Electronic Colloquium on Computational Complexity, Report No. 73 (2004)

ECCC

A Top-Down Approach to Search-Trees: Improved Algorithmics for 3-Hitting Set

Henning Fernau
Universität Tübingen, WSI für Informatik, Sand 13,
72076 Tübingen, Germany
fernau@informatik.uni-tuebingen.de

University of Newcastle, School of Electr. Eng. & Computer Sci.,
University Drive, Callaghan, NSW 2308, Australia

July 9, 2004

Abstract

In this paper, we show how to systematically improve on parameterized algorithms and their analysis, focusing on search-tree based algorithms for d-Hitting Set, especially for $d = 3$. We concentrate on algorithms which are easy to implement, in contrast with the highly sophisticated algorithms which have been elsewhere designed to improve on the exponential base in the algorithms.

The algorithm analysis is based on a novel way to exploit a so-called auxiliary parameter, a technique which we believe can be used in other circumstances, as well.

1 Introduction

1.1 Our approach—in general

We exhibit how to systematically design and analyze search-tree algorithms within the framework of parameterized algorithmics. Here, we advocate a top-down design as opposed to a rather bottom-up approach, because

- in terms of algorithm design, the top-down approach appears to be more natural and the correctness of the resulting algorithms can be comparatively easily shown in a generic fashion,
- we can defer often nasty and intricate analysis of cases as far as possible, only driven and guided by demand, and

ISSN 1433-8092

- the resulting algorithms tend to be simpler than via the opposite approach, pretty much resembling heuristic pruning techniques as used in branch-and-cut algorithms for solving hard problems.

We will exemplify this approach by developing and analyzing simple algorithms for Hitting Set problems. Note that previous algorithms, notably for 3-Hitting Set, were much more elaborate; yet, their analysis yielded worse upper bounds for the running time of those algorithms. On the downside, our approach requires a more sophisticated analysis of the running time of the (simple) search-tree algorithm. This might of course also indicate that, by a tricky algorithm analysis, the earlier published more intricate algorithms can be proven to have better worst-case running times than previously attributed to them.

Hitting Set problems show up in many places. For example, Reiter's *theory of diagnosis* is based upon so-called Hitting Set Trees, see [10].

This problem has attracted interest not from the point of view of approximation (a recent reference being [4]) but also from the parameterized viewpoint [9]. We will follow the latter approach here, too.

1.2 General notions and definitions

Since we are focusing on Hitting Set, we now introduce some terminology on hypergraphs. A *hypergraph* $G = (V, E)$ is given by its finite set of *vertices* V and its set of *(hyper)-edges* E, where a hyperedge is a subset of V. The cardinality of a hyperedge e is also called its *degree*, written $\delta(e)$. Likewise, the cardinality of the set of edges which contain a specific vertex v is called the *degree* of v, written $\delta(v)$. A hypergraph is called *d-regular* if all vertices have degree d.

While $|S|$ is used to denote the cardinality of an arbitrary set S, we use $\#^d S$ to count the number of edges (or vertices) of the edge (or vertex) set S which have degree d. Moreover, for a vertex x, $\delta^d(x)$ is the number of hyperedges of degree d that contain x.

d-Hitting Set can be viewed as a "vertex cover problem" on hypergraphs More formally, this problem can be stated as follows:
Problem name: d-Hitting Set
Given: A hypergraph $G = (V, E)$ with edge degree bounded by d
Parameter: k
Output: Is there a *hitting set* (or *cover*) of size at most k: $\exists C \subseteq V \forall e \in E(C \cap e \neq \emptyset)$?

We are going to develop search-tree algorithms and apply a parameterized analysis of the search-tree size. If we then say that the algorithm

has $\mathcal{O}^*(f(k))$ running time, where k is the parameter, this means that the search-tree has size $T(k) = \mathcal{O}(f(k))$, since the work in each search-tree node will be at worst polynomial in n. In actual fact, all analysis that follows will be a clever estimate on the size of the search-tree.

1.3 Previous work

We are only aware of one paper dealing with d-Hitting Set for $d > 2$ from a parameterized perspective, namely Niedermeier and Rossmanith's [9]. Niedermeier and Rossmanith also present a simple linear-time kernelization, i.e., a preprocessing step typical for parameterized algorithms which leaves us with an instance of size $g(k)$ (instead of n). By a quite generic approach, they can bound the size of the search-tree for $d = 3$ by $T(k) \leq 2.42^k$. The base c of the exponential term in the formula estimating $T(k) \leq c^k$ is also known as *branching number*, since it is derived from the analysis of the branching behavior of search trees. By an intricate case analysis of a comparatively complicated algorithm, they were able to arrive at an $\mathcal{O}(2.270^k + n)$ algorithm for 3-Hitting Set.

For the special case of 2-Hitting Set, likewise known as Vertex Cover, in a kind of race (using more and more involved case analysis) an $\mathcal{O}(1.285^k + n)$-algorithm [2] has been obtained. As the so-called auxiliary parameter (discussed below) we chose for our approach to Hitting Set problems would become trivial, we cannot tackle Vertex Cover in this paper.

1.4 The results of this paper

We are improving on Niedermeier and Rossmanith's results for 3-Hitting Set Basically, we are doing a better analysis of a simple search-tree algorithm guided by so-called *heuristic priorities*. In contrast to earlier approaches, we don't produce complicated algorithms this way. The better analysis is based on the introduction of a second *auxiliary parameter,* a technique which can be useful in other analyses of parameterized algorithms, as we believe. This way, we obtain an $\mathcal{O}(2.179^k + n)$ algorithm for 3-Hitting Set.

Most advanced search-tree algorithms for parameterized problems were developed in a bottom-up fashion. By this, we mean that in order to lower the basis c in $\mathcal{O}^*(c^k)$ algorithms, situations were identified that produced better branchings by a local analysis. To find these "good situations," new case distinctions had to be incorporated in the program code for the search-tree itself. This not only makes implementations of the proposed algorithms a rather tedious task, but we were informed by practitioners (personal com-

munication by P. Shaw) that sometimes when they leave out these "sub-case optimizations", the code runs faster. Moreover, it would be hard to rigorously prove the correctness of such an elaborate search-tree algorithm.

The top-down approach we advocate here produces comparatively simple algorithms. The best published example of such a *simple* search-tree algorithm (where a formal correctness proof would be close to trivial) is PLANAR DOMINATING SET, see [1]. These algorithms are created by applying easily understandable *heuristic rules* that decide how to branch in a given situation. For DOMINATING SET, it is obviously good to always branch on vertices of smallest degree. The burden now lies mainly on the analysis of the running time. This analysis is undertaken in this paper by using a so-called *auxiliary parameter*. In the case of d-HITTING SET, this is the number of *small edges* (i.e., edges of degree at most $(d-1)$). The intuition is: the more small edges we have, the better the branching.

2 Heuristics and reductions for d-HITTING SET

2.1 A generic algorithm and its correctness

```
simple-HS-heuristic-binary(G, k, S):
  Exhaustively apply the reduction rules.
  // call the resulting instance (G, k) and the intermediate solution S
  IF (G, k) is a simple instance
  THEN solve in polynomial time and return solution S'
  ELSE
    choose some edge e and some x ∈ e; // acc. to heuristic priorities
    S' = ∅; // solution to be constructed
    G' = (V \ {x}, {e ∈ E | x ∉ e});
    S' = simple-HS-heuristic-binary(G', k − 1, S ∪ {x});
    IF S' == ∅ THEN // try x not in solution
      G − x = (V \ {x}, {e \ x | e ∈ E});
      S' = simple-HS-heuristic-binary(G − x, k, S)
    return S'
```

In contrast to earlier approaches, we use *binary* branching in our algorithm for d-HITTING SET: take x into the cover or not. This generic algorithm contains a couple of black boxes which will be explained in the following:

- *Reduction rules* are a very powerful technique to simplify given instances. In general, they will modify the given graph and also the parameter. Such rules have been previously employed to prove small so-called *problem kernels* for parameterized problems (an issue completely neglected in this paper) and are here for the first time *systemat-*

ically used for a different purpose: to obtain substantial improvements in the running time analysis.

- *Simple instances* belong to classes of instances for which polynomial time algorithms are known. We will also refer to these classes as *stopping situations*.
- *Heuristic priorities* specify, in our case, how to "choose" e and $x \in e$ for the subsequent branching, e.g., preferring to branch on small edges.

The main advantages of this "generic approach" are the following ones:

- The whole algorithm is very transparent and modular, which pretty much simplifies the task of producing a correct implementation of the algorithm. Moreover, different parts of the algorithm can be tested rather independently, and a quick implementation of a running prototype is possible.
- As can be seen in the proof of the correctness of the algorithm, the correctness is only affected by the correctness of the reduction rules and the correctness of the rules for simple instances.
- When we try to improve the algorithm (in order to prove better run time bounds), we will henceforth only change the heuristic priorities or add new sound reductions. This can never affect the correctness of the algorithm.

The reader is encouraged to compare these points with earlier approaches to improve on running times for parameterized algorithms. There, in principle, it is crucial to always prove the correctness of the algorithm after seemingly small changes of the algorithm performed in order to tweak the efficiency.

Lemma 1 *If the reduction rules and the procedure solving simple instances are correct, then calling* `simple-HS-heuristic-binary`$(G, k, \emptyset)$ *either returns a correct solution to the* d-Hitting Set *instance* (G, k) *or it returns* $\emptyset$*, which is only a correct solution if* G *has no edges; otherwise,* $\emptyset$ *signals that* (G, k) *has no solution.*

2.2 Reduction rules

We now define *reduction rules* which we will always exhaustively apply at the beginning of each recursive call. When arguing about the branching in the search-tree, we can therefore always assume that we deal with *reduced instances*, i.e., instances to which none of the reduction rules are applicable.

1. (hyper)edge domination: A hyperedge e is *dominated* by another hyperedge f if $f \subset e$. In that case, delete e.

2. tiny edges: Delete all hyperedges of degree one and place the corresponding vertices into the hitting set.

