* if every vertex is added or removed exactly once in the entire sequence. The length of a monotonous sequence is therefore necessarily: $\ell = |L \cup R| = |L| + |R|$. Theorem 2 from [3] tells us that if G, ρ is a yes-instance of bipartite independent set reconfiguration, then there exists a *monotonous* reconfiguration between L and R respecting the constraints. We will therefore restrict without loss of generality our study to this simpler case. In the more restricted set studied in [18], this was also independently shown.

Hardness for the barrier parameter. In the general case where L is not necessarily a maximal independent set, the range and barrier parameters (respectively ρ and $k = \rho - (\alpha(G) - |L|)$ may be arbitrarily different. The following result motivates our use of parameter ρ for the parameterized analysis of BISR.

Proposition 1 BISR *is* Para-NP*-hard for the energy barrier parameter k (i.e.* NP*-hard even for a constant value of k, here with $k = 0$).*

Proof We use additional vertices in R to prove this result. Informally, such a vertex may always be inserted first in a realization: it improves the starting IS from $|L|$ to $|L| + 1$, so the lower bound on the rest of the sequence is shifted from $|L| - k$ to $|L| - (k - 1)$, effectively reducing the barrier without simplifying the instance. Thus, we build a reduction from the general version of BISR: given a bipartite graph G with parts L and R and an integer ρ, we construct a new instance G' with parts $L' = L$ and R' equal to $R \cup N_R$ and $\rho' = \rho$. N_R is composed of $|L| - (\alpha(G) - \rho)$ isolated vertices (we can assume without loss of generality that this quantity is non-negative, otherwise (G, ρ) is a trivial no-instance), completely disconnected from the rest of the graph.

Note that $\alpha(G') = \alpha(G) + |N_R| = |L| + \rho$, so the barrier in (G', ρ') is $k = \rho - (\alpha(G') - |L|) = 0$. A realization for (G, ρ) can be transformed into a realization for (G', ρ) by inserting vertices from N_R first, and conversely, vertices from N_R can be ignored in a realization for (G', ρ) to obtain a realization for (G, ρ). Therefore, since BISR is NP-Complete, it is also Para-NP-hard w.r.t the barrier k. □

2.3 Definitions

The following definitions and notations will be used throughout the paper. They allow to link, at an intermediary step along a reconfiguration, the set of processed vertices to the current independent set and its size.

Licit subsets. Given a subset X of vertices, we define $I(X) = (L \setminus X) \cup (R \cap X) = L \Delta X$. We say that X is *licit* if $I(X)$ is an independent set. Intuitively, in a bipartite graph G with sides L and R, $I(X)$ is the independent set obtained after processing the vertices of X, starting from L ($L \cap X$ removed, $R \cap X$ added).

Permutation formulation. An equivalent representation of a monotonous reconfiguration pathway $I_0 \dots I_\ell$ from L to R for a graph G is a *permutation* S of $L \cup R$. We will also use the term of *schedule*. The i-th vertex of the permutation is the vertex that is *processed* (i.e. added or removed) between I_{i-1} and I_i (this formulation lightens the representation of a solution, from a list of vertex sets to a list of vertices). We write $P \sqsubseteq S$ if P is a prefix of S, and $V(P)$ (or simply P if the context is clear) for the set of vertices appearing in P. A permutation S is licit if $V(P)$ is licit for each prefix P of S; note that S is licit if and only if $\forall r \in R$, the neighborhood $N(r)$ of r in G appears before r in S.

Balance δ and ρ-realizations. Given a subset X of vertices, we write $\delta(X) = |L \cap X| - |R \cap X|$. With this quantity, $|I(X)| = |L| - \delta(X)$. $\delta(X)$ is called the *balance* of X, as it corresponds to the size difference between the initial IS L and the current IS $I(X)$. Then, S is a permutation (or *schedule*) of barrier k if S is licit and for each prefix $P \sqsubseteq S$, $\delta(V(S)) \leq k$. Equivalently, S is a *ρ-realization* if S is licit and such that for each prefix $P \subseteq S$, $|I(P)| \geq \alpha(G) - \rho$ (i.e. $\delta(V(P)) \leq \rho + |L| - \alpha(G)$). This is consistent with the fact that $\rho = k + \alpha(G) - |L|$.

Budget. Finally, given a bipartite graph G and a licit permutation S for G, we write $bg(S)$ for the barrier of S, i.e.

$$bg(S) = \max_{P \sqsubseteq S} |I(P)| - |L| = \max_{P \sqsubseteq S} \delta(P)$$

The *budget* of a graph G is the best possible budget of a licit permutation of G. Denoting by $\mathcal{L}(G)$ the set of licit permutations of G:

$$bg(G) = \min_{S \in \mathcal{L}(G)} bg(S)$$

A related quantity is the best possible *range* of a graph G, which we write $\rho(G)$. It verifies $\rho(G) = bg(G) + \alpha(G) - |L|$. Note that with these definitions, the BISR problem can be equivalently defined as deciding, given a graph G and an upper-bound ρ, whether $\rho(G) \leq \rho$.

3 Connection with Directed Pathwidth

We first present a parameterized reduction from bipartite independent set reconfiguration to an input-restricted version, on graphs allowing for a perfect matching. Then, this version of the problem is shown to be simply equivalent to the computation of directed pathwidth on general directed graphs.

3.1 Definitions

Parameterized reduction. In this section, we provide a definition of directed pathwidth, and then prove its parameterized equivalence to the bipartite independent set reconfiguration problem. We say two problems $\mathcal{P}_1$ and $\mathcal{P}_2$ are parametrically equivalent when there exists both a *parameterized reduction* from $\mathcal{P}_1$ to $\mathcal{P}_2$ and another from $\mathcal{P}_2$ to $\mathcal{P}_1$. A sufficient condition to obtain a parameterized reduction [21] from problem $\mathcal{P}$ to problem $\mathcal{Q}$ is to have a function φ from instances of $\mathcal{P}$ to instances of $\mathcal{Q}$ such that (i) $\varphi(x)$ is a yes-instance of $\mathcal{Q} \Leftrightarrow x$ is a yes-instance of $\mathcal{P}$, (ii) φ can be computed in polynomial time (iii) the parameter of x and the parameter of $\varphi(x)$ are equal.

Interval representation. Our definition of directed pathwidth relies on interval embeddings. Alternative definitions can be found, for instance in terms of directed path decomposition or directed vertex separation number [1, 9, 12].

Definition 1 (Interval representation) An *interval representation* of a directed graph H associates each vertex $u \in H$ with an interval $I_u = [a_u, b_u]$, with a_u, b_u integers. An interval representation is *valid* when $(u, v) \in E \Rightarrow a_u \leq b_v$. I.e, the interval of u must start before the interval of v ends. If m, M are such that $\forall u, m \leq a_u, b_u \leq M$, we define the *width* of an interval representation as $\max_{m \leq i \leq M} |\{u | i \in I_u\}| - 1$

Definition 2 (directed pathwidth) The *directed pathwidth* of a directed graph H is the minimum possible width of a valid interval representation of H. We note this number $dpw(H)$.

Nice interval representation. An interval representation is said to be *nice* when no more than one interval bound is associated to any given integer, and the integers associated to interval bounds are exactly $[1 \dots 2 \cdot |V(H)|]$. Any interval representation may be turned into a nice one without changing the width by introducing new positions and "spreading events". See Appendix B.1 for more details.

Directed graph from perfect matching. Given a bipartite graph G allowing for a perfect matching M, we construct an associated directed graph H in the following way: the vertices of H are the edges of the matching, and $(l, r) \to (l', r')$ is an arc of H iff $(l, r') \in G$. Alternatively, H is obtained from G, M by orienting the edges of G from L to R, and then contracting the edges of M. We will denote this graph $H(G, M)$, and simply call it the *directed*

graph associated to G, M. Such a construction is relatively standard and can be found in [25, 26], for instance.

3.2 Directed pathwidth ⇔ Bipartite independent set reconfiguration

Perfect matching case. Our main structural result regarding directed pathwidth is the following. Its proof relies on interval representations, with the intuition that the number of open intervals at a given position is the number of dependencies that have been lifted, but not compensated for yet.

Proposition 2 *Let G be a bipartite graph allowing for a perfect matching M. Then G allows for a ρ-realization iff $dpw(H(G,M)) \leq \rho$.*
Conversely, given any directed graph H, there exists a bipartite graph G allowing for a perfect matching M such that $H = H(G,M)$ and G allows for a ρ-realization iff $dpw(H) \leq \rho$.

Proof We start with the first statement, the equivalence between $dpw(H(G,M)) \leq \rho$ and the existence of a ρ-realization for G. First note that, since G allows for a perfect matching, we have $|L| = |R|$, and by König's theorem, if K is a minimum vertex cover of G, $|K| = |L| = |R|$. Since $\alpha(G) = |L| + |R| - |K|$ we have $\alpha(G) = |L| = |R|$. I.e. L and R are maximum independent sets of G.

⇒ If G allows for a ρ-realization, then $\exists P$ ordering of the vertices of G such that every prefix X_i of P verifies $|I(X_i)| = |L| - \delta(X_i) = \alpha(G) - \delta(X_i) \geq \alpha(G) - \rho$. Therefore $\delta(X_i) = |X_i \cap L| - |X_i \cap R| \leq \rho$.
Consider a vertex (l,r) of $H(G,M)$, with (l,r) an edge of M. We associate to (l,r) the interval $[a_{(l,r)}, b_{(l,r)}]$ where $a_{(l,r)}$ is such that $P[a_{(l,r)}] = l$. i.e, it corresponds to the step in the reconfiguration where l is removed. Likewise, $b_{(l,r)}$ is such that $P[b_{(l,r)}] = r$.
For any edge $(l,r) \to (l',r')$ of H, necessarily $(l,r') \in G$, which implies that in the reconfiguration sequence, l has to be removed before r' is added. l appears therefore earlier than l in P, and $a_{(l,r)} \leq b_{(l',r')}$. The intervals we have defined therefore form a valid interval representation of H.
In addition, the intervals intersecting a given position i correspond to pairs (l,r) where, at step i, l has already been removed while r is yet to be added. Since the decrease in independent set size incurred by the removal of l is compensated by the addition of its match r, the number of intervals intersecting position i is exactly $\delta(X_i)$, the imbalance of the i-prefix of P, which by hypothesis is $\leq \rho$.

⇐ Suppose the directed graph $H(G,M)$ associated to G, M has directed pathwidth $\leq \rho$. Consider an optimal nice interval representation for H.
In this representation, a vertex (l,r) of H is associated to an interval $[a_{(l,r)}, b_{(l,r)}]$. Thanks to the structure of *nice* interval representation, we

simply define a permutation P of $L \cup R$ with, $\forall (l, r)$ $P[a_{(l,r)}] = l$ and $P[b_{(l,r)}] = r$.
If (l, r') is an edge of G, with r the match of l and l' the match of r', then the construction above ensures that l is before r' in P. For two matched vertices, this is also immediate. Then, as for two matched vertices l, r, the removal of l is compensated by the addition of r, for any prefix X_i of P, the imbalance $\delta(X_i)$ is exactly the number of intervals intersecting position i. By assumption, we therefore have $\delta(X_i) \leq \rho$ and P is a ρ-realization.

For the second part of the statement, given a directed graph H, we construct a bipartite graph G with sides L, R allowing for a perfect matching M in the following way: for each vertex $u \in H$ we introduce two vertices (l_u, r_u) in G. We assign l_u to L and r_u to R, connect l_u and r_u and add the edge to the matching M. We now add an edge from l_u to r_v in G for any $(u, v) \in E(H)$. G now verifies $H = H(G, M)$, and by the result above, $dpw(H) \leq \rho$ iff G allows for a ρ-realization.

□

The first half of Proposition 2 is a parameterized reduction from an input-restricted version of BIPARTITE INDEPENDENT SET RECONFIGURATION to directed pathwidth. The restriction is on bipartite graphs allowing for a perfect matching. The second half is a parameterized reduction in the other direction. In both cases, the parameter value is directly transferred, which allows to retain the same complexity when transferring an algorithm from one problem to the other.

Non-perfect-matching case. In the case where G does not allow for a perfect matching, we construct G' allowing for a perfect matching M', and such that $\rho(G) = \rho(G') = dpw(H(G', M'))$. G' is obtained from G through the addition of new vertices. Specifically, with a bipartite graph G with sides L, R, a maximum matching M of G, and the set U of unmatched vertices in G, we extend G with $|U|$ new vertices in two sets N_L, N_R, giving a new graph G', with sides $L' = L \cup N_L, R' = R \cup N_R$, in the following way ($M'$ is initialized to M):

- For each $u \in L \cap U$, we introduce a new vertex $r(u) \in N_R$, connect it to *all* vertices of L', and add the edge $(u, r(u))$ to M'.
- Likewise, for each $v \in R \cap U$, we introduce $l(v) \in N_L$, connect it to all vertices of R' and add $(v, l(v))$ to M'.

Note that M' is a perfect matching of the extended bipartite graph G'.

Proposition 3 *With G, G' defined as above, we have that G allows for a ρ-realization iff G' allows for a ρ-realization.*

Proof First note that by König's Theorem, $\alpha(G') = |M'| = |M| + |U| = \alpha(G)$, so it suffices to ensure that any realization for G can be transformed into a realization for G' where independent sets are lower-bounded by the same value, and vice versa.

Let P be any ρ-realization of G, then $P' = N_L \cdot P \cdot N_R$ is a ρ-realization for G', with N_L and N_R laid out in any order. Indeed, P' satisfies the precedence constraint, and any intermediate set I in P' satisfies one of the following cases: $L \subseteq I$, $R \subseteq I$, or I is an intermediate set from P, so in any case it has size at least $\alpha(G)-\rho = \alpha(G')-\rho$.

Conversely, because of the all-to-all connectivity between N_L and R and between L and N_R, a realization for G' needs to have N_L before any vertex from R, and have N_R after all vertices from L. Without loss of generality, it is therefore of the form $N_L \cdot P \cdot N_R$ with P a realization of G, and G allows for a ρ-realization. □

The construction above in fact yields a parameterized reduction from Bipartite Independent Set Reconfiguration to its input-restricted version on bipartite graphs allowing for a perfect matching. This input-restricted version is in turn parametrically equivalent to directed pathwidth by Proposition 2. Hence the following corollary:

Corollary 1 Bipartite Independent Set Reconfiguration *is parametrically equivalent to* Directed Pathwidth

It allows to import algorithmic results for Directed Pathwidth and apply them to Bipartite Independent Set Reconfiguration. In particular:

Corollary 2 *There exists a $O(n^{\rho+1})$-space, $O(n^{\rho+2})$-time* XP *algorithm for* Bipartite Independent Set Reconfiguration.

Proof Application of the algorithm from [27]. See also Section 7.1 for more details. □

An implementation and a benchmark of this algorithm in the context of RNA kinetics is presented in Section 7.

Limitations. The high space-complexity of [27] may hinder the practicality of the algorithm, and the ρ parameterization may not necessarily be adapted to the RNA kinatics context. This is why we also explored direct algorithms parameterized by ρ for the problem, presented in Section 5, and another parameterization (arboricity) in Section 6. They rely on the technical elements presented in the following section, regarding the role of *mixed* maximum independent sets (i.e. intersecting both L and R) in G as *separators* and the problem of optimally *merging* optimal solutions for disconnected instances.

4 Lemmata: algorithmic building blocks

This section introduces our main technical results, which are the building blocks of the algorithms we propose for Bipartite Independent Set Reconfiguration. They consist of a *separation lemma* and a *merge* procedure.

