Technical Report TR-05-04
Improving k-Vertex Cover Algorithms for Graphs of Bounded Degree

P.J. Taillon
School of Computer Science
Carleton University
Ottawa, Ontario, Canada
ptaillon@scs.carleton.ca

ABSTRACT

We describe a new approach to improve algorithms for solving the k-VERTEX COVER problem. The algorithm uses graph cutting during the tree search phase as a basis for disconnecting and explicitly processing small fractions of the residual graph. Our algorithm applies to graphs of small bounded degree, adapts existing branching rules, and incurs no additional complexity. Bounded degree graph instances are generated during tree search phases of the current best FPT k-vertex cover algorithms. Previous separator-based approaches were constrained to planar graphs, or graphs of bounded-genus, and required complex dynamic programming. We also investigate the applicability of our new algorithm to solving the MAXIMUM INDEPENDENT SET problem in graphs of small bounded degree.

Categories and Subject Descriptors

G.2 [**Discrete Mathematics**]: Graph Theory

General Terms

Theory, Algorithms, Experimentation

Keywords

Complexity, Parameterized complexity, Graph algorithms, Vertex cover, Exact algorithms

1. INTRODUCTION

The problem of finding a *vertex cover* in a graph, $G = (V, E)$, that a subset of vertices $VC \subseteq V$, $|VC| \leq K$, such that every edge is adjacent to one of the vertices in VC, is one of the original six problems shown to be NP-complete by Karp [28]. This problem, among its many applications, figures prominently in VLSI design, computational biology, to name a few.

Downey and Fellows [17, 18, 19, 20, 21] developed parameterized tractability as a framework for studying classes of problems that are NP-complete and yet for which there exist algorithms that decide the problem in time bounded by an exponential function of some fixed parameter, k. Algorithms for problems that are *fixed-parameter tractable*, or FPT, have a complexity described by $O(n^{O(1)}+f(k))$, where n is the size of the problem instance and f is an arbitrary function. Downey and Fellows introduced the notion of *klam* value as a means of evaluating and comparing FPT algorithms. The klam value of an FPT algorithm is the largest k for which $f(k) \leq \mathcal{U}$, where $\mathcal{U} = 10^{20}$ is taken as a bound on the maximum number of steps of any computation. Although there are many problems that are fixed-parameter tractable not all parameterized problems are in the class FPT. The W-hierarchy consists of classes of problems that are not likely to be fixed-parameter tractable, with the class $W[1]$ representing the lowest level of intractability [21, 11]. For example, k-INDEPENDENT SET is complete for $W[1]$ and k-DOMINATING SET is complete for $W[2]$.

1.1 The k-Vertex Cover problem

Recent FPT algorithms for the k-VERTEX COVER problem consist of two phases: a series of *kernelization* operations followed by some form of *bounded tree search*. Kernelization is a process in which a problem instance, $\langle G = (V, E), k\rangle$, is reduced to an instance, $\langle G' = (V', E'), k'\rangle$, where the size of V' is bounded by a function of the parameter k and $k' \leq k$.

Bounded tree search is a powerful combinatorial search method in which a kernelized problem instance, $\langle G', k'\rangle$, can be exhaustively analyzed, and the associated search tree is bounded in size by a function, $f(k')$. While kernelization is performed in polynomial time, bounded tree search is exponential in the size of the parameter and so (typically) requires the most time to complete.

One of the first algorithms to use kernelization for finding a k-vertex cover is attributed to Buss and Goldsmith [9]: given a instance, $\langle G = (V, E), k\rangle$, their algorithm derives a new instance, $\langle G' = (V', E'), k'\rangle$, $k' \leq k$, $|V'| \leq k^2$, such that G' has a k'-vertex cover if and only if G has a k-vertex cover. Recent developments have focused on establishing linear problem kernels, i.e., $|V'| \leq 2k$. A theorem of Nemhauser and Trotter [30] proved that there exists an optimal solution to the linear programming relaxation of VERTEX COVER, such that the value returned by the objective function of the linear programming problem is a lower bound on the objective function of the related integer programming problem (cf. [27, 29]). The LP-relaxation of the vertex cover instance can be solved by finding an optimal vertex cover in a bipartite graph derived from the original graph. This cover can be determined by converting the bipartite graph into a flow network, and finding the minimum cut using Dinic's maximum flow algorithm with complexity $O(m\sqrt{n})$ [15]. As the number of edges is $O(n^2)$, this yields an overall complexity of $O(n^{5/2})$. Chen, Kanj, and Jia [12] showed this could be used in a parameterized setting to derive a linear problem kernel of size $2k$ for an instance of k-VERTEX COVER. A recent technique known as *crown reduction* [1] generates independent sets with specific properties, using randomly chosen maximum matchings. The maximum matching problem on the underlying bipartite graph can be recast as a network flow problem, and solved efficiently using Dinic's algorithm.

For the bounded tree search phase, the focus has been on formulating branching rules based on identifying small local vertex-adjacency patterns, and inferring the possible local minimum vertex covers for these patterns. Balasubramanian, Fellows, and Raman [6] describe two algorithms that use non-trivial branching rules to reduce the complexity of the tree search phase, with overall complexities of $O(kn + (\sqrt{3})^k k^2)$ and $O(kn + 1.324718^k k^2)$, respectively. A subsequent algorithm of Downey, Fellows, and Stege [22] improved this complexity to $O(kn + 1.31951^k k^2)$. Niedermeier and Rossmanith [31] extend the approach introduced by Balasubramanian, et al. where the goal of their search tree phase is to concentrate on processing vertices of small degree, i.e., 2, 3, 4, 5, to develop an algorithm with complexity $O(kn + 1.29175^k)$. A subsequent algorithm by Stege and Fellows [36] improves this result slightly to $O(kn + \max\{1.25542^k k^2, 1.2906^k 2.5k\})$. The current best result is that of Chen, et al. [12]. They describe an algorithm for graphs of arbitrary degree with complexity $O(kn + 1.2852^k)$. When the graph has small degree, i.e., bounded by 3 or bounded by 4, they developed algorithms with complexities of $O(kn + 1.273^k)$ and $O(kn + 1.277^k)$, respectively.

1.2 FPT algorithms and separators

Baker [5] exploited the outerplanarity property of planar graphs to develop approximation algorithms for the MAXIMUM INDEPENDENT SET problem. Alber, Bodlaender, Fernau, and Niedermeier [2, 3, 4] extended the work of Baker to develop an FPT algorithm for the k-DOMINATING SET problem with complexity $O(3^{6\sqrt{34k}}n)$. In a similar manner they propose an algorithm that finds a k-vertex cover in a planar graph with complexity $O(2^{4\sqrt{3}\sqrt{k}}n)$. Demaine, Hajiaghayi, and Thilikos [13] extended the result of Alber, et al. to non-planar graph classes: they prove that for a single-crossing graph H, the treewidth of any H-minor-free graph G having a k-dominating set is bounded by $O(\sqrt{k})$, leading to an algorithm with complexity $O(3^{6\sqrt{34k}}n^{O(1)})$. In contrast to previous work, Fomin and Thilikos [24, 25] base their algorithm on branchwidth rather than treewidth. They show that any n-vertex planar graph, G, has a branchwidth at most $2.122\sqrt{n}$ and treewidth at most $3.182\sqrt{n}$. Their algorithm for planar vertex cover has complexity $O(2^{4.5\sqrt{k}}k + k^4 + kn)$. Finally, Demaine, Fomin, Hajiaghayi, and Thilikos [14] introduced a framework for designing fixed-parameter algorithms of complexity $O(2^{O(\sqrt{k})}n^{O(1)})$ that applies to classes of graphs excluding a fixed graph H as a minor.

