



HAL
open science

A new parametrization for independent set reconfiguration and applications to RNA kinetics

Laurent Bulteau, Bertrand Marchand, Yann Ponty

► To cite this version:

Laurent Bulteau, Bertrand Marchand, Yann Ponty. A new parametrization for independent set reconfiguration and applications to RNA kinetics. IPEC 2021 - International Symposium on Parameterized and Exact Computation, Sep 2021, Lisbon, Portugal. hal-03272963v1

HAL Id: hal-03272963

<https://hal.science/hal-03272963v1>

Submitted on 28 Jun 2021 (v1), last revised 6 Oct 2021 (v2)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

A new parametrization for independent set reconfiguration and applications to RNA kinetics

Laurent Bulteau  

LIGM, CNRS, Univ Gustave Eiffel, F77454 Marne-la-vallée France

Bertrand Marchand   

LIX, CNRS UMR 7161, Ecole Polytechnique, Institut Polytechnique de Paris, France

LIGM

Yann Ponty   

LIX, CNRS UMR 7161, Ecole Polytechnique, Institut Polytechnique de Paris, France

Abstract

In this paper, we study the Independent Set (IS) reconfiguration problem in graphs. An IS reconfiguration is a scenario transforming an IS L into another IS R , inserting/removing vertices one step at a time while keeping the cardinalities of intermediate sets greater than a specified threshold. 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* problem from bioinformatics, for which a natural parameter would be the difference between the initial IS size and the threshold.

We first show the para-NP hardness of the problem with respect to this parameter. We then investigate a new parameter, the *cardinality range*, denoted by ρ which captures the maximum deviation of the reconfiguration scenario from optimal sets (formally, ρ is the maximum difference between the cardinalities of an intermediate IS and an optimal IS). We give two different routes to show that this problem is in XP for ρ : The first is a direct $O(n^2)$ -space, $O(n^{2\rho+2.5})$ -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 [20]. 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.

We demonstrate the practicality of these algorithms, and the relevance of our introduced parameter, by considering the application of our algorithms on random small-degree instances for our problem. Moreover, we reformulate the computation of the energy barrier between two RNA secondary structures, a classic hard problem in computational biology, as an instance of bipartite reconfiguration. Our results on IS reconfiguration thus yield an XP algorithm in $O(n^{\rho+2})$ for the energy barrier problem, improving upon a partial $O(n^{2\rho+2.5})$ algorithm for the problem.

2012 ACM Subject Classification Theory of computation \rightarrow Parameterized complexity and exact algorithms ; Applied computing \rightarrow Bioinformatics

Keywords and phrases reconfiguration problems - parameterized algorithms - RNA bioinformatics - directed pathwidth

Digital Object Identifier 10.4230/LIPIcs.IPEC.2021.YY

Acknowledgements The authors would like to thank H. Tamaki and Y. Kobayashi for fruitful email exchanges.

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



© Laurent Bulteau, Bertrand Marchand and Yann Ponty;
licensed under Creative Commons License CC-BY 4.0

16th International Symposium on Parameterized and Exact Computation (IPEC 2021).



Leibniz International Proceedings in Informatics

LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

44 of these problems, configurations correspond to *feasible solutions* of some *optimization*
 45 *problem*, and a feasible solution is legal when its quality is higher than a specified threshold.

46 Examples of optimization problems for which reconfiguration versions have been studied
 47 include DOMINATING SET, VERTEX COVER, SHORTEST PATH or INDEPENDENT SET, which
 48 is our focus in this article. Associated complexities range from polynomial (see [23] for
 49 examples) to NP-complete (for bipartite independent set reconfiguration [13]), and even
 50 PSPACE-complete for many of them [13, 9]. Such computational hardness motivates the
 51 study of these problems under the lens of *parametrized complexity* [18, 14, 15, 9], in the
 52 hope of identifying tractable sub-regimes. Typical parameters considered by these studies
 53 focus on the value of the *quality threshold* (typically a *solution size* bound) defining legal
 54 configurations and the length of the reconfiguration sequences.

55 **Directed pathwidth.** *Directed pathwidth*, originally defined in [1] and attributed to
 56 Robertson, Seymour and Thomas, represents a natural extension of the notions of path-
 57 width and path decompositions to directed graphs. Like its undirected restriction, it may
 58 alternatively be defined in terms of *graph searching* [24], *path decompositions* [4, 6] or *vertex*
 59 *separation number* [11, 20]. An intuitive formulation can be stated as the search for a visit
 60 order of the directed graph, using as few active vertices as possible at each step, and such that
 61 no vertex may be deactivated until all its in-neighbors have been activated. Although an FPT
 62 algorithm is known for the undirected pathwidth [2], it remains open whether computing
 63 the directed pathwidth admits a FPT algorithm. XP algorithms [20, 11] are known, and have
 64 been implemented in practice [19, 12].

65 **RNA energy barrier.** RNAs are single-stranded biomolecules, which fold onto themselves
 66 into 2D and 3D structures through the pairing of nucleotides along their sequence [22].
 67 Thermodynamics then favors low-energy structures, and the RNA energy barrier problem
 68 asks, given two structures, whether there exists a re-folding pathway connecting them that
 69 does not go through unlikely high-energy intermediate states [17, 21]. Interestingly, the
 70 problem falls under the wide umbrella of reconfiguration problems described above, namely
 71 the reconfiguration of solutions of optimization problems (here, energy minimization). An
 72 important specificity of the problem is that the probability of a refolding pathway depends
 73 on the energy difference between intermediate states and the *starting point* rather than the
 74 absolute energy value. Another aspect of this problem is that, since some pairings of the
 75 initial structure may impede the formation of new pairings for the target structure, it induces
 76 a notion of *precedence constraints*, and may therefore also be treated as a *scheduling* problem,
 77 as carried out in [8, 10].

78 **Problem statement.** In our work, we focus on independent set reconfigurations where
 79 only vertices from the start or end ISs (L and R) are allowed within intermediate ISs. This
 80 amounts to considering the induced subgraph $G[L \cup R]$, bipartite by construction. We write
 81 $\alpha(G)$ for the size of a maximum independent set of G (recall that $\alpha(G)$ can be computed in
 82 polynomial time on bipartite graphs).

BIPARTITE INDEPENDENT SET RECONFIGURATION (BISR)

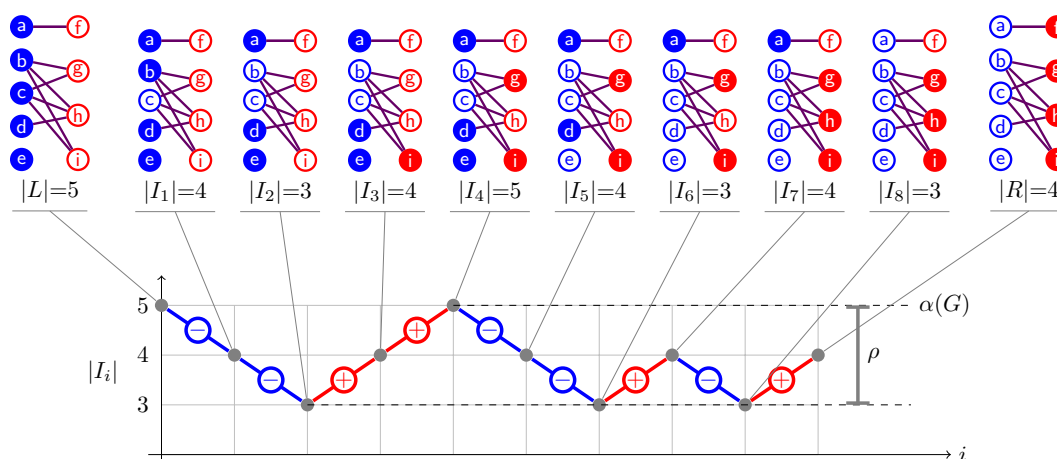
Input: Bipartite graph $G = (V, E)$ with partition $V = L \cup R$; integer ρ

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.



■ **Figure 1** Example of a bipartite independent set reconfiguration from vertices in 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.

