A Faster Algorithm for Vertex Cover Parameterized by Solution Size

David G. Harris
Department of Computer Science, University of Maryland, College Park, MD, USA

N. S. Narayanaswamy
Department of Computer Science and Engineering, Indian Institute of Technology Madras, India

Abstract

We describe a new algorithm for vertex cover with runtime $O^*(1.25284^k)$, where k is the size of the desired solution and O^* hides polynomial factors in the input size. This improves over the previous runtime of $O^*(1.2738^k)$ due to Chen, Kanj, & Xia (2010) standing for more than a decade. The key to our algorithm is to use a measure which simultaneously tracks k as well as the optimal value λ of the vertex cover LP relaxation. This allows us to make use of prior algorithms for Maximum Independent Set in bounded-degree graphs and Above-Guarantee Vertex Cover.

The main step in the algorithm is to branch on high-degree vertices, while ensuring that both k and $\mu = k - \lambda$ are decreased at each step. There can be local obstructions in the graph that prevent μ from decreasing in this process; we develop a number of novel branching steps to handle these situations.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis

Keywords and phrases Vertex cover, FPT, Graph algorithm

Digital Object Identifier 10.4230/LIPIcs.STACS.2024.40

Related Version *Full Version*: `https://arxiv.org/abs/2205.08022`

1 Introduction

For an undirected graph $G = (V, E)$, a subset $S \subseteq V$ is called a *vertex cover* if every edge has at least one endpoint in S. It is closely related to an *independent set*, since if S is an inclusion-wise minimal vertex cover then $V - S$ is an inclusion-wise maximal independent set, and vice-versa. Finding the size of the smallest vertex cover is a classic NP-complete problem [6]. In particular, G has a vertex cover of size at most k if and only if it has an independent set of size at least $n - k$.

There is natural LP formulation for vertex cover which we denote by LPVC(G):

$$\begin{array}{rll} \text{minimize} & \sum_{v \in V} \theta(v) & \\ \text{subject to} & \theta(u) + \theta(v) \geq 1 & \text{for all edges } e = (u,v) \in E \\ & \theta(v) \in [0,1] & \text{for all vertices } v \in V \end{array}$$

The optimal solution to LPVC(G), denoted $\lambda(G)$ or just λ if G is clear from context, is a lower bound on the size of a minimum vertex cover of G. We also define $\mu(G) = k - \lambda(G)$, i.e. the gap between solution size and LP lower bound. This linear program has remarkable properties which have been exploited for a variety of algorithms [12, 13, 9]. For instance, an optimum basic solution is half-integral and can be found efficiently by a network flow computation.

Fixed-parameter tractable (FPT) and exact algorithms explore a landscape of parameters to understand the complexity of different problems [14, 4, 3, 2, 5]. Vertex Cover was one of the first studied FPT problems with respect to the parameter k, the optimal solution size. Building on a long line of research, this culminated in an algorithm with $O^*(1.2738^k)$

41st International Symposium on Theoretical Aspects of Computer Science (STACS 2024).
Editors: Olaf Beyersdorff, Mamadou Moustapha Kanté, Orna Kupferman, and Daniel Lokshtanov; Article No. 40; pp. 40:1–40:18
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

runtime [1]. (Throughout, we write $O^*(T)$ as shorthand for $T \cdot \mathrm{poly}(n)$.) This record has been standing for more than a decade. Assuming the Exponential Time Hypothesis, no algorithm with runtime $O^*(2^{o(k)})$ is possible [2].

1.1 Outline of our results

The main result of this paper is an improved algorithm for vertex cover parameterized by k.

▶ **Theorem 1.** *There is an algorithm for* VertexCover *with runtime $O^*(1.25284^k)$. Moreover, depending on the maximum vertex degree of the graph G, better bounds can be shown:*

- *If* $\mathrm{maxdeg}(G) \leq 3$*, we get runtime $O^*(1.14416^k)$.*
- *If* $\mathrm{maxdeg}(G) \leq 4$*, we get runtime $O^*(1.21131^k)$.*
- *If* $\mathrm{maxdeg}(G) \leq 5$*, we get runtime $O^*(1.24394^k)$.*
- *If* $\mathrm{maxdeg}(G) \leq 6$*, we get runtime $O^*(1.25214^k)$.*
- *All of these algorithms use polynomial space and are deterministic.*

The FPT algorithms for vertex cover, including our new algorithm, are built out of recursively branching on high-degree vertices or other structures, while tracking some "measure" of the graph. Our main new idea is to use a measure which is a piecewise-linear function of k and μ. To illustrate, consider branching on whether some degree-r vertex u is in the cover: we recurse on the subproblem $G_1 = G - u$ where k is reduced by one and on the subproblem $G_0 = G - u - N(u)$ where k is reduced by r. We must also show that μ is significantly reduced in the two subproblems.

Suppose that, wishfully speaking, $\vec{\frac{1}{2}}$ remains the optimal solution to LPVC(G_0) and LPVC(G_1). This is what should happen in a "generic" graph. Then $\lambda(G_0) = |V(G_0)|/2 = (n - r - 1)/2$ and $\lambda(G_1) = |V(G_1)|/2 = (n-1)/2$. Since $k_0 = k - r$ and $k_1 = k - 1$, this implies that μ is indeed significantly decreased:

$$\mu(G_0) = \mu(G) - (r-1)/2, \qquad \mu(G_1) = \mu(G) - 1/2$$

But suppose, on the other hand, that $\vec{\frac{1}{2}}$ is not the optimal solution to LPVC(G_0) or LPVC(G_1). For a concrete example, suppose the neighborhood of x in G is a subset of that of u, so G_0 has an isolated vertex x and an optimal solution θ^* to LPVC(G_0) would set $\theta^*(x) = 0$. In this situation, we develop an alternate branching rule for G: rather than branching on u itself, we branch on the two subproblems where (i) both u, x are in the cover and (ii) x is not in the cover. See Figure 1 for an illustration.

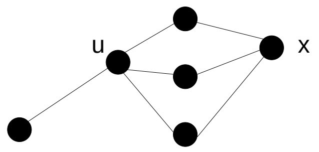

Figure 1 In the subgraph $G - N[u]$, the vertex x becomes isolated.

We emphasize that this is just one example of our branching analysis. We need to handle more general situations where $\vec{\frac{1}{2}}$ is not the optimal solution to LPVC(G_0) or LPVC(G_1), which can require more complex branching rules. We emphasize that these branching rules are more powerful than simply using conventional branching rules along with the preprocessing rules; the latter may not be able to ensure a good reduction in μ.

In Section 6, we use these ideas for a relatively simple algorithm with runtime $O^*(1.2575^k)$, along with algorithms for bounded-degree graphs. This is already much better than [1].

In the full paper, we discuss how to improve the runtime by branching on a *targeted* vertex to create additional simplifications in the graph. Carrying out this secondary goal is much more complex. This would give the results stated in Theorem 1. Due to space limitations, we omit this analysis from the present version of the paper.

1.2 Related algorithms

Many different kinds of algorithms have been developed for the Vertex Cover problem. We cannot summarize this fully here; we describe some of the most relevant works for our own paper.

As we have mentioned, there is a long line of research into FPT algorithms parametrized by the solution size k. Currently, the fastest such algorithm has $O^*(1.2738^k)$ runtime [1]. In addition, [15] describes algorithms in graphs of maximum degree 3 and 4 with runtime respectively $O^*(1.1558^k)$ and $O^*(1.2403^k)$ respectively.

An important variant is Above-Guarantee Vertex Cover (AGVC), where the parameter is the difference between k and various lower bounds on vertex cover size [10, 11, 9, 7, 8]. Of these, the parameter $\mu(G) = k - \lambda(G)$ plays a particularly important role in this paper. We quote the following main result:

▶ **Theorem 2** ([9]). *Vertex cover can be solved in time $O^*(2.3146^\mu)$.*

Another natural choice is to measure runtime in terms of the graph size n. In this setting, the problem is more commonly referred to as Maximum Independent Set (MaxIS). Xiao and Nagamochi have developed a number of algorithms in this vein. These algorithms, and in particular their performance on bounded-degree graphs, will also play a crucial role in our analysis. We refer to the algorithm targeted for graphs of maximum degree Δ by the *MaxIS-Δ* algorithm.

▶ **Theorem 3.** *MaxIS-3 can be solved with runtime $O^*(1.083506^n)$ by [17].*[1]
MaxIS-4 can be solved with runtime $O^*(1.137595^n)$ by [16].
MaxIS-5 can be solved with runtime $O^*(1.17366^n)$ by [18].[2]
MaxIS-6 can be solved with runtime $O^*(1.18922^n)$ by [19].
MaxIS-7 can be solved with runtime $O^*(1.19698^n)$ by [19].
MaxIS in graphs of arbitrary degree can be solved with runtime $O^*(1.19951^n)$ by [19].

1.3 Notation

We consider throughout a simple, undirected, unweighted graph $G = (V, E)$, and we write $n = |V|$. We write an ordered pair $\langle G, k\rangle$ for a graph G where we need to decide if there is a vertex cover of size at most k, and we say it is *feasible* if such a vertex cover exists. We say C is a *good cover* if it is a vertex cover of size at most k.

For a subset S of V, the subgraphs of G induced by S and $V \setminus S$ are denoted by $G[S]$ and $G - S$, respectively. We write $u \sim v$ if $(u, v) \in E$ and $u \nsim v$ otherwise. For vertex sets X, Y, we write $X \sim Y$ if there exists $x \in X, y \in Y$ with $x \sim y$.

[1] This runtime is not claimed directly in [17], see [16] instead.
[2] This runtime is not claimed directly in [18], see [19].

