Parameterized Algorithms for Hitting Set: the Weighted Case

Henning Fernau[1234]

[1] Univ.Trier, FB 4—Abteilung Informatik, 54286 Trier, Germany
[2] Univ.Hertfordshire, Comp. Sci., College Lane, Hatfield, Herts AL10 9AB, UK
[3] Univ.Tübingen, WSI für Informatik, Sand 13, 72076 Tübingen, Germany
fernau@informatik.uni-tuebingen.de
[4] Univ. Newcastle, Comp. Sci., University Drive, Callaghan, NSW 2308, AUS

Abstract. We are going to analyze simple search tree algorithms for Weighted d-Hitting Set. Although the algorithms are simple, their analysis is technically rather involved. However, this approach allows us to even improve on elsewhere published algorithm running time estimates for the more restricted case of (unweighted) d-Hitting Set.

1 Introduction

Our approach—in general. We exhibit how to systematically design and analyze search tree algorithms within the framework of parameterized algorithmics [2]. Here, we advocate a top-down approach as opposed to a rather bottom-up design, because the resulting algorithms tend to be simpler than via the opposite approach, and they sometimes pretty much resemble heuristic pruning techniques as used in branch-and-cut algorithms for solving hard problems. Moreover, this approach is quite modular in the sense that it produces algorithms whose *search tree backbone*, i.e., the branching pattern of the algorithm as such, is not affected by the optimization techniques reflected in what we will call *heuristic priorities* (according to which the branching is performed) and the employed *reduction rules*. This not only modularizes correctness proofs for such algorithms, but also favors rapid prototyping of implementations. We will exemplify this approach by developing and analyzing simple algorithms for Weighted d-Hitting Set (d-WHS) problems. No prior research on parameterized algorithms has been reported for these problems.

Problem statement. Weighted d-Hitting Set (d-WHS) can be viewed as a "weighted vertex cover problem" on hypergraphs. More formally, this problem can be stated as follows:

Given: A weighted hypergraph $G = (V, E, w)$ with *edge size* bounded by d, i.e., $\forall e \in E(|e| \leq d)$, and a *weight function* $w : V \to [1, \infty)$
Parameter: a non-negative integer k
Question: Is there a *(weighted) hitting set* C of total weight of at most k, i.e., $\exists C \subseteq V \forall e \in E(C \cap e \neq \emptyset)$ and $w(C) := \sum_{x \in C} w(x) \leq k$?

Why HITTING SET*?* HITTING SET problems show up in many places; e.g., Reiter's ground-breaking research on *model-based diagnosis* [10,14] relates the automatic diagnosis of systems to HITTING SET. The thrive for minimum hitting sets is in that context motivated by the parsimony principle in two ways: (a) the simplest diagnosis tends to find the actual cause, and (b) when the diagnosis implies exchanging (possibly) faulty components (as a consequence of a self-diagnosis of an autonomous system, e.g., in space), then a minimum hitting set might also be the cheapest repair solution; in that particular scenario, however, the weighted case seems to be even more interesting than the unweighted one. As a further application, in [8], connections between a two-tree drawing problem that is important in bioinformatics and 4-WHS are shown, where the weights reflect further natural restrictions from biological background knowledge. The algorithmics of this paper can be immediately transferred to both applications.

Previous work. For the unweighted case (which is a special case of the weighted setting if all weights are equal to one), there is one published paper presenting a search tree algorithm for UNWEIGHTED d-HITTING SET (d-HS), $d > 2$, from a parameterized perspective [11]. The exponential base of the running time estimate for these algorithms tends to $d-1$ with growing d, although in the simplest case $d = 3$, it is still relatively far off from that bound: that basis is $1 + \sqrt{2}$. By an intricate case analysis of a comparatively complicated algorithm, they were able to arrive at an $\mathcal{O}^*(2.270^k)$ algorithm for the (unweighted) 3-HS problem (i.e., all weights equal one). This was improved in [5] to about $\mathcal{O}^*(2.179^k)$ by using a similar methodology as explained here for the weighted case.

Notice that we are dealing with 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 (number of leaves) $\mathcal{O}(f(k))$, 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.

For the special case of 2-HS, likewise known as VERTEX COVER, in a kind of race (using more and more intricate case analysis) an $\mathcal{O}(1.285^k+kn)$-algorithm [1] has been obtained. For 2-WHS, likewise known as WEIGHTED VERTEX COVER, the best that was obtained is on $\mathcal{O}^*(1.396^k)$, see [12]. Our approach seems not to be suitable to tackle the case $d = 2$.