Unlike planar graphs or graphs of bounded genus, there exist no good separator theorems for graphs of bounded degree. Disconnecting a small, fixed-size piece of a graph is difficult to do [23], and the problem of determining the genus of a graph is NP-complete, even when restricted to cubic graphs [37, 38].

1.3 Parameterized graph cutting

We now turn our attention to a more general problem of disconnecting an input graph into some fixed number of components by removing a subset of edges. The decision version of the GRAPH k-CUT problem asks whether it is possible, given an edge-weighted graph, $G = (V, E)$, and integers k and m, to determine a set of edges of weight at most m, whose removal disconnects the graph into at least k connected components. The case where $k = 2$ is solvable in polynomial time using a maximum-flow/minimum-cut algorithm. For arbitrary k there is a hardness proof that reduces the CLIQUE problem to the GRAPH k-CUT problem [26]. Given this result, one may ask whether the parameterized version of this problem is fixed-parameter tractable, that is where the number of components is fixed for some k. Downey, Estivill-Castro, Fellows, Prieto, and Rosamond et al. [16] prove that the GRAPH k-CUT problem is hard for $W[1]$, using a reduction from k-CLIQUE. The authors also prove that the CUTTING k VERTICES FROM A GRAPH problem, that asks whether there is an edge set with cost at most m whose removal disconnects exactly k vertices, is $W[1]$-hard.

1.4 *k*-Vertex Cover and graphs of small degree

Efficient algorithms for processing graphs of small bounded degree (e.g., 3 or 4) serve two important purposes: firstly, to provide algorithms of better complexity for input instances that are graphs of bounded degree; secondly, as all k-vertex cover algorithms for general graph instances reduce the original graphs to ones of bounded degree, we have efficient subroutines that can be used during these phases of processing. In what follows, we describe a new algorithm for solving k-vertex cover based on recognizing the benefit of disconnecting a bounded-degree graph and explicitly processing components. The algorithm attempts to find small separators very quickly, and process them as efficiently as possible. There are notable characteristics to this approach:

- Unlike the separator-based approaches of Alber, et al. and Demaine, et al. our algorithm is not restricted to planar or bounded-genus graphs, and does not require the components be approximately equal in size.
- Our algorithm differs also in that we are using separators merely as a means of inducing components in the subproblem, within as few steps as possible. We avoid the complex dynamic programming phase incurred by the algorithm of Alber, et al. and Demaine, et al. while incorporating the existing and asymptotically effective branching rules. The space complexity is linear in the depth of the search tree, as the tables required for dynamic programming are not required.

As will be shown, the strategic branching technique incurs no additional polynomial cost: the complexity of calculating the strategic vertices is bounded by the complexity of the kernelization algorithms and so the general interleaving technique (cf.([32])) and its analytical implications still apply.

2. ALGORITHM FOR *K*-VERTEX COVER IN GRAPHS WITH SMALL DEGREE

Conventional k-vertex cover branching rules for general graphs are applied obliviously, with emphasis on facilitating

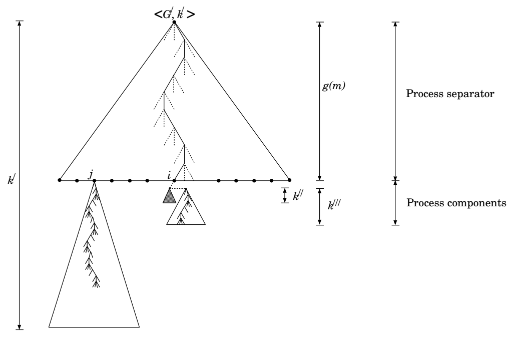


Figure 1: Tree-search phase, contrasting the exploration in the original algorithm (node j) and the improved algorithm (node i: subproblem consists of a disconnected graph).

rule development and complexity analysis. We introduce the concept of *strategic branching* in which specific vertices are processed with the intent of affecting a particular state on the remaining graph after a certain number of steps. Consider a problem instance $\langle G' = (V', E'), k' \rangle$, where we wish to determine if G' has a vertex cover of size at most k'. The objective is to process a set of vertices, $S \subseteq V'$, $m = |S|$, such that after applying branching rules on each $s \in S$, the subproblems at the intermediate levels of the search tree consist of $\langle G_i'', k_i' - g(m) \rangle$, where G_i'' comprise connected components of non-trivial size. In the next section we sketch how this set of vertices is found. Once the subproblem has been reduced to a graph consisting of two or more components, these components can be processed explicitly: as soon as the graph becomes disconnected, the smallest of the two components is solved exhaustively, wherein we determine k'' cover vertices. The remaining component is processed using the bound $k''' = k' - g(m) - k''$. (For example, see Figure 1.) We wish to ensure the branching rules used to process the separator are as efficient as possible, and where possible, remain within the complexity of the underlying algorithm. The algorithm consists of the following steps:

1. Perform kernelization on the problem instance $\langle G = (V, E), k \rangle$.
2. The tree search comprises two distinct phases:
   (a) generate a binary search tree by processing all vertices where $\deg(v) \geq 5$;
   (b) let $\langle G_i' = (V_i', E_i'), k_i' \rangle$ be an instance at some depth of the search tree, where $\forall v \in V_i'$, $\deg(v) \leq 4$:
      i. determine a set of strategic vertices, $S_i \subseteq V_i'$;
      ii. for each $s \in S_i$, apply the search tree branching rules to s until $S_i = \emptyset$; after this step, the remaining problem instance G_i'' will consist of components $C_1, \ldots, C_j$;
      iii. find minimum cover in $C_1, \ldots, C_{j-1}$; process C_j using bounded search.
3. If at any point in Step 2(a), $k_i' = 0$ and $E_i' \neq \emptyset$ (or accordingly, in Step 2(b) that $k_i'' = 0$ and $E_i'' \neq \emptyset$) then backtrack.

2.1 Step 2(b)i: Determining strategic vertices

The separation strategy will focus on a distinct phase of the graph processing, that being when the graph consists of vertices such that $\forall v \in V'$, $1 \leq \deg(v) \leq 4$. As it can be expensive to compute separators, we propose to use the fast network flow algorithm crucial to the kernelization routines described in Section 1.1. Bui, Chaudhuri, Leighton, and Sipser [8] presented an algorithm that finds a minimum bisection in a random regular graph, G, under the assumption that the bisector of G small, by reformulating it as a network flow problem (cf. [35]). This approach was motivated by the fact that graphs that occur in practice rarely have a bisection width as large as the theoretical upper bound.

