The Computational Complexity Column

by

Michal Koucký

Computer Science Institute, Charles University
Malostranské nám. 25, 118 00 Praha 1, Czech Republic
`koucky@iuuk.mff.cuni.cz`
`https://iuuk.mff.cuni.cz/~koucky/`

Seven Ways of Deciding Whether Most Vertex Sets Cover a Graph

Till Tantau
Institute for Theoretical Computer Science
Universität zu Lübeck, Lübeck, Germany
till.tantau@uni-luebeck.de

Abstract

Given an undirected graph $G = (V, E)$, it is a difficult problem (complete for #P) to determine the total number of vertex covers $C \subseteq V$ of G. In contrast, deciding whether *most* C cover a graph (meaning, at least half of all possible $C \subseteq V$ are covers) turns out to be tractable. Intriguingly, this can be proved in very different ways, namely using search trees from FPT theory, using backdoor sets from SAT solving, using randomized sampling in conjunction with gaps in probability spectra, using algorithmic meta-theorems, using forbidden minors from graph minor theory, using forbidden subgraphs in conjunction with bounded tree-depth, and using the construction of algorithmically simple well-quasi-orderings. The present paper explains and celebrates how this diverse crowd of approaches, ideas, and methods, which Theoretical Computer Science has developed over the last fifty years, can be applied to solve a basic problem like "decide whether most vertex sets cover a given graph," but also to solve many (at least superficially) harder counting–threshold problems.

1 Introduction

One of the central endeavours of Theoretical Computer Science in general, and of Algorithmics and Complexity Theory in particular, is to find different and ever better ways of solving computational problems. For some fundamental problems, such as matrix multiplication or sorting, there are impressive and seemingly endless (and not-yet-dried-up) streams of results in the literature that offer incremental run-time improvements in theory or in practice or both. Besides improving algorithms, one can also analyze problems from different viewpoints: For instance, the vertex cover problem and the clique problem are both NP-complete via extremely weak reductions (and thus, in a sense, just “different encodings of the same question”), but these problems differ strongly when viewed from other viewpoints: The vertex cover problem is easily 2-approximated (just take all vertices in a maximal matching) while the clique problem does not even permit an $n^{1-\epsilon}$-approximation (unless $\mathrm{P} = \mathrm{NP}$, see [35]), the vertex cover problem is fixed-parameter tractable [10], while the clique problem is W[1]-hard [9]. Studying computational problems using the different analytic tools developed by computer scientists over last fifty years, rather than focusing on a single algorithm for a problem, resembles exploring a landscape by hiking to different viewpoints and taking in the different views, rather than looking at a pixelated photo.

The landscape that we will explore in the present paper is centered on the following problem: “Given an undirected graph $G = (V, E)$, is it true that at least half of all possible vertex subsets $C \subseteq V$ are *covers* of G, meaning that each edge $\{u, v\} \in E$ intersects C?” For example, triangles (also known as K_3) have this property: The empty set and the three one-element sets (, , and) are no covers of the triangle, while the three two-element subsets (, , and) as well as the complete set () are covers. So, four out of eight subsets are covers and, thus, at least half. Similarly, 4-vertex paths (also known as P_4) have this property and so does any star like or (known as $K_{1,k}$ in general), while a 4-vertex cycle (also known as C_4) does not, since out of the 16 possible subsets only seven are covers.

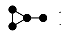
At first glance and from afar, this problem appears to be a rather hard problem since counting vertex covers is a hard problem in general: It is complete [33] for the class #P, which is high up in the polynomial hierarchy. Of course, our task is not actually to determine the number $\#_{\mathrm{vc}}(G)$ of vertex covers of an arbitrary graph $G = (V, E)$, but “just” to determine whether $\#_{\mathrm{vc}}(G) \geq 2^{|V|}/2$ or $\#_{\mathrm{vc}}(G) < 2^{|V|}/2$ holds. Nevertheless, it seems, at least at first glance, that in order to determine whether this very fine and precise distinction holds, we need to compute $\#_{\mathrm{vc}}(G)$.

When we approach the problem more closely, we suddenly see that it is very much tractable: Indeed, in the earlier spirit of looking at a problem from different viewpoints, *seven different ways will be explored of proving that the problem can be solved efficiently.*

Before we have a closer look, some more rigorous definitions will be useful: As already indicated, an *undirected graph G* is a pair (V, E) of a *vertex set V* and an *edge set E* consisting of subsets of V of size 1 or 2 (so we allow loops, but no “empty edges”). A *(vertex) cover of G* is a set $C \subseteq V$ with $C \cap e \neq \emptyset$ for all $e \in E$. Let $\#_{\mathrm{vc}}(G)$ be the size of the set $\{C \subseteq V \mid C \text{ is a cover of } G\}$, so $\#_{\mathrm{vc}}() = 4$ and $\#_{\mathrm{vc}}() = 7$, and let $\Pr_{\mathrm{vc}}[G] := \#_{\mathrm{vc}}(G)/2^{|V|}$ be the fraction of subsets that are covers (or, equivalently, the probability that a randomly chosen subset is a cover), so $\Pr_{\mathrm{vc}}[\,] = \frac{1}{2}$ and $\Pr_{\mathrm{vc}}[\,] = \frac{7}{16}$. Then the graph problem whose complexity we study in this paper is:

Definition 1.1 (Majority of Vertex Covers Problem). MAJ-VC := $\{G \mid \Pr_{\text{vc}}[G] \geq \frac{1}{2}\}$.

The fact that covers $C \subseteq V$ intersect all edges of a graph clearly means that the remaining vertices $I = V \setminus C$ must form an independent set. Thus, MAJ-VC is the *same* problem (not just a different encoding, but literally the same problem) as asking whether at least half of all possible vertex sets are independent sets in a given graph.

A different way of looking at MAJ-VC is to think of the vertices of G as propositional variables and to think of a set $C \subseteq V$ as an assignment β that makes the variables in C true and all other variables false (formally, let $\mathbb{B} = \{\mathit{false}, \mathit{true}\}$ contain the two possible truth values and define $\beta\colon V \to \mathbb{B}$ by $\beta(v) = \mathit{true}$ for $v \in C$ and $\beta(v) = \mathit{false}$ for $v \notin C$). Then the "cover property," by which $C \cap e \neq \emptyset$ must hold for all $e \in E$, means that for each $\{u, v\} \in E$ we must have $\beta(u) = \mathit{true}$ or $\beta(v) = \mathit{true}$. In other words, if we form a propositional formula ϕ_G that is a conjunction, taken over all $\{u, v\} \in E$, of the clauses $(u \vee v)$, then β must be a satisfying assignment of ϕ_G. For example, the graph  is the formal tuple $G = (\{v_1, v_2, v_3, v_4\}, \{\{v_1, v_2\}, \{v_2, v_3\}, \{v_1, v_3\}, \{v_3, v_4\}\})$ and it corresponds to the formula $\phi_G = (v_1 \vee v_2) \wedge (v_2 \vee v_3) \wedge (v_1 \vee v_3) \wedge (v_3 \vee v_4)$. Note that ϕ_G is in conjunctive normal form with at most two literals per clause and without negations; and also note that for any formula ϕ of this form there is a graph G such that $\phi = \phi_G$. Phrased differently, MAJ-VC can be seen as the problem of telling for a 2CNF formula ϕ without negations (a *monotone* formula) whether at least half of all assignments are satisfying. In the notation of [30], MAJ-VC would be written as MON-2SAT-PR$_{\geq 1/2}$.

Instead of switching to logic in order to generalize MAJ-VC, we can also stay in the context of graph theory, but consider *hypergraphs.* A *k-hypergraph* is a pair $H = (V, E)$ where V is still a vertex set and each $e \in E$ is a subset of V of size *at most k.* Thus, undirected graphs are 2-hypergraphs in this sense (ignoring the possibility of an empty edge $e = \emptyset$, which are allowed in hypergraphs, but forbidden in normal graphs). The generalization of vertex covers are *hitting sets,* which are sets $X \subseteq V$ such that $e \cap X \neq \emptyset$ holds for all $e \in E$. The problem of telling whether at least half of all vertex subsets of a, say, 3-hypergraph are hitting sets is then the problem MAJ-3HS. Note that this is the same problem as MON-3SAT-PR$_{\geq 1/2}$, so, once more, we can recast these graph problems as SAT problems.

Contributions of This Paper. As already mentioned, MAJ-VC, the problem of telling whether most vertex sets are covers of a graph, is a tractable problem – and readers are invited to try to come up with a polynomial-time algorithm at this point, if they have not already done so (it is not too hard, but certainly not trivial). The central contribution of the present paper is *not* showing that MAJ-VC $\in$ P holds: This follows easily from the work of Akmal and Williams [2], who show not only MAJ-VC = MON-2SAT-PR$_{\geq 1/2}$ $\in$ P, but even the *much* more general statement

kSAT-PR$_{\geq p}$ $\in$ P for all $k \geq 1$ and all rational $p \in [0,1]$, by which tractability still holds if we allow negated literals, if we allow clauses of any fixed size $k > 2$, if we allow other thresholds $p \neq 1/2$, and if we allow any combination thereof.

Rather, the present paper is about presenting seven different ways of proving the tractability of MAJ-VC (or, equivalently, of MON-2SAT-PR$_{\geq 1/2}$). We focus on this problem (and not on more general problems like the aforementioned kSAT-PR$_{\geq p}$) to keep the presentation simple (the analysis of kSAT-PR$_{\geq p}$ in [2] and also in [30] quickly becomes extremely technical), but also because some of the ideas presented in the following only work for monotone clauses or for size-2 clauses or both. Nevertheless, in each case we will have a look at how the presented ideas generalize (or do not).

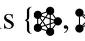
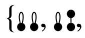
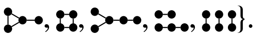
The first way is quite straight(forward): Apply the *search tree technique* [10] from FPT theory. However, we need a small "twist" as our problem is not really a parameterized problem: Build a search tree by finding an arbitrary edge $\{u, v\} \in E$ in the graph and then add three numbers, namely the number of G's covers C with $C \cap \{u, v\} = \{u\}$, with $C \cap \{u, v\} = \{v\}$, and with $C \cap \{u, v\} = \{u, v\}$, which are obtained recursively. In a normal search tree, we end the recursion at a parameter-dependent depth since, when this depth is reached, "no solution" can exist; in our case, we end it with the answer "no majority of covers" when depth 3 is reached. We will show that this answer is, then, correct.

The second way is based on a technique from SAT solving: Backdoor sets [34]. We will see that whenever most vertex sets are covers, we can always find a backdoor set of size 4 into 1CNFS and this will allow us to compute the number of covers quite easily.

The third way of solving the problem is through randomization and is certainly the easiest to state in full: *On input G, randomly sample 2,250 subsets $C_1, \dots, C_{2{,}250} \subseteq V$ and claim "$G \in$ MAJ-VC" if at least 1,089 of them are covers.* While easy to state, deep insights into the *spectra of satisfaction probabilities of* 2CNF *formulas* will be needed to show that this algorithm gives a correct answer with probability at least $2/3$, placing the problem in BPP, the randomized version of polynomial time (which will also count as "tractable").

The fourth way leaves the solid path of concrete algorithmics and turns to algorithmic *meta* theorems. We will show that we can apply Courcelle's Theorem [8] as follows: If the tree-width of G is at most 4, apply (a counting version of) Courcelle's Theorem to the free second-order variable C in the formula $\forall x \forall y (x \sim y \to (C(x) \lor C(y)))$; and if the tree-width is larger, output "no majority of vertex covers." While this application is superficially simple (at least for meta-theorem enthusiasts), intriguingly, it sadly does *not* generalize to, say, MAJ-3HS, meaning that we do not get a (much) simpler tractability result for that problem compared to the complex analysis in [2, 30].

The fifth way, which arguably leads us into an even more "abstract" landscape,

is to apply the famous graph minor theorem of Robertson and Seymour. By this theorem, if a property is "closed under taking minors" (a classical example is the class of planar graphs), then it can be characterized by a finite set of graphs that are "forbidden as minors" (in the case of planar graphs, by Kuratowski's Theorem [19] the set is {}, consisting of the 5-clique K_5 and the complete bipartite graph $K_{3,3}$). It is simple (but not trivial) to show that the property "most vertex sets are covers" is closed under taking minors and, hence, it can be characterized by a finite set of forbidden minors. Even better, we can explicitly state this set: It is {
}.

The sixth way replaces the powerful minors by forbidden induced subgraphs. This has the advantage of allowing us to generalize the approach (while the property "most vertex sets are covers" is closed under taking minors, this is no longer true for interesting variations), but has the disadvantage that the analogue of the Robertson–Seymour Theorem ("there is a finite set of forbidden induced subgraphs") is not generally true. Nevertheless, we will see that for MAJ-VC we can restrict attention to tree-depth 5 and use this to solve MAJ-VC via a finite set of forbidden induced subgraphs.

The seventh way starts where the fifth and sixth way meet: We define a *well-quasi-ordering* on graphs (the graph minor relation is also a well-quasi-ordering) that is easy to decide and relative to which the property "most vertex sets are covers" is closed. The theory of well-quasi-orderings will then provide us with a set of forbidden elements that characterizes the property. In contrast to the subgraph relation and to the graph minor relation, the presented ordering is, on the one hand, a well-quasi-ordering and, on the other hand, lends itself to generalizations for problems like kSAT-PR$_{\geq p}$.

Related Work. As already indicated, MAJ-VC is a special case of counting–threshold problems for propositional formulas in conjunctive normal form (CNF formulas). These more general problems have, of course, attracted *a lot* of attention since already the basic problem SAT ("Is the number of satisfying assignments of ϕ at least 1?") is NP-complete by the Cook–Levin Theorem [7,20]. It remains NP-complete when the clauses are required to have size at most 3 (resulting in the 3SAT problem) and becomes NL-complete when the clause size is at most 2 (the 2SAT problem). The problem becomes trivial, of course, when no negations are allowed as, then, setting all variables to *true* is always a satisfying assignment.

When it comes to thresholds larger than 1 for the number of vertex covers of a graph or the number of satisfying assignments of a formula, the complexity landscape gets even more interesting. The mapping $\#\colon \text{CNFS} \to \mathbb{N}$ that maps CNF formulas ϕ to the number of their satisfying assignments is a classical #P-complete problem [33] and, hence, believed to be very hard. Furthermore, it remains complete when we only allow only 3CNFS as input; and it *still* remains #P-complete when we only allow 2CNFS and *even still* when only monotone 2CNFS are allowed, see [33]. In other words, the function $\#_{\text{vc}}(\cdot)$ considered in the present paper, which maps graphs to the number of their vertex covers, is a #P-complete function and so is $\Pr_{\text{vc}}[\cdot]$ since it is just a rescaled version.

When it comes to the "at most" or "majority" version, that is, to MAJ-VC or more generally kSAT-PR$_{\geq 1/2}$ or even SAT-PR$_{\geq 1/2}$, an astounding thing happens: While SAT-PR$_{\geq 1/2}$ is complete for PP (basically "a decision version of #P" that is high up in the polynomial hierarchy by Toda's Theorem [31]), Akmal and Williams [2] showed that kSAT-PR$_{\geq 1/2} \in \text{P}$ holds for all $k \geq 1$ (see also [30] for a simpler proof) and, thus, MAJ-VC $\in$ P. It is noteworthy that Akmal and Williams start their discussion of kSAT-PR$_{\geq 1/2}$ with the case $k = 2$ as an introductory example and one of the "ways" presented in the present paper (namely the use of backdoor sets) is also sketched in their paper.

The basics of the theoretical backgrounds for the seven ways of proving the tractability of MAJ-VC will be sketched in the main text, but for readers interested in more details, here are some possible entry points to the literature: Concerning fixed-parameter tractability theory, two classical textbooks are [10, 13]; for a gentle introduction to algorithmic meta-theorems, see for instance [29]; for more background on well-quasi-orderings, see [18]. Concerning the ubiquitous technique of randomization, finding an "entry point" to the literature seems hopeless, but in the particular case of finding satisfying assignments for kCNF formulas, readers should have a look at Schöning's ingenious use [28] of randomization to solve the kSAT problem at some point (although [23] shows that randomization is not *actually* necessary).

Organization of This Paper. The paper will take you on a tour via seven ways, each of which is described in one of the following sections, through a fascinating landscape of complexity and algorithms. Each section starts with a bit of "scenery description," followed by background on the method or tools employed (like search trees or well-quasi-orderings and so on), followed by a theorem stating and a proof showing the tractability of MAJ-VC. Each section then finishes with some observations concerning whether or not the way can be generalized.

2 The First Way: Via Search Trees

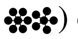
As you start your tour through the complexity landscape surrounding MAJ-VC, the first thing you notice is, of course, that there are lots of trees: Computer Science in general, and Theoretical Computer Science in particular, is a land of lush forests filled with trees of all kinds. The first way soon leads you to a fenced-off area with majestic trees growing in it. Next to an inviting entrance gate, a gardener greets you: “Welcome to the Search Tree Nursery. We grow – and sell – all kinds of *Quaerere arbore!* Can I interest you in any of our trees? And, of course, it would be helpful to know a bit about your budget.” You respond that you need a tree for a counting–threshold problem and that your budget is polynomially bounded. The gardener seems a bit at a loss upon hearing about your budget (apparently, polynomial time is considered a *very* tight budget in the Search Tree Nursery), but then her eyes light up: “That is an uncommon request, I dare say, but I like a challenge. You see, most people use search trees for, well, searching, not for counting and certainly not for counting–threshold frivolities. And your budget *is* restricted. But do not fear, I think I have just the right tree for you.” She leads you to a part of the nursery where a sign says “Bonsai and assorted $O^*(1)$” and then hands you a small, but beautiful specimen of *Quaerere arbore.*

We will examine this particular specimen more closely in a moment, but let us first have a brief closer look at the vertex cover problem.

Background on the Vertex Cover Problem in FPT Theory. The vertex cover problem is the poster child of FPT theory, the theory of fixed-parameter tractability, as the core ideas of this theory (search trees and kernelization, in particular) work spectacularly well for it (see [15] for a recent record). However, the parameterized problem p-vc (or, in full, p-VERTEX-COVER) studied in this theory is fundamentally different from the counting–threshold problem MAJ-VC we study in the present paper: For p-vc the input is a pair (G, k) of an undirected graph G and a natural number $k \in \mathbb{N}$ (the *parameter* in FPT parlance) and the question is whether there is a cover C of G of size $|C| \leq k$. The problems are connected, nevertheless as, intuitively, if a graph has “many” covers, there must be some “small” covers among them. The following lemma shows that this intuition is very much correct:

Lemma 2.1. *Let $G = (V, E)$ be a graph with* $\Pr_{\text{vc}}[G] \geq 1/2$. *Then all matchings in G have size at most* 2 *and G has a vertex cover of size at most* 4.

Proof. Suppose G contains a matching of size 3 (⁝⁝⁝ $= M_3$) as a subgraph, that is, there are six different vertices $v_1, \dots, v_6 \in V$ with $\{\{v_1, v_2\}, \{v_3, v_4\}, \{v_5, v_6\}\} \subseteq E$. Then the probability that a random subset $C \subseteq V$ covers the edge $\{v_1, v_2\}$ is $3/4$ and the same is true for the other edges. Since these probabilities are independent, the probability that all three edges are covered is at most $(\frac{3}{4})^3 = \frac{27}{64} < 1/2$. Hence, the probability $\Pr_{\text{vc}}[G]$ that *all* edges of G are covered is less than $1/2$. For the second claim, consider a maximal matching $M \subseteq E$. Then $|M| \leq 2$ and the at most 4 vertices in M (that is, in $\bigcup M$) form a (vertex) cover of G since, otherwise, there would be an uncovered edge in E and the matching would not be maximal. □

The reverse direction of the lemma's statement is not true: A graph consisting of two large stars (like ) can be covered by just two vertices ( ), but just a little more than one quarter of all vertex sets are covers. This means that we will have to put forward some extra ideas to make the *search tree technique,* which works so well for the p-vc problem, also work for maj-vc.

