Approximating Highly Inapproximable Problems on Graphs of Bounded Twin-Width

Pierre Bergé
Univ Lyon, CNRS, ENS de Lyon, Université Claude Bernard Lyon 1, LIP UMR5668, France

Édouard Bonnet
Univ Lyon, CNRS, ENS de Lyon, Université Claude Bernard Lyon 1, LIP UMR5668, France

Hugues Déprés
Univ Lyon, CNRS, ENS de Lyon, Université Claude Bernard Lyon 1, LIP UMR5668, France

Rémi Watrigant
Univ Lyon, CNRS, ENS de Lyon, Université Claude Bernard Lyon 1, LIP UMR5668, France

Abstract

For any $\varepsilon > 0$, we give a polynomial-time n^{ε}-approximation algorithm for Max Independent Set in graphs of bounded twin-width given with an $O(1)$-sequence. This result is derived from the following time-approximation trade-off: We establish an $O(1)^{2^q-1}$-approximation algorithm running in time $\exp(O_q(n^{2^{-q}}))$, for every integer $q \geqslant 0$. Guided by the same framework, we obtain similar approximation algorithms for Min Coloring and Max Induced Matching. In general graphs, all these problems are known to be highly inapproximable: for any $\varepsilon > 0$, a polynomial-time $n^{1-\varepsilon}$-approximation for any of them would imply that P=NP [Håstad, FOCS '96; Zuckerman, ToC '07; Chalermsook et al., SODA '13]. We generalize the algorithms for Max Independent Set and Max Induced Matching to the independent (induced) packing of any fixed connected graph H.

In contrast, we show that such approximation guarantees on graphs of bounded twin-width given with an $O(1)$-sequence are very unlikely for Min Independent Dominating Set, and somewhat unlikely for Longest Path and Longest Induced Path. Regarding the existence of better approximation algorithms, there is a (very) light evidence that the obtained approximation factor of n^{ε} for Max Independent Set may be best possible. This is the first in-depth study of the approximability of problems in graphs of bounded twin-width. Prior to this paper, essentially the only such result was a polynomial-time $O(1)$-approximation algorithm for Min Dominating Set [Bonnet et al., ICALP '21].

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis; Theory of computation → Design and analysis of algorithms

Keywords and phrases Approximation algorithms, bounded twin-width

Acknowledgements We thank Colin Geniet, Eunjung Kim, and Stéphan Thomassé for useful discussions.

1 Introduction

Twin-width is a graph parameter introduced by Bonnet, Kim, Thomassé, and Watrigant [10]. Its definition involves the notions of *trigraphs* and of *contraction sequences*. A *trigraph* is a graph with two types of edges: black (regular) edges and red (error) edges. A (vertex) *contraction* consists of merging two (non-necessarily adjacent) vertices, say, u, v into a vertex w, and keeping every edge wz black if and only if uz and vz were previously black edges. The other edges incident to w become red (if not already), and the rest of the trigraph remains the same. A *contraction sequence* of an n-vertex[1] graph G is a sequence of trigraphs

[1] In this introduction, we might implicitly use n to denote the number of vertices, and m, the number of edges of the graph at hand.

$G = G_n, \ldots, G_1 = K_1$ such that G_i is obtained from G_{i+1} by performing one contraction. A *d-sequence* is a contraction sequence in which every vertex of every trigraph has at most d red edges incident to it. The *twin-width* of G, denoted by $\mathrm{tww}(G)$, is then the minimum integer d such that G admits a d-sequence. Figure 1 gives an example of a graph with a 2-sequence, i.e., of twin-width at most 2. Twin-width can be naturally extended to matrices (with unordered [10] or ordered [8] row and column sets) over a finite alphabet, and thus to binary structures.

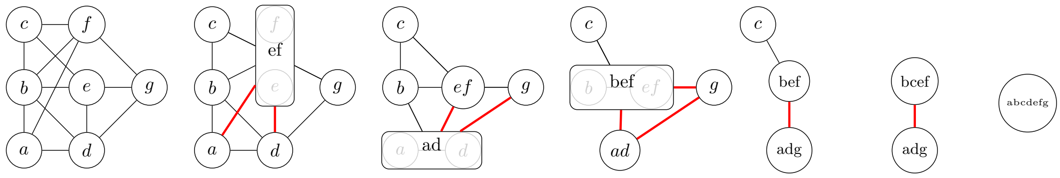


Figure 1 A 2-sequence witnessing that the initial graph has twin-width at most 2.

An equivalent viewpoint that will be somewhat more convenient is to consider a d-sequence as a sequence of partitions $\mathcal{P}_n := \{\{v\} : v \in V(G)\}, \mathcal{P}_{n-1}, \ldots, \mathcal{P}_1 := \{V(G)\}$ of $V(G)$, such that for every integer $1 \leqslant i \leqslant n-1$, $\mathcal{P}_i$ has i parts and is obtained by merging two parts of $\mathcal{P}_{i+1}$ into one. Now the *red degree* of a part $P \in \mathcal{P}_i$ is the number of other parts $Q \in \mathcal{P}_i$ such that there is in G at least one edge and at least one non-edge between P and Q. A d-sequence is such that no part of no partition of the sequence has red degree more than d. In that case the *maximum red degree* of each partition is at most d. And we similarly get the twin-width of G as the minimum integer d such that G admits a (partition) d-sequence. The *quotient trigraph* $G/\mathcal{P}_i$ is the trigraph G_i, if the (contraction) d-sequence $G_n, \ldots, G_1$ and the (partition) d-sequence $\mathcal{P}_n, \ldots, \mathcal{P}_1$ correspond.

Classes of binary structures with bounded twin-width include graph classes with bounded treewidth, and more generally bounded clique-width, proper minor-closed classes, posets with antichains of bounded size, strict subclasses of permutation graphs, as well as $\Omega(\log n)$-subdivisions of n-vertex graphs [10], and some classes of (bounded-degree) expanders [5]. A notable variety of geometrically defined graph classes have bounded twin-width such as map graphs, bounded-degree string graphs [10], classes with bounded queue number or bounded stack number [5], segment graphs with no $K_{t,t}$ subgraph, visibility graphs of 1.5D terrains without large half-graphs, visibility graphs of simple polygons without large independent sets [4].

For every class $\mathcal{C}$ mentioned so far, $O(1)$-sequences can be computed in polynomial time[2] on members of $\mathcal{C}$. For classes of binary structures including a binary relation interpreted as a linear order on the domain (called *ordered binary structures*), there is a fixed-parameter approximation algorithm for twin-width [8]. More precisely, given a graph G and an integer k, there are computable functions f and g such that one can output an $f(k)$-sequence of G or correctly report that $\mathrm{tww}(G) > k$ in time $g(k)n^{O(1)}$. Such an approximation algorithm is currently missing for classes of general (not necessarily ordered) binary structures, and in particular for the class of all graphs. We also observe that deciding if the twin-width of a graph is at most 4 is an NP-complete task [3].

We will therefore assume that the input graph is given with a d-sequence, and treat d as a constant (or that the input comes from any of the above-mentioned classes). Thus far, this is the adopted setting when designing faster algorithms on bounded twin-width

[2] Admittedly, for the geometric classes, a representation is (at least partially) needed.

graphs [10, 7, 33, 30, 19]. From the inception of twin-width [10] –actually already from the seminal work of Guillemot and Marx [21]– it was clear that structures wherein this invariant is bounded may *often* allow the design of parameterized algorithms. More concretely, it was shown [10] that, on graphs G given with a d-sequence, model checking a first-order sentence φ is fixed-parameter tractable –it can be solved in time $f(d, \varphi) \cdot n$–, the special cases of, say, k-Independent Set or k-Dominating Set admit single-exponential parameterized algorithms [7], an effective data structure almost linear in n can support constant-time edge queries [33], the triangles of G can be counted in time $O(d^2n + m)$ [30].

So far, however, the connection between *having bounded twin-width* and *enjoying enhanced approximation factors* was tenuous. The only such result concerned Min Dominating Set, known to be inapproximable in polynomial-time within factor $(1-o(1)) \ln n$ unless P=NP [16], but yet admits a constant-approximation on graphs of bounded twin-width given with an $O(1)$-sequence [7]. We start filling this gap by designing approximation algorithms on graphs of bounded twin-width given with an $O(1)$-sequence for notably Max Independent Set (MIS, for short), Max Induced Matching, and Coloring. Getting better approximation algorithms for MIS and Coloring in that particular scenario was raised as an open problem [7]. Before we describe our results and elaborate on the developed techniques, let us briefly present the notorious inapproximability of these problems in general graphs.

MIS and Coloring are NP-hard [20], and very inapproximable: for every $\varepsilon > 0$, it is NP-hard to approximate these problems within ratio $n^{1-\varepsilon}$ [23, 34]. The same was shown to hold for Max Induced Matching [13]. Besides, there is only little room to improve over the brute-force algorithm in $2^{O(n)}$: Unless the Exponential Time Hypothesis[3] [25] (ETH) fails, no algorithm can solve MIS in time $2^{o(n)}$ [26] (nor the other two problems). For any r (possibly a function of n) WMIS can be r-approximated in time $2^{O(n/r)}$ [15, 12]. Bansal et al. [2] essentially shaved a $\log^2 r$ factor to the latter exponent. It is known though that polynomial shavings are unlikely. Chalermsook et al. [14] showed that, for any $\varepsilon > 0$ and sufficiently large r (again r can be function of n), an r-approximation for MIS and Max Induced Matching cannot take time $2^{O(n^{1-\varepsilon}/r^{1+\varepsilon})}$, unless the ETH fails. For instance, investing time $2^{O(\sqrt{n})}$, one cannot hope for significantly better than a $\sqrt{n}$-approximation.

Contributions and techniques

Our starting point is a constant-approximation algorithm for MIS running in time $2^{O(\sqrt{n})}$ when presented with an $O(1)$-sequence, which is very unlikely to hold in general graphs by the result of Chalermsook et al. [14].

▶ **Theorem 1.** *On n-vertex graphs given with a d-sequence Max Independent Set can be $O_d(1)$-approximated in time $2^{O_d(\sqrt{n})}$.*

Our algorithm builds upon the functional equivalence between twin-width and the so-called *versatile twin-width* [5]. We defer the reader to Section 2 for a formal definition of versatile twin-width. For our purpose, one only needs to know the following useful consequence of that equivalence. From a d'-sequence of G, we can compute in polynomial time another partition sequence $\mathcal{P}_n, \dots, \mathcal{P}_1$ of G of width $d := f(d')$, for some computable function f, such that for every integer $1 \leqslant i \leqslant n$, all the i parts of $\mathcal{P}_i$ have size at most $d \cdot \frac{n}{i}$. Even if some parts of P_i can be very small, this partition is balanced in the sense that no part can be larger than d times the part size in a perfectly balanced partition. Of importance to us is $\mathcal{P}_{\lfloor \sqrt{n} \rfloor}$ when the

[3] That is, the assumption that there is a $\delta > 0$ such that n-variable 3-SAT cannot be solved in time δ^n.

number of parts ($\lfloor\sqrt{n}\rfloor$) and the size of a larger part in the partition (at most $d\frac{n}{\lfloor\sqrt{n}\rfloor} \approx d\sqrt{n}$) are somewhat level.

We can then properly color the red graph (made by the red edges on the vertex set $\mathcal{P}_{\lfloor\sqrt{n}\rfloor}$) with $d+1$ colors. Any color class X is a subset of parts of $\mathcal{P}_{\lfloor\sqrt{n}\rfloor}$ such that between two parts there are either all edges (black edge) or no edge at all (non-edge). In graph-theoretic terms, the subgraph G_X of G induced by all the vertices of all the parts of X have a simple modular decomposition: a partition of at most $\sqrt{n}$ modules each of size at most $d\sqrt{n}$. It is thus routine to compute a largest independent set of G_X essentially in time exponential in the maximum between the number of modules and the maximum size of a module, that is, in at most $d\sqrt{n}$. As one color class X^* contains more than a $\frac{1}{d+1}$ fraction of the optimum, we get our $d+1$-approximation when computing a largest independent set of G_{X^*}. Figure 2 serves as a visual summary of what we described so far.

The next step is to substitute recursive calls of our approximation algorithm to exact exponential algorithms on induced subgraphs of size $O_d(\sqrt{n})$. Following this inductive process at depth $q = 2, 3, 4, \ldots$, we degrade the approximation ratio to $(d+1)^3, (d+1)^7, (d+1)^{15}$, etc. but meanwhile we boost the running time to $2^{O_d(n^{1/4})}, 2^{O_d(n^{1/8})}, 2^{O_d(n^{1/16})}$, etc. In effect we show by induction that:

▶ **Theorem 2.** *On n-vertex graphs given with a d-sequence* Max Independent Set *has an $O_d(1)^{2^q-1}$-approximation algorithm running in time $2^{O_{d,q}(n^{2^{-q}})}$, for every integer $q \geqslant 0$.*

The following polynomial-time algorithm is a corollary of Theorem 2 choosing $q = O_{d,\varepsilon}(\log\log n)$.

▶ **Theorem 3.** *For every $\varepsilon > 0$,* Max Independent Set *can be n^{ε}-approximated in polynomial-time $O_{d,\varepsilon}(1) \cdot \log^{O_d(1)} n \cdot n^{O(1)}$ on n-vertex graphs given with a d-sequence.*

Note that the exponent of the polynomial factor is an absolute constant (not depending on d nor on ε).

We then apply our framework to Coloring and Max Induced Matching.

▶ **Theorem 4.** *For every $\varepsilon > 0$,* Coloring *and* Max Induced Matching *admit polynomial-time n^{ε}-approximation algorithms on n-vertex graphs of bounded twin-width given with an $O(1)$-sequence.*

The main additional difficulty for Coloring is that one cannot satisfactorily solve/approximate that problem on a modular decomposition by simply coloring its modules and its quotient graph. One needs to tackle a more general problem called Set Coloring. Fortunately this generalization is the fixed point we are looking for: approximating Set Coloring can be done in our framework by mere recursive calls (to itself).

For Max Induced Matching, we face a new kind of obstacle. It can be the case that no decent solution is contained in any color class X –in the chosen $d+1$-coloring of the red graph $G/\mathcal{P}_{\lfloor\sqrt{n}\rfloor}$. For instance, it is possible that any such color class X induces in G an edgeless graph, while very large induced matchings exist with endpoints in two distinct color classes. We thus need to also find large induced matchings within the black edges and within the red edges of $G/\mathcal{P}_{\lfloor\sqrt{n}\rfloor}$. This leads to a more intricate strategy intertwining the coloring of bounded-degree graphs (specifically the red graph and the square of its line graph) and recursive calls to induced subgraphs of G, and to special induced subgraphs of the total graph (i.e., made by both the red and black edges) of $G/\mathcal{P}_{\lfloor\sqrt{n}\rfloor}$. Although this is not necessary, one can observe that the latter graphs are also induced subgraphs of G itself.

We then explore the limits of our results and framework in terms of amenable problems. We give the following technical generalization to the approximation algorithms for MIS and Max Induced Matching.

▶ **Theorem 5.** *For every connected graph H and $\varepsilon > 0$, Mutually Induced H-packing admits a polynomial-time n^ε-approximation algorithms on n-vertex graphs of bounded twin-width given with an $O(1)$-sequence.*

In this problem, one seeks for a largest induced subgraph that consists of a disjoint union of copies of H. All the previous technical issues are here combined. We try all the possibilities of batching the vertices of H into at most $|V(H)|$ parts of $G/\mathcal{P}_{\lfloor\sqrt{n}\rfloor}$, based on the trigraph that these parts define. For instance with $H = K_2$ (an edge), i.e., the case of Max Induced Matching, the three possible trigraphs are the 1-vertex trigraph, two vertices linked by a red edge, and two vertices linked by a black edge. In the general case, the problem generalization is quite delicate to find. We have to keep some partitions of $V(G)$ and $V(H)$ to enforce that the copies of H in G follow a pattern that the algorithm committed to higher up in the recursion tree, and a weight function on $|V(H)|$-tuples of vertices of G, not to forget how many mutually induced copies of H can be packed *within* these vertices. The other novelty is that some recursive calls are on induced subgraphs of the total graph of $G/\mathcal{P}_{\lfloor\sqrt{n}\rfloor}$ that are *not* induced subgraphs of G. Fortunately, these graphs keep the same bound of versatile twin-width, and thus our framework allows it.

Defining, for a family of graphs $\mathcal{H}$, Mutually Induced $\mathcal{H}$-packing as the same problem where the connected components of the induced subgraph should all be in $\mathcal{H}$, we get a similar approximation factor when $\mathcal{H}$ is a finite set of connected graphs. (Note that Mutually Induced H-packing is sometimes called Independent Induced H-Packing.) In particular, we can similarly approximate Independent H-Packing, which is the same problem but the copies of H need not be induced. (Our approximation algorithms could extend to other H-packing variants without the independence requirement, but these problems can straightforwardly be $O(1)$-approximated in general graphs.)