3. vertex domination: A vertex x is *dominated* by a vertex y if, whenever x belongs to some hyperedge e, then y also belongs to e. Then, we can simply delete x from the vertex set and from all edges it belongs to.

Lemma 2 *The three reduction rules are sound.*

The rules themselves are not new: the last two are also used by Niedermeier and Rossmanith. Actually, the rules seem to be "around" since many years. The oldest reference (which was found by Regina Barretta, Newcastle) is to our knowledge [6, Chapter 8]. They are also known for related problems as, e.g., the Path Cover Problem, see [11]. Note that in the advanced analysis of the 3-Hitting Set algorithm, we finally propose two other reduction rules (left out in the extended abstract).

Lemma 3 *In a reduced instance with at least two vertices, no vertex has degree less than two.*

The following observation (a consequence of edge domination) is used as a guideline for heuristics and in some parts of the algorithm analysis:

Lemma 4 *If e is an edge in a hypergraph $G = (V, E)$ (as an instance of d-Hitting set) to which the edge domination rule is not applicable, then in the branch where $x \in e$ is* not *taken into the cover, the instance $G - x$ will have*

$$\#^{d-1}E + \delta^d(x) - \delta^{d-1}(x) \tag{1}$$

many edges of degree $(d-1)$.

2.3 Simple instances

It is possible to classify instances with maximum vertex degree of two as "simple:" they can be solved in polynomial time by matching techniques.[1] This then gives the following corollary:

[1] This result was communicated to us by G. Woeginger; the idea is to relate it with an edge covering set problem by interpreting the vertices of degree two as edges of a graph and the hyperedges as vertices.

Corollary 5 *Some vertex in a non-simple reduced instance has degree at least three.*

Since the correctness of the stopping condition is hence known and the soundness of the reduction rules is stated in Lemma 2, Lemma 1 proves:

Theorem 6 *The algorithm* `simple-HS-heuristic-binary` *is correct.*

2.4 Heuristic priorities

We will further the analysis of `simple-HS-heuristic-binary`. To this end, we will refine the conditions telling us which edges and vertices the algorithm will select next for branching. Besides the mentioned reduction rules, these conditions (which we call *heuristic priorities*) are a key ingredient for the analysis of the algorithm. One simple rule that will be always used is to preferably branch on edges of low degree. Heuristic priorities have to be always designed in a way that if in a certain branch no sufficient gain in terms of the (main) parameter is possible, an (analyzable) gain in the auxiliary parameter should show up.

3 A simple approach to 3-Hitting Set

3.1 How to analyze the generic d-Hitting Set algorithm

The idea of making favorable branches first has also another bearing, this time on the way we are going to analyze the running time of our algorithm. Let $T_d^\ell(k)$, $\ell \geq 0$ denote the *size* (more precisely, the number of leaves) of the search-tree when assuming that exactly ℓ edges in the given instance (with parameter k) have a degree of (at most) $d-1$. So, ℓ is the auxiliary parameter in our analysis. Sometimes, we will shortcut our discussions by using $T_d^{\geq \ell}(k)$, denoting the situation of a search-tree assuming that at least ℓ edges in the given instance (with parameter k) have a degree of (at most) $(d-1)$. The intuition is that, e.g., in the case of 3-Hitting Set, $T_3^4(k)$ would describe a situation which is "more like" 2-Hitting Set than $T_3^3(k)$. Therefore, we can conclude: $T_d^{\geq \ell}(k) \leq T_d^\ell(k)$. Regarding an upperbound on the size $T_d(k)$ of the search-tree of the whole problem, we can estimate $T_d(k) \leq T_d^0(k)$, since the worst case is that we have no edges of low degree. In the following, we are exhibiting mutually recursive relationships between different $T_d^\ell(k)$; solving these recursions will yield the bounds on the size of the search-tree and hence on the running time.

Lemma 7 $T_d^0(k) \leq T_d^0(k-1) + T_d^3(k)$.

Proof. Due to Cor. 5, the instance G contains (immediately before the branching) a vertex x of degree 3 (or larger). One branch is that x is put into the cover. If x is not put into the cover, then at least three new edges of degree $(d-1)$ are created. ∎

Lemma 8 $T_d^1(k) \leq T_d^0(k-1) + T_d^1(k-1) + T_d^2(k-1) + \cdots + T_d^{d-2}(k-1)$.

Proof. Due to Lemma 3, all vertices have degree at least two. If the chosen $x \in e$ is not put into the cover, then at least one new edge of degree $(d-1)$ is created besides the edge $e' = e \setminus \{x\}$ of degree $(d-2)$. According to the heuristic priorities, we would continue branching at some vertex from e'. The argument then repeats, yielding the formula as claimed, since every time a new edge of degree $(d-1)$ is created, since otherwise the reduction rules would trigger and reduce the number of branches, so that in that case an even better analysis would be possible. ∎

Observe that, when plugging in $T_d^\ell \leq T_d^1$ for $\ell \geq 1$, we obtain the following recurrences:

$$\begin{aligned} T_d^0(k) &\leq T_d^0(k-1) + (d-1)T_d^1(k-1) \\ T_d^1(k) &\leq T_d^0(k-1) + (d-2)T_d^1(k-1) \end{aligned}$$

These are exactly the recurrences derived by Niedermeier and Rossmanith (actually for a different algorithm), which immediately entails that we can only beat their results with our analysis. We will show this in the following for the special case $d = 3$ and an analysis involving T_3^0, T_3^1 and T_3^2.

3.2 Heuristics for 3-Hitting Set

We us the following *heuristic priorities.* $H1$ is motivated by Lemma 4.

$H0$ *Prefer small edges.* More formally, let $E_0 = \{e \in E \mid \delta(e) = 2\}$. If $E_0 = \emptyset$, set $E_0 = E$.

$H1$ *Maximize Eq. (1)*, i.e., let $V_1 = \{x \in \bigcup_{e \in E_0} \mid \delta^3(x) - \delta^2(x) \text{ is maximum}\}$.

$H2$ Choose some $x \in V_1$ of *maximum degree.*

In the next lemma, we show a first step into a strategy which will finally give us better branching behaviors. Namely, we try to exploit the effect of reduction rules triggered in different sub-cases.

Lemma 9 $T_3^1(k) \leq \max\{2^k, T_3^0(k-1) + T_3^2(k-1)\}$.

Proof. The instance G has an edge $e = \{x, y\}$ of degree two. Assume that $\delta(x) \geq \delta(y)$, so that we branch at x. By Lemma 3, $\delta(x) \geq \delta(y) \geq 2$.

We distinguish now two cases: 1. $\delta(y) = 2$. If we take x into the cover, then y will become of degree one and hence will get deleted by the reduction rules (see Lemma 3), producing one new edge e' of degree at most two with $e' \neq e$ due to edge domination. So this gives a $T_3^1(k-1)$-branch. Not taking x into the cover means to take y into the cover by the tiny edge rule, and at least one other edge of degree two (from the ones which have been incident to x) is created in the instance $G - x$. This gives another $T_3^1(k-1)$-branch. The corresponding recurrence can be solved by 2^k as claimed.

2. $\delta(y) > 2$. The worst case is that $\delta(y) = \delta(x) = 3$. If x goes into the cover, we only get a $T_3^0(k-1)$-branch. If x is not put into the cover, y will be in the cover. Moreover, at least two new edges of degree two are created, yielding a $T_3^2(k-1)$-branch in this subcase. ∎

Lemma 10 *Assuming two edges of degree two, we get the following bound:*

$$T_3^2(k) \leq \max\{T_3^1(k-1)+T_3^2(k-1), T_3^0(k-2)+T_3^1(k-1), T_3^0(k-1)+T_3^2(k-2)\}.$$

Proof. We consider first the situation that the two edges e_1 and e_2 of degree two are disjoint. Then, basically the analysis of the previous lemma applies, showing that

$$T_3^2(k) \leq \max\{2^k, T_3^1(k-1) + T_3^3(k-1)\} \leq T_3^1(k-1) + T_3^2(k-1). \quad (2)$$

Otherwise, let $e_1 = \{x, y\}$ and $e_2 = \{x, z\}$. Then, $\delta(y) \geq 2$ and $\delta(z) \geq 2$ according to Lemma 3, because when branching, we always have reduced instances. We again make some case distinctions:

1. If $\delta(x) = 2$, then (since $\delta^3(x) = 0$ and $\delta^3(y) > 0$, $\delta^3(z) > 0$) we can assume we branch (w.l.o.g.) at y. If y is put into the cover, then x will be removed from the vertex set by the reduction rules (since it then has degree one) and z will be put into the cover, as well, giving a $T_3^0(k-2)$-branch. If y is not put into the cover, x will be by the tiny edge rule, covering both e_1 and e_2. Moreover, since $\delta^3(y) > 0$, at least one new edge e_y of degree two is created. Since the edge domination rule was not applicable to the instance under consideration, $x \notin e_y$, because otherwise $e_1 \subset e_y \cup \{y\}$. This branch is of type $T_3^1(k-1)$. This yields:
$$T_3^2(k) \leq T_3^0(k-2) + T_3^1(k-1). \quad (3)$$

2. If $\delta(x) = 3$ and say $\delta(y) = 2$, assume we branch at x (due to heuristic priority H2). If x goes into the cover, then the vertex domination rule eliminates y, producing a new edge of degree two. This gives a $T_3^1(k-1)$-branch. Otherwise, both y and z must go into the cover, and $\delta^3(x) = 1$ gives us a new edge of degree two (due to edge domination). This yields a $T_3^1(k-2)$-branch. In conclusion, we have:

$$T_3^2(k) \leq T_3^1(k-1) + T_3^1(k-2).$$

This is always better than Eq. (3) and won't be considered any further.