Background on the Search Tree Method. Suppose we are given a pair (G, k) and wish to determine whether there is a cover $C \subseteq V$ of G of size $|C| \leq k$. If $k = 0$, the answer is obviously "no" unless $E = \emptyset$. The other way round, if $E = \emptyset$, the answer is always "yes." The interesting case is $k > 0$ and $E \neq \emptyset$. Then there must be an edge $\{u, v\} \in E$. The key insight behind the search tree method is that *for every cover C of G we must have $u \in C$ or $v \in C$.* Admittedly, when written down like this, the insight feels utterly trivial, but fear not: Theoretician have found ways to complicate the idea beyond all recognition. Anyway, in the case $u \in C$, the vertex u will cover all edges $e \in E$ that contain u, but $C \setminus \{u\}$ must then cover "the rest" of the edges, that is, the edges in $E \setminus \{e \in E \mid u \in e\}$. (The situation for the second case is symmetric, with u and v exchanged.) In particular, we can only cover G with a size-k cover, if we can cover either $G \setminus \{u\}$ or $G \setminus \{v\}$ with a size-$(k-1)$ cover (here, $G \setminus X := (V \setminus X, E \cap \{\{u, v\} \mid u, v \in V \setminus X\})$ results from G by removing all vertices in X together with any adjacent edges) – and we can then, of course, try to answer these questions recursively, see Figure 1 for an example. The resulting recursion tree is called the *search tree,* hence the name of the method. Note that its depth is at most k (we stopped the recursion at $k = 0$) and we have two recursive calls in the algorithm, resulting in two children at each inner node of the search tree. All told, the search tree has size 2^k and the runtime of the algorithm will be something like $O(2^k|E|)$, depending a bit on which data structures we use for representing E.

An advanced version of this idea, which will be useful in our counting–threshold setting in a moment, is that instead of branching to the two cases "C contains u" and "C contains v", we cleverly choose the edge $\{u, v\} \in E$ and then branch to *three* cases, namely "C contains u, but not v" and "C contains v, but not u" and "C contains both u and v." The insight is that in the branch that looks for covers that "contain u, *but not v,*" we know that when v is not part of a cover, *all neighbors of v must be part of the cover,* allowing us to remove all edges adjacent to these neighbors and to look for a much smaller cover for the rest, see Figure 2 for an example. By choosing the edge $\{u, v\} \in E$ to maximize the number of these neighbors, we can ensure that while there will always be three branches, the subtrees will have only half the depth, leading to a better overall runtime. Of course, we can try to take this idea further and cleverly choose more than a single edge and branch more broadly to get even more shallow trees. A small cottage

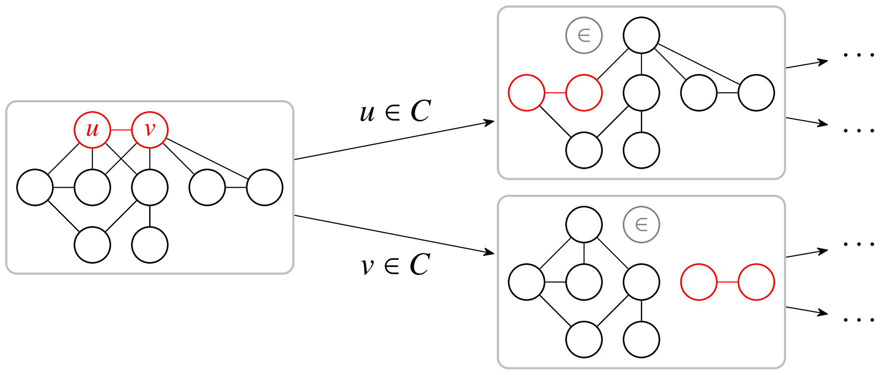


Figure 1: A binary search tree for finding a vertex cover of size k in an undirected graph $G = (V, E)$: At each node of the tree, starting with the graph itself at the root, shown left, an edge $\{u, v\} \in E$ is selected (shown in red). Then any vertex cover C of G must contain either u, resulting in the graph $G \setminus \{u\}$ shown above (with u no longer being a vertex of the graph), or v, resulting in $G \setminus \{v\}$. Note how, in either case, we have identified one vertex of the cover, meaning that the rest of the graph must be covered with $k - 1$ vertices.

industry has sprung up around this method, resulting in improved search tree sizes (and, hence, improved runtimes [15]) for the vertex cover problem and related problems, but for our purposes the "branching to $\{u, v\} \cap C = \{u\}$, $\{u, v\} \cap C = \{v\}$, and $\{u, v\} \cap C = \{u, v\}$" will suffice.

Applying the Search Tree Method. Adapting the search tree method so that we *count* the number of vertex covers of a graph, is not very hard: Instead of just answering "yes" or "no," we let the algorithm output the number of vertex covers it has found. For instance, when $E = \emptyset$ holds, instead of just answering "yes," we answer "$2^{|V|}$," which is the number of vertex covers that an empty graph on $|V|$ vertices has. Crucially, when we recursively branch for the three cases $\{u, v\} \cap C = \{u\}$, $\{u, v\} \cap C = \{v\}$, and $\{u, v\} \cap C = \{u, v\}$, the total number of covers will be the *sum* of the three numbers computed – and note that in this three-way branching we do not double-count any covers (which we would, if we just branched to the cases $u \in C$ and $v \in C$).

What about the depth of the search tree? Usually, we stop the search and get a leaf node in the tree when we know that "any cover below this node" (meaning, any cover satisfying all the requirements imposed by the nodes on the path from the root) will have size greater than k. In our case, however, we clearly cannot ignore these covers as most covers will be large. The trick is to use Lemma 2.1: Each time we recurse, the chosen edge will be completely disjoint from the previously chosen edges. By the lemma, if we still find an edge at recursion depth 3, we can

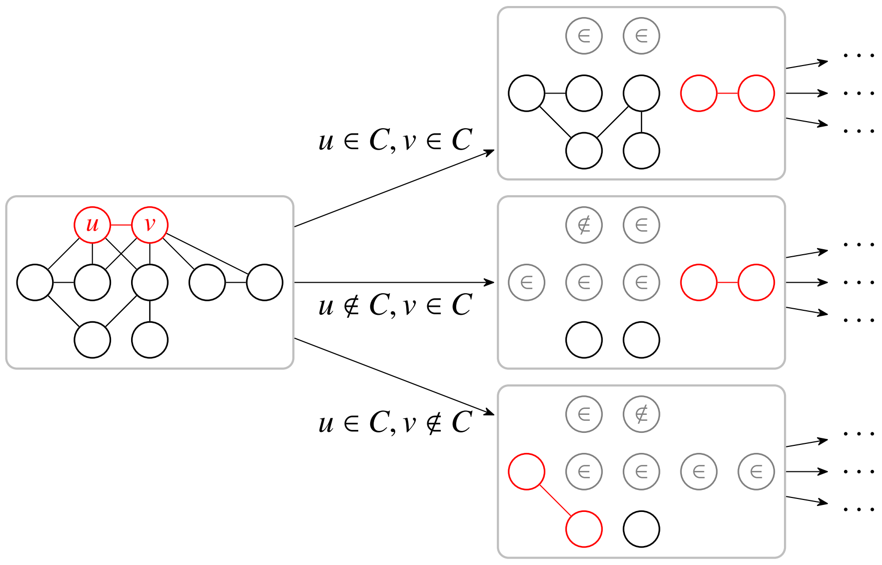


Figure 2: A ternary search tree for counting the number of vertex covers of an undirected graph $G = (V, E)$. In contrast to the recursion in Figure 1, at each node we classify the possible covers C of G according to whether $C \cap \{u, v\} = \{u, v\}$ holds (top case) or $C \cap \{u, v\} = \{u\}$ or $C \cap \{u, v\} = \{v\}$ (the other two cases). In the lower cases, $u \notin C$ (or $v \notin C$) forces the neighbours of u (or v) to be part of the cover. Crucially, the number of covers that the graphs in the branches have (39 for the top graph, 12 for the middle graph, 6 for the lower graph) add up to the number of covers the root graph has (57 out of 512 possible vertex sets).

break off the whole computation.

Algorithm 1: An algorithm based on search trees for deciding whether an undirected input graph $G = (V, E)$ is an element of MAJ-VC. In the algorithm, $N[u] := N(u) \cup \{u\}$ is the closed neighbourhood of u and $N(u) = \{w \in V \mid \{u, w\} \in E\}$ is the (open) neighbourhood. For the correctness of the algorithm, see the proof of Theorem 2.2.

```
input $G$
if $count\text{-}vc(G, 0) \geq 2^{|V|}/2$ then output ``$G \in$ MAJ-VC'' else output ``$G \notin$ MAJ-VC''

function $count\text{-}vc(G = (V, E), d)$ // $d$ is the size of an already found matching
    if $E = \emptyset$ then return $2^{|V|}$ // every subset is a cover
    if $d \geq 2$ then output ``$G \notin$ MAJ-VC'' and halt // see the proof for the correctness
    if $\{v\} \in E$ for some $v \in V$ then // self-loop at $v$
        return $count\text{-}vc(G \setminus \{v\}, d + 1)$
    else
        $\{u, v\} \leftarrow$ an arbitrary element of $E$
        $c_1 \leftarrow count\text{-}vc(G \setminus N[u], d + 1)$
        $c_2 \leftarrow count\text{-}vc(G \setminus N[v], d + 1)$
        $c_3 \leftarrow count\text{-}vc(G \setminus \{u, v\}, d + 1)$
        return $c_1 + c_2 + c_3$
```

Theorem 2.2. *Algorithm 1 decides* MAJ-VC *in time* $O(|E|)$.

Proof. The algorithm is a simple recursion, in which the function *count-vc*(G, d) returns the number of vertex covers of G and the number d is the size of an already found matching (and also the depth of the recursion). Clearly, if we can show that the implementation correctly computes this function, the overall algorithm is correct.

If $E = \emptyset$ holds in line 5, every subset $C \subseteq V$ is a cover of G, so $2^{|V|}$ is the correct return value. If this is not the case, there must be an edge in E. However, if $d \geq 2$ holds, we know that this edge would constitute the third edge in a matching in E and by Lemma 2.1 we correctly output "$G \notin$ MAJ-VC" in this case (and halt the whole computation, not just this branch of the recursion). Otherwise, we check whether there is a vertex v with a loop, in which case every cover *must* contain v, so the total number of covers of G equals the number of covers of $G \setminus \{v\}$, which results from G by removing v and all edges containing v. In other words, the value returned in line 8 is correct.

Finally, we consider a size-2 edge $\{u, v\} \in E$. The three values c_1, c_2, and c_3 are now exactly the number of covers C of G such that $C \cap \{u, v\} = \{v\}$, $C \cap \{u, v\} = \{u\}$, and $C \cap \{u, v\} = \{u, v\}$, respectively. To see this, just note that $C \cap \{u, v\} = \{v\}$ means that u is *not* in the cover and, hence, all neighbours of u must be in the cover.

In other words, for every cover C' of $G \setminus N[u]$, the set $C' \cup N(u) = C \cup \{w \in V \mid \{u, w\} \in E\}$ is a cover C of G with $C \cap \{u, v\} = \{v\}$. By a similar argument, we get that the numbers c_2 and c_3 are correct, meaning that the final output $c_1 + c_2 + c_3$ is also correct.

For the runtime, just observe that because of the cap on the recursion depth, the search tree has size at most $3^2 = 9$. All other computations take time linear in $|E|$ assuming appropriate data structures. □

Extensions. Given that the search tree method is a rather powerful and generic tool, it should come as no big surprise that we can use it to solve not only MAJ-VC, but also more general problems:

- The threshold "1/2" used in the definition of MAJ-VC for the fraction of vertex sets that must be covers is both natural and a bit arbitrary (why not "1/3" or "3/4"?). If we change the threshold from 1/2 to a different number $p \in [0, 1]$, we get the problem MON-2SAT-PR$_{\geq p}$ (see the introduction). Having a look at the proof of Lemma 2.1, we see immediately that the following generalization holds:

  Lemma 2.3. *Let G be a graph with $\Pr_{\text{vc}}[G] \geq p > 0$. Then all matchings in G have size at most $\log_{3/4} p$ and G has a vertex cover of size at most $2 \log_{3/4} p$.*

  This means that Theorem 2.2 still works if we replace "*count-vc*$(G, 0) \geq 2^{|V|}/2$" by "*count-vc*$(G, 0) \geq 2^{|V|} \cdot p$" in line 2 and "$d \geq 2$" by "$d \geq \log_{3/4} p$" in line 6. Observe that while the size of the search tree is constant for any *fixed p,* for arbitrary p, the size of the search tree and hence also the runtime is $3^{\log_{4/3}(1/p)}$ and hence exponential in the number of bits of $\lceil 1/p \rceil$ – and this is no coincidence: A polynomial dependence of the runtime on the number of bits of $\lceil 1/p \rceil$ would allow us to compute $\Pr_{\text{vc}}[G]$ and hence also $\#_{\text{vc}}(G)$ in polynomial time using binary search, but $\#_{\text{vc}}(\cdot)$ is a #P-complete function.

- We can modify the test from line 2 further and check whether we have "*count-vc*$(G, 0) > 2^{|V|} \cdot p$" (rather than "$\ldots \geq \ldots$") and get an algorithm for solving MON-2SAT-PR$_{>p}$ in polynomial time. The difference may seem slight, but as first observed by Akmal and Williams in [2] and then studied in more detail in [30], for counting–threshold problems it can make a *huge* difference whether "$\geq p$" or "$> p$" is considered.

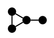
- It is not too hard – but no longer a trivial code modification – to extend Algorithm 1 to work for the more general 2SAT-PR$_{\geq p}$ problem, where negated literals are allowed. It will, however, be more natural to look at this problem in the context of backdoor sets, which we do in the next section.

- Concerning the even more general problem $\textsc{maj-3hs} = \textsc{mon-3sat-pr}_{\geq 1/2}$, let alone $k\textsc{sat-pr}_{\geq p}$ for arbitrary $k > 2$ and p, the search tree method (at least, as presented) simply fails: It is not hard to construct 3-hypergraphs for which there is no way to "stop early in line 6". For instance, the hypergraph H with the edges $\{a, x_1, y_1\}$, $\{a, x_2, y_2\}$, $\dots$, $\{a, x_m, y_m\}$ is covered by every vertex set containing a (so $\Pr_{\textsc{3hs}}[H] > 1/2$), but the search tree would have depth m.

3 The Second Way: Via Backdoor Sets

Leaving the Search Tree Nursery behind you, you continue on your journey through the landscape on a second way that leads you further through hills and forests. Just as you exit a small valley, an impressive vista opens before you and reveals a view of a huge mountain in the distance. It takes you quite a while to get to its foot, where you meet a courteous hobbit who seems to be waiting for someone. He immediately addresses you: "Well met, Master Traveller. What brings you to the Lonely Mountain? The treasures inside, I would venture. Well, I, for one, am waiting for a group of dwarves and a wizard, but they seem to be *quite* late, I must say." You are about to inquire about the treasures, but the hobbit just keeps talking: "I, for one, *really* do not agree with the dragon that the gold is the real treasure. He really should get out more, I *keep* telling him. Anyway, in this gentlehobbit's opinion, the real treasure is the machinery! All these wonderful devices for *counting* the gold and what-have-you." Your interest sparks at this, but before you can squeeze in a question about counting machinery, the oration continues relentlessly: "Of course, getting in is out of the question, this mountain is a real fortress: As you can see, the old dwarven entrance here is *very* securely locked down. And, of course, all the little windows further up and the tunnels on the sides are barred by large iron grilles. I guess, one might be able to use one of the chimneys that come out at the mountain top, but you would have to be an *exceptionally* skilled climber just to get up there, let alone then getting down them. So, getting in is totally out of the question. Unless, of course..." Now the hobbit stops talking, obviously expecting you to ask him to elaborate. Since you would very much like to get into the mountain, preferably avoiding the dragon, you coax him to continue, which he happily does: "You see, I happen to have this map with a secret *backdoor* marked on it. A bit difficult to read, if I may say so, but I can lend you my moonshine lamp. I was going to give the map to the dwarves, but as I told you, they are *quite* late. So, here, you take it and go inside and have a look around!"

As we will see in the following, the backdoor does, indeed, grant us access to $\textsc{maj-vc}$.

Recall that $\textsc{maj-vc}$ is a special case of the problem $2\textsc{sat-pr}_{\geq 1/2}$, where the input is a propositional formula ϕ in conjunctive normal form with at most two literals per clause (a 2cnf formula) and the question is whether at least half of all possible assignments $\beta\colon \operatorname{vars}(\phi) \to \mathbb{B}$ satisfy ϕ (with $\operatorname{vars}(\phi)$ of course denoting the set of variables in ϕ and $\mathbb{B} = \{\mathit{false}, \mathit{true}\}$ denoting the two possible truth values): Covering all edges $\{u, v\} \in E$ corresponds exactly to satisfying all clauses of the form $(u \lor v)$. For example, the number of vertex covers of $G =$  is exactly the number of satisfying assignments of $\phi_G = (v_1 \lor v_2) \land (v_1 \lor v_3) \land (v_2 \lor v_3) \land (v_3 \lor v_4)$. It is, thus, no surprise that the tools that have been developed for solving sat problems also apply to maj-vc. One such tool are backdoor sets [34].

Background on Backdoor Sets. The paramount goal of sat solving is to determine on input of a formula ϕ in conjunctive normal form whether it has a satisfying assignment. Since sat is *the* NP-complete problem *par excellence,* we cannot really hope to devise an efficient algorithm for solving this problem for arbitrary ϕ, *but* if ϕ has a special form and is sufficiently "simple," even highly efficient algorithms are known. For instance, if $\phi \in$ 2cnfs holds (that is, when ϕ is in conjunctive normal form with at most two literals per clause), we can decide whether ϕ is satisfiable in polynomial time (more precisely, the 2sat problem is NL-complete), and this is also true when $\phi \in$ horn-cnf holds (meaning that there is at most one positive literal per clause) as horn-sat is P-complete [22].

The idea behind backdoor sets is that while a cnf formula ϕ may not be syntactically simple in one of the above senses, it may be "near" to such a formula as only a small number of variables cause a deviation. For instance, $\phi \notin$ 2cnfs might hold, but there might be only, say, four variables x_1, x_2, x_3, and x_4 (which will be called a *backdoor set* in a moment) such that removing them from the formula would result in an element of 2cnfs. The key observation is that there actually is a simple way of "removing" variables from any cnf formula ϕ: Iterate over all possible assignments to the variables in the backdoor set and then use unit propagation to remove the variables. These ideas are detailed in the following two definitions:

Definition 3.1. Let ϕ be a cnf formula, let X be some variables, and let $\beta\colon X \to \mathbb{B}$ be a function (a "partial assignment"). Define $\phi|_\beta$ as the formula obtained from ϕ by

1. removing all clauses from ϕ containing a literal made true by β, that is, containing a variable $v \in X$ with $\beta(v) = \mathit{true}$ or containing $\neg v$ for some $v \in X$ with $\beta(v) = \mathit{false}$; and
2. removing from the remaining clauses all literals containing a variable from X.

As an example, for $\phi = (x \lor \neg y \lor a) \land (\neg x \lor b \lor \neg c) \land (y \lor d) \land (d)$ and $X = \{x, y\}$ and $\beta(x) = \mathit{true}$ and $\beta(y) = \mathit{false}$, we have $\phi|_\beta = \cancel{(x \lor \neg y \lor a)} \land (\cancel{\neg x} \lor b \lor \neg c) \land$

$(\cancel{y} \vee d) \wedge (d) = (b \vee \neg c) \wedge (d)$. Note how $\phi \notin$ 2CNFS holds, while $\phi|_\beta \in$ 2CNFS holds both for our particular β and also for the three other possible $\beta\colon X \to \mathbb{B}$.

Definition 3.2. Let Ψ be a set of CNF formulas and let ϕ be any CNF formula. A *(strong) backdoor set into Ψ for ϕ* is a set X of variables such that for all $\beta\colon X \to \mathbb{B}$ we have $\phi|_\beta \in \Psi$.

To get an intuition for the above definition, consider a formula ϕ, a fixed set $V \supseteq \operatorname{vars}(\phi)$ of variables, and an arbitrary set $X \subseteq V$. If we restrict an assignment $\alpha\colon V \to \mathbb{B}$ that satisfies ϕ to just the variables in $V \setminus X$, we get a satisfying assignment α' of $\phi|_\beta$, where β is the restriction of α to X. The other way round, we can extend any satisfying assignment α' of $\phi|_\beta$ to one of ϕ by joining α' with β. In particular, if we write $\#(\phi)$ for the number of satisfying assignments that ϕ has relative to V, we get (the division by $2^{|X|}$ is needed as $\#(\phi|_\beta)$ is also counted relative to the original V):

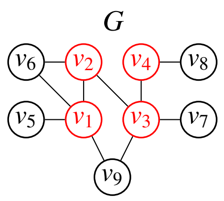
$$\#(\phi) = \sum_{\beta\colon X \to \mathbb{B}} \frac{\#(\phi|_\beta)}{2^{|X|}}. \tag{1}$$

For instance, for $V = \{a, b, c, d\}$ and $\phi = (a \vee \neg b) \wedge (\neg a \vee b \vee c)$ and $X = \{a\}$, we have: $\#(\phi) = 10$ as ϕ has ten satisfying assignments $\alpha\colon \{a, b, c, d\} \to \mathbb{B}$; for $\beta_0(a) = \mathit{false}$ we have $\phi|_{\beta_0} = (\neg b)$ and $\#(\phi|_{\beta_0}) = 8$ as $(\neg b)$ has eight satisfying assignments $\alpha\colon \{a, b, c, d\} \to \mathbb{B}$; for $\beta_1(a) = \mathit{true}$ we have $\phi|_{\beta_0} = (b \vee c)$ and $\#(\phi|_{\beta_1}) = 12$; and, indeed, $10 = (8 + 12)/2^{|x|} = 20/2$.