4.1 Definitions

We use the permutation representation of reconfiguration scenarios, i.e. licit permutations of vertices. Note that the intersection, as well as the union, of two licit set of vertices are licit:

Property 1 *Given X, Y two licit subsets of a graph G, both $X \cap Y$ and $X \cup Y$ are licit subsets.*

Proof Let us check that $I(X \cap Y) = L \setminus (X \cap Y) \cup (R \cap X \cap Y)$ is indeed an independent set.

Consider $r \in R$ and suppose $r \in I(X \cap Y)$. Then $r \in R \cap X \cap Y$, and since X is licit, $r \in X \cap R$ implies $N(r) \cap I(X) = \emptyset$ and therefore $N(r) \subset L \cap X$. Likewise, $N(r) \subset L \cap Y$. Therefore $N(r) \subset L \cap X \cap Y$ and $I(X \cap Y)$ does not contain $N(r)$.

Likewise, consider $\ell \in L$, and suppose $\ell \in I(X \cap Y)$. Then $\ell \in L \setminus (X \cap Y)$, so either $\ell \in L \setminus X$ or $\ell \in L \setminus Y$. Since X and Y are licit, either $N(r) \cap (R \cap X) = \emptyset$ or $N(r) \cap (R \cap Y) = set$. In any case, $N(\ell) \cap I(X \cap Y) = \emptyset$ and $I(X \cap Y)$ is indeed an independent set.

Let us now check that $I(X \cup Y)$ is also an independent set. In a similar fashion, consider $r \in I(X \cup Y) \cap R$. $r \in R \cap (X \cup Y)$ implies $r \in R \cap X$ or $r \in R \cap Y$, which implies since X and Y are licit $N(r) \cap (L \setminus X) = \emptyset$ or $N(r) \cap (L \setminus Y) = \emptyset$, i.e. $N(r) \subseteq X$ or $N(r) \subseteq Y$. In any case, $N(r) \cap (L \setminus (X \cup Y)) = \emptyset$.

To finish, consider $\ell \in I(X \cup Y) \cap L$. We have $\ell \in L \setminus (X \cup Y)$, so $\ell \notin X$ and $\ell \notin Y$. Since X and Y are licit, $N(\ell) \cap X = \emptyset$ and $N(\ell) \cap Y = \emptyset$, so $I(X \cup Y) \cap N(\ell) = R \cap (X \cup Y) \cap N(\ell) = \emptyset$, and $I(X \cup Y)$ is indeed and independent set.

□

Permutation sub-sampling. Given a realization P of G and a set of vertices X, we write $P \cap X$ for the sub-sequence of P consisting of the vertices of X, without changing the order. Likewise, $P \setminus X$ denotes the sub-sequence of P consisting of vertices *not* in X.

Definitions: separators. A *mixed maximum independent set* I of G is an independent set of G of maximum cardinality containing at least a vertex from both parts. Note that not every bipartite graph contains such a set. A *separator* X is a subset of $L \cup R$ such that $I(X)$ is a mixed maximum independent set of G.

Separators and inversions. When otherwise specified, a bipartite graph G has sides L and R, and the independent set reconfiguration goes from L to R. In the proofs below however, it will be useful to also consider the instance of BISR in which R is reconfigured into L. To differentiate both, given a graph G, we write $G_{L \to R}$ (resp. $G_{R \to L}$) or the instance of BISR in which L is reconfigured into R (resp. R into L). Likewise, we write $I_{L \to R}(X) = (L \setminus X) \cup (R \cap X)$ to denote the independent set obtained by processing X starting from L, while $I_{R \to L} = (R \setminus X) \cup (L \cap X)$ is the result of processing X starting from R. Interestingly, a separator for $G_{L \to R}$ is then also a separator for $G_{R \to L}$.

Property 2 *Let X be a separator of $G_{L\to R}$. Then $Y = G \setminus X$ is a separator of $G_{R\to L}$*

Proof $I_{R\to L}(Y) = (R\backslash Y)\cup(L\cap Y) = (R\backslash(G\backslash X)\cup(L\cap(G\backslash X))) = (R\cap X)\cup(L\backslash X) = I_{L\to R}(X)$, which is a mixed maximum independent set of $G_{L\to R}$ (same graph as $G_{R\to L}$) □

The two technical results presented in this section, Lemma 4.2 (separation Lemma) and Theorem 4 (merging procedure) are expressed in terms of a notion of *preferability* and *canonical schedules*. The preferability order relation allows to choose between different schedules equivalent in terms of barrier, while a canonical schedule is the "most preferable". These two notions are defined in the following paragraphs.

Notations and definitions: canonicity and preferability. Formally, given a schedule S for a graph G, and $-1 \geq -i \geq |L| - \alpha(G)$, we define $\ell_S(-i)$ as the smallest strictly positive integer, if it exists, such that $\delta(S_{\leq \ell_S(i)}) = -i$. Likewise, for i such that $-1 \geq -i \geq |R| - \alpha(G)$ we define $r_S(-i)$ as the largest integer $< |S|$, if it exists, such that $\delta(S_{\leq r_S(-i)}) + |R| - |L| = i$.

We then write $\text{pref}_{-i}(S) = bg(S_{\leq \ell_S(-i)})$ and $\text{suff}_{-i}(S) = bg(S_{\geq r_S(-i)}) - \delta(r_S(-i))$. The corresponding prefixes and suffixes are denoted by $\text{Pref}_{-i}(S)$ and $\text{Suff}_{-i}(S)$. If $\ell_S(-i)$ does not exist, then $\text{pref}_{-i}(S) = +\infty$, and likewise for $r_S(-i)$ and $\text{suff}_{-i}(S)$. Informally, these quantities are "the budget needed to reach level $-i$" in the forward and reverse directions.

These definitions are illustrated in Figure 3. Note that upon inverting the start set L and final set R, then $\ell_S(-i), \text{Pref}_{-i}(S)$ and $\text{pref}_{-i}(S)$ become $r_S(-i), \text{Suff}_{-i}(S), \text{suff}_{-i}(S)$ and vice-versa. To be more precise, given S a schedule, i.e. an order on vertices, let us denote by $\overleftarrow{S}$ its reverse schedule, with opposite order. Then, we have:

$$\text{Pref}_{-i}(S) = \overleftarrow{\text{Suff}_{-i}(\overleftarrow{S})}$$

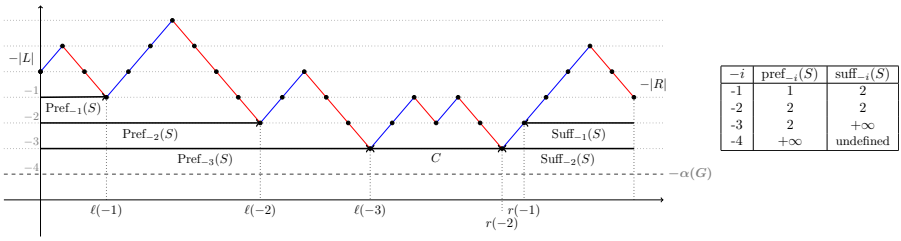
and

$$\text{pref}_{-i}(S) = \text{suff}_{-i}(\overleftarrow{S})$$

Based on these quantities, we define a partial order on schedules for G:

Definition 3 (preferability) Given S and S' two schedules for a bipartite graph G, we say that S is *preferable* to S' (denoted $S \preccurlyeq S'$) if $bg(S) \leq bg(S')$ and $\forall i$, $\text{pref}_{-i}(S) \leq \text{pref}_{-i}(S')$ and $\text{suff}_{-i}(S) \leq \text{suff}_{-i}(S')$. In the case of an equality for all criteria, $S \preccurlyeq S'$ if $\forall i$, $\ell_S(-i) \leq \ell_{S'}(-i)$ and $r_S(-i) \geq r_{S'}(-i)$.

Finally, we say that S is *strictly preferable* to S' if $S \preccurlyeq S'$ and $S' \not\preccurlyeq S$



$-i$	$\mathrm{pref}_{-i}(S)$	$\mathrm{suff}_{-i}(S)$
-1	1	2
-2	2	2
-3	2	$+\infty$
-4	$+\infty$	undefined

Fig. 3 Illustration of the definition of $\mathrm{pref}_{-i}(S)$ and $\mathrm{suff}_{-i}(S)$

Remark that this relation is neither total nor antisymmetric (there are pairs S, S' with either both $S \preccurlyeq S'$ and $S' \preccurlyeq S'$ or both $S \not\preccurlyeq S'$ and $S' \not\preccurlyeq S'$), but it is easily seen to be transitive from the definition. We are mostly interested in the search of global optimums, as formulated below.

Definition 4 (canonical solution) A schedule S that is preferable to any other schedule S' for G is called a *canonical solution* for G.

4.2 Separation lemma

Lemma 4.2 (below) on which our algorithm XP in ρ is based is proved using the following "modularity" property of the balance functions. Interestingly, it is almost the same property (sub-modularity), on a different quantity (the in-degrees of vertices) on which rely the XP algorithm for directed pathwidth [1].

Lemma 1 (modularity) *Given licit subsets X and Y, we have:*

$$|I(X)| + |I(Y)| = |I(X \cup Y)| + |I(X \cap Y)|$$

and

$$\delta(X \cup Y) + \delta(X \cap Y) = \delta(X) + \delta(Y)$$

Proof We have $I(X) = (L \setminus X) \cup (R \cap X)$. Therefore, $|I(X)| = |L \setminus X| + |R \cap X| = |L| - |L \cap X| + |R \cap X|$. Furthermore, $|(X \cup Y) \cap L| = |(X \cap L) \cup (Y \cap L)| = |X \cap L| + |Y \cap L| - |X \cap Y \cap L|$, and likewise for R. The result stems from a substraction of one equation to the other, and an addition of $|L|$. As for the second part, it comes from the definition $\delta(X) = |L| - |I(X)|$. □

Based on this "modularity", the following separation lemma is shown by "re-shuffling" a solution into another one going through a mixed MIS.

Lemma 2 (separation lemma) *Let X be a separator of G. If S is a schedule for G, then $(S \cap X) \cdot (S \setminus X) \preccurlyeq S$.*

Proof Let us write $S' = (S \cap X) \cdot (S \setminus X)$, and start by showing $bg(S') \leq bg(S)$. Let $\rho' \sqsubseteq S'$. We first introduce the following notation: given a prefix ρ' of S' with $(S \cap X) \sqsubseteq \rho'$, we write $\mathrm{rem}_X(\rho')$ for the smallest prefix ρ of S such that $V(\rho \cup X) = V(\rho')$. This definition is illustrated on Figure 4.

- if $\rho' \sqsubseteq (S \cap X)$, then $\exists \rho \sqsubseteq S$ such that $\rho' = \rho \cap X$, and $\delta(\rho') = \delta(\rho \cap X) = \delta(\rho) + \delta(X) - \delta(\rho \cup X)$ (by the modularity property, Lemma 1). $\delta(X)$ is the smallest possible value for δ, therefore $\delta(X) - \delta(\rho \cup X) \leq 0$ and $\delta(\rho') \leq \delta(\rho) \leq bg(S)$.
- else if $(S \cap X) \sqsubseteq \rho'$, then let $\rho = \mathrm{rem}_X(\rho')$. We have $\delta(\rho') = \delta(\rho \cup X) = \delta(\rho) + \underbrace{\delta(X) - \delta(\rho \cap X)}_{\leq 0} \leq bg(S)$.

Let us now prove that $\forall i \in [1 \dots \alpha(G) - |L|]$, $\mathrm{pref}_{-i}(S') \leq \mathrm{pref}_{-i}(S)$, and in the case of equality $\ell_{S'}(-i) \leq \ell_S(-i)$. Let us first note that, since $\delta(X)$ reaches the minimum possible value for δ over all licit subsets, we have $\forall \sigma \sqsubseteq (S \setminus X)$, $\delta(\sigma) \geq 0$ (otherwise, $\sigma \cup X$ would be a licit subset with $\delta(\sigma \cup X) < \delta(X)$). In addition, remark that since $\delta(X) = |L| - \alpha(G)$, $\mathrm{Pref}_{-i}(S') \sqsubseteq S \cap X$. Given these elements, let $\rho' \sqsubseteq \mathrm{Pref}_{-i}(S')$, and $\rho \sqsubseteq S$ the smallest prefix of S such that $\rho' = \rho \cap X$. We have $\delta(\rho) = \delta(\rho') + \underbrace{\delta(\rho \setminus X)}_{\geq 0}$, so $\delta(\rho') \leq \delta(\rho)$. It simply remains to show that $\rho \sqsubseteq \mathrm{Pref}_{-i}(S)$ to get $\delta(\rho') \leq \mathrm{pref}_{-i}(S)$. Let therefore τ be a strict prefix of ρ. We indeed have $\delta(\tau) = \underbrace{\delta(\tau \cap X)}_{> -i} + \underbrace{\delta(\tau \setminus X)}_{\geq 0} > -i$. Therefore overall $\mathrm{pref}_{-i}(S') = \max_{\rho' \sqsubseteq \mathrm{Pref}_{-i}(S')} \delta(\rho') \leq \mathrm{pref}_{-i}(S)$. In addition, $\delta(\tau) > -i \ \forall \tau \sqsubseteq \rho$ with $\rho \neq \tau$ implies, when $\mathrm{pref}_{-i}(S) < +\infty$, $\ell_S(-i) \geq \ell_{S'}(-i)$.

As for $\mathrm{suff}_{-i}(S')$, we have $\mathrm{suff}_{-i}(S') = \mathrm{pref}_{-i}(\overleftarrow{S'})$. By Property 2, $Y = G \setminus X$ is a separator for $G_{R \to L}$, and $\overleftarrow{S'} = (\overleftarrow{S} \cap Y) \cdot (\overleftarrow{S} \setminus Y)$. Per the arguments above, $\mathrm{pref}_{-i}(\overleftarrow{S'}) \leq \mathrm{pref}_{-i}(\overleftarrow{S})$ and $\ell_{\overleftarrow{S'}}(-i) \leq \ell_{\overleftarrow{S}}(-i)$. Overall, $\mathrm{suff}_{-i}(S') = \mathrm{pref}_{-i}(\overleftarrow{S'}) \leq \mathrm{pref}_{-i}(\overleftarrow{S}) = \mathrm{suff}_{-i}(S)$, and in the case of equality, $r_{S'}(-i) \leq r_S(i)$. □

Therefore, if G allows for a mixed independent set, any optimal schedule can be assumed to go through this independent set. A schedule that reaches IS cardinality $\alpha(G)$ is said to be *simple*. Lemma 2 thus yields the following:

Corollary 3 *Any graph G has a* simple *optimal schedule.*

Corollary 4 *For any schedule S of a bipartite graph G, $\exists S'$* simple *such that $S' \preccurlyeq S$*

Proof Either G does not allow for a mixed MIS, in which case $\alpha(G) = \max(\mathsf{L}, \mathsf{R})$ and any schedule is simple, or G allows for a mixed MIS and we can apply Lemma 4.2 to S. □

This separation result will be used in Section **??**. We now turn to another algorithmic building block, that is a *merge* procedure for combining solutions for disjoint graphs into a global optimal.

4.3 Merge Procedure

Merging problem. Given two independent graphs G_1 and G_2, and two optimal orderings S_1 and S_2, it is natural to ask whether it is always sufficient to simply *interlace* S_1 and S_2 to get an optimal solution G, or if a *rearrangement* of S_1 and/or S_2 may be required.

In this section, we answer this question by showing that not only is rearranging is necessary in some cases, but figuring out the optimal rearrangement is NP-hard (Lemma 3). However, we also show that interlacing is enough when the two input schedules are in *canonical form*, as defined in Definition 4. The merging procedure can then be done in linear time, as shown in Theorem 4.