We can handle some cases when $\mathcal{H}$ is infinite, too. For instance, by slightly adapting the case of MIS, we can get an n^ε-approximation when $\mathcal{H}$ is the set of all cliques. We show this more involved example, also expressible as Mutually Induced $\mathcal{H}$-packing for $\mathcal{H}$ the set of all trees or the set all stars.

▶ **Theorem 6.** *For every $\varepsilon > 0$, finding the induced (star) forest with the most edges admits a polynomial-time n^ε-approximation algorithms on n-vertex graphs of bounded twin-width given with an $O(1)$-sequence.*

As we already mentioned, our framework is exclusively useful for problems that are very inapproximable in general graphs; at least for which an n^ε-approximation algorithm is not known for every $\varepsilon > 0$. Are there natural such problems that cannot be approximated better in graphs of bounded twin-width? We answer this question positively with the example of Min Independent Dominating Set.

▶ **Theorem 7.** *For every $\varepsilon > 0$, Min Independent Dominating Set does not admit an $n^{1-\varepsilon}$-approximation algorithm in n-vertex graphs given with an $O(1)$-sequence, unless P=NP.*

The reduction is the same as the one for general graphs [22], but performed from a planar variant of 3-SAT. The obtained instances are not planar but can be contracted to planar trigraphs, hence overall have bounded twin-width.

Finally the case of Longest Path and Longest Induced Path is interesting. The best approximation factor for the former [18] is worse than $n^{0.99}$, while the latter is known to have the same inapproximability as MIS [31]. However an n^ε-approximation algorithm (for every $\varepsilon > 0$) is not excluded for Longest Path. We show that the property of bounded twin-width is unlikely to help for these two problems, as it would lead to better approximation algorithms for Longest Path in general graphs. This is mainly because subdividing at least $2 \log n$ times every edge of any n-vertex graph gives a graph with twin-width at most 4 [3].

▶ **Theorem 8.** *For any $r = \omega(1)$, an r-approximation for Longest Induced Path or Longest Path on graphs given with an $O(1)$-sequence would imply a $(1 + o(1))r$-approximation for Longest Path in general graphs.*

In turn, this can be used to exhibit a family $\mathcal{H}$ with an infinite antichain for the *induced subgraph* relation such that Mutually Induced $\mathcal{H}$-packing is *hard* to n^ε-approximate on graphs of bounded twin-width. The family $\mathcal{H}$ is simply the set of all paths terminated by triangles at both ends.

▶ **Theorem 9.** *There is an infinite family $\mathcal{H}$ of connected graphs such that if for every $\varepsilon > 0$, Mutually Induced $\mathcal{H}$-packing admits an n^ε-approximation algorithm on n-vertex graphs given with an $O(1)$-sequence, then so does Longest Path on general graphs.*

Table 1 summarizes our results and hints at future work.

Problem name	lower bound general graphs	upper bound bounded tww	lower bound bounded tww
Max Independent Set	$n^{1-\varepsilon}$	$\boxed{n^\varepsilon}$	?, self-improvement
Coloring	$n^{1-\varepsilon}$	$\boxed{n^\varepsilon}$	$4/3 - \varepsilon$
Max Induced Matching	$n^{1-\varepsilon}$	$\boxed{n^\varepsilon}$	?
Mut. Ind. H-Packing	$n^{1-\varepsilon}$	$\boxed{n^\varepsilon \text{ } (H \text{ connected})}$	?
Mut. Ind. $\mathcal{H}$-Packing	$n^{1-\varepsilon}$	$\boxed{n^\varepsilon \text{ for some } \mathcal{H}}$	$\boxed{\text{Longest Path-hard}}$
Min Ind. Dom. Set	$n^{1-\varepsilon}$	$n/\mathsf{polylog}(n)$	$\boxed{n^{1-\varepsilon}}$
Longest Path	$2^{\log^{1-\varepsilon} n}$	$n/\exp(\Omega(\sqrt{\log n}))$	$\boxed{\text{Longest Path-hard}}$
Longest Induced Path	$n^{1-\varepsilon}$	$n/\mathsf{polylog}(n)$	$\boxed{\text{Longest Path-hard}}$
Min Dominating Set	$(1-\varepsilon)\ln n$	$O(1)$	?

Table 1 Approximability status of graph problems in general graphs and in graphs of bounded twin-width given with an $O(1)$-sequence. Everywhere "ε" should be read as "$\forall \varepsilon > 0$". Our results are enclosed by boxes. "Longest Path-hard" means that getting an r-approximation would yield essentially the same ratio for Longest Path in general graphs. The other lower bounds are under standard complexity-theoretic assumptions, mostly P≠NP. Not to clutter the table, we do not put the references, which can all be found in the paper.

For the main highly inapproximable graph problems, we either obtain an n^ε-approximation algorithm on graphs of bounded twin-width given with an $O(1)$-sequence, or a conditional obstruction to such an algorithm. In the former case, can we improve further the approximation factor? The next theorem was observed using the self-improvement reduction of Feige et al. [17], which preserves the twin-width bound. This reduction consists of going from a graph G to the lexicographic product $G[G]$, where every vertex of G is replaced by a module inducing a copy of G (and iterating this trick).

▶ **Theorem 10** ([7]). *Let $r : \mathbb{N} \to \mathbb{R}$ be any non-decreasing function such that for every $\varepsilon > 0$, $r(n) = o(n^\varepsilon)$. If Max Independent Set admits an $r(n)$-approximation algorithm* *on n-vertex graphs of bounded twin-width given with an $O(1)$-sequence, then it further admits an $r(n)^{\varepsilon}$-approximation.*

To our knowledge, the application of the self-improvement trick is always to strengthen a lower bound, and never to effortlessly obtain a better approximation factor. Therefore, we may take Theorem 10 as a weak indication that our approximation ratio is best possible. Still, not even a polynomial-time approximation scheme (PTAS) is ruled out for MIS (nor for Max Induced Matching, Min Dominating Set, etc.) and we would like to see better approximation algorithms. For Coloring, as was previously observed [7], a PTAS is ruled out by the NP-hardness of deciding if a planar graph is 3-colorable or 4-chromatic, since planar graphs have twin-width at most 9 and a 9-sequence can be found in linear time [24].

2 Preliminaries

For i and j two integers, we denote by $[i, j]$ the set of integers that are at least i and at most j. For every integer i, $[i]$ is a shorthand for $[1, i]$.

2.1 Handled graph problems

We will consider several problems throughout the paper. We recall here the definition of the most central ones. Some technical problem generalizations will be defined along the way.

Weighted Max Independent Set (WMIS, for short)
Input: A graph G and a weight function $V(G) \to \mathbb{Q}$.
Output: A set $S \subseteq V(G)$ such that $\forall u, v \in S$, $uv \notin E(G)$ maximizing $w(S) := \sum_{v \in S} w(v)$.

A feasible solution to WMIS is called an *independent set*. The Max Independent Set (MIS, for short) problem is the particular case with $w(v) = 1$, $\forall v \in V(G)$. We may denote by $\alpha(G)$, the *independence number*, that is the optimum value of WMIS on graph G.

Coloring
Input: A graph G.
Output: A partition $\mathcal{P}$ of $V(G)$ into independent sets minimizing the cardinality of $\mathcal{P}$.

Equivalently, Coloring can be expressed as finding an integer k and a map $c : V(G) \to [k]$ such that for every $uv \in E(G)$, $c(u) \neq c(v)$, while minimizing k.

Max Induced Matching
Input: A graph G, possibly together with a weight function $w : E(G) \to \mathbb{Q}$.
Output: A set $S \subseteq E(G)$ such that $\forall uv \neq u'v' \in S$, $\{u, v\} \cap \{u', v'\} = \emptyset$ and $G[\{u, v, u', v'\}]$ has exactly two edges, maximizing $w(S) := \sum_{e \in S} w(e)$.

An *induced matching* is a pairwise disjoint set of edges (i.e., a matching) with no edge bridging them. We now give a common generalization of WMIS and Max Induced Matching.

Mutually Induced $\mathcal{H}$-packing
Input: A graph G, possibly together with a weight function $w : V(G) \to \mathbb{Q}$.
Output: A set $S \subseteq V(G)$ such that $G[S]$ is a disjoint union of graphs each isomorphic to a graph in $\mathcal{H}$, maximizing $w(S) := \sum_{v \in S} w(v)$.

When $\mathcal{H}$ consists of a single graph, say H, we simply denote the former problem Mutually Induced H-packing. WMIS and Max Induced Matching are the special cases when H is a vertex and an edge, respectively.

2.2 The contraction and partition viewpoints of twin-width

A *trigraph* G has vertex set $V(G)$, black edge set $E(G)$, red edge set $R(G)$ such that $E(G) \cap R(G) = \emptyset$ (and $E(G), R(G) \subseteq \binom{V(G)}{2}$). A *contraction* in a trigraph G replaces a pair of (non-necessarily adjacent) vertices $u, v \in V(G)$ by one vertex w that is linked to $G - \{u, v\}$ in the following way to form a new trigraph G'. For every $z \in V(G) \setminus \{u, v\}$, $wz \in E(G')$ whenever $uz, vz \in E(G)$, $wz \notin E(G') \cup R(G')$ whenever $uz, vz \notin E(G) \cup R(G)$, and $wz \in R(G')$, otherwise. The *red graph* $(V(G), R(G))$ will be denoted by $\mathcal{R}(G)$. We denote by $\mathcal{T}(G)$ the *total graph* of G defined as $(V(G), E(G) \cup R(G))$. An *induced subtrigraph* of a trigraph G is obtained by removing vertices (but no edges) to G, analogously to induced subgraphs. A partial contraction sequence of an n-vertex (tri)graph G (to a trigraph H) is a sequence of trigraphs $G = G_n, \cdots, G_t = H$ for some $t \in [n]$ such that G_i is obtained from G_{i+1} by performing one contraction. A (complete) contraction sequence is such that $t = 1$, that is, H is the 1-vertex trigraph. A *d-sequence* $\mathcal{S}$ of G is a contraction sequence of G in which the red graph of every trigraph of $\mathcal{S}$ has maximum degree at most d.

Assume that there is a partial contraction sequence from a (tri)graph G to a trigraph H. If u is a vertex of H, then $u(G) \subseteq V(G)$ denotes the set of vertices eventually contracted into u in H. We denote by $\mathcal{P}(H)$ the partition $\{u(G) : u \in V(H)\}$ of $V(G)$. If G is clear from the context, we may refer to a *part* of H as any set in $\{u(G) : u \in V(H)\}$. We will mostly see d-sequences as sequences of partitions, that is, $\mathcal{P}_n, \dots, \mathcal{P}_t$ with $\mathcal{P}_i := \{u(G) : u \in V(G_i)\}$ when $G_n, \dots, G_t$ is a partial (contraction) d-sequence.

Given a graph G and a partition $\mathcal{P}$ of $V(G)$, the *quotient graph* of G with respect to $\mathcal{P}$ is the graph with vertex set $\mathcal{P}$, where PP' is an edge if there is $u \in P$ and $v \in P'$ such that $uv \in E(G)$. Given a (tri)graph G and a partition $\mathcal{P}$ of $V(G)$, the *quotient trigraph $G/\mathcal{P}$* is the trigraph with vertex set $\mathcal{P}$, where PP' is a black edge if for every $u \in P$ and every $v \in P'$, $uv \in E(G)$, and a red edge if either there is $u \in P$ and $v \in P'$ such that $uv \in R(G)$, or there is $u_1, u_2 \in P$ and $v_1, v_2 \in P'$ such that $u_1v_1 \in E(G)$ and $u_2v_2 \notin E(G)$.

A trigraph H is a *cleanup* of another trigraph G if $V(H) = V(G)$, $R(H) \subseteq R(G)$, and $E(G) \subseteq E(H) \subseteq E(G) \cup R(G)$. That is, H is obtained from G by turning some of its red edges into black edges or non-edges. We further say that H is *full cleanup* of G if H has no red edge, and thus, is considered as a graph. Note that the total graph $\mathcal{T}(G)$ and the *black graph* $(V(G), E(G))$ of a trigraph G are extreme examples of full cleanups of G.

2.3 Balanced partition sequences

The notion of *versatile twin-width* is a crucial opening step to our algorithms; see [5]. Let us call *d-contraction* a contraction between two trigraphs of maximum red degree at most d. A *tree of d-contractions* of a trigraph G (of maximum red degree at most d) is a rooted tree, whose root is labeled by G, whose leaves are all labeled by 1-vertex trigraphs K_1, and such that one can go from any parent to any of its children by performing a single d-contraction. Observe that d-sequences coincide with trees of d-contractions that are paths. A trigraph G has *versatile twin-width d* if G admits a tree of d-contractions in which every internal node, labeled by, say, F, has at least $|V(F)|/d$ children each obtained by contracting one of a list of $|V(F)|/d$ pairwise disjoint pairs of vertices of F.

It was shown that twin-width and versatile twin-width are functionally equivalent [5]. The relevant consequence for our purposes is that every graph G with a d'-sequence admits a *balanced* d-sequence, where $d = h(d')$ depends only on d', i.e., one for which the partitions $\mathcal{P}_n, \dots, \mathcal{P}_1$ are such that for every $i \in [n]$ and $P \in \mathcal{P}_i$, $|P| \leqslant d \cdot \frac{n}{i}$. As we will resort to recursion on induced subtrigraphs and quotient trigraphs, we need to keep more information on those subinstances that the mere fact that they have twin-width at most d (otherwise the twin-width bound could quickly diverge).

This will be done by opening up the proof in [5], and handling divided $0, 1, r$-matrices with some specific properties. Thus we need to recall the relevant definitions.

Given two partitions $\mathcal{P}, \mathcal{P}'$ of the same set, we say that $\mathcal{P}'$ is a *coarsening* of $\mathcal{P}$ if every part of $\mathcal{P}$ is contained in a part of $\mathcal{P}'$, and $\mathcal{P}, \mathcal{P}'$ are distinct. Given a matrix M, we call *row division* (resp. *column division*) a partition of the rows (resp. columns) of M into parts of consecutive rows (resp. columns). A (k, ℓ)*-division*, or simply *division*, of a matrix M is a pair $(\mathcal{R} = \{R_1, \ldots, R_k\}, \mathcal{C} = \{C_1, \ldots, C_\ell\})$ where $\mathcal{R}$ is a row division and $\mathcal{C}$ is a column division. In a matrix division $(\mathcal{R}, \mathcal{C})$, each part $R \in \mathcal{R}$ is called a *row part*, and each part $C \in \mathcal{C}$ is called a *column part*. Given a subset R of rows and a subset C of columns in a matrix M, the *zone* $M[R, C]$ denotes the submatrix of all entries of M at the intersection between a row of R and a column of C. A *zone* of a matrix partitioned by $(\mathcal{R}, \mathcal{C}) = (\{R_1, \ldots, R_k\}, \{C_1, \ldots, C_\ell\})$ is any $M[R_i, C_j]$ for $i \in [k]$ and $j \in [\ell]$. A zone is *constant* if all its entries are identical, *horizontal* if all its columns are equal, and *vertical* if all its rows are equal. A *0,1-corner* is a 2×2 $0, 1$-matrix which is not horizontal nor vertical.

Unsurprisingly, $0, 1, r$*-matrices* are such that each entry is in $\{0, 1, r\}$ where r is an error symbol that should be understood as a red edge. A *neat division* of a $0, 1, r$-matrix is a division for which every zone either contains only r entries or contains no r entry and is horizontal or vertical (or both, i.e., constant). Zones filled with r entries are called *mixed*. A *neatly divided matrix* is a pair $(M, (\mathcal{R}, \mathcal{C}))$ where M is a $0, 1, r$-matrix and $(\mathcal{R}, \mathcal{C})$ is a neat division of M. A t*-mixed minor* in a neatly divided matrix is a (t, t)-division which coarsens the neat subdivision, and contains in each of its t^2 zones at least one mixed zone (i.e., filled with r entries) or a 0,1-corner. A neatly divided matrix is said t*-mixed free* if it does not admit a t-mixed minor.

