Raising The Bar For Vertex Cover: Fixed-parameter Tractability Above A Higher Guarantee*

Shivam Garg† Geevarghese Philip‡

Abstract

The standard parameterization of the Vertex Cover problem (Given an undirected graph G and $k \in \mathbb{N}$ as input, does G have a vertex cover of size at most k?) has the solution size k as the parameter. The following more challenging parameterization of Vertex Cover stems from the observation that the size MM of a *maximum matching* of G lower-bounds the size of any vertex cover of G: Does G have a vertex cover of size at most $MM + k_\mu$? The parameter is the excess k_μ of the solution size over the lower bound MM.

Razgon and O'Sullivan (ICALP 2008) showed that this *above-guarantee* parameterization of Vertex Cover is fixed-parameter tractable and can be solved in time $\mathcal{O}^\star(15^{k_\mu})$, where the $\mathcal{O}^\star$ notation hides polynomial factors. This was first improved to $\mathcal{O}^\star(9^{k_\mu})$ (Raman et al., ESA 2011), then to $\mathcal{O}^\star(4^{k_\mu})$ (Cygan et al., IPEC 2011, TOCT 2013), then to $\mathcal{O}^\star(2.618^{k_\mu})$ (Narayanaswamy et al., STACS 2012) and finally to the current best bound $\mathcal{O}^\star(2.3146^{k_\mu})$ (Lokshtanov et al., TALG 2014). The last two bounds were in fact proven for a different parameter: namely, the excess k_λ of the solution size over LP, the value of the *linear programming relaxation* of the standard LP formulation of Vertex Cover. Since $LP \geq MM$ for any graph, we have that $k_\lambda \leq k_\mu$ for Yes instances. This is thus a *stricter* parameterization—the new parameter is, in general, smaller—and the running times carry over directly to the parameter k_μ.

We investigate an even stricter parameterization of Vertex Cover, namely the excess $\hat{k}$ of the solution size over the quantity $(2LP - MM)$. We ask: Given a graph G and $\hat{k} \in \mathbb{N}$ as input, does G have a vertex cover of size at most $(2LP - MM) + \hat{k}$? The parameter is $\hat{k}$. It can be shown that $(2LP - MM)$ is a lower bound on vertex cover size, and since $LP \geq MM$ we have that $(2LP - MM) \geq LP$, and hence that $\hat{k} \leq k_\lambda$ holds for Yes instances. Further, $(k_\lambda - \hat{k})$ could be as large as $(LP - MM)$ and—to the best of our knowledge—this difference cannot be expressed as a function of k_λ alone. These facts motivate and justify our choice of parameter: this is indeed a stricter parameterization whose tractability does not follow directly from known results.

We show that Vertex Cover is fixed-parameter tractable for this stricter parameter $\hat{k}$: We derive an algorithm which solves Vertex Cover in time $\mathcal{O}^\star(3^{\hat{k}})$, thus pushing the envelope further on the parameterized tractability of Vertex Cover.

1 Introduction

The input to the Vertex Cover problem consists of an undirected graph G and an integer k, and the question is whether G has a *vertex cover*—a subset $S \subseteq V(G)$ of vertices such that every edge in G has at least one endpoint in S—of size at most k. This problem is among Karp's original list of 21 NP-complete problems [14]; it is also one of the best-studied problems in the field of parameterized algorithms and complexity [5, 15, 18, 24].

The input to a parameterized version of a classical decision problem consists of two parts: the classical input and a specified *parameter*, usually an integer, usually denoted by the letter k. A fixed-parameter tractable (FPT) algorithm [8, 10] for this parameterized problem is one which solves the underlying decision problem in time $f(k) \cdot n^c$ where (i) f is a computable function of k alone, (ii) n is the size of the (classical) input instance, and (iii) c is a constant independent of n and k. This running time is often written as $\mathcal{O}^\star(f(k))$; the $\mathcal{O}^\star$ notation hides constant-degree polynomial factors in the running time. A parameterized problem which has an FPT algorithm is itself said to be (in) FPT.

The "standard" parameter for Vertex Cover is the number k which comes as part of the input and represents the *size of the vertex cover* for which we look—hence referred to, loosely, as the "solution size". This is the most extensively studied parameterization of Vertex Cover [1, 4, 5, 15, 22]. Starting with a simple two-way branching algorithm (folklore) which solves the problem in time $\mathcal{O}^\star(2^k)$ and serves as the *de facto* introduction-cum-elevator-pitch to the field, a number of FPT algorithms with improved running times have been found for Vertex Cover, the current fastest of which solves the problem in $\mathcal{O}^\star(1.2738^k)$ time [5]. It is also known that unless the Exponential Time Hypothesis (ETH) fails, there is no algorithm which solves Vertex Cover in $\mathcal{O}^\star(2^{o(k)})$ time [13].

This last point hints at a fundamental drawback of this parameterization of Vertex Cover: In cases where the size of a smallest vertex cover (called the *vertex cover number*) of the input graph is "large"—say, $\Omega(n)$ where n

*Work done at the Max Planck Institute for Informatics, Saarbrücken, Germany, and supported by the Indo-German Max Planck Center for Computer Science (IMPECS).

†Indian Institute of Technology Bombay, Mumbai, India, shivamgarg@iitb.ac.in

‡Chennai Mathematical Institute, Chennai, India, gphilip@cmi.ac.in

is the number of vertices in the graph—one cannot hope to use these algorithms to find a smallest vertex cover "fast". This is, for instance, the case when the input graph G has a large *matching*, which is a set of edges of G no two of which share an end-point. Observe that since each edge in a matching has a distinct representative in any vertex cover, the size of a largest matching in G is a *lower bound* on the vertex cover number of G. So when input graphs have matchings of size $\Omega(n)$—which is, in a real sense, *the most common case* by far (see, e.g.; [2, Theorem 7.14])—FPT algorithms of the kind described in the previous paragraph take $\Omega(c^n)$ time for some constant c, and one cannot hope to improve this to the form $\mathcal{O}(c^{o(n)})$ unless ETH fails. Put differently: consider the standard parameterization of VERTEX COVER, and let MM denote the size of a maximum matching—the *matching number*—of the input graph. Note that we can find MM in polynomial time [9]. When $k < MM$ the answer is trivially NO, and thus such instances are uninteresting, and when $k \geq MM$, FPT algorithms for this parameterization of the problem are impractical for those (most common) instances for which $MM = \Omega(n)$.

Such considerations led to the following alternative parameterization of VERTEX COVER, where the parameter is the "excess" above the matching number:

ABOVE-GUARANTEE VERTEX COVER (AGVC)	
Input:	A graph G and $k_\mu \in \mathbb{N}$.
Parameter:	k_μ
Question:	Does G have a vertex cover of size at most $MM + k_\mu$?

The parameterized complexity of AGVC was settled by Razgon and O'Sullivan [25] in 2008; they showed that the problem is FPT and can be solved in $\mathcal{O}^\star(15^{k_\mu})$ time. A sequence of faster FPT algorithms followed: In 2011, Raman et al. [24] improved the running time to $\mathcal{O}^\star(9^{k_\mu})$, and then Cygan et al. [6, 7] improved it further to $\mathcal{O}^\star(4^{k_\mu})$. In 2012, Narayanaswamy et al. [19] developed a faster algorithm which solved AGVC in time $\mathcal{O}^\star(2.618^{k_\mu})$. Lokshtanov et al. [16] improved on this to obtain an algorithm with a running time of $\mathcal{O}^\star(2.3146^{k_\mu})$. This is currently the fastest FPT algorithm for ABOVE-GUARANTEE VERTEX COVER.

The algorithms of Narayanaswamy et al. and Lokshtanov et al. in fact solve a "stricter" parameterization of VERTEX COVER. Let LP denote the minimum value of a solution to the *linear programming relaxation* of the standard LP formulation of VERTEX COVER (See Section 2 for definitions.). Then LP is a lower bound on the vertex cover number of the graph. Narayanaswamy et al. introduced the following parameterization[1] of VERTEX COVER, "above" LP:

VERTEX COVER ABOVE LP (VCAL)	
Input:	A graph G and $k_\lambda \in \mathbb{N}$.
Parameter:	k_λ
Question:	Does G have a vertex cover of size at most $LP + k_\lambda$?

The two algorithms solve VCAL in times $\mathcal{O}^\star(2.618^{k_\lambda})$ and $\mathcal{O}^\star(2.3146^{k_\lambda})$, respectively. Since the inequality $LP \geq MM$ holds for every graph we get that VCAL is a *stricter* parameterization of VERTEX COVER, in the sense that these algorithms for VCAL directly imply algorithms which solve AGVC in times $\mathcal{O}^\star(2.618^{k_\mu})$ and $\mathcal{O}^\star(2.3146^{k_\mu})$, respectively. To see this, consider an instance (G, k_μ) of AGVC where MM is the matching number and VC_{opt} is the (unknown) vertex cover number of the input graph G, and let $x = MM + k_\mu$. The question is then whether $VC_{opt} \leq x$. To resolve this, find the value LP for the graph G (in polynomial time) and set $k_\lambda = x - LP$. Now we have that $VC_{opt} \leq x \iff VC_{opt} \leq LP + k_\lambda$, and we can check if the latter inequality holds—that is, we can solve VCAL—in $\mathcal{O}^\star(2.3146^{k_\lambda})$ time using the algorithm of Lokshtanov et al. Now $LP \geq MM \implies (x - LP) \leq (x - MM) \implies k_\lambda \leq k_\mu$, and so this algorithm runs in $\mathcal{O}^\star(2.618^{k_\mu})$ time as well.