83 Figure 1 shows an example of an instance of BISR and a possible reconfiguration pathway.
 84 We introduce the *cardinality range* (or simply *range*) $\rho = \max_{1 \leq i \leq \ell} \alpha(G) - |I_i|$ as a natural
 85 parameter for this problem, since it measures a distance to optimality. As mentioned above,
 86 the related parameter in RNA reconfiguration is the *barrier*, denoted k , and defined as
 87 $k = \max_{1 \leq i \leq \ell} |L| - |I_i|$. Intuitively, k measures the size difference from the starting point
 88 rather than from an “absolute” optimum. Note that $k = \rho - (\alpha(G) - |L|)$, so one has
 89 $0 \leq k \leq \rho$. Both parameters are obviously similar for instances where L is close to being a
 90 maximum independent set, which is generally the case in RNA applications, but in theory
 91 the range ρ can be arbitrarily larger than the barrier k .

92 **Our results.** We first prove that in general, the barrier k may not yield any interesting
 93 parameterized algorithm, since BISR is Para-NP-hard for this parameter. We thus focus on
 94 the range parameter for BIPARTITE INDEPENDENT SET RECONFIGURATION, and prove that
 95 it is in XP by providing two distinct algorithmic strategies to tackle it.

96 Our first algorithmic strategy stems from a parameterized equivalence we draw between
 97 BISR and the problem of computing the directed pathwidth of directed graphs. Within this
 98 equivalence, the range parameter ρ maps exactly to the directed pathwidth. This allows to
 99 apply XP algorithms for DIRECTED PATHWIDTH to BISR while retaining their complexity,
 100 such as the $O(n^{\rho+2})$ -time, $O(n^{\rho+1})$ -space algorithm from Tamaki [20] (with $n = |V|$). This
 101 equivalence between directed pathwidth and bipartite independent set reconfiguration is itself
 102 an interesting result, as it connects a *structural* problem, whose parameterized complexity is
 103 open, with a reconfiguration problem of the kind that is routinely studied in parameterized
 104 complexity [18, 14, 15, 9].

105 We also present another more direct algorithm for BISR, with a time complexity of
 106 $O(n^{2\rho} \sqrt{nm})$ (with $m = |E|$) but using only $O(n^2)$ space. It relies on a separation lemma
 107 involving, if it exists, a *mixed* maximum independent set of G containing at least one vertex
 108 from both parts of the graph. In the specific case of bipartite graphs arising from RNA
 109 reconfiguration, we improve the run-time of the subroutine computing a mixed MIS to $O(n^2)$
 110 (rather than $O(\sqrt{nm})$), with a dynamic programming approach.

111 We present benchmark results for both algorithms, on random instances of general
 112 bipartite graphs as well as instances of the RNA ENERGY BARRIER problem. The approach
 113 based on directed pathwidth yields reasonable solving times for RNA strings of length up
 114 to ~ 150 .

115 **Outline.** To start with, Section 2 presents some previously known results related to BISR,
 116 as well as some alternative formulations or parameters. Then, Section 3 shows that BISR
 117 is in fact equivalent to the computation of *directed pathwidth* in directed graphs. We first
 118 present a parameterized reduction from bipartite independent set reconfiguration to an
 119 input-restricted version, on graphs allowing for a perfect matching. Then, this version of
 120 the problem is shown to be simply equivalent to the computation of directed pathwidth on
 121 general directed graphs.

122 Section 4 presents our direct algorithm for bipartite independent set reconfiguration.
 123 More precisely, Section 4.2 presents the separation lemma on which the divide-and-conquer
 124 approach of the algorithm is based, while Section 4.3 details the algorithm and its analysis.

125 To finish, Section 5 explains some optimizations specific to RNA reconfiguration instances,
 126 and presents our numerical results.

127 2 Preliminaries

128 **Previous results.** BIPARTITE INDEPENDENT SET RECONFIGURATION was proven NP-
 129 complete in [13], through the equivalent k -VERTEX COVER RECONFIGURATION problem.
 130 Formulated in terms of RNAs, and restricted to secondary structures (i.e. the subset of
 131 bipartite graphs that can be obtained in RNA reconfiguration instances), it was independently
 132 proven NP-hard in [17]. To the authors' knowledge, its parameterized complexity remains
 133 open.

134 Independent set reconfiguration in an unrestricted setting (allowing vertices which are
 135 outside from the start or end independent sets, i.e. in possibly non-bipartite graphs) when
 136 parameterized by the minimum allowed size of intermediate sets has been proven W[1]-hard
 137 [18, 9], and fixed-parameter tractable for planar graphs or graphs of bounded degree [14].
 138 Whether this more general problem is in XP for this parameter remains open. We note that
 139 in this setting, parameter ρ seems slightly less relevant since it involves computing a maximal
 140 independent set in a general graph (i.e. testing if there exists a reconfiguration from \emptyset to \emptyset
 141 with range ρ is equivalent to deciding if $\alpha(G) \geq \rho$).

142 As for algorithms for BISR, the closest precedent is an algorithm by Thachuk et al. [21].
 143 It is restricted to RNA secondary structure conflict graphs, and additionally to conflict
 144 graphs for which both parts L and R are *maximum independent sets* of G . In this restricted
 145 setting, although it is not stated as such, [21] provides an XP algorithm with respect to the
 146 barrier parameter k which then coincides with the range parameter ρ that we introduce.

147 **Restriction to the monotonous case.** A reconfiguration pathway for BIPARTITE INDE-
 148 PENDENT SET RECONFIGURATION is called *monotonous* or *direct* if every vertex is added
 149 or removed exactly once in the entire sequence. The length of a monotonous sequence is
 150 therefore necessarily: $\ell = |L \cup R| = |L| + |R|$.

151 Theorem 2 from [13] tells us that if G, ρ is a yes-instance of bipartite independent set
 152 reconfiguration, then there exists a *monotonous* reconfiguration between L and R respecting
 153 the constraints. We will therefore restrict without loss of generality our study to this simpler
 154 case. In the more restricted set studied in [21], this was also independently shown.

155 **Hardness for the barrier parameter.** In the general case where L is not necessarily
 156 a maximal independent set, the range and barrier parameters (respectively ρ and $k =$

157 $\rho - (\alpha(G) - |L|)$ may be arbitrarily different. The following result motivates our use of
 158 parameter ρ for the parameterized analysis of BISR.

159 ► **Proposition 1.** BISR is Para-NP-hard for the energy barrier parameter (i.e. NP-hard even
 160 with $k = 0$).

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

170 Note that $\alpha(G') = \alpha(G) + |N_R| = |L| + \rho$, so the barrier in (G', ρ') is $k = \rho - (\alpha(G') - |L|) =$
 171 0 . A realization for (G, ρ) can be transformed into a realization for (G', ρ) by inserting
 172 vertices from N_R first, and conversely, vertices from N_R can be ignored in a realization for
 173 (G', ρ) to obtain a realization for (G, ρ) . Therefore, since BISR is NP-Complete, it is also
 174 Para-NP-hard w.r.t the barrier k . ◀

175 **Permutation formulation and ρ -realizations.** An equivalent representation of a mono-
 176 tonous reconfiguration pathway $I_0 \dots I_\ell$ from L to R for a graph G is a *permutation* P of
 177 $L \cup R$. The i -th vertex of the permutation is the vertex that is *processed* (i.e. added or
 178 removed) between I_{i-1} and I_i (this formulation lightens the representation of a solution,
 179 from a list of vertex sets to a list of vertices). Given a subset X of vertices, we write
 180 $\delta(X) = |L \cap X| - |R \cap X|$ and $I(X) = (L \setminus X) \cup (R \cap X) = L \Delta X$ for the set obtained from
 181 L after processing vertices from X . Then $|I(X)| = |L| - \delta(X)$. We say that X is *licit* if $I(X)$
 182 is an independent set. For any prefix p of P of length i , we write $V(p)$ (or simply p if the
 183 context is clear) for the set of vertices appearing in p , and $I_i = I(V(p))$. Permutation P is
 184 licit if $V(p)$ is licit for each prefix p of P ; note that P is licit if and only if $\forall r \in R$, the
 185 neighborhood $N(r)$ of r in G appears before r in P . Thus, permutation P is a ρ -*realization*
 186 that is licit and such that for each prefix p , $|I(p)| \geq \alpha(G) - \rho$ (i.e. $\delta(V(p)) \leq \rho + |L| - \alpha(G)$).