Related work. A similar merge procedure had already been designed in [22] for the cumulative cost-optimal scheduling problem, of which Bipartite Independent Set Reconfiguration is an instance. However, the corresponding "canonical form", *strictly-optimal schedules*, was not adaptable to algorithmic application, described in Section 6. The unpublished pre-prints [19, 28] also claimed to derive a merge procedure for the same problem. However, the merge ("combine") procedure of [28] relies on an unproven Observation (Observation 5.5 in [28]). Even if this Overvation was correct, the merge procedure presented here achieves a better (linear) complexity. As for [19], it is unclear (in a similar fashion as [22]) how it could adapt to our algorithmic application.

Lemma 3 *Given two bipartite graphs G_1 and G_2, S_1 and S_2 optimal schedules for G_1 and G_2 respectively, and an integer k, the problem of deciding whether $bg(G_1 \cup G_2) \leq k$ is* `NP-hard`.

Proof We prove the NP-hardness by reduction from the barrier problem. Given therefore a bipartite graph G with n vertices and an integer k', we build G_1, G_2, S_1, S_2 and k as follows. G_1 consists of G augmented with a $(n+1, n+1)$-biclique B_1, such that the $n+1$ left vertices of B_1 are dependencies of all the right vertices of G. Additionally, one left vertex b_1 is added as a dependency of all the right vertices of G. G_2 consists of a single biclique with $n+1$ left vertices and $k'+1$ right vertices.

Let S_1 start with B_1, followed by b_1, and a simplistic schedule for G, consisting of all the left vertices of G followed by all the right vertices of G. As B_1 is a biclique, $bg(G_1) \geq n+1$ and S_1 is optimal. As for S_2, the only possible schedule consists of all its left vertices followed by all of its right vertices.

We will now show that $bg(G_1 \cup G_2) \leq n+1$ if and only if $bg(G) \leq k'$. To start with, if G admits a schedule S with $bg(S) \leq k'$, then $B_1 \cdot B_2 \cdot b_1 \cdot S$ is a $n+1$-schedule for $G_1 \cup G_2$ (with B_1, B_2 in any order in which all left vertices are before the right vertices) and $bg(G_1 \cup G_2) \leq n+1$.

In the other direction, if $bg(G_1 \cup G_2) \leq n+1$, then B_1 is necessarily scheduled first, as scheduling any part of B_2 before B_1 would increase the baseline by a strictly positive amount and yield an overall budget $> n+1$. Likewise, scheduling any part of $\{b_1\} \cup G$ before B_2 would yield a barrier $> n+1$. Therefore, an optimal schedule

is necessary of the same form as before $B_1 \cdot B_2 \cdot b_1 \cdot S$ for some schedule S of G. If it has barrier $\leq n+1$, then necessarily $bg(S) \leq k'$, concluding the proof. □

The notion of preferability gives us a simple criteria to choose between different schedules with the same overall budget. This will be exploited algorithmically in our dynamic programming approach to the bipartite independent set reconfiguration problem in the case for bipartite circle graphs, presented in Section 6.

As for the merge procedure, it will exploit another aspect of canonical solutions: they start with the shortest way, if possible, to get the budget below the original baseline. This is illustrated by the following lemma.

Definition 5 A licit subset X of a graph G is a *lump* if:

1. $\delta(X) < 0$
2. $\forall X' \subsetneq X$ licit, we have $\delta(X') \geq 0$

Moreover, a lump X is *harmless* if its budget $bg(X)$ is minimal among all lumps, and $|X|$ is minimal among all lumps with this budget.

In other words, when ordering lumps according to the lexicographic order over $(bg(X), |X|)$, a harmless lump is an absolute minimum.

Property 3 *A lump X induces a connected subgraph of G.*

Proof Suppose $G[X]$ is composed of two components induced by $X_1 \subsetneq X$ and $X_2 \subsetneq X$. Both are licit, so by the definition of a lump, $\delta(X_1) \geq 0$ and $\delta(X_2) \geq 0$. However $\delta(X) = \delta(X_1) + \delta(X_2) < 0$, hence a contradiction. □

Lemma 4 *If S is a simple schedule for a bipartite graph G and X is a lump of G admitting an optimal schedule S_X such that $bg(S_X) \leq \min(pref_{-1}(S), bg(S))$, then $S' := S_X \cdot (S \setminus X)$ is preferable to S. It is moreover strictly preferable if either $bg(S_X) < pref_{-1}(S)$ or $|X| < |Pref_{-1}(S)|$.*

Proof We first recall the following notation: given a prefix ρ' of S' with $S_X \sqsubseteq \rho'$, we write $\mathrm{rem}_X(\rho')$ for the smallest prefix ρ of S such that $V(\rho \cup X) = V(\rho')$. Conversely, we also define for a prefix ρ of S $\mathrm{add}_X(\rho)$ as the smallest prefix ρ' of S such that $V(\rho \cup X) = V(\rho')$. Note that add_X and rem_X are monotonous under the prefix relation. These definitions are illustrated in Figure 4. Note also that for any prefix ρ of S, $\rho'' = \mathrm{rem}_X(\mathrm{add}_X(\rho))$ is the smallest prefix σ of S such that $V(\sigma \cup X) = V(\rho \cup X)$, so in particular $\rho'' \sqsubseteq \rho$. For any such pair ρ, ρ' with $V(\rho \cup X) = V(\rho')$, we show that $\delta(\rho') \leq \delta(\rho)$. Indeed, $\delta(\rho') = \delta(\rho \cup X) = \delta(\rho) + \delta(X) - \delta(\rho \cap X)$ (per the modularity of δ, Lemma 1). Since $\rho \cap X$ is licit (by Property 1) by the definition of lump we have $\delta(\rho \cap X) \geq \delta(X)$ and $\delta(\rho') \leq \delta(\rho)$.

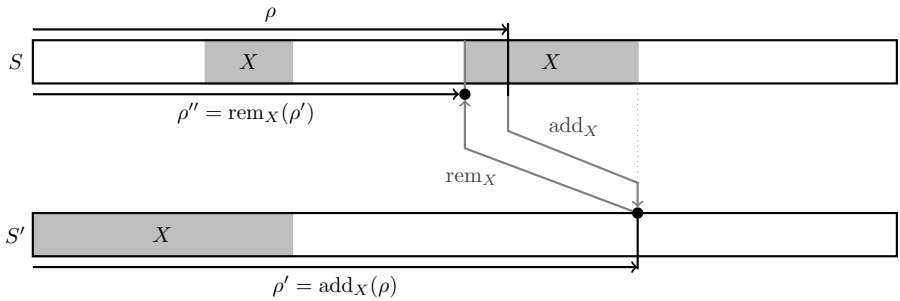


Fig. 4 Illustration of the definitions of add_X and rem_X, used in the proofs of Lemmas 4.2 (only *add*$_X$),4 and 6. They apply to a typical situation encountered in these Lemmas: a schedule S is shuffled so that a licit set X is processed first. The new schedule S' is valid since X is licit. The purpose of add_X and rem_X is then to draw connections between prefixes of S and corresponding prefixes in S', in order to infer bounds on $bg(S')$, $\mathrm{pref}_{-i}(S')$ or $\mathrm{suff}_{-i}(S')$.

We can now show that S' is preferable to S, starting with $bg(S') \leq bg(S)$. Consider a prefix ρ' of S'. If $\rho' \sqsubseteq S_X$, then $\delta(\rho') \leq bg(X) \leq bg(S)$. Otherwise, $S_X \sqsubseteq \rho'$, and $\delta(\rho') \leq \delta(\mathrm{rem}_X(\rho')) \leq bg(S)$.

Then, we have $\mathrm{pref}_{-1}(S') = bg(S_X) \leq \mathrm{pref}_{-1}(S)$ by assumption. To continue, for i such that $\mathrm{pref}_{-i}(S) < +\infty$, we have $\mathrm{pref}_{-i}(S') \sqsubseteq \mathrm{add}_X(\mathrm{pref}_{-i}(S))$ (indeed, $\delta(\mathrm{add}_X(\mathrm{pref}_{-i}(S))) \leq \delta(\mathrm{pref}_{-i}(S)) = -i$, so $\mathrm{add}_X(\mathrm{pref}_{-i}(S))$ is some prefix, not necessarily smallest, of S' with balance no more than $-i$). Thus, for any ρ', $S_X \sqsubseteq \rho' \sqsubseteq \mathrm{Pref}_{-i}(S')$, we have $\delta(\rho') \leq \delta(\mathrm{rem}_X(\rho'))$ and $\mathrm{rem}_X(\rho')$ is a prefix of $\mathrm{rem}_X(\mathrm{add}_X(\mathrm{Pref}_{-i}(S')))$ so $\delta(\mathrm{rem}_X(\rho')) \leq \mathrm{pref}_{-i}(S')$. Overall, any prefix of $\mathrm{pref}_{-i}(S')$ has balance at most $\max(bg(S_X), \mathrm{pref}_{-i}(S))$ and $bg(S_X) \leq \mathrm{pref}\_1(S') \leq \mathrm{pref}_{-i}(S')$ so $\mathrm{pref}_{-i}(S') \leq \mathrm{pref}_{-i}(S)$.

To finish, consider if it exists an i such that $\mathrm{suff}_{-i}(S) < +\infty$. By the existence of a licit subset X with $\delta(X) < 0$ and of $\mathrm{Suff}_{-i}(S)$, we have the existence of a mixed maximum independent set in G. As S is simple, it does reach this minimal balance, and there is $\rho \sqsubseteq S$ such that $\delta(\rho) = |L| - \alpha(G)$, the lowest possible value for δ. A useful consequence is that we must have $X \subseteq \rho$, as otherwise, $\delta(X \cap \rho) \geq 0$ (by definition of lump), and when reshuffling we would obtain $\delta(\rho \cup X) = \delta(\rho) + \delta(X) - \delta(X \cap \rho) \leq \delta(\rho) - 1$, which is not possible by minimality of $\delta(\rho)$ over licit subsets. As a consequence $\forall i$, $\mathrm{Suff}_{-i}(S') = \mathrm{Suff}_{-i}(S)$ and $\mathrm{suff}_{-i}(S') = \mathrm{suff}_{-i}(S)$.

Overall, $S' = S_X \cdot S \setminus X$ is indeed preferable to S. Note also that if $bg(S_X) < \mathrm{pref}_{-1}(S)$, then $\mathrm{pref}_{-1}(S') < \mathrm{pref}_{-1}(S)$, and if $|X| < |\mathrm{Pref}_{-1}(S)\|$, then $\ell_{S'}(-1) < \ell_S(-1)$: in both cases, S' is strictly preferable to S.

□

Lemma 4 is akin to the "commitment lemma" of [27] and Lemma 4.6 of [28]. However, in this paper, we link this result to newly-introduced notions of preferability (Definition 3) and canonicity (Definition 4). This is the case in particular of Lemma 6. It relies itself on the following existence result for lumps, essentially saying that a balanced licit set can always be reduced to a lump.

Lemma 5 *If X is a licit set with $\delta(X) < 0$, then either X is a lump or there is a lump $Y \subsetneq X$ with $bg(Y) \leq bg(X)$.*

Proof The proof is by induction on $|X|$. Pick $X' \subseteq X$ licit and minimal-by-inclusion under the condition $\delta(X') < 0$. Per the minimality criteria, X' is a lump. If $bg(X') \leq bg(X)$, we are done.

If $bg(X') > bg(X)$, consider S an optimal schedule for X, and ρ the largest prefix of S such that $\delta(\rho \cap X') = bg(X) + 1$: such a prefix exists, since any schedule of X' (and in particular $S \cap X'$) reaches balance $bg(X) + 1$ at some point. Let also σ denote the suffix of S corresponding to ρ, i.e. such that $S = \rho \cdot \sigma$. Consider now the set $Y' = \rho \setminus X'$. We have $\delta(\rho) = \delta(Y') + \delta(\rho \cap X') = \delta(Y') + (bg(X) + 1)$ and $\delta(\rho) \leq bg(X)$ so $\delta(Y') \leq -1$. Overall, $Y' \cup X'$ is a licit set $(= \rho \cup X')$ with balance ≤ -2. In addition, $bg(Y' \cup X') \leq bg(X)$ with the schedule $S' = \rho \cdot (\sigma \cap X')$. The definitions of ρ, X', Y', σ and their relations to one another are illustrated in Figure 5

Then, $Z = V(\text{Pref}_{-1}(S'))$ yields a licit subset with $\delta(Z) < 0$ and $Z \subsetneq Y' \cup X'$ (as $\delta(Z) = -1$ while $\delta(S') = -2$). Therefore $|Z| < |X|$, and we can apply the induction hypothesis to it: Z is either a lump or contains one. In either case, there is a lump strictly included in X. □

Lemma 6 *If G is a bipartite graph for which there exists an optimal schedule S with $pref_{-1}(S) < +\infty$, then a canonical schedule S_C for G necessarily starts with a harmless lump $X = V(Pref_{-1}(S_C))$.*

Proof We first show that $X = V(\text{Pref}_{-1}(S_C))$ is indeed a lump. By Lemma 5, since X is licit and $\delta(X) < 0$, either it is a lump or there exists a lump $Y \subseteq X$ with $|Y| < |X|$ and $bg(Y) \leq bg(X) = \text{pref}_{-1}(S_C) \leq bg(G)$. Applying Lemma 4 to S_C and Y would yield a schedule strictly preferable to S_C, which is not possible. X is therefore indeed a lump.

Finally, X is indeed harmless. Otherwise, there would exist a lump Y such that $bg(Y) < bg(X)$ or $|Y| < |X|$ if $bg(X) = bg(Y)$. By Lemma 4, $S_Y \cdot (S \setminus Y)$, with S_Y an optimal schedule for Y would be strictly preferable to the canonical schedule S, which is not possible.

□

Lemma 7 *If X is a harmless lump of G with optimal schedule S_X, and S_C a canonical schedule for $G \setminus X$, then $S_X \cdot S_C$ is a canonical schedule for G*

Proof Let S_G be a schedule for G. We will prove that $S_X \cdot S_C \preccurlyeq S_G$.

Let us first apply Corollary 4 to get S'_G simple and preferable to S_G. We can then apply Lemma 4 to S'_G and X. The only criteria to verify is $bg(S_X) \leq \min(\text{pref}_{-1}(S'_G), bg(S'_G))$. Since X is a licit subset with $\delta(X) < 0$, $\alpha(G) \geq |I(X)| \geq |L|$, and since S'_G is simple, $\text{pref}_{-1}(S'_G) < +\infty$. By Lemma 5, there exists a lump Y in $V(\text{Pref}_{-1}(S'_G))$ with $bg(Y) \leq \text{pref}_{-1}(S'_G)$. As X is a harmless lump, $bg(S_X) \leq$

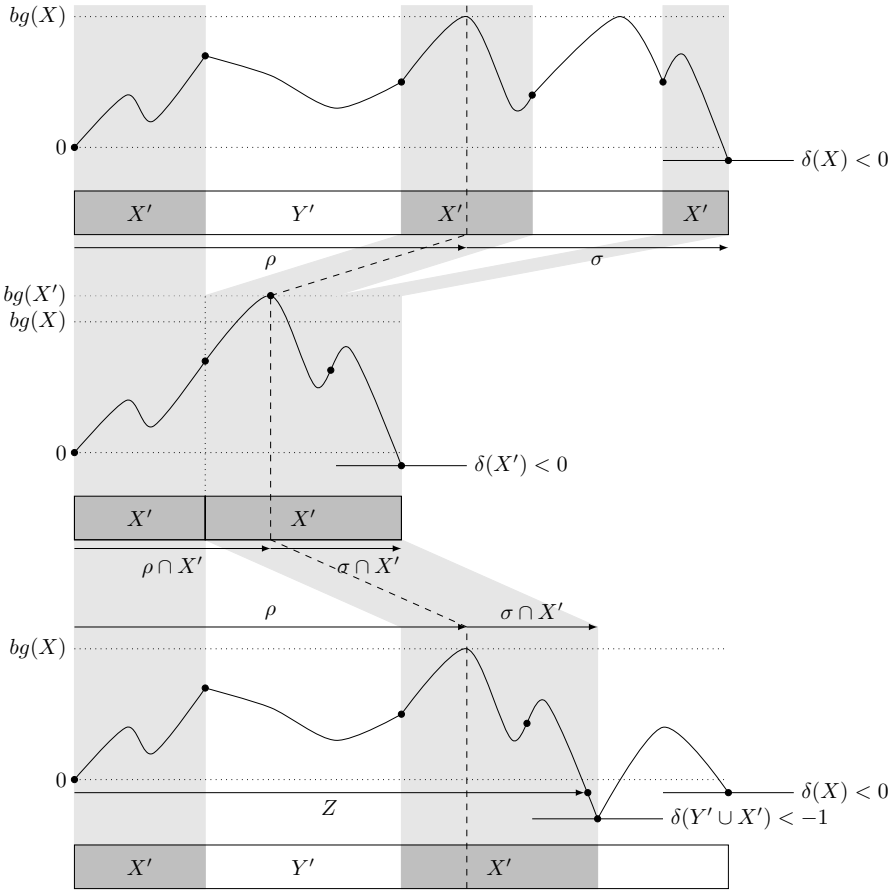


Fig. 5 Illustration of the objects used in the proof of Lemma 5. The purpose is to show that, given a licit set X such that $\delta(X) < 0$, if it is not a lump itself, a smaller licit set with negative balance $Z \subsetneq X$ can be found. The induction hypothesis can then be applied to Z to show that it contains a lump.