A *mixed cut of a row part* $R \in \mathcal{R}$ of a neatly divided matrix $(M, (\mathcal{R}, \mathcal{C} = \{C_1, C_2, \ldots\}))$ is an index i such that both $M[R, C_i]$ and $M[R, C_{i+1}]$ are not mixed, and there is a $0, 1$-corner in the 2-by-$|R|$ zone defined by the last column of C_i, the first column of C_{i+1}, and R. The *mixed value of a row part* $R \in \mathcal{R}$ of a neatly divided matrix $(M, (\mathcal{R}, \mathcal{C} = \{C_1, C_2, \ldots\}))$ is the number of mixed zones $M[R, C_j]$ plus the number of mixed cuts between two (adjacent non-mixed) zones $M[R, C_j]$ and $M[R, C_{j+1}]$. We similarly define the mixed value of a column part $C \in \mathcal{C}$. The *mixed value of a neat division* of a $0, 1, r$-matrix is the maximum of the mixed values taken over every part. The *part size* of a division $(\mathcal{R}, \mathcal{C})$ is defined as $\max(\max_{R \in \mathcal{R}} |R|, \max_{C \in \mathcal{C}} |C|)$. A division is *symmetric* if the largest row index of each row part and the largest column index of each column part define the same set of integers. We call *symmetric fusion* of a symmetric division the fusion of two consecutive parts in $\mathcal{C}$ and of the two corresponding parts in $\mathcal{R}$. A symmetric fusion on a symmetric division yields another symmetric division. A matrix $A := (a_{i,j})_{i,j}$ is said *symmetric* in the usual sense, namely, for every entry $a_{i,j}$ of A, $a_{i,j} = a_{j,i}$.

In what follows, we set $c_d := 8/3(t+1)^2 2^{4t}$. The following definition is key.

▶ **Definition 11.** *Let $\mathcal{M}_{n,d}$ be the class of the neatly divided $n \times n$ symmetric $0, 1, r$-matrices $(M, (\mathcal{R}, \mathcal{C}))$, such that $(\mathcal{R}, \mathcal{C})$ is symmetric and has:*

- *mixed value at most $4c_d$,*
- *part size at most 2^{4c_d+2}, and*
- *no d-mixed minor.*

The *red number* of a matrix is the maximum number of r entries in a single column or row of the matrix.

▶ **Lemma 12.** *Let $(M, (\mathcal{R}, \mathcal{C})) \in \mathcal{M}_{n,d}$. The red number of M is at most $c_d \cdot 2^{4c_d+4}$. Thus, the trigraph whose adjacency matrix is M has maximum red degree at most $c_d \cdot 2^{4c_d+4}$.*

Proof. Any row or column intersects at most $4c_d$ mixed zones (filled with r entries). Each mixed zone has width and length bounded by the part size 2^{4c_d+2}. Hence the maximum total number of r entries on a single row or column is at most $4c_d \cdot 2^{4c_d+2} = c_d \cdot 2^{4c_d+4}$. ◀

A *coarsening* of a neatly divided matrix $(M, (\mathcal{R}, \mathcal{C}))$ is a neatly divided matrix $(M', (\mathcal{R}', \mathcal{C}'))$ such that $(\mathcal{R}', \mathcal{C}')$ is a coarsening of $(\mathcal{R}, \mathcal{C})$, and M' is obtained from M by setting to r all entries that lie, in M divided by $(\mathcal{R}', \mathcal{C}')$, in a zone with at least one r entry or a 0,1-corner. We also refer to the process of going from $(M, (\mathcal{R}, \mathcal{C}))$ to $(M', (\mathcal{R}', \mathcal{C}'))$ as *coarsening operation*. A coarsening operation from $(M, (\mathcal{R}, \mathcal{C})) \in \mathcal{M}_{n,d}$ to $(M', (\mathcal{R}', \mathcal{C}'))$ is said *invariant-preserving* if $(M', (\mathcal{R}', \mathcal{C}')) \in \mathcal{M}_{n,d}$.

The following lemma is the crucial building block of the current section.

▶ **Lemma 13** ([6, Lemma 18]). *We set $s := 2^{4c_d+4}$. Every neatly divided matrix $(M, (\mathcal{R}, \mathcal{C})) \in \mathcal{M}_{n,d}$ has an invariant-preserving coarsening $(M', (\mathcal{R}', \mathcal{C}')) \in \mathcal{M}_{n,d}$ with $\lfloor n/s \rfloor$ disjoint pairs of identical columns. Given $(M, (\mathcal{R}, \mathcal{C}))$, both $(M', (\mathcal{R}', \mathcal{C}'))$ and the pairs of columns can be computed in $n^{O(1)}$ time.*

In [6], it is not explicitly stated that the invariant-preserving coarsening (hence the pairs of identical columns) can be found in polynomial time. However it is easy to check that the proof is effective, since it greedily symmetrically fuses two consecutive parts, provided the resulting divided matrix remains in $\mathcal{M}_{n,d}$. A special case of the following observation is shown in [6, Lemma 19].

▶ **Lemma 14.** *Let $(M, (\mathcal{R}, \mathcal{C})) \in \mathcal{M}_{n,d}$ be a neatly divided matrix. Removing a set of h columns and the h corresponding rows, and possibly removing from the division the parts that are now empty, results in a neatly divided matrix in $\mathcal{M}_{n-h,d}$.*

Proof. By construction, the new matrix and division are symmetric. The new neatly divided matrix remains d-mixed free. The part size and the mixed value can only decrease. ◀

▶ **Lemma 15** ([6, Beginning of Lemma 20]). *Given any graph G with a d-sequence, one can find in polynomial-time an adjacency matrix M of G, such that $(M, (\mathcal{R}, \mathcal{C}))$ is a neatly divided matrix of $\mathcal{M}_{n,2d+2}$ with $(\mathcal{R}, \mathcal{C})$ the finest division of M (i.e., the one where all parts are of size 1).*

The adjacency matrix of a trigraph extends the one of a graph by putting r symbols when the vertices of the corresponding row and column are linked by a red edge. A neatly divided matrix $(M, (\mathcal{R}, \mathcal{C}))$ is said *conform* to a trigraph G if M is the adjacency matrix of a trigraph G' such that G is a cleanup of G'. Furthermore, we assume (and keep implicit) that we know the one-to-one correspondence between each row (and corresponding column) of M and vertex of G.

▶ **Lemma 16.** *Let d be a natural, $s := 2^{4c_d+4}$, and $d' := c_d \cdot 2^{4c_d+4}$. Let G be an n-vertex trigraph given with a neatly divided matrix $(M, (\mathcal{R}, \mathcal{C})) \in \mathcal{M}_{n,d}$ conform to G. A partial d'-sequence $\mathcal{S}$ from G to a trigraph H satisfying*

- *$|V(H)| = \lfloor \sqrt{n} \rfloor$, and*
- *$\forall u \in V(H), |u(G)| \leqslant s\sqrt{n}$,*

and a neatly divided matrix $(M', (\mathcal{R}', \mathcal{C}')) \in \mathcal{M}_{\lfloor \sqrt{n} \rfloor,d}$ conform to H can be computed in time $n^{O(1)}$.

Proof. This is a consequence of Lemmas 13 and 14; see the proof of the more general Lemma 18. ◀

Combining Lemmas 15 and 16, one obtains the following.

▶ **Lemma 17.** *Let d be a natural, $s := 2^{4c_d+4}$, and $d' := c_d \cdot 2^{4c_d+4}$. Given an n-vertex graph G with a d-sequence, one can compute in time $n^{O(1)}$ a partition $\mathcal{P} = \{P_1, P_2, \ldots, P_{\lfloor\sqrt{n}\rfloor}\}$ of $V(G)$ satisfying*

- *for every integer $1 \leqslant i \leqslant \lfloor\sqrt{n}\rfloor$, $|P_i| \leqslant s\sqrt{n} \leqslant d'\sqrt{n}$, and*
- *the red graph of $G/\mathcal{P}$ has maximum degree at most d'.*

We will need a stronger inductive form of Lemma 17, also a consequence of Lemmas 15 and 16.

▶ **Lemma 18.** *Let $\hat{d}$ be a natural, $d = 2\hat{d} + 2$, and set $s := 2^{4c_d+4}$, and $d' := c_d \cdot 2^{4c_d+4}$. Given an n-vertex graph G given with a $\hat{d}$-sequence, or an n-vertex trigraph G with a neatly divided matrix $(M, (\mathcal{R}, \mathcal{C})) \in \mathcal{M}_{n,d}$ such that M is conform to G, one can compute in time $n^{O(1)}$ a partition $\mathcal{P} = \{P_1, P_2, \ldots, P_{\lfloor\sqrt{n}\rfloor}\}$ of $V(G)$ with maximum red degree at most d' satisfying that, for every integer $1 \leqslant i \leqslant \lfloor\sqrt{n}\rfloor$, $|P_i| \leqslant s\sqrt{n} \leqslant d'\sqrt{n}$, and for any trigraph H that is*

- *a cleanup of an induced subtrigraph of $G/\mathcal{P}$, or*
- *an induced subtrigraph $G[\bigcup_{i \in J \subseteq [\lfloor\sqrt{n}\rfloor]} P_i]$,*

a neatly divided matrix $(M', (\mathcal{R}', \mathcal{C}')) \in \mathcal{M}_{|V(H)|,d}$ conform to H can be computed in time $n^{O(1)}$.

Proof. If we are given a graph G with a $\hat{d}$-sequence, we immediately compute a neatly divided matrix $(M, (\mathcal{R}, \mathcal{C})) \in \mathcal{M}_{n,d}$ conform to G, by Lemma 15. We then proceed as if we received the second kind of input.

We will build iteratively the partition $\mathcal{P} = \{P_1, P_2, \ldots, P_{\lfloor\sqrt{n}\rfloor}\}$ starting from the finest partition. At each step we merge two parts, until the number of parts is $\lfloor\sqrt{n}\rfloor$. At this point, we have the desired partition $\mathcal{P}$.

We iteratively maintain a trigraph G^z and a neatly divided matrix $(M^z, (\mathcal{R}^z, \mathcal{C}^z)) \in \mathcal{M}_{n-z+1,d}$ conform to it. The maintained partition is just the one corresponding to the parts of G^z. Initially, G^1 is G, and $(M^1, (\mathcal{R}^1, \mathcal{C}^1)) = (M, (\mathcal{R}, \mathcal{C})) \in \mathcal{M}_{n,d}$. At step z we do the following. We apply Lemma 13 on $(M^z, (\mathcal{R}^z, \mathcal{C}^z)) \in \mathcal{M}_{n-z+1,d}$ and obtain, in polynomial-time, an invariant-preserving coarsening $(M'^z, (\mathcal{R}'^z, \mathcal{C}'^z)) \in \mathcal{M}_{n-z+1,d}$, and $h := \lfloor (n-z+1)/s \rfloor$ disjoint pairs of equal columns $\{c_1, c'_1\}, \ldots, \{c_h, c'_h\}$ in $(M'^z, (\mathcal{R}'^z, \mathcal{C}'^z))$. Let $\{r_1, r'_1\}, \ldots, \{r_h, r'_h\}$ be the corresponding rows, and $\{v_1, v'_1\}, \ldots, \{v_h, v'_h\}$ the corresponding vertices. Observe that a coarsening of a neatly divided matrix conform to a trigraph is still conform to that trigraph, since the new matrix may only have some r entries in place of some previously 0 or 1 entries. In particular, $(M'^z, (\mathcal{R}'^z, \mathcal{C}'^z))$ is conform to G^z.

There is at least one pair $\{v_i, v'_i\}$ whose contraction forms a part of size at most n/h. Indeed, otherwise the union of the parts corresponding to $v_1, v'_1 \ldots, v_h, v'_h$ is larger than n. We remove c'_i and r'_i from $(M'^z, (\mathcal{R}'^z, \mathcal{C}'^z))$. By Lemma 14, we obtain a neatly divided matrix of $M_{n-z,d}$ that we denote by $(M^{z+1}, (\mathcal{R}^{z+1}, \mathcal{C}^{z+1}))$. As we stop when $n - z + 1 = \lfloor\sqrt{n}\rfloor$, it means that the maximum size of a part of our partition is at most $n/h \leqslant sn/\sqrt{n} = s\sqrt{n}$. The bound on the maximum red degree of the obtained partition (actually of all maintained partitions) is given by Lemma 12.

We now show to find, for any cleanup H of an induced subtrigraph of $G/\mathcal{P}$, a neatly divided matrix $(M', (\mathcal{R}', \mathcal{C}')) \in \mathcal{M}_{|V(H)|,d}$ conform to H. We first observe, as a consequence of Lemmas 13 and 14, that $(M^{\lfloor\sqrt{n}\rfloor}, (\mathcal{R}^{\lfloor\sqrt{n}\rfloor}, \mathcal{C}^{\lfloor\sqrt{n}\rfloor})) \in \mathcal{M}_{\lfloor\sqrt{n}\rfloor,d}$ is conform to $G/\mathcal{P}$. Taking an induced subgraph H' of $G/\mathcal{P}$ (i.e., removing vertices from it), we get, by removing the corresponding rows and columns in $(M^{\lfloor\sqrt{n}\rfloor}, (\mathcal{R}^{\lfloor\sqrt{n}\rfloor}, \mathcal{C}^{\lfloor\sqrt{n}\rfloor}))$ a neatly divided matrix $(M', (\mathcal{R}', \mathcal{C}')) \in \mathcal{M}_{|V(H')|,d}$ conform to H', by Lemma 14. Note finally that taking a cleanup H of H', we can simply keep $(M', (\mathcal{R}', \mathcal{C}'))$ as a neatly divided matrix of $\mathcal{M}_{|V(H)|,d}$ conform to G. The second item, concerning induced subtrigraphs $G[\bigcup_{i \in J \subseteq [\lfloor\sqrt{n}\rfloor]} P_i]$ is a simple application of Lemma 14, and works more generally for any induced subtrigraph of G. ◀

In effect, we will only apply Lemma 18 for graphs G and H, i.e., when H is an induced subgraph of G or a full cleanup of an induced subtrigraph of $G/\mathcal{P}$. Indeed, the structures H will correspond to subinstances. We want those to be graphs, so that the tackled graph problem is well-defined on them.

3 Approximation algorithms for Max Independent Set

We naturally start our study with Max Independent Set, a central problem that is very inapproximable [23, 34], and yet constitutes the textbook example of our approach.

3.1 Subexponential-time constant-approximation algorithm

We present a subexponential-time $O_d(1)$-approximation for WMIS on graphs given with a d-sequence, which we recall, is unlikely to exist in general graphs [14].

▶ **Lemma 19.** *Let d' be a natural, $s := 2^{4c_{d'}+4}$, and $d := c_{d'} \cdot 2^{4c_{d'}+4}$. Assume n-vertex inputs G, vertex-weighted by w, are given with a d'-sequence. Weighted Max Independent Set can be $(d+1)$-approximated in time $2^{O_d(\sqrt{n})}$ on these inputs.*

Proof. By Lemma 17, we compute in polynomial time a partition $\mathcal{P} = \{P_1, \dots, P_{\lfloor\sqrt{n}\rfloor}\}$ of $V(G)$ whose parts have size at most $s\sqrt{n}$ and such that $\mathcal{R}(G/\mathcal{P})$ has maximum degree at most d.

For every integer $1 \leqslant i \leqslant \lfloor\sqrt{n}\rfloor$, we compute a heaviest independent set in $G[P_i]$, say S_i. Even with an exhaustive algorithm, this takes time $\sqrt{n} \cdot s^2\sqrt{n} \cdot 2^{s\sqrt{n}} = 2^{O_d(\sqrt{n})}$. We then $(d+1)$-color (in linear time) $\mathcal{R}(G/\mathcal{P})$, which is possible since this graph has maximum degree at most d. This defines a coarsening of $\mathcal{P}$ in $d+1$ parts $\mathcal{Q} = \{C_1, \dots, C_{d+1}\}$. Thus, $\mathcal{Q}$ is a partition of $V(G)$ such that C_j consists of all the parts $P_i \in \mathcal{P}$ receiving color j in the $(d+1)$-coloring of $\mathcal{R}(G/\mathcal{P})$.

For every $j \in [d+1]$, let H_j be the *graph* $(G/\mathcal{P})[C_j]$[4] vertex-weighted by $P_i \subseteq C_j \mapsto w(S_i)$. Note that $(G/\mathcal{P})[C_j]$ can indeed be assimilated to a graph, since it has, by design, no red edge. We compute a heaviest independent set in H_j, say R_j. This takes time $(d+1) \cdot n \cdot 2^{\sqrt{n}} = 2^{O_d(\sqrt{n})}$. We output $\bigcup_{P_i \subseteq R_j} S_i$ for the index $j \in [d+1]$ maximizing $\sum\limits_{P_i \subseteq R_j} w(S_i)$.

This finishes the description of the algorithm. We already argued that its running time is $2^{O_d(\sqrt{n})}$. We shall justify that it does output an independent set of weight at least a $\frac{1}{d+1}$ fraction of the optimum $\alpha(G)$.

***I* is indeed an independent set.** For any $j \in [d+1]$, consider two vertices $x, y \in \bigcup_{P_i \subseteq R_j} S_i$. If $\{x, y\} \in S_i$ for some i, then x and y are non-adjacent since S_i is an independent set of $G[P_i]$. Else $x \in S_i$ and $y \in S_{i'}$ for some $i \neq i'$. P_i and $P_{i'}$ are not linked by a black edge in $(G/\mathcal{P})[C_j]$ since R_j is an independent set in H_j, nor they can be linked by a red edge (there are none in $(G/\mathcal{P})[C_j]$). Thus again, x and y are non-adjacent in G.