187 **3 Connection to Directed Pathwidth**

188 **3.1 Definitions**

189 **Parameterized reduction.** In this section, we provide a definition of directed pathwidth,
 190 and then prove its parameterized equivalence to the bipartite independent set reconfiguration
 191 problem. We say two problems \mathcal{P}_1 and \mathcal{P}_2 are parametrically equivalent when there exists
 192 both a *parameterized reduction* from \mathcal{P}_1 to \mathcal{P}_2 and another from \mathcal{P}_2 to \mathcal{P}_1 . A parameterized
 193 reduction [5] from problem \mathcal{P} to problem \mathcal{Q} is a function φ from instances of \mathcal{P} to instances
 194 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) $\varphi(x)$ can be
 195 computed in time $f(k) \cdot |x|^{O(1)}$, where k is the parameter of x , and (iii) if k is the parameter
 196 of x and k' is the parameter of $\varphi(x)$, then $k' \leq g(k)$ for some (computable) function g .

197 **Interval representation.** Our definition of directed pathwidth relies on interval embeddings.
 198 Alternative definitions can be found, for instance in terms of directed path decomposition or
 199 directed vertex separation number [24, 20, 11].

200 ► **Definition 2** (Interval representation). An interval representation of a directed graph H
 201 associates each vertex $u \in H$ with an interval $I_u = [a_u, b_u]$, with a_u, b_u integers. An interval
 202 representation is valid when $(u, v) \in E \Rightarrow a_u \leq b_v$. I.e., the interval of u must start before
 203 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
 204 interval representation as $\max_{m \leq i \leq M} |\{u \mid i \in I_u\}|$

205 ► **Definition 3** (directed pathwidth). The directed pathwidth of a directed graph H is the
 206 minimum possible width of a valid interval representation of H . We note this number $\text{dpw}(H)$.

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

212 **Directed graph from perfect matching.** Given a bipartite graph G allowing for a
 213 perfect matching M , we construct an associated directed graph H in the following way: the
 214 vertices of H are the edges of the matching, and $(l, r) \rightarrow (l', r')$ is an arc of H iff $(l, r') \in G$.
 215 Alternatively, H is obtained from G, M by orienting the edges of G from L to R , and then
 216 contracting the edges of M . We will denote this graph $H(G, M)$, and simply call it the
 217 *directed graph associated to G, M* . Such a construction is relatively standard and can be
 218 found in [7, 26], for instance.

219 3.2 Directed pathwidth \Leftrightarrow Bipartite independent set reconfiguration

220 **Perfect matching case.** Our main structural result is the following. Its proof, relying on
 221 interval representations, is quite straightforward and postponed to appendix B:

222 ► **Proposition 4.** Let G be a bipartite graph allowing for a perfect matching M , and let
 223 $H(G, M)$ be the directed graph associated to G, M . Then G allows for a ρ -realization iff
 224 $\text{dpw}(H(G, M)) \leq \rho$.

225 Conversely, given any directed graph H , there exists a bipartite graph G allowing for a perfect
 226 matching M such that $H = H(G, M)$ is the directed graph associated to G, M and G allows
 227 for a ρ -realization iff $\text{dpw}(H) \leq \rho$.

228 The first half of Proposition 4 is a parameterized reduction from an input-restricted
 229 version of BIPARTITE INDEPENDENT SET RECONFIGURATION to directed pathwidth. The
 230 restriction is on bipartite graphs allowing for a perfect matching. The second half is a
 231 parameterized reduction in the other direction. In both cases, the parameter value is directly
 232 transferred, which allows to retain the same complexity when transferring an algorithm from
 233 one problem to the other.

234 **Non-perfect-matching case.** In the case where G does not allow for a perfect matching,
 235 we construct an equivalent instance G' allowing for a perfect matching M' , through the
 236 addition of new vertices. Specifically, with a bipartite graph G with sides L, R , a maximum
 237 matching M of G , and the set U of unmatched vertices in G , we extend G with $|U|$ new
 238 vertices in two sets N_L, N_R , giving a new graph G' , with sides $L' = L \cup N_L, R' = R \cup N_R$,
 239 in the following way (M' is initialized to M):

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

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

245 ► **Proposition 5.** *With G, G' defined as above, we have that G allows for a ρ -realization iff*
 246 *G' allows for a ρ -realization.*

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

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

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

258 The construction above in fact yields a parameterized reduction from BIPARTITE INDE-
 259 PENDENT SET RECONFIGURATION to its input-restricted version on bipartite graphs, allowing
 260 for a perfect matching. This input-restricted version is in turn parametrically equivalent to
 261 directed pathwidth by Proposition 4. Hence the following corollary:

262 ► **Corollary 6.** BIPARTITE INDEPENDENT SET RECONFIGURATION *is parametrically equival-*
 263 *ent to DIRECTED PATHWIDTH*

264 4 An XP algorithm for independent set reconfiguration

265 4.1 Definitions

266 We use the permutation representation of reconfiguration scenarios, i.e. licit permutations of
 267 vertices. Note that the intersection, as well as the union, of two licit set of vertices are licit.
 268 Given a realization P of G and a set of vertices X , we write $P \cap X$ for the sub-sequence of
 269 P consisting of the vertices of X , without changing the order. Likewise, $P \setminus X$ denotes the
 270 sub-sequence of P consisting of vertices *not* in X .

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

275 4.2 Separation lemma

276 The separation lemma on which our algorithm is based is proved using the following “mod-
 277 ularity” property of the imbalance functions. Interestingly, it is almost the same property
 278 (sub-modularity), on a different quantity (the in-degrees of vertices) on which rely the XP
 279 algorithm for directed pathwidth [20].

► **Lemma 7 (modularity).** *The function associating a licit subset to its corresponding inde-*
pendent set $I(X)$ verifies:

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

YY:8 Parameterized Independent Set Reconfiguration

280 **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 subtraction of one equation to the other, and an addition of $|L|$. ◀

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

286 ▶ **Lemma 8 (separation lemma).** *Let X be a separator of G . If P is a ρ -realization for G , then $(P \cap X) \cdot (P \setminus X)$ is also a ρ -realization for G .*

288 **Proof.** Let P be a ρ -realization for G and $P' = (P \cap X) \cdot (P \setminus X)$ a reshuffling, where X is processed first.

290 Consider p' a prefix of P' . There are two cases:

- 291 1. p' is included in (or equal to) $P \cap X$. In this case, $\exists p$ prefix of P such that: $p' = p \cap X$. We therefore have $|I(p')| = |I(p)| + |I(X)| - |I(p \cup X)|$, and since $|I(X)|$ is a maximum independent set of G , $|I(p')| \geq |I(p)| \geq \alpha(G) - \rho$.
- 292 2. $P \cap X$ is included in p . In that case, $\exists p$ prefix of P such that $p' = p \cup X$. We have, likewise, $|I(p')| = |I(p)| + |I(X)| - |I(p \cap X)|$ and conclude the same way.

296 ◀

297 The separation 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 8 implies that no ρ -realizations exists for G . The algorithm of the following section is based on this approach.

303 4.3 XP algorithm

304 **Algorithm details.** We present here a direct algorithm for BIPARTITE INDEPENDENT SET RECONFIGURATION, detailed in Algorithm 1. 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 9 below.

310 **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* [16], namely the *Dulmage-Mendelsohn* decomposition [3, 16], as well as the decomposition of bipartite graphs with a perfect matching into *elementary subgraphs* [16](part 4.1). Its full details are described in Appendix A.

315 ▶ **Proposition 9.** *Algorithm 1 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.*

318 **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.

321 As for time, let $C(n_1, n_2, \rho)$ be the number of recursive calls of the function *Realize* of Algorithm 1 when initially called with $|L| = n_1$, $|R| = n_2$, and some value of ρ . We will show

322

■ **Algorithm 1** XP algorithm for BIPARTITE INDEPENDENT SET RECONFIGURATION

<pre> Input : bipartite graph G (with sides L and R), integer ρ Output : a ρ-realization for G, if it exists 1 Function Realize(G, ρ): 2 if $R > L$ then swap L and R 3 if $\rho < 0$ then return \perp 4 if $L = \emptyset$ then return R in any order 5 // Trying to find a separator (cf Algorithm 2) 6 $I = \text{MixedMIS}(G)$ 7 if $I \neq \perp$ then 8 $S = (L \setminus I) \cup (R \cap I)$ // intermediate point. 9 return Realize($G[S], \rho$) · Realize($G[V \setminus S], \rho$) 10 else 11 for $\ell \in L$ do 12 if Realize($G \setminus \{\ell\}, \rho - 1$) $\neq \perp$ then 13 return $(\ell) \cdot \text{Realize}(G \setminus \{\ell\}, \rho - 1)$ 14 return \perp </pre>

323 by induction that $C(n_1, n_2, k) \leq (n_1 + n_2)^{2\rho}$. Since each call involves one computation of a
 324 maximum matching, this will prove our result.