The crucial property of equation (1) is that $\phi \in$ SAT *holds (the left hand side is positive) iff $\phi|_\beta \in$ SAT holds for at least one $\beta\colon X \to \mathbb{B}$ (at least one summand in the sum on the right hand side is positive).* Now, if X is a small backdoor set into, say, $\Psi =$ 2CNFS, deciding $\phi \in$ SAT becomes easy: We just have to run $2^{|X|}$ many test of the form "Is this 2CNF formula satisfiable?"

Of course, many formulas will not have a small backdoor set and, even when they do, it may be difficult to find. A rich theory has been developed (see for instance [11,21] for some recent results) that tries to deal with these and other problems, but the simple above form of backdoor sets will suffice for solving MAJ-VC.

Applying Backdoor Sets. Recall from Lemma 2.1 that all $G \in$ MAJ-VC have a vertex cover of size 4 or less. Let $X = \{v_1, v_2, v_3, v_4\}$ be such a small cover and now consider what $\phi_G|_\beta$ must look like: Since X is a cover, in each clause $(u \vee v)$ of ϕ_G at least one of the two variables is present in X and the clause is either completely removed in $\phi_G|_\beta$ or one of the variables is removed. Thus, *all clauses of $\phi_G|_\beta$ have size at most 1* (see Figure 3 for an example) and it is *trivial to compute the number of satisfying assignments that $\phi_G|_\beta$ has.* Equation (1) then tells us how to compute the total number of satisfying assignments of ϕ. The details follow.



$$\begin{aligned}\phi_G = {} & (v_1 \lor v_2) \\ & \land (v_1 \lor v_5) \\ & \land (v_1 \lor v_6) \\ & \land (v_2 \lor v_3) \\ & \land (v_2 \lor v_6) \\ & \land (v_3 \lor v_4) \\ & \land (v_3 \lor v_7) \\ & \land (v_3 \lor v_9) \\ & \land (v_4 \lor v_8)\end{aligned}$$

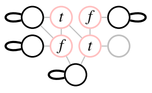
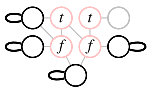
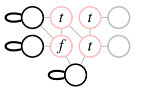
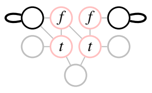
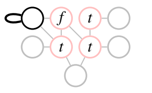
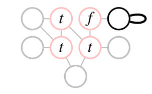
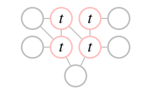
	$\phi_G\vert_\beta = (v_5) \land (v_6) \land (v_8) \land (v_9)$ $\#(\phi_G\vert_\beta) = 32$
	$\phi_G\vert_\beta = (v_5) \land (v_6) \land (v_7) \land (v_9)$ $\#(\phi_G\vert_\beta) = 32$
	$\phi_G\vert_\beta = (v_5) \land (v_6) \land (v_9)$ $\#(\phi_G\vert_\beta) = 64$
	$\phi_G\vert_\beta = (v_6) \land (v_8)$ $\#(\phi_G\vert_\beta) = 128$
	$\phi_G\vert_\beta = (v_6) \land$ $\#(\phi_G\vert_\beta) = 256$
	$\phi_G\vert_\beta = (v_8)$ $\#(\phi_G\vert_\beta) = 256$
	$\phi_G\vert_\beta = \mathit{true}$ $\#(\phi_G\vert_\beta) = 512$

Figure 3: A graph in which $X = \{v_1, \dots, v_4\}$ forms a vertex cover and, thus, these variables form a backdoor set into 1cnfs for ϕ_G. Out of the sixteen possible $\beta\colon X \to \mathbb{B}$ only those are indicated that do not produce an empty clause in $\phi_G|_\beta$ (meaning that some internal edge of the X-set is not covered). For them, the resulting formula $\phi_G|_\beta$ is given and $\#(\phi_G|_\beta)$ relative to the original size-9 vertex set $V = \{v_1, \dots, v_9\}$. By equation (1), we get $\#(\phi) = (32 + 32 + 64 + 128 + 256 + 256 + 512)/16 = 80$.

Algorithm 2: An algorithm based on backdoor sets for deciding whether an undirected input graph $G = (V, E)$ is an element of MAJ-VC. Since Lemma 2.1 tells us such G cannot contain a matching of size 3, there will be at most four vertices in the backdoor set X in 1CNFS.

```
input G

M ← greedily compute a matching in G that is maximal or has size 3
if $|M| > 2$ then output "$G \notin$ MAJ-VC" and halt // correct by Lemma 2.1
$X \leftarrow \bigcup M$ // the at most 4 vertices in the matching

$\phi_G \leftarrow$ the 2CNF formula representing G
$c \leftarrow 0$ // counter for the number of covers
foreach $\beta\colon X \to \mathbb{B}$ do  // at most 16 possible $\beta$
    $\psi \leftarrow \phi_G|_\beta$  // a shorthand
    if $\psi$ contains no empty clause and not both a clause $(v)$ as well as $(\neg v)$ then
        $c \leftarrow c + 2^{|V|-|\mathrm{vars}(\psi)|}/2^{|X|}$

if $c \geq 2^{|V|}/2$ then output "$G \in$ MAJ-VC" else output "$G \notin$ MAJ-VC"
```

Theorem 3.3. *Algorithm 2 decides* MAJ-VC *in time* $O(|E|)$.

Proof. The algorithm first greedily computes a maximal matching M in G, but we can actually stop this computation early when $|M|$ exceeds 2 since, by Lemma 2.1, we then know that $G \notin$ MAJ-VC holds and stop in line 4. Otherwise, the set $X = \bigcup M$ of vertices in M can have size at most 4 and is a vertex cover of G. Crucially, this implies that X is a backdoor set for ϕ_G into 1CNFS: The clauses of ϕ_G are exactly of the form $(u \lor v)$ for $\{u, v\} \in E$ and the fact that X is a cover of G means that in $\phi_G|_\beta$ every such clause is either missing completely or missing one variable. This means that by equation (1), the main loop correctly computes $\#(\phi_G) = \#_{\mathrm{vc}}(G)$ provided that we correctly sum up the values of $\#(\phi_G|_\beta)$. However, for a 1CNF formula ψ, the number of satisfying assignments relative to a fixed size-n set of variables is either 0 (when it is a contradiction, which we rule out by the checks in line 11) or is 2 raised to the number of variables *not* mentioned in ψ.

All told, the output at the end of the algorithm is correct and can be computed easily in time linear in $|E|$ assuming appropriate data structures since the main loop iterates over at most 16 different β. □

Extensions.

- While Algorithm 2 ostensibly solves MAJ-VC, it is really an algorithm for deciding 2SAT-PR$_{\geq 1/2}$ in thinly veiled disguise: Just take a formula $\phi \in$ 2CNFS as input instead of constructing it from a graph G and replace the notion of a "matching" by "a set of variable-disjoint clauses." The rest of the algorithm is already stated in a way so that it also works in the presence of negations (namely in line 11, where we test the presence of contradictory clause pairs (v) and $(\neg v)$, which actually cannot arise when ϕ_G contains no negations).

- Just as in Algorithm 1, we can easily adapt the algorithm to work for $2\textsc{sat-pr}_{\geq p}$ for arbitrary $p \in [0, 1]$ by replacing the test "$M > 2$" by "$M > \log_{3/4} p$" in line 4 and adapting the final line of the algorithm. An adaption of the last line is also all that is needed to decide $2\textsc{sat-pr}_{>p}$.

- When we try to extend the algorithm to 3cnf formulas ϕ in order to solve $3\textsc{sat-pr}_{\geq 1/2}$, we get further than we did on the first way via search trees, but still run into a roadblock: It is not too hard to see that when at least half of all assignments of $\phi \in$ 3cnfs are satisfying, there must still be a small backdoor set, but now into 2cnf formulas rather than 1cnf formulas. Unfortunately, recall that it is #P-compete to compute the number of satisfying assignments of 2cnf formulas, so we cannot simply use a recursion in the main loop.

4 The Third Way: Via Random Sampling

You backtrack out of the mountain, return the map and the moonshine lamp to the hobbit, and thank him once more for showing you the backdoor. Then you are back to hiking, following a third way. As the Lonely Mountain gets lost in the distance behind you, on the side of the road you come by two large silos with the sign *Theoretical Foods Inc.* prominently displayed on both of them. An agitated company employee sees your approach and immediately seeks your help: "Can you *please* help me? You see, we produce two delicious rice mixtures consisting of white rice and wild rice (whose grains are black). The mixtures differ only slightly in the composition: In one there are at least as many black grains as white ones, while in the other there are less black than white ones. Two deliveries have just been made, one of each mixture, and we placed them in two silos, each holding one billion grains. Unfortunately, it is no longer clear which mixture was put in which silo!" Counting all the grains in either silo is clearly hopeless, but you come up with a simple test: From the mixture in the first silo, scoop up a handful of grains (say, a thousand) and count (just) them. Intuitively, if there are much fewer black grains in your scoop than white ones (say, 400 black to 600 white), it is (highly) unlikely that the total number of black grains in the silo is larger than that of the white ones. Unfortunately, in your scoop there are exactly 498 black and 502 white grains – and you really cannot tell whether this really means that are less black grains overall than white ones.

The connection to MAJ-VC is, of course, that while for a large graph G it is hopeless to examine all vertex subsets of G (there are simply too many of them), if we "randomly sample a scoop of them" we can easily count the covers (the "black" grains) and the non-covers (the "white" grains). If there are many more covers in the sample than non-covers, we can be fairly sure that $G \in$ MAJ-VC holds, and if there are many more non-covers than covers, then $G \notin$ MAJ-VC is very likely. The tricky part is, as for the rice grains, the case that the numbers are the same or almost the same in our sample: What does that mean for the "overall mixture of covers and non-covers"?

You are about to tell the employee that, sadly, your idea does not seem to work, when they suddenly volunteer additional information: "You know, in our mixture with less black grains than white grains, we actually put in only at most 46.875% black grains (rather than 49.999%). Does that help in any way?" You soon realize that it sure does: In your scoop of 1,000 grains with 498 black ones and 502 white ones, you are now pretty confident that there are actually at least as many black grains as white grains overall. Otherwise, you would expect to find only 469 black grains, but have found 498 – a strong deviation.

The connection to MAJ-VC is now rather surprising: It turns out that by the *Spectral Well-Ordering Theorem* [30] the "mixture of covers and non-covers of graphs" has the peculiar property that for any graph G, the fraction $\Pr_{\text{vc}}[G]$ of covers is either at least $\frac{1}{2}$ or at most $\frac{15}{32} = 46.875\%$, exactly as for the mixtures of Theoretical Foods Inc. In other words, the *spectrum of values that* $\Pr_{\text{vc}}[G]$ *can have* has a gap in the interval $(\frac{15}{32}, \frac{1}{2})$.

In the following, let us have a closer look at the math behind the sampling technique and these peculiar spectra.

Background on Random Sampling. The mathematical analysis of random sampling (here in the form of a simple Bernoulli process) dates back hundreds of years and is among the best-studied topics of statistics and perhaps mathematics in general. Accordingly, our simple problem of counting how many randomly chosen vertex subsets are covers, is very well-understood from a statistical point of view. In detail, given an undirected graph $G = (V, E)$, if we pick $C \subseteq V$ uniformly at random, then by definition $p := \Pr_{\text{vc}}[G]$ is the probability that C is a cover of G. Sampling t different $C_1, \dots, C_t \subseteq V$ independently and uniformly at random and checking for each whether it is a cover of G yields a Bernoulli process for the probability p. The expected number of C_i that are covers is clearly $p \cdot t$, and bounding the probability that we deviate strongly from this value is well-studied:

Fact 4.1 (Chernoff–Hoeffding Bound [16])**.** *Let $Z_1, \dots, Z_t$ be independent random variables, each taking value* 1 *with probability p and* 0 *with probability* $1 - p$. *Let*

$\epsilon > 0$. *Then*

$$\Pr\left[\tfrac{1}{t}\textstyle\sum_{i=1}^{t} Z_i \geq p + \epsilon\right] \leq \left(\left(\frac{p}{p+\epsilon}\right)^{p+\epsilon}\left(\frac{1-p}{1-p-\epsilon}\right)^{1-p-\epsilon}\right)^{t}, \tag{2}$$

$$\Pr\left[\tfrac{1}{t}\textstyle\sum_{i=1}^{t} Z_i \leq p - \epsilon\right] \leq \left(\left(\frac{p}{p-\epsilon}\right)^{p-\epsilon}\left(\frac{1-p}{1-p+\epsilon}\right)^{1-p+\epsilon}\right)^{t}. \tag{3}$$

In our setting, $Z_i = 1$ when C_i is a cover of G and $Z_i = 0$ otherwise, so $\frac{1}{t}\sum_{i=1}^{t} Z_i$ is exactly the fraction of C_i that are covers. For, say, $p = 1/2$ and $\epsilon = 1/64$, the theorem tells us that the probability that this fraction deviates from the expected value $1/2$ by more than $1/64$ is at most b^t for the constant $b \approx 0.9995117584$. While this constant is quite close to 1, it is strictly less than 1 and b^t will start to tend to 0 quickly for larger t. Indeed, for $t = 2{,}250$ we have $b^t < 1/3$. Figure 5 on page 26 visualizes this and similar situations.

Random sampling has also become an indispensable tool in complexity theory, for instance in the form of the complexity class BPP, which stands for "bounded-error polynomial time." A language L lies in this class if there is an algorithm for deciding L that on input of some x also gets some extra random bits and outputs the correct answer (namely whether $x \in L$ or $x \notin L$ holds) with probability at least $2/3$ (taken over all possible random bits). The choice of "2/3" is, of course, somewhat arbitrary, and any number strictly larger than $1/2$ can be used instead. To increase the likelihood of a correct answer of a BPP-algorithm, one can rerun the algorithm many times and output the majority – and when "many times" means "a polynomial number of times," the likelihood of making an error will become exponentially small. Indeed, it is easy and practical to ensure that even though the algorithm *could* output a wrong answer, it is much more likely that the computer running the algorithm is spontaneously hit by a meteor (or consumed by dragon fire, for that matter). For this reason, "being in BPP" – and not only "being in P" – is a "common theoretician's notion of tractability."

Background on Spectra. The set of possible values that $\Pr_{\text{vc}}[G]$ can have for different graphs G is called the *spectrum of* $\Pr_{\text{vc}}[\cdot]$, and following [30] we will denote it as VC-PR-SPECTRUM, see Figure 4 for a visualization. Clearly, the spectrum is a subset of the interval $[0, 1]$ and it is not hard to see that many values are *not* part of this spectrum:

- The number 0 is not in the spectrum as every graph has a vertex cover (the probability that a random vertex set covers the graph is never 0).
- The number $1/3$ is not in the spectrum as $\Pr_{\text{vc}}[G]$ is always of the form $x/2^y$ for integers x and y (such numbers are called *dyadic* and the spectrum only contains dyadic numbers).

- No number strictly between $\frac{3}{4}$ and 1 is in the spectrum: Once there is a single edge in the graph, at most three quarters of all vertex sets cover this single edge (and hence, only less sets can cover the whole graph).
- Likewise, no number strictly between $\frac{5}{8}$ and $\frac{3}{4}$ can be in the spectrum, since a *second* edge in G already drops $\Pr_{\mathrm{vc}}[G]$ to at most $\frac{5}{8}$.

So, we see that there are some "gaps" in the spectrum near 1, namely the empty intervals $(\frac{3}{4}, 1)$ and $(\frac{5}{8}, \frac{3}{4})$. Rephrased in terms for the following definition of the "spectral gap below p," we have $\text{spectral-gap}_{\mathrm{vc}}(1) = \frac{1}{4}$ and $\text{spectral-gap}_{\mathrm{vc}}(\frac{3}{4}) = \frac{1}{8}$:

Definition 4.2. $\text{spectral-gap}_{\mathrm{vc}}(p) := \sup\{\epsilon \geq 0 \mid (p - \epsilon, p) \cap \textsc{vc-pr-spectrum} = \emptyset\}$.

However, these gaps must rapidly get smaller as we get near to $\frac{1}{2}$: Each star (that is, each $K_{1,k}$) has $\Pr_{\mathrm{vc}}[K_{1,k}] = \frac{1}{2}(1 + 2^{-k})$ as we can cover a star either by picking the central vertex or by picking all other vertices (which happens with probability 2^{-k}); so any gaps below different $p > \frac{1}{2}$ in the spectrum get smaller and smaller as p approaches $\frac{1}{2}$. For instance, $\text{spectral-gap}_{\mathrm{vc}}(\frac{1}{2} + 2^{-1000}) \leq 2^{-1001}$.

Formally, $1/2$ is an accumulation point of $\textsc{vc-pr-spectrum}$ and it is tempting to assume that the spectrum below $1/2$ is simply a dense set as we have reached an accumulation point. However, something surprising happens: Even though *above* $1/2$ the gaps got smaller and smaller, directly *below* $1/2$ there is a sizable spectral gap once more!

Lemma 4.3. $\text{spectral-gap}_{\mathrm{vc}}(\frac{1}{2}) \geq 2^{-10}$.

Proof. Recall that Algorithm 2 from the previous section internally precisely computed the number $\#_{\mathrm{vc}}(G)$ for graphs $G \in \textsc{maj-vc}$. A closer look at the algorithm reveals two things:

1. We actually precisely compute $\#_{\mathrm{vc}}(G)$ for any graph G with $\Pr_{\mathrm{vc}}[G] > \frac{27}{64}$ (rather than only for the case "$\cdots \geq \frac{1}{2}$") since all such graphs do not have a matching of size 3 and, hence, pass the test from line 4.
2. There can be at most nine $\beta\colon X \to \mathbb{B}$ that pass the contradiction test in line 11 (as G restricted to X contains a size-2 matching). So *c is the sum of at most nine powers of two.*

To reiterate, for any graph G for which $\Pr_{\mathrm{vc}}[G]$ is "just below $\frac{1}{2}$" we have that $c = \#_{\mathrm{vc}}(G)$ is the sum of at most nine powers of two. But this means that $\Pr_{\mathrm{vc}}[G] = 2^{-i_1} + 2^{-i_2} + \cdots + 2^{-i_9}$ (or less summands, but let us focus on this case for a moment) and *in the binary representation of* $\Pr_{\mathrm{vc}}[G]$ *there are (at most) nine* 1*s*.

All told, for any graph G with $\frac{27}{64} < p := \Pr_{\mathrm{vc}}[G] < \frac{1}{2}$, we have $p = 0.0b_1b_2b_3b_4\cdots_2$ where at most nine of the b_i are 1 (the first bit after "0." must be

0 in order to have $p < \frac{1}{2}$). But the largest possible number with this property is $p = 0.0111111111_2 = \frac{1}{2} - 2^{-10}$ proving the claimed lower bound on the size of the spectral gap below $1/2$. □

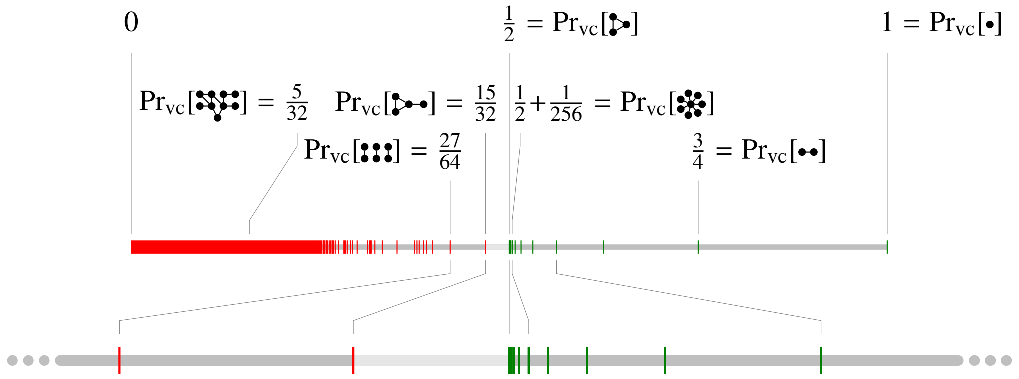

0
$\frac{1}{2} = \mathrm{Pr}_{\mathrm{vc}}[\,]$
$1 = \mathrm{Pr}_{\mathrm{vc}}[\bullet]$
$\mathrm{Pr}_{\mathrm{vc}}[\,] = \frac{5}{32}$
$\mathrm{Pr}_{\mathrm{vc}}[\,] = \frac{15}{32}$
$\frac{1}{2} + \frac{1}{256} = \mathrm{Pr}_{\mathrm{vc}}[\,]$
$\mathrm{Pr}_{\mathrm{vc}}[\,] = \frac{27}{64}$
$\frac{3}{4} = \mathrm{Pr}_{\mathrm{vc}}[\,]$

Figure 4: Visualization of VC-PR-SPECTRUM, the set of all values that $\mathrm{Pr}_{\mathrm{vc}}[G]$ can have for any undirected graph G. Each line represents one possible value, with some example graphs shown that have this value as their cover probability, and red lines indicate values that are less than $\frac{1}{2}$ while green lines corresponds to values at least $\frac{1}{2}$. The lower part is just a zoom where the *spectral gap* between $\frac{15}{32}$ and $\frac{1}{2}$ is easier to see. No graph has a cover probability that lies in this interval, as shown in Lemma 4.3 (for a less tight bound, though).