This leads us naturally to the next question: can we push this further? Is there an even stricter lower bound for VERTEX COVER, such that VERTEX COVER is still fixed-parameter tractable when parameterized above this bound? To start with, it is not clear that a stricter lower bound even exists for VERTEX COVER: what could such a bound possibly look like? It turns out that we can indeed derive such a bound; we are then left with the task of resolving tractability above this stricter bound.

Note that while the idea of parameterizing above LP and related values is somewhat new, it has already yielded a number of significant results including faster FPT algorithms for NODE MULTIWAY CUT [6, 7] and ODD CYCLE TRANSVERSAL [16, 19]. An exciting new development in this direction is the recent work of Wahlström [26] who explored the more general notion of the so-called k-submodular relaxations of valued CSPs to obtain a range of new and improved FPT algorithms, including faster FPT algorithms for UNIQUE LABEL COVER and GROUP FEEDBACK VERTEX SET.

Our Problem. Motivated by an observation of Lovász and Plummer [17] we show—see Lemma 2.1—that the quantity $(2LP - MM)$ is a lower bound on the vertex

[1] To be precise, Narayanaswamy et al. used the value $\lceil LP \rceil$ instead of just LP in their definition of VERTEX COVER ABOVE LP, but this makes no essential difference.

cover number of a graph, and since $LP \geq MM$ we get that $(2LP - MM) \geq LP$. This suggests the following parameterization of Vertex Cover:

Vertex Cover Above Lovász-Plummer (VCAL-P)	
Input:	A graph G and $\hat{k} \in \mathbb{N}$.
Parameter:	$\hat{k}$
Question:	Does G have a vertex cover of size at most $(2LP - MM) + \hat{k}$?

Since $(2LP - MM) \geq LP$ we get, following similar arguments as described above, that Vertex Cover Above Lovász-Plummer is a stricter parameterization than Vertex Cover Above LP. The difference between these lower bounds can be arbitrarily large; for instance, consider a graph G which is a disjoint union of t triangles. Then $MM = t$, $LP = 1.5t$, and $2LP - MM = 2t$. Further, $(k_\lambda - \hat{k})$ could be as large as $(LP - MM)$ and—to the best of our knowledge—this difference cannot be expressed, in general, as a function of k_λ alone for the purpose of solving Vertex Cover. These facts justify our choice of parameter: Vertex Cover Above Lovász-Plummer is indeed a stricter parameterization than both Above-Guarantee Vertex Cover and Vertex Cover Above LP, and its tractability does not follow directly from known results.

Our Results. The main result of this work is that Vertex Cover is fixed-parameter tractable even when parameterized above this stricter lower bound:

Theorem 1.1. Vertex Cover Above Lovász-Plummer *is fixed-parameter tractable and can be solved in* $\mathcal{O}^\star(3^{\hat{k}})$ *time.*

By the discussions above, this directly implies similar FPT algorithms for the two weaker parameterizations:

Corollary 1.1. Vertex Cover Above LP *can be solved in* $\mathcal{O}^\star(3^{k_\lambda})$ *time, and* Above-Guarantee Vertex Cover *can be solved in* $\mathcal{O}^\star(3^{k_\mu})$ *time.*

Our Methods. We now sketch the main ideas behind our FPT algorithm for Vertex Cover Above Lovász-Plummer; the details are in Section 3. Let $k = (2LP - MM) + \hat{k}$ denote the "budget", which is the maximum size of a vertex cover in whose existence we are interested; we want to find if there is a vertex cover of size at most k. At its core, our algorithm is a simple branching algorithm which (i) manages to drop the *measure* $\hat{k}$ by at least 1 on each branch, and (ii) has a worst-case branching factor of 3. To achieve this kind of branching we need to find, in polynomial time, constant-sized structures in the graph, on which we can branch profitably. It turns out that none of the "local" branching strategies traditionally used to attack Vertex Cover—from the simplest "pick an edge and branch on its end-points" to more involved ones based on complicated structures (e.g., those of Chen et al. [5])—serve our purpose. All of these branching strategies give us a drop in k in each branch—because we pick one or more vertices into the solution—but give us *no control* over how LP and MM change.

So we turn to the more recent ideas of Narayanaswamy et al. [19] who solve a similar problem: they find a way to *preprocess* the input graph using reduction rules in such a way that branching on small structures in the resulting graph would make *their measure* $(k - LP)$ drop on each branch. We find that their reduction rules do not increase our measure, and hence that we can safely apply these rules to obtain a graph where we can pick up to two vertices in each branch and still have control over how LP changes. The *branching rules* of Narayanaswamy et al., however, are not of use to us: these rules help control the drop in LP, but they provide no control over how MM changes. Note that for our measure $\hat{k}$ to drop we need, roughly speaking, (i) a good drop in k, (ii) a small drop in LP, and (iii) a *good drop* in MM. None of the branching strategies of Narayanaswamy et al. or Lokshtanov et al. (or others in the literature which we tried) help with this.

To get past this point we look at the classical *Gallai-Edmonds decomposition* of the reduced graph, which can be computed in polynomial time [9,11,12,17]. We prove that by carefully choosing edges to branch based on this decomposition, we can ensure that both LP and MM change in a way which gives us a net drop in the measure $\hat{k}$. The key ingredient and the most novel aspect of our algorithm—as compared to existing algorithms for Vertex Cover—is the way in which we exploit the Gallai-Edmonds decomposition to find small structures—edges and vertices—on which we can branch profitably. While this part is almost trivial to implement, most of the technical effort in the paper has gone into proving that our choices are correct. See Algorithm 1 on the next page for an outline which highlights the new parts, and Algorithm 2 on page 8 for the complete algorithm.

2 Preliminaries

We use $\uplus$ to denote the disjoint union of sets. All our graphs are undirected and simple. $V(G)$ and $E(G)$ denote, respectively, the vertex and edge sets of a graph G. $G[X]$ is the subgraph of G *induced* by a vertex subset $X \subseteq V(G)$: $G[X] = (X, F)$; $F = \{\{v, w\} \in E(G) \; ; \; v, w \in X\}$. $MM(G)$ is the matching number of graph G, and $OPT(G)$ is the vertex cover number of G. A matching M in graph G *saturates* each vertex which is an end-point of an edge in M, and *exposes* every other vertex in G. M is a *perfect matching* if it

Algorithm 1 An outline of the algorithm for VERTEX COVER ABOVE LOVÁSZ-PLUMMER.

function VCAL-P($(G, \hat{k})$)
  Exhaustively apply the three reduction rules of Narayanaswamy et al. to $(G, \hat{k})$.
  Let $(G, \hat{k})$ denote the resulting graph on which no rule applies.
  if $(G, \hat{k})$ is a trivial instance **then**
    return `True` or `False` as appropriate.
  Compute the Gallai-Edmonds decomposition $V(G) = O \uplus I \uplus P$ of G.
  if $G[I \cup P]$ contains at least one edge $\{u, v\}$ **then**
    Branch on the edge $\{u, v\}$. $\hat{k}$ drops by 1 on each branch.
  else ▷ Now $P = \emptyset$.
    Branch on a vertex $u \in O = V(G)$ and two of its neighbours $v, w \in O$:
      When we pick both of v, w into the solution in one branch, $\hat{k}$ drops by 1.
      When we pick u into the solution in the other branch, $\hat{k}$ *may not* drop. We find a suitable edge in $G' = (G \setminus u)$ and branch on its end-points to make $\hat{k}$ drop.

saturates all of $V(G)$. M is a *near-perfect matching* if it saturates all but one vertex of $V(G)$. Graph G is said to be *factor-critical* if for each $v \in V(G)$ the induced subgraph $G[(V(G) \setminus \{v\})]$ has a perfect matching.

For $X \subseteq V(G)$, $N(X)$ is the set of neighbours of X which are not in X: $N(X) = \{v \in (V(G) \setminus X) \ ; \ \exists w \in X \ : \ \{v, w\} \in E(G)\}$. $X \subseteq V(G)$ is an *independent set* in graph G if no edge in G has both its end-points in X. The *surplus* of an independent set $X \subseteq V(G)$ is $\textbf{surplus}(X) = (|N(X)| - |X|)$. The *surplus of a graph G*, $\textbf{surplus}(G)$, is the minimum surplus over all independent sets in G. Graph G is a *bipartite graph* if $V(G)$ can be partitioned as $V(G) = X \uplus Y$ such that every edge in G has exactly one end point in each of the sets X, Y. Hall's Theorem tells us that a bipartite graph $G = ((X \uplus Y), E)$ contains a matching which saturates all vertices of the set X if and only if $\forall S \subseteq X \ : \ |N(S)| \geq |S|$. König's Theorem tells us that for a bipartite graph G, $OPT(G) = MM(G)$.