For a vertex u, the neighborhood $N_G(u)$ is the set $\{u \in V(G) : u \sim v\}$ and the closed neighborhood $N_G[u]$ is the set $N_G(u) \cup \{u\}$. Extending this notation, for a vertex set $S \subseteq V(G)$, we write $N_G(S) = \big(\bigcup_{v \in S} N_G(v)\big) \setminus S$ and $N_G[S] = N_G(S) \cup S = \bigcup_{v \in S} N_G[v]$. For readability, we sometimes write $N(x, y)$ or $N[x, y]$ as shorthand for $N(\{x, y\})$ or $N[\{x, y\}]$.

The degree of vertex v, denoted by $\deg_G(v)$, is the size of $N_G(v)$. We call a vertex of degree i an *i-vertex*; for a vertex v, we refer to a neighbor u which has degree j as a *j-neighbor of v*. An isolated vertex is a 0-vertex. We say a vertex is *subquartic* if it has degree in $\{1, 2, 3, 4\}$ and *subcubic* if it has degree in $\{1, 2, 3\}$, and *subquadratic* if it has degree in $\{1, 2\}$. (We do not count isolated vertices).

We write $\mathrm{maxdeg}(G)$ and $\mathrm{mindeg}(G)$ for the minimum and maximum vertex degrees in G, respectively. The graph is *r-regular* if $\mathrm{maxdeg}(G) = \mathrm{mindeg}(G) = r$. We write V_i for the set of degree-i vertices and $n_i = |V_i|$.

We say that a pair of vertices u, v *share neighbor* y if $y \in N(u) \cap N(v)$. We denote by $\mathrm{codeg}(u, v) = |N(u) \cap N(v)|$ the number of vertices shared by u, v.

An ℓ-cycle denotes a set of ℓ vertices $x_1, \ldots, x_\ell$ with a path $x_1, x_2, \ldots, x_\ell, x_1$.

2 Preliminaries

We review some basic facts about branching and preprocessing rules for vertex cover. Much of this material is standard, see e.g. [9]. We include proofs in Appendix A for completeness.

Our algorithm will heavily use the LP and its properties. This LP is closely related to properties of independent sets. For brevity, we refer to these as *indsets*, i.e. sets $X \subseteq V$ where no two vertices in X are adjacent. For an indset I in G, we define the *surplus* by $\mathrm{surp}_G(I) = |N_G(I)| - |I|$. If G is clear from context, we write just $\mathrm{surp}(I)$ or $N(I)$. We define $\mathrm{minsurp}(G)$ to be the minimum value of $\mathrm{surp}_G(I)$ over all *non-empty* indsets I in G, and a *min-set* to be any non-empty indset I achieving this minimum, i.e. $\mathrm{surp}_G(I) = \mathrm{minsurp}(G)$.

Since this comes up in a number of places, we define $\mathrm{minsurp}^-(G) = \min\{0, \mathrm{minsurp}(G)\}$.

▶ **Proposition 4.** $\lambda(G) = \frac{1}{2}(|V| + \mathrm{minsurp}^-(G))$. *Moreover, in a basic solution to* LPVC(G)*, the set I of vertices with value zero forms an indset (possibly empty) with* $\mathrm{surp}_G(I) = \mathrm{minsurp}^-(G)$.

We define a *critical-set* to be a non-empty indset I with $\mathrm{surp}_G(J) \geq \mathrm{surp}_G(I)$ for all non-empty subsets $J \subseteq I$. Clearly, any min-set or any singleton set is a critical-set.

▶ **Lemma 5.** *For any critical-set I, there is a good cover C with either $I \subseteq C$ or $I \cap C = \emptyset$.*
If $\mathrm{surp}(I) \leq 0$*, there is a good cover C with $I \cap C = \emptyset$.*
If $\mathrm{surp}(I) = 1$*, there is a good cover C with either $N[I] \cap C = I$ or $N[I] \cap C = N(I)$.*

▶ **Proposition 6.** *For any non-empty indset I of G, there holds* $\mathrm{minsurp}(G) \leq \mathrm{minsurp}^-(G - N[I]) + \mathrm{surp}_G(I)$*. The equality* $\mathrm{minsurp}(G) = \mathrm{minsurp}^-(G - N[I]) + \mathrm{surp}_G(I)$ *holds if and only if I is contained in a min-set of G.*

▶ **Proposition 7.** *The value* $\mathrm{minsurp}(G)$ *can be determined in polynomial time. Moreover, for any indset X (possibly $X = \emptyset$), we can efficiently find a min-set of G containing X (if any such exists).*

2.1 Preprocessing and branching rules

Our algorithm uses three major preprocessing rules. For this, we need to define the following graph structure: a *funnel* is a vertex u with a neighbor x such that $G[N(u) \setminus \{x\}]$ forms a clique. Here, we call x the *out-neighbor* of u. For example, if a degree-3 vertex u has a triangle with neighbors t_1, t_2, and one other vertex x, then u is a funnel with out-neighbor x; since this comes up so often, we refer to this as a *3-triangle*. A well-known result is that, for a funnel u with out-neighbor x, there exists a good cover C with either $u \notin C$ or $x \notin C$.

Preprocessing Rule 1 (P1): Given a critical-set I with $\mathrm{surp}_G(I) \leq 0$, form graph G' with $k' = k - |N(I)|$ by deleting $N[I]$.

Preprocessing Rule 2 (P2): Given a critical-set I with $\mathrm{surp}_G(I) = 1$, form graph G' with $k' = k - |I|$ by deleting $N[I]$ and adding a new vertex y adjacent to $N(N(I))$.

Preprocessing Rule 3 (P3): Given a funnel u with out-neighbor x, form graph G' with $k' = k - 1 - \mathrm{codeg}(u, x)$ by removing vertices $N[u] \cap N[x]$ and adding edges between all vertices in $N(u) \setminus N[x]$ to all vertices in $N(x) \setminus N[u]$.

See Figure 2 for an illustration of rule (P3) applied to a funnel.

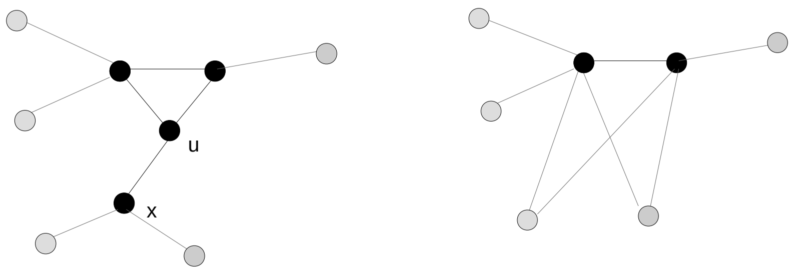

Figure 2 A funnel (here, a 3-triangle) u before (left) and after (right) applying P3.

These rules can be applied to any eligible sets in any order. Note that if a vertex v has degree zero or one, then applying (P1) to $\{v\}$ removes v and its neighbor (if any). If a vertex v has degree two, then applying (P2) to $\{v\}$ removes v and contracts its neighbors x, y into a new vertex v'. When no further preprocessing rules can be applied, we say the graph G is *simplified*; the resulting graph has a number of nice properties, for instance, it has $\mathrm{minsurp}(G) \geq 2$, and the solution $\vec{\frac{1}{2}}$ is the unique optimal solution to $\mathrm{LPVC}(G)$, and the minimum degree of any vertex is 3.

▶ **Proposition 8.** *After applying any preprocessing rule, we have $\mu(G') \leq \mu(G)$, and $\langle G, k\rangle$ is feasible if and only if $\langle G', k'\rangle$ is feasible.*

There is one situation where rule P3 is particularly powerful. Consider a 3-vertex u which has neighbors x, y, z where $x \sim y$ and $y \sim z$. We refer to this as a *kite*; see Figure 3.

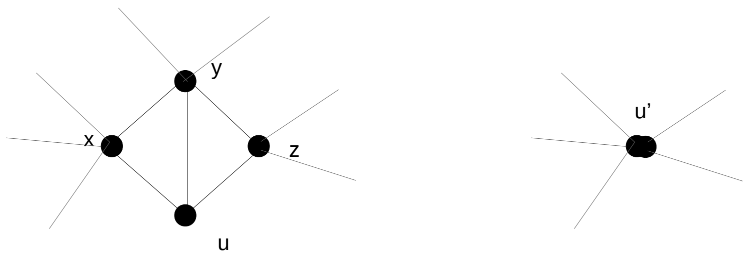

Figure 3 A kite u, x, y, z before (left) and after (right) applying P3.

▶ **Lemma 9.** *Suppose* $\operatorname{minsurp}(G) \geq 1$. *Then applying (P3) to a kite in* G *yields a graph* G' *with* $k' \leq k - 2$ *and* $\mu(G') \leq \mu(G) - 1/2$.