The bound from the lemma is not necessarily tight as it is not clear whether there actually is a graph G whose satisfaction probability is below $1/2$ and is exactly the largest possible sum $\sum_{i=2}^{10} 2^{-i}$ of nine powers of two with this property. Indeed, it turns out that such a graph does not exist and we will see in the proof of Theorem 7.12 that the spectral gap is, in fact, $\text{spectral-gap}_{\mathrm{vc}}(\frac{1}{2}) = 2^{-5} = 1/32$. So returning to Theoretical Foods Inc. we now see that the extra information volunteered by the employee, namely that the fraction of black rice grains is either at least 50% or at most 46.875%, directly corresponds to a *spectral gap in the spectrum of fractions of black grains below* $1/2$ *of size* $3.125\% = \frac{1}{32}$ and that this is exactly the spectral gap we have below $1/2$ for the fraction of vertex covers of graphs, see also Figure 4.

The natural next question is, of course, what about other values even smaller than $1/2$? Does the spectrum get dense anywhere further down? It does not! But this is perhaps now less surprisingly than earlier:

Theorem 4.4. $\text{spectral-gap}_{\mathrm{vc}}(p) > 0$ *for all* $p \in (0, 1]$.

Proof. In Lemma 4.3 we argued that in the binary representation of $\mathrm{Pr}_{\mathrm{vc}}[G]$ there can be at most nine 1s whenever $\mathrm{Pr}_{\mathrm{vc}}[G] > \frac{27}{64}$ holds. The reason was that such a graph cannot contain a matching of size 3 (as this would lower $\Pr_{vc}[G]$ to at most $\frac{27}{64}$) and at most nine $\beta\colon X \to \mathbb{B}$ can contribute to the sum in equation (1). In the same way, for *any* $p \in (0, 1]$, there will be a constant c such that $(\frac{3}{4})^c < p/2$ holds, so any graph G with $\Pr_{vc}[G] > p/2$ does not contain a matching of size c. This means that we can write $\#_{vc}(G)$ as a sum of at most $j := 3^{c-1}$ powers of two and, hence, the binary representation of $\Pr_{vc}[G]$ contains at most j many 1s. Consider the binary representation $0.b_1b_2b_3b_4\cdots_2$ of p such that infinitely many of the $b_i \in \{0, 1\}$ are 1s (for a number like $p = 1/3 = 0.01010101\ldots_2$ this is automatically the case, but when p is a dyadic number like $3/8$, then we have $3/8 = 0.011_2 = 0.001111111\cdots_2$ and we choose the latter representation). Now consider any graph G with $p/2 < \Pr_{vc}[G] < p$ (if there is no such graph, the spectral gap has size at least $p/2 > 0$ and we are done). Since the binary representation of $\Pr_{vc}[G]$ contains at most j many 1s, the largest number with this property that is still strictly less than p is *the number obtained from p by setting all bits after the jth* 1 *to* 0. Since this number is strictly smaller than p, there is a spectral gap below p as claimed. □

The above theorem is the special case of the *Spectral Well-Ordering Theorem* from [30] for monotone 2CNF formulas. The curious term "well-ordering" comes from the fact we can state Theorem 4.4 equivalently as *"*VC-PR-SPECTRUM *is well-ordered by >,"* but this is really just a different way of stating that there are "spectral gaps below all p".

Applying Random Sampling. With all the preparations done, it is relatively straightforward to apply random sampling to solve MAJ-VC:

Algorithm 3: A random-sampling-based algorithm for deciding MAJ-VC: We sample 2,250 many subsets of V ("grains") and count how many of them are covers ("black grains"). We claim membership in MAJ-VC whenever the number of covers is at least $2{,}250 \cdot \frac{31}{64} > 1{,}089$ many. As shown in the proof of Theorem 4.5, this claim is correct with probability at least 2/3.

```
input $G = (V, E)$
$c \leftarrow 0$
do 2,250 times:
    $C \leftarrow$ random subset of $V$
    if $C$ is a cover of $G$ then
        $c \leftarrow c + 1$

if $c > 1{,}089$ then output "$G \in$ MAJ-VC" else output "$G \notin$ MAJ-VC"
```

Theorem 4.5. *Algorithm 3 shows* MAJ-VC $\in$ BPP.

Proof. By Lemma 4.3 we know that there is a spectral gap of size at least 2^{-10} below $p = 1/2$ in the spectrum of the function $\Pr_{\text{vc}}[\cdot]$ and the more detailed analysis in the proof of Theorem 7.12 shows that the actual size is $\frac{1}{32}$. In particular, for any graph G the probability is at least $\frac{1}{2}$ or at most $\frac{15}{32}$. But, then, plugging the values $p_1 = \frac{1}{2}$ and $\epsilon = \frac{1}{64}$ into inequality (3) and and $p_2 = \frac{15}{32}$ and also $\epsilon = \frac{1}{64}$ into inequality (2) from the Chernoff–Hoeffding bound for $t = 2{,}250$, see Fact 4.1, we get that when we sample t random vertex sets and then check whether at least $\lceil 2{,}250 \cdot (p_2 + \epsilon) \rceil = \lceil 2{,}250 \cdot (p_1 - \epsilon) \rceil = 1{,}090$ of them are covers, our output will be correct with probability at least $2/3$. □

Extensions.

- The correctness of Algorithm 3 for deciding MAJ-VC hinges on the existence of a spectral gap below $1/2$. Since we saw in Theorem 4.4 that spectral gaps "are all over the place," the algorithm can be adjusted to MON-2SAT-PR$_{\geq p}$ for any p just by changing the number $t = 2{,}250$ of trials in accordance with whatever Fact 4.1 requires for a given p and $\epsilon = \text{spectral-gap}_{\text{vc}}(p)/2$ and changing the number 1,089 to $t \cdot (p - \epsilon)$. The fact that we can adapt the algorithm so easily also tells us something about the sizes of the spectral gaps: *They must get pretty small as we approach* 0 since, otherwise, we could use the randomized Algorithm 3 in conjunction with binary search to compute $\Pr_{\text{vc}}[G]$ quickly. (Formally, $\#\text{P} \subseteq \text{FP}^{\text{BPP}}$ would hold.) A more detailed analysis of the sizes of spectral gaps can be found in [30].

- The one place in the algorithm that directly refers to the vertex cover problem is line 5, where we check whether our sample C is a cover. We could replace this check by any other sensible check (like "Is C a hitting set of H?" or "Is β a satisfying assignment of ϕ?" and so on) and we would still get a correct algorithm *as long as there is a spectral gap in the spectrum of the function for which we wish to decide a threshold.* For instance, Algorithm 3 could be used to decide, for instance, 4SAT-PR$_{\geq 63/128}$, as long as $\text{spectral-gap}_{\text{4CNFS}}(63/128) > 0$ holds. By the *Spectral Well-Ordering Theorem* [30] this is, indeed, *always the case* and Algorithm 3 can be used to show kSAT-PR$_{\geq p}$ $\in$ BPP for all $k \geq 1$ and $p \in [0, 1]$.

- On the first two ways, it was trivial to adapt the algorithm so that it also works for MON-2SAT-PR$_{>1/2}$ rather than MON-2SAT-PR$_{\geq 1/2}$ = MAJ-VC. Perhaps surprisingly, trying to adapt our random sampling algorithm fails quite spectacularly. The reason is that while there was a spectral gap *below*

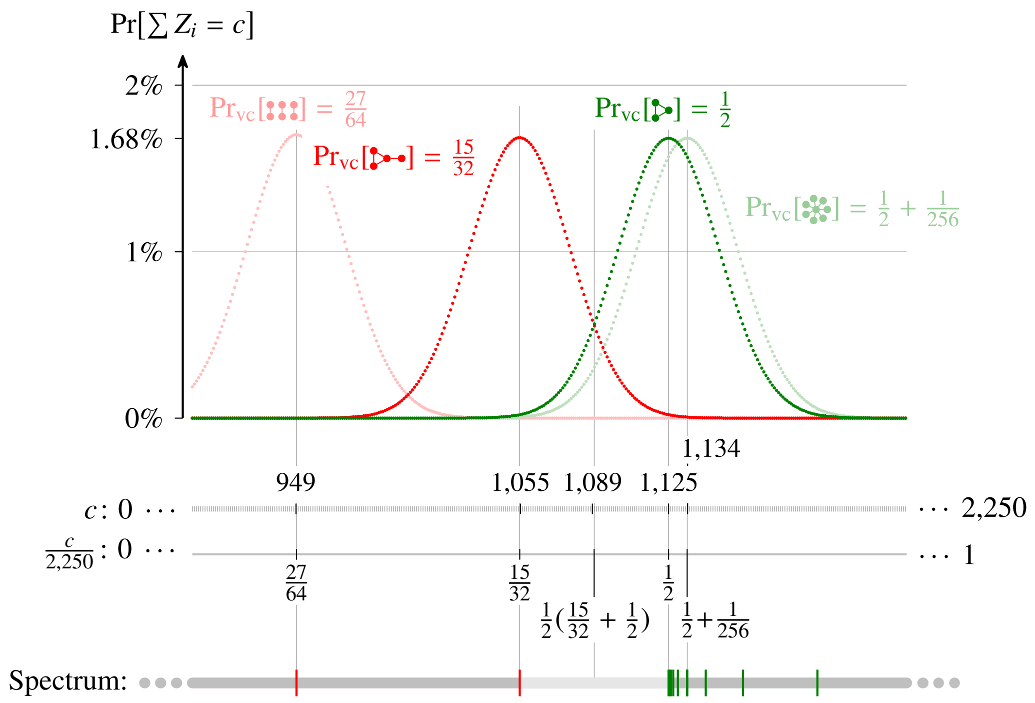


Figure 5: Plots of the probability that we see a certain count c of covers during a run of Algorithm 3: For a given graph G, the algorithm counts in c how many out of 2,250 sampled vertex sets are covers of G (for the ith sampled vertex set, the indicator variable Z_i is then 1). For the graph we have $\Pr_{\text{vc}}[\ ] = \frac{15}{32}$, so if we sample 2,250 random vertex sets, the expected number of covers (the number $\sum_{i=1}^{2{,}250} Z_i$) will be $\frac{15}{32} \cdot 2{,}250 = 1{,}054.6875$. The probability that on a run of the algorithm we get *exactly* 1,054 covers is $\binom{2{,}250}{1{,}054}(\frac{1}{2})^{1{,}054}(1 - \frac{1}{2})^{2{,}250-1{,}054} \approx$ 1.68454%, that we get *exactly* 1,055 covers is about 1.68501%, and that we get any $c > 1{,}089$ is the integral over all values to the right of the central line at 1,089. Plugging in the numbers into Fact 4.1 yields that the probability is less than 1/3 (actually, the integral is even less than 8% as the bound from the fact is not tight). For the green triangle graph, $\Pr_{\text{vc}}[\ ] = \frac{1}{2}$, and the probability that for a random sample of 2,250 vertex sets we have $c > 1{,}089$ is the integral over all dark green probabilities to the right of 1,089. Applying Fact 4.1 for the appropriate values gives that the probability is at least 2/3, and it is actually even more than 93%. At the bottom, the spectrum of possible values that $\Pr_{\text{vc}}[G]$ can attain is shown (see Figure 4 for details) with a prominent *spectral gap* between $\frac{15}{32}$ and $\frac{1}{2}$. Because of this gap, for all G with $\Pr_{\text{vc}}[G] < \frac{1}{2}$ we actually have $\Pr_{\text{vc}}[G] \leq \frac{15}{32}$ (light red graphs) and $c > 1{,}089$ is even less likely than for . For graphs with $\Pr_{\text{vc}}[G] \geq \frac{1}{2}$ (light green graphs), $c > 1{,}089$ is even more likely than for .

1/2, there is *none above* (see Figure 4). Indeed, no matter how many samples we take, we will never be able to differentiate with high confidence between the distribution of c-values generated by a triangle (which is not an element of $\textsc{mon-2sat-pr}_{>1/2}$) and the distribution generated by $K_{1,k}$ for a large k (which is an element of $\textsc{mon-2sat-pr}_{>1/2}$ for all k). It turns out that this "asymmetry" between the "$\geq p$" and the "$> p$" cases is fundamental: Akmal and Williams [2] show, for instance, that $\textsc{4sat-pr}_{>1/2}$ is NP-complete, while we just saw that $\textsc{4sat-pr}_{\geq 1/2}$ lies in BPP.

5 The Fourth Way: Via Algorithmic Meta-Theorems

The employee of *Theoretical Foods Inc.* thanks you yet again for your help as you take your leave. With the silos of grain behind you, you start to explore a fourth way that quickly leads to a large patch of vegetation that looks nothing like the orderly Search Tree Nursery of your first stop, but more like a jungle: There are still lots of trees, but these are now overgrown by ivy, have lianas hanging from the branches, and you spot mistletoe between the rich leaves. A woman stands next to the winding path through the jungle, wearing some kind of uniform with the inscription *Courcelle and Partners D&C Counting Corp.* and she immediately offers their services: "Do you need help with examining any of these plants? We specialize in answering all sorts of counting problems regarding them!"

With the "plants" corresponding to input graphs, it is a good thing that they need not be trees, but rather can deviate and have additional edges. Formally, we will classify our input graphs according to how "tree-like" they are, a measure known as *tree-width.* Graphs of tree-width 1 are just trees, but already for tree-width 2 there can be all sorts of interesting out-growth and cycles have, for instance, tree-width 2. The higher the tree-width, the more complicated the graphs can be.

You politely ask what "all sorts of counting problems" encompasses and whether it includes, in particular, "counting covers." Delightedly, she answers: "Oh, there are really *a lot* of problems we can solve very efficiently. Counting covers is actually pretty simple for us, so, yes, we can definitely help you with that." Indeed, determining the number of vertex covers of a graph is just one of a great many of problems that can be solved using *divide-and-conquer* on graphs of fixed tree-width.

Just as you are about to commission the *Courcelle and Partners D&C Counting Corp.,* the employee mentions a caveat: "Please be aware that our services become exponentially more expensive as the tree-width increases. Is that a problem for you?" At first, this seems like *quite* a problem to you since all sorts of graphs can appear as input, even cliques with very high tree-width. But then you notice that you do not need the services for graphs of tree-width higher than four: For them, we always have $G \notin$ MAJ-VC.

In the following, we have a closer look at the background of tree-width and the services offered. Then we show how *Courcelle's Theorem* [8] can be used to decide MAJ-VC.

Background on Tree-Width. Suppose we wish to apply the divide-and-conquer method to solve MAJ-VC. It is not immediately clear how that would work on an arbitrary graph, so let us start with trees: Suppose the graph $T = (V, E)$ is a tree (a connected, acyclic, undirected graph). Pick a root $r \in V$ and let its children be $c_1, \dots, c_t \in V$, each of which is the root of a subtree $T_i = (V_i, E_i)$. Now, knowing the numbers $\#_{\text{vc}}(T_i)$ of vertex covers of each T_i, is already helpful for determining $\#_{\text{vc}}(T)$, "but not quite": The product of these numbers is the number of vertex covers of T *that include the root r,* but it does not quite account for those vertex covers of T that *miss r.* Of course, for these, all children c_i of r must be in the cover (since all the edges from r to its children must be covered), but we do not know how many covers have these properties. The trick is to compute *two* values for each graph T in a recursion: The number of covers C of T *with* $r \in C$ and the number of covers *with* $r \notin C$. (By a similar argument as in the section on search trees, the first number is $\#_{\text{vc}}(T \setminus \{r\})$ and the second is $\#_{\text{vc}}(T \setminus N[r])$.) It is then not too hard to devise an algorithm that recursively computes $\#_{\text{vc}}(T)$ for any tree by computing these two numbers for each subtree.

Most graphs are not trees, so we can ask whether and how this idea can be taken any further. It turns out that there is a generalization of trees, namely *graphs of bounded tree-width,* for which a similar idea works: We also do a recursion on a tree (called the *decomposition tree of the graph*), but now instead of two values, we compute 2^{w+1} values for each node of the graph, where w is the tree-width of the graph (and which is obviously bounded for graphs of "bounded tree-width").

A bit more formally, given an undirected graph $G = (V, E)$, a *tree decomposition of G* is a tree $T = (N, F)$ (whose vertices will be called *nodes* to better distinguish them from the *vertices* of G) together with a mapping $bag\colon N \to \{U \mid U \subseteq V\}$ that assigns a *bag* to each node of the tree such that the following conditions hold:

1. *Covering condition:* For each $e \in E$ there is a node $n \in N$ such that $e \subseteq bag(n)$, that is, such that the two endpoints of the edge lie in the bag.
2. *Connectedness condition:* For each vertex $v \in V$, the set $\{n \in N \mid v \in bag(n)\}$, which contains all tree nodes whose bags contain v, is connected in T.

The *width* of a tree decomposition is the maximum size of any bag minus one, that is, $\max_{n \in N} |bag(n)| - 1$. The *tree-width of a graph G* is the minimal width of any tree decomposition of G, see Figure 6 for an example.

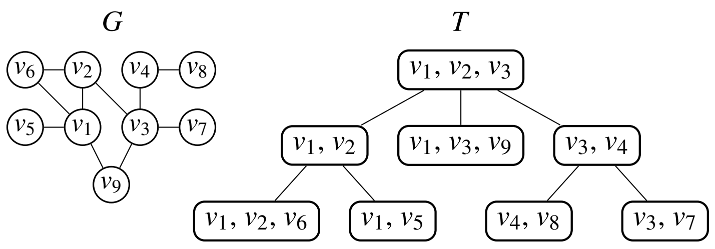


Figure 6: A graph G together with a tree decomposition T for it of width 2 (maximum bag size minus one). In the visualization, the bags attached to the nodes of T are just shown directly inside the nodes. Observe how each edge of G, such as the edge $\{v_1, v_6\}$, is a subset of some bag of T (namely the lower left bag $\{v_1, v_2, v_6\}$) and how for each vertex of G, such as v_2, the nodes of T containing v_2 (namely those on the path from the root to the lower left node) are connected in T. The tree-width of G is 2: The depicted tree decomposition T yields an upper bound of 2 and no tree decomposition of G can have width 1 as it is easy to see that a clique of G, such as $\{v_1, v_2, v_6\}$, must be a subset of some bag of any tree decomposition.

Returning to counting covers, given a tree decomposition $T = (N, F)$ of an input graph G, we can associate with every node $n \in N$ an induced subgraph G_n of G containing all vertices "mentioned in any bag of a node below n in T" (for the notion of "below n" to make sense, we must pick a root of T, so let us assume that we have done this). For instance, in Figure 6 for the left child n of the root (the node whose bag is $bag(n) = \{v_1, v_2\}$), the graph G_n would encompass the nodes $\{v_1, v_2, v_5, v_6\}$ and have the form . Now, we wish to compute for each node n of the tree T a vector of $2^{|bag(n)|} \leq 2^{w+1}$ numbers, namely *for each possible subset $S \subseteq bag(n)$ the number of covers C of G_n with $C \cap bag(n) = S$*. For instance, for the just-mentioned left child of the root in Figure 6 with $bag(n) = \{v_1, v_2\}$, the four numbers are how many covers of G_n there are that contain v_1 and v_2 (the number is 4), how many contain v_1 but not v_2 (only 2), how many contain v_2 but not v_1 (just 1), and how many contain neither v_1 nor v_2 (the number is 0). It is now possible to compute the 2^{w+1} numbers for any node of the tree, if one knows the numbers for all its children, meaning that we can compute the numbers recursively. This, by the way, explains why *Courcelle and Partners* charge extra for larger tree-width: The cost of keeping track of all these numbers per bag rises exponentially with the tree-width.