The *linear programming (LP) relaxation* of the standard LP formulation for VERTEX COVER for a graph G (the *relaxed* VERTEX COVER *LP for G* for short), denoted $LPVC(G)$, is:

$$\begin{aligned} \text{minimize} & \sum_{v \in V(G)} x_v \\ \text{subject to } & x_u + x_v \geq 1 \ , \{u, v\} \in E(G) \\ & 0 \leq x_v \leq 1 \quad , v \in V(G) \end{aligned}$$

A *feasible solution* to this LP is an assignment of values to the variables $x_v \ ; \ v \in V(G)$ which satisfies all the conditions in the LP, and an *optimum solution* is a feasible solution which minimizes the value of the objective function $\sum_{v \in V(G)} x_v$. We use $w(x)$ to denote the value (of the objective function) of a feasible solution x to $LPVC(G)$, and $LP(G)$ to denote the value of an optimum solution to $LPVC(G)$. $OPT(G)$ and $MM(G)$ are then the values of optimum solutions to the *integer* programs corresponding to $LPVC(G)$ and to its LP *dual*, respectively [3]. It follows that for any graph G, $MM(G) \leq LP(G) \leq OPT(G)$. Our stronger lower bound for $OPT(G)$ is motivated by a similar bound due to Lovász and Plummer [17, Theorem 6.3.3]:

LEMMA 2.1. *For any graph G, $OPT(G) \geq (2LP(G) - MM(G))$.*

Proof. Let S be a smallest vertex cover of graph G, and let $H = ((S \uplus (V(G) \setminus S)), F)$ be the bipartite subgraph of G with $F = \{u, v\} \in E(G) \ ; \ u \in S, v \in (V(G) \setminus S)$. That is, the vertex set of H is $V(G)$ with the bipartition $(S, (V(G) \setminus S))$, and the edge set of H consists of exactly those edges of G which have one end-point in S and the other in $(V(G) \setminus S)$. Let T be a smallest vertex cover of graph H. Then $|T| = OPT(H) = MM(H)$, where the second equality follows from König's Theorem. Consider the following assignment y of values to variables $y_v \ ; \ v \in V(G)$:

$$y_v = \begin{cases} 1 & \text{if } v \in (S \cap T) \\ \frac{1}{2} & \text{if } v \in ((S \cup T) \setminus (S \cap T)) \\ 0 & \text{otherwise.} \end{cases}$$

Observe that $0 \leq y_v \leq 1$ for each $v \in V(G)$. We claim that y is a feasible solution to $LPVC(G)$. Indeed, since S is a vertex cover of G, every edge $\{u, v\} \in E(G)$ must have at least one end-point in S. If $\{u, v\} \cap (S \cap T) \neq \emptyset$ then y assigns the value 1 to at least one of y_u, y_v, and so we have that $y_u + y_v \geq 1$. Otherwise, if $\{u, v\} \subseteq (S \setminus (S \cap T))$ then y assigns the value $\frac{1}{2}$ to both of y_u, y_v, and so we have that $y_u + y_v = 1$. In the only remaining case, exactly one of $\{u, v\}$ is in S, and the other vertex is in $(V(G) \setminus S)$. Without loss of generality, suppose $\{u,v\} \cap S = \{u\}, v \in (V(G) \setminus S)$. Then $\{u,v\} \in E(H)$ and $u \notin T$, hence we get—since T is a vertex cover of H—that $v \in T$. Thus $\{u,v\} \subseteq ((S \cup T) \setminus (S \cap T))$, and so y assigns the value $\frac{1}{2}$ to both of y_u, y_v and we have that $y_u + y_v = 1$. Thus the assignment y satisfies all the conditions in the $LPVC(G)$, and hence is a feasible solution to $LPVC(G)$. Thus $w(y) \geq LP(G)$.

Observe now that

$$\begin{aligned} w(y) &= \frac{|S| + |T|}{2} \\ &= \frac{OPT(G) + OPT(H)}{2} \\ &= \frac{OPT(G) + MM(H)}{2} \\ &\leq \frac{OPT(G) + MM(G)}{2}, \end{aligned}$$

where the first equality follows from the way we defined y, and the inequality follows from the observation that the matching number of the subgraph H of G cannot be *larger* than that of G itself. Putting these together we get that $LP(G) \leq w(y) \leq \frac{OPT(G)+MM(G)}{2}$, which in turn gives the bound in the lemma. □

For any graph G there exists an *optimum* solution to $LPVC(G)$ in which $x_v \in \{0, \frac{1}{2}, 1\}$; $v \in V(G)$ [20]. Such an optimum solution is called a *half-integral solution* to $LPVC(G)$, and we can find such a solution in polynomial time [21]. Whenever we refer to an optimum solution to $LPVC(G)$ in the rest of the paper, we mean a half-integral solution. Given a half-integral solution x to $LPVC(G)$, we define $V_i^x = \{v \in V(G)\ ;\ x_v = i\}$ for each $i \in \{0, \frac{1}{2}, 1\}$. For any optimal half-integral solution x we have that $N(V_0^x) = V_1^x$. Given a graph G as input we can, in polynomial time, compute an optimum half-integral solution x to $LPVC(G)$ such that for the induced subgraph $H = G[V_{1/2}^x]$, setting all variables to the value $\frac{1}{2}$ is the *unique* optimum solution to $LPVC(H)$ [21]. Graphs which satisfy the latter property must have positive surplus, and conversely:

LEMMA 2.2. **[23]** *For any graph G, all-$\frac{1}{2}$ is the unique optimum solution to $LPVC(G)$ if and only if* **surplus**$(G) > 0$.

In fact, the surplus of a graph G is a lower bound on the number of vertices that we can delete from G, and have a *guaranteed* drop of *exactly* $\frac{1}{2}$ *per deleted vertex* in $LP(G)$:

LEMMA 2.3. *Let $s \in \mathbb{N}$, and let G be a graph with* **surplus**$(G) \geq s$. *Then deleting any subset of s vertices from G results in a graph G' such that $LP(G') = LP(G) - \frac{s}{2}$.*

Proof. The proof is by induction on s. Let $n = |V(G)|$. The case $s = 0$ is trivially true. Suppose $s = 1$. Then by Lemma 2.2 all-$\frac{1}{2}$ is the unique optimum solution to $LPVC(G)$, and so $LP(G) = \frac{n}{2}$. Let $v \in V(G)$, and let $G' = G[V(G) \setminus \{v\}]$. Then $|V(G'| = (n-1)$. Since all-$\frac{1}{2}$ is a *feasible* solution to $LPVC(G')$, we get that $LP(G') \leq \frac{(n-1)}{2} = LP(G) - \frac{1}{2}$. If possible, let x' be an optimum solution for $LPVC(G')$ such that $w(x') < \frac{(n-1)}{2}$. From the half-integrality property of relaxed VERTEX COVER LP formulations we get that $w(x') \leq (\frac{n}{2} - 1)$. Now we can assign the value 1 to vertex v and the values x' to the remaining vertices of graph G, to get a solution x such that $w(x) = \frac{n}{2} = LP(G)$. Thus x is an *optimum* solution to $LPVC(G)$ which is *not* all-$\frac{1}{2}$, a contradiction. So we get that $LP(G') = LP(G) - \frac{1}{2}$, proving the case $s = 1$.

For the induction step, let $s \geq 2$. Then by Lemma 2.2 all-$\frac{1}{2}$ is the unique optimum solution to $LPVC(G)$, and so $LP(G) = \frac{n}{2}$. Let v be an arbitrary vertex in G, and let $G' = G[V(G) \setminus \{v\}]$. Since deleting a single vertex from G cannot cause the surplus of G to drop by more than 1, we get that **surplus**$(G') \geq (s-1) \geq 1$. So from Lemma 2.2 we get that $LP(G') = \frac{(n-1)}{2}$. Applying the induction hypothesis to G' and $s-1$, we get that deleting any subset of $s-1$ vertices from G' results in a graph G'' such that $LP(G'') = LP(G') - \frac{(s-1)}{2} = \frac{(n-1)}{2} - \frac{(s-1)}{2} = \frac{(n-s)}{2}$. This completes the induction step. □

There is a matching between the vertex sets which get the values 0 and 1 in an optimal half-integral solution to the LP.

LEMMA 2.4. *Let G be a graph, and let x be an optimal half-integral solution to $LPVC(G)$. Let $H = ((V_1^x \uplus V_0^x), F)$ be the bipartite subgraph of G where $F = \{u,v\} \in E(G)\ ;\ u \in V_1^x, v \in V_0^x$. Then there exists a (maximum) matching of H which saturates all of V_1^x.*

Proof. It is enough to show that for every $X \subseteq V_1^x$ the inequality $|N(X)| \geq |X|$ holds in the graph H, and the rest will follow by Hall's Theorem. So let there exist some $X \subseteq V_1^x$ such that $|N(X)| < |X|$ in H. Now consider a solution x^* to $LPVC(G)$ in which all the vertices of G have same values as in x, *except* that all vertices in $X \cup N(X)$ get the value $\frac{1}{2}$. It is not difficult to verify that x^* is a feasible solution to $LPVC(G)$. Now $w(x^*) = w(x) + \frac{|N(X)|-|X|}{2} < w(x)$, which is a contradiction since we assumed that x is a solution to $LPVC(G)$ with the minimum value. The lemma follows. □

We make critical use of the classical Gallai-Edmonds decomposition of graphs.

Definition 2.1. (Gallai-Edmonds decomposition) *The Gallai-Edmonds decomposition of a graph G is a partition of its vertex set $V(G)$ as $V(G) = O \uplus I \uplus P$ where:*

- $O = \{v \in V(G)$; *some* maximum *matching of G leaves v exposed* $\}$
- $I = N(O)$
- $P = V(G) \setminus (I \cup O)$

We now list a few of the many useful properties of Gallai-Edmonds decompositions.