325 Given (n_1, n_2, k) , suppose therefore that $\forall n'_1 < n_1, n'_2 < n_2, \rho' < \rho$ we have $C(n'_1, n'_2, \rho') <$
 326 $(n'_1 + n'_2)^{2\rho}$

- 327 1. If G allows for a mixed maximum independent set, the instance is split into two smaller
 328 instances, yielding $C(n_1, n_2, k) = C(n'_1, n_2, k') + C(n''_1, n'_2, k'')$ with $n'_1 + n''_1 = n_1$ and $n_2 =$
 329 $n'_2 + n''_2$. And $C(n_1, n_2, k) \leq ((n'_1 + n'_2)^{2\rho} + (n''_1 + n''_2)^{2\rho}) \leq (n'_1 + n''_1 + n'_2 + n''_2)^{2\rho} \leq$
 330 $(n_1 + n_2)^{2\rho}$.
- 331 2. else, we have the following relations: if $n_1 \leq n_2$ then $C(n_1, n_2, \rho) = n_1 \cdot C(n_1 - 1, n_2, \rho - 1)$,
 332 and if $n_2 < n_1$, then $C(n_1, n_2, \rho) = C(n_2, n_1, \rho)$. Let us treat the first case, as the second
 333 one falls back to it when applying the inversion:

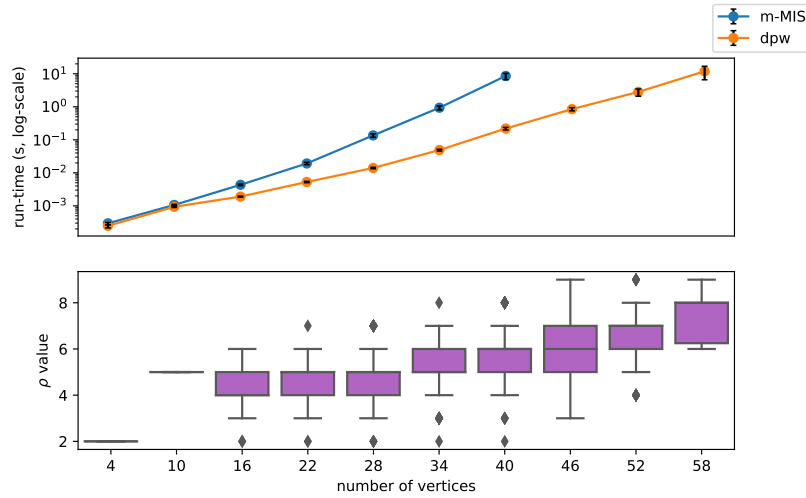
$$\begin{aligned}
 334 \quad C(n_1, n_2, \rho) &= n_1 \cdot C(n_1 - 1, n_2, \rho - 1) \\
 335 \quad &\leq n \cdot n^{2(\rho-1)} \quad \text{by induction hypothesis} \\
 336 \quad &\leq n^{2\rho} \\
 337
 \end{aligned}$$

338 ◀

339 The exponential part ($O(n^{2\rho})$) of the worst case complexity of Algorithm 1 is in fact
 340 tight, as it is met with a complete bi-clique $K_{n,n}$ with sides of size n . Indeed, in this case,
 341 no mixed MIS is found in any of the recursive calls.

342 5 Benchmarks and Applications

343 In this section, we report benchmark results for both algorithmic approaches. We first explain
 344 some details about the algorithm we implemented for directed pathwidth. Then, we present
 345 a general benchmark of our algorithms on random (Erdős-Rényi) bipartite graphs. Last, we
 346 give some background related to RNA bioinformatics and the application of our algorithm



■ **Figure 2** (top panel) Average run-time (seconds, log-scale) of our algorithms on random Erdős-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.

347 to the barrier energy problem.

348

349 5.1 Implementation details

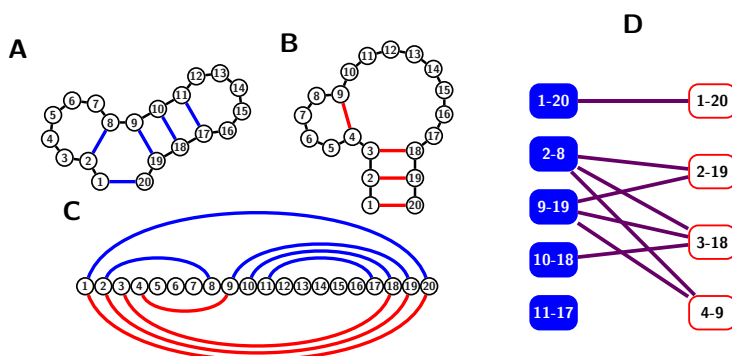
350 **Directed pathwidth.** We implemented and used an algorithm from Tamaki [20], with a
 351 runtime of $O(n^{\rho+2})$. This algorithm was originally published in 2011 [20]. In 2015, H. Tamaki
 352 and other authors described this algorithm as “flawed” in [11], and replaced it with another
 353 XP algorithm for directed pathwidth, with a run-time of $O(\frac{mn^{2\rho}}{(\rho-1)!})$.

354 Upon further analysis from our part, and discussions with H. Tamaki and the corresponding
 355 author of [11], it appears a small modification allowed to make the algorithm correct. In a
 356 nutshell, the algorithm involves *pruning* actions, and these need to be carried out *as soon as*
 357 *they are detected*. In [20], temporary solutions were accumulated before a general pruning
 358 step. With this modification, the analysis presented in [20] applies without modification, and
 359 yields a time complexity of $O(n^{\rho+2})$. The space complexity is unchanged at $O(n^{\rho+1})$. For
 360 completeness, a detailed re-derivation of the results of [20] is included in Appendix C

361 **Mixed-MIS algorithm implementation.** On Figure 2, the “m-MIS”-curve, corresponds
 362 to our mixed-MIS-based algorithm in $O(n^{2\rho} \sqrt{|V||E|})$. Compared to the algorithm presented
 363 in Algorithm 1, a more efficient rule is used in the non-separable case: we loop over all
 364 possible $r \in R$ and add $N(r) \cdot r$ to the schedule (instead of a single vertex $\ell \in L$).

365 5.2 Random bipartite graphs

366 **Benchmark details.** Figure 2 shows, as a function of the number of vertices, the average
 367 execution time of both our algorithms (top panel), as well as the distribution of parameter
 368 values (ρ - bottom panel), on a class of random bipartite graphs. These graphs are generated
 369 according to an Erdős-Rényi distribution (each pair of vertices has a constant probability
 370 p of forming an edge). We use a connection probability of d/n , dependent on the number



■ **Figure 3** 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 1.

371 of vertices. It is such that the average degree of vertices is d . The data of our benchmark
 372 (Figure 2) has been generated with $d = 5$.

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

375 5.3 Computing energy barriers in RNA kinetics

376 In this section, we give more detail about how our algorithms may apply to a bioinformatics
 377 problem, the RNA barrier energy problem. We present benchmark results, on a random
 378 class of RNA instances, showing the practicality of our approach.