$bg(Y) \leq \text{pref}_{-1}(S'_G) = \min(bg(S'_G), \text{pref}_{-1}(S'_G))$. In addition, if $bg(S_X) = bg(Y)$, we know that $|S_X| \leq |Y|$.

By Lemma 4, $S_X \cdot (S'_G \setminus X) \preccurlyeq S'_G \preccurlyeq S_G$. Let us know replace $S'_G \setminus X$ by S_C, a canonical schedule for $G \setminus X$, and show that we obtain a schedule preferable to $S_X \cdot (S'_G \setminus X)$. We have $\text{Pref}_{-1}(S_X \cdot S_C) = S_X = \text{Pref}_{-1}(S_X \cdot (S'_G \setminus X))$, and $\forall i \in [-2, \ldots, |L| - \alpha(G)]$:

$$\text{Pref}_{-i}(S_X \cdot S_C) = S_X \cdot \text{Pref}_{-i+1}(S_C)$$

and

$$\text{Pref}_{-i}(S_X \cdot (S'_G \setminus X)) = S_X \cdot \text{Pref}_{-i+1}(S'_G \setminus X)$$

Given that S_C is canonical for $G \setminus X$, it is preferable to $S'_G \setminus X$. Therefore $\forall i$, $\text{pref}_{-i}(S_C) \leq \text{pref}_{-i}(S'_G \setminus X)$, with $\text{Pref}_{-i}(S_C)$ shorter in case of equality.

As for $\text{Suff}_{-i}(S_X \cdot S_C)$ for any $i \in [-1, \ldots, |R| - \alpha(G)]$, we have from the definition of a lump

$$\text{Suff}_{-i}(S_X \cdot S_C) = \text{Suff}_{-i}(S_C)$$

and

$$\text{Suff}_{-i}(S_X \cdot (S'_G \setminus X)) = \text{Suff}_{-i}(S'_G \setminus X)$$

As above, since S_C is canonical, $\text{suff}_{-i}(S_C) \leq \text{suff}_{-i}(S'_G \setminus X)$ with $\text{Suff}_{-i}(S_C)$ shorter in case of equality, and the same goes for $S_X \cdot S_C$ and $S_X \cdot (S'_G \setminus X)$.

Finally,

$$\begin{aligned} bg(S_X \cdot S_C) &= \max\left(bg(S_X), -1 + bg(S_C)\right) \\ &\leq \max\left(bg(S_X), -1 + bg(S'_G \setminus X)\right) \\ &= bg(S_X \cdot S'_G \setminus X) \end{aligned}$$

Overall, $S_X \cdot S_C \preccurlyeq S_G$ and is therefore a canonical schedule. □

Corollary 5 (existence of a canonical solution) *There always exists a canonical schedule for a given bipartite graph G*

Proof Consider S a simple optimal schedule for G. If $\text{pref}_{-1}(S) < +\infty$, then $\exists p \sqsubseteq S$ such that $\delta(p) < 0$. By Lemma 5, either p is a lump or it contains a lump X with $bg(X) \leq bg(Y) \leq \text{pref}_{-1}(S) \leq bg(S)$. Let us pick X harmless. By Lemma 7 with S_X an optimal schdule for X, and by induction, we get a canonical schedule for G.

If $\text{suff}_{-1}(S) < +\infty$, then $\text{pref}_{-1}(\overleftarrow{S}) < +\infty$. With the analysis above, we get a canonical schedule for $G_{R \to L}$ which can be reversed into a canonical schedule for G.

If none of the cases above apply, and because S is simple, then necessarily $|L| = |R|$ and G does not allow for any mixed-MIS. In that case, $\text{suff}_{-i}(S) = \text{pref}_{-i}(S) = +\infty \ \forall i$ and any optimal schedule is canonical. □

Lemma 8 *Let (G_1, G_2) be two disjoint bipartite graphs and (S_1, S_2) canonical solutions for (G_1, G_2) respectively. If $\text{pref}_{-1}(S_1)$, $\text{pref}_{-1}(S_2)$, $\text{suff}_{-1}(S_1)$ and $\text{suff}_{-1}(S_2)$ are all equal to $+\infty$ then both schedules $S_1 \cdot S_2$ and $S_2 \cdot S_1$ are optimal and canonical.*

Proof First, let us note that $\text{pref}_{-1}(S_1) = \text{pref}_{-1}(S_2) = \text{suff}_{-1}(S_1) = \text{suff}_{-1}(S_2) = +\infty$ implies $\delta(G_1) = \delta(G_2) = 0$, by definition of pref_{-1} and suff_{-1}.

Then, G_1 (resp. G_2) cannot allow for a mixed independent set of size $> |L_1| = |R_1|$ (resp. $|L_2| = |R_2|$), as it would imply by Lemma 2 the existence of an optimal schedule with $\text{pref}_{-1}(S_1) < +\infty$ (resp, $\text{pref}_{-1}(S_2) < +\infty$). As a consequence, The maximum independent sets of G are exactly $L_1 \cup L_2$, $L_1 \cup R_2$, $R_1 \cup L_2$ and $R_1 \cup R_2$.

Both $L_1 \cup R_2$ and $R_1 \cup L_2$ are mixed maximum independent sets. Consider therefore an optimal (and therefore, canonical) schedule S for G. By Lemma 2 both $S_1 \cdot (S \setminus G_1)$ and $S_2 \cdot (S \setminus G_2)$ are preferable to S and canonical. Replacing $S \setminus G_1$ by S_2 and $S \setminus G_2$ by S_1 does not increase the budget, which is equal to $\max(bg(S_1), bg(S_2))$. Since the budget is the only criteria for preferability in this case, both $S_1 \cdot S_2$ and $S_2 \cdot S_1$ are canonical. □

The following Theorem is the main result of this section, and essentially states that a canonical (and therefore optimal) solution for $G = G_1 \cup G_2$ can be obtained by interleaving canonical solutions for two disjoint graphs G_1 and G_2. This suggests the merging procedure implemented by Algorithm 1, where $\text{Pref}_{-1}(S_1), \text{Pref}_{-1}(S_2), \text{Suff}_{-1}(S_1)$ and $\text{Suff}_{-1}(S_2)$ are treated as "canonical blocks", *i.e.* are not broken up in the interleaving process.

Theorem 4 (merge algorithm) *If G_1 and G_2 are two disjoint bipartite graphs and S_1 and S_2 two canonical solutions for G_1 and G_2 respectively, then Algorithm 1 yields a canonical solution for G in $O(|S_1| + |S_2|)$.*

Proof **Run-time:** Given a schedule S, $\text{pref}_{-1}(S)$ and $\text{suff}_{-1}(S)$ and (if they exist) $\text{Pref}_{-1}(S), \text{Suff}_{-1}(S)$ can be computed in $O(|S|)$ by a simple iteration over S that keeps track of the budget.
Correctness: We prove the correctness by induction, with the base case being when $S_1 = \emptyset$ or $S_2 = \emptyset$.

In the general case, if $\min(\text{pref}_{-1}(S_1), \text{pref}_{-1}(S_2)) < +\infty$, consider (w.l.o.g) that $(\text{pref}_{-1}(S_1), \ell_{S_1}(-1)) \leq_{\text{lex}} (\text{pref}_{-1}(S_2), \ell_{S_2}(-1))$, with $\leq_{\text{lex}}$ denoting the lexico-graphic order.

By Lemma 6 applied to G_1 and S_1, $\text{Pref}_{-1}(S_1)$ is a harmless lump of G_1. It is therefore a lump of G. Let us prove it is also harmless in G. To that end, note that lumps are connected (Property 3), so a lump of G is either a lump of G_1 or G_2. Therefore $(\text{pref}_{-1}(S_1), \ell_{S_1}(-1)) \leq_{\text{lex}} (\text{pref}_{-1}(S_2), \ell_{S_2}(-1))$ indeed implies that $\text{Pref}_{-1}(S_1)$ is of minimal budget among lumps of G, and shorter than $\text{Pref}_{-1}(S_2)$ in case of budget equality.

Since by the induction hypothesis $\textsc{Merge}(S_1 \setminus \text{Pref}_{-1}(S_1), S_2)$ is canonical, $\text{Pref}_{-1}(S_1) \cdot \textsc{Merge}(S_1 \setminus \text{Pref}_{-1}(S_1), S_2)$ is canonical by Lemma 7

The case $\min(\text{suff}_{-1}(S_1), \text{suff}_{-1}(S_2)) < +\infty$ (lines 10-16) is treated with the same arguments, given the symmetry of pref and suff when inverting L and R, namely $\forall S\ \text{suff}_{-1}(S) = \text{pref}_{-1}(\overleftarrow{S})$.

As for the justification of the concatenation if none of the conditions above apply, it is brought by Lemma 8.

□

5 Parameterized algorithms for bipartite independent set reconfiguration

In this section, we apply the technical results of the previous section to the design of XP algorithms for BISR. For example, Lemma 2 is used to formulate a $O(n^2)$-space, $O(n^{2\rho})$-time algorithm for BISR, described in Section 5.1. As for Lemma 4, it allows to build a $O(n^{\Phi+1})$ algorithm for BISR when restricted to *bipartite circle graphs*, described in Section 6. Bipartite circle graphs constitute a sub-case of interest to RNA kinetics, as we shall see in more details in Section 7.

Algorithm 1 Merge procedure for canonical schedules. $\leq_{\text{lex}}$ denotes the lexicographic order, applied here to couples of integers.

Input: S_1, S_2 canonical solutions for G_1, G_2 (disjoint graphs)
Output: a canonical solution S for $G = G_1 \cup G_2$

1: **function** MERGE(S_1, S_2):
2: ▷ If first or last canonical blocks exist: recurse
3: **if** $\min(\text{pref}_{-1}(S_1), \text{pref}_{-1}(S_2)) < +\infty$ **then**
4: **if** $(\text{pref}_{-1}(S_1), \ell_{S_1}(-1)) \leq_{\text{lex}} (\text{pref}_{-1}(S_2), \ell_{S_2}(-1))$ **then**
5: **return** $\text{Pref}_{-1}(S_1) \cdot \text{MERGE}(S_1 \setminus \text{Pref}_{-1}(S), S_2)$
6: **else**
7: **return** $\text{Pref}_{-1}(S_2) \cdot \text{MERGE}(S_1, S_2 \setminus \text{Pref}_{-1}(S_2))$
8: **end if**
9: **end if**
10: **if** $\min(\text{suff}_{-1}(S_1), \text{suff}_{-1}(S_2)) < +\infty$ **then**
11: **if** $(\text{suff}_{-1}(S_1), r_{S_1}(-1)) \leq_{\text{lex}} (\text{suff}_{-1}(S_2), r_{S_2}(-1))$ **then**
12: **return** $\text{MERGE}(S_1 \setminus \text{Suff}_{-1}(S_1), S_2) \cdot \text{Suff}_{-1}(S_1)$
13: **else**
14: **return** $\text{MERGE}(S_1, S_2 \setminus \text{Suff}_{-1}(S_2)) \cdot \text{Suff}_{-1}(S_2)$
15: **end if**
16: **end if**
17: ▷ If no first or last canonical block exist: simply concatenate
18: **return** $S_1 \cdot S_2$
19: **end function**

5.1 An XP algorithm in ρ

Lemma 2 allows for a divide-and-conquer approach: if we identify a separator X in G, i.e. a licit subset of G such that $I(X)$ is a mixed independent set, then we may independently solve the problem of finding a ρ-realization from L to $I(X)$ and then from $I(X)$ to R. If no solution is found for one of them, then the converse of Lemma 2 implies that no ρ-realizations exists for G. The algorithm presented in this section is based on this approach.

Algorithm details. We present here a direct algorithm for BIPARTITE INDEPENDENT SET RECONFIGURATION, detailed in Algorithm 2. The main function `Realize` is recursive. Its sub-calls arise either from a split with a mixed MIS I (in which case it is called on a smaller graph but with the same parameter), or from the loop over all possible starting points in the case where no separator is found (lines 13-18), in which case the parameter does reduce. The overall runtime is dominated by this loop, and is analyzed in Proposition 5 below.

Mixed MIS algorithm. The sub-routine allowing to find, if it exists, a maximum independent set intersecting both L and R is based on concepts from *matching theory* [29], namely the *Dulmage-Mendelsohn* decomposition [29, 30], as well as the decomposition of bipartite graphs with a perfect matching into *elementary subgraphs* [29](part 4.1). Its full details are described in the full version of the article.

Algorithm 2 XP algorithm for Bipartite Independent Set Reconfiguration

Input: bipartite graph G (with sides L and R), integer ρ
Output: a ρ-realization for G, if it exists

function Realize(G, ρ):

  if $\rho < 0$ **then return** $\perp$ ▷ // Terminal cases:
  end if
  if $L \cup R = \emptyset$ **then return** $\emptyset$
  end if
  if $\exists \ell \in L$ s.t $N(\ell) = \emptyset$ **then return** Realize($G \setminus \{\ell\}$, $\rho - 1$) $\cdot\ \ell$
  end if
  if $\exists r \in R$ s.t $N(r) = \emptyset$ **then return** $r \cdot$ Realize($G \setminus \{r\}$, $\rho - 1$)
  end if

  $I =$ MixedMIS(G) ▷ // Trying to find a separator
  if $I \neq \perp$ **then**
    $S = (L \setminus I) \cup (R \cap I)$
    return Realize($G[S], \rho$)$\cdot$ Realize($G[V \setminus S], \rho$)
  else
    for $(\ell, r) \in L \times R$ **do** ▷ // loop over all start-end possibilites
      if Realize($G \setminus \{\ell, r\}, \rho - 1$)$\neq \perp$ **then**
        return $\ell \cdot$ Realize($G \setminus \{\ell, r\}, \rho - 1$) $\cdot\ r$
      end if
    end for
  end if
end function

Proposition 5 *Algorithm 2 runs in $O(|V|^{2\rho}\sqrt{|V|}|E|)$ time, while using $O(|V|^2)$ space, where ρ is the difference between the minimum allowed and maximum possible independent set size, along the reconfiguration.*

Proof Let us start with space: throughout the algorithm, one needs only to maintain a description of G and related objects (independent set I, maximum matching M, associated directed graph $H(G, M)$) for which $O(|V|^2)$ is enough.