The results of this paper. As in the unweighted case [5], our analysis is based on the introduction of a second *auxiliary parameter* that allows us to account for "gains" obtained by using appropriate reduction rules and heuristic priorities. This technique can be useful in other areas of parameterized algorithms, as we believe. We get the following table for the bases c_d of an $\mathcal{O}^*(c_d^k)$ algorithm for d-WHS; the bases are better than those for the *unweighted case* published in [11]:

d	3	4	5	6	7	8	9	10	100
$c_d \leq$	2.2470	3.1479	4.1017	5.0640	6.0439	7.0320	8.0243	9.0191	99.0002

(1)

General notions and definitions. We introduce some terminology on hypergraphs as needed for HITTING SET. 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 $|e|$ of a hyperedge e is also called its *size*. The cardinality of the set of edges which contain the vertex v is called the *degree* of v, written $\delta(v)$.

2 Heuristics and reductions for WEIGHTED *d*-HITTING SET

2.1 A simple branching algorithm

Since each hyperedge must be covered and the weights are all at least one, there exists a trivial $\mathcal{O}^*(d^k)$-algorithm for d-WHS.

```
simple-WHS(G = (V, E, w), k, S):
  IF k > 0 AND G has some edges THEN
    choose some edge e; // to be refined
    S' = ∅; // solution to be constructed
    FOREACH x ∈ e DO // recursively branch
      G' = (V \ {x}, {e ∈ E | x ∉ e});
      S' = simple-WHS(G', k − w(x), S ∪ {x})
      IF S' ≠ failure THEN break
    return S'
  ELSIF E = ∅ THEN return S ELSE return failure
```

Obviously, the base of the exponential running time of this algorithm heavily depends on the necessary amount of branching. Observe that according to the problem specification, in a d-WHS instance, there might be edges of size *up to* d already in the very beginning. "Small edges" may also be introduced later during the run of the algorithm. A natural heuristic would first branch on small edges. We would therefore refine:

```
simple-WHS(G, k, S):
  IF k > 0  AND G has some edges THEN
    choose some edge e of smallest size;
  ... // as before
```

Can we make use of this "heuristic priority" in our analysis? We therefore now define reduction rules which we will always exhaustively apply at the beginning of each recursive call. Moreover, we switch towards a "binary branching" at vertices (instead of branching on edges), as can be seen in Alg. `WHS-ST` below.

2.2 Reduction rules

First reduction rule: vertex domination. The *vertex domination rule* that was used in [5,11] for the unweighted case is invalid in full generality in the weighted case, but has to be replaced by the following *weighted vertex domination rule*: If, for all edges e, $x \in e$ implies $y \in e$ and if $w(y) \leq w(x)$, then delete x.

This reduction rule implies the following one (reduction rule for degree-one-vertices): If $x, y \in e$ with $\delta(x) = 1$ and $w(y) \leq w(x)$, then remove x. The soundness of this rule is easily seen: the only reason for taking a vertex x into the hitting set, in a situation as described by the reduction rule, is that it might be *cheap*. Conserving expensive vertices makes no sense. This reduction rule immediately implies:

Lemma 1. *In a reduced instance, there is no edge with more than one vertex of degree one.*

The next lemma is again an easy consequence from the weighted vertex domination rule and is of particular importance when $d > 3$.

Lemma 2. *In a reduced instance, for any two edges e_1 and e_2, there is at most one $x \in e_1 \cap e_2$ with $\delta(x) = 2$.*

Other rules stated in [5] literally transfer to the weighted case:

Second reduction rule: edge domination. An edge e is dominated by another edge f if $f \subset e$. Then, we delete e, since covering f will automatically also cover e.

Third reduction rule: small edges. Delete all edges of size one and place the corresponding vertices into the hitting set.

The small edge rule, together with the vertex domination rule, proves the non-existence of *isolated edges* in the following precise sense:

Lemma 3. *In a reduced instance, there is no edge e such that all vertices $x \in e$ have degree one.*

Fourth reduction rule: edge cover rule. If G contains a component C that is of maximum vertex degree two, then resolve C in polynomial time.