Theorem 2.1. **[9, 11, 12, 17]** *The Gallai-Edmonds decomposition of a graph G is unique, and can be computed in polynomial time in the size of G. Let $V(G) = O \uplus I \uplus P$ be the Gallai-Edmonds decomposition of G. Then the following hold:*

1. *Every component of the induced subgraph $G[O]$ is factor-critical.*
2. *A matching M in graph G is a* maximum *matching of G if and only if:*
   (a) *For each connected component H of the induced subgraph $G[O]$, the edge set $M \cap E(H)$ forms a* near perfect *matching of H;*
   (b) *For each vertex $i \in I$ there exists some vertex $i \in O$ such that $\{i, o\} \in M$, and;*
   (c) *The edge set $M \cap E(G[P])$ forms a* perfect *matching of the induced subgraph $G[P]$.*
3. *In particular: Any maximum matching M of G is a disjoint union of (i) a perfect matching of $G[P]$, (ii) near-perfect matchings of each component of $G[O]$, and (iii) an edge from each vertex in I to a distinct component of $G[O]$.*
4. *The Stability Lemma: For a vertex $v \in V(G)$ let $G - v = G[V(G) \setminus v]$. Let $O(H), I(H), P(H)$ denote the three parts in the Gallai-Edmonds decomposition of a graph H.*
   - *Let $v \in O$. Then $O(G - v) \subseteq (O \setminus \{v\})$, $I(G - v) \subseteq I$, and $P(G - v) \supseteq P$.*
   - *Let $v \in I$. Then $O(G - v) = O$, $I(G - v) = (I \setminus \{v\})$, and $P(G - v) = P$.*
   - *Let $v \in P$. Then $O(G - v) \supseteq O$, $I(G - v) \supseteq I$, and $P(G - v) \subseteq (P \setminus \{v\})$.*

Corollary 2.1. *Let $V(G) = O \uplus I \uplus P$ be the Gallai-Edmonds decomposition of graph G. Then the following hold:*

1. *If $I \cup P$ is an independent set in G, then $P = \emptyset$.*
2. *Let $v, w \in O$ be two vertices which are part of the same connected component of the induced subgraph $G[O]$, and let $G' = G[(V(G) \setminus \{v, w\})]$. Then $MM(G') \leq (MM(G) - 1)$.*

Proof. We prove each statement.

1. From the assumption, $G[P]$ contains no edges. By Theorem 2.1, $G[P]$ has a perfect matching. Both of these can hold simultaneously only when $P = \emptyset$.
2. Let C be the connected component of $G[O]$ which contains both v and w. From part (3) of Theorem 2.1 we get that any maximum matching of graph G which *exposes* vertex v contains a *perfect* matching of the subgraph $C'' = C[(V(C) \setminus \{v\})]$. Therefore, every maximum matching of G which survives in $G'' = G[(V(G) \setminus \{v\})]$ contains a perfect matching of C''. It follows that if we delete $w \in V(C'')$ as well from G'' to get G', then the matching number reduces by at least one, since no perfect matching of C'' can survive the deletion of vertex $w \in V(C'')$. □

When the set $I \cup P$ is independent in a graph of surplus at least 2, we get more properties for the set O:

Lemma 2.5. *Let G be a graph with* **surplus**$(G) \geq 2$*, and let $V(G) = O \uplus I \uplus P$ be the Gallai-Edmonds decomposition of graph G. If $I \cup P$ is an independent set in G, then:*

1. *There is at least one vertex $o \in O$ which has at least two neighbours in the set O.*
2. *Let $o \in O$ be a vertex which has at least one neighbour in the set O. Let $G' = G[(V(G) \setminus \{o\})]$, and let $V(G') = O' \uplus I' \uplus P'$ be the Gallai-Edmonds decomposition of graph G'. Then the induced subgraph $G'[P']$ contains at least one edge.*

Proof. We prove each statement.

1. Each component of $G[O]$ is factor critical (Theorem 2.1), and so has an odd number of vertices. So to prove this part it is enough to show that there is at least one component in $G[O]$ which has at least two (and hence, at least three) vertices. Suppose to the contrary that each component in $G[O]$ has exactly one vertex. Then the set O is an independent set in graph G. Since the set $I \cup P$ is independent by assumption, we get from Corollary 2.1 that $P = \emptyset$. Thus G is a *bipartite* graph with O and I being two parts of the bipartition. In particular, $N(O) \subseteq I$ and $N(I) \subseteq O$ both hold.

But since $\mathbf{surplus}(G) \geq 2$ this implies that both $|I| \geq (|O| + 2)$ and $|O| \geq (|I| + 2)$ hold simultaneously, which cannot happen. The claim follows.

2. Let $v \in O$ be a neighbour of the vertex o. Observe that by the definition of the set O, we have that $MM(G') = MM(G)$. If $G'[P']$ contains no edge, then the fact that $G[P']$ has a perfect matching (Theorem 2.1) implies that $P' = \emptyset$. Together with the Stability Lemma (see Theorem 2.1) this implies that $v \in O'$ in graph G'. Then by the definition of the set O' there is a maximum matching MM' of G' which exposes the vertex v. But then the matching MM' together with the edge $\{o, v\}$ forms a matching in graph G of size $MM(G') + 1 = MM(G) + 1$, a contradiction. Therefore $G'[P']$ must contain at least one edge. □

3 The Algorithm

In this section we describe our algorithm which solves Vertex Cover Above Lovász-Plummer in $\mathcal{O}^\star(3^{\hat{k}})$ time. We start with an overview of the algorithm. We then state the reduction and branching rules which we use, and prove their correctness. We conclude the section by proving that the algorithm correctly solves Vertex Cover Above Lovász-Plummer and that it runs within the stated running time.

3.1 Overview. The Vertex Cover Above Lovász-Plummer problem can be stated as:

> Vertex Cover Above Lovász-Plummer (VCAL-P)
> *Input:* A graph G and $\hat{k} \in \mathbb{N}$.
> *Parameter:* $\hat{k}$
> *Question:* Let $k = (2LP(G) - MM(G)) + \hat{k}$. Is $OPT(G) \leq k$?

At its core the algorithm is a simple branching algorithm. Indeed, we employ only two basic branching steps which are both "natural" for Vertex Cover: We either branch on the two end-points of an edge, or on a vertex v and a pair u, w of its neighbours. We set $\hat{k}$ as the *measure* for analyzing the running time of the algorithm. We ensure that this measure decreases by 1 in each branch of a branching step in the algorithm. Since $OPT(G) \geq (2LP(G) - MM(G))$ for every graph G—See Lemma 2.1—and $k \geq OPT(G)$ holds—by definition—for a Yes instance, it follows that $\hat{k} = (k + MM(G) - 2LP(G))$ is never negative for a Yes instance. Hence we can safely terminate the branching at depth $\hat{k}$. Since the worst-case branching factor is three, we get that the algorithm solves VCAL-P in time $\mathcal{O}^\star(3^{\hat{k}})$.

The algorithm—see Algorithm 2—modifies the input graph G in various ways as it runs. In our description of the algorithm we will sometimes, for the sake of brevity, slightly abuse the notation and use G to refer to the "current" graph at each point in the algorithm. Since the intended meaning will be clear from the context, this should not cause any confusion.

The crux of the algorithm is the manner in which it finds in G a small structure—such as an edge—on which to branch such that the measure $\hat{k}$ drops on each branch. Clearly, picking an arbitrary edge—say—and branching on its end-points will not suffice: this will certainly reduce the budget k by one on each branch, but we will have no guarantees on how the values $MM(G)$ and $LP(G)$ change, and so we cannot be sure that $\hat{k}$ drops. Instead, we apply a two-pronged strategy to find a small structure in the graph which gives us enough control over how the values $k, MM(G), LP(G)$ change, in such a way as to ensure that $\hat{k} = (k + MM(G) - 2LP(G))$ drops in each branch.

First, we employ a set of three reduction rules which give us control over how $LP(G)$ changes when we pick a vertex into the solution and delete it from G. These rules have the following nice properties:

- Each rule is sound—see Section 3.2—and can be applied in polynomial time;
- Applying any of these rules does not increase the measure $\hat{k}$, and;
- If none of these rules applies to G, then we can delete up to two vertices from G with an assured drop of 0.5 *per deleted vertex* in the value of $LP(G)$.

In this we follow the approach of Narayanaswamy et al. [19] who came up with this strategy to find a small structure on which to branch so that their measure—which was $(k - LP)$—would reduce on each branch. Indeed, we *reuse* three of their reduction rules, after proving that these rules do not increase our measure $\hat{k}$.

While these rules give us control over how $LP(G)$ changes when we delete a vertex, they do not give us such control over $MM(G)$. That is: let G be a graph to which none of these rules applies and let v be a vertex of G. Picking v into a solution and deleting it from G to get G' would (i) reduce k by 1, and (ii) make $LP(G') = LP(G) - 0.5$, but will *not* result in a drop in $\hat{k}$ *if it so happens* that the matching number of G' is the same as that of G. Thus for our approach to succeed we need to be able to consistently find, in a graph reduced with respect to the reduction rules, a small structure—say, an edge—on which to branch, such that deleting a small set of vertices from this structure would reduce the matching number of the graph by at least 1.