3. If $\delta(x) \geq 3$ and $\delta(y) \geq 3$, assume we branch at y (due to heuristic priority H1). A simple analysis yields (cf. Eq. (2))

$$T_3^2(k) \leq T_3^1(k-1) + T_3^2(k-1).$$

4. If $\delta(x) \geq 4$, then assume that we branch at x. When x is not put into the cover, then we gain two new small edges. Therefore,

$$T_3^2(k) \leq T_3^0(k-1) + T_3^2(k-2). \qquad \blacksquare$$

3.3 Estimating running times

As has been successfully done in other search-tree analyses, we will focus in the the following on analyzing "pure" cases, where we assume that we fix the possible branching scenarios for T_3^1 and for T_3^2, assuming that the worst case will show up in these pure cases. Of course, in practice, "mixed cases" will happen, where in the same search-tree different branching cases occur. This restriction to the worst-case analysis of pure branching scenarios brings along another benefit: looking closely at the analysis given in Lemma 10, one can notice that the worst case is always happening if the inequalities are satisfied with equality. Hence, we have to deal with the following set of recursions.

$$\begin{array}{cll} & T_3^0(k) &= T_3^0(k-1) + T_3^2(k) \\ & T_3^1(k) &= T_3^0(k-1) + T_3^2(k-1) \\ (1) & T_3^2(k) &= T_3^1(k-1) + T_3^2(k-1) \\ (2) & T_3^2(k) &= T_3^0(k-2) + T_3^1(k-1) \\ (3) & T_3^2(k) &= T_3^0(k-1) + T_3^2(k-2) \end{array}$$

We are looking for a solution c_i such that $T_3^0(k) = c_i^k$ for $i = 1, 2, 3$, corresponding to case (i) in the list of equations above. Note that we left out the cases where obviously $T_3^j(k) \leq 2^k$ can be shown; this is justified by the following analysis which always gives worse branchings.

1. In this case, we have to tackle the following equations:

$$\begin{aligned} T_3^0(k) &= T_3^0(k-1) + T_3^2(k) \\ T_3^1(k) &= T_3^0(k-1) + T_3^2(k-1) \\ T_3^2(k) &= T_3^1(k-1) + T_3^2(k-1) = T_3^0(k-2) + T_3^2(k-1) + T_3^2(k-2) \end{aligned}$$

   The first equation gives $T_3^2(k) = T_3^0(k) - T_3^0(k-1)$; plugged in the last relation, we get:

$$(T_3^0(k) - T_3^0(k-1)) = (T_3^0(k-1) - T_3^0(k-2)) + T_3^0(k-2) + (T_3^0(k-2) - T_3^0(k-3))$$

   This simplifies as follows:

$$T_3^0(k) = 2T_3^0(k-1) + T_3^0(k-2) - T_3^0(k-3)$$

   which implies that $T_3^0(k) = c_1^k$ with $c_1 \leq \boxed{2.2470}$.

2. Similar algebra gives $T_3^0(k) = c_2^k \leq 2.1701^k$.

3. Some algebra yields $T_3^0(k) = c_3^k \leq \boxed{2.2470}^k$.

In other words, even with the two worst-case cases (highlighted by putting frames around them) we arrive at a better branching behavior than Niedermeier and Rossmanith did with their more intricate "bottom-up" approach.

By a more involved but similar analysis (based on slightly different heuristic priorities and two more reduction rules), together with the kernelization as explained by Niedermeier and Rossmanith, we can show:

Theorem 11 3-Hitting Set *can be solved in* $\mathcal{O}(2.1788^k + n)$ *time.*

4 Conclusions

We have developed and analyzed new, *simple* parameterized algorithms for 3-Hitting Set. We believe that the methodology we used (deeply analyzing search-tree behaviors of simple algorithms) is suitable for the development of other improved parameterized algorithms, as well. In order to apply this method, we need a kind of second *auxiliary parameter* in the problem which we try to improve on in case the main parameter cannot be improved. In the case of Hitting Set, the number of edges of small degree is such an auxiliary parameter.

We are currently working on similar problems in order to apply the "top-down" analysis methodology to them. More specifically, recent work on d-Hitting Set for $d \geq 4$, Weighted Hitting Set and on biplanarization problems (improving on the constants derived in [5]) gives promising results.

It would be interesting to see if the approach presented in this paper can be combined with other attempts at improving on the analysis of search-trees, as reported in [3, 7, 8]; this might lead to even better upper bounds on the running times of exact algorithms.

References

[1] J. Alber, H. Fan, M. R. Fellows, H. Fernau, R. Niedermeier, F. Rosamond, and U. Stege. Refined search tree techniques for the planar dominating set problem. In *Mathematical Foundations of Computer Science (MFCS)*, volume 2136 of *LNCS*, pp. 111–122, 2001.

[2] J. Chen, I. A. Kanj, and W. Jia. Vertex cover: further observations and further improvements. *Journal of Algorithms*, 41:280–301, 2001.

[3] J. Chen, I. A. Kanj, and G. Xia. Labeled search trees and amortized analysis: improved upper bounds for NP-hard problems. In *International Symposium on Algorithms and Computation (ISAAC)*, volume 2906 of *LNCS*, pp. 148–157, 2003.

[4] I. Dinur, V. Guruswami, S. Khot, and O. Regev. A new multilayered PCP and the hardness of hypergraph vertex cover. In *ACM Symp. on Theory of Computing (STOC)*, pp. 595–601, 2003.

[5] V. Dujmović, M. Fellows, M. Hallett, M. Kitching, G. Liotta, C. McCartin, N. Nishimura, P. Ragde, F. Rosemand, M. Suderman, S. Whitesides, and D. R. Wood. A fixed-parameter approach to two-layer planarization. In *International Symp. on Graph Drawing (GD'01)*, volume 2265 of *LNCS*, pp. 1–15, 2002.

[6] R. S. Garfinkel and G. L. Nemhauser. *Integer Programming*. John Wiley & Sons, 1972.

[7] J. Gramm, J. Guo, F. Hüffner, and R. Niedermeier. Automated generation of search tree algorithms for graph modification problems. In *Algorithms—ESA*, volume 2832 of *LNCS*, pp. 642–653, 2003.

[8] O. Kullmann. New methods for 3-SAT decision and worst-case analysis. *Theoretical Computer Science*, 223:1–72, 1999.

[9] R. Niedermeier and P. Rossmanith. An efficient fixed-parameter algorithm for 3-Hitting Set. *Journal of Discrete Algorithms*, 1:89–102, 2003.

[10] R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32:57–95, 1987.

[11] K. Weihe. Covering trains by stations or the power of data reduction. In *Algorithms and Experiments ALEX 98*, pp. 1–8. `http://rtm.science.unitn.it/alex98/proceedings.html`, 1998.

5 Appendix I: Omitted proofs

5.1 Proof of Lemma 1

Proof. Formally, the proof is done by induction on the number of vertices of the graph. The base is solved by rules for simple instances. Assume that the correctness of the algorithm has been shown for all graphs up to n vertices. Consider now an instance (G, k) where G has $n+1$ vertices. If after applying the reduction rules, G has n vertices, the correctness follows by the induction hypothesis. Otherwise, let us call the resulting instance (G, k), as well. Possibly, (G, k) can be correctly solved by rules for simple instances. If not, we encounter a binary branching, covering two mutually exclusive cases for some chosen vertex x:

1. If x is taken into the hitting set, then in the recursive call the following instance is created:
   - x is removed from the vertex set; hence, the induction hypothesis is applicable to the created instance.
   - The parameter k is accordingly decremented.
   - x is put into the hitting set S which is going to be recursively constructed.
   - All hyperedges to which x belongs are covered and hence deleted.
2. If x is not put into the hitting set, then in the recursive call the following instance is created:
   - x is removed from the vertex set; hence, the induction hypothesis is applicable to the created instance.
   - The hitting set S which is going to be recursively constructed is not changed.
   - The parameter k is not changed.
   - From all hyperedges to which x belongs, x is removed.

The second branch is only executed when the first branch returns the empty set. This is correct, since x will be in each correct cover that is returned in the first case, and hence if the empty set is returned, this clearly signals that the corresponding branch shows no solution.

Since the described actions are obviously correct, the correctness of the algorithm follows by induction. ∎

5.2 Proof of Lemma 2

Although the reduction rules were elsewhere mentioned, we are not aware of any formal correctness proof. Since the soundness of the reductions is essential to the correctness of the overall algorithm, we provide the corresponding soundness proofs right here.

Proof. We have to show that, whenever S is a solution to an instance (G, k), then there is a solution S' to the instance (G', k'), where (G', k') is obtained from (G, k) by applying any of the reduction rules. We must also show the converse direction.

1. (hyper)edge domination: Let e and f be hyperedges in G such that $f \subset e$. If S is a solution to (G, k), then trivially S is also a solution to the instance (G', k) obtained from (G, k) by applying the edge domination rule to the situation $f \subset e$. Conversely, if S' is a solution to (G', k), then in particular $S' \cap f \neq \emptyset$, which means that $S' \cap e \neq \emptyset$, so that S' is a solution to (G, k), as well.

2. tiny edges: Hyperedges of degree one can only be covered by the vertices they contain.

3. vertex domination: Let x and y be vertices in G such that x is dominated by y. If S is a solution to (G, k) which does not contain x, then S is also a solution to the instance (G', k) obtained from (G, k) by applying the vertex domination rule triggered by the domination of x by y. If S is a solution to (G, k) which contains x, then because x is dominated by y, $(S \setminus \{x\}) \cup \{y\}$ is also a solution to (G, k) and, by the preceding sentence, this is a solution to the reduced instance (G', k), as well. Conversely, if S' is a solution to (G', k), S' is a solution to (G, k), since no edges are deleted when forming G'. ∎

5.3 Proof of Lemma 3

Proof. In the reduced instance, there are no vertices of degree zero, since this would trigger the vertex domination rule, because there are at least two vertices. The vertex domination rule would also get rid of vertices of degree one that are connected to other vertices. Vertices of degree one that are not connected to other vertices are coped with by the tiny edge rule. ∎

5.4 Proof of Lemma 4

Proof. Not taking x into the cover means that, in order to cover edges $e \in E$ with $x \in e$, some $z \in e \setminus \{x\}$ must go into the cover. Therefore, the "next instance" is $G - x$. The question is if $e \setminus \{x\} \in E$ for any e with $x \in e$, because then less than $\delta^d(x)$ edges of degree $(d-1)$ would be created. But this is ruled out by the edge domination rule. Since $\delta^{d-1}(x)$ edges of degree $(d-1)$ are "destroyed" in $G - x$, the formula is valid. ∎

5.5 Some algebra left out before Theorem 11

We are looking for a solution c_i such that $T_3^0(k) = c_i^k$ for $i = 1, 2, 3$, corresponding to case (i) in the list of equations above. Note that we left out the cases where obviously $T_3^j(k) \leq 2^k$ can be shown; this is justified by the following analysis which always gives worse branchings.