There are, of course, several open questions at this point: How, exactly, does the recursion work? How do we obtain a tree decomposition in the first place? And what happens when the tree-width of an input graph is, indeed, large? Fortunately, it turns out that the first two questions can be side-stepped by delegating the work to an *algorithmic meta-theorem* (although the first question is not too hard to answer directly, see for instance [29], but invoking an algorithmic meta-theorem is way cooler). The answer to the third question was already hinted at: Graphs with large tree-width cannot be elements of MAJ-VC.

Background on Algorithmic Meta-Theorems. Let us get the problem of computing tree decompositions out of the way, first: It is not trivial and quite a bit of research has been done on that question. One culmination of this research is the following theorem, whose proof is anything but simple:

Fact 5.1 (Bodlaender's Theorem [5]). *For each tree-width w, there is a linear-time algorithm* BODLAENDER$_w(\cdot)$ *that maps any graph G either to a tree decomposition of G of width w or to the correct output "no width-w tree decomposition exists."*

There are also parallel variants of this result [3] and also a logspace version [12], but the above version is more than enough for our purposes.

Let us now have another look at the divide-and-conquer approach that we sketched to count the number of vertex covers of trees (and which can be generalized to larger tree-width, but let us focus on trees for a moment): Instead of counting the number of vertex covers, we could also use the method to count the number of independent sets – but, of course, that is not very surprising as these numbers are the same. More interestingly, we can use the same idea to count the number of, say, dominating sets: Knowing for each child node how many dominating sets there are that dominate the child and how many there are that dominate everything but the child, suffices to compute these two numbers for a parent node, yielding a recursion – and this, too, generalizes to larger tree-width. Indeed, once one starts investigating this question more closely, literally *hundreds* of counting problems turn out to be amenable to applying divide-and-conquer via tree decompositions. Here is an excerpt from a paper [4] by Bodlaender from 1989:

> **Theorem 4.4** *Each of the following problems is in* NC, *when restricted to graphs with tree width $\leq K$, for constant K: vertex cover [GT1], dominating set [GT2], domatic number [GT3], chromatic number [GT4], monochromatic triangle [GT5], feedback vertex set [GT7], feedback arc set [GT8], partial feedback edge set [GT9], minimum mammal matching [GT10], partition into triangles [GT11], partition into isomorphic subgraphs for fixed H [GT12],…*
>
> 47 (!) further problems
>
> *… maximum length-bounded disjoint paths for fixed J [ND41], maximum fixed-length disjoint paths for fixed J [ND42], chordal graph completion for fixed k, chromatic index, spanning tree parity problem, distance d chromatic number for fixed d and k, thickness $\leq$ k for fixed k, membership for each class C of graphs, which is closed under minor taking.*

Clearly, there must be some property that all (or at least most of) these problems share: It defies credibility that the same method just *happens* to work for all of them without some underlying reason. The honour of identifying this property is due to Bruno Courcelle, who observed that all of these problems can be *described in monadic second-order logic.*

In order not to get (further) side-tracked, no detailed account of how this logic works will be given (and it will not be important for our algorithms), a few simple examples will have to suffice (see for instance [29] for a detailed account): The idea is to interpret graphs as finite logical structures in the sense of predicate logic with the structure's universe being the vertices and interpreting the edge relation as a binary relation $\sim$. The subsets $C \subseteq V$ for which we wish to check whether they are covers now correspond to a set variable C. The statement "C is a vertex cover of G" gets translated to "(the logical structure corresponding to) G is a model of $\eta_{\text{VC}}(C) = \forall x \forall y (x \sim y \to (C(x) \lor C(y)))$" (we use η for formulas of predicate logic to avoid confusion with the ϕ used for *p*ropositional logic). As further examples, the statement "C is an independent set in G" gets translated to "G is a model of $\eta_{\text{IS}}(C) = \forall x \forall y (x \sim y \to (\neg C(x) \lor \neg C(y)))$"; and "$C$ is a dominating set of G" gets translated to "G is a model of $\eta_{\text{DS}}(C) = \forall x \exists y (C(y) \land (x = y \lor x \sim y))$."

Fact 5.2 (Courcelle's Theorem, Counting Version, [1])**.** *Let $\eta(C)$ be a monadic second-order formula with a free set variable C. Then for each tree-width w, there is a polynomial-time algorithm* COURCELLE-COUNT$_{\eta,w}(\cdot,\cdot)$ *that on input of any graph $G = (V, E)$ together with a width-w tree decomposition of G outputs the number of subsets $S \subseteq V$ such that $G \models \eta(S)$, that is, such that G is a model of η when S is assigned to the variable C.*

The above theorem is (one of many, many) *algorithmic meta-theorems* where the "meta" comes from the fact that for each $\eta(C)$ we get a new algorithm. Other meta-theorems differ in the logic used (some allow only less powerful logics like first-order logic, some allow more powerful ones), in the allowed class of graphs (some allow only more restricted graphs than graphs of fixed tree-width, other allow much more general graphs), and in the amount of resources used (some need only constant time, some need exponential time). It also makes a difference whether *counting* problems are considered (like above) or *decision* problems ("is there an S with $G \models \eta(S)$?") or *construction* problems ("find an S with $G \models \eta(S)$").

Applying Algorithmic Meta-Theorems.

Theorem 5.3. *Algorithm 4 shows* MAJ-VC $\in$ P.

Proof. The algorithm is a straightforward application of Courcelle's Theorem, Fact 5.2, and the only thing we have to show that the output in line 4 is correct:

Algorithm 4: Using Courcelle's Theorem to decide MAJ-VC: We first compute a tree decomposition of width 4 of an input graph G using Bodlaender's algorithm. This may result in a failure when G actually has a larger tree-width than 4, but, then, we can output that $G \notin$ MAJ-VC holds (see the proof of Theorem 5.3 for why this is correct). Otherwise, we have a tree decomposition of G of small width and can apply a counting version of Courcelle's powerful theorem to obtain the number of vertex covers of G.

```
input $G = (V, E)$

$T \leftarrow$ BODLAENDER$_4(G)$ // see Fact 5.1
if $T$ is ``no width-4 tree decomposition exists'' then output ``$G \notin$ MAJ-VC'' and stop

$c \leftarrow$ COURCELLE-COUNT$_{\forall x \forall y (x \sim y \to (C(x) \lor C(y))),4}(G, T)$ // see Fact 5.2
if $c \geq 2^{|V|}/2$ then output ``$G \in$ MAJ-VC'' else output ``$G \notin$ MAJ-VC''
```

We must argue that when the tree-width is larger than 4, then $G \notin$ MAJ-VC holds; or, by contraposition, that all $G \in$ MAJ-VC have tree-width at most 4. For this, let $G \in$ MAJ-VC be given. By Lemma 2.1, G contains no matching of size 3. Then G cannot contain any paths of length 6 since five edges $\{u_1, u_2\}$, $\{u_2, u_3\}$, $\{u_3, u_4\}$, $\{u_4, u_5\}$, $\{u_5, u_6\}$ for six different vertices u_i would yield $\{\{u_1, u_2\}, \{u_3, u_4\}, \{u_5, u_6\}\}$ as a matching in G. So, all paths in G have length 5 or less. By Lemma 5.4 below, the tree-width of G is at most 4. □

Lemma 5.4. *Let G be a graph in which all paths have length at most l. Then the tree-width of G is at most $l - 1$.*

Proof. We may assume that G is connected. The tree decomposition T will be a depth-first search tree of $G = (V, E)$ starting at some arbitrary root vertex r, see Figure 7 for an example. Assign bags to the nodes $n \in V$ of T as follows: Let $bag(n)$ be the set of vertices on the path from r to n in T. We claim that this yields a tree-decomposition of width $l - 1$ of G:

1. The *covering condition* is satisfied since for any edge $\{u, v\} \in E$, in any depth-first search tree, the vertices u and v lie on a path from r to some leaf of T.

2. The *connectedness condition* is satisfied since for any vertex $v \in V$, exactly the nodes in the subtree rooted at n contain v.

3. Concerning the width, note that the depth of T (the longest path from r to any leaf) is at most l since this longest path is also a path in G. Thus, all bags have size at most l and the width is $l - 1$. □

(The lemma actually even shows that the tree-*depth* of G is at most l, but this concept will only be introduced and needed on a later way.)

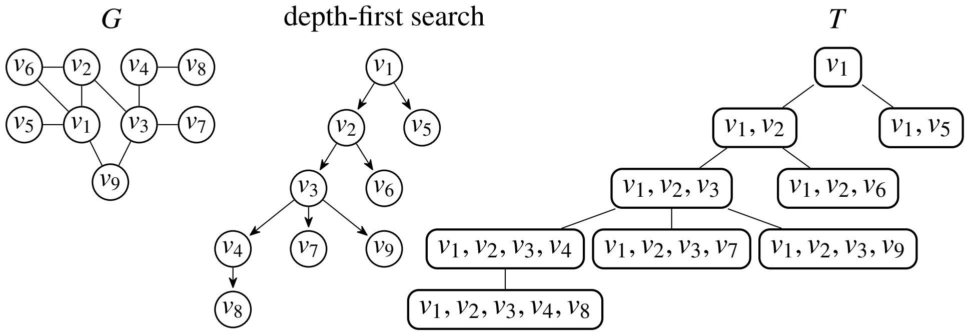


Figure 7: A graph G, the result of a depth-first search (DFS) starting at v_1, and the resulting tree decomposition T of G where each bag for a vertex/node n contains all vertices of G that lie on the path from n to the root v_1 in the DFS tree. Note that the width of T is $4 = |\{v_1, v_2, v_3, v_4, v_8\}| - 1$ and this is "caused" by the path $v_1 \to v_2 \to v_3 \to v_4 \to v_8$ in G.

Extensions.

- As for the previous ways, it is easy to generalize the presented ideas to MON-2SAT-PR$_{\geq p}$ for arbitrary p: All that needs to be changed is the bound on the tree-width in line 4. The reason is that when $\Pr_{\text{vc}}[G] \geq p$ holds, then G cannot contain large matchings (see Lemma 2.3) and hence no long path and hence cannot have large tree-width (see Lemma 5.4).

- Generalizing the approach via algorithmic meta-theorems to 2SAT-PR$_{\geq p}$ and also to 2SAT-PR$_{>p}$ is also pretty easy since, as hinted at, monadic second-order logic is extremely powerful (we just saw a small part of its power in the simple examples given) and it is no problem to encode negations and their behaviours into the formulas and logical structures. The arguments concerning the tree-width are not changed when negations are introduced.

- When we try to generalize our approach to kSAT-PR$_{\geq p}$ for $k \geq 3$, we run into a problem already for MAJ-3HS = MON-3SAT-PR$_{\geq 1/2}$. The trouble is not the algorithmic meta-theorem since it is quite simple to adjust the predicate logic formula $\eta(C)$ to cover hitting sets for 3-hypergraphs. The problem is that the tree-width of 3-hypergraphs (more precisely, the tree-width of the Gaifman graphs of these hypergraphs, but this is nitpicking) that are hit by at least half of all vertex sets *need no longer be bounded:* Just take any undirected graph $G = (V, E)$ of large tree-width and form $H = (V \cup \{a\}, \{e \cup \{a\} \mid e \in E\})$ where a is a fresh vertex. Then at least half of all subsets of $V \cup \{a\}$ are hitting sets of H (namely all that contain a), but the tree-width of H is that of G plus one and, thus, large.

6 The Fifth Way: Via Graph Minors

Having politely thanked *Courcelle and Partners* for their services and having written a 5-star online review in which you praised their modest fee for tree-width 4, you continue on a fifth way. After some time you draw nearer to a building of such *gigantic* proportions that it seems to try to rival the Lonely Mountain. It is a tower that stretches all the way up into the clouds and perhaps beyond, apparently made from ivory. At the base of the tower you see colorful tents with people milling about who seem to prepare to actually *scale* the tower and, indeed, you spot climbers going up the tower on the outside, each clearly trying to reach a different location. Sometimes, climbers enter the tower through one of the many windows and then come out at the base some time later, smiling broadly. The climbers use oddly formed protrusions on the outside of the tower to help them get from one floor up to the next.

A young man approaches you and poses a somewhat puzzling question: “Hi, I am Alex. Are you also on Team Non-Planar?” Sensing your overall bewilderment, he starts explaining: “Ah, just arrived? Welcome to the Minor Tower.” You remark that the tower is anything but “minor,” which is answered by a chuckle. “Good one! This refers to the ‘minor relation,’ which allows us climbers to get from one of the protrusions (we call them graphs) on a floor to some graph on the next floor. You see those two guys there? That’s Neil and Paul and they give you a target graph and a window that you should reach. Then you start climbing (only upwards) towards that graph. The tricky part is that for each team there are certain waypoint graphs, at least one of which you have to touch first before going up to your final target. Do you see those two graphs on the fifteenth floor that looks a bit like a pentagram ( ) and like a crown ( )?” You can, indeed, spot them. “We on Team Non-Planar have to touch them first. So, if you are not on our team, what team would you like to join? As you can see, there any many, many teams out there!”

It turns out that there is, indeed, a team than you should join (if you feel comfortable scaling towers of Babylonic proportions): Team Minority VC.

Background on Graph Minors. Given an undirected graph $G = (V, E)$, Robertson and Seymour [26] have proposed three simple ways of making the graph, well, simpler:

1. Remove an edge.
2. Remove an isolated vertex.
3. Contract an edge.

Here, contracting an edge $\{u, v\} \in E$ means removing both u and v from the graph together with all adjacent edges, but then adding a new vertex w and joining w to all former neighbours of u and v. If we start with a graph G and apply the above three operations repeatedly and end up with a graph H, we say that *H is a minor of G* and write $H \preceq G$, Figure 8 depicts a typical example. (To be perfectly precise, we *also* say that H is a minor of G if some graph that is isomorphic to H is a minor of G, so we do not care about the names of the vertices.)

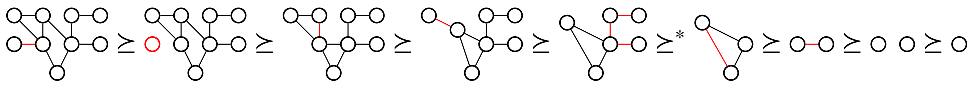


Figure 8: A series of modifications of a graph according to the three possible simplifications allowed by the minor relation: From the first graph to the second graph, the edge indicated in red is removed. Next, the vertex indicated in red is removed. Next, the edge indicated in red is *contracted,* that is, replaced by a vertex at the center of the former edge and all former neighbors of the endpoints are attached to the new vertex. In this way, we can simplify the graph further and further until only a single vertex remains. Any of the graphs in the sequence is a *minor* of all graphs to the left of it. Note that, in particular, any graph has the triangle as a minor iff it contains a cycle.

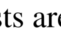
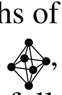
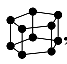
Many classes C of graphs have the property that they are "closed under taking minors," meaning that when a graph is in the class, so is any minor of the graph:

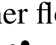
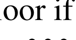
Definition 6.1. C is *closed under taking minors* if $H \preceq G \in \mathrm{C}$ implies $H \in \mathrm{C}$.

For instance, the class PLANAR of planar graphs is closed under taking minors: If you manage to draw a graph in a plane, removing edges or vertices clearly does not change this fact, and contracting an edge can also be done in such a way that the embedding into the plane is not destroyed. Another obvious example is the class FORESTS of all forests. Perhaps a bit less obvious, the classes TREE-WIDTH-w of graphs of tree-width at most w is closed for each w: The key insight is that if you have a width-w tree decomposition of G and apply one of the three simplifications, the tree decomposition is still valid for the simplified graph after possibly renaming some vertices in the bags. Some counterexamples are the class MAX-DEGREE-d of graphs of maximum degree d (as contracting an edge can raise the degree a lot) or CONNECTED (as removing an edge can disconnect a graph).

The great importance of the minor relation lies in the following seminal result:

Fact 6.2 (Robertson–Seymour Theorem [27], Forbidden Minor Formulation)**.** *For every class* C *that is closed under taking minors, there is a finite set* $\{H_1, \dots, H_s\}$ *of graphs, called* forbidden minors, *such that:* $G \in \mathrm{C}$ *iff for all* $i \in \{1, \dots, s\}$ *we have* $H_i \not\preceq G$.

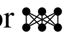
By this theorem, there must be finitely many graphs such that $G \in$ PLANAR holds iff none of these graphs is a minor of G. Such a set does, indeed, exists: By Kuratowski's Theorem [19] it is the set { , }. The forests are characterized by an even simpler set of forbidden minors: { , }. In contrast, graphs of tree-width 3 are characterized [6] by the already much more complex set { , , , }.

The connection to the climbers of Team Non-Planar is as follows: On the outside of the Minor Tower, the graphs are arranged in floors so that climbers can get from one graph G to a graph G' on a higher floor if $G \prec G'$. Then if a climber wants to reach a graph G, but must first go via  or , they can only do so if G is non-planar.

Of course, the results on graph minors presented up to now are purely graph theoretic – we still need a link to algorithmics. This link is provided by the fact that it is possible to efficiently check for a fixed graph H whether for another graph G we have $H \preceq G$:

Fact 6.3 ([17])**.** *For each* H *there is a quadratic-time algorithm for deciding on input* G *whether* $H \preceq G$.

In the parlance of fixed-parameter tractability theory, the minor relation is fixed parameter tractable when the left relation side is the parameter. In parlance of the Minor Tower, we can decide in quadratic time whether one can climb from a fixed waypoint graph to a graph anywhere higher up (and, hence, can also find a path for the climber in polynomial time).

Joining the Robertson–Seymour Theorem and the fixed-parameter tractability of the graph minor relation, we see that we can check in polynomial time whether a graph G is, say, planar simply by testing whether or  is a minor of G; and likewise for any other graph class that is closed under taking minors.

Applying Graph Minors. It should have become clear by now that we can apply the (really, really) powerful machinery of the Robertson–Seymour Theorem whenever a class of graphs is closed under taking minors. The obvious question is, then, what about MAJ-VC? With all the effort and preparations, it is hardly surprising that we will prove in a moment that MAJ-VC is, indeed, closed under taking minors, see Corollary 6.5. Figure 9 depicts how this translates into "Team Minority VC climbing the minor tower." The corollary follows directly from the following stronger result:

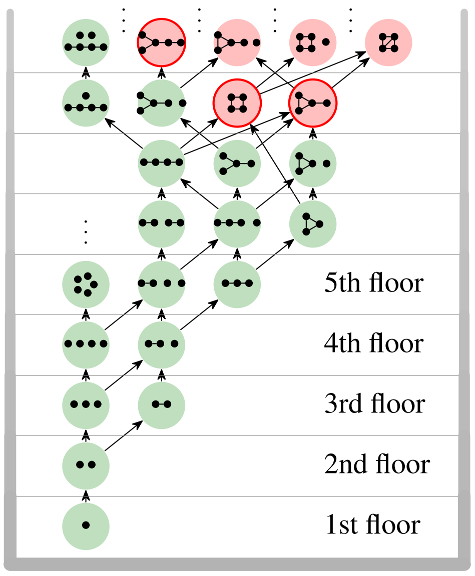
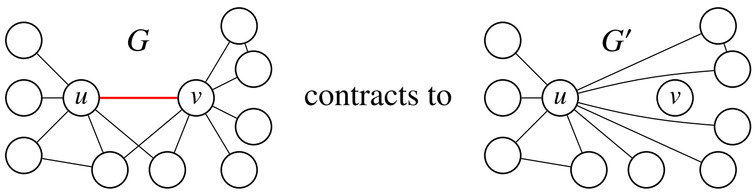


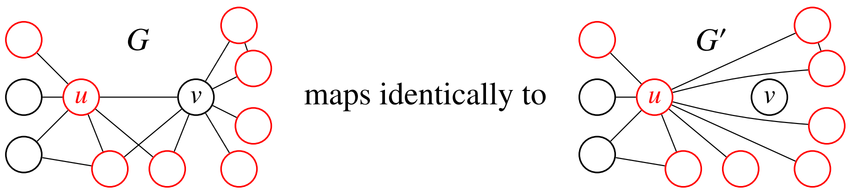
Figure 9: The Minor Tower with the first five floors shown completely (except for graphs having self-loops, to keep it simple). Climbers can get from graphs H on one floor to graphs G on the next floor (or sometimes also one higher up) if these are "in reach" as indicated by arrows. When a climber can get from H to G via several floors, $H \leq G$ holds, that is, H is a minor of G. The graphs in green are in MAJ-VC, the red ones are not. Note how MAJ-VC is "downward closed": Once you have climbed out of it, you cannot get back into it by climbing up. Some red graphs have a red circle around them: They have the special property that *every* red graph can be reached from one of them, see Theorem 6.6.