[4] We use this notation as a slight abuse of notation for $(G/\mathcal{P})[\{P_i : P_i \subseteq C_j\}]$.

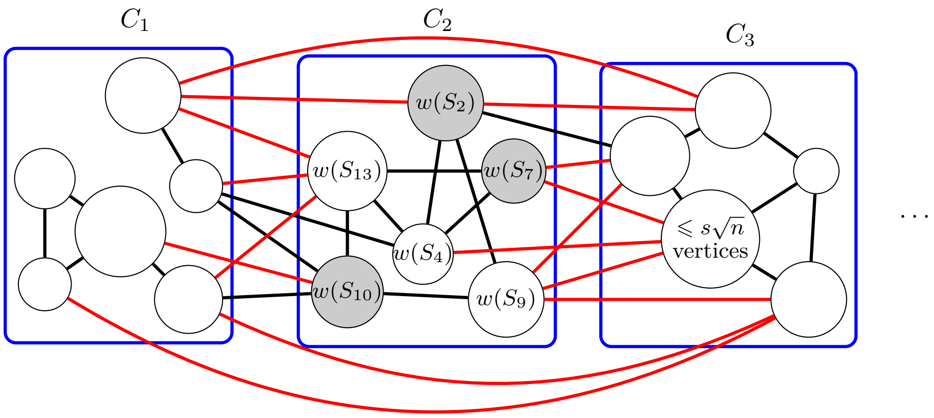


Figure 2 The trigraph $G/\mathcal{P}$ with its $\lfloor\sqrt{n}\rfloor$ vertices, each corresponding to a subset of at most $s\sqrt{n}$ vertices of G. The weights $w(S_i)$ of heaviest independent sets S_i of $G[P_i]$ for each part P_i of the color class C_2 of the $d+1$-coloring of $\mathcal{R}(G/\mathcal{P})$. A heaviest independent set in the so-weighted $(G/\mathcal{P})[C_2]$ (shaded) corresponds to an optimum solution in $G[\bigcup_{P_i \subseteq C_2} P_i]$. One of these $d+1$ independent sets is a $d+1$-approximation.

I **has weight at least** $\frac{\alpha(G)}{d+1}$. We claim that $\bigcup_{P_i \subseteq R_j} S_i$ is a heaviest independent set of $G[C_j]$. Note that the P_is that are included in C_j (and partition it) form a module partition of $G[C_j]$. In particular, any heaviest independent set intersecting some $P_i \subseteq C_j$ has to contain a heaviest independent of $G[P_i]$. This is precisely what the algorithm computes. Then a heaviest independent set in $G[C_j]$ packs such subsolutions to maximize the total weight, which is what is computed in H_j.

We conclude by the pigeonhole principle, since a heaviest independent set X of G is such that $w(X \cap C_j) \geqslant \frac{\alpha(G)}{d+1}$ for some $j \in [d+1]$. ◀

3.2 Improving the approximation factor

We notice in this short section that the approximation factor of Lemma 19 can be improved using the notion of *clustered coloring*. The *clustered chromatic number* of a class of graphs is the smallest integer k such that there is a constant c for which all the graphs of the class can be k-colored such that every color class induces a subgraph whose connected components have size at most c. A proper coloring is a particular case of clustered coloring when $c = 1$.

Instead of properly coloring the red graph, as we did in the proof of Lemma 19, we could use less colors and allow for small monochromatic components (in place of monochromatic components of size 1). We use for that the following bound due to Alon et al.

▶ **Theorem 20** ([1]). *The class of graphs of maximum degree at most d has clustered chromatic number at most* $\lceil \frac{d+2}{3} \rceil$.

We can use this lemma to improve our approximation algorithms.

▶ **Theorem 21.** *On inputs as in Lemma 19 with* $s := 2^{4c_{d'}+4}$*, and* $d := c_{d'} \cdot 2^{4c_{d'}+4}$*, Weighted Max Independent Set further admits an* $\lceil \frac{d+2}{3} \rceil$*-approximation algorithm in time* $2^{O_d(\sqrt{n})}$.

Proof. Again, we compute in polynomial time a partition $\mathcal{P} = \{P_1, \ldots, P_{\lfloor\sqrt{n}\rfloor}\}$ of $V(G)$ whose parts have size at most $s\sqrt{n}$ and such that $\mathcal{R}(G/\mathcal{P})$ has maximum degree at most d, using Lemma 17. Let c be the constant such that $\mathcal{R}(G/\mathcal{P})$ admits a clustered coloring using $\lceil \frac{d+2}{3} \rceil$ colors such that each color class C_j (with $j \in [\lceil \frac{d+2}{3} \rceil]$) is such that the connected components $C_j^1, C_j^2, \ldots, C_j^{h_j} \subseteq C_j$ of $\mathcal{R}(G/\mathcal{P})[C_j]$ have size at most c each. This coloring is guaranteed to *exist* by Theorem 20. Due to the overall running time, we might as well compute it by exhaustive search, in time $2^{O_d(\sqrt{n})}$.

For every $j \in \lceil \frac{d+2}{3} \rceil$ and $h \in [h_j]$, we denote by $P_1(C_j^h), \ldots, P_{c(j,h)}(C_j^h)$ the $c(j,h) \leqslant c$ parts $P_i \in \mathcal{P}$ that are included in C_j^h. For every $j \in \lceil \frac{d+2}{3} \rceil$, every $h \in [h_j]$, and every $J \subseteq [c(j,h)]$, we compute a heaviest independent set in $G[\bigcup_{z \in J} P_z(C_j^h)]$, which we denote by $S_{j,h,J}$. This takes time $O(\sqrt{n} \cdot 2^c \cdot 2^{sc\sqrt{n}}) = 2^{O_d(\sqrt{n})}$ since $|\bigcup_{z \in J} P_z(C_j^h)| \leqslant c \cdot s\sqrt{n}$.

For each C_j, in time $(2^c)^{\sqrt{n}} = 2^{c\sqrt{n}}$, we exhaustively try all subsets $X \subseteq \bigcup_{P_i \in C_j} P_i$ that are unions of $S_{j,h,J}$ filtering them out when $G[X]$ is not edgeless, and keep a heaviest of them, say R_j. Since there can only be black edges or non-edges between some $P_i \in C_j^h$ and $P_{i'} \in C_j^{h'}$ with $h \neq h'$, it is clear that a heaviest independent set of $G[\bigcup_{P_i \in C_j} P_i]$ is indeed a union of $S_{j,h,J}$ (with fixed j). We output a heaviest set among the R_js, which is the desired $\lceil \frac{d+2}{3} \rceil$-approximation. The running time is as claimed. ◀

3.3 Time-approximation trade-offs

Lemma 19 and Theorem 21 run exhaustive algorithms on induced subgraphs of size $O_d(\sqrt{n})$. As such, the latter inputs keep the same twin-width upper bound. To speed up the algorithm (admittedly while worsening the approximation factor) it is tempting to recursively call our very algorithm. We show that this leads to a time-approximation trade-off parameterized by an integer $q = 0, \ldots, O_d(\log \log n)$. At one end of this discrete curve, one finds the exact exponential algorithm ($q = 0$), and more interestingly the $d+1$-approximation in time $2^{O_d(\sqrt{n})}$ ($q = 1$), while at the other end lies a polynomial-time algorithm with approximation factor n^ε, where $\varepsilon > 0$ can be made as small as desired.

As we will deal with the same kind of recursions for several problems, we show the following generic abstraction.

▶ **Lemma 22.** *Let $\hat{d}$ be a natural, $d' = 2\hat{d}+2$, and $d := c_{d'} \cdot 2^{4c_{d'}+4}$. Let Π be an optimization graph problem where inputs come with a $\hat{d}$-sequence of their n-vertex graph G, or with a neatly divided matrix $(M, (\mathcal{R}, \mathcal{C})) \in \mathcal{M}_{n,d'}$ conform to G. Let $\mathcal{P}$ be the partition of $V(G)$ given by Lemma 18. Assume that*

1. *Π can be exactly solved in time $2^{O(n)}$, and there are constants c_1, c_2, c_3, and a function $f \geqslant 1$ such that*
2. *a $d^{c_3} r^2$-approximation of Π on G can be built in time n^{c_2} by using at most n^{c_1} calls to an r-approximation of Π –or another optimization problem Π' already satisfying the conclusion of the lemma– on an induced subgraph of G with at most $f(d)\sqrt{n}$ vertices or a full cleanup of an induced subtrigraph of $G/\mathcal{P}$ (on at most $\sqrt{n}$ vertices).*

Then Π can be $d^{c_3(2^q-1)}$-approximated in time

$$(f(d)^q n)^{(2-2^{-q})(c_1+c_2)} \cdot 2^{f(d)^{2(1-2^{-q})} n^{2^{-q}}},$$

for any non-negative integer q.

Proof. The proof is by induction on q. The case $q = 0$ is implied by Item 1. The case $q = 1$, and the induction step in general, is nothing more than an abstraction of Lemma 19, where exhaustive algorithms are replaced by recursive calls.

For any $q \geqslant 0$, we assume that Π can $d^{c_3(2^q-1)}$-approximated in the claimed running time, and show the same statement for the value $q+1$. Following Item 2, we run this algorithm –or one for another optimization problem Π' satisfying the conclusion of the lemma– at most n^{c_1} times on $f(d)\sqrt{n}$-vertex induced subgraphs of the input graph G or on full cleanups of induced subtrigraphs of $G/\mathcal{P}$. The latter graphs have at most $\sqrt{n} \leqslant f(d)\sqrt{n}$ vertices. By Lemma 18, we can compute in polynomial time a neatly divided matrix $(M', (\mathcal{R}', \mathcal{C}')) \in \mathcal{M}_{|V(H)|,d'}$ conform to H, for each graph H of a recursive call; hence the induction applies.

Overall this takes time at most

$$n^{c_1} + n^{c_2} \cdot \left((f(d)^q \cdot f(d)\sqrt{n})^{(2-2^{-q})(c_1+c_2)} \cdot 2^{f(d)^{2(1-2^{-q})}(f(d)\sqrt{n})^{2^{-q}}} \right)$$

$$\begin{aligned}
&\leqslant (f(d)^{q+1}n)^{c_1+c_2+\frac{1}{2}(2-2^{-q})(c_1+c_2)} \cdot 2^{f(d)^{2(1-2^{-q})+2^{-q}} n^{\frac{2^{-q}}{2}}} \\
&= (f(d)^{q+1}n)^{(2-\frac{2^{-q}}{2})(c_1+c_2)} \cdot 2^{f(d)^{2-2^{-q+1}+2^{-q}} n^{2^{-(q+1)}}} \\
&= (f(d)^{q+1}n)^{(2-2^{-(q+1)})(c_1+c_2)} \cdot 2^{f(d)^{2(1-2^{-(q+1)})} n^{2^{-(q+1)}}}.
\end{aligned}$$

For the first inequality, we assume that the two summands are larger than 2, so their sum can be bounded by their product.

Besides we get an approximation of factor at most $(d^{c_3(2^q-1)})^2 d^{c_3} = d^{c_3(2^{q+1}-1)}$. ◀

In more legible terms we have proved that:

▶ **Lemma 23.** *Problems Π satisfying the assumptions of Lemma 22 can be $d^{O(1)(2^q-1)}$-approximated in time $2^{O_{d,q}(\sqrt[2^q]{n})}$, for any non-negative integer q.*

If most graph problems admit single-exponential algorithms, we will deal with such a problem that is only known to be solvable in time $2^{O(n \log n)}$. Therefore we prove a variant of Lemma 22 with a slightly worse running time.

▶ **Lemma 24.** *Let Π be solvable in time $2^{O(n \log n)}$ and satisfy the second item of Lemma 22. Then Π can be $d^{c_3(2^q-1)}$-approximated in time*

$$2^{\left((c_1+c_2)(2-2^{-q}) \log f(d) + f(d)^{2(1-2^{-q})} n^{2^{-q}}\right) \log n},$$

for any non-negative integer q.

Proof. We follow the proof of Lemma 22 when the induction now gives a running time of

$$\begin{aligned}
&n^{c_2} + n^{c_1} \cdot 2^{\left((c_1+c_2)(2-2^{-q}) \log f(d) + (f(d)\sqrt{n})^{2^{-q}}\right) \log(f(d)\sqrt{n})} \\
&\leqslant 2^{\left((c_1+c_2)(2-2^{-(q+1)}) \log f(d) + f(d)^{2(1-2^{-(q+1)})} n^{2^{-(q+1)}}\right) \log n}.
\end{aligned}$$

◀

Again the previous lemma can be rewritten as:

▶ **Lemma 25.** *Problems Π satisfying the assumptions of Lemma 24 can be $d^{O(1)(2^q-1))}$-approximated in time $2^{O_{d,q}(\sqrt[2^q]{n} \log n)}$, for any non-negative integer q.*

We derive from Lemma 24 the following notable regimes.

▶ **Theorem 26.** *Problems Π satisfying the assumptions of Lemma 24 admit polynomial-time n^{ε}-approximation algorithms, for any $\varepsilon > 0$.*

Proof. This is the particular case $q = \lceil \log \frac{\varepsilon \log n}{2c_3 \log d} \rceil$.

Indeed the approximation factor is then at most $d^{c_3(2^q-1)} \leqslant d^{2c_3 \frac{\varepsilon \log n}{2c_3 \log d}} = 2^{\varepsilon \log n} = n^\varepsilon$, while the running time is at most

$$2^{\left((c_1+c_2)(2-2^{-q}) \log f(d) + f(d)^{2(1-2^{-q})} n^{2^{-q}}\right) \log n} \leqslant 2^{\left(2(c_1+c_2) \log f(d) + f(d)^2 n^{\frac{2c_3 \log d}{\varepsilon \log n}}\right) \log n}$$
$$= n^{2(c_1+c_2) \log f(d) + f(d)^2 d^{\frac{2c_3}{\varepsilon}}}.$$

If further Π can be solved exactly in time $2^{O(n)}$ (hence satisfies the assumptions of Lemma 22), one obtains a better running time, where the exponent of n does not depend on ε. Indeed,

$$(f(d)^q n)^{(2-2^{-q})(c_1+c_2)} 2^{f(d)^{2(1-2^{-q})} n^{2^{-q}}} \leqslant \left(\frac{\varepsilon \log n}{c_3 \log d}\right)^{2(c_1+c_2) \log f(d)} 2^{f(d)^2 d^{\frac{2c_3}{\varepsilon}}} n^{2(c_1+c_2)}. \blacktriangleleft$$

▶ **Theorem 27.** *Problems Π satisfying the assumptions of Lemma 22, resp. Lemma 24, admit a $\log n$-approximation algorithm running in time $2^{O_d(n^{\frac{1}{\log \log n}})}$, resp. $2^{O_d(n^{\frac{1}{\log \log n}} \log n)}$.*

Proof. This is the particular case $q = \lfloor \log \left(\frac{\log \log n}{c_3 \log d} + 1\right) \rfloor$.

This value is computed such that the approximation factor $d^{c_3(2^q-1)}$ is at most $\log n$. It can be easily checked that the running times are as announced. ◀

We derive the following for Weighted Max Independent Set.

▶ **Theorem 28.** *Weighted Max Independent Set on n-vertex graphs G (vertex-weighted by w) given with a d'-sequence satisfies the assumptions of Lemma 22. In particular, this problem admits*

- *a $(d+1)^{2^q-1}$-approximation in time $2^{O_{d,q}(n^{2^{-q}})}$, for every integer $q \geqslant 0$,*
- *an n^ε-approximation in polynomial-time $O_{d,\varepsilon}(1) \log^{O_d(1)} n \cdot n^{O(1)}$, for any $\varepsilon > 0$, and*
- *a $\log n$-approximation in time $2^{O_d(n^{\frac{1}{\log \log n}})}$,*

with $d := c_{2d'+2} \cdot 2^{4c_{2d'+2}+4}$.

Proof. Even the exhaustive algorithm exactly solves WMIS in time $2^{O(n)}$. We thus focus on showing that WMIS satisfies the second item of Lemma 22. We set $c_1 \geqslant 1$ as the required exponent to turn a d'-sequence into a neatly divided matrix of $\mathcal{M}_{n,2d'+2}$ conform to G, $c_2 = \frac{1}{2} + \eta$ for any fixed $\eta > 0$, the appropriate $1 < c_3 \leqslant 2$, and $f(d) = d \geqslant 1$.

The algorithm witnessing the second item is simply the proof of Lemma 19. We first check that this algorithm makes $\lfloor \sqrt{n} \rfloor + d + 1$ recursive calls on induced subgraphs of the input G: each of the $\lfloor \sqrt{n} \rfloor$ graphs $G[P_i]$ where P_i has indeed size at most $O_d(\sqrt{n})$, and each of the $d+1$ graphs $(G/\mathcal{P})[C_j]$ (indeed an induced subgraph of G by definition of the black graph of a trigraph) on at most $\sqrt{n}$ vertices.