As for time, let $C(n_1, n_2, \rho)$ be the number of recursive calls of the function *Realize* of Algorithm 2 when initially called with $|L| = n_1$, $|R| = n_2$, and some value of ρ. We will show by induction that $C(n_1, n_2, \rho) \leq (n_1 + n_2)^{2\rho}$. Since each call involves one computation of a maximum matching, this will prove our result.

Given (n_1, n_2, ρ), suppose therefore that $\forall (n_1', n_2', \rho') \neq (n_1, n_2, \rho)$ with $n_1' \leq n_1, n_2' \leq n_2, \rho' \leq \rho$ we have $C(n_1', n_2', \rho') < (n_1' + n_2')^{2\rho'}$

1. If G allows for a mixed maximum independent set, the instance is split into two smaller instances, yielding $C(n_1, n_2, \rho) = C(n_1', n_2, \rho) +$

$C(n_1'', n_2'', \rho)$ with $n_1' + n_1'' = n_1$ and $n_2 = n_2' + n_2''$. And $C(n_1, n_2, \rho) \leq \left((n_1' + n_2')^{2\rho} + (n_1'' + n_2'')^{2\rho}\right) \leq (n_1' + n_1'' + n_2' + n_2'')^{2\rho} \leq (n_1 + n_2)^{2\rho}$.

2. else, we have the following relation: $C(n_1, n_2, \rho) = n_1 n_2 \cdot C(n_1 - 1, n_2 - 1, \rho - 1)$. Which yields:

$$\begin{aligned} C(n_1, n_2, \rho) &= n_1 n_2 \cdot C(n_1 - 1, n_2 - 1, \rho - 1) \\ &\leq n^2 \cdot n^{2(\rho-1)} \quad \text{by induction hypothesis} \\ &\leq n^{2\rho} \end{aligned}$$

□

The exponential part ($O(n^{2\rho})$) of the worst case complexity of Algorithm 2 is in fact tight, as it is met with a complete bi-clique $K_{n,n}$ with sides of size n. Indeed, in this case, no mixed MIS is found in any of the recursive calls.

6 RNA case: bipartite circle graphs

6.1 RNA basics and arboricity parameter

RNA structures. RiboNucleic Acids (RNAs) are biomolecules of outstanding interest for molecular biology, which can be represented as strings over an alphabet $\Sigma := \{\mathsf{A}, \mathsf{C}, \mathsf{G}, \mathsf{U}\}$. Importantly, these strings may *fold* on themselves to adopt one or several conformation(s), also called *structures*. For a string of length N, a conformation is typically described by a set S of base pairs (i, j), with $1 \leq i < j \leq N$. Then, a standard class of conformations to consider in RNA bioinformatics are *secondary structures*, which are pairwise non-crossing ($\nexists (i, j), (k, l) \in S$ such that $i \leq k \leq j \leq l$, in particular, they involve distinct positions). Due to this non-crossing property, secondary structures are in bijection with *well-parenthesized* strings, as illustrated in Figure 6 (B).

RNA Energy barrier problem. In this section, we more precisely work on the problem of finding a reconfiguration pathway between two *secondary structures* (i.e conflict-free sets of base pairs). The reconfiguration may only involve secondary structures, and remain of energy as low as possible. We work with a simple energy model consisting of the opposite of *number of base pairs* in a configuration ($-N_{bps}$). The RNA Energy-Barrier problem can then be stated as such:

RNA Energy-Barrier
Input: Secondary structures L and R; Energy barrier $k \in \mathbb{N}^+$
Output: True if there exists a sequence $S_0 \cdots S_\ell$ of secondary structures such that

- $S_0 = L$ and $S_\ell = R$;
- $|S_i| \geq |L| - k, \forall i \in [0, \ell]$;
- $|S_i \triangle S_{i+1}| = 1, \forall i \in [0, \ell - 1]$.

False otherwise.

Problem motivation. Since the number of secondary structures available to a given RNA grows exponentially with n, RNA energy landscapes are notoriously *rugged*, *i.e.* feature many local minima, and the folding process of an RNA from its *synthesis* to its theoretical final state (a thermodynamic equilibrium around low energy conformations) can be significantly slowed down. Consequently, some RNAs end up being degraded before reaching this final state. This observation motivates the study of RNA kinetics, which encompass all time-dependent aspects of the folding process. In particular, it is known (Arrhenius law) that the energy barrier is the dominant factor influencing the transition rate between two structures, with an exponential dependence.

BISR **on circle graphs.** Two arcs (i, j) and (k, l) are said to be in *conflict* or *crossing* if $i \leq k \leq j \leq l$ or $k \leq i \leq l \leq j$ (i.e. when there is not one of them nested in the other). It simply means that they cannot be both present at the same time in an RNA secondary structure.

To capture this constraint in reconfiguring RNA secondary structures, we define the *conflict graph* $G(L, R)$ as having $L \cup R$ as vertices, and an edge connecting two arcs if they are in conflict. L and R being two valid secondary structures, the graph is bipartite. More generally, a valid secondary structure is then an *independent set* of $G(L, R)$. Reconfiguring L into R while minimizing energy along the way then consists in solving BISR on $G(L, R)$. The following proposition characterizes the set of bipartite graphs that emerge from this construction, as *bipartite circle graph*. A circle graph is an intersection-graph of chords of a circle.

Proposition 6 *The* RNA ENERGY BARRIER *problem as defined above is* BISR *restricted to bipartite circle graphs*

Proof Given L and R two well-nested arc sets (i.e. two RNA secondary structures) over $[1 \dots n]$, denoted $L = \{(l_i, r_i)\}$ and $R = \{b_j, e_j\}$. Consider a circle with n regularly-spaced positions on it, and the set of chords $L \cup R$. The associated circle graph (chord-intersection graph) is exactly the conflict-graph $G(L, R)$. There is then an exact correspondance between independent sets of $G(L, R)$ and valid secondary structures composed of arcs from L and R.

Conversely, given a bipartite circle graph, its two sides L and R yield two well-nested arc sets that can be seen as RNA secondary structures. The correspondence is highlighted on Figure 6. □

Notations for base-pair relations. Given two base-pairs (i, j) and (k, l), we write $(k, l) \subset (i, j)$ if (k, l) is *nested* in (i, j), i.e. if $i < k < l < j$. One may see this notation as "the interval $[k, l]$ is a proper subset of the interval $[i, j]$". For two non-conflicting base-pairs, if no one of them is nested in the other, we write $(i, j) \parallel (k, l)$, which means either $i < j < k < l$ or $k < l < i < j$.

Arboricity (Φ). Given an RNA secondary structure S (a set of well-nested base-pairs) the arbroricity Φ of S is the number of "terminal" base-pairs, i.e.

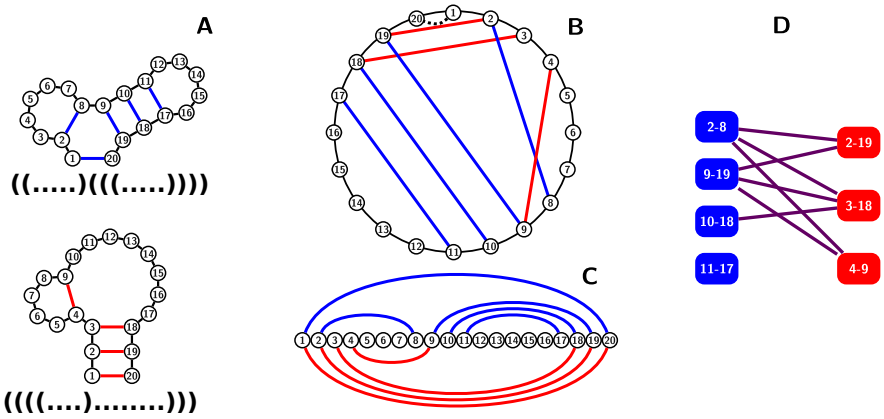


Fig. 6 Conflict bipartite graph (D) associated with an instance of the RNA Energy-Barrier problem, consisting of an initial (A) and final (B) structure, both represented as an arc-annotated sequence (C). The sequence of valid secondary structures, achieving minimum energy barrier can be obtained from the solution given in Figure 6.

the number of base-pairs that do not contain any nested base-pair. A formal definition is given below:

Definition 6 The arboricity $\Phi(S)$ of a set of well-nested base-pairs is:

$$\Phi(S) = |\{(i,j) \in S \mid \nexists (k,l) \in S \text{ with } (k,l) \subset (i,j)\}|$$

When seeing an RNA secondary structure as a set of well-parenthesized strings (Figure 6.A for instance), it is the number of matching opening/closing parenthesis symbols that only have dots between them.

Separating inside and outside sub-instances. Given L and R two well-nested arc sets, and $\ell = (i,j)$ an element of L. ℓ defines naturally "inside" and "outside" sub-instances in L and R, that are only connected through $N(\ell)$, the elements of R in conflict with ℓ. Formally these "inside" and "outside" sub-instances are $(L^\ell_{\text{IN}}, R^\ell_{\text{IN}})$ and $(L^\ell_{\text{OUT}}, R^\ell_{\text{OUT}})$ with $L^\ell_{\text{IN}} = \{\ell' \in L \mid \ell' \subset \ell\}$, $R^\ell_{\text{IN}} = \{\ell' \in R \mid \ell' \subset \ell\}$, $L^\ell_{\text{OUT}} = \{\ell' \in L \mid \ell \subset \ell' \text{ or } \ell \parallel \ell'\}$ and $R^\ell_{\text{OUT}} = \{\ell' \in R \mid \ell \subset \ell' \text{ or } \ell \parallel \ell'\}$ Note that $\{\ell\}, L^\ell_{\text{IN}}, L^\ell_{\text{OUT}}$ form a partition of L, and $N(\ell), R^\ell_{\text{IN}}, R^\ell_{\text{OUT}}$ form a partition of R.

6.2 An XP algorithm for Φ

Dynamic programming table. The algorithm we present in this Section (Algorithm 3) is based on dynamic programming, using a memorization strategy. There is therefore a table in which solutions to partial instances are stored. Given L, R input secondary structures to the RNA Energy Barrier problem, the indices to this table are sets of the form $\{\ell, \ell_1 \dots \ell_p\}$, with $\ell \in L \cup \{(1,N)\}$ and $\ell_i \in L$, such that $\forall i\ \ell_i \subset \ell$, and $\forall i \neq j,\ \ell_i \parallel \ell_j$.

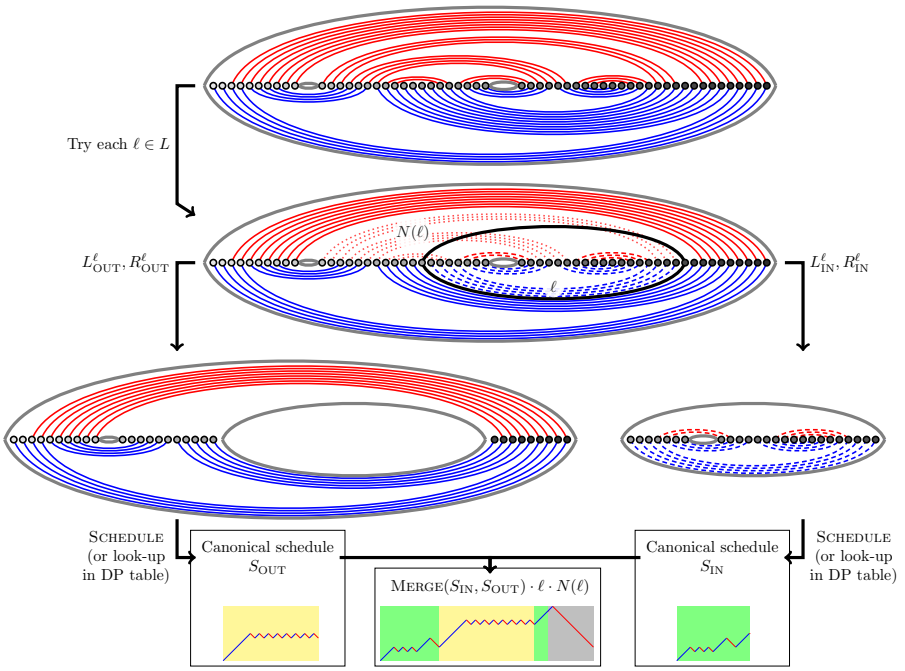


Fig. 7 Illustration of Algorithm 3. Given a sub-instance L, R (bordered by gray ovals, top figure), each $\ell \in L$ is tried (middle figure), yielding two smaller sub-instances corresponding to the outside and inside of ℓ (bottom left and right figures). After solving each sub-instance independently, using the DP-table for memorization, a solution is obtained for (L, R) by merging both solutions and appending ℓ and all its neighborhood (which were not part of any sub-instance). Here the arboricity is 3 (there are 3 minimal arcs in L), so any border uses at most 4 arcs, giving the upper bound of $\binom{n}{4}$ on the number of sub-instances

The reason $(1, N)$ is a possible value for ℓ is that it defines an interval to which partial instances are restricted. Originally, there is no restriction and $\ell = (1, N)$. The partial instance associated to such a set is L', R' with $L' = L^{\ell}_{\text{IN}} \cap \left[\bigcap_{1 \leq i \leq p} L^{\ell_i}_{\text{OUT}} \right]$ and $R' = R^{\ell}_{\text{IN}} \cap \left[\bigcap_{1 \leq i \leq p} R^{\ell_i}_{\text{OUT}} \right]$. We also denote these structures by $L(\ell, \ell_1, \ldots, \ell_p)$ and $R(\ell, \ell_1, \ldots, \ell_p)$.

Informally, seeing L as a tree structure, the arcs $\{\ell, \ell_1, \ldots, \ell_p\}$ define a "sub-tree" of L. ℓ sets the root of this sub-tree, while $\ell_1 \ldots, \ell_p$ cut out some branches. Let us write

$$\mathcal{ST}(L) = \left\{ (\ell, \ell_1, \ldots, \ell_p) \in \left(L \cup \{(1, N)\}\right) \times L^p \mid \forall i \; \ell_i \subset \ell \text{ and } \forall i \neq j \; \ell_i \parallel \ell_j \right\}$$

for the set of all such "sub-trees", i.e. the set of all indices to the dynamic programming table.

Lemma 9 *Given an RNA structure L of arboricity Φ, $|\mathcal{ST}(L)| = O(\frac{n^{\Phi+1}}{\Phi!})$*

Proof Let us start by noting that in an RNA structure, each arc is either terminal or contains a terminal arc nested in it. The set $\{\ell_i\}_{1\leq i\leq p}$ being composed of arcs mutually not nested in one another, each of them contains (or is) a different terminal arc. As there are less than Φ terminal arcs in total, given ℓ, there are less than $\binom{n}{\Phi}$ possibilities for $\ell_1 \dots \ell_p$. Multiplied by the number of possibilities for ℓ, we get an upper bound of $(n+1)\cdot\binom{n}{\Phi} = O(\frac{n^{\Phi+1}}{\Phi!})$.

□

Theorem 7 *Algorithm 3 outputs a canonical (and therefore optimal) schedule in time* $O(\frac{n^{\Phi+2}}{\Phi!})$.