Lemma 6.4. *Let $G' \preceq G$. Then* $\Pr_{\text{vc}}[G'] \geq \Pr_{\text{vc}}[G]$.

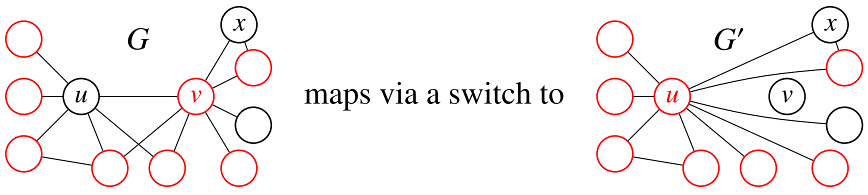
Proof. We show the claim just for the case that $G' = (V', E')$ results from $G = (V, E)$ by a single application of one of the simplification rules, the general statement then follows by induction on the number of simplifications needed (which is always finite). Clearly, removing an edge can only increase the number of vertex covers and removing an isolated vertex does not change the number at all. Thus, let G' result from G through the contraction of an edge $\{u, v\}$. Rather than adding a new vertex after removing u and v, we can also think of such a contraction as disconnecting v from all its neighbors and then making u a neighbor of this former neighborhood:



We claim that there is an injective mapping of covers C of $G = (V, E)$ to covers of $G' = (V, E')$: If C is also a cover of G', map it to itself:



Otherwise, $u \notin C$ and $v \in C$ must hold and there must be some $x \notin C \cup \{u, v\}$ with $\{x, v\} \in E$. Map C to $C' = (C \cup \{u\}) \setminus \{v\}$.

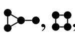
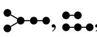
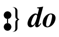


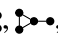
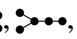
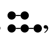
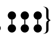
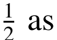
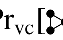
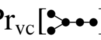
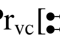
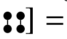
To see that the whole mapping is injective, note that C' was *not* a cover of G since it does not cover $\{x, v\} \in E$. ◻

Corollary 6.5. MAJ-VC *is closed under taking minors.*

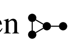
Theorem 6.6. *Algorithm 5 decides* MAJ-VC *in quadratic time.*

Algorithm 5: Any graph property that is closed under taking minors, see Definition 6.1, can be decided in quadratic time with the following algorithm, provided the set of forbidden graph minors in line 3 is replaced appropriately. It is shown in the proof of Theorem 6.6 that this particular set characterizes MAJ-VC.

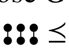
```
input $G = (V, E)$

foreach $H \in \{$ , , , , , , $\}$ do
    if $H$ is a minor of $G$ then // use Fact 6.3 for this check
        output "$G \notin$ MAJ-VC" and halt

output "$G \in$ MAJ-VC"
```

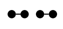
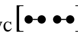
Proof. We know by Corollary 6.5 in conjunction with the Robertson–Seymour Theorem, Fact 6.2, that there must be *some* finite set $F = \{H_1, \dots, H_s\}$ of forbidden minors for the class MAJ-VC. Thus, if we use that set F in line 3, we get a correct decision procedure for MAJ-VC that runs in quadratic time by Fact 6.3. It remains to argue that the particular set $\{$, , , , , , $\}$ is the sought set F. To see this, first note that for all graphs $H \in F$ we have $\Pr_{\text{vc}}[H] < \frac{1}{2}$ as $\Pr_{\text{vc}}[$$] = \frac{1}{4}$, $\Pr_{\text{vc}}[$ $] = \frac{3}{8}$, $\Pr_{\text{vc}}[$$] = \frac{15}{32}$, $\Pr_{\text{vc}}[$ $] = \frac{7}{16}$, $\Pr_{\text{vc}}[$$] = \frac{7}{16}$, $\Pr_{\text{vc}}[$$] = \frac{15}{32}$, and $\Pr_{\text{vc}}[$$] = \frac{27}{64}$. We must now argue that for any graph G with $\Pr_{\text{vc}}[G] < \frac{1}{2}$ we have $H \preceq G$ for one of the $H \in F$. For this, let us go over the possible cases:

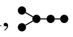
1. Suppose G contains a loop at some vertex v. Then it must contain at least one further edge (otherwise $\Pr_{\text{vc}}[G] = \Pr_{\text{vc}}[$ $] = \frac{1}{2}$ would hold, contradicting the assumption $\Pr_{\text{vc}}[G] < \frac{1}{2}$). If the extra edge is the only other edge and if one endpoint is v, then $G =$ and we would also have $\Pr_{\text{vc}}[G] = \frac{1}{2}$. Thus, the extra edge must be $\{u, w\}$ for $u \neq v$ and $w \neq v$. But then one of the two $H \in \{$, $\} \subseteq F$ is a minor of G.

2. Suppose G contains a triangle $\{a, b, c\}$. Then it must once more contain at least one further edge. If this edge is attached to a or b or c, then  $\preceq G$. Otherwise, the edge must be separate from the triangle. Removing one edge from the triangle yields a path of length 2, showing that $\preceq G$.

3. Suppose G contains a cycle of length 4 or longer. Then $\preceq G$.

4. Suppose G is a forest with at least three trees (each having at least one edge). Then  $\preceq G$.

5. Suppose G is a forest with two trees (each having at least one edge). If both are just an edge, then $G =$  and $\Pr_{\text{vc}}[$$] = \frac{9}{16} \geq \frac{1}{2}$. Otherwise, $\preceq G$.

6. Suppose G is a tree with at least one edge. If G contains a path of length at least four edges, then $\leq G$. Otherwise, G must be a star where at at most one leaf we have added an edge. If G is a star without such an added edge, $\Pr_{\mathrm{vc}}[G] > \frac{1}{2}$. If G is a star with up to two leaves and an edge added to one of them, then G is a triangle or a path of at most three edges and $\Pr_{\mathrm{vc}}[G] \geq \frac{1}{2}$. The only remaining case is that G is a star with at least three leaves, to one of which an edge is added. But, then,  $\leq G$ or $\leq G$.

All told, whenever $\Pr_{\mathrm{vc}}[G] < \frac{1}{2}$, either G contains a cycle and, then, by the first items $H \leq G$ for some $H \in F$, or G is a forest and then, by the last items, we also have $H \leq G$ for some $H \in F$.

For future reference, observe that in all items, except for item 3, we obtain H as a minor of G purely through deleting edges and vertices, we do not need contractions (which we only need to contract long cycles in item 3). □

Extensions.

- Using graph minors algorithmically is a very powerful method, provided the graph classes that we try to decide are closed under taking minors. We saw in Corollary 6.5 that MAJ-VC has this closure property since by Lemma 6.4 the minor relation is (anti)monotone with respect to covering probabilities. This implies, immediately, that for any $p \in [0, 1]$ the sets MON-2SAT-PR$_{\geq p}$ are also closed and, hence, have a forbidden minor characterization.

- Perhaps more intriguingly, the sets MON-2SAT-PR$_{>p}$ are *also* closed under taking minors and for the same reason. Thus, the graph minor method naturally also applies to these sets.

- Unlike all methods presented earlier, there is no obvious way to extend the approach to also cover 2SAT-PR$_{\geq 1/2}$, that is, to handle negations. It is clear that we cannot just ignore the negations, but would have to come up with some sort of gadget constructions to turn formulas into graphs. However, it seems that contractions are anathema to satisfaction probabilities: Consider a set of 2CNF clauses that express that a large cycle of even length is 2-colorable. But, now, contracting (perhaps the representation of) a clause results in a formula expressing that a large cycle of odd length is 2-colorable (which is not the case). Contracting another clause yields a statement about even cycles once more, meaning that the satisfaction probabilities of the graphs representing the formulas seem to fluctuate wildly when we apply contractions.

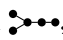
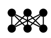
- Another indication why it is unlikely that we can extend the method to decide, say 4SAT-PR$_{\geq 1/2}$, is that if we could map all 4CNF formulas ϕ to graphs G_ϕ

such that $G_\phi \preceq G_\psi$ implies $\Pr[\phi] \geq \Pr[\psi]$, then our approach would not only work for 4SAT-PR$_{\geq 1/2}$, but also for 4SAT-PR$_{>1/2}$, which is known to be an NP-complete problem.

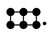
- Whether or not we can use the method to solve MAJ-3HS or, more generally, MON-kSAT-PR$_{\geq p}$, is open.

7 The Sixth Way: Via Forbidden Subgraphs

As you embark on a sixth way, you leave the mighty Minor Tower and the bustling crowd of climbers behind you. The way leads you through some beautiful theory meadows and then towards what can only be called a fairy tale castle with *a lot* of spires of different sizes. Indeed, you soon loose track of how many spires there are. At the gate of the castle, which comes with a moat and a drawbridge, a guard challenges you: “Well met, stranger. What is thy quest?” A bit unsure, you respond that you wish to solve MAJ-VC, to which the guard proclaims: “A most noble quest! Be welcome to the Castle of Thousand Spires and enter. In the Spire of Most Shallow Depths thou shalt prevail.” You thank the guard and enter.

After a bit of searching you find the Spire of Most Shallow Depths. Inside, a festive crowd of twelve dwarves makes quite a ruckus as they try to climb the spire from the inside, seldom getting very far and soon dropping back to the floor, much to the merriment of the rest. In the middle, an exasperated wizard futilely tries to impose some kind of order: “No, no, no! You must climb to *before* you go on to , not the other way round!” As he sees your approach, he stops berating the dwarves and addresses you instead: “Well met, Master Theoretician!” You politely ask what is going on, to which the wizard answers: “These fine dwarves have set it upon themselves to train for the Climb of the Minor Tower, which itself is just training for climbing the Lonely Mountain. As you can see, in each spire of the Castle of Thousand Spires one can safely train climbing as getting from one graph to one on the next floor is much easier than on the Minor Tower. Once you have mastered a spire you can go to another one, they get bigger and bigger with more and more graphs in them (the ones in the Spire of Most Shallow Depths are, well, shallow). Otherwise, it is the same as with the Minor Tower: Go to the waypoints first, then to your target. Speaking of which: No, no, no, you have to touch , not !” You get a feeling that the dwarves will need to practice a *lot,* before they can ever scale the Minor Tower, let alone the Lonely Mountain. So, you casually ask why they do not use the backdoor that the hobbit showed you. Your remark is met by stunned silence, followed by a cry along the lines of “The hobbit has found the secret backdoor! It cannot be! But what, if it true? We must meet him at once!”, followed by twelve dwarves rushing out of the Spire of Most Shallow Depths. The wizard seems deeply lost in thought, but you think you hear him mumbling: “The hobbit must have used a ring to find the backdoor. Perhaps even the nilpotent ring? I must study this!”

Let us now have a look what a “simpler way of scaling the Minor Tower” might look like and why we need many spires instead of a single tower.

Background on Induced Subgraphs. Recall from the previous section that a graph H is a minor of a graph G if we can obtain H from G by deleting edges, deleting isolated vertices, and contracting edges (and renaming vertices). If we only allow deleting edges and deleting isolated vertices, but no longer allow the contraction of edges, then H is (isomorphic to a) *subgraph* of G, written $H \subseteq G$. If we do not even allow deleting edges, but just allow deleting (no longer necessarily isolated) vertices, H is (isomorphic to an) *induced subgraph* of G, written $H \sqsubseteq G$ in the following. For isomorphic graphs H and G, we have $H \subseteq G \subseteq H \sqsubseteq G \sqsubseteq H$. For concrete examples, $\sqsubseteq$ and thus also $\subseteq$ and $\preceq$ . In contrast, $\not\sqsubseteq$, but $\subseteq$ and thus $\preceq$. Finally, $\not\sqsubseteq$ and $\not\subseteq$, but still $\preceq$.

Checking for a fixed H whether $H \sqsubseteq G$ holds, is easy enough, as the following analogue of Fact 6.3 shows:

Lemma 7.1. *For each $H = (V_H, E_H)$ there is an algorithm that on input $G = (V, E)$ decides both $H \sqsubseteq G$ and $H \subseteq G$, and that runs in time $O(|E| \cdot |V|^{|V_H|}) = |V|^{O(1)}$.*

Proof. Iterate over all size-$|V_H|$ vertex set of G and check for each whether it induces H (or a supergraph of H for $\subseteq$). □

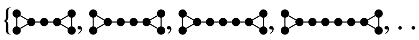
Intuitively, it feels much easier to check whether $H \sqsubseteq G$ holds than to check $H \preceq G$. But Fact 6.3 tells us that the test $H \preceq G$ is fixed-parameter tractable with respect to the parameter H, while testing $H \sqsubseteq G$ is easily seen to be W[1]-hard with respect to the same parameter (as setting $H = K_k$ to size-k cliques shows that the problem is at least as hard as the parameterized clique problem). So, the intuition is wrong.

Nevertheless, working with induced subgraphs or just subgraphs instead of minors has a *big* advantage: Results on the subgraph relation will generalize naturally to 2cnf and partly even to $k$cnf formulas. For instance, $H \subseteq G$ trivially implies $\Pr_{\text{vc}}[H] \geq \Pr_{\text{vc}}[G]$ (less edges can only mean more covers), so we have:

Lemma 7.2. maj-vc *is downward closed with respect to $\sqsubseteq$ and $\subseteq$.*

Crucially, we likewise have that if every clause of a $k$cnf formula ϕ is also a clause of another cnf formula ψ (after possible variable renaming), then $\Pr[\phi] \geq \Pr[\psi]$. In other words, it is easy to come up with generalizations of the above lemma for CNF formulas instead of graphs. In contrast, while by Lemma 6.4 we also had that $H \leq G$ implies $\Pr_{\mathrm{vc}}[H] \geq \Pr_{\mathrm{vc}}[G]$, that implication immediately breaks down when we try to add negations.

Background on Forbidden Induced Subgraphs. In order to "switch" from the minor relation to the induced subgraph relation for solving MAJ-VC, we need an analogue of the Robertson–Seymour Theorem. That is, we need a theorem telling us something like this: "Whenever a graph class C is downward closed under $\sqsubseteq$, then there is a finite set F of graphs such that $G \in \mathrm{C}$ iff $H \not\sqsubseteq G$ holds for all $H \in F$." This statement, however, is false, as the class C of all graphs that do not contain any graph in the following set I as an induced subgraph shows (note how no graph in the infinite I is a subgraph of any other):

$I = \{$  $, \dots\}.$ (4)

Fortunately, Nešetřil and Ossona de Mendez [24] came up with a way to save the idea: We need to restrict attention only to graphs with constant *tree-depth,* as in the Spire of Most Shallow Depths, which is defined as follows:

Definition 7.3. The *tree-depth* of a graph $G = (V, E)$ is as follows: When G is the empty graph, it is 0. When G is unconnected, it is the maximum tree-depth of G's components. When G is connected, it is 1 plus the minimum tree-depth of $G \setminus \{v\}$ taken over all $v \in V$.

Since the definition is in terms of recursively eliminating vertices and results in an *elimination tree,* the tree-depth is sometimes also known as *(recursive) elimination depth.* It is closely related to concepts we saw earlier, namely DFS trees and tree-*width,* see Figure 10.

Let $\mathrm{SPIRE}_t = \{G \mid G$ has tree-depth at most $t\}$. Figure 10 shows an example of a graph $G \in \mathrm{SPIRE}_4$ as the depth of the depicted elimination tree is 4. Nešetřil and Ossona de Mendez [24] proved an analogue of the Robertson–Seymour Theorem for the graphs in each SPIRE_t and for the induced subgraph relation $\sqsubseteq$, see Theorem 7.4 below. Since we wish to generalize the underlying ideas on the seventh and final way, let us have a closer look at the proof of the theorem.

Theorem 7.4 ([24]). *Let* $\mathrm{C} \subseteq \mathrm{SPIRE}_t$ *be a graph class that is downward closed under* $\sqsubseteq$. *Then there is a finite set* F *of graphs such that for all* $G \in \mathrm{SPIRE}_t$ *we have:* $G \in \mathrm{C}$ *holds iff* $H \not\sqsubseteq G$ *for all* $H \in F$.

Proof. We start with a definition:

Definition 7.5. A sequence $(G_0, G_1, G_2, \dots)$ of graphs, also written as just (G_i), is *good,* if $G_i \sqsubseteq G_j$ holds for some indices $i, j \in \mathbb{N}$ with $i < j$; otherwise it is *bad.*

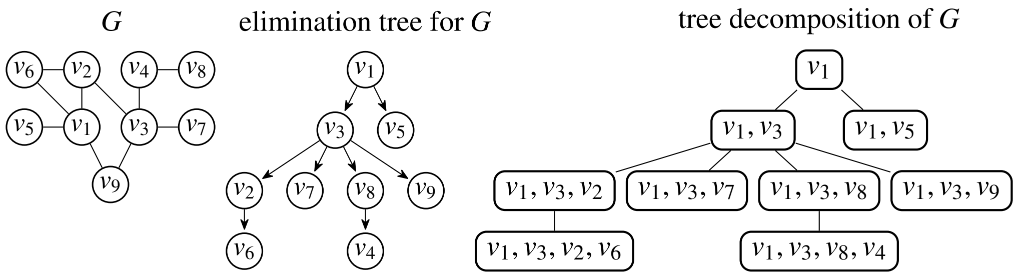


Figure 10: The graph G from Figure 7 on page 33 together with an elimination tree: When we remove the vertex v_1, we get two connected components (one with just v_5 and the other with the remaining vertices). Then, removing v_3 from the large component leaves us with four components ($\{v_2, v_6\}$, $\{v_4, v_8\}$, $\{v_7\}$, $\{v_9\}$). Note that the depicted elimination tree is *not* a DFS tree (there is no edge between v_1 and v_3 is G), but every DFS tree like the one from Figure 7 is an elimination tree (elimination trees of a graph can be more shallow than any DFS trees for the graph). Just as for DFS trees, every elimination tree provides us with a tree decomposition of the graph by putting all vertices along the paths from the root to a node into the node's bag.

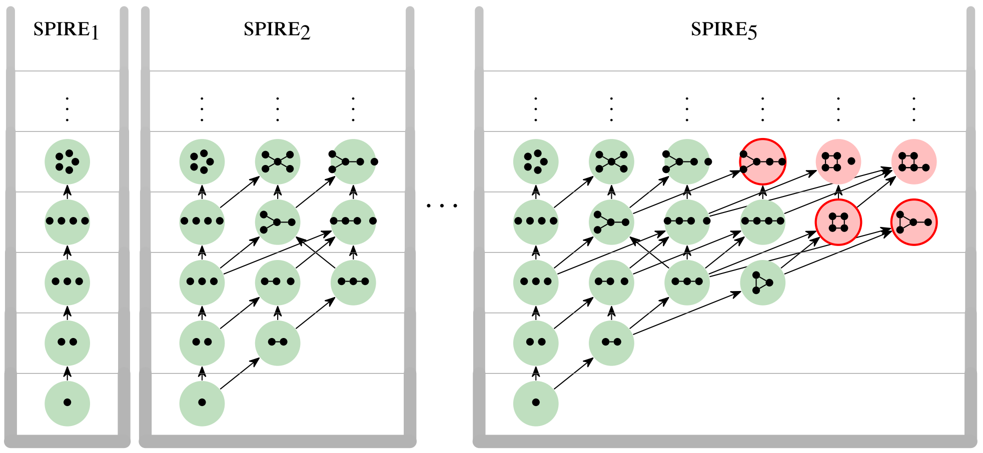


Figure 11: The three spires SPIRE$_1$, SPIRE$_2$, and SPIRE$_5$ depicted as in Figure 9, so green graphs are in MAJ-VC while red graphs are not, but now an arrow from H to G means that $H \sqsubseteq G$, that is, H is an induced subgraph of G (and not only a minor). In the first spire, for tree-depth 1, there are only edge-less graphs. In the second, the graphs are forests of stars. Both spires miss elements of MAJ-VC like the triangle. In SPIRE$_5$, we have MAJ-VC $\subseteq$ SPIRE$_5$. As in Figure 9, minimal elements of the complement of MAJ-VC are indicated by a red circle (but now minimality is with respect to $\sqsubseteq$ rather than the minor relation $\preceq$).