1. In this case, we have to tackle the following equations:

$$\begin{aligned} T_3^0(k) &= T_3^0(k-1) + T_3^2(k) \\ T_3^1(k) &= T_3^0(k-1) + T_3^2(k-1) \\ T_3^2(k) &= T_3^1(k-1) + T_3^2(k-1) = T_3^0(k-2) + T_3^2(k-1) + T_3^2(k-2) \end{aligned}$$

The first equation gives $T_3^2(k) = T_3^0(k) - T_3^0(k-1)$; plugged in the last relation, we get:

$$(T_3^0(k) - T_3^0(k-1)) = (T_3^0(k-1) - T_3^0(k-2)) + T_3^0(k-2) + (T_3^0(k-2) - T_3^0(k-3))$$

This simplifies as follows:

$$T_3^0(k) = 2T_3^0(k-1) + T_3^0(k-2) - T_3^0(k-3)$$

which implies that $T_3^0(k) = c_1^k$ with $c_1 \leq \boxed{2.2470}$.

2.

$$\begin{aligned} T_3^0(k) &= T_3^0(k-1) + T_3^2(k) \\ T_3^1(k) &= T_3^0(k-1) + T_3^2(k-1) \\ T_3^2(k) &= T_3^0(k-2) + T_3^1(k-1) \end{aligned}$$

Hence, $T_3^2(k) = 2T_3^0(k-2) + T_3^2(k-2)$ which implies

$$T_3^0(k) - T_3^0(k-1) = 3T_3^0(k-2) - T_3^0(k-3),$$

meaning that $T_3^0(k) \leq 2.1701^k$.

3.

$$\begin{aligned} T_3^0(k) &= T_3^0(k-1) + T_3^2(k) \\ T_3^2(k) &= T_3^0(k-1) + T_3^2(k-2) \end{aligned}$$

By using the first equation, we get:

$$(T_3^0(k) - T_3^0(k-1)) = T_3^0(k-1) + (T_3^0(k-2) - T_3^0(k-3))$$

which yields

$$T_3^0(k) = 2T_3^0(k-1) + T_3^0(k-2) - T_3^0(k-3)$$

Therefore $T_3^0(k) = c_3^k \leq \boxed{2.2470}^k$.

6 Appendix II: 3-Hitting Set; advanced analysis

Lemma 7 may give us another perspective on our analysis: we finally must analyze T_3^3, as well. This will be our first attempt at improving the analysis of 3-Hitting Set. To this end, we will refine our heuristic priorities (in order to simplify the case analysis for T_3^3) and design two new reduction rules (special to the case $d = 3$). The following time analysis shows that the "weak point" of the algorithm (leading to the worst-case running times) is when the three edges of degree two intersect in a star-like fashion. This motivates a further modification of the heuristic priorities. A further look at the situations which then lead to the worst-case branchings drives us into a deeper combinatorial analysis, again slightly modifying the heuristic priorities. Since at that time, many different situations are at least very close to the worst-case branching, we stop the analysis.

The message to be taken from the presentation is that we can afford having our analysis guided by the demand for improving on the "weakest spot" of the earlier running time analysis. As long as the changes in the heuristic priorities only affect these special situations, we need not have another look at the other parts of the earlier analysis. Adding further reduction rules won't affect the validity of earlier analysis, either. This sort of modularity is therefore very helpful to improve on algorithms to an extent that is hard to achieve with the traditional, bottom-up methodology.

6.1 Refined heuristic priorities and more reduction rules

In order to be able to better handle the situation that some of the small edges don't intersect, we refine our heuristic priorities as follows. Note that the additional heuristic rule we introduce does not affect the analysis of T_3^ℓ for $\ell < 3$.

*H*0 *Prefer small edges.* More formally, let $\delta^E_{\min} = \min\{\delta(e) \mid e \in E\}$ and let $E_0 = \{e \in E \mid \delta(e) = \delta^E_{\min}\}$.

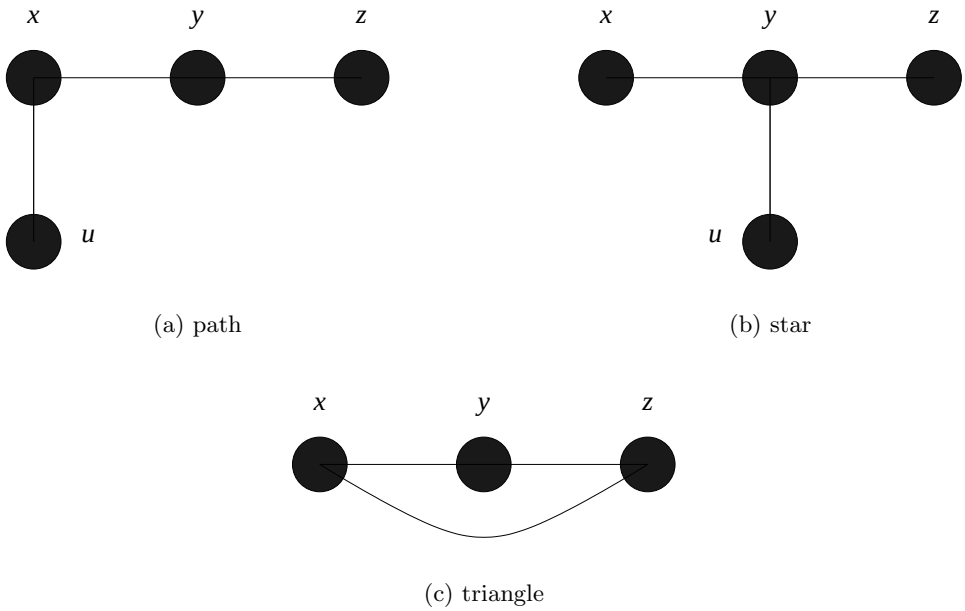
*H*0′ *Prefer small lonely edges* If $\delta^E_{\min} = 2$, then if there is an edge $e' \in E_0$ such that $\forall e \in E_0 \setminus \{e'\}(e \cap e' = \emptyset)$ then set $E_0 = \{e'\}$.

*H*1 *Maximize Eq. (1)*, i.e., let $V_1 = \{x \in \bigcup_{e \in E_0} \mid \delta^3(x) - \delta^2(x)$ is maximum$\}$.

*H*2 Choose some $x \in V_1$ of *maximum degree.*

Moreover, we employ the following two *special case reduction rules*, specific to the case of 3-Hitting Set:

1. path reduction Let $G = (V, E)$ contain the small edges $e_1 = \{x, y\}$, $e_2 = \{y, z\}$ and $e_2 = \{u, x\}$, $u \neq z$. Assume that $\delta(x) = \delta(y) = 2$. If z is dominated by u in $G - e_2 = (V, E \setminus \{e_2\})$, then delete z, i.e., reduce G to $G' = G - z = (V \setminus \{z\}, \{e \setminus \{z\} \mid e \in E\})$ without changing the value of the parameter.



(a) path (b) star

(c) triangle

Figure 1: Three situations for arranging three edges.

2. triangle reduction Consider the instance (G, k), where $G = (V, E)$ contains the small edges $e_1 = \{x, y\}$, $e_2 = \{y, z\}$ and $e_3 = \{x, z\}$ with $\delta(x) = 2$. Then, put y into the cover, i.e., reduce to $(G', k-1)$ with $G' = (V', E')$, $E' = \{e \in E \mid y \notin e\}$ and $V' = \{v \in V \mid \exists e \in E'(v \in e)\}$.

These rules can be seen as a variant of the vertex domination rule; small pictures can be found in Fig. 1(a) and Fig. 1(c), respectively.

Lemma 12 *The path reduction rule is sound.*

Proof. Consider a situation as described in the reduction rule. Assume that S is a solution to the instance $G = (V, E)$. Then if $z \notin S$, S is also a solution to $G - z$. If $z \in S$, then

- if $u \in S$, then since z is dominated by u in $G - e_3$, the only "purpose" of $z \in S$ is to cover e_2, so that $S \cup \{y\} \setminus \{z\}$ is another solution of G and of $G - z$;
- if $u \notin S$, then $x \in S$ in order to cover e_3; then, $S \cup \{u, y\} \setminus \{x, z\}$ is another solution of G and of $G - z$.

Conversely, if S' is a solution to the instance $G - z$, we can assume that S' was obtained by first applying the (general) reduction rules to $G - z$. Hence, $y \in S'$ by the tiny edge rule. Therefore, $x \notin S'$ by the vertex domination rule that puts

$u \in S'$. Because z is dominated by u in $G - e_3$, S' is also a solution to the instance G. ∎

Lemma 13 *The triangle reduction rule is sound.*

Proof. Consider a situation as described in the reduction rule. Assume that S is a solution to the instance $G = (V, E)$. If $y \in S$, then $S' = S \setminus \{y\}$ is clearly a solution to the reduced instance G'. Conversely, if S' is a solution to G', then any instance obtained from G' by adding a vertex y plus some incidence relations with y can be solved by $S = S' \cup \{y\}$; in particular, this is true for G.