In our case, it is not necessary to achieve a split that generates a bisection in order to improve the algorithm complexity. Our algorithmic objective is to disconnect one (or more) small piece(s) of the bounded degree graph, using small separators. We will assume the pieces contain $\Omega(\sqrt{2k})$ vertices and the separator has size $o(\sqrt{k})$, hence $O(\sqrt{k})$ cover vertices are required to process the separator. We note that the complexity is significantly improved if the smaller component, disconnected by the separator, is larger than $\sqrt{2k}$.

Minimum cuts and separators

We now consider the technique described by Bui, et al. for finding a minimum bisection in an unweighted graph with small bounded degree, based on a flow-network algorithm. The main idea is to convert the graph into a flow network and then use minimum (s,t)-cuts to determine a set of edges that represent the bisection of the input graph. Because finding a minimum bisection in a graph is NP-hard, the claim is that the algorithm finds such a cut most of the time, for a specific class of graphs known to have a small bisection width. For example, consider a class of graphs, $G(n, d, b)$, that are n-node, d-regular graphs with a unique minimum bisector of size b. The main result of [8] states that, given $d \geq 3$ and $b(n) = o(n^{1-1/\lfloor d+1/2 \rfloor})$, the algorithm determines a unique minimum bisector for almost all graphs in $G(n, d, b)$, in polynomial time. It fails to find a minimum bisection when there is a minimum cut smaller than b that is not a bisection.

We can convert an undirected graph, $G = (V, E)$, into a flow network as follows: replace each edge, $(u, v) \in E$, with a pair of directed edges, (u, v) and (v, u); designate a source vertex s and a sink vertex t, $s, t \in V$; give each edge a symmetric capacity: for $u, v \in V$, $c(u, v) = c(v, u)$. If each edge is given unit capacity then a minimum (s,t)-cut algorithm will find the smallest set of edges that disconnects s from t. A graph where the degree is bound by some small constant, d, will have a minimum cut that never exceeds some constant factor of d. To adapt a network flow algorithm to the separator problem requires modifying the input graph. We consolidate the source vertex, s, and the sink vertex, t, by contracting their respective neighborhoods to some distance, $p \geq 0$, and replacing them with two new vertices, s' and t'. Contracting a neighborhood around vertex s consists of a series of edge contractions where we replace vertex s with s', and for all edges (s, x), we delete vertex x and replace any edge (x, y), $y \neq s$, with (s', y). (Similarly for vertex t.) Let $D(s, p)$ represent the set of vertices collapsed into s', the farthest vertex being at distance at most p from s. Given that $|D(s, p)| < d^{p+1}$, the number of edges connecting

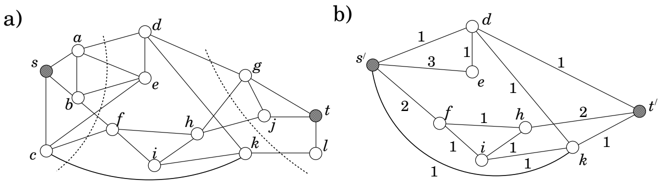


Figure 2: Converting a bounded degree graph into a flow network (for simplicity, edges represent opposing arcs): (a) graph with vertices s, t; (b) corresponding flow network, with $p = 1$, $D(s, p) = \{a, b, c\}$, $D(t, p) = \{g, j, l\}$.

$D(s, p)$ with $V \setminus D(s, p)$ is at most d^{p+1}.

The capacities of edges connecting vertices in $D(s, p)$ and $V \setminus D(s, p)$ are determined as follows: let $v \notin D(s, p)$; an edge (s', v) is assigned capacity $|(u, v) \in E|$, $\forall u \in D(s, p)$. (Similarly for t' and $D(t, p)$.) All other edges in the graph are assigned a unit value capacity. (Similarly for t' and $D(t, p)$.) Finally, an edge (s', t') is added with capacity $|(u, v) \in E|$, for all u, v where $u \in D(s, p)$ and $v \in D(t, p)$. For example, see Figure 2.

The algorithm to find a minimum bisection consists of the following steps: for all pairs of vertices, $s, t \in V$, $s \neq t$, and s, t are least distance of $2p + 1$ apart, construct a consolidated graph, G'; run a minimum (s, t)-cut on the graph G', reconsolidate the graph, and extract the minimum cut. The algorithm as stated solves $O(n^2)$ maximum-flow minimum-cut problems and has polynomial complexity.

Improved cutting strategy

It is essential that the separator be sufficiently small, and that the small component(s) be of some nontrivial size. We propose the following algorithm as an means of reducing the complexity of finding a satisfactory minimum cut while guaranteeing an advantageous size for small components. We repeat the following process some fixed number of times: randomly designate some vertex to be source vertex, s, and a corresponding maximally distant sink vertex, t; we consolidate s and t by contracting their respective neighborhoods to some distance, $p \geq 0$, while ensuring $D(s', p) = D(t', p) = \sqrt{2k}$, and then replacing them with two new vertices, s' and t'. We then use Dinic's algorithm to determine a minimum (s, t)-cut in complexity $O(k^{3/2})$ rather than the more expensive minimum cut calculation.

We note that most algorithms during the search tree phase assume an interleaving of branching and kernelization steps. Generating the separator set for strategic branching is thus asymptotically no more expensive than in-tree kernelization (and recall from Section 1.1 that Dinic's algorithm is fundamental to two state-of-the-art kernelization routines).

2.2 Step 2(b)ii: Processing the separator set

Unlike the approaches described in Section 1.2 that use dynamic programming on tree decompositions, we will adapt existing branching rules for processing the vertices that constitute the separator set. The advantage of this approach is that any state-of-the-art algorithm can be adapted and we can avoid the need for a complex dynamic programming solution. In what follows, we will consider two underlying k-vertex cover algorithms: that of Balasubramanian, et al. [6], and that of Chen, et al. [12]. We first consider the algorithm of Balasubramanian, et al. with complexity $O(kn + 1.324718^k)$. (See Appendix A for details.)

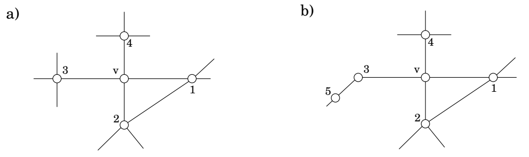


Figure 3: Branching cases for Rule 5(a).