Algorithm 2 The $\mathcal{O}^\star(3^{\hat{k}})$ FPT algorithm for VERTEX COVER ABOVE LOVÁSZ-PLUMMER.

function VCAL-P($(G, \hat{k}, reduce)$) ▷ Initially invoked with *reduce* set to `True`.
  $k \leftarrow 2LP(G) - MM(G) + \hat{k}$
  if $reduce == True$ **then** ▷ Do the preprocessing only if *reduce* is `True`.
    while At least one of Reduction Rules 1, 2, and 3 applies to (G, k) **do**
      Apply the *first* such rule to (G, k), to get (G', k').
      $G \leftarrow G', k \leftarrow k'$

    $\hat{k} \leftarrow k + MM(G) - 2LP(G)$ ▷ Preprocessing ends.

  if $\hat{k} < 0$ **then** ▷ Check for the base cases.
    return `False`
  else if G has no edges **then** ▷ **surplus**$(G) \geq 2$ implies G has no vertices either.
    return `True`

  Compute the Gallai-Edmonds decomposition $V(G) = O \uplus I \uplus P$ of G.

  if $G[I \cup P]$ contains at least one edge $\{u, v\}$ **then** ▷ Branch on edge $\{u, v\}$.
    $G_1 \leftarrow G \setminus u$
    $G_2 \leftarrow G \setminus v$
    $k \leftarrow k - 1$
    $\hat{k}_1 \leftarrow k + MM(G_1) - 2LP(G_1)$
    $\hat{k}_2 \leftarrow k + MM(G_2) - 2LP(G_2)$
    return (VCAL-P($G_1, \hat{k}_1, True$) $\vee$ VCAL-P($G_2, \hat{k}_2, True$))

  else ▷ Since $I \cup P$ is an independent set, we have $P = \emptyset$.
    Choose a vertex $u \in O$ and two of its neighbours $v, w \in O$. ▷ Such vertices must exist.
    $G_1 \leftarrow G \setminus u$ ▷ Pick u in the solution.
    $k \leftarrow k - 1$
    $\hat{k}_1 \leftarrow k + MM(G_1) - 2LP(G_1)$,
    if VCAL-P($G_1, \hat{k}_1, False$)==`True` **then** ▷ No reduction rule will be applied now.
      return `True`
    else ▷ Pick both v and w in the solution.
      $G_2 \leftarrow G \setminus \{v, w\}$
      $k \leftarrow k - 1$ ▷ In effect reducing k by **two**.
      $\hat{k}_2 \leftarrow k + MM(G_2) - 2LP(G_2)$
      return VCAL-P($G_2, \hat{k}_2, True$)

The search for such a structure led us to the second and novel part of our approach, namely the use of the classical *Gallai-Edmonds decomposition* of graphs (Definition 2.1). We found that by first reducing the input graph with respect to the three reduction rules and then carefully choosing edges to branch based on the Gallai-Edmonds decomposition of the resulting graph, we could make the measure $\hat{k}$ drop on every branch. We describe the reduction and branching rules in the next two subsections.

3.2 The Reduction Rules. Given an instance $(G, \hat{k})$ of Vertex Cover Above Lovász-Plummer our algorithm first computes the number $k = (2LP(G) - MM(G) + \hat{k})$ so that (G, k) is the equivalent instance of (classical) Vertex Cover. Each of our three reduction rules takes an instance $(G = (V, E), k)$ of Vertex Cover, runs in polynomial time, and outputs an instance $(G' = (V', E'), k')$ of Vertex Cover. We say that a reduction rule is *sound* if it always outputs an *equivalent* instance. That is, if it is always the case that G has a vertex cover of size at most k if and only if G' has a vertex cover of size at most k'. We say that a reduction rule is *safe* if it never increases the measure $\hat{k}$. That is, if it is always the case that $k' + MM(G') - 2LP(G') \leq k + MM(G) - 2LP(G)$.

In the algorithm we apply these reduction rules *exhaustively*, in the order they are presented. That is, we take the input instance (G, k) and apply the first among Reduction Rule 1, Reduction Rule 2, and Reduction Rule 3 which *applies*—see definitions below—to (G, k) to obtain a modified instance (G', k'). We now set $G \leftarrow G', k \leftarrow k'$ and repeat this procedure, till none of the rules applies to the instance (G, k). We say that such an instance is *reduced* with respect to our reduction rules. The point of these reduction rules is that they help us push the *surplus* of the input graph to at least 2 in polynomial time, while not increasing the measure $\hat{k}$.

Reduction Rule 1. *Compute an optimal solution x to LPVC(G) such that all-$\frac{1}{2}$ is the unique optimum solution to LPVC($G[V^x_{1/2}]$). Set $G' = G[V^x_{1/2}], k' = k - |V^x_1|$.*

Reduction Rule 1 *applies* to (G, k) if and only if all-$\frac{1}{2}$ is *not* the unique solution to LPVC(G).

Reduction Rule 2. *If there is an independent set $Z \subseteq V(G)$ such that* $\mathbf{surplus}(Z) = 1$ *and $N(Z)$ is* not *an independent set in G, then set $G' = G \setminus (Z \cup N(Z)), k' = k - |N(Z)|$.*

Reduction Rule 2 *applies* to (G, k) if and only if (i) Reduction Rule 1 does *not* apply, and (ii) G has a vertex subset Z with the properties stated in Reduction Rule 2.

Reduction Rule 3. *If there is an independent set $Z \subseteq V(G)$ such that* $\mathbf{surplus}(Z) = 1$ *and $N(Z)$ is an independent set in G, then remove Z from G and* identify *the vertices of $N(Z)$—that is, delete all of $N(Z)$, add a new vertex z, and make z adjacent to all vertices of the set $(N(N(Z)) \setminus Z)$—to get G', and set $k' = k - |Z|$.*

Reduction Rule 3 *applies* to (G, k) if and only if (i) neither of the previous rules applies, and (ii) G has a vertex subset Z with the properties stated in Reduction Rule 3.

All our reduction rules are due to Narayanaswamy et al. [19, Preprocessing Rules 1 and 2]. The soundness of these rules and their running-time bounds also follow directly from their work. We need to argue, however, that these rules are safe for our measure $\hat{k}$.

Lemma 3.1. **[19]** *All the three reduction rules are sound, and each can be applied in time polynomial in the size of the input (G, k).*

It remains to show that none of these reduction rules increases the measure $\hat{k} = k + MM(G) - 2LP(G)$. That is, let $(G = (V, E), k)$ be an instance of Vertex Cover to which one of these rules applies, and let $(G' = (V', E'), k')$ be the instance obtained by applying the rule to (G, k). Then we have to show, for each rule, that $\hat{k}' = k' + MM(G') - 2LP(G') \leq \hat{k}$ holds. We establish this by examining how each rule changes the values k, MM and LP. In each case, let x be an optimum half-integral solution to $LPVC(G)$ such that all-$\frac{1}{2}$ is the unique optimum solution to $LPVC(G[V^x_{1/2}])$, and let $LP(G)$ be the (optimum) value of this solution. Also, let x' be an optimum half-integral solution to $LPVC(G')$, and let $LP(G')$ be the value of this solution.

Lemma 3.2. *Reduction Rule 1 is safe.*

Proof. From the definition of the rule we get that $k' = k - |V^x_1|$. Now since $x' \equiv \frac{1}{2}$ is the unique optimum solution to $LPVC(G')$, and $G' = G[V^x_{1/2}]$, we get that $LP(G') = LP(G) - |V^x_1|$, and hence that $2LP(G') = 2LP(G) - 2|V^x_1|$. From Lemma 2.4 and the construction of the graph G' we get that $MM(G) \geq MM(G') + |V^x_1|$, and hence that $MM(G') \leq MM(G) - |V^x_1|$. Putting these together, we get that $\hat{k}' \leq \hat{k}$. □

Lemma 3.3. *Reduction Rule 2 is safe.*

Proof. From the definition of the rule we get that $k' = k - |N(Z)|$. We bound the other two quantities.

Claim 1. $LP(G') \geq LP(G) - |N(Z)| + \frac{1}{2}$.

Proof. Since Reduction Rule 1 did not apply to (G, k) we know that all-$\frac{1}{2}$ is the unique optimum solution to $LPVC(G)$. It follows from this and the construction of graph G' that $LP(G') = \sum_{u \in V'} x'(u) = LP(G) - \frac{1}{2}(|Z| + |N(Z)|) + \frac{1}{2}(|V_1^{x'}| - |V_0^{x'}|)$. Adding and subtracting $\frac{1}{2}(|N(Z)|)$, we get $LP(G') = LP(G) - |N(Z)| + \frac{1}{2}(|N(Z)| - |Z|) + \frac{1}{2}(|V_1^{x'}| - |V_0^{x'}|) = LP(G) - |N(Z)| + \frac{1}{2}(|N(Z)| + |V_1^{x'}|) - \frac{1}{2}(|Z| + |V_0^{x'}|)$. Now since $V(G') = (V(G) \setminus (Z \cup N(Z)))$ and $V_0^{x'} \subseteq V(G')$, we get that $Z \cup V_0^{x'}$ is an independent set in G, and that $N(Z \cup V_0^{x'}) = N(Z) \cup V_1^{x'}$ in G (Recall that $N(V_0^x) = V_1^x$ for any half-integral optimal solution x.). Since $\textbf{surplus}(G) \geq 1$—Lemma 2.2—we get that in G, $|N(Z \cup V_0^{x'})| - |Z \cup V_0^{x'}| \geq 1$, which gives $(|N(Z)| + |V_1^{x'}|) - (|Z| + |V_0^{x'}|) \geq 1$, and so we get that $\frac{1}{2}(|N(Z)| + |V_1^{x'}|) - \frac{1}{2}(|Z| + |V_0^{x'}|) \geq \frac{1}{2}$. Substituting this in the equation for $LP(G')$, we get that $LP(G') \geq LP(G) - |N(Z)| + \frac{1}{2}$. □

Now we bound the drop in $MM(G)$.

Claim 2. $MM(G') \leq MM(G) - |Z|$.