This (last) rule is justified by the following lemma:

Lemma 4. *If G is a weighted hypergraph of maximum vertex degree of two, then a minimum weighted hitting set can be found in polynomial time.*

Proof. To G, there corresponds an edge-weighted graph G' whose vertices are the edges of G and whose vertex-adjacency relation is the edge-adjacency relation of G. Then, a minimum weighted hitting set of G corresponds to a minimum weighted edge cover of G' that can be computed in polynomial time. ∎

2.3 Branching rules and their analysis

The idea of making favorable branches first has also another bearing, this time on the way we are going to analyze the search tree algorithm, based on an *auxiliary parameter* ℓ. Let $T^{\ell}(k)$, $\ell \geq 0$ denote the size (more precisely, the number of leaves) of the search tree when assuming that at least ℓ edges in the given instance (with parameter k) have a size of (at most) $d-1$. The intuition

is that $T^3(k)$ would describe a situation which is "more like" $(d-1)$-WHS than $T^2(k)$. The underlying idea is that search trees with many small edges are smaller than search trees with only a few; hence:

$$\forall k : T^{\ell}(k) \geq T^{\ell+1}(k). \tag{2}$$

Regarding an upper bound on the size $T(k)$ of the search tree of the whole problem, we can equate $T(k) = T^0(k)$ by following the same intuition. Eq. (2) also shows that, upon analyzing a T^{ℓ}-situation, we can always assume that there are exactly ℓ edges that have a size of at most $d-1$, and these small edges do have a size of exactly $d-1$.

Our algorithm will make choices with the bias of what we will call *heuristic priorities*. They can be refined if necessary along the analysis of the algorithm. The simplest list to start with might contain a single rule that should be intuitively clear: Choose a vertex of highest degree within an edge of smallest size. We will update the list of priorities whenever necessary.

```
WHS-ST(G = (V, E, w), k, S):
  exhaustively apply reduction rules;
  IF k > 0 THEN
    IF E = ∅ THEN return S;
    choose some vertex x according to the heuristic priorities
    S' = ∅; // solution to be constructed
    E' = { e ∈ E | x ∉ e };
    S' = WHS-ST((V \ {x}, E'), k − w(x), S ∪ {x});
    IF S' ==failure THEN
       E'' = { e \ {x} | e ∈ E };
       S' = WHS-ST((V \ {x}, E''), k, S);
    return S'
  ELSIF G contains some edges or k < 0
    return failure
  ELSE // G contains no edges and k is zero
    return S
```

In the very beginning, given the instance (G, k), we call WHS-ST$(G, k, \emptyset)$. We assume that reduction rules may also change the parameter value k and the solution S. The algorithm is quite generic: the list of reduction rules may grow and we might also change the heuristic priorities. The simple *binary branching* structure of WHS-ST enables a straight-forward inductive proof of its correctness:

Theorem 1. *If the reduction rules are correct, then* WHS-ST$(G, k, \emptyset)$ *either returns a correct hitting set to the* d-WHS *instance* (G, k) *or it returns* `failure`*, if there is no solution of size at most* k*.*

3 A simple branching analysis

We will now undertake a *simple* analysis, only considering T^0, T^1 and (partially) T^2 and T^3.

Lemma 5. $T^0(k) \leq T^0(k-1) + T^3(k)$.

Proof. Whenever we select an edge of size d to branch on (according to the heuristic priorities), we can find an edge that contains a vertex x of degree three or larger due to the edge cover rule. One branch is that x is put into the hitting set. This reduces the admissible weight by at least one. If x is not put into the hitting set, then at least three new edges of size two are created. ∎

T^1-branching. 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. This already necessitates a refinement in the choice of heuristic priorities: within a smallest edge e of size $j < d$, we prefer branching at $x \in e$ that maximizes the number of incident edges of size $j+1$.

Lemma 6. $T^1(k) \leq \max\{T^0(k-1) + T^1(k-1) + T^2(k-1) + (d-4)T^3(k-1), T^0(k-1) + T^1(k-1) + (j-2)T^2(k-1) + (d^2-(2j+1)d+(j^2+j))T^0(k-2) : j = 2,3,\ldots,d-2\}$ *if* $d \geq 4$*; if* $T^0(k) \geq (d-1)^k$ *or if* $d = 3$*, this may be simplified:* $T^1(k) \leq T^0(k-1) + (d-2)T^1(k-1)$.