Hence, we only have to consider that S is a solution to the instance $G = (V, E)$ with $y \notin S$. Obviously, the "triangle" formed by x, y, z requires two out of these three vertices to go into the cover. Hence, $x, z \in S$. Since $z \in S$, then only "purpose" of x being in the hitting set is to cover the edge $\{x, y\}$. This can be done equally well by putting y instead of x into the hitting set, so that $S' = S \setminus \{x\}$ is a solution to the reduced instance G' (as argued before). ∎

6.2 Analyzing T_3^3

Lemma 14 $T_3^3(k) \leq \max\{T_3^0(k-2) + T_3^1(k-2), T_3^1(k-1) + T_3^1(k-2), T_3^0(k-3) + T_3^1(k-1) + T_3^1(k-2), T_3^0(k-1) + T_3^1(k-3), 2T_3^2(k-1)\}$.

Proof. Let e_1, e_2 and e_3 denote the edges of degree two.

If not all the three edges of degree two interact, say $e_3 \cap e_2 = \emptyset$ and $e_3 \cap e_2 = \emptyset$, then the heuristic priority $H0'$ would choose say e_3. The chosen vertex $x \in e_3$ which would be branched at satisfies $\delta(x) \geq 2$ according to Lemma 3, so that $\delta^3(x) \geq 1$ follows. Since the other two small edges remain untouched, a trivial branching analysis yields:

$$T_3^3(k) \leq T_3^2(k-1) + T_3^3(k-1). \tag{4}$$

Observe that—in the case when x is not taken into the cover—the fact that the graph instance is reduced implies that at least one new small edge is created.

Assume then that all three edges of degree two interact. Formally, this means that the graph induced by the edge set $E_0 = \{e \in E \mid \delta(e) = 2\}$ is connected. This could happen in three different ways, as depicted in Figure 1. Namely, without loss of generality, we may assume that $e_1 = \{x, y\}$ and $e_2 = \{y, z\}$ are two of the edges. A third edge of degree two can interact with these two given ones in essentially the following different ways:

1. The third edge is connected to exactly one of x or z, so that the edges form a *path*. Without loss of generality, we discuss the case of an edge $e_3 = \{u, x\}$ in the following, with $u \neq z$.
2. The third edge is connected to y: $e_1 = \{x, y\}$, $e_2 = \{y, z\}$, $e_3 = \{y, u\}$. This means that the edge-2-component forms a small *star*.
3. The third edge is connecting both x and z, yielding a *triangle*.

We will discuss the mentioned three situations in details in what follows. Observe that Lemma 3 applies, since we are always dealing with reduced instances, so that all vertices of interest have degree at least two.

1. $e_1 = \{x, y\}$, $e_2 = \{y, z\}$, $e_3 = \{u, x\}$, with $u \neq z$. We distinguish several sub-cases:

   (a) $\delta(x) = \delta(y) = 2$. After having applied the reduction rules, $\delta(u), \delta(z) \geq 2$. W.l.o.g., $\delta(z) \leq \delta(u)$. $H1$ would let us branch at u. If u is put into the cover, then x would not go into the cover by the vertex domination rule (see Lemma 3), so that the tiny edge rule puts y into the cover, as well. This is a $T_3^0(k-2)$-branch. If u is not put into the cover, then x and z will go into the cover, the latter by vertex domination. Unfortunately, the new edge (formerly incident with u) which would be created could be already covered by z, so we cannot count this edge that easily. But if all edges incident with u (besides e_3) are also incident with z, the path reduction rule would have applied. Hence, we can be sure that in the case that u is not put into the cover, always one new small edge is created that does not contain z. Therefore,

   $$T_3^3(k) \leq T_3^0(k-2) + T_3^1(k-2). \tag{5}$$

   (b) If $\delta(x) > 2$ (the case $\delta(y) > 2$ is symmetric), then first assume that we branch at u. Due to the priority H2, this means that $\delta(u) > 2$. A simple branching analysis then gives:

   $$T_3^3(k) \leq T_3^2(k-1) + T_3^3(k-1).$$

   This equation already showed up in (4).

   (c) If $\delta(x) > 2$ and we branch at x, then two sub-cases arise:

   - If $\delta(u) = 2$, $\delta(x) = 3$ is the worst case (cf. H2). In the case that we put x into the cover, u will be dominated by its two neighbors and be hence discarded by the vertex domination rule, producing another edge of degree two this way. Thereby, we get

     $$T_3^3(k) \leq T_3^2(k-1) + T_3^2(k-2).$$

     Observe that this inequality is always better than Eq. (9) below (also better than the refined analysis of that situation). Therefore, we can ignore it in the following.
   - If $\delta(u) > 2$, then $\delta(x) > 3$. A simple branching analysis then gives:

     $$T_3^3(k) \leq T_3^1(k-1) + T_3^2(k-2).$$

     Again, this situation is better than Eq. (9).

2. $e_1 = \{x, y\}$, $e_2 = \{y, z\}$, $e_3 = \{y, u\}$. These three edges form a small star. Again, we consider some sub-cases:

   (a) If $\delta(y) = 3$, then we branch (w.l.o.g.) at x due to the reduction rules, since $\min\{\delta(u), \delta(x), \delta(z)\} \geq 2$, so that $\delta^3(x) \geq 1$. If x is taken into the cover, then in the next recursive call of our algorithm, the edge-2-component $\{e_2, e_3\}$ will be analyzed.[2] Looking into the proof of Lemma 10 tells us that the $T_3^2(k) \leq T_3^0(k-2) + T_3^1(k-1)$ case is encountered, since $\delta(y) = 2$ then. If x is not put into the cover, y will be. Moreover, at least one new edge of degree two is created. Altogether, we obtain:

   $$T_3^3(k) \leq T_3^0(k-3) + T_3^1(k-1) + T_3^1(k-2). \tag{6}$$

   (b) If $\delta(y) \geq 4$ and if we branch at y, then a trivial analysis yields:

   $$T_3^3(k) \leq T_3^0(k-1) + T_3^1(k-3). \tag{7}$$

   (c) If $\delta(y) \geq 4$ and if we branch at say x. By the heuristic priorities, We have $\delta(x) \geq 3$. So, not taking x into the cover gives two new edges of degree two. A simple branching analysis would then yield:

   $$T_3^3(k) \leq 2T_3^2(k-1). \tag{8}$$

   Observe that this relation is always worse than the inequality derived in Eq. (4), so that Eq. (4) is not mentioned in the formulation of the lemma.

3. $e_1 = \{x, y\}$, $e_2 = \{y, z\}$, $e_3 = \{x, z\}$. Hence, the edge-2-component is a triangle. By the triangle reduction rule, $\delta(x), \delta(y), \delta(z) \geq 3$. If x is going into the cover, then the edge $\{y, z\}$ will remain "untouched." If x is not put into the cover, then both y and z must go into the cover, and at least one edge (incident to x) is now an edge of degree two. Therefore, we get

   $$T_3^3(k) \leq T_3^1(k-1) + T_3^1(k-2). \tag{9}$$

   Note that Eq. (9) is always better than Eq. (6) and will be hence ignored.

The reader can easily check that this is a complete case distinction.

Note that we already simplified the "limiting expression" in the formulation of this lemma as far as possible: whenever two "comparable" branching arose, we erased the better one. ∎

[2] A locality-preserving implementation will do so also in the case of $T_3^{\geq 3}$.

6.3 Solving the recurrence equations

Note that we can stop our analysis here, since we have no recurrences for T_3^3 which refer to say T_3^4. Since we have only one case for T_3^1 to analyze—namely, $T_3^1(k) \leq T_3^0(k-1) + T_3^2(k-1)$—and since the expressions for T_3^3 don't contain T_3^3 itself, we can rewrite the equations for T_3^3 as equations for T_3^0 as follows:

$$T_3^0(k) \overset{\text{Eq.(5)}}{\leq} T_3^0(k-1) + T_3^0(k-2) + T_3^0(k-3) + T_3^2(k-3) \tag{10}$$

$$T_3^0(k) \overset{\text{Eq.(6)}}{\leq} T_3^0(k-1) + T_3^0(k-2) + 2T_3^0(k-3) + T_3^2(k-2) + T_3^2(k-3) \tag{11}$$

$$T_3^0(k) \overset{\text{Eq.(7)}}{\leq} 2T_3^0(k-1) + T_3^0(k-4) + T_3^2(k-4) \tag{12}$$

$$T_3^0(k) \overset{\text{Eq.(8)}}{\leq} T_3^0(k-1) + 2T_3^2(k-1) \tag{13}$$

This plugging-in also shows that Eq. (10) is always better than Eq. (11) and needs not be considered in what follows. Let us now discuss the two worst-case scenarios from Lemma 10. Note that the left-out sub-case in the analysis of that Lemma already gives a satisfactory branching number when analysed up to T_3^2, and hence needs no further refined analysis with the help of T_3^3.

Case 1: $T_3^2(k) = T_3^1(k-1) + T_3^2(k-1) = T_3^0(k-2) + T_3^2(k-1) + T_3^2(k-2)$ **I**

A simple argument shift in Eq. (11) gives:

$$T_3^0(k+1) = T_3^0(k) + T_3^0(k-1) + 2T_3^0(k-2) + T_3^2(k-1) + T_3^2(k-2) \text{ II}$$

Subtracting II from I, we get:

$$T_3^2(k) = T_3^0(k+1) - T_3^0(k) - T_3^0(k-1) - T_3^0(k-2)$$

Eq. (11) can be hence rewritten as follows:

$$\begin{aligned} T_3^0(k) &= T_3^0(k-1) + T_3^0(k-2) + 2T_3^0(k-3) \\ &\quad + (T_3^0(k-1) - T_3^0(k-2) - T_3^0(k-3) - T_3^0(k-4)) \\ &\quad + (T_3^0(k-2) - T_3^0(k-3) - T_3^0(k-4) - T_3^0(k-5)) \\ &= 2T_3^0(k-1) + T_3^0(k-2) - 2T_3^0(k-4) - T_3^0(k-5). \end{aligned}$$

This gives

$$\boxed{T_3^0(k) \leq 2.2274^k}.$$

Since Eq. (12) only contains one T_3^2 occurrence, we can immediately get the following equation by plugging into I: $T_3^0(k+4) - 2T_3^0(k+3) - T_3^0(k) = T_3^0(k-$