379 **RNA basics.** RiboNucleic Acids (RNAs) are biomolecules of outstanding interest for
 380 molecular biology, which can be represented as strings over an alphabet $\Sigma := \{A, C, G, U\}$
 381 (in this context, n denotes the length of the string). Importantly, these strings may *fold* on
 382 themselves to adopt one or several conformation(s). A conformation is typically described
 383 by a set of base pairs $(i, j), i < j$. Then, a standard class of conformations to consider in
 384 RNA bioinformatics are *secondary structures*, which are pairwise non-crossing ($\nexists (i, j), (k, l) \in$
 385 S such that $i \leq k \leq j \leq l$, in particular, they involve distinct positions). In this section,
 386 we more precisely work on the problem of finding a reconfiguration pathway between two
 387 *secondary structures* (i.e conflict-free sets of base pairs). The reconfiguration may only involve
 388 secondary structures, and remain of energy as low as possible. We work with a simple energy
 389 model consisting of the opposite of *number of base pairs* in a configuration ($-N_{bps}$). The
 390 RNA ENERGY-BARRIER problem can then be stated as such:

RNA ENERGY-BARRIER

Input: Secondary structures L and R ; Energy barrier $k \in \mathbb{N}^+$

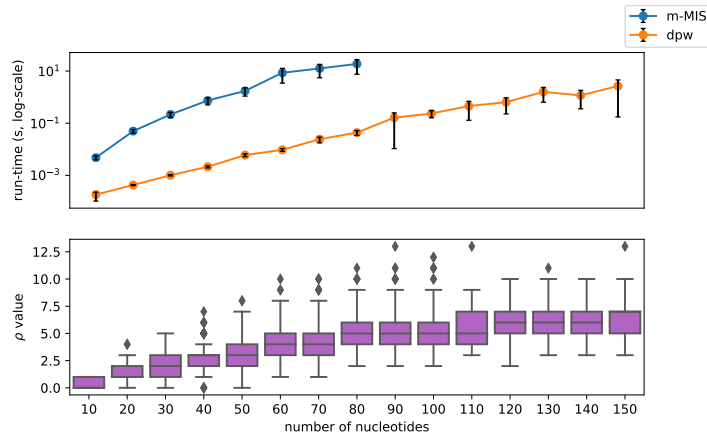
Output: True if there exists a sequence $S_0 \cdots S_\ell$ of secondary structures such that

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

False otherwise.

391 **Bipartite representation.** Given two secondary structures L and R , represented as sets of
 392 base pairs, we define a *conflict graph* $G(L, R)$ such that: the vertex set of $G(L, R)$ is $L \cup R$;

YY:12 Parameterized Independent Set Reconfiguration



■ **Figure 4** Execution time of our algorithms on random RNA reconfiguration instances (top panel). On the bottom panel, the distribution of the parameter value (ρ) is plotted against the length of the RNA string. Error bars (top panel) are obtained using a bootstrapping method.

393 and two vertices $(i, j), (k, l)$ are connected if they are crossing (see Figure 3). Since base
 394 pairs in both L and R are both pairwise non-crossing, $G(L, R)$ is bipartite with parts L
 395 and R . In this context, a maximum independent set of $G(L, R)$ is a *minimum free-energy*
 396 *structure* of the RNA, and we write $\text{MFE}(L, R) = \alpha(G(L, R))$. We then see how the RNA
 397 ENERGY-BARRIER problem is simply BIPARTITE INDEPENDENT SET RECONFIGURATION
 398 restricted to a specific class of bipartite graphs: the conflict graphs of secondary structures,
 399 with a range of $\rho = k + \text{MFE}(L, R) - |L|$.

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

409 **Related works in bioinformatics.** The problem was shown to be NP-hard by Mañuch *et*
 410 *al* [17]. Thachuk *et al* [21] also proposed an XP algorithm in $O(n^{2k+2.5})$ parameterized by
 411 the energy barrier k , restricted to instances such that the maximum independent set of
 412 $G(L, R)$ has cardinality equal to $|L|$ and $|L| = |R|$.

413 **Benchmark details.** Figure 4 shows (top panel) the average execution time of our algo-
 414 rithms on random RNA instances. The bottom panel shows the parameter distribution as
 415 a function of the length of the RNA string. Random instances are generated according to
 416 the following model: two secondary structures L, R are chosen *uniformly* at random (within
 417 the space of all possible secondary structure). Base pairs are constrained to occur between
 418 nucleotides separated by a distance of at least $\theta = 5$.

419 5.4 RNA specific optimizations

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

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

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

431 **Proof.** We have the following recurrence formula:

$$432 \quad MCF_{i,i'} = 0, \forall i' < i$$

$$433 \quad MCF_{i,j} = \max \begin{cases} MCF_{i+1,j} \\ \max_{(i,k) \in L \cup R} 1 + MCF_{i+1,k-1} + MCF_{k+1,j} \end{cases}$$

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

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

$$441 \quad MCF_{i,i'}^{\alpha,\beta} = \begin{cases} 0 & \text{if } (\alpha, \beta) = (\text{False}, \text{False}), \forall i' < i \\ -\infty & \text{otherwise} \end{cases}$$

$$442 \quad MCF_{i,j}^{\alpha,\beta} = \max \begin{cases} MCF_{i+1,j}^{\alpha,\beta} \\ \max_{\substack{(i,k) \in E \\ \alpha', \alpha'', \beta', \beta'' \in \mathbb{B}^4}} 1 + MCF_{i+1,k-1}^{\alpha',\beta'} + MCF_{k+1,j}^{\alpha'',\beta''} \end{cases} \left| \begin{array}{l} \text{if } \neg\alpha \vee \alpha' \vee \alpha'' \vee ((i, k) \in L) \\ \text{and } \neg\beta \vee \beta' \vee \beta'' \vee ((i, k) \in R) \end{array} \right.$$

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

447 6 Conclusion

448 Our work so far sheds a new light on both BIPARTITE INDEPENDENT SET RECONFIGURATION
 449 and DIRECTED PATHWIDTH problems. The former can thus be solved with a parameterized
 450 algorithm, having important applications in RNA kinetics since the range parameter is
 451 particularly relevant in this context. We hope the newly drawn connection will help settle the
 452 fixed parameter tractability of computing the directed pathwidth. A slightly more accessible
 453 open problem would be to design an FPT algorithm for BISR in the context of secondary
 454 structure conflict graphs (i.e. those graphs arising in RNA reconfiguration).

455 — References

- 456 1 János Barát. Directed path-width and monotonicity in digraph searching. *Graphs and*
457 *Combinatorics*, 22(2):161–172, 2006.
- 458 2 Hans L Bodlaender. Fixed-parameter tractability of treewidth and pathwidth. In *The*
459 *Multivariate Algorithmic Revolution and Beyond*, pages 196–227. Springer, 2012.
- 460 3 Jianer Chen and Iyad A Kanj. Constrained minimum vertex cover in bipartite graphs:
461 complexity and parameterized algorithms. *Journal of Computer and System Sciences*, 67(4):833–
462 847, 2003.
- 463 4 David Coudert, Dorian Mazaauric, and Nicolas Nisse. Experimental evaluation of a branch-and-
464 bound algorithm for computing pathwidth and directed pathwidth. *Journal of Experimental*
465 *Algorithmics (JEA)*, 21:1–23, 2016.
- 466 5 Marek Cygan, Fedor V Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin
467 Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized algorithms*, volume 5. Springer,
468 2015.
- 469 6 Joshua Erde. Directed path-decompositions. *SIAM Journal on Discrete Mathematics*, 34(1):415–
470 430, 2020.
- 471 7 Komei Fukuda and Tomomi Matsui. Finding all the perfect matchings in bipartite graphs.
472 *Applied Mathematics Letters*, 7(1):15–18, 1994.
- 473 8 Marinus Gottschau, Felix Happach, Marcus Kaiser, and Clara Waldmann. Budget minimization
474 with precedence constraints. *CoRR*, abs/1905.13740, 2019. URL: [http://arxiv.org/abs/](http://arxiv.org/abs/1905.13740)
475 [1905.13740](http://arxiv.org/abs/1905.13740), [arXiv:1905.13740](https://arxiv.org/abs/1905.13740).
- 476 9 Takehiro Ito, Marcin Kamiński, Hirotaka Ono, Akira Suzuki, Ryuhei Uehara, and Katsuhisa
477 Yamanaka. Parameterized complexity of independent set reconfiguration problems. *Discrete*
478 *Applied Mathematics*, 283:336–345, 2020.
- 479 10 Jeff Kinne, Ján Manuch, Akbar Rafiey, and Arash Rafiey. Ordering with precedence constraints
480 and budget minimization. *CoRR*, abs/1507.04885, 2015. URL: [http://arxiv.org/abs/1507.](http://arxiv.org/abs/1507.04885)
481 [04885](http://arxiv.org/abs/1507.04885), [arXiv:1507.04885](https://arxiv.org/abs/1507.04885).
- 482 11 Kenta Kitsunai, Yasuaki Kobayashi, Keita Komuro, Hisao Tamaki, and Toshihiro Tano.
483 Computing directed pathwidth in $o(1.89^n)$ time. *Algorithmica*, 75(1):138–157, 2016.
- 484 12 Yasuaki Kobayashi, Keita Komuro, and Hisao Tamaki. Search space reduction through
485 commitments in pathwidth computation: An experimental study. In *International Symposium*
486 *on Experimental Algorithms*, pages 388–399. Springer, 2014.
- 487 13 Daniel Lokshtanov and Amer E Mouawad. The complexity of independent set reconfiguration
488 on bipartite graphs. *ACM Transactions on Algorithms (TALG)*, 15(1):1–19, 2018.
- 489 14 Daniel Lokshtanov, Amer E Mouawad, Fahad Panolan, MS Ramanujan, and Saket Saurabh.
490 Reconfiguration on sparse graphs. *Journal of Computer and System Sciences*, 95:122–131,
491 2018.
- 492 15 Daniel Lokshtanov, Amer E Mouawad, Fahad Panolan, and Sebastian Siebertz. On the
493 parameterized complexity of reconfiguration of connected dominating sets. *arXiv preprint*
494 *arXiv:1910.00581*, 2019.
- 495 16 László Lovász and Michael D Plummer. *Matching theory*, volume 367. American Mathematical
496 Soc., 2009.
- 497 17 Ján Maňuch, Chris Thachuk, Ladislav Stacho, and Anne Condon. Np-completeness of
498 the energy barrier problem without pseudoknots and temporary arcs. *Natural Computing*,
499 10(1):391–405, 2011.
- 500 18 Amer E Mouawad, Naomi Nishimura, Venkatesh Raman, Narges Simjour, and Akira Suzuki.
501 On the parameterized complexity of reconfiguration problems. *Algorithmica*, 78(1):274–297,
502 2017.
- 503 19 Hisao Tamaki. A directed path-decomposition approach to exactly identifying attractors of
504 boolean networks. In *2010 10th International Symposium on Communications and Information*
505 *Technologies*, pages 844–849. IEEE, 2010.