Proof. The instance G has an edge $e = \{x_1, x_2, \ldots, x_{d-1}\}$ of size $(d-1)$.
Case 1. If there is an edge f of size d such that $2 \leq j := |e \cap f| \leq d-2$ (this can only happen if $d \geq 4$), then we would first branch at the vertices in $e \cap f$; due to weighted vertex domination, at least one of the j branches that take one of the vertices of $e \cap f$ into the hitting set is an $T^1(k-1)$-branch and $j-2$ are even $T^2(k-1)$-branches (or better). If none of the vertices from $e \cap f$ goes into the hitting set, then, in order to cover e, there are $d-1-j$ many possibilities left, and in order to cover f, there are $d-j$ remaining possibilities. This explains the other $((d-j)-1)(d-j)$ many $T^0(k-2)$-branches. We can neglect these cases in our time analysis when assuming $T^0(k) \geq (d-1)^k$ as shown in the appendix.
Case 2. If the previous case does not occur, then for all edges $f \neq e$, $|e \cap f| \leq 1$. Assume that x_1 is the vertex of maximum degree in e, so that we branch at x_1. If $\delta(x_1) = 1$, we can deterministically resolve the case with the reduction rules (apply $d-1$ times the weighted vertex domination rule and then the small edge rule) and get *one* $T^0(k-1)$-branch. This is obviously better than the inequality claimed in the lemma. Therefore, we can now assume that $\delta(x_1) \geq 2$. If we take x_1 into the hitting set, then we get a $T^0(k-1)$-branch. If we do not take x_1 into the hitting set, we create one new edge e_1 of size $(d-1)$ and we get the edge $e' = e \setminus \{x_1\}$ of size $(d-2)$. In the next recursive call, e' is the edge of smallest size. There is no other edge of that size, since Case 1 did not apply. We therefore continue branching at the vertex (say x_2) of maximum degree in e'. Again, $\delta(x_2) = 1$ is better than the case we are going to pursue next. If $\delta(x_2) \geq 2$, then we again have two cases: either we take x_2 into the hitting set or not. If x_2 goes into the hitting set, then this is a $T^1(k-1)$-branch; namely, since Case 1 did not apply, $x_2 \notin e_1$, so that the small edge e_1 will be preserved. If x_2 does not go into the hitting set, then there will be a new edge e_2 of size $(d-1)$ ("new" due to edge domination). In the next recursive call, $e'' = e \setminus \{x_1, x_2\}$ is the edge of

smallest size. The argument continues and shows that branches of type $T^j(k-1)$ will show up, for $j = 2, 3, \ldots, d-2$. This shows the claim, taking into account that $T^j(k-1) \leq T^3(k-1)$ for $j \geq 3$ due to Eq. (2). ∎

Estimating branching numbers. By using the inequality $T^3(k) \leq T^2(k) \leq T^1(k)$, Lemmas 5 and 6 yield:

$$\begin{aligned} T^0(k) &\leq T^0(k-1) + T^1(k) \\ T^1(k) &\leq T^0(k-1) + (d-2)T^1(k-1) \end{aligned} \tag{3}$$

With c_d being the largest positive real root of the characteristic polynomial $x^2 - dx + d - 2$, i.e.,

$$c_d = \frac{d + \sqrt{d^2 - 4d + 8}}{2} = \frac{d + \sqrt{(d-2)^2 + 4}}{2} \geq d - 1 \tag{4}$$

we can see that by setting $T^0(k) = c_d^k$ and $T^1(k) = (c_d - 1)c_d^{k-1}$, the inequalities system (3) can be solved. The larger d, the closer c_d gets to $d - 1$. Hence:

d	3	4	5	6	10	100
$T(k) \leq$	2.62^k	3.42^k	4.31^k	5.24^k	9.13^k	99.0103^k

Obviously, this is worse than what Niedermeier and Rossmanith got in [11] for the (general) unweighted case (due to the lack of the vertex domination rule in full generality), but shows the same "limit behavior" (when d is large). Can we do better? Let us give a simple trial to incorporate T^2 and T^3 into the analysis in the special case of Weighted 3-Hitting Set.

4 Weighted 3-Hitting Set

We will use subscripts in the functions that describe the search tree sizes to indicate this special case. We branch according to the following heuristic priorities.
Let s be the size of the smallest edge in the instance $G = (V, E, w)$.
Let E_s be the collection of smallest size edges.
($P_3$1) Let the set of (first) branching candidates B be $\bigcup_{e \in E_s} e$.
($P_3$2) If e is a smallest edge that is disjoint with all other $e' \in E_s$, refine $B = e$.
($P_3$3) If no such isolated smallest edge exists, then update B to collect the vertices of maximum degree in the hypergraph $(\bigcup_{e \in E_s} e, E_s)$.
($P_3$4) Select $x \in B$ to be a vertex of maximum degree in G.
It is easy to check that the analyses of Lemmas 5 and 6 are still valid under these heuristic priorities.