Proof. Consider the bipartite graph $\hat{G}$ obtained from the induced subgraph $G[Z \cup N(Z)]$ of G by deleting every edge which has both endpoints in $N(Z)$. Observe that since $\textbf{surplus}(Z) = 1$ in G, we get that $|N(X)| \geq |X| + 1 \; \forall X \subseteq Z$, both in G and in $\hat{G}$. Hence by Hall's Theorem we get that $\hat{G}$ contains a matching saturating Z, and hence that $MM(\hat{G}) = |Z|$. But from the construction of graph G' we get that $MM(G) \geq MM(G') + MM(\hat{G})$. Substituting for $MM(\hat{G})$, we get that $MM(G') \leq MM(G) - |Z|$. □

Putting all these together, we get that $\hat{k}' = k' + MM(G') - 2LP(G') \leq (k - |N(Z)|) + (MM(G) - |Z|) - 2(LP(G) - |N(Z)| + \frac{1}{2}) = k + MM(G) - 2LP(G) + (|N(Z)| - |Z| - 1) = k + MM(G) - 2LP(G) = \hat{k}$, where the last-but-one equality follows from the fact that $\textbf{surplus}(Z) = 1$. Thus we have that $\hat{k}' \leq \hat{k}$. □

Lemma 3.4. *Reduction Rule 3 is safe.*

Proof. From the definition of the rule we get that $k' = k - |Z|$. We bound the other quantities. Let z be the vertex in G' which results from identifying the vertices of $N(Z)$ as stipulated by the reduction rule.

Claim 3. $LP(G') \geq LP(G) - |Z|$.

Proof. Suppose not. Then we get—from the half-integrality property of the relaxed LP for Vertex Cover—that $LP(G') \leq LP(G) - |Z| - \frac{1}{2}$. We show that this inequality leads to a contradiction. We consider three cases based on the value of $x'(z)$, which must be one of $\{0, \frac{1}{2}, 1\}$. Recall that $\textbf{surplus}(Z) = 1$ and so $|N(Z)| = |Z| + 1$.

Case 1: $x'(z) = 1$. Consider a function $x'' : V \to \{0, \frac{1}{2}, 1\}$ defined as follows. For every vertex v in $V' \setminus \{z\}$, x'' retains the value assigned by x'; that is, $x''(v) = x'(v)$. For every vertex v in the set $N(Z)$, set $x''(v) = 1$ and for every vertex v in the set Z, $x''(v) = 0$. It is not difficult to check that x'' is a feasible solution to the relaxed Vertex Cover LP for G. But now the value of this solution, $w(x'') = LP(G') - x'(z) + |N(Z)| = LP(G') - 1 + (|Z| + 1) \leq LP(G) - \frac{1}{2}$. Thus x'' is a feasible solution for G whose value is less than that of the *optimal* solution, a contradiction.

Case 2: $x'(z) = 0$. Now consider the following function $x'' : V \to \{0, \frac{1}{2}, 1\}$. For every vertex v in $V' \setminus \{z\}$, x'' retains the value assigned by x': $x''(v) = x'(v)$. For every vertex $v \in Z$ set $x''(v) = 1$ and for every vertex $v \in N(Z)$, set $x''(v) = 0$. It is again not difficult to check that x'' is a feasible solution to the relaxed Vertex Cover LP for G. And the value of this solution, $w(x'') = LP(G') + |Z| \leq LP(G) - \frac{1}{2}$, again a contradiction.

Case 3: $x'(z) = \frac{1}{2}$. Consider again a function $x'' : V \to \{0, \frac{1}{2}, 1\}$, defined as follows. For every vertex v in $V' \setminus \{z\}$, x''—once again—retains the value assigned by x': $x''(v) = x'(v)$. For every vertex $v \in (Z \cup N(Z))$, set $x''(v) = \frac{1}{2}$. This is once again easily verified to be a feasible solution to the relaxed Vertex Cover LP for G, and its value is $w(x'') = LP(G') - x'(z) + \frac{1}{2}(|Z| + |N(Z)|) = LP(G') - \frac{1}{2} + \frac{1}{2}(|Z| + |Z| + 1) \leq LP(G) - \frac{1}{2}$, once again a contradiction.

Thus our contrary assumption leads to a contradiction in all possible cases, and this proves the claim. □

Now we bound the drop in $MM(G)$.

Claim 4. $MM(G') \leq MM(G) - |Z|$.

Proof. For an arbitrary vertex $u \in N(Z)$, let G_u be the bipartite subgraph $G[Z \cup (N(Z) \setminus \{u\})]$, which is an induced subgraph of G. Since $\textbf{surplus}(Z) = 1$ in G and only one vertex from $N(Z)$ is missing in G_u, we get that $|N(X)| \geq |X| \; \forall X \subseteq Z$ holds in G_u. Hence by Hall's Theorem we get that G_u contains a matching saturating Z, and hence that $MM(G_u) = |Z|$.

Now consider a maximum matching MM' of G'. Starting with MM' we can construct a matching MM'' of size $|MM'|$ in the original graph G which saturates at most one vertex of $N(Z)$ and none of Z, as follows: There is at most one edge in MM' which saturates the vertex z. If there is *no* edge in MM' which saturates the vertex z, then we set $MM'' = MM'$. It is not difficult to see that MM'' saturates no vertex in $Z \cup N(Z)$. If there is an edge $\{z, v\} \in MM'$, then we pick an arbitrary vertex $u \in N(Z)$ such that $\{u, v\}$ is an edge in G—such a vertex must exist since the edge $\{z, v\}$ exists in G'. We set $MM'' = (MM' \setminus \{\{z, v\}\}) \cup \{\{u, v\}\}$. It is not difficult to see that MM'' is a matching in G which saturates exactly one vertex—$u \in N(Z)$—in $Z \cup N(Z)$.

If the matching MM'', constructed as above, does not saturate any vertex of $N(Z)$, then we choose u to be an arbitrary vertex of $N(Z)$. If MM'' does saturate a vertex of $N(Z)$, then we set u to be that vertex. In either case, the union of MM'' and any maximum matching of the induced bipartite subgraph G_u is itself a matching of G, of size $MM(G_u) + MM(G') = |Z| + MM(G')$. It follows that $MM(G) \geq |Z| + MM(G')$, which implies $MM(G') \leq MM(G) - |Z|$. □

Putting all these together, we get that $\hat{k}' = k' + MM(G') - 2LP(G') \leq (k - |Z|) + (MM(G) - |Z|) - 2(LP(G) - |Z|) = k + MM(G) - 2LP(G) = \hat{k}$. Thus we have that $\hat{k}' \leq \hat{k}$. □

Observe that if $\mathbf{surplus}(G) = 1$ then at least one of Reduction Rule 2 and Reduction Rule 3 necessarily applies. From this and Lemma 2.2 we get the following useful property of graphs on which none of these rules applies.

LEMMA 3.5. *None of the three reduction rules applies to an instance (G, k) of* VERTEX COVER *if and only if $\mathbf{surplus}(G) \geq 2$.*

Summarizing the results of this section, we have:

LEMMA 3.6. *Given an instance $(G, \hat{k})$ of* VERTEX COVER ABOVE LOVÁSZ-PLUMMER *we can, in time polynomial in the size of the input, compute an instance $(G', \hat{k}')$ such that:*

1. *The two instances are equivalent: G has a vertex cover of size at most $(2LP(G) - MM(G)) + \hat{k}$ if and only if G' has a vertex cover of size at most $(2LP(G') - MM(G')) + \hat{k}'$;*

2. *$\hat{k}' \leq \hat{k}$, and $\mathbf{surplus}(G') \geq 2$.*

3.3 The Branching Rules.

Let $(G = (V, E), k)$ be the instance of VERTEX COVER obtained after exhaustively applying the reduction rules to the input instance, and let $\hat{k} = k + MM(G) - 2LP(G)$. Then $\mathbf{surplus}(G) \geq 2$. If $\hat{k} < 0$ then the instance $(G, \hat{k})$ of VCAL-P is trivially a NO instance (See Section 3.1), and we return NO and stop. In the remaining case $\hat{k} \geq 0$, and we apply one of two branching rules to the instance to reduce the measure $\hat{k}$. Each of our branching rules takes an instance $(G, \hat{k})$ of VCAL-P, runs in polynomial time, and outputs either two or three instances $(G_1, \hat{k}_1), (G_2, \hat{k}_2)[, (G_3, \hat{k}_3)]$ of VCAL-P. The algorithm then recurses on each of these instances in turn. We say that a branching rule is *sound* if the following holds: $(G, \hat{k})$ is a YES instance of VCAL-P if and only if *at least one* of the (two or three) instances output by the rule is a YES instance of VCAL-P. We now present the branching rules, prove that they are sound, and show that the measure $\hat{k}$ drops by at least 1 on each branch of each rule.

Before starting with the branching, we compute the *Gallai-Edmonds decomposition* of the graph $G = (V, E)$. This can be done in polynomial time [17] and yields a partition of the vertex set V into three parts $V = O \uplus I \uplus P$, one or more of which may be empty—see Definition 2.1. We then branch on edges in the graph which are carefully chosen with respect to this partition. Branching Rule 1 applies to the graph G if and only if the induced subgraph $G[I \cup P]$ contains at least one edge.

BRANCHING RULE 1. *Branch on the two end-points of an edge $\{u, v\}$ in the induced subgraph $(G[I \cup P])$. More precisely, the two branches generate two instances as follows:*

Branch 1: $G_1 \leftarrow (G \setminus \{u\}), k_1 \leftarrow (k - 1), \hat{k_1} \leftarrow k_1 + MM(G_1) - 2LP(G_1)$

Branch 2: $G_2 \leftarrow (G \setminus \{v\}), k_2 \leftarrow (k - 1), \hat{k_2} \leftarrow k_2 + MM(G_2) - 2LP(G_2)$

This rule outputs the two instances $(G_1, \hat{k}_1)$ and $(G_2, \hat{k}_2)$.