2.2 Runtime and branching framework

Our algorithm follows the measure-and-conquer approach. Given an input graph G, it runs some preprocessing steps and then generates subproblems $G_1, \dots, G_t$ such that G is feasible if and only if at least one G_i is feasible. It then runs recursively on each G_i. Such an algorithm has runtime $O^*(e^{\phi(G)})$ as long as the preprocessing steps have runtime $O^*(e^{\phi(G)})$ and it satisfies

$$\sum_{i=1}^{t} e^{\phi(G_i) - \phi(G)} \leq 1. \tag{1}$$

(Here and throughout $e = 2.718...$ is the base of the natural logarithm.) We refer to ϕ as the *measure* of the algorithm. For the most part, we will use measures of the form

$$\phi(G) = a\mu + bk \qquad \text{for } a, b \geq 0; \tag{2}$$

We say a subproblem G' has *drop* $(\Delta\mu, \Delta k)$ if $k' \leq k - \Delta k$ and $\mu' \leq \mu - \Delta\mu$. We say a branching rule has *branch sequence* (or *branch-seq* for short) $B = [(\Delta\mu_1, \Delta k_1), \dots, (\Delta\mu_t, \Delta k_t)]$ if generates subproblems $G'_1, \dots, G'_t$ with the given drops. Given values of a, b, we define the *value* of B to be

$$\operatorname{val}_{a,b}(B) = \sum_{i=1}^{t} e^{-a\Delta\mu_i - b\Delta k_i}. \tag{3}$$

We say branch-seq B' *dominates* B if $\operatorname{val}_{a,b}(B') \leq \operatorname{val}_{a,b}(B)$ for all $a, b \geq 0$, and G has a branch-seq B *available* if we can efficiently find a branching rule which has some branch-seq B' dominating B. In general, a "balanced" branch-seq dominates an "imbalanced" one; formally, if $\Delta k_1 \leq \Delta k_2$ and $\Delta\mu_1 \leq \Delta\mu_2$, then branch-seq $[(\Delta\mu_1, \Delta k_1), (\Delta\mu_2, \Delta k_2)]$ dominates $[(\Delta\mu_1 - f, \Delta k_1 - g), (\Delta\mu_1 + f, \Delta k_2 + g)]$ for any $f, g \geq 0$.

We use the following important compositional property throughout: if we have a branching rule B generating ℓ subproblems with drops $(\Delta\mu_i, \Delta k_i)$, and we then apply a branching rule with branch-seq B'_i to each subproblem $i = 1, \dots, \ell$, then, overall, we get

$$\operatorname{val}_{a,b}(B) = \sum_{i=1}^{\ell} e^{-a\Delta\mu_i - b\Delta k_i} \operatorname{val}_{a,b}(B'_i).$$

▶ **Lemma 10.** *Suppose that* $\mathcal{G}$ *is a class of graphs closed under vertex deletion, and for which there are vertex cover algorithms with runtimes* $O^*(e^{a\mu + bk})$ *and* $O(e^{cn})$ *for* $a, b, c \geq 0$. *Then we can solve vertex cover in* $\mathcal{G}$ *with runtime* $O^*(e^{dk})$ *where* $d = \frac{2c(a+b)}{a+2c}$.

To illustrate Lemma 10, consider graphs of maximum degree 3. We can combine the AGVC algorithm with runtime $O^*(2.3146^{\mu})$ (corresponding to $a = \log(2.3146), b = 0$), and the MaxIS-3 algorithm with runtime $O^*(1.083506^{n})$ (corresponding to $c = \log(1.083506)$), to get an algorithm with runtime $O^*(1.14416^{k})$; this already gives us one of the results in Theorem 1.

3 Preprocessing rules and reductions in k

We define $S(G)$ to be the largest decrease in k obtainable from applying rules (P1) – (P3) exhaustively via some efficiently-computable sequence of operations.[3] As a point of notation, we define $R(X) = S(G - X)$ for a vertex set X. For vertices $x_1, \dots, x_\ell$ and vertex sets $X_1, \dots, X_r$, we write $R(x_1, \dots, x_\ell, X_1, \dots, X_r)$ as shorthand for the more cumbersome $R(\{x_1, \dots, x_\ell\} \cup X_1 \cup \dots \cup X_r) = S(G - \{x_1, \dots, x_\ell\} - (X_1 \cup \dots \cup X_r))$. Note that we may have $\ell = 0$ or $r = 0$, e.g. $R(x_1, \dots, x_\ell)$ is shorthand for $R(\{x_1, \dots, x_\ell\})$.

We record a few observations on simplifications of various structures.

▶ **Observation 11.** *Suppose x, y are non-adjacent subquadratic vertices. Then $S(G) \geq 2$ if any of the following three conditions hold: (i) $\operatorname{codeg}(x, y) = 0$ or (ii) $\deg(x) + \deg(y) \geq 3$ or (iii) $\operatorname{minsurp}(G) \geq 0$.*

Proof. If $\deg(x) = \deg(y) = 1$ and $\operatorname{codeg}(x, y) = 0$, then $\{x, y\}$ is a min-set with $\operatorname{surp}(\{x, y\}) = 0$, and applying P1 reduces k by two. Otherwise, if $\deg(x) \geq 2$, then we can apply (P2) to x, and vertex y remains subquadratic, and we can follow up by applying (P1) or (P2) again to y. Note that if $\operatorname{minsurp}(G) \geq 0$, then at least one of the conditions (i) or (ii) must hold. ◀

▶ **Proposition 12.** *If G has 2-vertices $x_1, \dots, x_\ell$ which all have pairwise distance at least 3, then $S(G) \geq \ell$.*

Proof. We show it by induction on ℓ. The base case $\ell = 0$ is vacuous. For the induction step, we apply (P2) to x_ℓ, forming a graph G' where the neighbors y, z get contracted into a single new vertex t. By hypothesis, y, z are distinct from $x_1, \dots, x_\ell$. We claim that $\operatorname{dist}_{G'}(x_i, x_j) \geq 3$ and $\deg_{G'}(x_i) \geq 2$ for any pair $i < j < \ell$. For, clearly $x_i \not\sim x_j$ in G'. If x_i, x_j share a neighbor in G', it must be vertex t as no other vertices were modified. This is only possible if $\{x_i, x_j\} \sim \{y, z\}$; but this contradicts our hypothesis that $\operatorname{codeg}(x_i, x_\ell) = \operatorname{codeg}(x_j, x_\ell) = 0$. Finally, suppose that $\deg_{G'}(x_i) \leq 1$. This is only possible if the two neighbors of x_i got merged together, i.e. $x_i \sim y$ and $x_i \sim z$. Again, this contradicts that $\operatorname{codeg}(x_i, x_\ell) = 0$.

By induction hypothesis applied to G', we have $S(G') \geq \ell - 1$ and hence $S(G) \geq \ell$. ◀

▶ **Lemma 13.** *If $\operatorname{minsurp}(G) \geq 0$ and indset I has $\operatorname{surp}_G(I) \leq 1$, then $S(G) \geq |I|$.*

Proof. Let $r = |I|$. If $\operatorname{surp}_G(I) = 0$, or I is a critical-set, we can apply (P1) or (P2) to I. So suppose that $\operatorname{surp}_G(I) = 1$ and $\operatorname{surp}_G(X) = 0$ for a non-empty subset $X \subsetneq I$; among all such subsets X, choose one of maximum size, and let $s = |X|$. We apply (P1) to X, reducing k by s and obtaining a graph $G' = G - N[X]$. Now consider the indset $J = I \setminus X$ in G', where $|J| = r - s$. We have $|N_{G'}(J)| = |N_G(I) \setminus N_G(X)| = r + 1 - s$ and so $\operatorname{surp}_{G'}(J) = 1$.

We claim that J is a critical-set in G'. For, if some non-empty subset $X' \subseteq J$ has $|N_{G'}(X')| \leq |X'|$, then indset $I' = X \cup X' \subseteq I$ would have $|N_G(I')| \leq |N_G(X)| + |N_{G'}(X')| \leq |X| + |X'| = |I'|$. Hence $\operatorname{surp}_G(I') \leq 0$, contradicting maximality of X. So, we can apply (P2) to J in G', getting a further drop of $r - s$. Overall, we get a net drop of $s + (r - s) = r$ as desired. ◀

[3] It is not clear how to calculate the *absolute* largest decrease in k via preprocessing rules, since applying some rules may prevent opportunities for other rules. We assume that we have fixed some polynomial-time computable sequences of potential preprocessing rules in order to reduce k as small as much as possible. At various points in our algorithm, we will describe simplifications available for intermediate graphs. We always assume that our preprocessing rules are chosen to find such simplifications.

4 Branching rules

Most of our branching rules can be viewed in terms of guessing that a certain set of vertices X_1 is in the cover. In the graph $G - X_1$, a set of vertices X_0 may become isolated. We apply rule (P1) to remove X_0; the resulting subproblem $\langle G - (X_0 \cup X_1), k - |X_1| \rangle$ is called *principal*, and we define the *excess* to be $|X_0|$. Note that the graph $G' = G - (X_0 \cup X_1)$ may still contain isolated vertices, or have additional simplifications available. We say $\langle G', k - |X_1| \rangle$ has drop $(\Delta\mu', |X_1|)$ *directly*, and the final subproblem $G'' = \langle G'', k'' \rangle$ has drop $(\Delta\mu'', \Delta k'')$ *after simplification.*

Since this plays a central role in our analysis, we define the *shadow* of a vertex set X, denoted $\operatorname{shad}(X)$, to be $\operatorname{minsurp}(G - X)$. For readability, we write $\operatorname{shad}(x_1, \dots, x_\ell, X_1, \dots, X_r)$ as shorthand for $\operatorname{shad}(\{x_1, \dots, x_\ell\} \cup X_1 \cup \dots \cup X_r) = \operatorname{minsurp}(G - \{x_1, \dots, x_\ell\} - (X_1 \cup \dots \cup X_r))$.

▶ **Proposition 14.** *If* $\operatorname{minsurp}(G) \geq 2$, *then a principal subproblem* $\langle G - X, k' \rangle$ *with excess* s *has* $\Delta\mu = \frac{1}{2}(\Delta k - s + \operatorname{minsurp}^-(G')) = \frac{1}{2}(\Delta k - s + \min\{0, \operatorname{shad}(X)\})$.