Lemma 7. $T_3^2(k) \leq \max\{T_3^1(k-1) + T_3^2(k-1), T_3^0(k-1) + T_3^0(k-2)\}$

Proof. We consider first the situation that the two edges e_1 and e_2 of size two are disjoint (see ($P_3$2)). Then, basically the analysis of Lemma 6 applies, showing the claim. More precisely, we have $T_3^2(k) \leq T_3^1(k-1) + T_3^2(k-1)$.

Otherwise, $e_1 \cap e_2 \neq \emptyset$, i.e., $e_1 = \{x, y\}$ and $e_2 = \{x, z\}$. According to the heuristic priority ($P_3$3), we branch at x. If we take x into the hitting set, we get a $T_3^0(k-1)$-branch. Not taking x into the hitting set enforces y and z into the hitting set, which is a $T_3^0(k-2)$-branch. ∎

Lemma 8. $T_3^3(k) \leq \max \left\{ \begin{array}{l} T_3^1(k-1) + T_3^0(k-2), \\ T_3^0(k-1) + T_3^0(k-3), \\ T_3^2(k-1) + T_3^3(k-1) \end{array} \right\}$

Proof. If there is a edge e of size two that has non-empty intersection with any other edge of size two, due to ($P_3$2) we branch on e without destroying the at least two other edges of size two. The reasoning given in Lemma 6 therefore yields the upper bound $T_3^2(k-1) + T_3^3(k-1)$ in this case.

If the first case does not apply, the all edges of size two are connected. Let e_1, e_2, e_3 be three connected edges of size two. If $x \in e_1 \cap e_2 \cap e_3$ exists, then we branch at x due to ($P_3$3). This gives the (trivial) upper bound of $T_3^0(k-1) + T_3^0(k-3)$. Otherwise, we branch at some x contained in two small edges due to ($P_3$3); w.l.o.g.: $x \in e_1 \cap e_2$. Since $x \notin e_3$, the case that we take x into the hitting set is indeed a $T_3^1(k-1)$-branch. This explains the upper bound $T_3^1(k-1) + T_3^0(k-2)$. ∎

Theorem 2. Weighted 3-Hitting Set *can be solved in time* $\mathcal{O}^*(2.2470^k)$.

The algebra justifying this claim can be found in the appendix. We only mention that the exact solution of the inequalities system can be described by the largest positive root c_3 of the polynomial $x^3 - 2x^2 - x + 1$, which then gives $T_3^0(k) = c_3^k$, $T_3^1 = c_3^k/(c_3 - 1)$, $T_3^2(k) = c_3^k/(c_3 - 1)^2$, and $T_3^3(k) = c_3^{k-1}(c_3 - 1)$. This worst case is realized when all T^3-branches are according to the $T_3^1(k-1) + T_3^0(k-2)$-estimate. Improving on that particular case would not help too much, however, since the other extreme cases show also branching behaviors worse than 2.2^k. Observe that this also means that a search tree in the $T^3(k)$-case is only about half the size of a search tree in the $T^0(k)$-case.

5 Weighted d-Hitting Set with $d \geq 4$

How well do our considerations transfer to the more general case? We analyze possible T_d^2-branches in what follows. Since the obtained bases are quite satisfactory, we refrain from analyzing the T_d^3-branches. In our analysis, we apply the following heuristic priorities to a given (reduced) instance $G = (V, E, w)$:
Let s be the size of the smallest edge in the instance $G = (V, E, w)$.
Let E_s be the collection of smallest size edges.
(P1) Let the set of (first) branching candidates B be $\bigcup_{e \in E_s} e$.
(P2) Define $G_B = (B, E_s)$ and update B to be the set of vertices in G_B of maximum degree.
(P3) Choose a vertex $x \in B$ of maximum degree in G.

One can check that Lemmas 5 and 6 are still valid when assuming these priorities.