$2) + (T_3^0(k+3) - 2T_3^0(k+2) - T_3^0(k-1)) + (T_3^0(k+2) - 2T_3^0(k+1) - T_3^0(k-2))$. This yields

$$0 = T_3^0(k+4) - 3T_3^0(k+3) + T_3^0(k+2) + 2T_3^0(k+1) - T_3^0(k) + T_3^0(k-1).$$

Hence,

$$T_3^0(k) \leq 2.1638^k.$$

Similarly, Eq. (13) can be used, after multiplying the equation for T_3^2 with two. $T_3^0(k+1) - T_3^0(k) = 2T_3^0(k-2) + (T_3^0(k) - T_3^0(k-1)) + (T_3^0(k-1) - T_3^0(k-2))$. Therefore,

$$0 = T_3^0(k+1) - 2T_3^0(k) + 0T_3^0(k-1) - T_3^0(k-2).$$

This gives

$$\boxed{T_3^0(k) \leq 2.2056^k}.$$

Case 2: $T_3^2(k) = T_3^0(k-1) + T_3^2(k-2)$
Eq. (11) is now a bit tricky. We first make the ansatz that $T_3^0(k) = c^k$. This gives: $0 = c^k - c^{k-1} - c^{k-2} - 2c^{k-3} - (T_3^2(k-2) + T_3^2(k-3))$. Applying the recursion given by Case 2 entails:

$$T_3^2(k-2) + T_3^2(k-3) = T_3^0(k-3) + T_3^2(k-3) + T_3^2(k-4).$$

Repeating this argument, we obtain:

$$T_3^2(k-2) + T_3^2(k-3) = \sum_{i=0}^{k-3} T_3^0(i).$$

Hence, our ansatz yields:

$$\begin{aligned} 0 &= c^k - c^{k-1} - c^{k-2} - 2c^{k-3} - \sum_{i=0}^{k-3} c^i \\ &= c^k - c^{k-1} - c^{k-2} - 2c^{k-3} - (c^{k-2} - 1)/(c-1) \\ &= c^{k+1} - c^k - c^{k-1} - 2c^{k-2} - c^{k-2} + 1 - (c^k - c^{k-1} - c^{k-2} - 2c^{k-3}) \\ &\approx c^{k+1} - 2c^k + 0c^{k-1} - 2c^{k-2} + 2c^{k-3} \end{aligned}$$

This gives

$$\boxed{T_3^0(k) \leq 2.2227^k}.$$

Eq. (12) can be solved as before: $T_3^0(k+4) - 2T_3^0(k+3) - T_3^0(k) = T_3^0(k-1) + (T_3^0(k+2) - 2T_3^0(k+1) - T_3^0(k-2))$. Hence,

$$T_3^0(k+4) - 2T_3^0(k+3) - T_3^0(k+2) + 2T_3^0(k+1) - T_3^0(k) - T_3^0(k-1) + T_3^0(k-2).$$

This gives

$$T_3^0(k) \leq 2.1582^k.$$

Similarly, Eq. (13) can be used, after multiplying the equation for T_3^2 with two. $T_3^0(k+1) - T_3^0(k) = 2T_3^0(k-1) + (T_3^0(k-1) - T_3^0(k-2))$. Therefore, $0 = T_3^0(k+1) - T_3^0(k) - 3T_3^0(k-1) + T_3^0(k-2)$. This gives

$$T_3^0(k) \leq 2.1701^k.$$

We boxed the branching numbers which (albeit being better than the worst case in the preceding section) are worse than the "best case" in the previous section.

6.4 A further hit at 3-Hitting Set

Regarding Lemma 14, we can observe that all worst cases are found in the "star case," of which the branch at the center appears to yield the best branching behavior. Is it after all favorable to branch in centers of "large stars" even if this contradicts the heuristic priorities we have set up to now? Let us try. The trivial branch then yields

$$T_3^3(k) \leq T_3^0(k-1) + T_3^0(k-3).$$

Hence,

$$T_3^0(k) \leq 2T_3^0(k-1) + T_3^0(k-3) \leq \boxed{2.2056^k}. \tag{14}$$

Is there some intuition why this branch is (sometimes) more favorable then the analysis we did up to now? Maybe, we simply neglected the effect which is incurred by taking lots of vertices into the cover in one stroke. So, our heuristic priorities would now become:

*H*0 *Prefer small edges.* More formally, let $\delta^E_{\min} = \min\{\delta(e) \mid e \in E\}$ and let $E_0 = \{e \in E \mid \delta(e) = \delta^E_{\min}\}$.

*H*0′ *Prefer small lonely edges* If $\delta^E_{\min} = 2$, then if there is an edge $e' \in E_0$ such that $\forall e \in E_0 \setminus \{e'\}(e \cap e' = \emptyset)$ then set $E_0 = \{e'\}$.

H_{new} *Prefer center branch* If there is an $x \in \bigcup_{e \in E_0} e$ with $\delta^2(x) > 2$, then update $E_0 = \{x\}$.

*H*1 *Maximize Eq. (1)*, i.e., let $V_1 = \{x \in \bigcup_{e \in E_0} \mid \delta^3(x) - \delta^2(x)$ is maximum$\}$.

*H*2 Choose some $x \in V_1$ of *maximum degree.*

Observe that rule H_{new} can only trigger if there are at least three edges of small degree in the instance, so that it does not influence our earlier analysis of T_3^0, T_3^1 or T_3^2. Furthermore, it is obvious that the mentioned "star case" is the only place where H_{new} applies to, so that the rest of our analysis is still valid.

The revised lemma for T_3^3 now reads as follows, where the recurrences are listed according to the sequence they appear; they correspond to Eq. (4), Eq. (5), Eq. (9), and Eq. (14). Notice that the first and the third equations were "neglected" by the earlier analysis, since they were already "covered" by worse cases in the former "star analysis." Since the analysis of the star case did change, they have to be reconsidered now.

Lemma 15 $T_3^3(k) \leq \max\{T_3^2(k-1) + T_3^3(k-1), T_3^0(k-2) + T_3^1(k-2), T_3^1(k-1) + T_3^1(k-2), T_3^0(k-1) + T_3^0(k-3)\}$.

Since the first recurrence in that list is surely better than Eq. (13) analyzed above (which in turn was based on Eq. (8)), and since that analysis gave the same branching number as Eq. (14), we need not deal with that case further here. The second and third recurrences still have to be analyzed.

The second recursion yields

$$T_3^3(k) = T_3^0(k-2) + T_3^1(k-2) = T_3^0(k-2) + T_3^0(k-3) + T_3^2(k-3),$$

while the third recursion gives:

$$T_3^3(k) = T_3^1(k-1) + T_3^1(k-2) = T_3^0(k-2) + T_3^0(k-3) + T_3^2(k-2) + T_3^2(k-3).$$

Since this is always worse than the previous recursion, we only have to consider the following case:

$$T_3^0(k) = T_3^0(k-1) + T_3^0(k-2) + T_3^0(k-3) + T_3^2(k-2) + T_3^2(k-3). \quad (15)$$

According to the remarks from the previous section, the possible worst case scenarios left to be analyzed are two. Observe that Eq. (15) is always better than Eq. (11).

1. $$\begin{aligned} T_3^0(k) &= T_3^0(k-1) + T_3^0(k-2) + T_3^0(k-3) + T_3^2(k-2) + T_3^2(k-3) \\ T_3^2(k-1) &= T_3^0(k-3) + T_3^2(k-2) + T_3^2(k-3) \end{aligned}$$

   Hence, $T_3^2(k-1) = T_3^0(k) + T_3^0(k-3) - T_3^0(k-1) - T_3^0(k-2) - T_3^0(k-3) = T_3^0(k) - T_3^0(k-1) - T_3^0(k-2)$, which yields
   $0 = (-T_3^0(k) + T_3^0(k-1) + T_3^0(k-2)) + T_3^0(k-3) + (T_3^0(k-1) - T_3^0(k-2) - T_3^0(k-3)) + (T_3^0(k-2) - T_3^0(k-3) - T_3^0(k-4)) =$
   $-T_3^0(k) + 2T_3^0(k-1) + T_3^0(k-2) - T_3^0(k-3) - T_3^0(k-4)$. Hence,

   $$T_3^0(k) \leq 2.1479^k.$$

2. $$\begin{aligned} T_3^0(k) &= T_3^0(k-1) + T_3^0(k-2) + T_3^0(k-3) + T_3^2(k-2) + T_3^2(k-3) \\ T_3^2(k) &= T_3^0(k-1) + T_3^2(k-2) \end{aligned}$$

   We again make the ansatz that $T_3^0(k) = c^k$. After some algebra as in the previous section, this gives

   $$T_3^0(k) \leq 2.1177^k.$$

Let us look again where the worst case scenarios appeared, leading to the exponential base 2.2056:

1. We took over the analysis of Eq. (13).
2. According to the revised heuristic priorities, Eq. (14) arose.

We try to cope with these situations in the next section.

6.5 A final hit at 3-Hitting Set

First of all, we should look what happens if we actually and explicitly analyze Eq. (4). Recall that this relation explicitly appears in the formulation of Lemma 15, but we refrained from an explicit analysis by referring to the earlier analysis of Eq. (13) which was "good enough" at that point. This is important, since if we would gain nothing compared to our "shortcut analysis," we would not like to analyze this case further.