Proof. Here Proposition 4 gives $\lambda(G) = n/2$. Let $G' = G - X$, where $|X| = \Delta k + s$. Then $\mu(G') = k' - \lambda(G') = k' - \frac{1}{2}(n' + \operatorname{minsurp}^-(G'))$, where $n' = n - X = 2\lambda(G) - (s + \Delta k)$, which simplifies to $\mu(G') = \mu(G) - \frac{1}{2}(\Delta k - s + \operatorname{minsurp}^-(G'))$ as desired. ◀

▶ **Observation 15.** *Suppose* $\operatorname{minsurp}(G) \geq 2$. *Then for any indset* I, *there holds* $\operatorname{shad}(I) \geq 2 - |I|$ *and* $\operatorname{shad}(N[I]) \geq 2 - \operatorname{surp}_G(I)$.

In particular, for a vertex u, *there holds* $\operatorname{shad}(u) \geq 1$ *and* $\operatorname{shad}(N[u]) \geq 3 - \deg(u)$.

Proof. If J is a min-set of $G - I$, then $2 \leq \operatorname{surp}_G(J) \leq \operatorname{surp}_{G-I}(J) + |I|$. So $\operatorname{surp}_{G-I}(J) \geq 2 - |I|$. Similarly, if J is a min-set of $G - N[I]$, then $2 \leq \operatorname{surp}_G(I \cup J) = \operatorname{surp}_{G-N[I]}(J) + \operatorname{surp}_G(I)$. ◀

By combining Observation 11 and Observation 15, we immediately get the following simplification rule (which is used ubiquitously):

▶ **Observation 16.** *If* G *is simplified and* $G - \{u, v\}$ *has two non-adjacent subquadratic vertices, then* $R(u, v) \geq 2$.

Our bread-and-butter branching rule is to choose some vertex u, and branch on whether it is in the cover. We refer to this as *splitting on* u. This generates two principal subproblems $\langle G - u, k - 1 \rangle$ and $\langle G - N[u], k - \deg(u) \rangle$. More generally, in light of Lemma 5, we know that for any critical-set I, there is either a good cover containing I or omitting I. We can branch on subproblems $\langle G - I, k - |I| \rangle$ and $\langle G - N[I], k - N[I] \rangle$, and we refer to this as *splitting on* I. Splitting on a vertex is a special case where I is a singleton.

Consider the effect of splitting on u. By Observation 15, the subproblem $\langle G - u, k - 1 \rangle$ has drop $(0.5, 1)$ and the subproblem $\langle G - N[u], k - \deg(u) \rangle$ has drop $(1, \deg(u))$. The latter bound, however, is very loose and should be improved to get an optimized analysis. In light of Proposition 14, it revolves around the value of $\operatorname{shad}(N[u])$. If $\operatorname{shad}(N[u]) \geq 0$, then λ drops by $\frac{\deg(u)+1}{2}$ and so the subproblem has drop $(\frac{\deg(u)-1}{2}, \deg(u))$. This is the "generic" situation for splitting on u.

When $\operatorname{shad}(N[u]) \leq 0$, we say that u is *blocked.* If x is a vertex in a min-set of $G - N[u]$, we say that x is a *blocker* of u. Necessarily, $x \not\sim u$. This motivates the following powerful branching rule:

(B) If x is a blocker of vertices $u_1, \dots, u_\ell$, then branch on subproblems $\langle G - \{u_1, \dots, u_\ell, x\}, k - \ell - 1 \rangle$ and $\langle G - N[x], k - \deg(x) \rangle$.

▶ **Proposition 17.** *Rule (B) is a valid branching rule.*

Proof. The subproblem $\langle G - N[x], k - \deg(x)\rangle$ is feasible if and only if G has a good cover omitting x. The subproblem $\langle G - \{u_1, \dots, u_\ell, x\}, k - \ell - 1\rangle$ is feasible if and only if G has a good cover which includes all vertices $u_1, \dots, u_\ell, x$. We claim that, if G is feasible, at least one of these cases holds. For, suppose a good cover of G omits vertex u_i. Then $G - N[u_i]$ has a cover of size at most $k - \deg(u_i)$. By Lemma 5, this implies that $G - N[u_i]$ has such a cover C' which omits any min-set I of $G - N[u_i]$, and in particular $x \notin C'$. Then G has a good cover $C \cup N(u_i)$ omitting x. ◀

In the vast majority of cases where we use (B), we have $\ell = 1$, that is, x is a blocker of a single vertex u. To provide intuition, keep in mind the picture that if $G - N[u]$ has s isolated vertices $v_1, \dots, v_s$, then $\{v_1, \dots, v_s\}$ would be an indset in $G - N[u]$ with zero neighbors and surplus $-s$, in particular, $\text{shad}(N[u]) \le -s$. These vertices were hidden in the "shadow" of $N[u]$. See Figure 4.

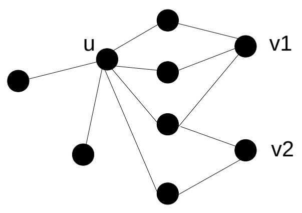

Figure 4 Here, vertices v_1, v_2 are both blockers of u; in particular, we have $\text{shad}(N[u]) \le -2$.

Also, note that if $\text{shad}(N[u]) = 0$ exactly, then we could still split on u if desired and get the ideal drop in μ. However, we can also apply rule (B) if desired, and this is usually more profitable. Vertices with $\text{shad}(N[u]) = 0$ share many properties with vertices with $\text{shad}(N[u]) < 0$ strictly. This is the reason for the somewhat unintuitive definition of *blocked vertex*.

5 Analysis of blockers and splitting

We will develop a series of branching rules (typically using rule (B)) to handle blocked vertices, along with a number of other related cases. The most significant consequence of these rules is that we are able to get near-ideal branch sequences for high-degree vertices. Specifically, we will show the following main result which is the heart of our branching algorithm:

▶ **Theorem 18.** *Suppose G is simplified, and let $r = \text{maxdeg}(G)$. Depending on r, then following branch-seqs are available:*
If $r \ge 4$: $[(1,3),(1,5)]$ or $[(0.5,1),(1.5,4)]$.
If $r \ge 5$: $[(1,3),(1,5)]$ or $[(0.5,1),(2,5)]$.
If $r \ge 6$: $[(1,3),(1,5)]$, $[(0.5,2),(2,5)]$, or $[(0.5,1),(2.5,r)]$.

The proof of Theorem 18 has many cases which will require us to build up slowly from analysis of low-degree vertices. **For Section 5 only, we always assume that the starting graph G is simplified.** We begin with a few elementary observations.

▶ **Observation 19.** *Suppose u is blocked and let I be a min-set of $G - N[u]$. There is at least one vertex $x \in I$ with $\text{codeg}(x, u) \ge 1$.*

Proof. If not, we would have $\mathrm{surp}_G(I) = \mathrm{surp}_{G-N[u]}(I) \leq 0$, contradicting that G is simplified. ◀

▶ **Proposition 20.** *Suppose vertex u has $\mathrm{shad}(N[u]) \leq 4 - \deg(u)$ and x is a blocker of u. If we apply (B) to vertices u, x, then subproblem $G' = \langle G - N[x], k - \deg(x) \rangle$ has drop $(1, 4)$. Moreover, if $\deg(x) = 3$ and x is contained in a non-singleton min-set of $G - N[u]$, then G' has drop $(1, 5)$.*

Proof. Since G is simplified, we have $\deg(x) \geq 3$ and $\mathrm{minsurp}(G) \geq 2$. If $\deg(x) = 4$, then G' has drop $(1, 4)$ directly. So, suppose $\deg(x) = 3$ exactly. Consider any min-set I of $G - N[u]$ with $x \in I$. The indset $J = I \cup \{u\} \setminus \{x\}$ has $\mathrm{surp}_{G-N[x]}(J) \leq \mathrm{surp}_{G-N[u]}(I) + (\deg(u) - \deg(x)) \leq 1$. On the other hand, $\mathrm{shad}(N[x]) \geq \mathrm{minsurp}(G) - \deg(x) + 1 \geq 0$. So Lemma 13 gives $R(N[x]) \geq |J| = |I|$.

By Proposition 14 we see that G' has drop $(1, 3)$ directly. Since $|I| \geq 1$ trivially, we have $R(N[x]) \geq 1$ and G' has drop $(1, 4)$ after simplification. If $|I| \geq 2$, then $R(N[x]) \geq 2$ and likewise G' has drop $(1, 5)$ after simplification. ◀

5.1 Branching rules for indsets with surplus two

It will be very important to develop special branching rules for indsets with surplus two. Since G is assumed to be simplified, note that any such set I will be a critical set, and we can split on I to get subproblems $\langle G - I, k - |I| \rangle$ and $\langle G - N[I], k - |I| - 2 \rangle$. However, this branching rule is not powerful enough for us; we develop specialized branching rules for smaller indsets (indsets with cardinality two or three).