- 506 20 Hisao Tamaki. A polynomial time algorithm for bounded directed pathwidth. In Petr Kolman
507 and Jan Kratochvíl, editors, *Graph-Theoretic Concepts in Computer Science*, pages 331–342,
508 Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.
- 509 21 Chris Thachuk, Jan Manuch, Arash Rafiey, Leigh-Anne Mathieson, Ladislav Stacho, and
510 Anne Condon. An Algorithm for the Energy Barrier Problem Without Pseudoknots and
511 Temporary Arcs. In *Biocomputing 2010*, pages 108–119. World Scientific, oct 2009. doi:
512 10.1142/9789814295291_0013.
- 513 22 Ignacio Tinoco Jr and Carlos Bustamante. How rna folds. *Journal of molecular biology*,
514 293(2):271–281, 1999.
- 515 23 Jan van den Heuvel. The complexity of change. *Surveys in combinatorics*, 409(2013):127–160,
516 2013.
- 517 24 Boting Yang and Yi Cao. Digraph searching, directed vertex separation and directed pathwidth.
518 *Discrete Applied Mathematics*, 156(10):1822–1837, 2008.
- 519 25 Zan-Bo Zhang and Dingjun Lou. Bipartite graphs with a perfect matching and digraphs.
520 *arXiv preprint arXiv:1011.4359*, 2010.
- 521 26 Zan-Bo Zhang, Xiaoyan Zhang, and Xuelian Wen. Directed hamilton cycles in digraphs
522 and matching alternating hamilton cycles in bipartite graphs. *SIAM J. Discret. Math.*,
523 27(1):274–289, 2013. doi:10.1137/110837188.

524 **A** Mixed MIS in bipartite graphs

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

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

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

535 We present here a more efficient strategy, based on a decomposition taking place in two
536 rounds. It results into Algorithm 2. The first round is based on the Dulmage-Mendelsohn
537 decomposition of bipartite graphs. It yields a partition of the vertices of G into three sets
538 D, A, C , defined as such: for each vertex v of D , there exists a maximum matching in which
539 v is not matched, $A = N(D)$ is the union of the neighborhoods of the vertices of D , and
540 $C = V \setminus (D \cup A)$ contains the remaining vertices. D, A, C verify the following result:

541 ► **Theorem 11** (Dulmage-Mendelsohn decomposition, Proposition 2.1 of [3], theorem 3.2.4 of
542 [16]). *Given G bipartite graph and D, A, C defined as above, we have that:*

- 543
- 544 a. ■ D is the intersection of all maximum independent sets of G .
545 ■ A is the intersection of all minimum vertex covers of G .
546 ■ the subgraph $G[C]$ induced by C has a perfect matching, which may be deduced from
547 restricting any maximum matching of G to C .
- 548
- 549 b. In addition, D may be computed from any maximum matching M of G using the following
550 characterization ([3], lemma 2.2): $D = \overline{W}$ where \overline{W} is composed of the vertices left
551 unmatched by M , as well as all vertices connected to an unmatched vertex through an
552 alternating path of even length.

YY:16 Parameterized Independent Set Reconfiguration

553 This decomposition may allow to conclude in some cases (see Algorithm 2). In general,
 554 however, a second round of decomposition is needed. In this second round, the set C ,
 555 which allows for a perfect matching M , is further decomposed into *elementary sub-graphs*
 556 (section 4.1 of [16], theorem 4.1.1 and exercise 4.1.5) and [25]. It consists in computing
 557 the strongly connected components of a directed graph $H(M, C)$ associated to M and C
 558 (same construction as in Section 3). The vertices of H are the edges of the matching, and
 559 $(l, r) \rightarrow (l', r')$ iff l is connected to r' in C . The strongly connected components of H constitute
 560 a decomposition of G into elementary sub-graphs. A bipartite graph is elementary iff the
 561 sides L, R are the only minimum vertex covers/maximum independent sets [16](theorem
 562 4.1.1). If it is not elementary, then a mixed maximum independent set may be obtained by
 563 ordering the elementary sub-graphs $\{(L_i, R_i)\}_{1 \leq i \leq p}$ along a topological order induced by
 564 $H(C, M)$. 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
 565 independent set of C .

566 The discussion above results in Algorithm 2, whose run-time is dominated by the compu-
 567 tation of maximum matching in $O(\sqrt{|V|}|E|)$.

■ Algorithm 2 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
           $R$ .