Proof **Run-time.** The initial call to SCHEDULE(L, R) corresponds to the entry $\ell = (1, N)$ and $\{\ell_1 \dots \ell_p\} = \emptyset$. Then, consider a call of SCHEDULE on two structures $L(\ell, \ell_1, \dots, \ell_p), R(\ell, \ell_1, \dots, \ell_p)$ respectively equal to $L^{\ell}_{\text{IN}} \cap \left[\bigcap_{1\leq i\leq p} L^{\ell_i}_{\text{OUT}}\right]$ and $R^{\ell}_{\text{IN}} \cap \left[\bigcap_{1\leq i\leq p} R^{\ell_i}_{\text{OUT}}\right]$, for $(\ell, \ell_1, \dots, \ell_p) \in \mathcal{ST}(L)$. The recursive calls to the "inside" and "outside" of some $\ell' \in L'$ (line 14-15) will give rise to the instances corresponding to the elements of $\mathcal{ST}(L)$ $(\ell', \{\ell_i \mid \ell_i \subset \ell'\})$ (inside) and $(\ell, \ell', \{\ell_i \mid \ell_i \parallel \ell'\})$ (outside). By induction, all recursive calls to SCHEDULE are of these forms, and the indices to the memorization table are elements of $\mathcal{ST}(L)$. Conversely, any element $(\ell, \ell_1, \dots, \ell_p)$ of $\mathcal{ST}$ sees its corresponding instance $L(\ell, \ell_1, \dots, \ell_p), R(\ell, \ell_1, \dots, \ell_p)$ emerge in some recursive call (e.g. taking the inside of ℓ in the first recursive call and then the outside of $\ell_1 \dots \ell_p$). Their number is smaller than $O(\frac{n^{\Phi+2}}{\Phi!})$ by Lemma 9. Let now us call $c(L, R)$ the computational cost of SCHEDULE(L, R), and $i(L, R)$ the "internal" cost of SCHEDULE, i.e. of all lines of Algorithm 3 except lines 18-19 (recursive calls). Given $\ell \in L$, we write $L^{\ell}_{\text{IN}}, R^{\ell}_{\text{IN}}$ and $L^{\ell}_{\text{OUT}}, R^{\ell}_{\text{OUT}}$ the sub-instances composed of arcs strictly inside or outside of ℓ. Then, we have:

$$c(L, R) = i(L, R) + \sum_{\ell\in L} c(L^{\ell}_{IN}, R^{\ell}_{IN}) + c(L^{\ell}_{OUT}, R^{\ell}_{OUT})$$

Which, by induction, given the discussion above, allows to show that:

$$c(L, R) = \sum_{(\ell,\ell_1,\dots,\ell_p)\in\mathcal{ST}(L)} i\left(L(\ell, \ell_1, \dots, \ell_p), R(\ell, \ell_1, \dots, \ell_p)\right)$$

$i(L, R)$ is $O(n^2)$ (linear MERGE for each $\ell \in L$), which yields $O(\frac{n^{\Phi+2}}{\Phi!})$ overall.

Correctness. By Corollary 5, a canonical schedule S for G exists. Some element $\ell \in L$ is necessarily processed last, such that $S = S' \cdot \ell \cdot N(\ell)$. When ℓ is considered as part of the for loop line 17 of Algorithm 1, the candidate solution is $S'' \cdot \ell \cdot N(\ell)$, with $S'' = \text{MERGE}(S_{IN}, S_{OUT})$. Sequence S'' is canonical by induction and the correctness of MERGE (Theorem 4). Thus, $S'' \preccurlyeq S'$ and the candidate solution is preferable to S, and therefore canonical.

□

Algorithm 3 XP algorithm in Φ for Bipartite Independent Set Reconfiguration

Input: bipartite circle graph G (with sides L and R)
Output: a canonical schedule for G
Global variable: Dynamic programming table $M : (L, R) \to S$, storing input/output pairs for SCHEDULE.

function SCHEDULE(L, R):
  if (L, R) is in $M.keys()$ **then** ▷ If already computed then return;
    return $M[(L, R)]$;
  end if
  $M[(L, R)] = L \cdot R$ ▷ Initializing $M[(L, R)]$ with a simple value
  if $L = \emptyset$ **then**
    return $M[(L, R)]$
  end if
  if $\exists r \in R$ such that $N(r) = \emptyset$ **then**
    $M[(L, R)] = r \cdot$ SCHEDULE$(L, R \setminus \{r\})$
    return $M[(L, R)]$
  end if
  for ℓ `in` L **do**
    $S_{IN} =$ SCHEDULE(L_{IN}, R_{IN}) ▷ ℓ defines an inside and an outside
    $S_{OUT} =$ SCHEDULE(L_{OUT}, R_{OUT})
    $S' =$ MERGE$(S_{IN}, S_{OUT}) \cdot \ell \cdot N(\ell)$
    if $S' \preccurlyeq M[(L, R)]$ **then** ▷ If S' is preferable to $M[(L, R)]$
      $M[(L, R)] = S'$
    end if
  end for
  return $M[(L, R)]$
end function

7 Benchmarks and Applications

In this section, we report benchmark results for all of our algorithms. We first explain some details about the algorithm we implemented for directed pathwidth. Then, we present a general benchmark of Algorithm 2 and the directed pathwidth approach, on random (Erdös-Rényi) bipartite graphs.

Last, we compare Algorithm 3 with the directed pathwidth approach on bipartite circle graphs, i.e. RNA instances.

Code availability. The code used for our benchmarks, including a Python/C++ implementation of our two algorithms, is available at https://gitlab.inria.fr/bmarchan/bisr-dpw (Algorithm 2 and directed pathwidth algorithm [27]) and https://gitlab.inria.fr/bmarchan/barrier-subtree (for Algorithm 3).

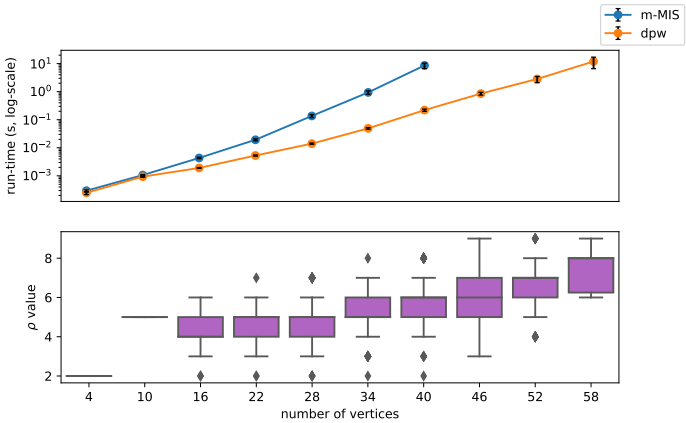


Fig. 8 (top panel) Average run-time (seconds, log-scale) of our algorithms on random Erdös-Rényi bipartite graphs, with a probability of connection such that the average degree of a vertex is 5 (i.e $p = 5/n$). (bottom panel) Average parameter value of generated instances, as a function of input size.

7.1 Implementation details

Directed pathwidth. We implemented and used an algorithm from Tamaki [1], with a runtime of $O(n^{\rho+2})$. This algorithm was originally published in 2011 [1]. In 2015, H.Tamaki and other authors described this algorithm as "flawed" in [12], and replaced it with another XP algorithm for directed pathwidth, with a run-time of $O(\frac{mn^{2\rho}}{(\rho-1)!})$.

Upon further analysis from our part, and discussions with H. Tamaki and the corresponding author of [12], it appears a small modification allowed to make the algorithm correct. In a nutshell, the algorithm involves *pruning* actions, and these need to be carried out *as soon as they are detected*. In [1], temporary solutions were accumulated before a general pruning step. With this modification, the analysis presented in [1] applies without modification, and yields a time complexity of $O(n^{\rho+2})$. The space complexity is unchanged at $O(n^{\rho+1})$. For completeness, a detailed re-derivation of the results of [1] is included in the full version of the article.

Mixed-MIS algorithm implementation. On Figure 8, the "m-MIS"-curve, corresponds to our

mixed-MIS-based algorithm in $O(n^{2\rho}\sqrt{|V||E|})$. Compared to the algorithm presented in Algorithm 2, a more efficient rule is used in the non-separable case: we loop over all possible $r \in R$ and add $N(r) \cdot r$ to the schedule (instead of a single vertex $\ell \in L$).

7.2 Random bipartite graphs

Benchmark details. Figure 8 shows, as a function of the number of vertices, the average execution time of both our algorithms (top panel), as well as the

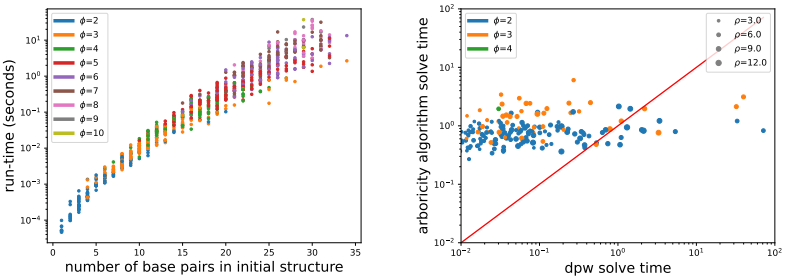


Fig. 9 (left) Execution time of Algorithm 3 on pairs of random RNA secondary structures. Points are colored as a function of the smallest arboricity Φ between the two structures. As expected, exact computing the energy barrier between the structures tends to become more expensive for larger values of Φ. (right) Comparison with a scatter plot of the execution times of Algorithm 3 and our implemention of [27] (directed pathwidth algorithm), on random pairs of RNA secondary structures. The color of the points denote the arboricity (Φ) value while the size is the directed pathwidth (range ρ). Surprisingly, the execution time of the directed pathwidth algorithm, whose complexity is $O(n^{\rho+2})$, does not correlate with the value of ρ. It suggests the existence of a structural property of directed graphs emerging from RNA instances making [27] faster.

distribution of parameter values (ρ - bottom panel), on a class of random bipartite graphs. These graphs are generated according to an Erdös-Rényi distribution (each pair of vertices has a constant probability p of forming an edge). We use a connection probability of d/n, dependent on the number of vertices. It is such that the average degree of vertices is d. The data of our benchmark (Figure 8) has been generated with $d = 5$.

Comments on Figure 8. The difference in trend between the execution times of the two algorithms is quite coherent with the difference in their exponents ($n^{\rho+2}$ vs. $n^{2\rho+2.5}$).

7.3 random RNA instances (bipartite circle graphs)

Benchmark details. Figure 9 shows the average execution time of Algorithm 3 on random RNA instances, and compares it with the directed pathwidth algorithm (right panel). Random instances are generated according to the following model: two secondary structures L, R are chosen *uniformly* at random (within the space of all possible secondary structure). Base pairs are constrained to occur between nucleotides separated by a distance of at least $\theta = 1$ (left panel) and $\theta = 3$ (right panel).

Random secondary structure generation. The random RNA secondary structure of Figure 9 are obtained by *uniform sampling* of well-parenthesized strings of a given length N. Two parameters control the probability distributions: the minimal distance θ between an opening bracket and its corresponding closing bracket and the probability p_{pb} of being base-paired.

7.4 RNA specific optimizations

Dynamic Programming and RNA. Given two secondary structures L and R, a mixed MIS of $G(L, R)$ is a *maximum conflict-free* subset of $L \cup R$, containing at least a base pair from L and R. As is the case for many algorithmic problems involving RNA, the fact that RNAs are *strings* and that base pairs define *intervals* suggests a dynamic programming approach to the mixed maximum independent set problem in RNA conflict graphs. Subproblems will correspond to *intervals* of the RNA string. Let us start with a simple dynamic programming scheme allowing to compute an unconstrained MIS.

Unconstrained MIS DP scheme. A maximum conflict-free subset of $L \cup R$ can be computed by dynamic programming, using the following DP table: for each $1 \leq i \leq j \leq n$, let $MCF_{i,j}$ be the size of a maximum conflict-free subset of all base pairs included in $[i, j]$.

Lemma 10 *$MCF_{1,n}$ can be computed in time $O(n^2)$*

Proof We have the following recurrence formula:

$$MCF_{i,i'} = 0, \forall i' < i$$

$$MCF_{i,j} = \max \begin{cases} MCF_{i+1,j} \\ \max_{(i,k)\in L\cup R} 1 + MCF_{i+1,k-1} + MCF_{k+1,j} \end{cases}$$

Note that the last *max* is over at most two possible pairs (i, k) (1 from L and 1 from R), per the fact that L and R are both conflict-free. □

Mixed MIS DP scheme. The following modifications to the DP scheme above allow to compute a mixed MIS of $G(L, R)$ while retaining the same complexity. In addition to the interval, we index the table by Boolean α and β which, when true, further restricts the optimization to subsets with > 0 pair from L (iff $\alpha = \mathsf{True}$) or R (iff $\beta = \mathsf{True}$):

$$MCF^{\alpha,\beta}_{i,i'} = \begin{cases} 0 & \text{if } (\alpha, \beta) = (\mathsf{False}, \mathsf{False}) \\ -\infty & \text{otherwise} \end{cases}, \forall i' < i$$

$$MCF^{\alpha,\beta}_{i,j} = \max \begin{cases} MCF^{\alpha,\beta}_{i+1,j} & \\ \max\limits_{\substack{(i,k)\in E \\ \alpha',\alpha'',\beta',\beta''\in\mathbb{B}^4}} 1 + MCF^{\alpha',\beta'}_{i+1,k-1} + MCF^{\alpha'',\beta''}_{k+1,j} & \left| \begin{array}{l} \text{if } \neg\alpha \vee \alpha' \vee \alpha'' \vee ((i,k) \in L) \\ \text{and } \neg\beta \vee \beta' \vee \beta'' \vee ((i,k) \in R) \end{array} \right. \end{cases}$$

Through a suitable memorization, the system can be used to compute in $\mathcal{O}(n^2)$ the maximum cardinality $MCF^{\mathsf{True},\mathsf{True}}_{1,n}$ of a subset over the whole sequence. A backtracking procedure is then used to rebuild the maximal subset.

8 Conclusion

Motivated by the development of exact parameterized algorithms for the RNA Energy Barrier problem, we studied several parameterizations for Bipartite Independent Set Reconfiguration. For the *range* ρ of possible

cardinalities for the independent sets along the reconfiguration as a parameter, we give a direct $O(n^2)$-space, $O(n^{2\rho+2.5})$-time algorithm (Algorithm 2), and a indirect $O(n^{\rho+1})$-space, $O(n^{\rho+2})$-time algorithm [27] through an equivalence with directed pathwidth.

In the case of RNA instances, i.e. BISR on bipartite circle graphs, we additionally study an *arboricity* parameter denoted Φ, that should intuitively be much smaller than the size of instances on natural RNA structures. For this parameter, we also provide an XP algorithm, with complexity $O(\frac{n^{\Phi}}{\Phi!})$ in space and $O(\frac{n^{\Phi+2}}{\Phi!})$ in time. This algorithm involves a novel *merge* procedure for optimal solutions of disjoint instances of *mimimum cumulative-cost* scheduling, which may be of independent interest.

The fixed-parameter tractability of Bipartite Independent Set Reconfiguration restricted to bipartite circle graphs, with respect to ρ, Φ and $\rho + \Phi$ remains open. It implies that the fixed-parameter tractability of directed pathwidth (i.e. ρ for BISR on general instances) also remains open. We nevertheless hope that this newly-drawn connection between a width parameter (directed pathwidth and a reconfiguration problem (BISR), may help shed new light onto this problem. In that respect, combining [31] and [11] to try to formulate an "obstacle theory" for directed pathwidth might be an interesting avenue.

Appendix A Mixed MIS in bipartite graphs

Our Divide-and-Conquer strategy to the BISR problem relies on the computation of maximum independent sets containing at least one vertex in each part of the input bipartite graph.

We informally call *mixed bipartite maximum independent set* (Mixed-MIS) the problem of deciding whether an input bipartite graph G has a maximum independent set intersecting both of its parts. It is trivially polynomial, as one may check for each pair $(l, r) \in L \times R$, whether $I' \cup \{l, r\}$ is a maximum independent set of G; with I' maximum independent set of G', and G' obtained from G by removing l, r as well as their neighborhoods.