Branching Rule 2 applies to graph G if and only if Branching Rule 1 does not apply to G.

BRANCHING RULE 2. *Pick a vertex $u \in O$ of G which has at least two neighbours $v, w \in O$. Construct the graph $G' = G \setminus \{u\}$ and compute its Gallai-Edmonds decomposition $O' \uplus I' \uplus P'$. Pick an edge $\{x, y\}$ in $G'[P']$. The three branches of this branching rule generate three instances as follows:*

Branch 1: $G_1 \leftarrow (G \setminus \{v, w\}), k_1 \leftarrow (k - 2), \hat{k_1} \leftarrow k_1 + MM(G_1) - 2LP(G_1)$

Branch 2: $G_2 \leftarrow (G' \setminus \{x\}), k_2 \leftarrow (k - 2), \hat{k_2} \leftarrow k_2 + MM(G_2) - 2LP(G_2)$

Branch 3: $G_3 \leftarrow (G' \setminus \{y\}), k_3 \leftarrow (k - 2), \hat{k_3} \leftarrow k_3 + MM(G_3) - 2LP(G_3)$

This rule outputs the three instances $(G_1, \hat{k}_1)$, $(G_2, \hat{k}_2)$, and $(G_3, \hat{k}_3)$.

Note that in the pseudocode of Algorithm 2 we have not, for the sake of brevity, written out the three branches of the second rule. Instead, we have used a boolean switch (*reduce*) as a third argument to the procedure to simulate this behaviour.

LEMMA 3.7. *Both branching rules are sound, and each can be applied in time polynomial in the size of the input* (G, k)*.*

Proof. We first prove that the rules can be applied in polynomial time. Recall that we can compute the Gallai-Edmonds decomposition of graph G in polynomial time [17].

Branching Rule 1: Follows more or less directly from the definition of the branching rule.

Branching Rule 2: It is not difficult to see that the applicability of this rule can be checked in polynomial time. If the rule does apply, then the set P is empty and the set I is an independent set in G (Corollary 2.1). By Lemma 2.5, there is at least one vertex in the set O which has at least two neighbours in O. Hence vertices u, v, w of the kind required in the rule necessarily exist in G. From Lemma 2.5 we also get that an edge $\{x, y\}$ of the required kind also exists. Given the existence of these, it is not difficult to see that the rule can be applied in polynomial time.

We now show that the rules are sound. Recall that a branching rule is sound if the following holds: any instance $(G, \hat{k})$ of VCAL-P is a YES instance if and only if at least one of the instances obtained by applying the rule to $(G, \hat{k})$ is a YES instance.

Branching Rule 1: The soundness of this rule is not difficult to see since it is a straightforward branching on the two end-points of an edge. Nevertheless, we include a full argument for the sake of completeness. So let $(G, \hat{k})$ be a YES instance of VCAL-P, and let $(G_1, \hat{k_1}), (G_2, \hat{k_2})$ be the two instances obtained by applying Branching Rule 1 to $(G, \hat{k})$. Since $(G, \hat{k})$ is a YES instance, the graph G has a vertex cover, say S, of size at most $k = 2LP(G) - MM(G) + \hat{k}$. Then $(S \cap \{u, v\}) \neq \emptyset$. Suppose $u \in S$. Then $S_1 = S \setminus \{u\})$ is a vertex cover of the graph $G_1 = G \setminus \{u\}$, of size at most $k_1 = (k - 1)$. It follows that for $\hat{k_1} = k_1 + MM(G_1) - 2LP(G_1)$, $(G_1, \hat{k_1})$ is a YES instance of VCAL-P. A symmetric argument gives us the YES instance $(G_2, \hat{k_2})$ of VCAL-P for the case $v \in S$.

Conversely, suppose $(G_1, \hat{k_1})$ is a YES instance of VCAL-P where $G_1 = G \setminus \{u\}, k_1 = (k - 1), \hat{k_1} = k_1 + MM(G_1) - 2LP(G_1)$. Then graph G_1 has a vertex cover, say S_1, of size at most $k_1 = (k - 1)$. $S = S_1 \cup \{u\}$ is then a vertex cover of graph G of size at most k, and so $(G, \hat{k})$, where $\hat{k} = k + MM(G) - 2LP(G)$ is a YES instance of VCAL-P as well. An essentially identical argument works for the case when $(G_2, \hat{k_2})$ is a YES instance of VCAL-P.

Branching Rule 2: It is not difficult to see that this rule is sound, either, since it consists of exhaustive branching on a vertex u. We include the arguments for completeness. So let $(G, \hat{k})$ be a YES instance of VCAL-P, and let $(G_1, \hat{k_1}), (G_2, \hat{k_2}), (G_2, \hat{k_2})$ be the three instances obtained by applying Branching Rule 2 to $(G, \hat{k})$. Since $(G, \hat{k})$ is a YES instance, the graph G has a vertex cover, say S, of size at most $k = 2LP(G) - MM(G) + \hat{k}$. We consider two cases: $u \in S$, and $u \notin S$. First consider the case $u \notin S$. Then all the neighbours of u must be in S. In particular, $\{v, w\} \subseteq S$. It follows that the set $S \setminus \{v, w\}$ is a vertex cover of the graph $G_1 = (G \setminus \{v, w\})$, of size at most $k_1 = (k - 2)$. Hence we get that for $\hat{k_1} = k_1 + MM(G_1) - 2LP(G_1)$, $(G_1, \hat{k_1})$ is a YES instance of VCAL-P.

Now consider the case $u \in S$. Then the set $S' = S \setminus \{u\}$ is a vertex cover of the graph $G' = G \setminus \{u\}$, of size at most $k - 1$. Since $\{x, y\}$ is an edge in the graph G', we get that $(S' \cap \{x, y\}) \neq \emptyset$. Suppose $x \in S'$. Then $S_2 = (S' \setminus \{x\})$ is a vertex cover of the graph $G_2 = G' \setminus \{x\}$, of size at most $k_2 = (k - 2)$. It follows that for $\hat{k_2} = k_2 + MM(G_2) - 2LP(G_2)$, $(G_2, \hat{k_2})$ is a YES instance of VCAL-P. A symmetric argument gives us the YES instance $(G_3, \hat{k_3})$ of VCAL-P for the case $y \in S$.

Conversely, suppose $(G_1, \hat{k_1})$ is a YES instance of VCAL-P where $G_1 = G \setminus \{v, w\}, k_1 = (k - 2), \hat{k_1} = k_1 + MM(G_1) - 2LP(G_1)$. Then graph G_1 has a vertex cover, say S_1, of size at most $k_1 = (k - 2)$. $S = S_1 \cup \{v, w\}$ is then a vertex cover of graph G of size at most k, and so $(G, \hat{k})$ where $\hat{k} = k + MM(G) - 2LP(G)$, is a YES instance of VCAL-P as well. Essentially identical arguments work for the cases when $(G_2, \hat{k_2})$ or $(G_3, \hat{k_3})$ is a YES instance of VCAL-P. □

Our choice of vertices on which we branch ensures that the measure drops by at least one on each branch of the algorithm.

LEMMA 3.8. *Let* $(G, \hat{k})$ *be an input given to one of the branching rules, and let* $(G_i, \hat{k}_i)$ *be an instance output by the rule. Then* $\hat{k}_i \leq (\hat{k} - 1)$*.*

Proof. Recall that by definition, $\hat{k} = k + MM(G) - 2LP(G)$. We consider each branching rule. We reuse the notation from the description of each rule.

Branching Rule 1: Consider **Branch 1**. Since $u \in (I \cup P)$, *every* maximum matching of graph G saturates vertex u (Theorem 2.1). Hence we get that $MM(G_1) = (MM(G) - 1)$. Since **surplus**$(G) \geq 2$ we get—from Lemma 2.3—that $LP(G_1) = (LP(G) - \frac{1}{2})$. And since $k_1 = (k - 1)$ by definition, we get that $\hat{k}_1 = k_1 + MM(G_1) - 2LP(G_1) = (k-1)+(MM(G)-1)-2(LP(G)-\frac{1}{2}) = k+MM(G)-2LP(G)-1 = (\hat{k}-1)$. An essentially identical argument applied to the (symmetrical) **Branch 2** tells us that $\hat{k}_2 = (\hat{k} - 1)$.

Branching Rule 2: Consider **Branch 1**. Since $\{u, v, w\} \subseteq O$ and v, w are neighbours of vertex u, we get that vertices v and w belong to the same connected component of the induced subgraph $G[O]$ of G. It follows (Corollary 2.1) that $MM(G_1) \leq (MM(G) - 1)$. Since **surplus**$(G) \geq 2$ and $G_1 = G \setminus \{v, w\}$ we get—from Lemma 2.3—that $LP(G_1) = (LP(G) - 1)$. And since $k_1 = (k - 2)$ by definition, we get that $\hat{k}_1 = k_1 + MM(G_1) - 2LP(G_1) \leq (k-2)+(MM(G)-1)-2(LP(G)-1) = k + MM(G) - 2LP(G) - 1 = (\hat{k} - 1)$.