1 // Compute a maximum matching of  $G$ 
2  $M = \text{MaximumMatching}(G)$   $\triangleright O(\sqrt{|V|} \cdot |E|)$ 
3 // Compute a Maximum Independent Set  $I$  from  $M$  (König's theorem).
4  $I = \text{MaximumIndependentSet}(G, M)$   $\triangleright O(|E|)$ 
5 if  $(I \cap L \neq \emptyset)$  and  $(I \cap R \neq \emptyset)$  then return  $I$ 
6 // Now  $|I| = \max(|L|, |R|)$  and  $I = L$  or  $I = R$ 
7  $D, A, C = \text{DulmageMendelsohn}(M, G)$   $\triangleright O(|E|)$ 
8 if  $|L| > |R|$  then
9   if  $R \setminus A \neq \emptyset$  then
10     pick  $r \in R \setminus A$  //  $A$  is the intersection of all minimum vertex covers
11      $G' = G \setminus \{r \cup N(r)\}$ 
12      $M' = \text{MaximumMatching}(G')$ 
13      $I' = \text{MaximumIndependentSet}(G', M')$ 
14     return  $I' \cup \{r\}$ 
15   else return  $\perp$ ; // Not possible,  $L$  is the only MIS
16 if  $|L| = |R|$  then
17   //  $L$  and  $R$  are two MIS. So necessarily  $D = \emptyset, A = \emptyset, C = G$ 
18    $(L_1, R_1), \dots, (L_p, R_p) = \text{elementarySubgraphsDec}(M, C)$   $\triangleright O(|V|^2)$ 
19   if  $p=1$  then return  $\perp$ 
20   else
21     Topological sort of the SCCs of  $H$ 
22      $s = \text{TopologicalSort}(\{(L_i, R_i)\})$   $\triangleright O(|V| + |E|)$ 
23      $(L_i, R_i) = s[0]$  // first in topological sort
24     return  $R_i \cup (\cup_{j \neq i} L_j)$ 

```

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 $a_{u_0} = \dots = a_{u_l} = b_{v_0} = \dots = b_{v_p} = n$, we may modify the representation as such:

- Interval bounds associated to integers $> n$ are increased by $p + l - 1$, to make room for “spreading” $a_{u_1} \dots a_{u_\ell}, b_{v_1} \dots b_{v_p}$.
- $\forall i$, 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 $[1 \dots 2 \cdot |V(H)|]$ by taking the interval bounds in order and setting them to their final position.

B.2 Proof of Proposition 4:

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 .

\Rightarrow 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 $(l, r) \rightarrow (l', r')$ 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 $\leq \rho$.

\Leftarrow 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

613 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
 614 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
 615 any $(u, v) \in E(H)$. G now verifies $H = H(G, M)$, and by the result above, $dpw(H) \leq \rho$ iff
 616 G allows for a ρ -realization.

617

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

619 For completeness, we include here a re-derivation of the results of [20], with the slight
 620 modification mentioned in the main text related to *pruning*. It results in an algorithm with a
 621 $O(n^{\rho+2})$ complexity, slightly different from the $O(n^{\rho+1})$ announced in [20]. The re-derivation
 622 follows the same strategy as in the original article, and re-uses most of the notations.

623 C.1 Commitment lemma - shortest non-expanding extensions 624 (SNEKFEs)

625 **Notations and definitions.** In a directed graph, $d^-(u)$ denotes the in-degree of a node
 626 u . We work with layouts of vertices, i.e. ordered sequences of vertices, not necessarily
 627 containing all vertices. A partial layout σ is called *feasible/valid* if \forall prefix p of σ we have
 628 $d^-(p) = |N^-(p)| \leq k$. A partial layout which is *completable* into a valid full layout (for
 629 the entire digraph G) is called *strongly feasible* or just *completable into a full solution*. An
 630 *extension* τ of σ is a valid partial layout with σ as one of its prefixes. A *shortest non-expanding*
 631 *extension* of σ is an extension τ such that $d^-(\tau) \leq d^-(\sigma)$ and $\forall \rho$ s.t. $V(\sigma) \subsetneq V(\rho) \subsetneq V(\tau)$,
 632 $d^-(\rho) > d^-(\sigma)$. In the rest of this note, we will write SNEKFE for shortest non-expanding
 633 extension.

634 **Lemma 1 - Commitment Lemma - shortest non-expanding extensions.** *If σ is*
 635 *completable into a full solution, and τ is a SNEKFE of σ , then τ is also completable into a*
 636 *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 :

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

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

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

641 Else, if P contains some of F' , then $P = P' \cup \tau$ for P' a certain prefix of $\sigma \cdot F$. As for
 642 $P' \cap \tau$, which we call ρ it verifies $V(\sigma) \subset V(\rho) \subset V(\tau)$ and therefore $d^-(\rho) \geq d^-(\sigma) \geq d^-(\tau)$
 643 by definition of a SNEKFE, with the equality only potentially happening if $\rho = \sigma$ or $\rho = \tau$.

644 We therefore have:

$$\begin{aligned} 645 \quad d^-(P) &= d^-(P' \cup \tau) \\ 646 \quad &\leq d^-(P') + d^-(\tau) - d^-(\rho) \\ 647 \quad &\leq d^-(P') \leq k \end{aligned}$$

649 $\tau \cdot F'$ is therefore a valid complete layout for G , and τ is completable into a full solution. ◀

650 Let us now describe more precisely what SNEKFEs might look like. We show that they
 651 can only be of three types, and formalize it into the next lemma. Its proof relies on the

652 fact that, by adding a single vertex u to a partial layout σ , we may only decrease $d^-(\sigma)$ by
 653 at most 1, since $d^-(\sigma) = |N^-(\sigma)|$. We obtain this decrement of 1 if u is a predecessor to a
 654 vertex of σ , and does not introduce any new predecessor itself when added.

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

- 657 ■ *type-(i): single-vertex “decreasing” extension: $\tau = \sigma \cdot u$ for some vertex u and $d^-(\sigma \cdot u) =$
 658 $d^-(\sigma) - 1$*
- 659 ■ *type-(ii): single-vertex “non-decreasing” extension: $\tau = \sigma \cdot u$ for some vertex u and
 660 $d^-(\sigma \cdot u) = d^-(\sigma)$*
- 661 ■ *type-(iii): several vertices “shortcut” extension: τ adds strictly more than one vertex to σ
 662 and $d^-(\tau) = d^-(\sigma)$.*

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

665 For SNEKFEs composed of more than one vertex, observe that if $d^-(\tau) < d^-(\sigma)$, then
 666 by considering the prefix ρ of τ obtained by removing just 1 vertex to τ , we would have
 667 $d^-(\rho) \leq d^-(\tau) + 1 \leq d^-(\sigma)$. This stems from the observation above that d^- may only decrease
 668 by at most 1 when adding a vertex. ρ would be a non-expanding extension of σ shorter than
 669 τ , yielding a contradiction. ◀

670 C.2 Algorithm

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

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

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

678 For each $\sigma \in S_i$:

- 680 1. We generate all *feasible immediate extensions* to σ and add them to the tree. I.e the
 681 node σ now has the following children set: $\{\sigma \cdot u \text{ s.t. } d^-(\sigma \cdot u) \leq k\}$
- 682 2. If some of these immediate extensions verify $d^-(\sigma \cdot u) \leq d^-(\sigma)$, then they are SNEKFEs
 683 of σ . In that case, we do the following:
 - 684 a. We choose 1 arbitrarily and prune the others.
 - 685 b. If the chosen element verifies $d^-(\sigma \cdot u) = d^-(\sigma) - 1$ (the only possibility if $d^-(\sigma \cdot u) <$
 686 $d^-(\sigma)$), then we in addition look for a prefix η of σ verifying $d^-(\eta) = d^-(\sigma \cdot u)$ and
 687 $d^-(\rho) > d^-(\eta) \forall \rho \text{ s.t. } \eta \sqsubseteq \rho \sqsubseteq \sigma \cdot u, \rho \neq \eta, \rho \neq \sigma \cdot u$.

688 If such an η is found, then any part of tree branching off the path from η to $\sigma \cdot u$ is