As a maximum independent set of a bipartite graph may be derived from a maximum matching, this simple strategy yield a $O(|V|^2 \cdot \sqrt{|V||E|})$ algorithm for our Mixed-MIS problem.

We present here a more efficient strategy, based on a decomposition taking place in two rounds. It results into Algorithm 4. The first round is based on the Dulmage-Mendelsohn decomposition of bipartite graphs. It yields a partition of the vertices of G into three sets D, A, C, defined as such: for each vertex v of D, there exists a maximum matching in which v is not matched, $A = N(D)$ is the union of the neighborhoods of the vertices of D, and $C = V \setminus (D \cup A)$ contains the remaining vertices. D, A, C verify the following result:

Theorem 8 (Dulmage-Mendelsohn decomposition, Proposition 2.1 of [30], theorem 3.2.4 of [29]) *Given G bipartite graph and D, A, C defined as above, we have that:*

a. – *D is the intersection of all maximum independent sets of G.*
   – *A is the intersection of all minimum vertex covers of G.*
   – *the subgraph $G[C]$ induced by C has a perfect matching, which may be deduced from restricting any maximum matching of G to C.*

b. *In addition, D may be computed from any maximum matching M of G using the following characterization ([30], lemma 2.2): $D = \overline{W}$ where $\overline{W}$ is composed of the vertices left unmatched by M, as well as all vertices connected to an unmatched vertex through an alternating path of even length.*

This decomposition may allow to conclude in some cases (see Algorithm 4). In general, however, a second round of decomposition is needed. In this second round, the set C, which allows for a perfect matching M, is further decomposed into *elementary sub-graphs* (section 4.1 of [29], theorem 4.1.1 and exercise 4.1.5) and [32]. It consists in computing the strongly connected components of a directed graph $H(M, C)$ associated to M and C (same construction as in Section 3). The vertices of H are the edges of the matching, and $(l, r) \to (l', r')$ iff l is connected to r' in C. The strongly connected components of H constitute a decomposition of G into elementary sub-graphs. A bipartite graph is elementary iff the sides L, R are the only minimum vertex covers/maximum independent sets [29](theorem 4.1.1). If it is not elementary, then a mixed maximum independent set may be obtained by ordering the elementary sub-graphs $\{(L_i, R_i)\}_{1 \leq i \leq p}$ along a topological order induced by $H(C, M)$. Any set of the form $(\cup_{i \leq t} R_i) \cup (\cup_{i > t} L_i)$ for some $t > 1$ is then a mixed maximum independent set of C.

The discussion above results in Algorithm 4, whose run-time is dominated by the computation of maximum matching in $O(\sqrt{|V|}|E|)$.

Appendix B Delayed proofs

B.1 Making an interval representation *nice*

Let $\{(a_u, b_u) \mid u \in V\}$ be an interval representation for a directed graph H with vertex set V. We explain here how to turn it into a nice interval representation:

If an integer n is such that $a_{u_0} = \cdots = a_{u_l} = b_{v_0} = \cdots = b_{v_p} = n$, we may modify the representation as such:

- Interval bounds associated to integers $> n$ are increased by $p + l - 1$, to make room for "spreading" $a_{u_1} \ldots a_{u_\ell}, b_{v_1} \ldots b_{v_p}$.
- $\forall i$, a_{u_i} is set to $n + i$ and b_{v_i} to $l + i$.

None of these modifications change the way intervals intersect one another, leaving the width unchanged. The representation is then "packed" into

Algorithm 4 Mixed bipartite maximum independent set

Input: a bipartite graph G with sides L and R. We suppose w.l.o.g that $|L| \geq |R|$.

Output: If it exists, a Maximum Independent Set I of G intersecting both L and R.

M = MaximumMatching(G) ▷ $O(\sqrt{|V|} \cdot |E|)$

I = MaximumIndependentSet(G, M) ▷ $O(|E|)$
if $(I \cap L \neq \emptyset)$ ***and*** $(I \cap R \neq \emptyset)$ **then**
  return I
end if ▷ // Now $|I| = \max(|L|, |R|)$ and $I = L$ or $I = R$

D, A, C = coarseDulmageMendelsohn(M, G) ▷ $O(|E|)$
if $|L| > |R|$ **then**
  if $R \setminus A \neq \emptyset$ **then**
    ▷//A is the intersection of all minimum vertex covers
    pick $r \in R \setminus A$
    $G' = G \setminus \{r \cup N(r)\}$
    M' = MaximumMatching(G')
    I' = MaximumIndependentSet(G', M')
    return $I' \cup \{r\}$
  else
    return $\perp$; ▷ // Not possible, L is the only MIS
  end if
else if $|L| = |R|$ **then**
  ▷ // L and R are two MIS. So necessarily $D = \emptyset$, $A = \emptyset$, $C = G$
  $\{(L_i, R_i)\}_{1 \leq i \leq p}$ = fineDulmageMendelsohn(M, C) ▷ $O(|V|^2)$
  if p=1 **then**
    return $\perp$
  else
    ▷ Topological sort of the SCCs of H
    s=TopologicalSort($\{(L_i, R_i)\}$) ▷ $O(|V| + |E|)$
    $(L_i, R_i) = s[0]$ ▷ // first in topological sort
    return $R_i \cup (\cup_{j \neq i} L_j)$
  end if
end if

$[1 \ldots 2.|V(H)|]$ by taking the interval bounds in order and setting them to their final position.

B.2 Proof of Proposition 2:

Appendix C Re-derivation of Tamaki's algorithm for directed pathwidth

For completeness, we include here a re-derivation of the results of [1], with the slight modification mentioned in the main text related to *pruning*. It results in an algorithm with a $O(n^{\rho+2})$ complexity, slightly different from the $O(n^{\rho+1})$ announced in [1]. The re-derivation follows the same strategy as in the original article, and re-uses most of the notations.

C.1 Commitment lemma - shortest non-expanding extensions (SNEKFEs)

Notations and definitions. In a directed graph, $d^-(u)$ denotes the in-degree of a node u. We work with layouts of vertices, i.e. ordered sequences of vertices, not necessarily containing all vertices. A partial layout σ is called *feasible/valid* if $\forall$ prefix p of σ we have $d^-(p) = |N^-(p)| \leq k$. A partial layout which is *completable* into a valid full layout (for the entire digraph G) is called *strongly feasible* or just *completable into a full solution*. An *extension* τ of σ is a valid partial layout with σ as one of its prefixes. A *shortest non-expanding extension* of σ is an extension τ such that $d^-(\tau) \leq d^-(\sigma)$ and $\forall \rho$ s.t. $V(\sigma) \subsetneq V(\rho) \subsetneq V(\tau)$, $d^-(\rho) > d^-(\sigma)$. In the rest of this note, we will write SNEKFE for shortest non-expanding extension.

Lemma 1 - Commitment Lemma - shortest non-expanding extensions. *If σ is completable into a full solution, and τ is a SNEKFE of σ, then τ is also completable into a full solution.*

In fact, a more general version is true: ρ could be allowed to be equal in d^- to τ before rising again. The proof relies on the fact that, for any two subsets X, Y of vertices of G:

$$d^-(X \cup Y) + d^-(X \cap Y) \leq d^-(X) + d^-(Y)$$

Proof If σ is completable into a full solution, then $\exists F$ such that $\sigma \cdot F$ is a valid layout for G. Let us reshuffle F into $(\tau \setminus \sigma) \cdot F'$. Within both parts, the ordering of elements is the same as in F. $\tau \cdot F'$ is now a complete layout for G. Is it valid ?

Consider a prefix P of $\tau \cdot F'$. If P is contained within τ, $d^-(P) \leq k$ by the validity of τ.

Else, if P contains some of F', then $P = P' \cup \tau$ for P' a certain prefix of $\sigma \cdot F$. As for $P' \cap \tau$, which we call ρ it verifies $V(\sigma) \subset V(\rho) \subset V(\tau)$ and therefore $d^-(\rho) \geq d^-(\sigma) \geq d^-(\tau)$ by definition of a SNEKFE, with the equality only potentially happening if $\rho = \sigma$ or $\rho = \tau$.

We therefore have:

$$\begin{aligned} d^-(P) &= d^-(P' \cup \tau) \\ &\leq d^-(P') + d^-(\tau) - d^-(\rho) \end{aligned}$$

$$\leq d^-(P') \leq k$$

$\tau \cdot F'$ is therefore a valid complete layout for G, and τ is completable into a full solution. □

Let us now describe more precisely what SNEKFEs might look like. We show that they can only be of three types, and formalize it into the next lemma. Its proof relies on the fact that, by adding a single vertex u to a partial layout σ, we may only decrease $d^-(\sigma)$ by at most 1, since $d^-(\sigma) = |N^-(\sigma)|$. We obtain this decrement of 1 if u is a predecessor to a vertex of σ, and does not introduce any new predecessor itself when added.

Lemma 2 - SNEKFE types. *a SNEKFE τ of a partial layout σ may only be of three types:*

- *type-(i): single-vertex "decreasing" extension: $\tau = \sigma \cdot u$ for some vertex u and $d^-(\sigma \cdot u) = d^-(\sigma) - 1$*
- *type-(ii): single-vertex "non-decreasing" extension: $\tau = \sigma \cdot u$ for some vertex u and $d^-(\sigma \cdot u) = d^-(\sigma)$*
- *type-(iii): several vertices "shortcut" extension: τ adds strictly more than one vertex to σ and $d^-(\tau) = d^-(\sigma)$.*

Proof For single vertex extensions, the two possible types follow from the observation above that the addition of one vertex to a layout can only decrease d^- by at most 1.

For SNEKFEs composed of more than one vertex, observe that if $d^-(\tau) < d^-(\sigma)$, then by considering the prefix ρ of τ obtained by removing just 1 vertex to τ, we would have $d^-(\rho) \leq d^-(\tau)+1 \leq d^-(\sigma)$. This stems from the observation above that d^- may only decrease by at most 1 when adding a vertex. ρ would be a non-expanding extension of σ shorter than τ, yielding a contradiction. □

C.2 Algorithm

In this section, we restrict ourselves to a pure description of the algorithm, delaying the justification of its correctness and complexity to the "Analysis" section below.

Tree of prefixes (trie). We will build a tree of prefixes of all possible layouts. We prune the tree during its construction thanks to the commitment lemma, as justified in the next section. We call S_i the i^{th} level of the tree of prefixes. I.e. the elements of the tree of length i. $S_0 = \{\emptyset\}$.

Algorithm. S_{i+1} is generated in the following way given S_i:

For each $\sigma \in S_i$:

1. We generate all *feasible immediate extensions* to σ and add them to the tree. I.e the node σ now has the following children set: $\{\sigma \cdot u \quad s.t \quad d^-(\sigma \cdot u) \leq k\}$
2. If some of these immediate extensions verify $d^-(\sigma \cdot u) \leq d^-(\sigma)$, then they are SNEKFEs of σ. In that case, we do the following:

   a. We choose 1 arbitrarily and prune the others.

b. If the chosen element verifies $d^-(\sigma \cdot u) = d^-(\sigma) - 1$ (the only possibility if $d^-(\sigma \cdot u) < d^-(\sigma)$), then we in addition look for a prefix η of σ verifying $d^-(\eta) = d^-(\sigma \cdot u)$ and $d^-(\rho) > d^-(\eta)$ $\forall \rho$ s.t. $\eta \sqsubseteq \rho \sqsubseteq \sigma \cdot u$, $\rho \neq \eta$, $\rho \neq \sigma \cdot u$. If such an η is found, then any part of tree branching off the path from η to $\sigma \cdot u$ is removed. Note that this might shorten the overall loop over $\sigma \in S_i$.

End Algorithm

C.3 Analysis

This section will be composed of three parts. In the first one, we define an invariant property ("internally pruned") for trees of prefixes of layouts of vertices. In the second one, we show that, in the algorithm presented in the previous section, the tree of prefixes verifies the invariant at all times, and prove the correctness of the algorithm. Finally, in the third part, we analyze the size of trees of prefixes verifying the invariant, proving that each level S_i of such a tree has a size $\leq n^k$, yielding a complexity analysis of the algorithm.

C.3.1 Internally pruned trees of prefixes

Definition - Internally pruned. *A tree $\mathcal{T}$ of prefixes of layouts of vertices (such as the one used in the algorithm in the previous section) is said to be internally pruned if for all pairs (σ, τ) of nodes of $\mathcal{T}$ such that τ is a shortest non-expanding extension of σ, all nodes on the path from τ (included) to σ (excluded) in $\mathcal{T}$ have degree exactly 2. I.e. there are no sub-parts of the tree rooted on the path from τ (included) to σ (excluded)*

We use the term "internally" to emphasize the fact that, in a context where we apply the definition of "internally pruned" to a partially constructed $\mathcal{T}$ within the algorithm of the previous section, More ("external") pruning of the tree might be achieved further in the construction of the tree, as new SNEKFEs are discovered (see below for the justification of why new SNEKFEs are indeed discovered at step 2.b of the algorithm).

C.3.2 Invariant and correctness

Lemma 3 - Invariant. *Throughout the execution of the algorithm presented in the previous section, the tree $\mathcal{T}$ of prefixes of layouts of vertices remains "internally pruned" at all times*

Proof The tree $\mathcal{T}$ starts off with one node for the empty sequence. It is therefore internally pruned.

Suppose now that the tree of prefixes $\mathcal{T}$ is internally pruned at an intermediate step in the algorithm, then the next building step always consists in considering a leaf σ and executing step 1. and 2. of the algorithm. Several cases may arise:

- If all of the immediate extensions are such that $\{d^-(\sigma) < d^-(\sigma \cdot u) \leq k\}$, then no new SNEKFEs are generated when adding them to the tree. (if $\sigma \cdot u$ is a SNEKFE of some η up the tree, then σ is shorter and also non-expanding). After the addition of the immediate extension, the tree is therefore still internally pruned.
- If one of these immediate extensions verifies $d^-(\sigma \cdot u) = d^-(\sigma)$ but none of them verify $d^-(\sigma \cdot u) < d^-(\sigma)$, then one of these extensions is a SNEKFE of σ, and is kept while the others are pruned. However, this is the only SNEKFE introduced by the extension. Therefore, the pruning of immediate extensions other than the selected one is enough to keep the tree internally pruned.
- If one of the immediate extensions verifies $d^-(\sigma \cdot u) = d^-(\sigma) - 1$, then one of the immediate extensions is selected and the others are pruned, as in the previous case. However, in addition, $\sigma \cdot u$ might be a new shortest non-expanding extension of a node η up the tree.
  If this is the case, then there is only one such η, per the definition of shortest non-expanding extensions.
  We argue that the conditions used in the algorithm indeed detect such an η.
  If $\sigma \cdot u$ is a SNEKFE of η, then the conditions described in the algorithm (that $d^-(\sigma \cdot u) = d^-(\eta)$, and $d^-(\rho) > d^-(\eta)$ for any ρ on the path from η to $\sigma \cdot u$) are verified.
  Conversely, if the conditions are verified, then suppose η has a shorter non-expanding extensions τ. τ cannot be on the path from η to $\sigma \cdot u$ as that would imply $d^-(\tau) > d^-(\eta)$. Since τ is shorter than $\sigma \cdot u$, τ has been generated in a previous step of the algorithm. At this point, step 2.*b* of the algorithm would have pruned the path to σ, which cannot be visited, leading to a contradiction.

  Therefore, the potentially newly introduced SNEKFE is detected, and the corresponding pruning is carried out, leaving the tree internally pruned

Therefore, after each extension of the tree throughout the algorithm, the tree remains internally pruned. □

We quickly finish this sub-section with a proof of correctness of the algorithm.