Analyzing T^2. We will distinguish several cases in what follows:

Lemma 9. *Let e_1 and e_2 be two edges of size $d-1$. If $e_1 \cap e_2 = \emptyset$, then we can estimate $T_d^2(k) \leq T_d^1(k-1) + (d-2)T_d^2(k-1)$.*

This can be basically inherited from Lemma 6 due to edge domination. As we will see, this is the second worst case branching. Being the simplest case, we give some details. As justified in the appendix, we solve the next set of equations:

$$\begin{aligned} T_d^0(k) &= T^0(k-1) + T^2(k) \\ T_d^1(k) &= T^0(k-1) + T^1(k-1) + (d-3)T^2(k-1) \\ T_d^2(k) &= T_d^1(k-1) + (d-2)T_d^2(k-1) \end{aligned} \tag{5}$$

This yields, after some algebra:

$$0 = T_d^0(k+1) - dT_d^0(k) + (d-1)T_d^0(k-1) - T_d^0(k-2). \tag{6}$$

Theorem 3. *Let c_d denote the largest positive real root of the polynomial $x^3 - dx^2 + (d-1)x - 1$. Then $T_d^0(k) = c_d^k$, $T_d^1(k) = \alpha_{d,1}c_d^k$ with $\alpha_{d,1} = (c_d - d + 2)(c_d - 1)/c_d$ and $T_d^2(k) = \alpha_{d,2}c_d^k$ with $\alpha_{d,2} = (c_d - 1)/c_d$ solve the system (5).*

The following table lists some of the exponential bases c_d for (5):

d	3	4	5	6	7	8	9	10	100
$c_d \leq$	2.3248	3.1479	4.0780	5.0490	6.0330	7.0237	8.0178	9.0139	99.0002

(7)

Lemma 10. *Let e_1 and e_2 be two edges of size $d-1$. If $|e_1 \cap e_2| = j \in \{1, 2, \ldots, d-2\}$, then we can estimate*

$$T_d^2(k) \leq T_d^0(k-1) + T_d^1(k-1) + (j-2)T_d^2(k-1) + (d-1-j)^2 T_d^0(k-2),$$

thereby assuming that $T_d^0(k) \geq (d-1)^k$, i.e., $c_d \geq d-1$.

Proof. The priorities $(P1)$ and $(P2)$ let us branch at a vertex $x \in e_1 \cap e_2$. If $j > 1$, the weighted vertex domination rule moreover guarantees that there is a vertex of degree at least three in $e_1 \cap e_2$, and $(P3)$ will select one such vertex x for branching. Hence, when x is not taken into the hitting set, then we gain at least one edge of size $d-1$ if $j > 1$ due to vertex domination, see Lemma 2, since we will continue selecting vertices within $e_1 \cap e_2$ according to $(P2)$. The case that edges that intersect with $e_1 \cap e_2$ might contain more than one vertex in this intersection turns out not to be the worst case (assuming $(d-1)^k$ as a lower bound of our approach) along the lines of Lemma 6. If $e_1 \cap e_2$ is "exhausted", then in the case that we take none of the vertices from $e_1 \cap e_2$ into the hitting set, we are left with two very small edges $e_1' = e_1 \setminus e_2$ and $e_2' = e_2 \setminus e_1$. $P1$ lets us continue branching at say e_1'. Having selected $x \in e_1'$ to go into the hitting set, e_2' will be the smallest edge (of size $(d-1-j)$), and hence $P1$ continues to branch on e_2' in the next recursion step. This explains that we get (very grossly estimated) $(d-1-j)^2$ many $T_d^0(k-2)$-branches. ∎

In order to prove Theorem 4, the following technical lemma is important:

Lemma 11. *If $j > 1$ and $d > 3$, then*

$$T_d^1(k-1) + (d-2)T_d^2(k-1) \\ \geq T_d^0(k-1) + T_d^1(k-1) + (j-2)T_d^2(k-1) + (d-1-j)^2 T_d^0(k-2)$$

for $T_d^0(k) = c^k$ and $T_d^2(k) = c^k - c^{k-1}$ with $d-1 \leq c$, independent of T_d^1.

We need a somewhat stronger result (compared to Lemma 10) in the case $j = 1$ that describes our worst case (for $d > 4$):

Lemma 12. *Let e_1 and e_2 be two edges of size $d-1$. If $|e_1 \cap e_2| = 1$, then we can estimate*

$$T_d^2(k) \leq T_d^0(k-1) + (d-3)T_d^1(k-2) + [(d-2)(d-3)+1]T_d^0(k-2).$$

Moreover,

$$T_d^1(k-1)+(d-2)T_d^2(k-1) \geq T_d^0(k-1)+(d-3)T_d^1(k-2)+[(d-2)(d-3)+1]T_d^0(k-2)$$

for T_d^ℓ as defined in Theorem 4 below.