Branching rule complexity

Branching rules are derived by considering combinations of subsets of vertices that could form a vertex cover in a small subgraph. The order in which rules are applied determines the complexity bound as the rules assume vertices in the subgraph have some known degree: the algorithms generally process all vertices of degree δ using rule set $\mathcal{X}$, before attempting to process vertices of degree γ using rule set $\mathcal{X}+1$, where the latter set is constructed assuming the graph contains no vertices of degree δ. Once a rule is applied, a recurrence relation describes the size of the remaining search tree in terms of the new (reduced) parameter. In most cases the rules cannot be applied in mixed fashion. For example, consider Figure 3a. Inspecting the subgraph around vertex v prescribes the following possible subsets of vertices to cover the edges: $\{1, 2, 3, 4\} \subseteq VC$, or $N(3) \subseteq VC$, or $N(4) \cup \{3\} \subseteq VC$. Assuming vertex 3 has degree 4, the recurrence describing the size of the underlying search tree given these three branching choices is $C(k) \leq 1 + 3C(k-4)$, which is bounded by 1.316074^k and thus falls within the complexity 1.324718^k claimed for the algorithm of Balasubramanian, et al. Now consider the case where vertex 3 has degree 2, as in Figure 3b. If we apply the rule as before, we have the recurrence $C(k) \leq 1 + C(k-4) + C(k-2) + C(k-4)$, which is bounded by 1.414214^k and clearly the claim no longer holds. This example highlights two important facts. First, special consideration must be paid to applying branching rules to separator vertices so as to remain within the original complexity bound of the algorithm. Second, given a separator vertex, v, adjacent to a vertex, x, where x is degree 2, it is always the case that we can process the subgraph around v, using Rule (2), (2)a, or (2)b on vertex x, within the stated complexity.

Any time two cut-set edges have a common endpoint, we will designate the common vertex as the separator. The idea behind our processing will be to shift the role of separator vertex (as necessary) between vertices in a subgraph so as to enable us to invoke branching rules that remove the cut-set edge(s).

We will first present a theorem that establishes that we can process vertices in a separator using the branching and reduction rules, outlined in Appendix A, within the same complexity as the underlying algorithm of Balasubramanian, et al.

THEOREM 1. *Given a graph, $G = (V, E)$, $\forall v \in V, 1 \leq \deg(v) \leq 4$, and an integer parameter k, the branching and reduction rules of the k-vertex cover algorithm of Balasubramanian, Fellows, and Raman [6] can be applied in any order to process some subset of vertices, $S \subseteq V$, $|S| \leq k$, to*

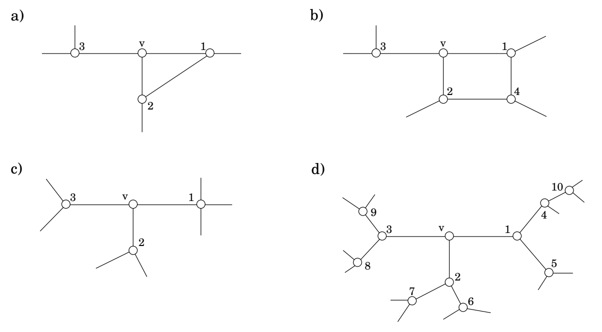


Figure 4: Degree-3 separator vertex branching rules.

find a partial vertex cover of G, without affecting the overall complexity $O(1.324718^k)$ for the algorithm.

Proof. By inspection it is clear that branching Rules (1), (2), (2)a, (2)b, (3)[1], (4)b, (4)c, (4)d, (5)b can be applied without restriction. As we will see, the case of degree-3 and degree-4 separator vertices require closer analysis. In what follows, we can assume the vertices neighboring a separator vertex are of degree 3 or degree 4, as any degree-2 vertex can be processed directly within the complexity bound. We also assume that kernelization rules are interleaved with search tree branching rules, so that there are no vertices of degree 1.

We will show the branching rules either apply directly to processing the subgraph containing a separator vertex (and the cut edges), or that one (or two) rules can be applied to vertices adjacent to the separator resulting in its removal, while remaining within the complexity bound of the algorithm.

Branching rules for degree-3 separator vertices

Most of the degree-3 vertex branching rules apply directly but we include their analysis for completeness.

Rule (4)a: (separator) vertex v is of degree 3, $N(v) = \{1, 2, 3\}$, all vertices adjacent to v have degree 3, and there exists an edge between two neighboring vertices, e.g., $(1, 2)$. Then $\{1, 2, 3\} \subseteq VC$ or $N(3) \subseteq VC$, yielding a recurrence of $C(k) \leq 1 + 2C(k - 3)$. This rule applies directly. (See Figure 4a.) As stated before, if it were the case that at least one of $\{1, 2\}$ was a degree-2 vertex, e.g., vertex 1, then reduction Rule (2)b applies directly, with $\{v, 1\} \subseteq VC$.

Rule (4)b: (separator) vertex v is of degree 3, $N(v) = \{1, 2, 3\}$, all vertices adjacent to v have degree 3, and there is a common neighbor, vertex 4, between some pair of vertices, e.g., vertices 1 and 2. Then $\{1, 2, 3\} \subseteq VC$ or $\{v, 4\} \subseteq VC$, yielding a recurrence of $C(k) \leq 1 + C(k - 2) + C(k - 3)$. This rule applies directly. (See Figure 4b.) If it were the case that all of $\{1, 2, 4\}$ were degree 2, then reduction Rule (2)b applies directly, with $\{v, 4\} \subseteq VC$.

Rule (4)c: (separator) vertex v is of degree 3, $N(v) = \{1, 2, 3\}$, and there are no edges between vertices $\{1, 2, 3\}$ and at least one vertex, e.g., vertex 1, has degree 4. Note that $\{1, 2, 3\}$ have unique neighbors, otherwise we would invoke Rule (4)a or Rule (4)b. Also, vertices $\{1, 2\}$ are at least degree 3, otherwise they would have been processed using Rule (2)a. Then $\{1, 2, 3\} \subseteq VC$ or $N(1) \subseteq VC$ or

[1]For our purposes this rule is never invoked as the graph is assumed to be bounded in degree by 4.

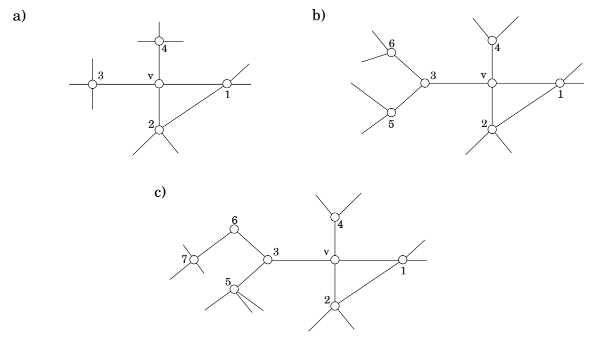


Figure 5: Special cases for branching Rule (5)a.