Lemma 4 - correctness. *If the graph G allows for a full k-feasible solution, then there is such a solution among the leaves of the tree of prefixes $\mathcal{T}$ generated by the algorithm.*

Proof Denote the set of full solutions S, and suppose all solutions are absent from $\mathcal{T}$.
$\forall \sigma \in S$, there is some (possibly empty) prefix of σ in $\mathcal{T}$.
We pick $\sigma \in S$ allowing for the largest prefix $\eta \in \mathcal{T}$, i.e:

$$\sigma = \underset{\sigma' \in S}{\operatorname{argmax}} \left[\max_{\eta \sqsubseteq \sigma', \eta \in \mathcal{T}} |\eta| \right]$$

Take η the largest prefix of σ belonging to $\mathcal{T}$. If the path from η to σ has been pruned, it is because η is on the path from η' to τ, with τ shortest non expanding extension of η', and τ is not a prefix of σ.

The path from η to σ is pruned only when τ is visited. Hence $\tau \in \mathcal{T}$, otherwise, the path from

Per the commitment lemma, τ is the prefix of a full solution σ''. But $|\tau| > |\eta|$, contradicting the choice of σ. □

C.3.3 Signature analysis

We show here that, at any point in the algorithm, thanks to the pruning, $\forall i, \quad |S_i| = O(n^k)$.

Definition - signature . Consider $\sigma \in S_i$ for some i, within the internally pruned tree generated by the algorithm, valid partial layout. We call *signature* of σ the set of vertices obtained from $V(\sigma)$ by removing, given any pair (η, ρ) of prefixes of σ such that ρ is a SNEKFE of η, all vertices in $\rho \setminus \eta$.

Given $\sigma \in S_i$, its signature can be easily computed by looking at the path from the root to σ: any vertex chosen out of several available possibilities is part of the signature, while any vertex that was the only possibility at the point of its choosing isn't.

Lemma 5 - Same signature same sequence. *If $sgn(\sigma) = sgn(\tau)$ within the pruned tree of layouts and $|\tau| = |\sigma|$ then $\sigma = \tau$*

Proof When starting at the root and building τ and σ by going down the tree, at every node, there are two cases:

- Either the next move is part of a SNEKFE. In this case there are no choices to be made, the added vertex is not part of the signature, and is the same for σ and τ.
- Or the next move is not part of a SNEKFE. In this case, several choices are possible, and the next added vertex will be part of the signature. Since the signatures of σ and τ are the same, the same vertex is added to σ and τ.

At the end of this process, σ and τ are therefore identical. □

Lemma 6 - overall strictly decreasing = SNEKFE only. *Consider $\tau \in S_i$ for some i partial valid layout, and σ a prefix of τ such that:*

- $d^-(\sigma) > d^-(\tau)$
- *For any ρ such that $\sigma \sqsubseteq \rho \sqsubseteq \tau$, $\rho \neq \tau$, we also have $d^-(\rho) > d^-(\tau)$.*

Then, the suffix $\tau \setminus \sigma$ of τ corresponding to σ can be entirely partitioned into SNEKFEs. In particular, none of its elements are part of the signature of τ.

Proof We prove the lemma by induction on the length of the suffix $\tau \setminus \sigma$. If $|\tau \setminus \sigma| = 1$, then $\tau = \sigma \cdot u$ and $d^-(\tau) = d^-(\sigma) - 1$. τ is a type-(i) SNEKFE of σ and the lemma is true.

If $|\tau \setminus \sigma| > 1$ and we assume the lemma true $\forall l < |\tau \setminus \sigma|$, then let us distinguish two cases related to the first element v of $\tau \setminus \sigma$:

- if $\sigma \cdot v$ is a type-(i) or type-(ii) SNEKFE of σ, then we apply the induction hypothesis to the suffix $\tau \setminus (\sigma \cdot v)$ of τ and we have the result.

- else, if $d^-(\sigma \cdot v) > d^-(\sigma)$, we know, since $d^-(\tau) < d^-(\sigma)$ and the d^--curve only decreases by steps of -1, that there must exist ρ such that $d^-(\rho) = d^-(\sigma)$, $\sigma \sqsubseteq \rho \sqsubseteq \tau$, and $d^-(\rho') > d^-(\sigma)$ for any ρ' such that $\sigma \sqsubseteq \rho' \subseteq \rho$ (ρ is the shortest prefix of τ which contains σ and has the same d^- value). ρ is then a type-(iii) SNEKFE of σ by Lemma 4, and we may apply the induction hypothesis to $\tau \setminus \sigma$

□

Lemma 7 - Signature size. $\forall \sigma \in S_i$ *for some* i *partial layout of vertices,* $|sgn(\sigma)| \leq d^-(\sigma)$

Proof The proof is by induction on $|\sigma|$. Suppose $|sgn(\sigma)| \leq d^-(\sigma)$, and consider the extension $\sigma \cdot u$, where u is a vertex.

- If $\sigma \cdot u$ is not a SNEKFE of σ, then $|sgn(\sigma \cdot u)| = |sgn(\sigma) \cup \{u\}| = \mathsf{sgn(u)} + 1 \leq d^-(\sigma) + 1 \leq d^-(\sigma \cdot u)$
- If σ is a type-(ii) SNEKFE of σ, then $sgn(\sigma) = sgn(\sigma \cdot u)$ and $d^-(\sigma \cdot u) = d^-(\sigma)$.
- If $\sigma \cdot u$ is a type-(i) SNEKFE of σ, then consider η, the closest node (up the tree) such that $d^-(\eta) < d^-(\sigma \cdot u)$, and $\eta \cdot v$ its successor on the path to $\sigma \cdot u$. We have $d^-(\eta) < d^-(\sigma \cdot u) \leq d^-(\eta \cdot v)$, by definition of η. The path from $\eta \cdot v$ to u is either a type-(iii) SNEKFE or overall-decreasing. Therefore $sgn(\sigma \cdot u) = sgn(\eta \cdot v)$. and $|sgn(\sigma \cdot u)| = |sgn(\eta)| + 1 \leq d^-(\eta) + 1$ by induction hypothesis, and $|sgn(\sigma \cdot u)| \leq d^-(\sigma \cdot u)$.

□

In particular, $\forall \sigma$ partial layout, $d^-(\sigma) \leq k$. Since two different elements of S_i need different signatures, we get the following corollary:

Corollary. $\forall i$, *at any point in the algorithm,* $|S_i| = O(n^k)$

The overall complexity of the algorithm is therefore $O(n^{k+O(1)})$. More precisely, it is $O(n^{k+2})$. (there are n levels of the tree to fill, $\leq n^k$ nodes per level and $O(n)$ work per node to generate the next level).

Appendix D Detailed RNA reconfiguration example

We provide in Figure D1 the intermediate sets of base pairs, and associated RNA secondary structures, for our running example, described in Figures 1 and **??**.

References

[1] Tamaki, H.: A polynomial time algorithm for bounded directed pathwidth. In: Kolman, P., Kratochvíl, J. (eds.) Graph-Theoretic Concepts in Computer Science, pp. 331–342. Springer, Berlin, Heidelberg (2011)

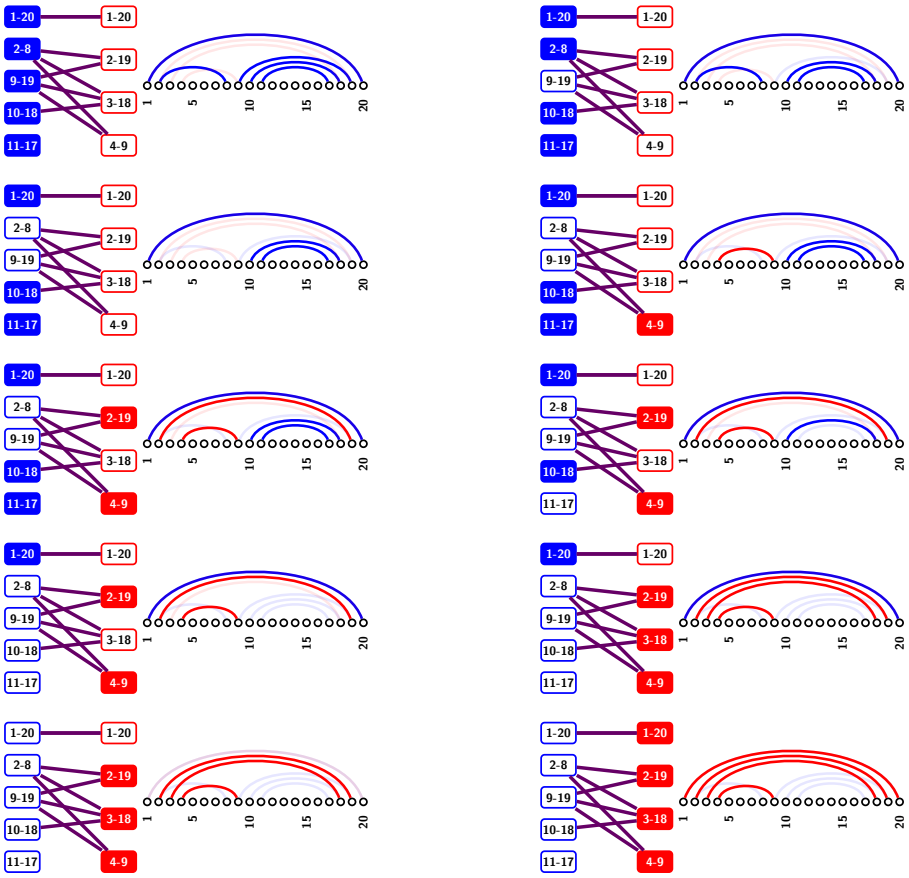


Fig. D1 Optimal (min barrier) refolding scenario between two RNA secondary structures. In each intermediate state, the conflict graph is given, featuring the selected independent set of base pairs (filled nodes), and the corresponding secondary structure.

[2] van den Heuvel, J.: The complexity of change. Surveys in combinatorics **409**(2013), 127–160 (2013)

[3] Lokshtanov, D., Mouawad, A.E.: The complexity of independent set reconfiguration on bipartite graphs. ACM Transactions on Algorithms (TALG) **15**(1), 1–19 (2018)

[4] Ito, T., Kamiński, M., Ono, H., Suzuki, A., Uehara, R., Yamanaka, K.: Parameterized complexity of independent set reconfiguration problems. Discrete Applied Mathematics **283**, 336–345 (2020)

[5] Mouawad, A.E., Nishimura, N., Raman, V., Simjour, N., Suzuki, A.: On the parameterized complexity of reconfiguration problems. Algorithmica **78**(1), 274–297 (2017)

[6] Lokshtanov, D., Mouawad, A.E., Panolan, F., Ramanujan, M., Saurabh, S.: Reconfiguration on sparse graphs. Journal of Computer and System Sciences **95**, 122–131 (2018)

[7] Lokshtanov, D., Mouawad, A.E., Panolan, F., Siebertz, S.: On the parameterized complexity of reconfiguration of connected dominating sets. arXiv preprint arXiv:1910.00581 (2019)

[8] Barát, J.: Directed path-width and monotonicity in digraph searching. Graphs and Combinatorics **22**(2), 161–172 (2006)

[9] Yang, B., Cao, Y.: Digraph searching, directed vertex separation and directed pathwidth. Discrete Applied Mathematics **156**(10), 1822–1837 (2008)

[10] Coudert, D., Mazauric, D., Nisse, N.: Experimental evaluation of a branch-and-bound algorithm for computing pathwidth and directed pathwidth. Journal of Experimental Algorithmics (JEA) **21**, 1–23 (2016)

[11] Erde, J.: Directed path-decompositions. SIAM Journal on Discrete Mathematics **34**(1), 415–430 (2020)

[12] Kitsunai, K., Kobayashi, Y., Komuro, K., Tamaki, H., Tano, T.: Computing directed pathwidth in $o(1.89^n)$ time. Algorithmica **75**(1), 138–157 (2016)

[13] Bodlaender, H.L.: Fixed-parameter tractability of treewidth and pathwidth. In: The Multivariate Algorithmic Revolution and Beyond, pp. 196–227. Springer, ??? (2012)

[14] Tamaki, H.: A directed path-decomposition approach to exactly identifying attractors of boolean networks. In: 2010 10th International Symposium on Communications and Information Technologies, pp. 844–849 (2010). IEEE

[15] Kobayashi, Y., Komuro, K., Tamaki, H.: Search space reduction through commitments in pathwidth computation: An experimental study. In: International Symposium on Experimental Algorithms, pp. 388–399 (2014). Springer

[16] Tinoco Jr, I., Bustamante, C.: How rna folds. Journal of molecular biology **293**(2), 271–281 (1999)

[17] Maňuch, J., Thachuk, C., Stacho, L., Condon, A.: Np-completeness of the energy barrier problem without pseudoknots and temporary arcs. Natural Computing **10**(1), 391–405 (2011)

[18] Thachuk, C., Manuch, J., Rafiey, A., Mathieson, L.-A., Stacho, L., Condon, A.: An Algorithm for the Energy Barrier Problem Without Pseudoknots and Temporary Arcs. In: Biocomputing 2010, pp. 108–119. World Scientific, ??? (2009). https://doi.org/10.1142/9789814295291_0013

[19] Gottschau, M., Happach, F., Kaiser, M., Waldmann, C.: Budget minimization with precedence constraints. CoRR **abs/1905.13740** (2019) arXiv:1905.13740

[20] Kinne, J., Manuch, J., Rafiey, A., Rafiey, A.: Ordering with precedence constraints and budget minimization. CoRR **abs/1507.04885** (2015) arXiv:1507.04885

[21] Cygan, M., Fomin, F.V., Kowalik, Ł., Lokshtanov, D., Marx, D., Pilipczuk, M., Pilipczuk, M., Saurabh, S.: Parameterized Algorithms vol. 5. Springer, ??? (2015)

[22] Abdel-Wahab, H.: On strictly optimal schedules for the cumulative cost-optimal scheduling problem. Computing **24**, 61–86 (1980)

[23] Wolfinger, M.T., Svrcek-Seiler, W.A., Flamm, C., Hofacker, I.L., Stadler, P.F.: Efficient computation of rna folding dynamics. Journal of Physics A: Mathematical and General **37**(17), 4731 (2004)

[24] Senter, E., Clote, P.: Fast, approximate kinetics of rna folding. Journal of Computational Biology **22**(2), 124–144 (2015)

[25] Fukuda, K., Matsui, T.: Finding all the perfect matchings in bipartite graphs. Applied Mathematics Letters **7**(1), 15–18 (1994)

[26] Zhang, Z., Zhang, X., Wen, X.: Directed hamilton cycles in digraphs and matching alternating hamilton cycles in bipartite graphs. SIAM J. Discret. Math. **27**(1), 274–289 (2013). https://doi.org/10.1137/110837188

[27] Tamaki, H.: A polynomial time algorithm for bounded directed pathwidth. In: International Workshop on Graph-Theoretic Concepts in Computer Science, pp. 331–342 (2011). Springer

[28] Rafiey, A., Kinne, J., Manuch, J., Rafiey, A.: Ordering with precedence constraints and budget minimization. arXiv preprint arXiv:1507.04885 (2015)

[29] Lovász, L., Plummer, M.D.: Matching Theory vol. 367. American Mathematical Soc., ??? (2009)

[30] Chen, J., Kanj, I.A.: Constrained minimum vertex cover in bipartite graphs: complexity and parameterized algorithms. Journal of Computer

and System Sciences **67**(4), 833–847 (2003)

[31] de Berg, M., Jansen, B.M., Mukherjee, D.: Independent-set reconfiguration thresholds of hereditary graph classes. Discrete Applied Mathematics **250**, 165–182 (2018)

[32] Zhang, Z.-B., Lou, D.: Bipartite graphs with a perfect matching and digraphs. arXiv preprint arXiv:1011.4359 (2010)