We finally assume that each recursive call outputs an r-approximation of WMIS. Let $j \in [d+1]$ be such that $w(C_j \cap I) \geqslant \frac{1}{d+1} w(I)$ for I a heaviest independent set of G vertex-weighted by w. Let $J \subseteq [\lfloor \sqrt{n} \rfloor]$ be the indices of the P_is that are intersected by $C_j \cap I$, that is, $J = \{i \ : \ P_i \cap (C_j \cap I) \neq \emptyset\}$. For every $i \in J$, set $w_i = w(P_i \cap I)$. Each recursive call on some P_i with $i \in J$, yields an independent set of weight at least $\frac{w_i}{r}$, by assumption. Thus the weights that our algorithm puts on $(G/\mathcal{P})[C_j]$ are such that it has an independent set of weight at least $\Sigma_{i \in J} \frac{w_i}{r} = \frac{w(C_j \cap I)}{r}$. As we run an r-approximation on this graph, we get an independent set of weight at least $\frac{w(C_j \cap I)}{r^2} \geqslant \frac{w(I)}{(d+1)r^2}$. Thus WMIS satisfies the assumptions of Lemma 22, and we conclude. ◀

4 Finding the suitable generalization: the case of Coloring

In this section, we deal with the Coloring problem. Unlike for WMIS, we cannot solely resort to recursively calling our Coloring algorithm on smaller graphs. The right problem generalization needs to be found for the inductive calls to work through, and it happens to be Set Coloring.

In the Set Coloring problem, the input is a couple (G, b) where G is a graph, and b is a function assigning a positive integer to each vertex of G. The goal is to find, for each $v \in V(G)$, a set S_v of at least $b(v)$ colors such that $S_u \cap S_v = \emptyset$ whenever $uv \in E(G)$, and minimizing $|\cup_{v \in V(G)} S_v|$. Let $\chi_b(G)$ be the optimal value of Set Coloring for (G, b). Observe that Coloring corresponds to the case where $b(v) = 1$ for every $v \in V(G)$.

▶ **Theorem 29.** *Set Coloring (and hence Coloring) on n-vertex graphs G given with a d'-sequence satisfies the assumptions of Lemma 24. In particular, this problem admits*

- *a $(d + 1)^{2^q-1}$-approximation in time $2^{O_{d,q}(n^{2^{-q}} \log n)}$, for every integer $q \geqslant 0$, and*
- *an n^ε-approximation in polynomial-time for any $\varepsilon > 0$.*

with $d := c_{2d'+2} \cdot 2^{4c_{2d'+2}+4}$.

Proof. It is known [32] that Set Coloring can be solved using the inclusion-exclusion principle in time $O^*(\max_{v \in V(G)} b(v)^n) = 2^{O(n \log n)}$. We now prove that it satisfies the second item of Lemma 22. We denote by $\mathcal{A}$ the r-approximation algorithm of the statement, which we will use on instances of Set Coloring. In particular, we will call it at most $\sqrt{n} + 1$ times, and will obtain at the end a $(d + 1)r^2$-approximation on our input (G, b) in polynomial time.

We first apply Lemma 18 to get, in polynomial-time, a partition $\mathcal{P} = \{P_1, \ldots, P_{\lfloor \sqrt{n} \rfloor}\}$ of $V(G)$ whose parts have size at most $d\sqrt{n}$ and such that $R(G/\mathcal{P})$ has maximum degree at most d. For every $i \in [\lfloor \sqrt{n} \rfloor]$, we use $\mathcal{A}$ to compute an r-approximated solution c_{P_i} of $(G[P_i], b_{|P_i})$. We denote by b' the function which assigns, to each P_i, the number of colors of c_{P_i}. We now compute, in polynomial-time, a proper $(d + 1)$-coloring of $R(G/\mathcal{P})$, which defines the sets $C_1, \ldots, C_{d+1}$. For each $j \in [d + 1]$, we construct another Set Coloring instance consisting of the graph $H_j = (G/\mathcal{P})[C_j]$ (recall that this trigraph has no red edge, and can thus be seen as a graph), together with the function $b'_{|C_j}$. Again we use $\mathcal{A}$ to compute an r-approximated solution on $(H_j, b'_{|C_j})$. We denote by c_H this solution. Let G_j be the subgraph of G induced by $\cup_{P_i \in C_j} P_i$, and b_j the restriction of b to $V(G_j)$. We now show how to construct a solution c_j of Set Coloring to (G_j, b_j) from c_H and all c_{P_i}. Recall that for every $P_i \in C_j$, every $v \in P_i$, we have that $c_{P_i}(v)$ is a subset of $\{1, \ldots, b'(P_i)\}$ of size at least $b(v)$, and that $c_H(P_i)$ is a subset of size at least $b'(P_i)$. Hence, for each $P_i \in C_j$, one can choose an arbitrary bijection τ from $\{1, \ldots, b'(P_i)\}$ to $c_H(P_i)$, and define to each vertex $v \in P_i$ the set $c_j(v)$ as $\{\tau(x) : x \in c_{P_i}(v)\}$.

By construction, this solution is a feasible one for the instance (G_j, b_j). Let us prove that it is an r^2-approximation of $\chi_{b_j}(G_j)$. First, by definition of c_H, our solution uses at most $r \cdot \chi_{b'_{|C_j}}(H_j)$ colors. Then, by definition of c_{P_i} for every $P_i \in C_j$, we have $b'_{C_j}(P_i) \leqslant r \cdot \chi_{b_{|P_i}}(G[P_i])$. Now, denote by Γ the function which assigns to each $P_i \in C_j$ the number $\chi_{b_{|P_i}}(G[P_i])$. We now use the following claim, whose proof is left to the reader.

▷ Claim 30. Let (G, b) be an instance of Set Coloring, and $r \in \mathbb{R}_+$. It holds that $\chi_{r \cdot b}(G) \leqslant r \cdot \chi_b(G)$, where $r \cdot b$ is the function which assigns $r \cdot b(v)$ to each $v \in V(G)$.

This implies $\chi_{b'_{|C_j}}(H_j) \leqslant r \cdot \chi_\Gamma(H_j)$, and thus our solution uses at most $r^2 \cdot \chi_\Gamma(H_j)$ colors. We now prove the following claim.

▷ Claim 31. $\chi_{\Gamma}(H_j) \leqslant \chi_{b_j}(G_j)$.

Proof of the claim. Let c be an optimal solution for (G_j, b_j). For every distinct P_i, $P_{i'} \in C_j$ such that $P_iP_{i'}$ is an edge of H_j, it holds that there are all possible edges between P_i and $P_{i'}$ in G_j (by definition of the coloring $C_1, \cdots, C_{d+1}$), hence it holds that $\bigcup_{v \in P_i} c(v)$ and $\bigcup_{v \in P_{i'}} c(v)$ have empty intersection. Moreover, by definition of Γ, we have that $\bigcup_{v \in P_i} c(v)$ is of size at least $\Gamma(P_i)$, hence the function which assigns $\bigcup_{v \in P_i} c(v)$ to each P_i is a feasible solution for (H_j, Γ) using at most $\chi_{b_j}(G_j)$ colors. ◀

We now have in hand an r^2-approximated solution of (G_j, b_j) for every $j \in [d+1]$, which can be turned into a $(d+1)r^2$-approximated solution of (G, b), as desired. ◀

5 Edge-based problems: the case of Max Induced Matching

So far, we only considered problems where approximated solutions in each part P_i of a partition $\mathcal{P}$ of $V(G)$ of small width, and in some selected induced subgraphs of $(V(G/\mathcal{P}), E(G/\mathcal{P}))$, were enough to build an approximated solution for G.[5] We now handle problems for which a number of edges is to be optimized. Now all competitive solutions can integrally lie in between pairs of parts P_i, P_j linked by a black or a red edge in $G/\mathcal{P}$. This complicates matters, and forces us to be competitive there as well, naturally splitting the algorithm into three subroutines.

We present the algorithms for Max Subset Induced Matching where one is given, in addition to the input graph G (possibly with edge weights), a subset $Y \subseteq E(G)$, and the goal is to find a heaviest induced matching S of G such that $S \subseteq Y$. Then Max Induced Matching is the particular case when $Y = E(G)$. Of course, we could solely use the edge weights to emulate Y (by giving negative weights to all the edges in $E(G) \setminus Y$). We believe this formalism is slightly more convenient for the reader to quickly and explicitly identify where our algorithm is seeking mutually induced edges.

Since the case of Max Induced Matching is more involved than were the treatment of MIS and Coloring, we again split the arguments into the design of a subexponential-time constant-approximation algorithm (Lemma 34) followed by how this algorithm meets the requirements of Lemma 22 (Theorem 33).

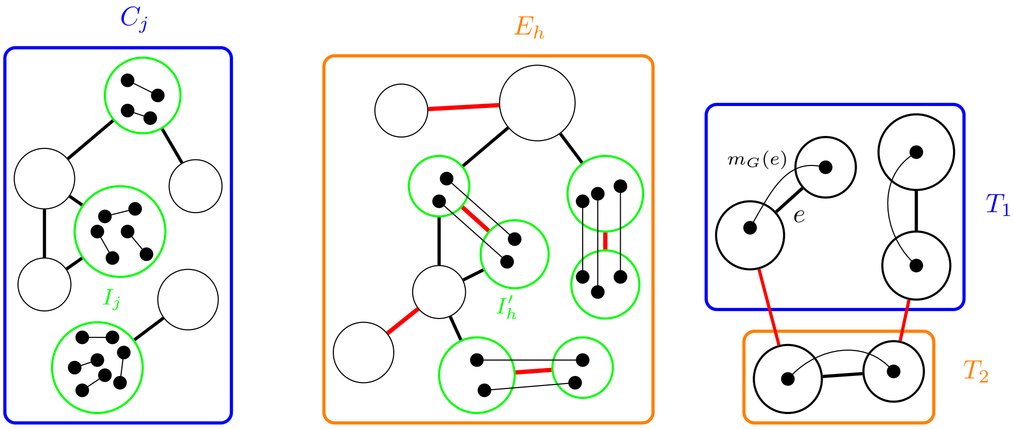
▶ **Lemma 32.** *Assume every input (G, Y) is given with a d'-sequence of the n-vertex, edge-weighted by w, graph G. We set $d := c_{d'} \cdot 2^{4c_{d'}+4}$, and $s := 2^{4c_{d'}+4}$. Max Subset Induced Matching can be $O(d^2)$-approximated in time $2^{O_d(\sqrt{n})}$ on these inputs.*

Proof. Again, by Lemma 17, we start by computing in polynomial time a partition of $V(G)$, $\mathcal{P} = \{P_1, \ldots, P_{\lfloor \sqrt{n} \rfloor}\}$, of parts with size at most $s\sqrt{n}$ and such that $\mathcal{R}(G/\mathcal{P})$ has maximum degree at most d.

We $(d+1)$-color $\mathcal{R}(G/\mathcal{P})$, which defines a coarsening $\{C_1, \ldots, C_{d+1}\}$ of $\mathcal{P}$. We also distance-2-edge-color $\mathcal{R}(G/\mathcal{P})$ with $z = 2(d-1)d+1$ colors, that is, properly (vertex-)color the square of its line graph. Observe that $z-1$ upperbounds the maximum degree of the square of the line graph of $\mathcal{R}(G/\mathcal{P})$. This partitions the edges of $\mathcal{R}(G/\mathcal{P})$ into $\{E_1, \ldots, E_z\}$. For each red edge $e = P_iP_j \in R(G/\mathcal{P})$, we denote by $p(e)$ the set $P_i \cup P_j$. We also set $X_h = p(E_h) = \bigcup_{e \in E_h} p(e)$ for each $h \in [z]$.

Let $M \subseteq Y$ be a fixed (unknown) heaviest induced matching of G contained in Y. Let M_v, M_r, M_b partition M, where M_v (as **v**ertex) consists of the edges of M with both

[5] The improvement based on clustered coloring slightly departed from that simple scheme.



(a) Computing R_j consists first of determining the heaviest induced matching in each part P_i and then, for color C_j, to compute the maximum independent set I_j (in green) weighted by the size of the matchings.

(b) Color E_h reveals a set of red edges from trigraph $G/\mathcal{P}$. Set R'_h corresponds to the heaviest matching among these edges which is mutually induced regarding the black edges. The weight of the red edges e is $w(S'_e)$.

(c) An example of set S of size 3 with two colors T_1 and T_2. The induced matching R''_i for color T_i is obtained by considering the maximum-weighted edge $m_G(e)$ between the two parts of e.

Figure 3 Illustration of how to determine the induced matching N_v, N_r, and N_b (in that order, from left to right).

endpoints in a same P_i, M_r (as **r**ed) corresponds to edges of M between some P_i and P_j with $P_iP_j \in R(G/\mathcal{P})$, and M_b (as **b**lack), the edges of M between some P_i and P_j with $P_iP_j \in E(G/\mathcal{P})$. We compute three induced matchings $N_v, N_r, N_e \subseteq Y$ of G, capturing a positive fraction of M_v, M_r, M_e, respectively. Figure 3 gives the intuition of the procedures which determine each of these approximated solutions.

Computing N_v. For every integer $1 \leqslant i \leqslant \lceil\sqrt{n}\rceil$, we compute a heaviest induced matching in $G[P_i]$ contained in Y, say S_i, in time $2^{O_d(\sqrt{n})}$. For each $j \in [d+1]$, let H_j be the *graph* $(G/\mathcal{P})[C_j]$ with every vertex $P_i \in C_j$ weighted by $w(S_i)$. We compute a heaviest independent set I_j in H_j, also in time $2^{O_d(\sqrt{n})}$.

Let R_j be the induced matching $\{e \in S_i : P_i \in I_j\}$. It is indeed an induced matching in G contained in Y, since each S_i is so, there is no red edge in $(G/\mathcal{P})[C_j]$, and I_j is an independent set of H_j. The solution N_v is then a heaviest among the R_js.

Computing N_r. For each $e = P_iP_j \in R(G/\mathcal{P})$, we compute a heaviest induced matching S'_e in $G[p(e)] = G[P_i \cup P_j]$ among those that are included in Y and have only edges with one endpoint in P_i and the other endpoint in P_j. This takes times at most $\frac{\sqrt{n}d}{2} \cdot 2^{O_d(\sqrt{n})} = 2^{O_d(\sqrt{n})}$ by trying out all vertex subsets, since $|P_i \cup P_j| \leqslant 2s\sqrt{n}$. For each $h \in [z]$, let H'_h be the *graph* $(G/\mathcal{P})[\{P_i : P_i \text{ is incident to an edge } e \in E_h\}]$ and the red edges $e \in E_h$ are turned black and get weight $w(S'_e)$. We compute a heaviest induced matching I'_h in H'_h among those *included in* E_h, in time $2^{O_d(\sqrt{n})}$. Note here that we changed the prescribed set of edges Y to E_h.

Let R'_h be the induced matching $\{f \in S'_e : e \in I'_h\} \subseteq Y$ of G. Indeed, each $S'_e \subseteq Y$ is an induced matching, and there is no red edge between an endpoint of $e \in I'_h$ and an endpoint of $e' \neq e \in I'_h$ (since E_h is a color class in a distance-2-edge-coloring of $\mathcal{R}(G/\mathcal{P})$), nor a black edge (by virtue of I'_h being an induced matching of H'_h). The solution N_r is then a heaviest among the R'_hs.

Computing N_b. Observe first that an induced matching of G can only contain at most one edge between P_i and P_j when $P_iP_j \in E(G/\mathcal{P})$. Thus in the graph $(V(G/\mathcal{P}), E(G/\mathcal{P}))$, we give weight $\max\{w(f) : f = uv \in Y, u \in P_i, v \in P_j\}$, with the convention that $\max \emptyset = -1$, to each edge $e = P_iP_j \in E(G/\mathcal{P})$, call G' the resulting edge-weighted graph, and denote by $m_G(e)$ an edge $f \in Y$ realizing this maximum. We compute a heaviest induced matching S of G' *included in* $E(G')$, in time $2^{O_d(\sqrt{n})}$. Let H_S be the graph with vertex set S, and an edge between e and e' whenever there is a red edge in $G/\mathcal{P}$ between an endpoint of e and an endpoint of e'. As H_S has degree at most $2d$, it can be $2d+1$-colored; let $T_1, \ldots, T_{2d+1}$ the corresponding color classes.

For each $i \in [2d+1]$, let R''_i be the induced matching $\{m_G(e) : e \in T_i\} \subseteq Y$ of G. Indeed, S is an induced matching in the black graph of $G/\mathcal{P}$, and the underlying vertices of T_i do not induce any red edge in $G/\mathcal{P}$, by design. The solution N_r is then a heaviest among the R''_is.