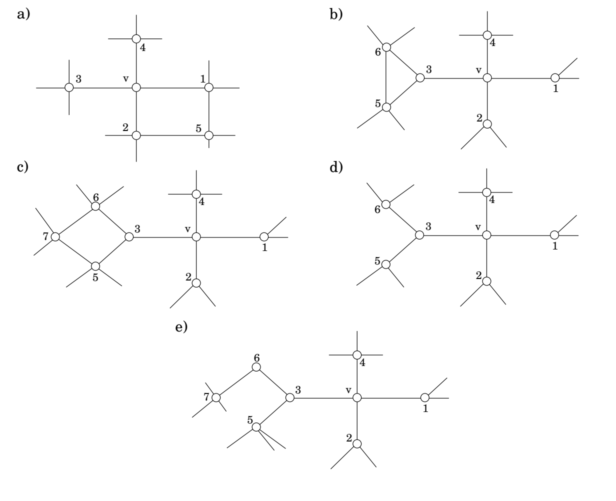


Figure 6: Special cases for branching Rule (5)c.

$N(\{2, 3\}) \cup \{1\} \subseteq VC$, yielding a recurrence of $C(k) \leq 1 + C(k - 3) + C(k - 4) + C(k - 6)$. This rule applies directly. (See Figure 4c.)

Rule (4)d: (separator) vertex v is of degree 3, $N(v) = \{1, 2, 3\}$, and there are no edges between vertices $\{1, 2, 3\}$ and each of the latter have at least two unique neighbors. Some vertex, e.g., vertex 1, has a neighboring vertex 4, that is adjacent to some vertex $10 \notin N(N(1) \cup N(2) \cup N(3))$. Then $\{1, 2, 3\} \subseteq VC$ or $\{v, 4, 5\} \subseteq VC$ or $N(\{2, 3, 4, 5\}) \subseteq VC$, yielding a recurrence of $C(k) \leq 1 + 2C(k-3) + C(k-7)$. This rule applies directly. (See Figure 4d.) Because the rule assumes only that one of the private neighbors has a unique vertex 10 (i.e., 4), we claim at most $|N(\{2, 3, 4, 5\})| = 7$.

Branching rules for degree-4 separator vertices

Rule (5)a: (separator) vertex v is of degree 4, $N(v) = \{1, 2, 3, 4\}$, and there exists an edge between two neighboring vertices, e.g., $(1, 2)$. Then $\{1, 2, 3, 4\} \subseteq VC$ or $N(3) \subseteq VC$ or $N(4) \cup \{3\} \subseteq VC$, yielding a recurrence of $C(k) \leq 1 + 3C(k - 4)$. If vertices $\{3, 4\}$ are degree 4 then Rule (5)a applies directly. (See Figure 5a.) If one of vertices $\{3, 4\}$ is degree 2, e.g., vertex 3, then process this subgraph using Rule (2)a with $v = 3$.

We now consider the case where one (or both) of vertices $\{3, 4\}$ have degree 3. Assume vertex 3 is of degree 3 and let $N(3) = \{5, 6, v\}$. If the degrees of neighboring vertices $\{5, 6\}$ are 3 then process the subgraph using Rule (4)a, Rule

(4)b, or Rule (4)c, with $v = 3$. (See Figure 5b.)

Assume the degree of one of $\{5, 6\}$ is 2, e.g., vertex 6, and let $N(6) = \{3, 7\}$ If there is an edge $(5, 7)$, we apply Rule (4)b, with $v = 3$. Then $\{3, 7\} \subseteq VC$ or $\{v, 5, 6\} \subseteq VC$, yielding a recurrence of $C(k) \leq 1 + C(k - 2) + C(k - 3)$ (1.324718^k). (See Figure 5c.)

Otherwise we apply Rule (2)a, with $v = 6$, to branch with $\{3, 7\} \subseteq VC$ or $N(\{3, 7\}) \subseteq VC$, yielding a recurrence of $C(k) \leq 1 + C(k-2) + C(k-4)$. Note that the subproblem generated by the branch $\{3, 7\}$ could require further processing if the cut-set edge in the subgraph was $(2, v)$, $(1, v)$, or $(4, v)$. In this case, we simply apply Rule (4)a directly.

Rule (5)c: (separator) vertex v is of degree 4, $N(v) = \{1, 2, 3, 4\}$, and there are no edges between vertices $\{1, 2, 3, 4\}$ and at most one pair of vertices, e.g., $\{1, 2\}$, are adjacent to some common vertex 5 (or possibly no vertex is shared). Then $\{1, 2, 3, 4\} \subseteq VC$ or $N(3) \subseteq VC$ or $N(4) \cup \{3\} \subseteq VC$ or $N(\{1, 2\}) \cup \{3, 4\} \subseteq VC$, yielding a recurrence of $C(k) \leq 1 + 2C(k - 4) + C(k - 5) + C(k - 8)$. If the degrees of vertices $\{1, 2, 3, 4\}$ are 4 then Rule(5)c applies directly. (See Figure 6a.) Consider the case where one of $\{1, 2\}$ is degree 3, e.g., vertex 1. We apply Rule (4)b, with $v = 1$.

Otherwise, at least one of $\{1, 2, 3, 4\}$ is a degree-3 vertex. Assume this to be vertex 3 and let $N(3) = \{5, 6, v\}$. We assume vertices $\{5, 6\}$ to be at least degree 3. If there is an edge $(5, 6)$, then we apply Rule (4)a, with $v = 3$. (See Figure 6b.)

If vertices $\{5, 6\}$ have a common vertex 7 as a neighbor, then we apply Rule (4)b, with $v = 3$. We then apply Rule (4)a, (4)b, or (4)c to the remaining subgraph. (See Figure 6c.)

If vertices $\{5, 6\}$ have no common neighbor and there is no edge between them then we apply Rule (4)c, with $v = 3$. (See Figure 6d.)

We now consider the case where one of vertices $\{5, 6\}$ has degree 2. Assume this is vertex 6 and $N(6) = \{3, 7\}$. We apply Rule (2)a, with $v = 6$, and then apply Rule (4)b or Rule(4)c to process the remaining subgraph. (See Figure 6e.) ☐

2.3 Asymptotic analysis

We now consider an asymptotic analysis of the improved k-vertex cover algorithm in graphs with degree bounded by 4.

THEOREM 2. *Consider a graph, $G = (V, E)$, and integer parameter k, where $\forall v \in V$, $1 \leq \deg(v) \leq 4$. Using the strategic branching scheme and the adapted branching rules, let $S \subseteq V$ represent a separator determined by the algorithm in Section 2.1, and let $\langle G' = (C_1 \cup C_2), k - g(m)\rangle$ be a subproblem, consisting of two connected components at some intermediate node of the search tree, derived from applying at most $m = |S|$ branching or reduction rules to the separator. Let $n_{C_1} = |V(C_1)|$, $n_{C_2} = |V(C_2)|$, and without loss of generality we assume that $n_{C_1} \leq n_{C_2}$.*

We can determine if G has a vertex cover of size bounded by k in time $O(kn + c_1^m(c_2^{k'} + c_2^{k-g(m)-k'}))$, where c_1, c_2 are constants, $k' = (3(n_{C_1}) + 1)/4$, m is the separator size, and $g(m) \leq k$ is a function of the number of cover vertices found after m applications of the branching and kernelization rules.