The importance of this definition lies in the following claim:

Claim 7.6. *If every sequence (G_i) with $G_i \in \textsc{spire}_t$ is good, then there is a finite set F such that for all $G \in \textsc{spire}_t$ we have: $G \notin \mathrm{C}$ holds iff $H \sqsubseteq G$ for some $H \in F$.*

Proof. Let F contain all $H \in \textsc{spire}_t \setminus \mathrm{C}$ for which there is no $H' \sqsubseteq H$ with $H' \in \textsc{spire}_t \setminus \mathrm{C}$ and $H \not\sqsubseteq H'$. Consider any $G \in \textsc{spire}_t$. Clearly, if $G \in \mathrm{C}$, no $H \in F$ exists with $H \sqsubseteq G$ as C is downward closed. So suppose $G \notin \mathrm{C}$. If $G \in F$, then there is an $H \in F$ (namely G) with $H \sqsubseteq G$. Otherwise, there is a $G' \sqsubseteq G$ with $G' \in \textsc{spire}_t \setminus \mathrm{C}$ and $G \not\sqsubseteq G'$. If $G' \in F$, then there is an $H \in F$ (namely G') with $H \sqsubseteq G$. Otherwise, there is a $G'' \sqsubseteq G$ with $G'' \in \textsc{spire}_t \setminus \mathrm{C}$ and $G'' \not\sqsubseteq G'$; and $G'' \in F$ once more implies that there is an $H \in F$ with $H \sqsubseteq G$. We need to repeat this argument only a finite number of times since, otherwise, $(G, G', G'', G''', \dots)$ would form a bad sequence. □

By the claim, in order to prove the theorem, it suffices to show that every sequence in $\textsc{spire}_t$ is good – and this is exactly Claim 7.11 below. To prove it by induction on t, we need a slight strengthening: Let us generalize the concepts to *colored* graphs $G = (V, E, c)$, where there is a fixed set C of colors and the function $c\colon V \to C$ assigns a color to each vertex. A colored graph H is an induced subgraph of a colored graph G if we still can obtain H (with the correct vertex colors and with, possibly, some vertex renamings) from G by deleting vertices. We can now reach our goal through a sequence of smaller claims:

Claim 7.7. *Suppose every sequence (G_i) of connected colored graphs $G_i \in \textsc{spire}_t$ is good. Then for every such sequence there is an infinite sequence $(i_0, i_1, i_2, \dots)$ of strictly increasing indices such that $G_{i_0} \sqsubseteq G_{i_1} \sqsubseteq G_{i_2} \sqsubseteq \cdots$.*

Proof. Suppose no such subsequence exists for (G_i). Then starting at $G = G_0$ there must be a subsequence $G \sqsubseteq G' \sqsubseteq G'' \sqsubseteq \cdots \sqsubseteq G^*_0$ of maximal length. If we remove the initial segment up to G^*_0 from (G_i), then starting at the new first graph G, there is once more a subsequence $G \sqsubseteq G' \sqsubseteq G'' \sqsubseteq \cdots \sqsubseteq G^*_1$ of maximal length. By once more removing the initial segment up to G^*_1 and repeating the argument indefinitely, we get an infinite subsequence (G^*_i). But since this sequence is good, $G^*_i \sqsubseteq G^*_j$ for some $i < j$, contradicting that the sequence ending at G^*_i had maximal length. □

Claim 7.8. *Let (G_i) be a sequence of colored graphs, where each G_i has just one vertex. Then (G_i) is good.*

Proof. After at most $|C|$ steps we see the same colored graph once more. □

Claim 7.9. *Suppose every sequence (H_i) of* connected *colored graphs $H_i \in \textsc{spire}_t$ is good. Then so is any sequence (G_i) of (possibly unconnected) colored graphs $G_i \in \textsc{spire}_t$.*

Proof. If there are any bad sequences (G_i) of colored graphs in $\textsc{spire}_t$, consider those for which G_0 has a minimal number of connected components, and then among those for which G_1 has a minimal number of connected components, then among those for which G_2 has a minimal number, and so on. No G_i can be empty (as $G_i \sqsubseteq G_{i+1}$ would follow and show that the sequence is good), so each G_i is the disjoint union of a connected H_i and some rest graph R_i, both of which lie in $\textsc{spire}_t$. By assumption, (H_i) is good and by Claim 7.7 we have $H_{i_0} \sqsubseteq H_{i_1} \sqsubseteq H_{i_2} \sqsubseteq \cdots$ for some indices $i_0 < i_1 < i_2 < \cdots$. Consider the sequence

$$(G_0, G_1, \dots, G_{i_0-1}, R_{i_0}, R_{i_1}, R_{i_2}, \dots).$$

Since R_{i_0} has one connected component less than G_{i_0}, this sequence cannot be bad as we would have picked it at the beginning. But then either for one of the initial G_i we have $G_i \sqsubseteq R_{i_j}$ and then also $G_i \sqsubseteq G_{i_j}$ with $i < i_j$, or we have $R_{i_j} \sqsubseteq R_{i_k}$ for $i_j < i_k$ and then also $G_{i_j} \sqsubseteq G_{i_k}$. In either case, (G_i) was not bad. □

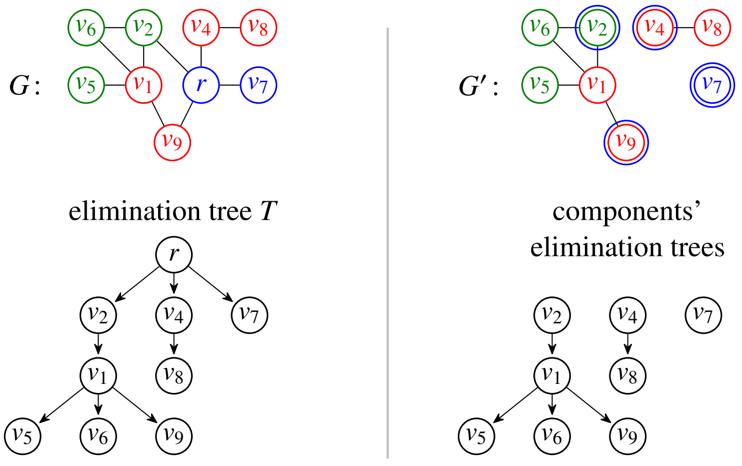


Figure 12: The graph G from Figure 10 but with a different elimination tree (which happens to also be a $\textsc{dfs}$ tree, but this is not relevant). The tree's depth is 4 and hence $G \in \textsc{spire}_4$. An example coloring for $C = \{\mathit{red}, \mathit{green}, \mathit{blue}\}$ is shown. If we remove r from G, we get a new graph G' with three components, each of which has an elimination tree of depth 3, showing $G' \in \textsc{spire}_3$. In the construction of Claim 7.10, the color of r (blue in the example) is added as a second color ring to all neighbors of r, so the color of v_4 in G' is the formal pair $(\mathit{red}, \mathit{blue})$ and of v_7 is $(\mathit{blue}, \mathit{blue})$, while the color of v_5 in G' is the pair $(\mathit{green}, \perp)$.

Claim 7.10. *Suppose every sequence (H_i) of colored graphs $H_i \in \textsc{spire}_t$ is good. Then so is any sequence (G_i) of* connected *colored graphs $G_i \in \textsc{spire}_{t+1}$.*

Proof. Let C be the set of colors and let (G_i) be a sequence of connected colored graphs $G_i \in \textsc{spire}_{t+1}$. By definition, each G_i has an elimination tree (see Figure 10) of depth $t + 1$, rooted at some root vertex r. Build a new colored graph G'_i with the new set of colors $C \times (C \cup \{\perp\})$ as follows: Remove r from $G_i = (V_i, E_i)$ and each vertex $v \in V_i \setminus \{r\}$ with the old color $c(v)$ gets the new color $(c(v), c(r))$ if $\{v, r\} \in E_i$ and otherwise gets the color $(c(v), \perp)$, see Figure 12 for an example. In other words, we mark the vertices of G'_i by whether they were connected to r and, if so, with r's color. The two crucial observations are that (1) if $G'_i \sqsubseteq G'_j$, then $G_i \sqsubseteq G_j$, see Figure 13 for an example; and (2) that each G'_i is an element of $\textsc{spire}_t$. Then by assumption, (G'_i) is good and hence also (G_i). □

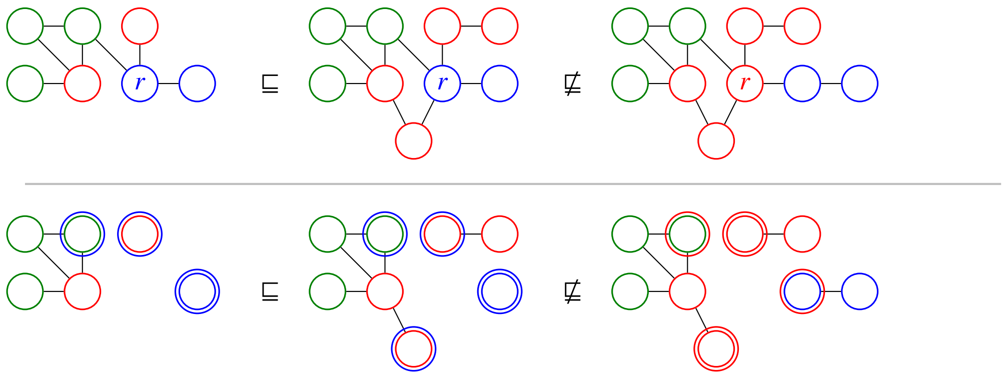


Figure 13: For the upper three graphs, the first is an induced subgraph of the second, while they are not induced subgraphs of the third as the color of r is wrong. The lower three graphs, which result from the construction from Claim 7.10 and which are in a smaller spire, reflect these induced subgraph relations.

Putting the claims together, we get:

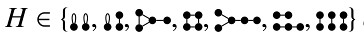
Claim 7.11. *For each t, every sequence of colored graphs in $\textsc{spire}_t$ is good.*

Proof. By induction on t. For $t = 1$, Claim 7.8 gives the statement for sequences of connected graphs in $\textsc{spire}_1$ and Claim 7.9 then also for unconnected graphs in $\textsc{spire}_1$. For the inductive step from t to $t + 1$, Claim 7.10 gives the statement for connected graphs in $\textsc{spire}_{t+1}$ and Claim 7.9 once more generalizes it to all graphs in $\textsc{spire}_{t+1}$. □

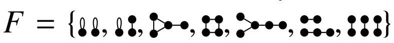
This concludes the proof of the analogue of the Robertson–Seymour Theorem for forbidden induced subgraphs and graphs of bounded tree-depth. □

Applying Forbidden Subgraphs. Although Theorem 7.4 and the whole discussion up to now was about *induced* subgraphs, for our algorithms it is more convenient to use subgraphs. (However, we could easily reformulate the algorithms to use induced subgraphs, by adding to the set in line 6 all graphs that can be obtained by adding edges.)

Algorithm 6: The algorithm is similar Algorithm 5, but needs a new check (line 3) and the main check "if H is a minor of G" is replaced by "if H is a subgraph of G." As shown in Theorem 7.12, the set in line 6 is the correct finite set of forbidden subgraphs to check to decide MAJ-VC.

```
1   input $G = (V, E)$
2
3   if $G \notin$ SPIRE$_5$ then
4       output "$G \notin$ MAJ-VC" and halt
5
6   foreach $H \in \{$$\}$ do
7       if $H \subseteq G$ then // use Lemma 7.1 to check this in time $O(n^6)$
8           output "$G \notin$ MAJ-VC" and halt
9
10  output "$G \in$ MAJ-VC"
```

Theorem 7.12. *Algorithm 6 decides* MAJ-VC *in polynomial time.*

Proof. A quick comparison reveals that there are only two deviations in Algorithm 6 from Algorithm 5: First, in line 3 we check whether $G \notin$ SPIRE$_5$ holds (which is easy to do in time $O(n^5)$ by recursively checking all possible ways to pick elimination vertices) and, if so, claim "$G \notin$ MAJ-VC". This output is correct since by the remark following Lemma 5.4, the tree-depth of all graphs in MAJ-VC is at most 5, so MAJ-VC $\subseteq$ SPIRE$_5$. The second deviation is that the minor relation got replaced by the subgraph relation. In the proof of Theorem 6.6 we use the Robertson–Seymour Theorem to obtain a set of forbidden minors; now we use Theorem 7.4 to obtain *some* finite set F of forbidden subgraphs that we can iterate over in line 6 to correctly decide MAJ-VC. It remains to argue that the particular set $F = \{$$\}$ is once more the right one. To see this, recall from the proof of Theorem 6.6, where we did that argument already for the minor relation, that we explicitly pointed out that the argument showed that $\Pr_{\text{vc}}[G] < \frac{1}{2}$ always implies not only $H \preceq G$ for some $H \in F$, but even $H \subseteq G$ for some $H \in F$; *except* for item 3 in the proof, where it was argued that if a graph G contains a cycle of length 4 *or longer,* then $\preceq G$. However, when G contains a cycle of length 5 or longer, then $\subseteq G$ and $\in F$. □

Extensions.

- As was already pointed out in the introduction of this section, the whole point of switching from the minor relation to the subgraph relation in the context of counting–thresholds, was the fact that if (after renaming) the clauses of a CNF formula ϕ are a subset of the clauses of another formula ψ, then $\Pr[\phi] \geq \Pr[\psi]$. This allows one to generalize the ideas from the present section to $2\text{SAT-PR}_{\geq p}$ and $2\text{SAT-PR}_{>p}$ with (relative) ease.

- However, generalizing the method to $\text{MAJ-3HS} = \text{MON-3SAT-PR}_{\geq 1/2}$, let alone $k\text{SAT-PR}_{\geq p}$, fails almost immediately as it is easy to construct 3-hypergraphs with "many covers and very long paths" as the following set of size-3 hyperedges shows: $\{\{a, x_i, x_j\} \mid 1 \leq i < j \leq n\}$ is hit ("covered") by any set containing a, but we can "walk n steps around the x_i." This is, of course, not a formal proof that the method cannot work, but it is the kind of problems one quickly runs into.

8 The Seventh Way: Via Well-Quasi-Orderings

You leave the Castle of Thousand Spires and embark on a last journey. The seventh way leads you deeper and deeper into a jungle. After some time, you start to hear two male voices that seem to be having a bit of an argument: "No, this is definitely *not* an old Khmer temple!" "Just look at the collection of edifices, Junior, does that not ring a bell?" "Do not call me Junior!" "Well then, *Dr. Jones,* what is your professional archaeological opinion?" "I am not sure, but it is definitely not Khmer." "That is the most unscientific answer I have heard in a long time, Junior. It's that upstart faculty of yours, I have no idea how any of your colleagues ever got tenure." "Do not call me Junior!" The bickering just continues as you draw nearer, till you are suddenly besides the two men and can see what they are discussing: It is a giant sprawl of temple-like spires and towers of all kinds and heights, all of which are richly decorated with symbols on the outside. An enormous number of ropes and vines connect many of the spires with one another, leading from symbols on one to other symbols on other spires. The older of the two gentlemen, who has a quite distinguished, if old-fashioned, demeanor, is about to address you, but the younger man, who is dressed in practical brown clothes and wears a fedora, beats him to it: "Great. *Another* archaeologist! Ah, well, perhaps *you* can shed some light on what all of that means. We have *clearly* established that this is not an old Khmer temple. But *Dr. Jones* here, and admittedly also myself, cannot make any sense of these symbols. Nor why there are so many ropes."

You take a closer look at the symbols and then inform the gentlemen that you do not know about the ropes, but you have seen them quite recently in the Castle of

Thousand Spires on the insides of the walls there. The two men exclaim in unison "Of course! The Castle of Thousand Spires!" and the younger Dr. Jones informs you that "I must find out what these ropes are for!" and then rushes off to scale the temple spires and from time to time climbs one of the ropes to get to another spire. The older Dr. Jones just shakes his head and yells to his son: "That all looks very exciting and adventurous, Junior, but do you really *need* to shimmy along these ropes?" As we will see in the following, the ropes are, indeed, useful as they turn the isolated spires into one great *well-quasi-ordering of all graphs* that is the basis for proving not only that MAJ-VC is tractable, but also generalizations.

Background on Well-Quasi-Orderings. Algorithms 5 and 6 were almost identical: On input of a graph G, we checked for a finite set F of graphs whether for some $H \in F$ we had $H \leq G$ (Algorithm 5) or $H \subseteq G$ (Algorithm 6) and, if so, knew $G \notin$ MAJ-VC. The reason this worked were the Robertson–Seymour Theorem (for the minor relation $\leq$) and Theorem 7.4 (for the induced subgraph relation $\sqsubseteq$ and hence also for the subgraph relation $\subseteq$). Hardly surprisingly, these sets F do not just "happen" to exist for the relations $\leq$, $\subseteq$, and $\sqsubseteq$. Rather, there is a deeper reason: These relations are *well-quasi-orderings* of graphs.

Well-quasi-orderings are a fundamental concept from order theory. In this theory, instead of relations between just graphs, we study pairs $(S, \leq)$ consisting of an arbitrary sets S together with a binary relation $\leq$ on S, meaning $\leq \subseteq S \times S$. Most readers will be familiar with, for instance, *total orderings* $(S, \leq)$, where $\leq$ is transitive ($a \leq b$ and $b \leq c$ implies $a \leq c$ for all $a, b, c \in S$), reflexive ($a \leq a$ for all $a \in S$), and total (exactly one of $a \leq b$ or $b \leq a$ or $a = b$ holds for all $a, b \in S$). A more general kind are *quasi-orderings* $(S, \leq)$, where we just require $\leq$ to be transitive and reflexive – so for two different elements $a, b \in S$ both $a \leq b$ and $b \leq a$ can be true (this peculiarity is emphasized by "quasi"). An example from the world of graphs is (GRAPHS, $\leq$) and note that for any two isomorphic graphs G and G' we have $G \leq G'$ and $G' \leq G$. Even (GRAPHS, $\subseteq$) is "just" a quasi-ordering as we defined $G \subseteq G'$ to mean that some isomorphic copy of G is a subgraph of G'.

Definition 8.1. A *well-quasi-ordering* $(S, \leq)$ is a quasi-ordering such that for all sequences $(s_0, s_1, s_2, \dots)$ with $s_i \in S$ for all i, there are indices $i, j \in \mathbb{N}$ with $i < j$ and $s_i \leq s_j$.

The definition is, admittedly, peculiar, but astute readers will notice that a very similar notion was introduced in the proof of Theorem 7.4 in the form of "good sequences." That is, of course, no coincidence: That proof generalizes from the special case of "the induced subgraph relation on graphs with constant tree-depth" to all well-quasi-orderings. Indeed, *very* astute readers may have noticed that the proof was phrased subtly in such a way that the proofs of the following generalizations can be copied almost verbatim from there (and, thus, will not be given in the following).

- Definition 7.5 of good sequences generalizes easily: *Let $(S, \leq)$ be a quasi-ordering. A sequence (s_i) with $s_i \in S$ for all i is* good *if $s_i \leq s_j$ holds for some indices $i < j$.* With this terminology, the "well" in "well-quasi-ordering" just means that every sequence is good.

- The crucial Claim 7.6, by which a set F of forbidden graphs exists when all sequences are good, generalizes as follows:

  Lemma 8.2. *Let $(S, \leq)$ be a well-quasi-ordering and let $C \subseteq S$ be downward closed under $\leq$, that is, $r \leq s \in S$ implies $r \in S$. Then there is a finite set $F \subseteq S$ such that for all $s \in S$, we have $s \in C$ iff $r \nleq s$ for all $r \in F$.*

  This lemma tells us that we will *always* find sets of forbidden graphs and will *always* be able to run an analogue to Algorithms 5 and 6, when (GRAPHS, $\leq$) is a well-quasi-ordering.

- Claim 7.7 generalizes to the statement: *Every infinite sequence (s_i) in a well-quasi-ordering $(S, \leq)$ has an infinite subsequence $s_{i_1} \leq s_{i_2} \leq s_{i_3} \leq \cdots$.* In other words, sequences in well-quasi-orderings are not only "good" in the sense "at some point, $s_{i_1} \leq s_{i_2}$ will hold," but "perfect" in the sense that "there will be $s_{i_1} \leq s_{i_2} \leq s_{i_3} \leq \cdots$."

- Claim 7.9, by which the general goodness of sequences of connected colored graphs extends to unconnected colored graphs, is just a special case of Higman's Lemma (since finite unconnected graphs can be seen as finite sequences of connected graphs):

  Fact 8.3 (Higman's Lemma, [14])**.** *Let $(S, \leq)$ be a well-quasi-ordering. Let S^* be the set of finite sequences of elements of S; and for $a_1 \dots a_n \in S^*$ and $b_1 \dots b_m \in S^*$ we define $a_1 \dots a_n \leq^* b_1 \dots b_m$ if there are indices $i_1 < i_2 < \cdots < i_n$ with $a_1 \leq b_{i_1}$, $a_2 \leq b_{i_2}$, $\dots$, $a_n \leq b_{i_n}$. Then $(S^*, \leq^*)$ is a well-quasi-ordering.*

- Finally, Claim 7.11, by which every sequence of graphs in a spire is good, translates to the following (and this is how Theorem 7.4 is actually stated in the original papers):

  Fact 8.4 ([24])**.** (SPIRE$_t$, $\sqsubseteq$) *is a well-quasi-ordering for all t.*

With all these preparations, it should come as no surprise that the Robertson–Seymour Theorem can also be rephrased very briefly:

Fact 8.5 ([27]). ($\textsc{graphs}, \preceq$) *is a well-quasi-ordering.*

Note that, in contrast, ($\textsc{graph}, \sqsubseteq$) is *not* a well-quasi-ordering as the bad sequence implicit in equation (4) shows. It is really just each individual ($\textsc{spire}_t, \sqsubseteq$) that is well-quasi-ordered, not the infinite union.