We finally output a heaviest set among N_v, N_r, N_b. The overall running time is $2^{O_d(\sqrt{n})}$ as we make a polynomial number of calls to (exhaustive) subroutines on graphs with $O_d(\sqrt{n})$ vertices, and color in linear time $O(n)$-vertex graphs of maximum degree Δ with $\Delta + 1$ colors. We already argued that $N_v, N_r, N_b \subseteq Y$ are all induced matchings in G, thus so is our output.

We shall just show that we meet the claimed approximation factor. First, one can observe $w(N_v) \geqslant \frac{w(M_v)}{d+1}$. Second, at least a $\frac{1}{z}$ fraction of the weight of M_r intersects some fixed E_i (with $i \in [z]$). Let $\mathcal{J}$ be the parts of $\mathcal{P}$ intersected by $M_r \cap X_i$. As there cannot be a black edge between two parts of $\mathcal{J}$ (otherwise M_r is not an induced matching as defined), our algorithm indeed computes an induced matching of $G[X_i]$ included in Y of weight at least $w(M_r \cap X_i)$. Hence $w(N_r) \geqslant \frac{w(M_r)}{z}$.

Third, we already argued that an induced matching in G' corresponds to an induced matching in the black graph of $G/\mathcal{P}$. Thus at least one of the R''_i (with $i \in [2d+1]$) contains at least a $\frac{1}{2d+1}$ fraction of the weight of M_b. Therefore $w(N_b) \geqslant \frac{w(M_b)}{2d+1}$.

Finally the output induced matching has at least weight

$$\frac{w(M)}{3 \cdot \max(d+1, z, 2d+1)} = \frac{w(M)}{3z} = \frac{w(M)}{3(2(d-1)d+1)}.$$ ◀

▶ **Theorem 33.** *Max Subset Induced Matching on an n-vertex graph G, edge-weighted by w, with prescribed set $Y \subseteq E(G)$, and given with a d'-sequence, satisfies the assumptions of Lemma 22. In particular, this problem admits*

- *a $(d+1)^{2^q-1}$-approximation in time $2^{O_{d,q}(n^{2^{-q}})}$, for every integer $q \geqslant 0$,*
- *an n^ε-approximation in polynomial-time $O_{d,\varepsilon}(1) \log^{O_d(1)} n \cdot n^{O(1)}$, for any $\varepsilon > 0$, and*
- *a $\log n$-approximation in time $2^{O_d(n^{\frac{1}{\log \log n}})}$,*

with $d := c_{2d'+2} \cdot 2^{4c_{2d'+2}+4}$.

Proof. The exhaustive algorithm (trying out all vertex subsets and checking whether they induce a matching included in Y) solves Max Subset Induced Matching in time $2^{O(n)}$. Thus we show Max Subset Induced Matching satisfies the second item of Lemma 22, as witnessed by Lemma 34 where subcalls are dealt with recursively. We set $c_2 \geqslant 1$ as the required exponent to turn a d'-sequence into a neatly divided matrix of $\mathcal{M}_{n,2d'+2}$, and compute the various needed colorings, the appropriate $\frac{1}{2} < c_1 < 1$, and $2 < c_3 < 3$, and $f(d) = 2d \geqslant 1$ with $s := 2^{4c_{d'}+4}$.

In computing N_v, the algorithm makes $\lfloor \sqrt{n} \rfloor$ recursive calls and $d+1$ calls to Weighted Max Independent Set on induced subgraphs of G. All of these induced subgraphs are on less than $f(d)\sqrt{n}$ vertices. Computing N_r makes at most $\frac{\sqrt{n}d}{2}$ recursive calls on induced subgraphs of G with at most $f(d)\sqrt{n}$ vertices, followed by at most $2(d-1)d+1$ recursive calls on full cleanups of induced subtrigraphs of $G/\mathcal{P}$ with at most $\sqrt{n}$ vertices (in fact, one can observe that the latter recursive calls happen to also be on induced subgraphs of G). Finally, computing N_b makes one recursive call to a full cleanup of $G/\mathcal{P}$ on $\lfloor\sqrt{n}\rfloor$ vertices.

In summary, we make $O_d(\sqrt{n})$ recursive calls or calls to another problem WMIS (which already satisfies Lemma 22 with better constants) on induced subgraphs of G or full cleanups of (the whole) $G/\mathcal{P}$, each on $O_d(\sqrt{n})$ vertices. Hence, by Lemma 18, the induction applies.

We check that getting r-approximations on every subcall allows to output a global $3(2(d-1)d+1)r^2$-approximation. For that we argue that N_v (resp., N_r, N_b) is a $(2(d-1)d+1)r^2$-*approximation* of M_v (resp., M_r, M_b). The fact that N_v is a $(d+1)r^2$-approximation (hence a $(2(d-1)d+1)r^2$-approximation, since we assume that $d \geqslant 1$) of M_v directly follows Theorem 28.

We now show that N_r is a $(2(d-1)d+1)r^2$-approximation of M_r. Let $h \in [z] = [2(d-1)d+1]$ be an index maximizing $w(M_r \cap E(G[X_h]))$. Thus $w(M_r \cap E(G[X_h])) \geqslant \frac{w(M_r)}{2(d-1)d+1}$. Let $F_h \subseteq E_h$ be the edges $e = P_iP_j$ of $\mathcal{R}(G/\mathcal{P})$ that are inhabited by M_r (i.e., M_r contains at least one edge between P_i and P_j). Note that our algorithm makes an r-approximation of the optimum such solutions on $p(e)$ (selecting only edges between P_i and P_j). Thus the r-approximation on H'_h yields the desired $(2(d-1)d+1)r^2$-approximation N_r.

Finally, one can easily see that N_b is a $(2d+1)r$-approximation of M_b (note, here, the absence of a 2 in the exponent of r). ◀

6 Technical generalizations

6.1 Mutually Induced H-packing

In this section we present a far-reaching generalization of the approximation algorithms for Max Independent Set and Max Induced Matching. For any fixed graph H, let Mutually Induced H-packing be the problem where one seeks a largest collection of mutually induced copies of H in the input graph G, that is, a largest set S such that $G[S]$ is a disjoint union of (copies of) graphs H. We get similar approximation guarantees for Mutually Induced H-packing, for any connected graph H. Observe that Max Independent Set and Max Induced Matching are the special cases when H is a single vertex and a single edge, respectively.

We in fact approximate a technical generalization that we call Annotated Mutually Induced H-packing. The input is a tuple $(G, w, z, \gamma, \gamma')$ where G is a graph, $w : V(G)^{|V(H)|} \to \mathbb{Q}$ is a weight function over the tuples *without repetition* of $V(G)$ of size $|V(H)|$ (that we will use to keep track of the number of mutually induced copies *within* a given tuple of vertices of G), z is an integer between 1 and $|V(H)|$, $\gamma : V(G) \to [z]$ is a labeled partition of $V(G)$ into z classes, and $\gamma' : V(H) \to [z]$ is a labeled partition of $V(H)$ into z classes. Note that the Mutually Induced H-packing is obtained when $w(Z) = [G[Z]$ is isomorphic to $H]$ (where [.] is the Iverson bracket, i.e., taking value 1 if the property it surrounds is true, and 0 otherwise) and $z = 1$ (which forces the value of γ and γ'). The goal is to find a subset S such that

- $G[S]$ is a disjoint union of copies of H,
- there is an isomorphism between each copy C of H (in S) and H which preserves γ, γ', i.e., every vertex v of C is mapped to a vertex $v' \in V(H)$ with $\gamma(v) = \gamma'(v')$, and
- $\sum_{C \text{ copy of } H \text{ in } S} w(V(C))$ is maximized.

We will need the notion of *compatible trigraphs* of a (labeled) graph. Given a graph H, we call *compatible trigraph of* H any trigraph on at most $|V(H)|$ vertices obtained by turning some (possibly none) black edges or non-edges of trigraph $H/\mathcal{Q}$ (for any fixed choice of a partition $\mathcal{Q}$ of $V(H)$) into red edges. In other words, a compatible trigraph H' of H is such that there is a cleanup H'' of H' that is also a quotient trigraph of H. Note that the number of compatible trigraphs of an h-vertex graph H is upperbounded by $B_h \cdot 2^{\binom{h}{2}} = 2^{O(h^2)}$, where B_h is the h-th Bell number, which counts the number of partitions of a set of size h.

Given a graph G vertex-partitioned by $\mathcal{P}$ and a trigraph H, a subset $S \subseteq V(G)$ is said *cut by* $\mathcal{P}$ *along* H if $G[S]/\mathcal{P}$ is isomorphic to H. By extension, the copy of $G[S]$ in G (induced by S) is also said cut by $\mathcal{P}$ along H.

▶ **Lemma 34.** *For any connected graph H, ANNOTATED MUTUALLY INDUCED H-PACKING, when every input $(G, w, z, \gamma, \gamma')$ is given with a d'-sequence of the n-vertex graph G, satisfies the assumptions of Lemma 22. In particular, this problem admits*

- *a $d^{O_h(2^q)}$-approximation in time $2^{O_{d,h,q}(n^{2^{-q}})}$, for every integer $q \geqslant 0$,*
- *an n^ε-approximation in polynomial-time $O_\varepsilon(1) \cdot n^{O_{d,h}(1)}$, for any $\varepsilon > 0$,*

with $h = |V(H)|$, and $d := c_{2d'+2} \cdot 2^{4c_{2d'+2}+4}$.

Proof. As the first item of Lemma 22 is satisfied, we describe an algorithm that fulfills the requirement of its second item. We proceed by induction on the number of vertices of H. Thus we can assume that ANNOTATED MUTUALLY INDUCED J-PACKING, with J a connected graph on less vertices than H, satisfies Lemma 22. We already did the base case of the induction, which was WEIGHTED MAX INDEPENDENT SET.

Algorithm. Again, by Lemma 18, we start by computing in polynomial time a partition of $V(G)$, $\mathcal{P} = \{P_1, \ldots, P_{\lfloor\sqrt{n}\rfloor}\}$, of parts with size at most $d\sqrt{n}$ and such that $\mathcal{R}(G/\mathcal{P})$ has maximum degree at most d. Let S be a fixed (unknown) heaviest (with respect to w) mutually induced H-Packing of G preserving γ, γ'.

For every compatible trigraph H' of H, we look for mutually induced copies of H in G cut by $\mathcal{P}$ along H', and preserving γ, γ'. As the number of compatible trigraphs of H is $2^{O(h^2)}$, a $1/2^{O(h^2)}$ fraction of the weight of S is made of mutually induced copies of H which are cut by $\mathcal{P}$ along a fixed compatible trigraph H'. We now focus on this particular "run."

We distinguish two cases:

- (A) H' has at least one black edge, or
- (B) H' has no black edge.

As H is connected, the total graph of H' is also connected. Indeed, switching some edges or non-edge to red edges in the quotient trigraph of H cannot disconnect the total graph, which can only gain edges. Thus in case (A), every red component of H' has at least one incident black edge, and in case (B), H' has a single red component (and no black edge).

In general, we want to individually pack red components of H' (first type of recursive calls in smaller induced subgraphs of G), then combine those red components by connecting them with the right pattern of black edges (second type of recursive calls in the total graph of $G/\mathcal{P}$). Handling both cases (A) and (B) in an unified way runs into the technical issue that the weight function may destroy our combined solutions in an uncontrollable manner. The case distinction works as a win-win argument. In case (A), due to the presence of a black edge in H', we can pack at most one mutually induced copy of H within any fixed subtrigraph of $G/\mathcal{P}$ matching H'. We thus exempt ourselves from the first type of recursive calls. In case (B), we do need the two types of recursive calls (as in WMIS), but the first type is done on the whole H. Thus the current weight function (on h-tuples) is informative enough.

Case (A). The essential element here is to build a new weight function w' on the h'-tuples of the total graph $\mathcal{T}(G/\mathcal{P})$, with $h' := |V(H')|$. For every injective map $\iota : V(H') \to \mathcal{P}$ inducing a trigraph isomorphism and preserving γ, γ', for every ordering of $\iota(V(H'))$ into an h'-tuple $(P_1, \ldots, P_{h'})$, we set

$$w'(P_1, \ldots, P_{h'}) := \max\{w(v_1^1, v_1^2, \ldots, v_1^{a_1}, \ldots, v_{h'}^1, v_{h'}^2, \ldots, v_{h'}^{a_{h'}}) \ : \ v_1^1, v_1^2, \ldots, v_1^{a_1} \in P_1, \ldots$$

$$v_{h'}^1, v_{h'}^2, \ldots, v_{h'}^{a_{h'}} \in P_{h'}, \text{ and } G[\{v_1^1, v_1^2, \ldots, v_1^{a_1}, \ldots, v_{h'}^1, v_{h'}^2, \ldots, v_{h'}^{a_{h'}}\}] \text{ is isomorphic to } H\}.$$

Indeed as we previously observed, in case (A), at most one mutually induced copy of H respecting the cut along H' can be packed in the subgraph of G induced by the vertices of $\iota(V(H'))$. (In the definition of w', we can further impose that a_i matches the number of vertices of H in the corresponding part of H' but this is not necessary.)

All the h'-tuples not getting an image by w' in the previous loop (realized in time $n^{O(h)}$) are assigned the value 0. We then make a recursive call to Annotated Mutually Induced $\mathcal{T}(H')$-packing on input $(\mathcal{T}(G/\mathcal{P}), w', 1, \gamma_0, \gamma'_0)$ where we recall that $\mathcal{T}(.)$ is the total graph, and γ_0, γ'_0 are the constant 1 functions.

Case (B). For every injective map $\iota : V(H') \to \mathcal{P}$ inducing a trigraph isomorphism and preserving γ, γ', we make a recursive call to Annotated Mutually Induced H-packing with input $(G_\iota = G[\bigcup_{P \in \iota(V(H'))} P], w, h, \gamma_\iota, \gamma'_\iota)$ where two vertices get the same label by γ_ι if and only if they have the same label by γ and lie in the same $P \in \iota(V(H'))$, and γ'_ι gives to a vertex $v' \in X \in V(H')$ of H the same label given to the vertices $v \in \iota(X)$ such that $\gamma'(v') = \gamma(v)$. Informally $\gamma_\iota, \gamma'_\iota$ forces the recursive call to commit to the map ι and the former functions γ, γ'.

Each such recursive call yields a mutually induced packing of H. Since the red graph of $G/\mathcal{P}$ has degree at most d, we can color the (ordered) tuples of $\mathcal{P}$ of length up to h and inducing a connected subgraph of $\mathcal{R}(G/\mathcal{P})$ with at most $p(h, d) = hd^{2h} \cdot d^{2h} \cdot h! + 1$ colors such that every color class consists of disjoint tuples pairwise not linked by a red edge in $G/\mathcal{P}$. Indeed the claimed number of colors minus 1 upperbounds, in $\mathcal{R}(G/\mathcal{P})$, the number of connected tuples of length up to h that can touch (i.e., intersect or be adjacent to) a fixed connected tuple of length up to h. One color class contains a fraction $1/p(h, d)$ of the weight of the optimal solution S (subject to the same constraints). Running through all color classes j (and focusing on one containing a largest fraction of the optimum), we define a weight function w' on the h'-tuples of $\mathcal{T}(G\mathcal{P})$, with $h' = |V(H')|$, by giving to a tuple the weight returned by the corresponding recursive call whenever it is part of color class j, and weight 0 otherwise. We then make a recursive call to Annotated Mutually Induced $\mathcal{T}(H')$-packing on input $(\mathcal{T}(G/\mathcal{P}), w', 1, \gamma_0, \gamma'_0)$ where we recall that $\mathcal{T}(.)$ is the total graph, and γ_0, γ'_0 are the constant 1 functions.

We output a heaviest solution among all runs. We now check that the algorithm is as prescribed by Lemma 22.

Number of recursive calls. We make at most $2^{O(h^2)} \cdot h \cdot |V(G/\mathcal{P})|^h = n^{O_h(1)}$ recursive calls *to Annotated Mutually Induced H-packing*, and at most $p(h, d) + 1 = O_{d,h}(1)$ recursive calls *to Annotated Mutually Induced $\mathcal{T}(H')$-packing*. Hence there is a constant c_1 (function of d and h) such that the number of calls is bounded by n^{c_1}.

Nature and size of the inputs of the recursive calls. Both H and $\mathcal{T}(H')$ have strictly less vertices than H or are equal to H. Thus the induction on h applies. Besides, $G[\bigcup_{P \in \iota(V'(H))} P]$ is an induced subgraph of G of size at most $h \cdot d\sqrt{n} = O_{d,h}(1) \cdot \sqrt{n}$, and $\mathcal{T}(G/\mathcal{P})$ is a full cleanup of $G/\mathcal{P}$ of size at most $\lfloor \sqrt{n} \rfloor$.