▶ **Proposition 21.** *Suppose G has an indset I with $\mathrm{surp}_G(I) = 2$, and let $z \in N(I)$. If we split on z, the subproblem $\langle G - z, k - 1 \rangle$ has drop $(0.5, 1 + |I|)$. If z has a blocker $x \notin I$ and we apply (B) to z, x, the subproblem $\langle G - \{z, x\}, k - 2 \rangle$ has drop $(1, 2 + |I|)$.*

Proof. Subproblem $\langle G - z, k - 1 \rangle$ has drop $(0.5, 1)$ directly. We have $\mathrm{surp}_{G-z}(I) = \mathrm{surp}_G(I) - 1 = 1$, and $\mathrm{shad}(G - z) \geq \mathrm{minsurp}(G) - 1 \geq 1$. So (P2) applied to I gives $R(z) \geq |I|$. Likewise, subproblem $\langle G - \{z, x\}, k - 2 \rangle$ has drop $(1, 2)$ directly, and $\mathrm{shad}(z, x) \geq \mathrm{minsurp}(G) - 2 \geq 0$ and $\mathrm{surp}_{G-\{z,x\}}(I) \leq \mathrm{surp}_G(I) - 1 \leq 1$. So Lemma 13 gives $R(z, x) \geq |I|$. ◀

▶ **Lemma 22.** *Suppose G has an indset I with $\mathrm{surp}_G(I) = 2, |I| \geq 3$. Then G has available branch-seq $[(1, 4), (1, 5)]$ or $[(0.5, 4), (2, 5)]$.*

Proof. If $|I| \geq 4$, then splitting on I gives branch-seq $[(1, 4), (1, 6)]$. So suppose $|I| = 3$. We first claim that there is some vertex z with $|N(z) \cap I| = 2$. For, consider the set of vertices $J = \{z : |N(z) \cap I| \geq 2\}$. The vertices in I are independent and have degree at least three, so there are at least 9 edges from I to $N(I)$. Since $|N(I)| = 5$, by the pigeonhole principle we must have $|J| \geq 2$. If $N(z) \not\subseteq I$ for all $z \in J$, then J would be an indset with surplus at most one, contradicting that G is simplified. Hence there is $z \in J$ with $|N(z) \cap I| = 2$ exactly.

Let z be any such vertex, and let $I = \{x_1, x_2, y\}$ where $z \sim x_1, z \sim x_2, z \not\sim y$ and $\deg(x_1) \leq \deg(x_2)$. If $\deg(z) \leq 4$, then we split on I, generating subproblems $\langle G - I, k - 3 \rangle$ and $\langle G - N[I], k - 5 \rangle$ with drops $(1, 3)$ and $(1, 5)$ directly, where $G - I$ has a subquadratic vertex z so it has drop $(1, 4)$ after simplification. If $\deg(z) \geq 5$ and $\mathrm{shad}(N[z]) \geq 5 - \deg(z)$, then we split on z; by Proposition 21, subproblem $\langle G - z, k - 1 \rangle$ has drop $(0.5, 4)$, and subproblem $\langle G - N[z], k - \deg(z) \rangle$ has drop $(2, 5)$ directly. If $\deg(z) \geq 5$ and z has a blocker $t \neq y$, then we apply (B) to z, t; by Proposition 20, subproblem $\langle G - N[t], k - \deg(t) \rangle$ has drop $(1, 4)$ and by Proposition 21, subproblem $\langle G - \{z, t\}, k - 2 \rangle$ has drop $(1, 5)$.

The only remaining possibility is if $\deg(z) \geq 5$ and y is the only blocker of z, i.e. $\deg(z) = 5, \deg(y) = 3$ and $N(y) \subseteq N(z)$. Each vertex x_i has $\deg(x_i) + \deg(y) - \operatorname{codeg}(x_i, y) \leq |N(I)|$, i.e. $\operatorname{codeg}(x_i, y) \geq \deg(x_i) - 2$. If $\deg(x_1) = 3$, this implies that x_1, y share some neighbor u; but in this case G has a 3-triangle x_1, u, z, contradicting that G is simplified.

So we suppose $\deg(x_2) \geq \deg(x_1) \geq 4$ and then $\operatorname{codeg}(x_1, y) \geq 2$ and $\operatorname{codeg}(x_2, y) \geq 2$. Equivalently, $\deg_{G-N[x_1]}(y) \leq 1$ and $\deg_{G-N[x_2]}(y) \leq 1$. So y is a blocker of all three vertices x_1, x_2, z. We apply (B) giving subproblems $\langle G-N[y], k-\deg(y)\rangle$ and $\langle G-\{x_1, x_2, z, y\}, k-4\rangle$, with drops $(1, 3)$ and $(1, 4)$ directly. Furthermore, $\operatorname{surp}_{G-N[y]}(\{x_1, x_2\}) \leq 2 - |N_{G-N[y]}(I)| = 2 - 2 = 0$ and neither x_1 or x_2 is isolated in $G - N[y]$ (they retain neighbor z). So we can apply (P1) to indset $\{x_1, x_2\}$ in $G - N[y]$, giving $R(N[y]) \geq 2$. So $G - N[y]$ has drop $(1, 5)$ after simplification. ◀

▶ **Lemma 23.** *Suppose G has an indset I with $\operatorname{surp}_G(I) = 2, |I| \geq 2$. Then G has available branch-seq $[(1, 4), (1, 4)]$ or $[(0.5, 3), (2, 5)]$.*

Proof. If $|I| \geq 3$, then we apply Lemma 22. So let $I = \{x, y\}$, where $\deg(x) \leq \deg(y)$ and let $A = N(x, y)$. Note that $|A| = \deg(x) + \deg(y) - 4 \geq \deg(x) - 1$. The vertices in A cannot form a clique, as then x would have a funnel. So consider vertices $z_1, z_2 \in A$ with $z_1 \not\sim z_2$.

If z_1 and z_2 are both subquartic, then we split on I; the subproblem $\langle G-N[I], k-|N[I]|-2$ has drop $(1, 4)$ directly. The subproblem $G - I = G - \{x, y\}$ has non-adjacent subquadratic vertices z_1, z_2, so by Observation 16, we have $R(x, y) \geq 2$ and subproblem $\langle G - I, k - |I|\rangle$ has drop $(1, 4)$ after simplification.

So suppose $\deg(z_1) \geq 5$. If $\operatorname{shad}(N[z_1]) \geq 0$, then split on z_1; by Proposition 21, the subproblem $\langle G - z_1, k - 1\rangle$ has drop $(0.5, 3)$, while the subproblem $\langle G - N[z_1], k - \deg(z_1)\rangle$ has drop $(2, 5)$ directly. Otherwise, suppose $\operatorname{shad}(N[z_1]) \leq -1$ and t is a blocker of z_1. Note that $t \notin I$ since $t \not\sim z_1$ whereas $z_1 \in N(x) \cap N(y)$. We then apply (B) to t, z_1; the subproblem $\langle G - N[z_1], k - \deg(z_1)\rangle$ has drop $(1, 5)$ directly, and by Proposition 21, the subproblem $\langle G - \{t, z\}, k - 2\rangle$ has drop $(1, 4)$. ◀

5.2 Branching rules for blocked vertices

We now turn to developing branching rules for blocked vertices of various types. We need to work our way from low to high degree.

▶ **Proposition 24.** *Suppose vertex u has $\operatorname{shad}(N[u]) \leq 5 - \deg(u)$ and u has a blocker x of degree at least 4. Then G has available branch-seq $[(1, 4), (1, 4)]$ or $[(0.5, 2), (2, 5)]$.*

Proof. If $\operatorname{shad}(N[x]) \leq 3 - \deg(x)$ and J is a min-set of $G - N[x]$, then $\operatorname{surp}_G(J \cup \{x\}) \leq (3 - \deg(x)) + (\deg(x) - 1) \leq 2$ and we can apply Lemma 23 to indset $J \cup \{x\}$.

So suppose $\operatorname{shad}(N[x]) \geq 4 - \deg(x)$. In this case, we apply (B) to u, x. Subproblem $\langle G - \{u, x\}, k - 2\rangle$ has drop $(1, 2)$ directly. If $\deg(x) \geq 5$, then subproblem $\langle G - N[x], k - \deg(x)\rangle$ has drop $(1.5, 5)$ directly. If $\deg(x) = 4$, then consider a min-set I of $G - N[u]$ with $x \in I$. We have $\operatorname{surp}_{G-N[x]}(I \cup \{u\} \setminus \{x\}) \leq \operatorname{surp}_{G-N[u]}(I) + \deg(u) - \deg(x) \leq 1$. So $R(N[x]) \geq 1$ and G' has drop $(1.5, 5)$ after simplification. In either case, $\langle G-\{u, x\}, k-2\rangle$ and $\langle G - N[x], k - \deg(x)\rangle$ give drops $(1.2), (1.5, 5)$ respectively. This branch-seq $[(1, 2), (1.5, 5)]$ dominates $[(0.5, 2), (2, 5)]$. ◀

▶ **Lemma 25.** *Suppose G has a vertex of degree at least 5 which has a 3-neighbor. Then G has available branch-seq $[(1, 3), (1, 5)]$ or $[(0.5, 2), (2.5)]$.*

Proof. Let u be a vertex of degree at least 5 and let z be a 3-neighbor of u. There are a number of cases to consider.

Case I: $\mathrm{shad}(N[u]) \geq 5 - \deg(u)$. Splitting on u gives subproblems $G - u, G - N[u]$ with drops $(0.5, 1), (2, 5)$ directly; the former has drop $(0.5, 2)$ after simplification due to its 2-neighbor z.

Case II: u has a blocker x with $\deg(x) \geq 4$. We suppose that $\mathrm{shad}(N[u]) \leq 4 - \deg(u)$ since otherwise it would be covered in Case I. Then the result follows from Proposition 24, noting that $[(1, 4), (1, 4)]$ dominates $[(1, 3), (1, 5)]$.

Case III: u has a blocker x with $R(u, x) \geq 2$. We apply (B) to vertices u, x. Subproblem $\langle G - \{u, x\}, k - 2\rangle$ has drop $(1, 4)$ after simplification, and by Proposition 20 subproblem $\langle G - N[x], k - \deg(x)\rangle$ has drop $(1, 4)$. We get branch-seq $[(1, 4), (1, 4)]$, which dominates $[(1, 3), (1, 5)]$.