Background on Algorithmic Aspects of Well-Quasi-Orderings. As stated earlier, the theory of well-quasi-orderings is purely order-theoretic and not concerned with questions of tractability. However, the existence of finite sets of forbidden elements for closed sets gives us analogues of Algorithms 5 and 6. The following theorem turns this into a formal statement, where P is the class of problems solvable in polynomial time and XP is the class of relations $\leq$ such that for each fixed $r \in S$ there is a polynomial-time algorithm for deciding on input $s \in S$ whether $r \leq s$ holds (in FPT parlance, the left-hand side of the relation is the parameter):

Algorithm 7: Abstract version of Algorithms 5 and 6 for well-quasi-orderings $(S, \leq)$ with $S \in \mathrm{P}$ and $\leq \in \mathrm{XP}$ and a set $C \subseteq S$ that is downward closed with respect to $\leq$. The set F is a set of forbidden elements, see Lemma 8.2, with respect to $(S, \leq)$ and C.

```
1  input $s$
2
3  if $s \notin S$ then
4      output "$s \notin C$" and halt
5
6  foreach $r \in F$ do
7      if $r \leq s$ then
8          output "$s \notin C$" and halt
9
10 output "$s \in C$"
```

Theorem 8.6. *Let $(S, \leq)$ be a well-quasi-ordering with $S \in \mathrm{P}$ and $\leq \in \mathrm{XP}$. Let $C \subseteq S$ be downward closed with respect to $\leq$. Then $C \in \mathrm{P}$.*

Proof. By Lemma 8.2, there is a finite set $F \subseteq S$ such that for each $s \in S$ we can decide whether we also have $s \in C$ by checking whether for all $r \in F$ we have $r \not\leq s$. Consider Algorithm 7 for this particular F. On input s, we first check whether $s \in S$ holds in line 3 (which can be done in polynomial time because of the assumption $S \in \mathrm{P}$) and, if not, know that $s \notin C \subseteq S$. Otherwise we check whether any $r \in F$ has the property $r \leq s$ (which can also be done in polynomial time because of the assumption $\leq \in \mathrm{XP}$) and, if so, again know $s \notin C$. If s passes all these tests, we correctly assert that $s \in C$ must hold. □

A classical way of instantiating Theorem 8.6 is for the well-quasi-ordering $(\textsc{graphs}, \preceq)$ and $C = \textsc{planar}$: $S = \textsc{graphs} \in \mathrm{P}$ holds trivially, $\preceq \in \mathrm{FPT} \subseteq \mathrm{XP}$ follows from Fact 6.3, and the class of planar graphs is clearly closed under taking minors. Thus, $\textsc{planar} \in \mathrm{P}$. For our purposes, we of course use $(\textsc{graphs}, \preceq)$ and $C = \textsc{maj-vc}$, which is closed under taking minors by Corollary 6.5, to get $\textsc{maj-vc} \in \mathrm{P}$ in the form of Theorem 6.6. Alternatively, we can instantiate Theorem 8.6 for $(\textsc{spire}_5, \subseteq)$ and $C = \textsc{maj-vc}$ to get Theorem 7.12.

Applying Well-Quasi-Orderings. While it is certainly elegant to restate our previous results in the abstract framework of well-quasi-orderings, does this actually give us any new insights? In other words, a weary traveller might sigh: *Why did I bother to walk this way?*

To answer this, let us rephrase the discussions of possible extensions at the end of the fifth and sixth way in terms of well-quasi-orderings: $(\textsc{graphs}, \preceq)$ is a well-quasi-ordering for which $\textsc{maj-vc}$ is downward closed, but when we try to go from graphs to $2\textsc{cnf}$, the set $2\textsc{sat-pr}_{\geq 1/2}$ does not seem closed under any sensible version of the minor relation. In contrast, the (induced or not) subgraph relation is more robust and allows generalizations, but neither $(\textsc{graphs}, \sqsubseteq)$ nor $(\textsc{graphs}, \subseteq)$ are well-quasi-orderings. Rather, only $(\textsc{spire}_t, \subseteq)$ was. What we really need is a *single "robust" well-quasi-ordering on all graphs that is anti-monotone with respect to* $\Pr_{\mathrm{vc}}[\cdot]$. Fortunately, this can be achieved by adding ropes (an idea borrowed from Rado [25]):

Definition 8.7. For each constant c, define a relation $\sqsubseteq^c$ as follows: Let $H \sqsubseteq^c G$ hold if

1. $H \sqsubseteq G$, that is, H is an induced subgraph of G, *or*
2. G contains a matching of size $c \cdot n_H$ where n_H is the number of vertices in H.

The idea behind this definition is that in addition to the induced subgraph relation, we add new ways to get from a graph H to a graph G (the "ropes") that do not respect the induced subgraph relation at all. Nevertheless, the resulting relation has all the desirable properties we need:

Lemma 8.8. *For each c,* $(\textsc{graphs}, \sqsubseteq^c)$ *is a well-quasi-ordering.*

Proof. Consider any sequence (G_i) of graphs. If there is a t such that $G_i \in \textsc{spire}_t$ holds for all i, we know that $G_i \sqsubseteq G_j$ holds for some $i < j$ by Fact 8.4 and hence also $G_i \sqsubseteq^c G_j$. Otherwise, the depth of the elimination trees of the G_i grows beyond all bounds; so some G_j with $j > 0$ has an elimination tree of depth $2c \cdot |V_0|$ where $G_0 = (V_0, E_0)$. Then this G_j contains a path of length $2c \cdot |V_0|$ (recall the remark

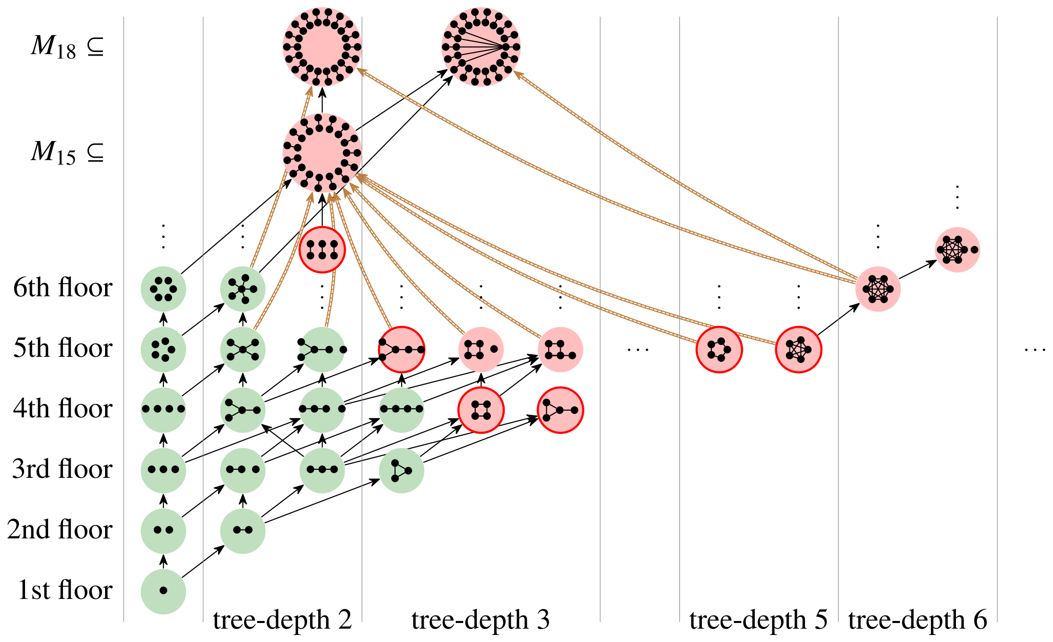


Figure 14: The well-quasi-ordering (GRAPHS, $\sqsubseteq^3$). As in Figure 13, the graphs are grouped according to tree-depth, but while in the spires of Figure 13 the graphs from a smaller spire got repeated, each graph is now present only once and we only use the tree-depth for visual grouping. Also as in that figure, green graphs lie in MAJ-VC while red graphs do not, and the circled red graphs are minimal elements of the complement of MAJ-VC with respect to $\sqsubseteq^3$. A normal arrow ($\longrightarrow$) still just indicates that $H \sqsubseteq G$ holds (H is an induced subgraph of G). The new "ropes" ($\rightsquigarrow$) indicate that $H \sqsubseteq^3 G$ holds (only) by item 2 in Definition 8.7: The number of vertices in H (5 on the fifth floor and 6 on the sixth floor) is at most a third of the size of a matching in G (15 for graphs containing the 15-edge matching M_{15} and 18 for those containing M_{18}). Note that these ropes do not respect the subgraph relation at all, but they *do* respect $\mathrm{Pr}_{\mathrm{vc}}[\cdot]$ by Lemma 8.10.

after Lemma 5.4) and, hence, also a *matching* of size $c \cdot |V_0|$. Thus, $G_i \sqsubseteq^c G_j$ for $0 = i < j$. □

Lemma 8.9. *For each c, we have* $\sqsubseteq^c \in \mathrm{XP}$.

Proof. Fix a graph H and let n_H be the number of its vertices. To decide on input H whether $H \sqsubseteq^c G$ holds, we have to test whether either $H \sqsubseteq G$ holds, which can be done in polynomial time by Lemma 7.1, or whether G contains a matching of size $c \cdot n_H$, which can also easily be checked in polynomial time (note that n_H is a constant parameter). □

Lemma 8.10. $H \sqsubseteq^3 G$ *implies* $\Pr_{\mathrm{vc}}[H] \geq \Pr_{\mathrm{vc}}[G]$.

Proof. If $H \sqsubseteq G$, then $\Pr_{\mathrm{vc}}[H] \geq \Pr_{\mathrm{vc}}[G]$. So assume that G contains a matching M of size $3n_H$ where n_H is the number of vertices of H. Since H has at least one cover, we have $\Pr_{\mathrm{vc}}[H] \geq 2^{-n_H}$. On the other hand, $\Pr_{\mathrm{vc}}[M]$, the probability of covering a size $c \cdot n_H$ matching, is $(\frac{3}{4})^{3n_H} = (\frac{27}{64})^{n_H} < 2^{-n_H}$. Thus $\Pr_{\mathrm{vc}}[H] \geq 2^{-n_H} > \Pr_{\mathrm{vc}}[M] \geq \Pr_{\mathrm{vc}}[G]$. □

In particular, MAJ-VC is downward closed with respect to $\sqsubseteq^3$. Putting it all together, we get:

Theorem 8.11. *Algorithm 7 decides* MAJ-VC *in polynomial time when instantiated for* (GRAPHS, $\sqsubseteq^3$) *and* $C =$ MAJ-VC.

Extensions.

- Just as not only MAJ-VC is closed with respect to taking minors, but all MON-2SAT-PR$_{\geq p}$ and even all MON-2SAT-PR$_{>p}$ are for any $p \in [0, 1]$, so are they all with respect to $\sqsubseteq^3$. In other words, like the minor relation, we get the tractability of MON-2SAT-PR$_{\geq p}$ and MON-2SAT-PR$_{>p}$ "alongside" that of MAJ-VC "for free."

- Unlike the minor relation, but like the induced subgraph relation, it is not hard (but also not *quite* trivial) to extend the definition to 2CNF formulas with negations. This allows us to give well-quasi-ordering-based proofs of 2SAT-PR$_{\geq p} \in \mathrm{P}$ and 2SAT-PR$_{>p} \in \mathrm{P}$.

- Unlike the induced subgraph relation, one *can* extend the idea to 3CNF formulas and beyond (the details are beyond the already too-large scope of this paper and will be discussed at some future time). That is, for each k it is possible to define a well-quasi-ordering $\leq$ of all kCNF formulas such that $\phi \leq \psi$ implies $\Pr[\phi] \geq \Pr[\psi]$, meaning that sets such as 4SAT-PR$_{>1/2}$ are closed with respect to this ordering. However, as 4SAT-PR$_{>1/2}$ is known to be NP-complete, we cannot hope to have $\leq \in \mathrm{XP}$, but only $\leq \in \mathrm{XNP}$.

9 Conclusion

It has been quite a long journey through the complexity landscape – just to visit the problem MAJ-VC, which is neither particularly difficult nor particularly relevant to solve in practice. Hopefully however, the journey was the reward, with some new (in)sights gained along the way. At many vantage points we glimpsed in the distance how the presented ideas applied to more general problems like 2SAT-PR$_{\geq p}$, MAJ-3HS, or kSAT-PR$_{\geq p}$. Experienced travellers of the complexity landscape may even have spotted applications to *constraint satisfaction problems* and *homomorphism counting* near the horizon, which are even more general than satisfiability problems, but to which many of the presented ideas still apply.

Readers who are still not yet tired may wonder how many ways of showing the tractability of MAJ-VC there are in total. On the one hand, one could argue that the last three ways presented in this paper are actually just *one* way as they are all variations of using well-quasi-orderings to solve MAJ-VC; just at different levels of abstraction. So, perhaps this paper only really presented five ways. On the other hand, there is at least one more way, presented in detail in [30], where *kernels* are used to solve kSAT-PR$_{\geq p}$. For the special case of MAJ-VC, this amounts to "as long as there is a star of size six, remove one ray." However, the correctness of this idea hinges on the Spectral Well-Ordering Theorem [30] just as that of the random sampling method from the third way, so perhaps this is not another way after all.

Conversely, it is also interesting to see which ways do *not* seem to lead to solving MAJ-VC efficiently. For instance, it is not clear how *linear programming* could be used to decide MAJ-VC naturally – despite the fact that linear programming is one of the most powerful tools we have to prove the tractability of problems.

In the end, many, but not all paths lead to the tractability of MAJ-VC and readers are very much invited to discover new ways and to share them with their fellow travellers in the theory landscape.

References

[1] Stefan Arnborg, Jens Lagergren, and Detlef Seese. Easy problems for tree-decomposable graphs. *Journal of Algorithms*, 12(2):308–340, 1991.

[2] Shyan Akmal and Ryan Williams. MAJORITY-3SAT (and related problems) in polynomial time. In *Proceedings of the 62nd Annual IEEE Symposium on Foundations of Computer Science (FOCS 2021)*, pages 1033–1043. IEEE Press, 2022.

[3] Hans L. Bodlaender and Torben Hagerup. Parallel algorithms with optimal speedup for bounded treewidth. *SIAM Journal on Computing*, 27(6):1725–1746, 1998.

[4] Hans L. Bodlaender. NC-algorithms for graphs with small treewidth. In *Proceedings of the 14th International Workshop on Graph-Theoretic Concepts in Computer Science (WG 1988)*, volume 344 of *Lecture Notes in Computer Science*, pages 1–10. Springer, 1989.

[5] Hans L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM Journal on Computing*, 25(6):1305–1317, 1996.

[6] Hans L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theoretical Computer Science*, 209(1):1–45, 1998.

[7] Stephen A. Cook. The complexity of theorem-proving procedures. In *Conference Record of the Third Annual ACM Symposium on Theory of Computing (STOC 1971)*, pages 151–158, Shaker Heights, Ohio, 1971.

[8] Brouno Courcelle. Graph rewriting: An algebraic and logic approach. In Jan van Leeuwen, editor, *Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics*, pages 193–242. Elsevier and MIT Press, 1990.

[9] Rod G. Downey and Michael R. Fellows. Fixed-parameter tractability and completeness II: On completeness for W[1]. *Theoretical Computer Science*, 141(1):109–131, 1995.

[10] Rod G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Springer, 2013.

[11] Jan Dreier, Sebastian Ordyniak, and Stefan Szeider. SAT backdoors: Depth beats size. *Journal of Computer and System Sciences*, 142(103520):1–22, 2024.

[12] Michael Elberfeld, Andreas Jakoby, and Till Tantau. Logspace versions of the theorems of Bodlaender and Courcelle. In *Proceedings of the IEEE 51st Annual Symposium on Foundations of Computer Science (FOCS 2010)*, pages 143–152, 2010.

[13] Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Springer, 2006.

[14] Graham Higman. Ordering by divisibility in abstract algebras. *Proceedings of the London Mathematical Society*, s3-2(1):326–336, 1952.

[15] David G. Harris and N. Sundararajan Narayanaswamy. A faster algorithm for vertex cover parameterized by solution size. In Olaf Beyersdorff, Mamadou M. Kanté, Orna Kupferman, and Daniel Lokshtanov, editors, *41st International Symposium on Theoretical Aspects of Computer Science (STACS 2024)*, volume 289 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 40:1–40:18, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[16] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

[17] Ken-ichi Kawarabayashi, Yusuke Kobayashi, and Bruce Reed. The disjoint paths problem in quadratic time. *Journal of Combinatorial Theory, Series B*, 102(2):424–435, 2012.

[18] Joseph B. Kruskal. The theory of well-quasi-ordering: A frequently discovered concept. *Journal of Combinatorial Theory, Series A*, 13(3):297–305, 1972.

[19] Kazimierz Kuratowski. Sur le problème des courbes gauches en topologie. *Fundamenta Mathematicae*, 15:271–283, 1930. In French.

[20] Leonid Levin. Универсальные задачи перебора (Universal search problems). *Проблемы передачи информации (Problems of Information Transmission)*, 9(3):115–116, 1973. See [32, pages 399–400] for a translation to English.

[21] Daniel Lokshtanov, Fahad Panolan, and M. Sridharan Ramanujan. Backdoor sets on nowhere dense SAT. In Mikołaj Bojańczyk, Emanuela Merelli, and David P. Woodruff, editors, *Proceedings of the 49th International Colloquium on Automata, Languages, and Programming (ICALP 2022)*, volume 229 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 91:1–91:20, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[22] Michel Minoux. LTUR: a simplified linear-time unit resolution algorithm for Horn formulae and computer implementation. *Information Processing Letters*, 29(1):1–12, 1988.

[23] Robin A. Moser and Dominik Scheder. A full derandomization of Schöning's k-SAT algorithm. In *Proceedings of the Forty-Third Annual ACM Symposium on Theory of Computing (STOC 2011)*, pages 245–252, New York, NY, USA, 2011. Association for Computing Machinery.

[24] Jaroslav Nešetřil and Patrice Ossona de Mendez. *Sparsity: Graphs, Structures, and Algorithms*, volume 28 of *Algorithms and Combinatorics*, chapter 6: Bounded height trees and tree-depth, pages 115–144. Springer, 2012.

[25] Richard Rado. Partial well-ordering of sets of vectors. *Mathematika*, 1(2):89–95, 1954.

[26] Neil Robertson and Paul D. Seymour. Graph minors. II. Algorithmic aspects of tree-width. *Journal of Algorithms*, 7(3):309–322, September 1986.

[27] Neil Robertson and Paul D. Seymour. Graph minors. XX. Wagner's conjecture. *Journal of Combinatorial Theory, Series B*, 92(2):325–357, November 2004.

[28] Uwe Schöning. A probabilistic algorithm for k-SAT and constraint satisfaction problems. In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science (FOCS 1999)*, pages 410–414, USA, 1999. IEEE Computer Society.

[29] Till Tantau. A gentle introduction to applications of algorithmic metatheorems for space and circuit classes. *Algorithms*, 9(3), 2016.

[30] Till Tantau. On the satisfaction probability of k-CNF formulas. In Shachar Lovett, editor, *Proceedings of the 37th Computational Complexity Conference (CCC 2022)*, volume 234 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 2:1–2:27, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[31] Seinosuke Toda. PP is as hard as the polynomial-time hierarchy. *SIAM Journal on Computing*, 20(5):865–877, 1991.

[32] Boris A. Trakhtenbrot. A survey of Russian approaches to *perebor* (brute-force searches) algorithms. *Annals of the History of Computing*, 6(4):384–400, 1984.

[33] Leslie G. Valiant. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8(3):410–421, 1979.

[34] Ryan Williams, Carla P. Gomes, and Bart Selman. Backdoors to typical case complexity. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI 2003)*, pages 1173–1178, San Francisco, CA, USA, 2003. Morgan Kaufmann Publishers Inc.

[35] David Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. In *Proceedings of the Thirty-Eighth Annual ACM Symposium on Theory of Computing (STOC 2006)*, pages 681–690, New York, NY, USA, 2006. Association for Computing Machinery.