Running time. Outside of the recursive calls, one can observe that our algorithm takes times $O_{d,h}(1) \cdot n^{O_h(1)}$. Hence there is a constant c_2 (function of d and h) such that the running time of that part is bounded by n^{c_2}.

Correctness and approximation guarantee. As all the recursive calls are on induced subgraphs of G or of the total graph $\mathcal{T}(G/\mathcal{P})$, we return a mutually induced collection of graphs of the size of H. All these graphs are indeed induced copies of H since the weight function prevents the false positives of copies of H in the total graph $\mathcal{T}(G/\mathcal{P})$ but not in G (these tuples are given weight 0). Finally it can be checked that the returned solution has weight a fraction $(2^{O(h^2)} \cdot \max(r, p(h,d)r^2))^{-1}$ of the optimum, which can also be seen as a $d^{c_3}r^2$-approximation for some constant c_3 depending on d and h. ◀

6.2 Independent induced packing of stars and forests

The techniques employed to design approximations algorithms for Max Subset Induced Matching can be extended in order to tackle more general problems. In particular, we show in this section a generalization of Theorem 33 for Max Edge Induced Star Forest and Max Edge Induced Forest. These two problems stand as the version of Mutually Induced $\mathcal{H}$-packing where $\mathcal{H}$ is respectively either the infinite family of stars or trees.

On the one hand, Max Edge Induced Star Forest asks, given a graph G and a subset $Y \subseteq E(G)$, for a collection of induced stars on G, made up of edges of Y only, maximizing the number of edges (or leaves).

Max Edge Induced Star Forest
Input: Graph G, subset $Y \subseteq E(G)$
Output: Collection $(A_i)_{i\in[k]}$ of induced stars on G, made up of edges in Y only, such that there is no edge between A_i and A_j, for any $i \neq j \in [k]$, which maximizes the number of edges.

On the other hand, given the same input, Max Edge Induced Forest asks for an induced forest F on G with the largest set of edges.

We would like to emphasize the fact that the objective function of both problems counts the number of *edges* in the solution, instead of *vertices*, as it is often the case in the literature when looking for a collection of stars or trees in a graph. The reason for this is because an approximated solution for these vertex versions can be obtained from an approximated solution of Weighted Max Independent Set (since any independent set is a star forest, and any forest is a bipartite graph).

Observe moreover that a solution of Max Edge Induced Forest can be 3-approximated with a solution of Max Edge Induced Star Forest. Indeed, the edge set of any tree can be partitioned into three distance-2-edge colors, which consist of a collection of stars. Therefore, the induced forest F can be partitioned into three collections of induced stars. In the remainder, we design approximation algorithms for Max Edge Induced Star Forest, and directly deduce results for Max Edge Induced Forest.

In the remainder, we propose approximation algorithms for Max Edge Induced Star Forest. We provide in particular a n^{ε}-approximation algorithm for Max Edge Induced Star Forest, running in polynomial time.

We need to find the suitable generalization of Max Edge Induced Star Forest, as it was done for Coloring in Section 4. We call this problem Max Leaves Induced Star Forest. Now, a weight function on vertices is added to the input, and we seek a collection of mutually induced stars with maximum weight, the weight of a star being the sum of the weights of its leaves (that is, the weight of the root is omitted).

Max Leaves Induced Star Forest
Input: Graph G, weights $w_V : V \to \mathbb{N}$, subset $Y \subseteq E(G)$
Output: Collection $(A_i)_{i \in [k]}$ of induced stars on G with root r_i, $A_i = \{r_i, s_i^1, \dots, s_i^{L_i}\}$, made up of edges in Y only, with no edge between A_i and A_j, for any $i \neq j \in [k]$, maximizing

$$\sum_{i=1}^{k} w_V(A_i) = \sum_{i=1}^{k} \sum_{\ell=1}^{L_i} w(s_i^\ell)$$

We prove that Max Leaves Induced Star Forest follows the framework proposed in Lemma 22. We begin with the design of a subexponential-time algorithm approximating a solution of Max Leaves Induced Star Forest with a ratio function of twin-width.

▶ **Lemma 35.** *Assume every input of Max Leaves Induced Star Forest is given with a d'-sequence of the n-vertex G, and $d := c_{2d'+2} \cdot 2^{4c_{2d'+2}+4}$. Max Leaves Induced Star Forest can be $O(d^2)$-approximated in time $2^{O_d(\sqrt{n})}$ on these inputs.*

Proof. We compute in polynomial time a partition of $V(G)$, $\mathcal{P} = \{P_1, \dots, P_{\lfloor \sqrt{n} \rfloor}\}$, of parts with size at most $d\sqrt{n}$ and such that $\mathcal{R}(G/\mathcal{P})$ has maximum degree at most d, by Lemma 17.

As in Lemma 19, we $(d+1)$-color $\mathcal{R}(G/\mathcal{P})$, which defines a coarsening $\{C_1, \dots, C_{d+1}\}$ of $\mathcal{P}$. Moreover, we distance-2-edge-color $\mathcal{R}(G/\mathcal{P})$ with $z = 2(d-1)d + 1$ colors. This partitions the edges of $\mathcal{R}(G/\mathcal{P})$ into $\{E_1, \dots, E_z\}$. For each red edge $e = P_iP_j \in R(G/\mathcal{P})$, we denote by $p(e)$ the set $P_i \cup P_j$.

Let $A = \bigcup_{i=1}^{k} A_i$ be the union of all stars present in an optimum solution of Max Leaves Induced Star Forest in G. We have $A \subseteq Y$. Let A_v, A_r, A_b partition A, where A_v contains the edges of A with both endpoints in a same P_i, A_r corresponds to edges of A between some P_i and P_j with $P_iP_j \in R(G/\mathcal{P})$, and A_b, the edges of A between some P_i and P_j with $P_iP_j \in E(G/\mathcal{P})$. The set of edges A_v (resp. A_r, A_b) still form a collection of mutually induced stars. At least one over the three solutions produced by the partition A_v, A_r, A_b gives us a 3-approximation for this problem. Our algorithm consists of computing three solutions for Max Leaves Induced Star Forest of G, capturing a positive fraction of A_v, A_r, A_b, respectively.

Computing a $d+1$-approx for A_v. *Construction.* For every integer $1 \leqslant i \leqslant \lceil \sqrt{n} \rceil$, we compute an optimum solution for Max Leaves Induced Star Forest in $G[P_i]$ contained in Y, say S_i, in time $2^{O_d(\sqrt{n})}$. This can be achieved with guesses of the vertices in P_i, as $|P_i| \leqslant d\sqrt{n}$.

Then, we focus on each color C_j of $\mathcal{R}(G/\mathcal{P})$, for $j \in [d+1]$. There is no red edge in $H_j = (G/\mathcal{P})\,[C_j]$. We compute a heaviest independent set I_j in H_j where the parts P_i are weighted by the edge weight of S_i. Let R_j be the union of all optimum solutions for Max Leaves Induced Star Forest on all P_i belonging to I_j. The solution returned is the maximum over all R_js.

Approximation ratio. Let A_v^j be the subset of A_v made up of edges belonging to parts of C_j. There is no red edge between two parts of C_j, therefore their neighborhood consists of either full adjacency or full non-adjacency. As a consequence, a maximum-weighted collection of stars in C_j with edges inside parts intersects parts which are pairwise non-adjacent in $(G/\mathcal{P})[C_j]$, otherwise the stars are not mutually induced. Consequently, this justifies that the set R_j returned for each C_j is a maximum-weighted collection of stars in C_j made up of edges inside parts. In summary, the weight of each collection R_j is greater than the weight of A_v^j. As $j \in [d+1]$, a heaviest collection among all R_js is a $d+1$-approximation of A_v.

Computing a $O(d^2)$-approx for A_r. *Construction.* For each $e = P_iP_j \in R(G/\mathcal{P})$, we compute an optimal solution for Max Leaves Induced Star Forest in $G[p(e)] = G[P_i \cup P_j]$ among those that are included in Y and have only edges with one endpoint in P_i and the other endpoint in P_j. Said differently, we determine a maximum-weighted collection of induced stars in $G[p(e)]$ over Y with a root on one side (for example, P_i) and all leaves on the other side (P_j). This costs at most $2^{O_d(\sqrt{n})}$ by trying out all vertex subsets, since $|P_i \cup P_i| \leqslant 2d\sqrt{n}$. The set of vertices of the solution returned on $G[p(e)]$ is denoted by $B_e \subseteq p(e)$.

For each $h \in [z]$, let H'_h be the trigraph $(G/\mathcal{P})[\{P_i \ : \ P_i \text{ is incident to an edge } e \in E_h\}]$. The red edges of H'_h form an induced matching on the red graph of H'_h as they are at distance 2 in $G/\mathcal{P}$. We associate with any edge $e \in E_h$ the edge weight of B_e. Then, we turn the red edges of H'_h in black: let H''_h be the graph obtained. We solve Max Subset Induced Matching on H''_h by restricting it to edges of E_h (which plays the role of Y): this is achieved in $2^{O(\sqrt{n})}$ as $|V(H''_h)| \leqslant \sqrt{n}$. Let I''_h be a maximum-weighted induced matching obtained. For each $h \in [z]$, we obtain the union R_h of all B_e, $e \in I''_h$: $R_h = \bigcup_{e \in I''_h} B_e$. We return an R_h which maximizes the total edge weight, among all $h \in [z]$.

Approximation ratio. Let A^h_r be the subset of A_r made up of edges being part of red edges E_h in $G/\mathcal{P}$, for $h \in [z]$. As the edges of E_h form an induced matching in $\mathcal{R}(G/\mathcal{P})$, the union of solutions of Max Leaves Induced Star Forest over graphs $G[p(e)]$ with $e \in E_h$ can only be connected through black edges of $G/\mathcal{P}$. Furthermore, two collections of stars over $G[p(e)]$ and $G[p(f)]$ are necessarily not mutually induced if there is a black edge between an endpoint of e and an endpoint of f. Consequently, R_h gives a maximum-weighted collection of mutually induced stars over E_h and its weight is at least the weight of A^h_r. The maximum-weighted collection over all R_h gives a z-approximation, as $h \in [z]$.

Computing a $2d+1$-approx for A_b. *Construction.* For each part P_i, we solve Weighted Max Independent Set on $G[P_i]$ with weight function w_V. Let $I(P_i)$ be the independent set returned and $w(P_i)$ its weight. We focus now on graph $G' = (V(G/\mathcal{P}), E(G/\mathcal{P}))$, made up of the black edges of $G/\mathcal{P}$, and solve Max Leaves Induced Star Forest on it with weights $w(P_i)$. As $|V(G')| \leqslant \sqrt{n}$, this is achieved in $2^{O(\sqrt{n})}$.

Let $(B_h)_{h\in[k]}$ be the collection of stars returned, $B_h = \{R_h, S^1_h, \ldots, S^{L_h}_h\}$ and $B \in E(G')$ be the set of edges belonging to this collection. Based on the bounded maximum red degree of $G/\mathcal{P}$, we determine a $O(d)$-partition of the edges of B, in order to produce collections of mutually induced stars. Let H^* be the graph where each edge e in the collection $(B_h)_{h\in[k]}$ is represented with a vertex and two of them e, f are adjacent if and only if there is a red edge in $G/\mathcal{P}$ connecting an endpoint of e with an endpoint of f. This graph has degree at most $2d$, so it can be $2d+1$-colored: let $T_1, \ldots, T_{2d+1}$ be the corresponding color classes. Any set of edges T_j gives us a collection of mutually induced stars on trigraph $G/\mathcal{P}$, in the sense that there is neither a black nor a red edge between two stars.

We fix some color class: say T_1 w.l.o.g. Let (B^*_h) be the collection of stars produced by T_1, where $B^*_h = \{R^*_h, S^{1,*}_h, \ldots, S^{L^*_h,*}_h\}$. For the root $R^*_h = P_i$ of each star B^*_h, we select an arbitrary vertex $r_h \in P_i$. Let $(B^*_h*)_{h\in[k]}$ be the following collection of stars (which are mutually induced) on G: $B^{**}_h = \{r_h\} \cup \bigcup_{\ell=1}^{L^*_h} I(S^{\ell,*}_h)$. In brief, the collection $(B^{**}_h)_{h\in[k]}$ is made up of an arbitrary vertex of each root of stars B^*_h and a maximum-weighted independent set of each leaf of B^*_h. Remember that we computed this collection of stars for T_1: we return a maximum-weighted collection $(B^{**}_h)_{h\in[k]}$ among all the ones determined for T_j, $j \in [2d+1]$.

Approximation ratio. Any collection B_b with stars belonging only to black edges of $G/\mathcal{P}$ reveals a collection of stars on the quotient graph. Concretely, two black edges of $G/\mathcal{P}$ containing each a branch of B_b must be either non-adjacent or form an induced 3-vertex path on $G' = (V(G/\mathcal{P}), E(G/\mathcal{P}))$. Conversely, considering a collection B^* of mutually induced stars of G' and, for each $e \in B^*$, a collection B^*_e of mutually induced stars on $G[p(e)]$ produces a global collection of stars of G: then, we can partition its edges into $2d+1$ parts (as with $T_1, \ldots, T_{2d+1}$) such that each part contains mutually induced stars. As the collection B computed above provides us with a heaviest collection of G', a maximum-weighted B^{**}_h over all T_j is a $2d+1$-approximation for B, whose weight is at least the weight of A_b.

Conclusion of the proof. We finally output a heaviest collection of mutually induced stars among the three approximating respectively A_v, A_r, and A_b. The overall running time is in $2^{O_d(\sqrt{n})}$. An upper bound for the approximation ratio of this algorithm is $3z = O(d^2)$. ◀

As for the other problems treated in this article, we apply to Max Leaves Induced Star Forest the time-approximation trade-off proposed in Lemma 22.

▶ **Theorem 36.** *Max Leaves Induced Star Forest on an n-vertex graph G, weight function w_V, with prescribed set $Y \subseteq E(G)$, and given with a d'-sequence, satisfies the assumptions of Lemma 22. In particular, this problem admits*

- *a $(d+1)^{2^q-1}$-approximation in time $2^{O_{d,q}(n^{2^{-q}})}$, for every integer $q \geqslant 0$,*
- *an n^{ε}-approximation in polynomial-time $O_{d,\varepsilon}(1) \log^{O_d(1)} n \cdot n^{O(1)}$, for any $\varepsilon > 0$, and*
- *a $\log n$-approximation in time $2^{O_d(n^{\frac{1}{\log \log n}})}$,*

with $d := c_{2d'+2} \cdot 2^{4c_{2d'+2}+4}$.

Proof. The exhaustive algorithm (trying out all vertex subsets and checking whether they induce a collection of mutually induced stars in Y) solves Max Leaves Induced Star Forest in time $2^{O(n)}$. Thus we show Max Leaves Induced Star Forest satisfies the second item of Lemma 22. We set $c_2 \geqslant 1$ as the required exponent to turn a d'-sequence into a neatly divided matrix of $\mathcal{M}_{n,2d'+2}$ conform to G, and compute the various needed colorings, the appropriate $\frac{1}{2} < c_1 < 1$, and $2 < c_3 < 3$, and $f(d) = 2d \geqslant 1$.

Approximating A_v. The algorithm makes $\lfloor \sqrt{n} \rfloor$ recursive calls to solve Max Leaves Induced Star Forest on parts P_i. Furthermore, $d+1$ calls to WMIS are needed on induced subgraphs of $G/\mathcal{P}$. All of these induced subgraphs are on at most $d\sqrt{n}$ vertices.

Approximating A_r. The algorithm makes at most $\frac{\sqrt{n}d}{2}$ recursive calls (one call per red edge of $G/\mathcal{P}$) on induced subgraphs of G with at most $2d\sqrt{n}$ vertices, followed by at most $2(d-1)d+1$ calls of Max Subset Induced Matching on full cleanups of induced subtrigraphs of $G/\mathcal{P}$ with at most $\sqrt{n}$ vertices.

Approximating A_b. The algorithm makes $\lfloor \sqrt{n} \rfloor$ calls to solve WMIS on parts P_i and one recursive call on a full cleanup of $G/\mathcal{P}$ on $\lfloor \sqrt{n} \rfloor$ vertices.

In summary, we make $O_d(\sqrt{n})$ recursive calls or calls to problems WMIS and Max Subset Induced Matching (which already satisfy Lemma 22 with better constants) on induced subgraphs of G or full cleanups of (the whole) $G/\mathcal{P}$, each on $O_d(\sqrt{n})$ vertices. Hence, by Lemma 18, the induction applies.

Getting r-approximations on every subcall allows us to output a global $3(2(d-1)d+1)r^2$-approximation for Max Leaves Induced Star Forest:

- collection A_v is approximated with ratio $(d+1)r^2$
- collection A_r is approximated with ratio $(2(d-1)d+1)r^2$
- collection A_b is approximated with ratio $(2d+1)r^2$.