Case IV: $G - N[u]$ has some non-singleton min-set I. Let $x \in I$. Necessarily $\deg(x) = 3$, else it would be covered in Case II. We apply (B) to u, x; subproblem $\langle G - \{u, x\}, k - 2\rangle$ has drop $(1, 3)$ after simplification due to subquadratic vertex z. By Proposition 20 subproblem $\langle G - N[x], k - \deg(x)\rangle$ has drop $(1, 5)$ after simplification.

Case V: No previous cases apply. Suppose that no vertices in the graph are covered by any of the previous cases. Let U denote the non-empty set of vertices u with $\deg(u) \geq 5$ and u having a 3-neighbor. We claim that every vertex $u \in U$ has degree exactly 5. For, suppose $\deg(u) \geq 6$, and since we are not in Case I we have $\mathrm{shad}(N[u]) \leq 4 - \deg(u) \leq -2$. Then necessarily a min-set of $G - N[u]$ would have size at least two, and it would be covered in Case IV.

So every $u \in U$ has degree exactly 5, and is blocked by a singleton 3-vertex x, i.e. $N(x) \subseteq N(u)$. Let us say in this case that x is *linked* to u; we denote by X the set of all such 3-vertices linked to any vertex in U. We claim that each $x \in X$ has at least two neighbors of degree at least 5, which must be in U due to their 3-neighbor x. For, suppose that x has two subquartic neighbors z_1, z_2; necessarily $z_1 \not\sim z_2$ since G is simplified and x has degree 3. Then $R(u, x) \geq 2$ by Observation 16, which would have been covered in Case III.

On the other hand, we claim that each vertex $u \in U$ has at most one neighbor in X. For, suppose u is linked to x and u has two 3-neighbors $x_1, x_2 \in X$, which must be non-adjacent since G is simplified. Then Observation 16 gives $R(u, x) \geq 2$ due to subquadratic vertices x_1, x_2. This would have been covered in Case III.

Thus, we see that each vertex in X has at least two neighbors in U and each vertex in U has at most one neighbor in X. Hence $|U| \geq 2|X|$, and so by the pigeonhole principle, there must be some vertex $x \in X$ which is linked to two vertices $u_1, u_2 \in U$. We apply (B) to u_1, u_2, x, getting subproblems $\langle G - \{u_1, u_2, x\}, k - 3\rangle$ and $\langle G - N[x], k - \deg(x)\rangle$. By Proposition 20, the latter subproblem has drop $(1, 4)$. Furthermore, if z_1, z_2 are two 5-neighbors of x, then $z_1 \not\sim z_2$ else x would have a 3-triangle. So $G - \{u_1, u_2, x\}$ has non-adjacent 2-vertices z_1, z_2, and subproblem $\langle G - \{u_1, u_2, x\}, k - 3\rangle$ has drop $(1, 5)$ after simplification. ◀

▶ **Proposition 26.** *Suppose a vertex u has* $\deg(u) \geq 4$ *and* $\mathrm{shad}(N[u]) \leq 4 - \deg(u)$. *Then G has available branch-seq* $[(1, 3), (1, 5)]$ *or* $[(0.5, 2), (2, 5)]$.

Proof. Let I be a min-set of $G - N[u]$. By Observation 19, there is a vertex $x \in I$ which shares some neighbor t with u. If $\deg(x) \geq 4$, then we apply Proposition 24, noting that $[(1, 4), (1, 4)]$ dominates $[(1, 3), (1, 5)]$. If $\deg(x) = 3$ and $\deg(t) \geq 5$, we apply Lemma 25 to

t. If $\deg(x) = 3, \deg(t) \leq 4, |I| \geq 2$, we apply (B) to u, x; subproblem $\langle G - \{u, x\}, k - 2\rangle$ has drop $(1, 3)$ after simplification (due to subquadratic vertex t) and by Proposition 20 subproblem $\langle G - N[x], k - \deg(x)\rangle$ has drop $(1, 5)$.

So finally take $I = \{x\}$ where $\deg(x) = 3$ and x shares two neighbors t_1, t_2 with u. Necessarily $t_1 \not\sim t_2$ since G has no 3-triangles. If either t_1 or t_2 has degree 5 or more, we can apply Lemma 25. If they are both subquartic, then we apply (B) to u, x; by Proposition 20, subproblem $\langle G - N[x], k - \deg(x)\rangle$ has drop $(1, 4)$ and by Observation 16 there holds $R(u, x) \geq 2$ and subproblem $\langle G - \{u, x\}, k - 2\rangle$ has drop $(1, 4)$ after simplification. Here, $[(1, 4), (1, 4)]$ dominates $[(1, 3), (1, 5)]$. ◀

▶ **Lemma 27.** *Suppose vertex u has $\deg(u) \geq 5$ and $\text{shad}(N[u]) \leq 5 - \deg(u)$.*

- *If $\deg(u) = 5$, then G has available branch-seq $[(1, 3), (1, 4)]$ or $[(0.5, 2), (2, 5)]$.*
- *If $\deg(u) \geq 6$, then G has available branch-seq $[(1, 3), (1, 5)]$ or $[(0.5, 2), (2, 5)]$.*

Proof. Let I be a min-set of $G - N[u]$ of smallest size. If any vertex in I has degree at least 4, the result follows from Proposition 24 (noting that $[(1, 4), (1, 4)]$ dominates $[(1, 3), (1, 5)]$. So suppose that all vertices in I have degree 3. By Observation 19, we may select some $x \in I$ with $\text{codeg}(x, u) \geq 1$. For this vertex x, define $Z = N(x) \cap N(u)$ and $Y = N_{G-N[u]}(x)$, where $|Z| + |Y| = \deg(x) = 3$ and $|Z| \geq 1$.

If any vertex $z \in Z$ has degree at least 5, we apply Lemma 25 to z with its 3-neighbor x. If any vertex $z \in Z$ has degree 3, we apply Lemma 25 to u with its 3-neighbor z. If $\text{codeg}(x, x') \geq 2$ for any vertex $x' \in I \setminus \{x\}$, then we apply Lemma 23 to the surplus-two indset $\{x, x'\}$. So we suppose that $\text{codeg}(x, x') \leq 1$ for all $x' \in I \setminus \{x\}$ and that all vertices in Z have degree exactly four.

The vertices in Z must be independent, as otherwise G would have a 3-triangle with x. If $\text{codeg}(z, z') \geq 3$ for any vertices $z, z' \in Z$, then $\text{shad}(N[z]) \leq 0$ (since $G - N[z]$ has a 1-vertex z'), and we can apply Proposition 26. So we suppose the vertices in Z share no neighbors besides u and x. In this case, in the graph $G - \{u, x\}$, the vertices in Z become subquadratic and share no common neighbors. By Proposition 12, we thus have $R(u, x) \geq |Z|$.

We next claim that if $|I| \geq 2$, then $R(u, x) \geq |Y|$. For, every vertex $y \in Y$ must have a neighbor $\sigma(y) \in I \setminus \{x\}$, as otherwise we would have $\text{surp}_{G-N[u]}(I') \leq \text{surp}_{G-N[u]}(I)$ for non-empty ind-set $I' = I \setminus \{x\}$, contradicting minimality of I. The vertices $\sigma(y) : y \in Y$ must be distinct, as a common vertex $x' = \sigma(y) = \sigma(y')$ would have $\text{codeg}(x, x') \geq 2$ which we have already ruled out. So, in the graph $G - N[x]$, each vertex $\sigma(y) \in I$ loses neighbor y. By Observation 15, we have $\text{shad}(N[x]) \geq 3 - \deg(x) = 0$. So $G - N[x]$ has $|Y| \leq 2$ non-adjacent subquadratic vertices. Observation 16 implies that $R(N[x]) \geq |Y|$.

At this point, our strategy is to apply (B) to u, x. If $|I| = 1$, then either $\deg(u) = 5, |Z| \geq 2$ or $\deg(u) \geq 6, |Z| = 3$. Subproblem $\langle G - N[x], k - \deg(x)\rangle$ has drop $(1, 3)$ directly and subproblem $\langle G - \{u, x\}, k - 2\rangle$ has drop $(1, 2 + |Z|)$ after simplification; that is, it has drop $(1, 4)$ if $\deg(u) = 5$ and drop $(1, 5)$ otherwise.

Otherwise, if $|I| \geq 2$, then subproblem $\langle G - \{u, x\}, k - 2\rangle$ has drop $(1, 2 + |Z|)$ after simplification and subproblem $\langle G - N[x], k - \deg(x)\rangle$ has drop $(1, 3 + |Y|)$ after simplification. Since $|Z| + |Y| = 3$ and $|Z| \geq 1$, this always gives drops $(1, 3), (1, 5)$ or $(1, 4), (1, 4)$. ◀

We can finally prove Theorem 18 (restated for convenience).

▶ **Theorem 18.** *Suppose G is simplified, and let $r = \text{maxdeg}(G)$. Depending on r, then following branch-seqs are available:*
If $r \geq 4$: $[(1, 3), (1, 5)]$ *or* $[(0.5, 1), (1.5, 4)]$.
If $r \geq 5$: $[(1, 3), (1, 5)]$ *or* $[(0.5, 1), (2, 5)]$.
If $r \geq 6$: $[(1, 3), (1, 5)]$, $[(0.5, 2), (2, 5)]$, *or* $[(0.5, 1), (2.5, r)]$.