Now consider **Branch 2**. Since $u \in O$ we get—from the definition of the Gallai-Edmonds decomposition—that $MM(G') = MM(G)$. Now since $x \in P'$ we get— Theorem 2.1—that $MM(G_2) = (MM(G') - 1) = (MM(G) - 1)$. Since **surplus**$(G) \geq 2$ and $G_2 = G \setminus \{u, x\}$ we get—from Lemma 2.3—that $LP(G_2) = (LP(G) - 1)$. And since $k_2 = (k - 2)$ by definition, we get that $\hat{k}_2 = k_2+MM(G_2)-2LP(G_2) = (k-2)+(MM(G)-1)-2(LP(G)-1) = k+MM(G)-2LP(G)-1 = (\hat{k}-1)$. An essentially identical argument applied to the (symmetrical) **Branch 3** tells us that $\hat{k}_3 = (\hat{k} - 1)$. □

3.4 Putting it All Together: Correctness and Running Time Analysis. The correctness of our algorithm and the claimed bound on its running time follow more or less directly from the above discussion.

Proof. [Proof of Theorem 1.1] We claim that Algorithm 2 solves Vertex Cover Above Lovász-Plummer in $\mathcal{O}^\star(3^{\hat{k}})$ time. The correctness of the algorithm follows from the fact that both the reduction rules and the branching rules are sound—Lemma 3.1 and Lemma 3.7. This means that (i) no reduction rule ever converts a Yes instance into a No instance or *vice versa*, and (ii) each branching rule outputs *at least one* Yes instance when given a Yes instance as output, and *all* No instances when given a No instance as input. It follows that if the algorithm outputs Yes or No, then the input instance must also have been Yes or No, respectively.

The running time bound follows from three factors. Firstly, all the reduction rules are safe, and so they never increase the measure $\hat{k}$—see Lemma 3.2, Lemma 3.3, Lemma 3.4, and Lemma 3.6. Also, each reduction rule can be executed in polynomial time, and since each reduction rule reduces the number of vertices in the graph by at least one, they can be exhaustively applied in polynomial time. Secondly, each branch of each branching rule reduces the measure by at least 1—Lemma 3.8—from which we get that each path from the root of the recursion tree—where the measure is the original value of $\hat{k}$—to a leaf—where the measure first becomes zero or less—has length at most $\hat{k}$. Further, the largest branching factor is 3, which means that the number of nodes in the recursion tree is $O(3^{\hat{k}})$. Thirdly, we know that the computation at each node in the recursion tree takes polynomial time. This follows from Lemma 3.7 for the nodes where we do branching. As for the leaf nodes: Since we know that the measure will never be negative for a Yes instance (Lemma 2.1), since we can check for the applicability of each branching rule in polynomial time, and since each branching rule reduces the measure by at least one, we can solve the instance at each leaf node in polynomial time. □

4 Conclusion

Motivated by an observation of Lovász and Plummer, we derived the new lower bound $2LP(G) - MM(G)$ for the vertex cover number of a graph G. This bound is at least as large as the bounds $MM(G)$ and $LP(G)$ which have hitherto been used as lower bounds for investigating above-guarantee parameterizations of Vertex Cover. We took up the parameterization of the Vertex Cover problem above our "higher" lower bound $2LP(G) - MM(G)$, which we call the Lovász-Plummer lower bound for Vertex Cover. We showed that Vertex Cover remains fixed-paramter tractable even when parameterized above the Lovász-Plummer bound. The main result of this work is an $\mathcal{O}^\star(3^{\hat{k}})$ algorithm for Vertex Cover Above Lovász-Plummer.

The presence of both $(-2LP(G))$ and $MM(G)$—in addition to the "solution size"—in our measure made it challenging to find structures on which to branch; we had to be able to control each of these values and their interplay in order to ensure a drop in the measure at each branch. The main new idea which we employed for overcoming this hurdle is the use of the Gallai-Edmonds decomposition of graphs for finding structures on which to branch profitably. The main technical effort in this work has been expended in proving that our choice of vertices/edges from the Gallai-Edmonds decomposition actually work in the way we want. Note, however, that the branching rules themselves are very simple; it is only the analysis which is involved.

The most immediate open problem is whether we can improve on the base 3 of the FPT running time. Note that any such improvement directly implies FPT algorithms of the same running time for Above-Guarantee Vertex Cover and Vertex Cover Above LP. Tempted by this implication, we have tried to bring this number down but, so far, in vain. Another question which suggests itself is: Is this the best lower bound for vertex cover number above which Vertex Cover is FPT? How far can we push the lower bound before the problem becomes intractable?

5 Acknowledgements

We thank our anonymous reviewers for their careful reading of the submitted version, and helpful comments.

References

[1] R. Balasubramanian, Michael R. Fellows, and Venkatesh Raman. An improved fixed-parameter algorithm for vertex cover. *Information Processing Letters*, 65(3):163–168, 1998.

[2] Béla Bollobás. *Random Graphs*. Cambridge University Press, second edition, 2001.

[3] Jean-Marie Bourjolly and William R Pulleyblank. König-egerváry graphs, 2-bicritical graphs and fractional matchings. *Discrete Applied Mathematics*, 24(1):63–82, 1989.

[4] L. Sunil Chandran and Fabrizio Grandoni. Refined memorization for vertex cover. *Information Processing Letters*, 93(3):125–131, 2005.

[5] Jianer Chen, Iyad A. Kanj, and Ge Xia. Improved upper bounds for vertex cover. *Theoretical Computer Science*, 411(40-42):3736–3756, 2010.

[6] Marek Cygan, Marcin Pilipczuk, Michal Pilipczuk, and Jakub Onufry Wojtaszczyk. On multiway cut parameterized above lower bounds. In Dániel Marx and Peter Rossmanith, editors, *Parameterized and Exact Computation - 6th International Symposium, IPEC 2011, Saarbrücken, Germany, September 6-8, 2011. Revised Selected Papers*, volume 7112 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2012.

[7] Marek Cygan, Marcin Pilipczuk, Michal Pilipczuk, and Jakub Onufry Wojtaszczyk. On multiway cut parameterized above lower bounds. *Theory of Computing Systems*, 5(1):3, 2013.

[8] Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013.

[9] Jack Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965.

[10] Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Springer-Verlag, 2006.

[11] Tibor Gallai. Kritische graphen ii. *Magyar Tud. Akad. Mat. Kutató Int. Közl*, 8:373–395, 1963.

[12] Tibor Gallai. Maximale systeme unabhängiger kanten. *Magyar Tud. Akad. Mat. Kutató Int. Közl*, 9(9):401–413, 1964.

[13] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001.

[14] Richard M. Karp. Reducibility Among Combinatorial Problems. In *Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York*, The IBM Research Symposia Series, pages 85–103. Plenum Press, New York, 1972.

[15] Michael Lampis. A kernel of order $2k - c \log k$ for vertex cover. *Information Processing Letters*, 111(23-24):1089–1091, 2011.

[16] Daniel Lokshtanov, N. S. Narayanaswamy, Venkatesh Raman, M. S. Ramanujan, and Saket Saurabh. Faster parameterized algorithms using linear programming. *ACM Transactions on Algorithms*, 11(2):15:1–15:31, 2014.

[17] L. Lovász and D. Plummer. *Matching Theory*. AMS Chelsea Publishing Series. American Mathematical Society, 2009.

[18] Sounaka Mishra, Venkatesh Raman, Saket Saurabh, Somnath Sikdar, and C. R. Subramanian. The complexity of könig subgraph problems and above-guarantee vertex cover. *Algorithmica*, 61(4):857–881, 2011.

[19] N. S. Narayanaswamy, Venkatesh Raman, M. S. Ramanujan, and Saket Saurabh. LP can be a cure for Parameterized Problems. In *29th International Symposium on Theoretical Aspects of Computer Science, STACS 2012, February 29th - March 3rd, 2012, Paris, France*, volume 14 of *LIPIcs*, pages 338–349, 2012.

[20] George L Nemhauser and Leslie E Trotter Jr. Properties of vertex packing and independence system polyhedra. *Mathematical Programming*, 6(1):48–61, 1974.

[21] George L Nemhauser and Leslie E Trotter Jr. Vertex packings: structural properties and algorithms. *Mathematical Programming*, 8(1):232–248, 1975.

[22] Rolf Niedermeier and Peter Rossmanith. Upper bounds for vertex cover further improved. In Christoph Meinel and Sophie Tison, editors, *STACS 99, 16th Annual Symposium on Theoretical Aspects of Computer Science, Trier, Germany, March 4-6, 1999, Proceedings*, volume 1563 of *Lecture Notes in Computer Science*, pages 561–570. Springer, 1999.

[23] WR Pulleyblank. Minimum node covers and 2-bicritical graphs. *Mathematical programming*, 17(1):91–103,

1979.

[24] Venkatesh Raman, M. S. Ramanujan, and Saket Saurabh. Paths, flowers and vertex cover. In Camil Demetrescu and Magnús M. Halldórsson, editors, *Algorithms - ESA 2011 - 19th Annual European Symposium, Saarbrücken, Germany, September 5-9, 2011. Proceedings*, volume 6942 of *Lecture Notes in Computer Science*, pages 382–393. Springer, 2011.

[25] Igor Razgon and Barry O'Sullivan. Almost 2-sat is fixed-parameter tractable (extended abstract). In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part I: Tack A: Algorithms, Automata, Complexity, and Games*, volume 5125 of *Lecture Notes in Computer Science*, pages 551–562. Springer, 2008.

[26] Magnus Wahlström. Half-integrality, LP-branching and FPT algorithms. In Chandra Chekuri, editor, *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 1762–1781. SIAM, 2014.