Proof. We only explain the branching in what follows (for the algebra, see the appendix). Assume that $\{x\} = e_1 \cap e_2$. x is selected for branching according to $(P1)$. If x does not go into the hitting set, then we may continue branching on e_1. The claim is that, for any $y \in e_1 \setminus \{x\}$ (with one possible exception, if $\delta(y) = 1$ for some $y \in e_1$; but due to Lemma 1, there is at most one vertex of degree one in e_1 and $(P3)$ avoids branching at that vertex), there is an edge $e_y \neq e_1$ with $y \in e_y$ such that there is a vertex $z_y \in e_2 \setminus e_1$ with $z_y \notin e_y$. For, if $(e_2 \setminus \{x\}) \subseteq e_y$, then the edge domination rule would have triggered. The branch that takes y and z_y into the hitting set is a $T_d^1(k-2)$-branch (possibly better). ∎

Theorem 4. d-WHS *can be solved in time $\mathcal{O}^*(c_d^k)$, where c_d is the largest positive root of the characteristic polynomial $x^4-3x^3-(d^2-5d+5)x^2+x+(d^2-6d+9)$. Some values of c_d are listed below:*

d	4	5	6	7	8	9	10	100
$c_d \leq$	3.1845	4.1017	5.0640	6.0439	7.0320	8.0243	9.0191	99.0002

(8)

With a more sophisticated analysis (and again varied heuristic priorities), we could let Theorem 3 describe the worst case for 4-WHS.

Corollary 1. 4-WHS *can be solved in time $\mathcal{O}^*(3.1479^k)$.*

Is it worthwhile trying to further improve on the exponential bases as derived in this section? In principle, yes of course; however, one would need a different approach for substantial improvements: (a) the second-worst case is only slightly better than the worst case that we analyzed, and (b) with growing d, the lower bound $(d-1)$ assumed in (some) estimates is already quite well approximated.

6 Conclusions

We are currently developing and analyzing a new, top-down methodology for parameterized search tree algorithms. Up to now, we have applied this methodology to d-Hitting Set [5], biplanarization problems [7] (thereby improving on the constants derived in [3]), linear arrangement problems (in the long version of [6]) and to Weighted d-Hitting Set (this paper). 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 upon *binary branching*. In the case of (Weighted) Hitting Set, the number of edges of small size is such an auxiliary parameter. Our results show that this methodology is a quite powerful tool of algorithm analysis. For example, while the gap between the running times of the (very sophisticated) best search tree algorithms for Weighted Vertex Cover and for Vertex Cover [1,12] do differ significantly (both algorithms being approximately of the same complexity), this paper shows that with our analysis method of a comparatively simple algorithm for 3-WHS, we can even (slightly) improve on the previous analysis of a much more sophisticated algorithm for Unweighted 3-HS [11].

It may be interesting to compare the way the analysis of the recurrences guided by the auxiliary parameter is undertaken in this paper with the analysis method of Wahlström [15] or with Eppstein's quasiconvex method [4]. It would be also interesting to see this approach applied to other, different problems with accordingly different auxiliary parameters.

More generally speaking, there seems to be a recent thrive in Exact Algorithmics towards "simple" algorithms. The recent Minimum Dominating Set algorithm of Fomin, Grandoni and Kratsch is only one more example (see [9]) that incidentally also uses a (special) Hitting Set algorithm. This direction of research certainly brings practical and theoretical research on attacking hard problems closer together, since one could also envisage a kind of interplay between algorithm analysis and algorithm testing in the near future. Can an appropriate analysis then "explain" certain observed phenomena of the implementation? The modular decomposition of such an algorithm into the actual recursive "search tree backbone" and the reduction rules and (in particular) the heuristic priorities also opens up a whole area of experimental algorithmics: under which circumstances (or, in a more theoretical formulation: for which classes of hypergraphs) is a certain set of rules the most successful? Can this be proved? Due to the simple overall structure of the algorithms, also an analysis of expected running times (possibly adding coin tossing into the heuristic priorities) might be possible.

Incidentally, improvements in parameterized algorithms for d-Hitting Set also entail improvements in exact algorithms for Minimum d-Hitting Set, measured in terms of number of vertices: in the case of 3-Hitting Set, the use of the algorithm exhibited in [5] improves Wahlström's algorithm [15] from $\mathcal{O}^*(1.6538^n)$ down to $\mathcal{O}^*(1.6483^n)$ (personal communication by Wahlström). The results of this paper will immediately entail new running time bounds for exact algorithms for Minimum Weighted Hitting Set. For example, along the lines

sketched by Raman, Saurabh and Sikdar, in [13], we get an exact algorithm for MINIMUM WEIGHTED 4-HITTING SET that runs in time $\mathcal{O}^*(1.97^n)$, using our parameterized WEIGHTED 4-HITTING SET algorithm.