Proof. Consider an r-vertex u. If $r = 4$ and $\mathrm{shad}(N[u]) \geq 0$, then split on u with branch-seq $[(0.5, 1), (1.5, 4)]$. If $r = 4$ and $\mathrm{shad}(N[u]) \leq -1$, then use Proposition 26. If $r = 5$ and $\mathrm{shad}(N[u]) \geq 0$, then split on u with branch-seq $[(0.5, 1), (2, 5)]$. If $r = 5$ and $\mathrm{shad}(N[u]) \leq -1$, then apply Proposition 26. If $r \geq 6$ and $\mathrm{shad}(N[u]) \geq 6 - r$, then split on u, getting branch-seq $[(0.5, 1), (2.5, r)]$. If $r \geq 6$ and $\mathrm{shad}(N[u]) \leq 5 - r$, then apply Lemma 27. ◀

6 A simple branching algorithm

We now describe our first branching algorithm for vertex cover. As discussed earlier, the measure is a piecewise-linear function of k and μ. It is more convenient to describe it in terms of multiple self-contained "mini-algorithms," with *linear* measure functions. As a starting point, we can describe the first algorithm:

Algorithm 1 Function `Branch4Simple`(G, k).

```
1 Simplify $G$.
2 If $\mathrm{maxdeg}(G) \leq 3$, then run either the algorithm of Theorem 2 or the MaxIS-3
    algorithm, whichever is cheaper, and return.
3 Otherwise, apply Theorem 18 to an arbitrary vertex of degree at least 4, and run
    Branch4Simple on the two resulting subproblems.
```

To clarify, despite the name `Branch4Simple`, the graph is allowed to have vertices of either higher or lower degrees. What we mean is that, depending on the current values of k and μ, it is advantageous to branch on a vertex of degree 4 or higher. Our algorithm is *looking* for such a vertex; if it is not present (i.e. the graph has maximum degree 3) then an alternate algorithm should be used instead. Specifically, here we use either the AGVC algorithm with runtime depending on μ or the MaxIS-3 algorithm with runtime depending on $n = 2(k - \mu)$.

Henceforth we describe our branching algorithms more concisely. Whenever we apply a branching rule, we assume that we recursively use the algorithm under consideration (here, `Branch4Simple`) on the resulting subproblems and return. Likewise, when we list some other algorithms for the problem, we assume they are dovetailed (effectively running the cheapest of them).

▶ **Proposition 28.** `Branch4Simple` *has measure* $a\mu + bk$ *for* $a = 0.71808, b = 0.019442$.

Proof. Simplifying the graph in Line 1 can only reduce $\phi(G) = a\mu + bk$. If Theorem 18 is used, it gives a branch-seq B with drops dominated by $[(1, 3), (1, 5)]$ or $[(0.5, 1), (1.5, 4)]$. To satisfy Eq. (3), we thus need:

$$\mathrm{val}_{a,b}(B) \leq \max\{e^{-a-3b} + e^{-a-5b}, e^{-0.5a-b} + e^{-1.5a-4b}\} \leq 1 \tag{4}$$

Otherwise, suppose G has maximum degree 3. Since it is simplified, it has $n = 2\lambda = 2(k - \mu)$, so the MaxIS-3 algorithm has runtime $O^*(1.083506^{2(k-\mu)})$. Then, it suffices to show that

$$\min\{\mu \log(2.3146), 2(k - \mu) \log(1.083506)\} \leq a\mu + bk. \tag{5}$$

This can be verified mechanically. ◀

By Lemma 10, this combined with the MaxIS-4 algorithm (with $a = 0.71808, b = 0.019442, c = \log 1.137595$) immediately gives runtime $O^*(1.2152^k)$ for degree-4 graphs.

Although Proposition 28 is easy to check directly, the choices for the parameters a and b may seem mysterious. The explanation is that the most constraining part of the algorithm is dealing with the lower-degree graphs, i.e. solving the problem when the graph has maximum degree 3. The inequality (5), which governs this case, should be completely tight: there should be a "triple point" with respect to `Branch4Simple`, the MaxIS-3 algorithm, and the algorithm of Theorem 2. That is, for chosen parameters a, b, we should have

$$\mu \log(2.3146) = 2(k - \mu) \log(1.083506) = a\mu + bk$$

This allows us to determine b in terms of a. Then our goal is to minimize a whilst respecting the branching constraints of Eq. (4). This same reasoning will be used for parameters in all our algorithms. For example, we can define algorithms for branching on degree 5 and degree 6 vertices.

Algorithm 2 Function `Branch5Simple`(G, k).

```
1 Simplify $G$
2 If maxdeg$(G) \leq 4$, then run either the MaxIS-4 algorithm or Branch4Simple
3 Otherwise, apply Theorem 18 to an arbitrary vertex of degree at least 5.
```

Algorithm 3 Function `Branch6Simple`(G, k).

```
1 Simplify $G$
2 If maxdeg$(G) \leq 5$, then run either the MaxIS-5 algorithm or Branch5Simple
3 Otherwise, apply Theorem 18 to an arbitrary vertex of degree at least 6.
```

Along similar lines, we immediately obtain the results:

▶ **Proposition 29.** `Branch5Simple` *has measure* $a\mu + bk$ *for* $a = 0.44849, b = 0.085297$.
`Branch6Simple` *has measure* $a\mu + bk$ *for* $a = 0.20199, b = 0.160637$.

Combined with the MaxIS-5 algorithm using Lemma 10, `Branch5Simple` immediately gives an algorithm for degree-5 graphs with runtime $O^*(1.2491^k)$. The final algorithm is very similar, but we only keep track of runtime in terms of k, not μ or n.

Algorithm 4 Function `Branch7Simple`(G, k).

```
1 If maxdeg$(G) \leq 6$, then use Lemma 10 with algorithms of Branch6Simple and
   MaxIS-6
2 Otherwise, split on an arbitrary vertex of degree at least 7.
```

▶ **Theorem 30.** *Algorithm* `Branch7Simple`(G, k) *runs in time* $O^*(1.2575^k)$.

Proof. If we split on a vertex of degree at least 7, then we generate subproblems with vertex covers of size at most $k-1$ and $k-7$. These have cost 1.2575^{k-1} and 1.2575^{k-7} by induction. Since $1.2575^{-1} + 1.2575^{-7} < 1$, the overall cost is $O^*(1.2575^k)$. Otherwise the two algorithms in question have measure $a\mu + bk$ and cn respectively; where $a = 0.20199, b = 0.160637, c = \log(1.1893)$; by applying Lemma 10, we get a combined algorithm with cost $O^*(1.2575^k)$ as desired. ◀

References

1 J. Chen, I.A. Kanj, and G. Xia. Improved upper bounds for vertex cover. *Theoretical Computer Science*, 411(40-42):3736–3756, 2010.

2 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.

3 R.G. Downey and M.R. Fellows. Parameterized complexity. *Springer*, 1999.

4 J. Flum and M. Grohe. Parameterized complexity theory. *Springer*, 2006.

5 F. V. Fomin and D. Kratsch. *Exact Exponential Algorithms*. Springer-Verlag Berlin Heidelberg, 2010.

6 M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H.Freeman and Company, 1979.

7 Shivam Garg and Geevarghese Philip. Raising the bar for vertex cover: Fixed-parameter tractability above a higher guarantee. In *Proc. 26th annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1152–1166, 2016.

8 Leon Kellerhals, Tomohiro Koana, and Pascal Kunz. Vertex cover and feedback vertex set above and below structural guarantees. In *Proc. 17th International Symposium on Parameterized and Exact Computation (IPEC)*, 2022.

9 D. Lokshtanov, N. S. Narayanaswamy, V. Raman, M. S. Ramanujan, and S. Saurabh. Faster parameterized algorithms using linear programming. *ACM Trans. Algorithms*, 11(2):15:1–15:31, 2014.

10 M. Mahajan and V. Raman. Parameterizing above guaranteed values: MaxSat and MaxCut. *Journal of Algorithms*, 31:335–354, 1999.

11 M. Mahajan, V. Raman, and S. Sikdar. Parameterizing above or below guaranteed values. *Journal of Computer and System Sciences*, 75(2):137–153, 2009.

12 G. L. Nemhauser and L. E. Trotter. Properties of vertex packing and independence system polyhedra. *Mathematical Programming*, 6(1):48–61, 1974.

13 G. L. Nemhauser and L. E. Trotter. Vertex packings: Structural properties and algorithms. *Mathematical Programming*, 8(1):232–248, 1975.

14 R. Niedermeier. Invitation to fixed-parameter algorithms. *Oxford University Press*, 2006.

15 Dekel Tsur. Above guarantee parameterization for vertex cover on graphs with maximum degree 4. *Journal of Combinatorial Optimization*, 45(1):34, 2023.

16 Mingyu Xiao and Hiorshi Nagamochi. A refined algorithm for maximum independent set in degree-4 graphs. *Journal of Combinatorial Optimization*, 34(3):830–873, 2017.

17 Mingyu Xiao and Hiroshi Nagamochi. Confining sets and avoiding bottleneck cases: A simple maximum independent set algorithm in degree-3 graphs. *Theoretical Computer Science*, 469:92–104, 2013.

18 Mingyu Xiao and Hiroshi Nagamochi. An exact algorithm for maximum independent set in degree-5 graphs. *Discrete Applied Mathematics*, 199:137–155, 2016.

19 Mingyu Xiao and Hiroshi Nagamochi. Exact algorithms for maximum independent set. *Inf. Comput.*, 255:126–146, 2017. doi:10.1016/j.ic.2017.06.001.