The extra factor 3 comes from the fact that we output the heaviest of these three solutions. ◀

Max Edge Induced Star Forest is a particular case of Max Leaves Induced Star Forest with $w_V(u) = 1$ for every vertex $u \in V(G)$. Furthermore, a solution of Max Edge Induced Star Forest is a 3-approximation of a solution of Max Edge Induced Forest. These observations together with Theorem 36 allow us to state the following result.

▶ **Corollary 37.** *Max Edge Induced Star Forest and Max Edge Induced Forest on an n-vertex graph G, with prescribed set $Y \subseteq E(G)$, and given with a d'-sequence, admit*

- *an n^{ε}-approximation in polynomial-time $O_{d,\varepsilon}(1) \log^{O_d(1)} n \cdot n^{O(1)}$, for any $\varepsilon > 0$, and*
- *a $\log n$-approximation in time $2^{O_d(n^{\frac{1}{\log \log n}})}$,*

with $d := c_{2d'+2} \cdot 2^{4c_{2d'+2}+4}$.

7 Limits

We now discuss the limits of our framework. We give some examples of problems that are unlikely to have an n^{ε}-approximation algorithm on graphs of bounded twin-width. The first such problem is Min Independent Dominating Set, where one seeks a minimum-cardinality set which is both an independent set and a dominating set. In general n-vertex graphs, this problem cannot be $n^{1-\varepsilon}$-approximated in polynomial time unless P=NP [22], and cannot be r-approximated in time $2^{o(n/r)}$ for any $r = r(n)$, unless the ETH fails [11].

We show that Min Independent Dominating Set has the same polytime inapproximability in graphs of bounded twin-width.

▶ **Theorem 38.** *For every $\varepsilon > 0$, Min Independent Dominating Set cannot be $n^{1-\varepsilon}$-approximated in polynomial time on n-vertex graphs of twin-width at most 9 given with a 9-sequence, unless P=NP.*

Proof. We perform the classic reduction of Halldórsson from SAT [22], but from Planar 3-SAT where each literal has at most two occurrences, which remains NP-complete [29]. More precisely we add a triangle d_i, t_i, f_i for each variable x_i (with $i \in [N]$), and an independent set I_j of size r for each 3-clause C_j (with $j \in [M]$). We link t_i to all the vertices of I_j whenever x_i appears positively in C_j, and we link f_i to all the vertices of I_j whenever x_i appears negatively in C_j. This defines a graph G with $n = 3N + rM$ vertices.

It can be observed that if the Planar 3-SAT instance is satisfiable, then there is an independent dominating set of size N, whereas if the formula is unsatisfiable then any independent dominating set has size at least r. Setting $r := N^{\frac{2-\varepsilon}{\varepsilon}}$, the gap between positive and negative instances is $\Theta_{\varepsilon}(1)n^{1-\varepsilon}$, while preserving the fact that the reduction is polynomial.

Let us now argue that G has twin-width at most 9, and that a 9-sequence of it can be computed in polynomial time. We can first contract each I_j into a single vertex without creating a red edge. Next we can contract every triangle d_i, t_i, f_i into a single vertex of red degree at most 4. At this point, the current trigraph is a planar graph of maximum degree at most 4. It was observed in [9] that planar trigraphs with maximum (total) degree at most 9 have twin-width at most 9. This is because any planar graph has a pair of vertices on the same face with at most 9 neighbors (outside of themselves) combined [28]. Hence we get a 9-sequence for G that can be computed in polynomial time. Incidentally the twin-width of planar graphs (that is, planar trigraphs without red edge) but no restriction on the maximum degree is also at most 9 [24]. ◀

Another very inapproximable is Longest Induced Path, which also does not admit a polytime $n^{1-\varepsilon}$-approximation algorithm unless P=NP [31], and cannot be r-approximated in time $2^{o(n/r)}$ for any $r = o(n)$, unless the ETH fails [11]. The non-induced version, the Longest Path problem, has a notoriously big gap between the best known approximation algorithm whose factor is $n/\exp(\Omega(\sqrt{\log n}))$ [18], and the sharpest conditional lower bound which states that, for any $\varepsilon > 0$, a $2^{\log^{1-\varepsilon} n}$-approximation would imply that NP $\subseteq$ QP [27].

Despite being an open question for decades the existence or conditional impossibility of an approximation algorithm for Longest Path with approximation factor, say, $\sqrt{n}$ has not been settled. Nor do we know whether an n^ε-approximation for any $\varepsilon > 0$ is possible. We now show that using our framework to obtain an n^ε-approximation for Longest Induced Path of Longest Path in graphs of bounded twin-width is unlikely to work, in the sense that it would immediately yield such an approximation factor for Longest Path in general graphs.

▶ **Theorem 39.** *For any $r = \omega(1)$, an r-approximation for Longest Induced Path or Longest Path on graphs of twin-width at most 4 given with a 4-sequence would imply a $(1 + o(1))r$-approximation for Longest Path in general graphs.*

Proof. It was shown in [3] that any graph obtained by subdividing every edge of an n-vertex graph at least $2 \log n$ has twin-width at most 4. Besides, a 4-sequence can then be computed in polynomial time.

Let G be any graph with minimum degree at least 2 (note that this restriction does not make Longest Path easier to approximate), and G' be obtained from G by subdividing each of its edges $2\lceil \log n \rceil$ times, and let $s := 2\lceil \log n \rceil + 1$. Let us observe that G has a path of length ℓ if and only if G' has a path of length $(\ell + 2)s - 2$ if and only if G' has an induced path of length $(\ell + 2)s - 4$. Hence a polytime r-approximation for Longest Induced Path or Longest Path in graphs of bounded twin-width given a 4-sequence would translate into a $(1 + o(1))r$-approximation for Longest Path in general graphs. ◀

We can use Theorem 39 to get a similar weak obstruction to an n^ε-approximation for Mutually Induced $\mathcal{H}$-packing in graphs of bounded twin-width, for some infinite family of connected graphs $\mathcal{H}$. Recall that by Lemma 34 such an approximation algorithm does exist when $\mathcal{H}$ is a *finite* collection of connected graphs.

Setting $\mathcal{H}$ to be the set of all paths does not serve that purpose, since one can then use the approximation algorithm for Max Induced Matching. Nevertheless this almost works. We just need to *decorate* the endpoints of the paths. For every positive integer n, let D_n be the *decorated path* of length n, obtained from the n-vertex path P_n by adding for each endpoint u two adjacent vertices u', u'' both adjacent to u. Informally, D_n is a path terminated by a triangle at each end.

▶ **Theorem 40.** *Let $\mathcal{H} := \{D_n \ : \ n \in \mathbb{N}^+\}$ be the family of all decorated paths. If for every $\varepsilon > 0$, Mutually Induced $\mathcal{H}$-packing admits an n^ε on n-vertex graphs of bounded twin-width given with a 4-sequence, then so does Longest Path on general graphs.*

Proof. Let G be any graph. For every pair $u \neq v \in V(G)$, define G_{uv} as the graph obtained from G by subdividing all its edges $2\lceil \log(n + 2) \rceil$ times, and adding two adjacent vertices u', u'' both adjacent to u, and two adjacent vertices v', v'' both adjacent to v. Since there are only two triangles in G_{uv}, only one graph of $\mathcal{H}$ can be present in a (mutually induced) packing. Thus Mutually Induced $\mathcal{H}$-packing is now equivalent to finding a longest path between u and v. An n^ε-approximation algorithm for this problem would, by Theorem 39, give a similar approximation algorithm for Longest Path in general graphs.

Despite u', u'', v', v'', G_{uv} still admits a 4-sequence. For instance, first contract u' and u'', and contract v' and v''; this does not create red edges, and has the same effect as deleting u'' and v''. The obtained graph is an induced subgraph of a $2\lceil \log(n + 2) \rceil$-subdivision (of a graph on at most $n + 2$ vertices). Hence it admits a polytime computable 4-sequence [3]. ◀

References

1 Noga Alon, Guoli Ding, Bogdan Oporowski, and Dirk Vertigan. Partitioning into graphs with only small components. *J. Comb. Theory, Ser. B*, 87(2):231–243, 2003. `doi:10.1016/S0095-8956(02)00006-0`.

2 Nikhil Bansal, Parinya Chalermsook, Bundit Laekhanukit, Danupon Nanongkai, and Jesper Nederlof. New tools and connections for exponential-time approximation. *Algorithmica*, 81(10):3993–4009, 2019. `doi:10.1007/s00453-018-0512-8`.

3 Pierre Bergé, Édouard Bonnet, and Hugues Déprés. Deciding twin-width at most 4 is NP-complete. In Mikolaj Bojanczyk, Emanuela Merelli, and David P. Woodruff, editors, *49th International Colloquium on Automata, Languages, and Programming, ICALP 2022, July 4-8, 2022, Paris, France*, volume 229 of *LIPIcs*, pages 18:1–18:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. `doi:10.4230/LIPIcs.ICALP.2022.18`.

4 Édouard Bonnet, Dibyayan Chakraborty, Eun Jung Kim, Noleen Köhler, Raul Lopes, and Stéphan Thomassé. Twin-width VIII: delineation and win-wins. *CoRR*, abs/2204.00722, 2022. `arXiv:2204.00722`, `doi:10.48550/arXiv.2204.00722`.

5 Édouard Bonnet, Colin Geniet, Eun Jung Kim, Stéphan Thomassé, and Rémi Watrigant. Twin-width II: small classes. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1977–1996, 2021. `doi:10.1137/1.9781611976465.118`.

6 Édouard Bonnet, Colin Geniet, Eun Jung Kim, Stéphan Thomassé, and Rémi Watrigant. Twin-width II: small classes. *CoRR*, abs/2006.09877, 2020. URL: `http://arxiv.org/abs/2006.09877`, `arXiv:2006.09877`.
In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1977–1996, 2021. `doi:10.1137/1.9781611976465.118`.

7 Édouard Bonnet, Colin Geniet, Eun Jung Kim, Stéphan Thomassé, and Rémi Watrigant. Twin-width III: max independent set, min dominating set, and coloring. In Nikhil Bansal, Emanuela Merelli, and James Worrell, editors, *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021, July 12-16, 2021, Glasgow, Scotland (Virtual Conference)*, volume 198 of *LIPIcs*, pages 35:1–35:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. `doi:10.4230/LIPIcs.ICALP.2021.35`.

8 Édouard Bonnet, Ugo Giocanti, Patrice Ossona de Mendez, Pierre Simon, Stéphan Thomassé, and Szymon Torunczyk. Twin-width IV: ordered graphs and matrices. In Stefano Leonardi and Anupam Gupta, editors, *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20 - 24, 2022*, pages 924–937. ACM, 2022. `doi:10.1145/3519935.3520037`.

9 Édouard Bonnet, Eun Jung Kim, Amadeus Reinald, and Stéphan Thomassé. Twin-width VI: the lens of contraction sequences. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1036–1056. SIAM, 2022.

10 Édouard Bonnet, Eun Jung Kim, Stéphan Thomassé, and Rémi Watrigant. Twin-width I: tractable FO model checking. *J. ACM*, 69(1):3:1–3:46, 2022. `doi:10.1145/3486655`.

11 Édouard Bonnet, Michael Lampis, and Vangelis Th. Paschos. Time-approximation trade-offs for inapproximable problems. *J. Comput. Syst. Sci.*, 92:171–180, 2018. `doi:10.1016/j.jcss.2017.09.009`.

12 Nicolas Bourgeois, Bruno Escoffier, and Vangelis Th. Paschos. Approximation of max independent set, min vertex cover and related problems by moderately exponential algorithms. *Discret. Appl. Math.*, 159(17):1954–1970, 2011. `doi:10.1016/j.dam.2011.07.009`.

13 Parinya Chalermsook, Bundit Laekhanukit, and Danupon Nanongkai. Graph products revisited: Tight approximation hardness of induced matching, poset dimension and more. In Sanjeev Khanna, editor, *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, Louisiana, USA, January 6-8, 2013*, pages 1557–1576. SIAM, 2013. `doi:10.1137/1.9781611973105.112`.

14 Parinya Chalermsook, Bundit Laekhanukit, and Danupon Nanongkai. Independent set, induced matching, and pricing: Connections and tight (subexponential time) approximation hardnesses.

In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 370–379, 2013. doi:10.1109/FOCS.2013.47.

15 Marek Cygan, Lukasz Kowalik, Marcin Pilipczuk, and Mateusz Wykurz. Exponential-time approximation of hard problems. *CoRR*, abs/0810.4934, 2008. URL: http://arxiv.org/abs/0810.4934, arXiv:0810.4934.

16 Irit Dinur and David Steurer. Analytical approach to parallel repetition. In David B. Shmoys, editor, *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 624–633. ACM, 2014. doi:10.1145/2591796.2591884.

17 Uriel Feige, Shafi Goldwasser, László Lovász, Shmuel Safra, and Mario Szegedy. Approximating clique is almost NP-complete (preliminary version). In *32nd Annual Symposium on Foundations of Computer Science, San Juan, Puerto Rico, 1-4 October 1991*, pages 2–12. IEEE Computer Society, 1991. doi:10.1109/SFCS.1991.185341.

18 Harold N. Gabow and Shuxin Nie. Finding a long directed cycle. *ACM Trans. Algorithms*, 4(1):7:1–7:21, 2008. doi:10.1145/1328911.1328918.

19 Jakub Gajarský, Michal Pilipczuk, Wojciech Przybyszewski, and Szymon Torunczyk. Twin-width and types. In Mikolaj Bojanczyk, Emanuela Merelli, and David P. Woodruff, editors, *49th International Colloquium on Automata, Languages, and Programming, ICALP 2022, July 4-8, 2022, Paris, France*, volume 229 of *LIPIcs*, pages 123:1–123:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ICALP.2022.123.

20 Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman, 1979.

21 Sylvain Guillemot and Dániel Marx. Finding small patterns in permutations in linear time. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 82–101, 2014. doi:10.1137/1.9781611973402.7.

22 Magnús M. Halldórsson. Approximating the minimum maximal independence number. *Inf. Process. Lett.*, 46(4):169–172, 1993. doi:10.1016/0020-0190(93)90022-2.

23 Johan Håstad. Clique is hard to approximate within $n^{1-\epsilon}$. In *37th Annual Symposium on Foundations of Computer Science, FOCS '96, Burlington, Vermont, USA, 14-16 October, 1996*, pages 627–636, 1996. doi:10.1109/SFCS.1996.548522.

24 Petr Hliněný. Twin-width of planar graphs is at most 9, 2022. URL: https://arxiv.org/abs/2205.05378, doi:10.48550/ARXIV.2205.05378.

25 Russell Impagliazzo and Ramamohan Paturi. On the Complexity of k-SAT. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001. doi:10.1006/jcss.2000.1727.

26 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001. doi:10.1006/jcss.2001.1774.

27 David R. Karger, Rajeev Motwani, and G. D. S. Ramkumar. On approximating the longest path in a graph. *Algorithmica*, 18(1):82–98, 1997. doi:10.1007/BF02523689.

28 Anton Kotzig. Contribution to the theory of eulerian polyhedra. *Mat. Cas. SAV (Math. Slovaca)*, 5:111–113, 1955.

29 László Kozma. Minimum average distance triangulations. In Leah Epstein and Paolo Ferragina, editors, *Algorithms - ESA 2012 - 20th Annual European Symposium, Ljubljana, Slovenia, September 10-12, 2012. Proceedings*, volume 7501 of *Lecture Notes in Computer Science*, pages 695–706. Springer, 2012. doi:10.1007/978-3-642-33090-2_60.

30 Stefan Kratsch, Florian Nelles, and Alexandre Simon. On triangle counting parameterized by twin-width. *CoRR*, abs/2202.06708, 2022. URL: https://arxiv.org/abs/2202.06708, arXiv:2202.06708.

31 Carsten Lund and Mihalis Yannakakis. The approximation of maximum subgraph problems. In Andrzej Lingas, Rolf G. Karlsson, and Svante Carlsson, editors, *Automata, Languages and Programming, 20nd International Colloquium, ICALP93, Lund, Sweden, July 5-9, 1993,*

Proceedings, volume 700 of *Lecture Notes in Computer Science*, pages 40–51. Springer, 1993. doi:10.1007/3-540-56939-1_60.

32 Jesper Nederlof. Inclusion exclusion for hard problems, 2008.

33 Michal Pilipczuk, Marek Sokolowski, and Anna Zych-Pawlewicz. Compact representation for matrices of bounded twin-width. In Petra Berenbrink and Benjamin Monmege, editors, *39th International Symposium on Theoretical Aspects of Computer Science, STACS 2022, March 15-18, 2022, Marseille, France (Virtual Conference)*, volume 219 of *LIPIcs*, pages 52:1–52:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.STACS.2022.52.

34 David Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. *Theory of Computing*, 3(1):103–128, 2007.