PROOF. From Theorem 1 and the algorithm of Balasubramanian, et al. ☐

The expression ${c_1}^m$ describes the number of leaves of the intermediate search tree generated after m branching rules have been applied. The constant c_2 can be 1.324718 or 1.277 for example, depending on the underlying algorithm. The value $(3(n_{C_1}) + 1)/4$ represents a worst case bound on the cover sought for component C_1. This result comes from approximating a maximum independent set of size $\lceil (n-1)/d \rceil$ for a graph G of bounded degree, d, by repeatedly adding the vertex with the smallest degree to the independent set. Because the branching rules can reduce the parameter by anywhere between 2 and 8, the function $g(m)$ bounds the parameter reduction after m applications of the rules.

Consider the behavior of the Balasubramanian, et al. algorithm in comparison with the algorithm with strategic branching. The first phase is identical for both algorithms, in that they both generate intermediate search trees of identical asymptotic sizes and the expected value of the parameters at the leaves will be similar. For the original algorithm each leaf of this intermediate tree consists of subproblems that will generate bounded trees with $O(c_2^{k-g(m)})$ nodes. The algorithm that processed strategic vertices will have instances at the leaves that will generate bounded trees of size $O(c_2^{k'} + c_2^{k-g(m)-k'})$, where k' depends on the split determined by the separation phase. In the case that the separator is large, the strategic branching algorithm has complexity equal to that of the conventional algorithm.

2.4 Performance analysis

In this section we analyze the impact of combining strategic branching with the state-of-the-art algorithm of Chen, et al. [12]. In practice it is often the case that FTP algorithms perform much better than a worst case analysis would indicate and so we will focus on the expected performance. We use the following formula (based on Theorem 2) to model the search tree phase of the strategic branching algorithm:

$$c_1^m \left({c_2}^{(3(\sqrt{2k})+1)/4} + {c_2}^{k-E[g(m)]-(3(\sqrt{2k})+1)/4} \right) \qquad (1)$$

where $m = \sqrt{k}$, $c_2 = 1.277$, and $\sqrt{2k}$ is the minimum size of the smallest component. Our results are independent of the kernel size so we can assume it is linear and hence $|V| \leq 2k$.

We now turn to the question of determining the values for c_1 and $g(m)$. At a given node in the tree any of the branching rules from Appendix A may be invoked during separator processing resulting in the creation of 2, 3, or 4 children. We assume a worst case branching of 4 children (Rule (5)c), hence $c_1 = 4$, leading to a tree with at most 4^m leaves. To determine the expected value of k_i at some leaf i of the search tree, i.e., $k - E[g(m)]$, we consider a path from the root of the tree to node i and the associated decremental values on each edge in the path. The sample space consists of the set of values $\{4, 5, 8\}$ and we define a random variable K_j, $1 \leq j \leq m$, to be the number of vertices added to the cover at some step j of the algorithm. The probability distribution is given by,

$$\Pr[K_j = 4] = 1/2,$$
$$\Pr[K_j = 5] = 1/4,$$
$$\Pr[K_j = 8] = 1/4.$$

The expected number of vertices added to the cover at a given node in the tree is,

$$E[K_j] = 4(1/2) + 5(1/4) + 8(1/4) = 5.25,$$

and so,

$$\sum_{j=1}^{m} E[K_j] = 5.25m,$$

Algorithm	Complexity	Klam value
Balasubramanian, et al. [6]	1.324718^k	163
Downey, et al. [22]	1.31951^k	166
Niedermeier, Rossmanith [31]	1.29175^k	179
Stege, Fellows [36]	1.2906^k	180
Chen, et al. [12]	1.2852^k	183
Chen, et al. [12]*	1.277^k	188

Table 1: Summary of complexities k-vertex cover algorithms. (*: restricted to graphs of degree bounded by 4).

		Number of components			
		2	3	4	5
Value for $k_{\max}$	200	1.263^k	1.251^k	—	—
	250	1.265^k	1.252^k	—	—
	500	1.268^k	1.259^k	1.250^k	—
	$1,000$	1.271^k	1.265^k	1.258^k	1.252^k
	$2,000$	1.273^k	1.269^k	1.264^k	1.260^k
	$4,000$	1.274^k	1.271^k	1.268^k	1.265^k

Table 2: Complexity assuming various hypothetical splits for k-vertex cover.

i.e., the expected remaining cover-budget for the subproblem at leaf i is $k - 5.25m$.

The complexity of Equation 1 converges to 1.277^k as $k \to \infty$ because the effect of cutting off and processing a small piece of the graph contributes little when k grows very large. Nonetheless, closer analysis shows the impact on the complexity is in fact more relevant within a practical range of values for k. In our analysis we fix k within a small range, specifically $200 \leq k \leq 4,000$, and study the effect on the complexity assuming separators generate two or more components.

As an initial comparison, Table 1 contains the complexities for various k-vertex cover algorithms. Let $T_S(k)$ denote the running for the strategic branching algorithm with parameter value k. For a fixed value of $k_{\max}$, we derive an exponential bound on $T_S(k)$ by determining some expression $T(k)$ such that $T_S(k)/T(k) \approx 1$, i.e., for all $k \leq k_{\max}$ the strategic branching algorithm is bounded by $T(k)$. Table 2 shows the overall complexity of Equation 1 in terms of $T(k)$, assuming various separator size and split combinations. Cases where the entry is missing indicate that the separator would too large and processing completely solves the instance.

Having examined the table of complexity values, two items are worthy of mention. First, in most cases the increase in the complexity for the overall algorithm is potentially quite dramatic. Each splitting scenario represents an improvement over the complexity of 1.277^k of the original algorithm. Secondly, we note that in case the projected complexity would be worse than the original algorithm, due to the absence of sufficiently small separators, we can merely process the graph as per the original algorithm, i.e., with no special consideration for components.

3. ALGORITHM FOR *K*-VERTEX COVER PROBLEM IN GENERAL GRAPHS

We can improve the complexity of the algorithm for processing general graphs using the strategic branching approach for accelerating processing of graphs of degree bounded by 4. In the case of general graphs, branching rules systematically remove vertices of high degree, i.e., degree ≥ 5. Indeed, this is a step common to the current best k-vertex cover algorithms. Once all vertices of large degree have been removed, the improved algorithm for graphs of degree bounded by 4 can be used as a subroutine for processing remaining subproblems in the lower branches of the search tree. As noted earlier, a search tree has most of its nodes in the lower branches, so this in fact will significantly improve the complexity of the algorithm for general graphs.

4. ALGORITHM FOR MAXIMUM INDEPENDENT SET PROBLEM FOR GRAPHS OF SMALL DEGREE

The result from Section 2 has a surprising corollary. Although k-INDEPENDENT SET $\in W[1]$, Chen, et al. [12] propose using their k-vertex cover algorithm to find a maximum independent set in a graph of degree bounded by 4. For several years the best result for this problem was due to Robson [33] with complexity $O(2^{0.296n}) \approx O(1.227^n)$ restricted to graphs of maximum degree 4. The complexity for solving the problem using the k-vertex cover algorithm of Chen, et al. is $O(1.277^{3/4n} \approx 1.201^n)$. Beigel [7] subsequently presented an algorithm with complexity $O(1.171^n)$ for graphs of maximum degree 4, and Robson [34] recently published an algorithm for general graphs with complexity $O(1.189^n)$.