 689 removed. Note that this might shorten the overall loop over $\sigma \in S_i$.

690 **End Algorithm**

691 C.3 Analysis

692 This section will be composed of three parts. In the first one, we define an invariant property
 693 (“internally pruned”) for trees of prefixes of layouts of vertices. In the second one, we show
 694 that, in the algorithm presented in the previous section, the tree of prefixes verifies the
 695 invariant at all times, and prove the correctness of the algorithm. Finally, in the third part,

YY:20 Parameterized Independent Set Reconfiguration

696 we analyze the size of trees of prefixes verifying the invariant, proving that each level S_i of
697 such a tree has a size $\leq n^k$, yielding a complexity analysis of the algorithm.

698 C.3.1 Internally pruned trees of prefixes

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

704

705

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

711 C.3.2 Invariant and correctness

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

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

716

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

720 ■ If all of the immediate extensions are such that $\{d^-(\sigma) < d^-(\sigma \cdot u) \leq k\}$, then no new
721 SNEKFEs are generated when adding them to the tree. (if $\sigma \cdot u$ is a SNEKFE of some
722 η up the tree, then σ is shorter and also non-expanding). After the addition of the
723 immediate extension, the tree is therefore still internally pruned.

724 ■ If one of these immediate extensions verifies $d^-(\sigma \cdot u) = d^-(\sigma)$ but none of them verify
725 $d^-(\sigma \cdot u) < d^-(\sigma)$, then one of these extensions is a SNEKFE of σ , and is kept while
726 the others are pruned. However, this is the only SNEKFE introduced by the extension.
727 Therefore, the pruning of immediate extensions other than the selected one is enough to
728 keep the tree internally pruned.

729 ■ If one of the immediate extensions verifies $d^-(\sigma \cdot u) = d^-(\sigma) - 1$, then one of the immediate
730 extensions is selected and the others are pruned, as in the previous case. However, in
731 addition, $\sigma \cdot u$ might be a new shortest non-expanding extension of a node η up the tree.
732 If this is the case, then there is only one such η , per the definition of shortest non-expanding
733 extensions.

734 We argue that the conditions used in the algorithm indeed detect such an η .

735 If $\sigma \cdot u$ is a SNEKFE of η , then the conditions described in the algorithm (that $d^-(\sigma \cdot u) =$
736 $d^-(\eta)$, and $d^-(\rho) > d^-(\eta)$ for any ρ on the path from η to $\sigma \cdot u$) are verified.

737 Conversely, if the conditions are verified, then suppose η has a shorter non-expanding
738 extensions τ . τ cannot be on the path from η to $\sigma \cdot u$ as that would imply $d^-(\tau) > d^-(\eta)$.
739 Since τ is shorter than $\sigma \cdot u$, τ has been generated in a previous step of the algorithm. At

740 this point, step 2.b of the algorithm would have pruned the path to σ , which cannot be
741 visited, leading to a contradiction.

742
743 Therefore, the potentially newly introduced SNEKFE is detected, and the corresponding
744 pruning is carried out, leaving the tree internally pruned
745 Therefore, after each extension of the tree throughout the algorithm, the tree remains
746 internally pruned. ◀

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

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

750 **Proof.** Denote the set of full solutions S , and suppose all solutions are absent from \mathcal{T} .

751 $\forall \sigma \in S$, there is some (possibly empty) prefix of σ in \mathcal{T} .

752 We pick $\sigma \in S$ allowing for the largest prefix $\eta \in \mathcal{T}$, i.e:

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

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

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

758 Per the commitment lemma, τ is the prefix of a full solution σ'' . But $|\tau| > |\eta|$, contra-
759 dicting the choice of σ . ◀

760 C.3.3 Signature analysis

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

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

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

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

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

773 ■ Either the next move is part of a SNEKFE. In this case there are no choices to be made,
774 the added vertex is not part of the signature, and is the same for σ and τ .

775 ■ Or the next move is not part of a SNEKFE. In this case, several choices are possible, and
776 the next added vertex will be part of the signature. Since the signatures of σ and τ are
777 the same, the same vertex is added to σ and τ .

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

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

YY:22 Parameterized Independent Set Reconfiguration

781 ■ $d^-(\sigma) > d^-(\tau)$

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

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

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

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

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

791 ■ else, if $d^-(\sigma \cdot v) > d^-(\sigma)$, we know, since $d^-(\tau) < d^-(\sigma)$ and the d^- -curve only decreases
792 by steps of -1 , that there must exist ρ such that $d^-(\rho) = d^-(\sigma)$, $\sigma \sqsubseteq \rho \sqsubseteq \tau$, and
793 $d^-(\rho') > d^-(\sigma)$ for any ρ' such that $\sigma \sqsubseteq \rho' \subseteq \rho$ (ρ is the shortest prefix of τ which contains
794 σ and has the same d^- value). ρ is then a type-(iii) SNEKFE of σ by Lemma 4, and we
795 may apply the induction hypothesis to $\tau \setminus \sigma$
796 ◀

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

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

800 ■ If $\sigma \cdot u$ is not a SNEKFE of σ , then $|sgn(\sigma \cdot u)| = |sgn(\sigma) \cup \{u\}| = |sgn(\sigma)| + 1 \leq$
801 $d^-(\sigma) + 1 \leq d^-(\sigma \cdot u)$

802 ■ If σ is a type-(ii) SNEKFE of σ , then $sgn(\sigma) = sgn(\sigma \cdot u)$ and $d^-(\sigma \cdot u) = d^-(\sigma)$.

803 ■ If $\sigma \cdot u$ is a type-(i) SNEKFE of σ , then consider η , the closest node (up the tree)
804 such that $d^-(\eta) < d^-(\sigma \cdot u)$, and $\eta \cdot v$ its successor on the path to $\sigma \cdot u$. We have
805 $d^-(\eta) < d^-(\sigma \cdot u) \leq d^-(\eta \cdot v)$, by definition of η . The path from $\eta \cdot v$ to u is either
806 a type-(iii) SNEKFE or overall-decreasing. Therefore $sgn(\sigma \cdot u) = sgn(\eta \cdot v)$. and
807 $|sgn(\sigma \cdot u)| = |sgn(\eta)| + 1 \leq d^-(\eta) + 1$ by induction hypothesis, and $|sgn(\sigma \cdot u)| \leq d^-(\sigma \cdot u)$.
808 ◀

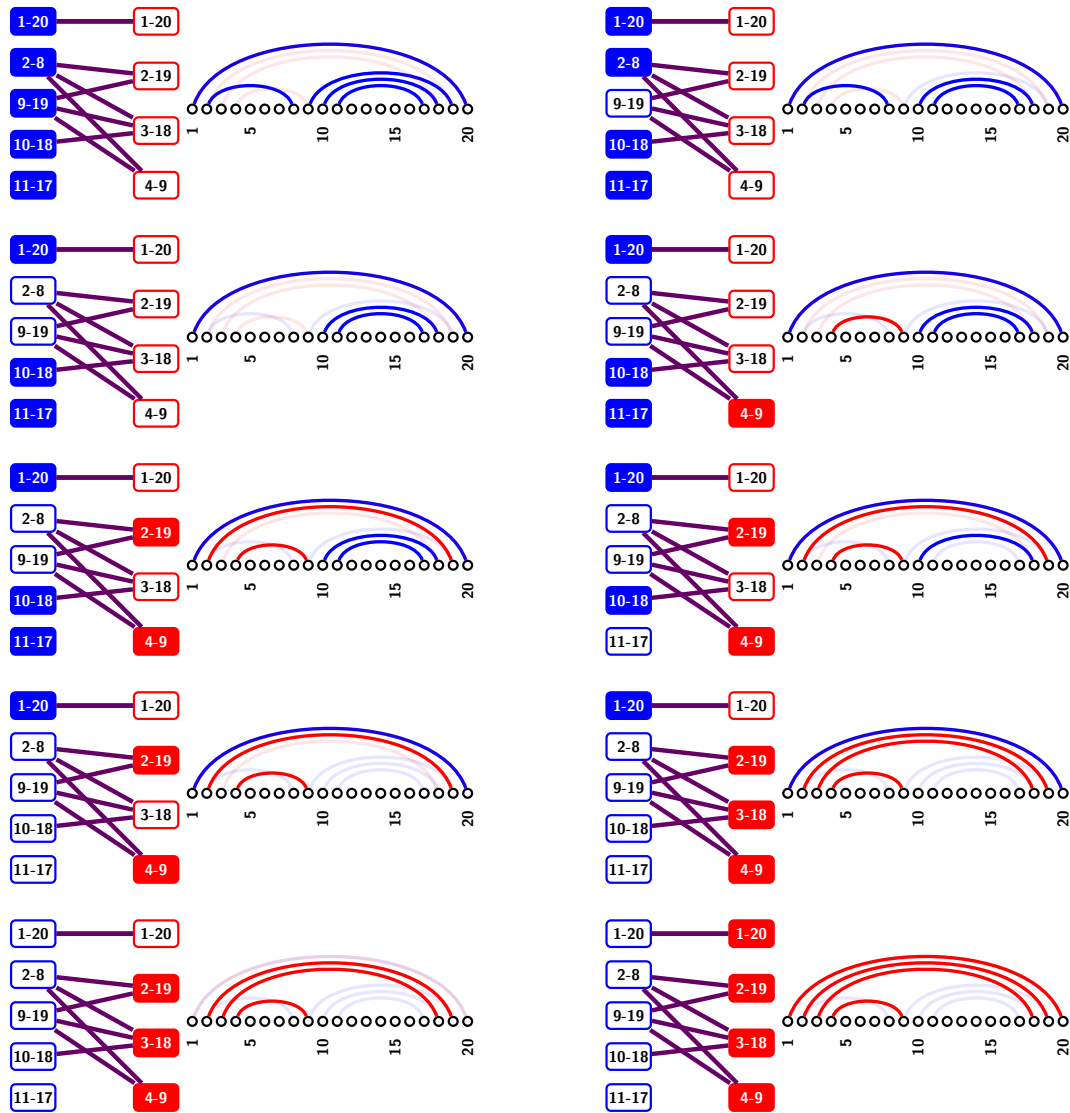
809 In particular, $\forall \sigma$ partial layout, $d^-(\sigma) \leq k$. Since two different elements of S_i need
810 different signatures, we get the following corollary:

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

812 The overall complexity of the algorithm is therefore $O(n^{k+O(1)})$. More precisely, it is
813 $O(n^{k+2})$. (there are n levels of the tree to fill, $\leq n^k$ nodes per level and $O(n)$ work per node
814 to generate the next level).

815 **D Detailed RNA reconfiguration example**

816 We provide in Figure 5 the intermediate sets of base pairs, and associated RNA secondary
817 structures, for our running example, described in Figures 1 and 3.



■ **Figure 5** 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.