Is it all worthwhile doing? With the identity $T_3^0(k) = T_3^0(k-1) + T_3^3(k)$, we get

$$\begin{aligned} T_3^2(k-1) &= T_3^3(k) - T_3^3(k-1) \\ &= T_3^0(k) - 2T_3^0(k-1) + T_3^0(k-2) \end{aligned}$$

Again, we have to distinguish two sub-cases stemming from our analysis of T_3^2:

- $T_3^2(k) = T_3^1(k-1) + T_3^2(k-1) = T_3^0(k-2) + T_3^2(k-1) + T_3^2(k-2)$:
  Plugging in the expression we just derived for $T_3^2(k-1)$ yields:
  $0 = (-T_3^0(k+1) + 2T^0(k) - T_3^0(k-1)) + T_3^0(k-2) + (T_3^0(k) - 2T_3^0(k-1) + T_3^0(k-2)) + (T_3^0(k-1) - 2T_3^0(k-2) + T_3^0(k-3)) =$
  $-T_3^0(k+1) + 3T_3^0(k) - 2T_3^0(k-1) + T_3^0(k-3)$. Hence,

  $$T_3^0(k) \leq \boxed{2.1788^k}.$$

- $T_3^2(k) = T_3^0(k-1) + T_3^2(k-2)$:
  A similar plug-in gives:

  $$0 = -T_3^0(k+1) + 2T_3^0(k) + T_3^0(k-1) - 2T_3^0(k-2) + T_3^0(k-3)$$

  Hence,

  $$T_3^0(k) \leq 2.1323^k.$$

In short, it might be worthwhile further analyzing the second bad situation after all. In fact, the framed branching number will turn out to be the worst case in this analysis, which again backs our intention to stop the analysis here, since otherwise a detailed analysis of T_3^4 seems to be inevitable. Correctly keeping track of all those situations is probably close to a nightmare.

Observe that up to now we only did a very rough analysis of the star case branching from the center. Can we improve on the general situation under some circumstances? In the following discussion, we refer again to Fig. 1(b).

1. If the center vertex y has degree four, then we would gain a new edge of degree two if we don't take y into the cover. More specifically, we get the recurrence:

   $$T_3^3(k) \leq T_3^0(k-1) + T_3^1(k-3) \tag{16}$$

2. If the center vertex y has degree three and at least one of the vertices u, x, z has degree two, we would gain a new edge of degree two if we take y into the cover by the reduction rules. Then, we get:

$$T_3^3(k) \leq T_3^1(k-1) + T_3^0(k-3) \tag{17}$$

3. The worst case leading to Eq. (4) actually happens when $\delta(y) = 3$ and $\delta(u), \delta(x), \delta(z) \geq 3$. This situation might be handled by again changing our heuristic priorities, since in that case $\delta^3(x) - \delta^3(y) \geq 2$, so that heuristic priority $H1$ (if that would be bothered about again) would very clearly prefer branching at x to branching at y. We will discuss details below.

To estimate if the idea of changing heuristic priorities (again) might be useful at all, let us compute the branching behavior for Eq. (16) and for Eq. (17):

Analyzing Eq. (16): Using Lemma 9, we get

$$T_3^3(k) = T_3^0(k-1) + T_3^1(k-3) = T_3^0(k-1) + T_3^0(k-4) + T_3^2(k-4).$$

With the identity $T_3^0(k) = T_3^0(k-1) + T_3^3(k)$, we obtain

$$T_3^2(k-4) = T_3^0(k) - 2T_3^0(k-1) - T_3^0(k-4).$$

Again, we have to distinguish two sub-cases stemming from our analysis of T_3^2:

- $T_3^2(k) = T_3^1(k-1) + T_3^2(k-1) = T_3^0(k-2) + T_3^2(k-1) + T_3^2(k-2)$:
  Plugging in the expression we just derived for $T_3^2(k-4)$ yields:
  $0 = T_3^2(k) - T_3^0(k-2) - T_3^2(k-1) - T_3^2(k-2)$
  $= (T_3^0(k+4) - 2T_3^0(k+3) - T_3^0(k)) - T_3^0(k-2) - (T_3^0(k+3) - 2T_3^0(k+2) - T_3^0(k-1)) - (T_3^0(k+2) - 2T_3^0(k+1) - T_3^0(k-2))$
  $= T_3^0(k+4) - 3T_3^0(k+3) + T_3^0(k+2) + 2T_3^0(k+1) - T_3^0(k) + T_3^0(k-1)$. Hence,

$$T_3^0(k) \leq 2.1638^k.$$

- $T_3^2(k) = T_3^0(k-1) + T_3^2(k-2)$:
  A similar plug-in gives:

  $$0 = T_3^0(k+4) - 2T_3^0(k+3) - T_3^0(k+2) + 2T_3^0(k+1) - T_3^0(k) + T_3^0(k-1) - T_3^0(k-2).$$

  Hence,

$$T_3^0(k) \leq 2.1582^k.$$

Analyzing Eq. (17): Using Lemma 9, we get

$$T_3^3(k) = T_3^1(k-1) + T_3^0(k-3) = T_3^0(k-2) + T_3^0(k-3) + T_3^2(k-2).$$

With the identity $T_3^0(k) = T_3^0(k-1) + T_3^3(k)$, we obtain

$$T_3^2(k-2) = T_3^0(k) - T_3^0(k-1) - T_3^0(k-2) - T_3^0(k-3).$$

Again, we have to distinguish two sub-cases stemming from our analysis of T_3^2:

- $T_3^2(k) = T_3^1(k-1) + T_3^2(k-1) = T_3^0(k-2) + T_3^2(k-1) + T_3^2(k-2)$:
  Plugging in the expression we just derived for $T_3^2(k-4)$ yields:
  $0 = T_3^2(k) - T_3^0(k-2) - T_3^2(k-1) - T_3^2(k-2) =$
  $(T_3^0(k+2) - T_3^0(k+1) - T_3^0(k) - T_3^0(k-1)) - T_3^0(k-2) - (T_3^0(k+1) - T_3^0(k) -$
  $T_3^0(k-1) - T_3^0(k-2)) - (T_3^0(k) - T_3^0(k-1) - T_3^0(k-2) - T_3^0(k-3)) =$
  $T_3^0(k+2) - 2T_3^0(k+1) - T_3^0(k) + T_3^0(k-1) + T_3^0(k-2) + T_3^0(k-3)$. Hence,
  $$T_3^0(k) \leq 2.0868^k.$$
- $T_3^2(k) = T_3^0(k-1) + T_3^2(k-2)$:
  A similar plug-in gives:
  $$0 = T_3^0(k+2) - T_3^0(k+1) - 2T_3^0(k) - T_3^0(k-1) + T_3^0(k-2) + T_3^0(k-3).$$
  Hence,
  $$T_3^0(k) \leq 2.0437^k.$$

The case when $\delta(y) = 3$ and $\delta(u), \delta(x), \delta(z) \geq 3$. Since we know that we would get no improvement when branching at y, let us assume for the moment we would have modified our heuristic priorities in a way that they force us to branch at say x. By our case distinction, if x is not taken into the cover, y would be put into the cover, and in addition, two new small edges show up, giving a $T_3^2(k-1)$-branch.

If x is put into the cover, we would get another $T_3^2(k-1)$-branch. But observe that, since $\delta(y) = 3$ before the branching, we are now in the first sub-case of Lemma 10. Overall, this yields—using Lemma 9:

$$T_3^3(k) \leq T_3^2(k-1) + T_3^0(k-3) + T_3^1(k-2) \leq 2T_3^0(k-3) + T_3^2(k-1) + T_3^2(k-3).$$

With the identity $T_3^0(k) = T_3^0(k-1) + T_3^3(k)$, we obtain

$$T_3^2(k-1) + T_3^2(k-3) = T_3^0(k) - T_3^0(k-1) - 2T_3^0(k-3). \tag{18}$$

Again, we have to distinguish two sub-cases stemming from our analysis of T_3^2:

- $T_3^2(k) = T_3^1(k-1) + T_3^2(k-1) = T_3^0(k-2) + T_3^2(k-1) + T_3^2(k-2)$:
  This time, we cannot simply "plug in" as in earlier computations, but we have to do some further algebra. An argument shift and some re-arranging of the latter equation gives:
  $$T_3^2(k-1) - T_3^2(k-3) = T_3^0(k-3) + T_3^2(k-2) \tag{19}$$

Subtracting Eq. (18), rearranging and an argument shift yields:

$$T_3^2(k-1) = T_3^0(k+1) - T_3^0(k) + 0T_3^0(k-1) - 3T_3^0(k-2) - 2T_3^2(k-2).$$

Conversely, taking twice the sum of Eq. (18) and (19) leads to:

$$4T_3^2(k-1) = 2T_3^0(k) - 2T_3^0(k-1) + 0T_3^0(k-2) - 2T_3^0(k-3) + 2T_3^2(k-2).$$

Adding the last two equations together gives:

$$5T_3^2(k-1) = T_3^0(k+1) + T_3^0(k) - 2T_3^0(k-1) - 3T_3^0(k-2) - 2T_3^0(k-3).$$

The last equation can be plugged into five times Eq. (18), giving:
$0 = -5T_3^0(k) + 5T_3^0(k-1) + 10T_3^0(k-3) + (T_3^0(k+1) + T_3^0(k) - 2T_3^0(k-1) - 3T_3^0(k-2) - 2T_3^0(k-3)) + (T_3^0(k-1) + T_3^0(k-2) - 2T_3^0(k-3) - 3T_3^0(k-4) - 2T_3^0(k-5)) =$
$T_3^0(k+1) - 4T_3^0(k) + 4T_3^0(k-1) - 2T_3^0(k-2) + 6T_3^0(k-3) - 3T_3^0(k-4) - 2T_3^0(k-5).$
Hence,

$$T_3^0(k) \leq \boxed{2.1834^k}.$$

- $T_3^2(k) = T_3^0(k-1) + T_3^2(k-2)$: This case is much easier to handle; an argument shift leads to:

$$T_3^2(k-1) - T_3^2(k-3) = T_3^0(k-2), \tag{20}$$

and simply adding the shift with Eq. (18) gives:

$$2T_3^2(k-1) = T_3^0(k) - T_3^0(k-1) + T_3^0(k-2) - 2T_3^0(k-3).$$

This can be plugged in twice of Eq. (20), leading to
$0 = -2T_3^0(k-2) + 2T_3^2(k-1) - 2T_3^2(k-3) =$
$-2T_3^0(k-2) + (T_3^0(k) - T_3^0(k-1) + T_3^0(k-2) - 2T_3^0(k-3)) - (T_3^0(k-2) - T_3^0(k-3) + T_3^0(k-4) - 2T_3^0(k-5)) =$
$T_3^0(k) - T_3^0(k-1) - 2T_3^0(k-2) - T_3^0(k-3) - T_3^0(k-4) + 2T_3^0(k-5).$
Hence,

$$T_3^0(k) \leq 2.1523^k.$$

The first of these cases is obviously still slightly worse than our "target base" 2.1788 announced before. What can we do about it? The idea we are following now is to re-analyze the "star-case branching" for T_3^2, as well. More formally, our heuristic priorities would now—and finally—become:

H0 *Prefer small edges.* Let $\delta_{\min}^E = \min\{\delta(e) \mid e \in E\}$ and let $E_0 = \{e \in E \mid \delta(e) = \delta_{\min}^E\}$.