Table 3 summarizes the complexities of the k-vertex cover algorithm with strategic branching (Table 2) when applied to the MAXIMUM INDEPENDENT SET problem in graphs of degree bounded by 4, assuming various split combinations. In all cases, the strategic branching is an improvement over the results of Chen, et al. and in some cases is an improvement over the recent result of Robson.

5. CONCLUSION

We describe a new approach to improve algorithms for solving the k-VERTEX COVER problem. The algorithm uses

	Number of components			
Value for n	2	3	4	5
200	1.191^n	1.183^n	—	—
250	1.193^n	1.184^n	—	—
500	1.195^n	1.189^n	1.182^n	—
$1,000$	1.197^n	1.193^n	1.188^n	1.184^n
$2,000$	1.198^n	1.196^n	1.192^n	1.189^n
$4,000$	1.199^n	1.197^n	1.195^n	1.193^n

Table 3: Complexity assuming various hypothetical splits for maximum independent set.

graph cutting during the tree search phase as a basis for disconnecting and explicitly processing small fractions of the residual graph. Unlike previous methods that use separator theorems for restricted graph classes and requiring complex dynamic programming, this algorithm applies to graphs of small bounded degree, adapts existing branching rules, and incurs no additional complexity. Such graph instances are generated during the tree search phases of the current best FPT k-vertex cover algorithms. Any future advances in sequential algorithms for the k-Vertex Cover problem can incorporate strategic branching, thus further improving their complexity.

We also investigated the applicability of our new algorithm to solving the Maximum Independent Set problem in graphs of small bounded degree. Depending on the connected components that are generated, the new k-vertex cover algorithm has complexity comparable to the current best algorithm for the Maximum Independent Set problem.

6. ACKNOWLEDGMENTS

The author wishes to thank Mike Fellows, Pat Morin, and Daniel Wevrick for their helpful discussion.

7. REFERENCES

[1] F.N. Abu-Khzam, R.L. Collins, M.R. Fellows, M.A. Langston, W.H. Suters, C.T. Symons. "Kernelization algorithms for the vertex cover problem: Theory and experiments". In *Proceedings of the ACM-SIAM Workshop on Algorithm Engineering and Experiments*, January, 2004.

[2] J. Alber, H. Fernau, R. Niedermeier. "Parameterized complexity: Exponential speed-up for planar graph problems". In *Proceedings of the 28th ICALP 2001*, LNCS 2076, pp.261–272, 2001.

[3] J. Alber, H. Fernau, R. Niedermeier. "Graph separators: A parameterized view". In *Proceedings of the 7th International Computing and Combinatorics Conference (COCOON 2001)*, LNCS 2108, pp.318–327, 2001.

[4] J. Alber, H.L. Bodlaender, H. Fernau, T. Kloks, R. Niedermeier. "Fixed-parameter algorithms for dominating set and related problems on planar graphs". In *Algorithmica*, Vol.33, pp.461–493, 2002.

[5] B.S. Baker. "Approximation algorithms for NP-complete problems on planar graphs". In *Journal of the ACM*, Vol.143, pp.153–180, 1994.

[6] R. Balasubramanian, M.R. Fellows, V. Raman. "An improved fixed-parameter algorithm for vertex cover". In *Information Processing Letters*, Vol.65, pp.163–168, 1998.

[7] R. Beigel. "Finding maximum independent sets in sparse and general graphs". In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA 1999)*, pp.856–857, 1999.

[8] T.N. Bui, S. Chaudhuri, F.T. Leighton, M. Sipser. "Graph bisection algorithms with good average case behavior". In *Combinatorica*, Vol.7, pp.171–191, 1987.

[9] J.F. Buss, J. Goldsmith. "Nondeterminism within P". In *SIAM Journal of Computing*, Vol.22, pp.560–572, 1993.

[10] L. Cai, J. Chen, R.G. Downey and M.R. Fellows. "On the parameterized complexity of short computation and factorization". In *Archive For Mathematical Logic*, Vol.36, pp. 321–337, 1997.

[11] M. Cesati. "The Turing way to parameterized complexity". In *Journal of Computer and System Sciences*, Vol.67, No.4, pp.654–685, 2003.

[12] J. Chen, I.A. Kanj, W. Jia. "Vertex cover: Further observations and further improvements". In *Proceedings of the 25th International Workshop on Graph-Theoretical Concepts in Computer Science (WG 1999)*, LNCS 1665, pp.313–324, 1999.

[13] E.D. Demaine, M. Hajiaghayi, D.M. Thilikos. "Exponential speedup of fixed-parameter algorithms on $K_{3,3}$-minor-free or K_5-minor-free graphs". In *Proceedings of the 13th Annual International Symposium on Algorithms and Computation (ISAAC 2002)*, LNCS 2581, pp. 262–273, 2002.

[14] E.D. Demaine, F.V. Fomin, M. Hajiaghayi, D.M. Thilikos. "Subexponential parameterized algorithms on graphs of bounded genus and H-minor-free graphs". In *Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2004)*, pp.830–839, 2004.

[15] E.A. Dinic. "Algorithm for solution of a problem of maximum flows in networks with power estimation". In *Soviet Math. Dokl.*, Vol.11, pp.1277–1280, 1970.

[16] R.G. Downey, V. Estivill-Castro, M.R. Fellows, E. Prieto, F.A. Rosamond. "Cutting up is hard to do: The parameterized complexity of k-cut and related problems". In *Computing the Australasian Theory Symposium (CATS 2003)*, Electronic Notes in Theoretical Computer Science, Elsevier, Vol.78, 2003.

[17] R.G. Downey, M.R. Fellows. "Fixed-parameter tractability and completeness." In *Congressus Numerantium*, Vol.87, pp.161–187, 1992.

[18] R.G. Downey, M.R. Fellows. "Parameterized computational feasibility". In *Feasible Mathematics II*, Clote, Remmel (Eds.), Birkhauser, pp.219–244, 1995.

[19] R.G. Downey, M.R. Fellows."Fixed parameter

tractability and completeness I: Basic theory". In *SIAM Journal of Computing*, Vol.24, pp.873–921, 1995.

[20] R.G. Downey, M.R. Fellows. "Fixed parameter tractability and completeness II: Completeness for $W[1]$". In *Theoretical Computer Science A*, Vol.141 , pp.109–131, 1995.

[21] R.G. Downey, M.R. Fellows. *Parameterized Complexity*. Springer-Verlag, 1998.

[22] R.G. Downey, M.R. Fellows, U. Stege. "Parameterized complexity: A framework for systematically confronting computational intractability". In *Contemporary Trends in Discrete Mathematics: From DIMACS and DIMATIA to the Future*, Roberts, Kratochvil, Nesetril (Eds.), *AMS-DIMACS Proceedings Series*, Vol.49, AMS, pp.49–99, 1999.