A Proofs for basic results

Proof of Proposition 4. Consider any basic solution $\theta \in \{0, \frac{1}{2}, 1\}^V$ to LPVC(G). Let $I_0 = \theta^{-1}(0)$ and $I_1 = \theta^{-1}(1)$; note that I_0 is an indset (possibly empty) and $N(I_0) \subseteq I_1$. So $\sum_v \theta(v) = |I_1| + \frac{1}{2}(n - |I_0| - |I_1|) \geq (n + |N(I_0)| - |I_0|)/2 = (n + \mathrm{surp}(I_0))/2 \geq (n + \mathrm{minsurp}^-(G))/2$. On the other hand, if I_0 is a min-set of G, then setting $\theta(v) = 1$ for $v \in N(I_0)$ and $\theta(v) = 0$ for $v \in I_0$ and $\theta(v) = 1/2$ for all other vertices v gives a fractional vertex cover with $\sum_v \theta(v) = (n + \mathrm{minsurp}^-(G))/2$. ◀

Proof of Lemma 5. Let I be a critical-set; we first claim there is a good cover C with $I \subseteq C$ or $I \cap C = \emptyset$. For, suppose that $I \cap C = J$ where $\emptyset \neq J \subsetneq I$. In this case, we must also have $N(I \setminus J) \subseteq C$, so overall $|C \cap N[I]| \geq |J| + |N(I \setminus J)| = |J| + |I \setminus J| + \mathrm{surp}(I \setminus J)$; since $I \setminus J$ is a non-empty subset of I, we have $\mathrm{surp}(I \setminus J) \geq \mathrm{surp}(I)$, hence this is at least $|J| + |I \setminus J| + \mathrm{surp}(I) = |I| + \mathrm{surp}(I) \geq N(I)$. Thus, replacing $C \cap N[I]$ by $N(I)$ would give a vertex cover C' with $C' \cap I = \emptyset$ and $|C'| \leq |C|$.

Furthermore, if $\mathrm{surp}(I) \leq 0$, then consider a good cover C with $I \subseteq C$. Then replacing I with $N(I)$ would give another cover C' with $|C'| \leq |C| + |N(I)| - |I| \leq |C|$ and $C' \cap I = \emptyset$.

Finally, suppose $\mathrm{surp}(I) = 1$ and let $J = N(I)$. If G has a good cover with $I \cap C = \emptyset$, then $J \subseteq C$ as desired. Otherwise, suppose G has a good cover with $I \subseteq C$. Then, $C' = (C \setminus I) \cup J$ would be another cover of size $|C| - |I| + (|J| - |C \cap J|)$; since $|J| = |I| + 1$, this is $|C| + 1 - |C \cap J|$. If $C \cap J \neq \emptyset$, this would be a good cover with $I \cap C = \emptyset$. ◀

Proof of Proposition 6. Let J be an indset with $\mathrm{surp}_{G-N[I]}(J) = \mathrm{minsurp}^-(G - N[I])$. Then $J \cup I$ is a non-empty indset of G with surplus $\mathrm{surp}_{G-N[I]}(J) + \mathrm{surp}_G(I) = \mathrm{minsurp}^-(G - N[I]) + \mathrm{surp}_G(I)$; in particular, if $\mathrm{minsurp}^-(G - N[I]) + \mathrm{surp}_G(I) = \mathrm{minsurp}(G)$, then I is a min-set of G. On ther other hand, if I is contained in a min-set J of G, then $\mathrm{surp}_{G-N[I]}(J \setminus I) = \mathrm{surp}_G(J) + |I| - |N(I)| = \mathrm{surp}_G(J) - \mathrm{surp}_G(I)$. So $\mathrm{minsurp}^-(G - N[I]) \leq \mathrm{minsurp}(G) - \mathrm{surp}_G(I)$. ◀

Proof of Proposition 7. By applying Proposition 6 to indsets $I = \{x\}$, we get $\mathrm{minsurp}(G) = \min_{x \in V} \big(\mathrm{minsurp}^-(G - N[x]) + \deg(x) - 1\big)$. We can use Proposition 4 to compute each value $\mathrm{minsurp}^-(G - N[x])$ in polynomial time (by solving the LP). This gives an algorithm to compute $\mathrm{minsurp}(G)$. Likewise, given any vertex-set X, we can use Proposition 6 to determine if X is contained in a min-set of G and, if so, we can use Proposition 4 to find a vertex set J attaining $\mathrm{surp}_{G-N[X]}(J) = \mathrm{minsurp}^-(G - N[X])$; then $J \cup X$ is a min-set of G containing X. ◀

Proof of Proposition 8. Let us first consider P1. If G' has a good cover C', then $C' \cup N(I)$ is a cover for G of size k. Conversely, by Lemma 5, there is a good cover C of G with $C \cap I = \emptyset$, and then then $C \setminus N(I)$ is a cover of G' of size $k - |N(I)| = k'$. Similarly, if G' has a fractional cover θ, then we can extend to G by setting $\theta(v) = 0$ for $v \in I$ and $\theta(v) = 1$ for $v \in N(I)$, with total weight $\lambda(G') + (k - k')$. So $\mu(G') = k' - \lambda(G') \leq k' - (\lambda(G) - (k - k')) = k - \lambda(G) = \mu(G)$.

Next, for rule P2, let $J = N(I)$. Since I is a critical-set with surplus one, it cannot contain any isolated vertex. Given any good cover C of G with $J \subseteq C$, observe that $(C \setminus J) \cup \{y\}$ is a cover of G' of size $k - |J| + 1 = k'$. Likewise, given a good cover C of G with $J \cap C = \emptyset$, we have $N(J) \subseteq C$; in particular, $I \subseteq C$ since I has no isolated vertices, so $C \setminus I$ is a cover of G' of size $k - |I| = k'$. Conversely, consider a good cover C' of G'; if $y \in C'$, then $(C' \cup J) \setminus \{y\}$ is a cover of G, of size $k' + |J| - 1 = k$. If $y \notin C'$, then $C' \cup I$ is a cover of G, of size $k' + |I| = k$. Similarly, if G has an optimal fractional cover θ, it can be extended to G by setting $\theta(v) = \theta(y)$ for $v \in J$, and $\theta(v) = 1 - \theta(y)$ for $v \in I$. This has weight $\lambda(G') + |I|(1 - \theta(y)) + |J|\theta(y) - \theta(y) = \lambda(G') + |I| \geq \lambda(G)$. So $\mu(G') = k' - \lambda(G') \leq (k - |I|) - (\lambda(G) - |I|) = \mu(G)$.

Now for P3, let $A = N_G(u) \cap N_G(x)$ and $B_x = N_G(x) \setminus N_G[u]$ and $B_u = N_G(u) \setminus N_G[x]$. To show the validity of the preprocessing rule, suppose $\langle G, k \rangle$ is feasible, and let C be a good cover of G with either $|C \cap \{u, x\}| = 1$; then $C \setminus (\{u, x\} \cup A)$ is a cover of G' of size k'. Conversely, suppose $\langle G', k' \rangle$ is feasible with a good cover C'. So $|B_u \setminus C'| \leq 1$. if $B_u \setminus C' = \emptyset$, then $C = C' \cup A \cup \{x\}$ is a cover of G. If $|B_u \setminus C'| = 1$, then necessarily $B_x \subseteq C'$ and $C = C' \cup A \cup \{u\}$ is a cover of G.

Finally, we claim that (P3) satisfies $\mu(G') \leq \mu(G)$; since $k' = k - 1 - |A|$, it suffices to show that $\lambda(G) \leq \lambda(G') + 1 + |A|$. For, take an optimal fractional cover θ for G'. To extend it to a fractional cover for G, we set $\theta(v) = 1$ for $v \in A$ and there are a few cases to determine the values for $\theta(u)$ and $\theta(v)$. If $B_u = \emptyset$, we set $\theta(u) = 0, \theta(x) = 1$. Otherwise, if $B_u \neq \emptyset$, let $b_u = \min_{v \in B_u} \theta'(v)$ and then set $\theta(u) = 1 - b_u, \theta(x) = b_u$; since G' has a biclique between B_u and B_x, this covers any edge between x and $v \in B_x$ with $\theta(x) + \theta(v) = b_u + \theta(v) \geq 1$. Overall, θ is a fractional vertex cover of G of weight $\sum_{v \in G'} \theta(v) + \theta(u) + \theta(x) + |A| = \lambda(G') + 1 + \text{codeg}(u, x)$. ◀

Proof of Lemma 9. Consider a kite u, x, y, z. We can view the application of (P3) to funnel u as a two-part process. First, we remove the shared neighbor $y \in N(u) \cap N(x)$; then, we merge vertices z, x into a new vertex u'. In the first step, we obtain a graph $G'' = G - y$ with $k'' = k - 1$ and $n'' = n - 1$. So, by Proposition 4, we have $\lambda(G'') = \lambda(G) - 1/2$ and hence $\mu(G'') \leq \mu(G) - 1/2$ and $k'' = k - 1$. By Proposition 8, the second step gives $\mu(G') \leq \mu(G'')$ and $k' = k'' - 1$. ◀

Proof of Lemma 10. First, exhaustively apply rule (P1) to G; the resulting graph G' has $n' \leq n$ and $k' \leq k$, and $\text{minsurp}(G') \geq 0$. In particular, $\mu(G') = k' - \lambda(G') = k' - n'/2$. Note that $G' \in \mathcal{G}$ since (P1) just deletes vertices. Now dovetail the two algorithms for G', running both simultaneously and returning the output of whichever terminates first. This has runtime $O^*(\min\{e^{cn'}, e^{a(k'-n'/2)+bk'}\})$. For fixed k', this is maximized at $n' = \frac{2k'(a+b)}{a+2c}$, at which point it takes on value $e^{dk'} \leq e^{dk}$. ◀