H0′ *Prefer small lonely edges* If $\delta_{\min}^E = 2$, then if there is an edge $e' \in E_0$ such that $\forall e \in E_0 \setminus \{e'\}(e \cap e' = \emptyset)$ then set $E_0 = \{e'\}$.

H'_{new} *Prefer sometimes center branch* If there is an $x \in \bigcup_{e \in E_0} e$

- with $\delta^2(x) > 2$ or with $\delta^2(x) = 2$ and $|E_0| = 2$,
- such that $\delta^3(y) = 1$ for all neighbors $y \in \bigcup_{e \in E_0} e$ of x,

then update $E_0 = \{x\}$.

$H1$ *Maximize Eq. (1)*, i.e., let $V_1 = \{x \in \bigcup_{e \in E_0} \mid \delta^3(x) - \delta^2(x) \text{ is maximum}\}$.

$H2$ Choose some $x \in V_1$ of *maximum degree*.

Analyzing T_3^2 in the star-case. This means that we are facing the following situation: we have edges $e_1 = \{x, y\}$ and $e_2 = \{x, z\}$. We discuss the following mutually exclusive possibilities:

1. $\delta(x) \geq 3$ and we branch at x: This gives according to the analysis contained in Lemma 10:
$$T_3^2(k) \leq \max\{T_3^1(k-1) + T_3^1(k-2), T_3^0(k-1) + T_3^2(k-2)\} \tag{21}$$
2. $\delta(x) \geq 3$ and we do not branch at x: The analysis contained in Lemma 10 shows:
$$T_3^2(k) \leq T_3^1(k-1) + T_3^2(k-1) \tag{22}$$
3. $\delta(x) = 2$ and $\delta(y) = \delta(z) = 2$. Then, according to H'_{new}, we will branch at x. Two sub-cases arise:
   - There is an edge $e = \{y, u, z\}$. If we then branch at x, the following scenario arises: If we put x into the cover, y and z will be dominated by u, so that u will go into the cover by the tiny edge rule, leading to a $T_3^0(k-2)$-branch. If x is not put into the cover, y and z will be, giving another $T_3^0(k-2)$-branch.
   - Otherwise, there are two different edges e_y and e_z (each of size three) such that $y \in e_y$ and $z \in e_z$. If x is put into the cover, then y and z won't be in the cover, leading to two new small edges $e_y \setminus \{y\}$ and $e_z \setminus \{z\}$. This is a $T_3^2(k-1)$-branch. If x is not going into the cover, then y and z will go. This gives a $T_3^0(k-2)$-branch.

   Summarizing, we can state:
$$T_3^2(k) \leq \max\{2T_3^0(k-2), T_3^0(k-2) + T_3^2(k-1)\} \tag{23}$$
4. $\delta(x) = 2$ and (w.l.o.g.) $\delta(y) > 2$. Then, the priority $H1$ applies and let us branch at y. If y is put into the hitting set, x will become of degree one and hence will be dominated by z, which is then put into the cover by the tiny edge rule. This is a $T_3^0(k-2)$-branch. If y is not put into the cover, x will go. Since $\delta^3(y) \geq 2$, at least two new small edges will be created, leading to a $T_3^2(k-1)$-branch. So, this case yields:
$$T_3^2(k) \leq T_3^0(k-2) + T_3^2(k-1),$$
   a branching behavior already observed in Eq. (23).

In short, the only relations that are "new" in comparison with Lemma 10 are that from Eq. (23). Now, let us first analyze these new relations for T_3^0, T_3^1 and T_3^2.

- Plugging $T_3^2(k) \leq 2T_3^0(k-2)$ directly into $T_3^0(k) \leq T_3^0(k-1) + T_3^0(k)$ gives:

$$T_3^0(k) \leq T_3^0(k-1) + 2T_3^0(k-2).$$

  This of course implies the not very stringent condition $T_3^0(k) \leq 2^k$.

- Into $T_3^2(k) \leq T_3^0(k-2) + T_3^2(k-1)$, we can plug $T_3^2(k) = T_3^0(k) - T_3^0(k-1)$, yielding
  $0 = T_3^0(k) - T_3^0(k-1) - T_3^0(k-2) - (T_3^0(k-1) - T_3^0(k-2)) =$
  $T_3^0(k) - 2T_3^0(k-1)$, which again gives $T_3^0(k) \leq 2^k$.

So, the only case where this new case analysis may affect the overall run time analysis is the special T_3^3-branching situation we discussed before making this detour, since that is the only case when we actually rely on that one of the new T_3^2 analyses applies.

Stars in T_3^3: the new special cases. When returning to the discussion of the star case, observe that we (again) can restrict our attention to assuming that "in other circumstances," the two branching scenarios for T_3^2 we always discussed apply, since otherwise the discussion of the "new cases" we just performed is applicable, showing a very nice estimate for $T_3^0(k)$. In fact, our earlier analysis shows that we may restrict ourselves on analyzing

$$T_3^2(k) = T_3^0(k-2) + T_3^2(k-1) + T_3^2(k-2).$$

The still critical case in Fig. 1(b) is if $\delta(y) = 3$ and $\delta(u), \delta(x), \delta(z) \geq 3$. Then, H'_{new} will not apply, so that $H1$ lets us branch at say x. If x does not go into the cover, then since $\delta(x) \geq 3$ at least two new small edges are created and y is put into the cover, giving a $T_3^2(k-1)$-branch. If x comes into the cover, then we enforce a T_3^2-situation as studied in the previous subsection. Overall, this leaves us with two cases for a final study:

1. If the enforced T_3^2-situation is solved with a branch yielding the recursion $T_3^2(k) \leq 2T_3^0(k-2)$, we are left with:

$$T_3^3(k) \leq 2T_3^0(k-3) + T_3^2(k-1).$$

   The equation $T_3^3(k) = T_3^0(k) - T_3^0(k-1)$ allows to rewrite:

$$T_3^0(k) - T_3^0(k-1) - 2T_3^0(k-3) = T_3^2(k-1).$$

   This has to be studied combined with

$$T_3^2(k) = T_3^0(k-2) + T_3^2(k-1) + T_3^2(k-2),$$

   giving
   $0 = (T_3^0(k+1) - T_3^0(k) - 2T_3^0(k-2)) - T_3^0(k-2) - (T_3^0(k) - T_3^0(k-1) -$

$2T_3^0(k-3)) - (T_3^0(k-1) - T_3^0(k-2) - 2T_3^0(k-4)) =$
$T_3^0(k+1) - 2T_3^0(k) + 0T_3^0(k-1) - 2T_3^0(k-2) + 2T_3^0(k-3) + 2T_3^0(k-4).$
This can be solved by $T_3^0(k) = 2.1372^k$.

2. If the enforced T_3^2-situation is solved with a branch yielding the recursion $T_3^2(k) \leq T_3^0(k-2) + T_3^2(k-1)$, we arrive at:

$$T_3^3(k) \leq T_3^0(k-3) + T_3^2(k-1) + T_3^2(k-2).$$

Hence, $T_3^3(k) = T_3^0(k) - T_3^0(k-1)$ allows to rewrite:

$$T_3^0(k) - T_3^0(k-1) - T_3^0(k-3) = T_3^2(k-1) + T_3^2(k-2).$$

This can be directly put into

$$T_3^2(k) = T_3^0(k-2) + T_3^2(k-1) + T_3^2(k-2),$$

giving

$$T_3^2(k) = T_3^0(k) - T_3^0(k-1) + T_3^0(k-2) - T_3^0(k-3).$$

Therefore,
$0 = T_3^0(k) - T_3^0(k-1) - T_3^0(k-3) - T_3^2(k-1) - T_3^2(k-2) =$
$T_3^0(k) - T_3^0(k-1) - T_3^0(k-3) - (T_3^0(k-1) - T_3^0(k-2) + T_3^0(k-3) - T_3^0(k-4)) - (T_3^0(k-2) - T_3^0(k-3) + T_3^0(k-4) - T_3^0(k-5)) =$
$T_3^0(k) - 2T_3^0(k-1) + 0T_3^0(k-2) - T_3^0(k-3) + 0T_3^0(k-4) + T_3^0(k-5).$
This amounts in $T_3^0(k) \leq 2.1676^k$.

Together with the kernelization as explained by Niedermeier and Rossmanith, we have now proved Theorem 11.

6.6 Some hints at the implementation

One of our arguments for advocating the top-down approach was the claim that it would simplify the implementation of parameterized algorithms. It should be clear that generically implementing `simple-HS-heuristic-binary` is a pretty straightforward task. However, what about implementing the "details," i.e., the reduction rules and the heuristic priorities? In actual fact, implementing them will go hand in hand. Since our main priority always is to branch on small edges, we would suggest keeping the list of edges sorted by increasing size. Since all operations are basically of local nature, maintaining this order during the run of the algorithm is comparatively straightforward. This ordering obviously supports the tiny edge rule and the edge selection for branching. Similarly, keeping track of the degrees and Eq. (1) helps select the vertex to branch at. Again by observing the local nature of the hypergraph modifications incurred by the algorithm, only a few vertices need to be checked in time for vertex domination, and a few edges for edge domination. A similar comment applies to the path and the triangle reduction rules.

ECCC ISSN 1433-8092
http://www.eccc.uni-trier.de/eccc
ftp://ftp.eccc.uni-trier.de/pub/eccc
ftpmail@ftp.eccc.uni-trier.de, subject 'help eccc'