[23] U. Feige, R. Krauthgamer, K. Nissim. "On cutting a few vertices from a graph". In *Discrete Applied Mathematics*, Vol.127, No.3, pp.643–649, 2003.

[24] F.V. Fomin, D.M. Thilikos. "A simple and fast approach for solving problems on planar graphs". Technical Report 258, University of Bergen, November 2003.

[25] F.V. Fomin, D.M. Thilikos. "New upper bounds on the decomposability of planar graphs and fixed parameter algorithms". Technical Report 240, University of Bergen, November 2003.

[26] O. Goldschmidt, D.S. Hochbaum. "A polynomial algorithm for the k-cut problem for fixed k". In *Mathematics of Operations Research*, Vol.19, No.1, pp.24–37, 1994.

[27] D. Hochbaum. *Approximation Algorithms For NP-Hard Problems*. PWS, 1997.

[28] R. Karp. "Reducibility among combinatorial problems". In *Complexity of Computer Computations*, Mille, Thatcher (Eds.), Plenum Press, pp.85–104, 1972.

[29] S. Khuller. "The vertex cover problem". In *ACM SIGACT News*, Vol.33, pp.31–33, 2002.

[30] G.L. Nemhauser, L.E. Trotter. "Vertex packing: Structural properties and algorithms". In *Mathematical Programming*, Vol.8, pp.232–248, 1975.

[31] R. Niedermeier, P. Rossmanith. "Upper bounds for vertex cover further improved". In *Proceedings of the 16th Symposium on Theoretical Aspects in Computer Science (STACS 1999)*, LNCS, 1999.

[32] R. Niedermeier, P. Rossmanith. "A general method to speed up fixed-parameter-tractable algorithms". Technical Report TUM-I9913, Institut für Informatik, Technische Universität München, 1999.

[33] J.M. Robson. "Algorithms for maximum independent sets". In *Journal of Algorithms*, Vol.7, pp.425–440, 1986.

[34] J.M. Robson. "Finding a maximum independent set in time $O(2^{n/4})$". Technical Report 1251-01, LaBRI, Université Bordeaux I, 2001.

[35] A.L. Rosenberg, L.S. Heath. *Graph Separators, With Applications*. Kluwer Academic/Plenum Publishers, 2001.

[36] U. Stege, M.R. Fellows. "An improved fixed-parameter tractable algorithm for vertex cover". Technical Report 318, Department of Computer Science, ETH Zürich, April 1999.

[37] C. Thomassen. "The graph genus problem is *NP*-complete". In *Journal of Algorithms*, Vol.10, pp.568–576, 1989.

[38] C. Thomassen. "The genus problem for cubic graphs". In *Journal of Combinatorial Theory Ser. B*, Vol.69, pp.52–58, 1997.

Rule no. (cf. [6])		Branching vector	Branching number
(1)	[1]	(1)	N/A
(2)	[2]	(2)	N/A
(2)a	[3]	(2,3)	1.324717957
(2)b	[4]	(2)	N/A
(3)	[5]	(1,5)	1.324717957
(4)a	[6]	(3,3)	1.259921050
(4)b	[7]	(3,2)	1.324717957
(4)c	[8]	(3,4,6)	1.304077155
(4)d	[9]	(3,3,7)	1.324717957
(5)a	[10]	(4,4,4)	1.316074013
(5)b	[11]	(4,2)	1.27019650
(5)c	[12]	(4,4,5,8)	1.324717957

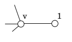
Table 4: Search tree cost summary.

APPENDIX

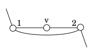
A. BALASUBRAMANIAN, FELLOWS, RAMAN ALGORITHM

Balasubramanian, et al. describe two algorithms in [6] that use non-trivial branching rules to reduce the complexity of the tree search phase. In this appendix we focus on the second algorithm (referred to as *Theorem 2*), which consists of a set of elaborate degree-based matching rules. The algorithm complexity is $O(kn + 1.324718^k)$, assuming interleaving as described in [22, 32].

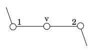
Rule (1): Degree 1
$\{v\} \subseteq VC$
$C(k) \leq 1 + C(k-1)$



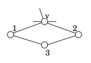
Rule (2): Degree 2
$\{1,2\} \subseteq VC$
$C(k) \leq 1 + C(k-2)$



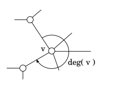
Subcase: Rule (2)a
$\{1,2\} \subseteq VC$
$N(\{1,2\}) \subseteq VC$
$C(k) \leq 1 + C(k-2) + C(k-3)$



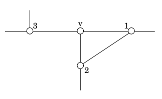
Subcase: Rule(2)b
$\{1,2\} \subseteq VC$
$C(k) \leq 1 + C(k-2)$



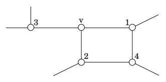
Rule (3): Degree ≥ 5
$\{v\} \subseteq VC$
$N(v) \subseteq VC$
$C(k) \leq 1 + C(k-1) + C(k-5)$



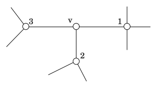
Rule (4): Degree 3, 4
Subcase: (4)a
$\{1,2,3\} \subseteq VC$
$N(3) \subseteq VC$
$C(k) \leq 1 + 2C(k-3)$



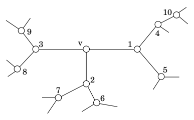
Subcase: (4)b
$\{v,4\} \subseteq VC$
$\{1,2,3\} \subseteq VC$
$C(k) \leq 1 + C(k-2) + C(k-3)$



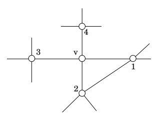
Subcase: (4)c
$\{1,2,3\} \subseteq VC$
$N(\{2,3\}) \cup \{1\} \subseteq VC$
$N(1) \subseteq VC$
$C(k) \leq 1 + C(k-3) + C(k-6) + C(k-4)$



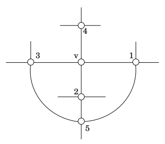
Subcase: (4)d
$\{1,2,3\} \subseteq VC$
$\{v,4,5\} \subseteq VC$
$N(\{2,3,4,5\}) \subseteq VC$
$C(k) \leq 1 + 2C(k-3) + C(k-7)$



Rule 5: Degree 4
Subcase: (5)a
$\{1,2,3,4\} \subseteq VC$
$N(3) \subseteq VC$
$N(4) \cup \{3\} \subseteq VC$
$C(k) \leq 1 + 3C(k-4)$



Subcase: (5)b
$\{1,2,3,4\} \subseteq VC$
$\{v,5\} \subseteq VC$
$C(k) \leq 1 + C(k-4) + C(k-2)$



Subcase: (5)c
$\{1,2,3,4\} \subseteq VC$
$N(3) \subseteq VC$
$N(4) \cup \{3\} \subseteq VC$
$N(\{1,2\}) \cup \{3,4\} \subseteq VC$
$C(k) \leq 1 + 2C(k-4) + C(k-5) + C(k-8)$