References

1. J. Chen, I. A. Kanj, and W. Jia. Vertex cover: further observations and further improvements. *Journal of Algorithms*, 41:280–301, 2001.
2. R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer, 1999.
3. V. Dujmović, M. R. Fellows, M. Hallett, M. Kitching, G. Liotta, C. McCartin, N. Nishimura, P. Ragde, F. A. Rosamond, M. Suderman, S. Whitesides, and D. R. Wood. A fixed-parameter approach to two-layer planarization. In P. Mutzel, M. Jünger, and S. Leipert, editors, *9th International Symp. on Graph Drawing GD'01*, volume 2265 of *LNCS*, pages 1–15. Springer, 2002.
4. D. Eppstein. Quasiconvex analysis of backtracking algorithms. In *Proc. 15th Symp. Discrete Algorithms SODA*, pages 781–790. ACM and SIAM, January 2004.
5. H. Fernau. A top-down approach to search-trees: Improved algorithmics for 3-Hitting Set. Technical Report TR04-073, Electronic Colloquium on Computational Complexity ECCC, 2004.
6. H. Fernau. Parameterized algorithmics for linear arrangement problems. In U. Faigle, editor, *CTW 2005: Workshop on Graphs and Combinatorial Optimization*, pages 27–31. University of Cologne, Germany, 2005. Long version submitted to a special issue of *Discrete Applied Mathematics*.
7. H. Fernau. Two-layer planarization: Improving on parameterized algorithmics. *Journal of Graph Algorithms and Applications*, 9:205–238, 2005.
8. H. Fernau, M. Kaufmann, and M. Poths. Comparing trees via crossing minimization. In R. Ramanujam and Sandeep Sen (editors): *Foundations of Software Technology and Theoretical Computer Science FSTTCS 2005*, vol. 3821 of *LNCS*, pp. 457–469. Berlin: Springer, 2005.
9. F. V. Fomin, F. Grandoni, and D. Kratsch. Measure and conquer: domination – a case study. In L. Caires, G. F. Italiano, L. Monteiro, C. Palamidessi, and M. Yung, editors, *Automata, Languages and Programming, 32nd International Colloquium, ICALP*, volume 3580 of *LNCS*, pages 191–203. Springer, 2005.
10. J. de Kleer, A. K. Mackworth, and R. Reiter. Characterizing diagnoses and systems. *Artificial Intelligence*, 56:197–222, 1992.
11. R. Niedermeier and P. Rossmanith. An efficient fixed-parameter algorithm for 3-Hitting Set. *Journal of Discrete Algorithms*, 1:89–102, 2003.
12. R. Niedermeier and P. Rossmanith. On efficient fixed parameter algorithms for weighted vertex cover. *Journal of Algorithms*, 47:63–77, 2003.
13. V. Raman, S. Saurabh, and S. Sikdar. Improved exact exponential algorithms for vertex bipartization and other problems. In M. Coppo et al., editors, *Italian Conference on Theoretical Computer Science ICTCS*, volume 3701 of *LNCS*, pages 375–389. Springer, 2005.
14. R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32:57–95, 1987.
15. M. Wahlström. Exact algorithms for finding minimum transversals in rank-3 hypergraphs. *Journal of Algorithms*, 51:107–121, 2004.

7 Appendix: Proof of Theorem 1

Let us assume that the reduction rules are correct (this has been argued for in the main text). Since the heuristic priorities only resolve nondeterminism in certain choice situations, we only have to prove that the "search tree backbone" is correct.

Proof. Formally, the proof is done by induction on the number of vertices of the hypergraph. If the weighted input hypergraph $G = (V, E, w)$ has no vertices (or one vertex), then it has no hyperedges, so that (correctly) $\emptyset$ is returned as a solution in the case that $k \geq 0$. The case $k < 0$ is no valid instance according to our problem definition, and therefore `failure` is correctly returned.

Consider now an instance $(G = (V, E, w), k)$ where $|V| = n + 1$. 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. If $k < 0$, `failure` is returned in accordance with our problem definition (the budget has been overspent). Similarly, if $k = 0$ and G still contains some edges, `failure` is returned; there is no way of covering the remaining edges for free (since all weights are at least one). Otherwise, if $k = 0$ and G has no more edges, we can (correctly) return the solution found so far.

If we found that $k > 0$, then we enter the 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 recursively constructed hitting set S 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 `failure`, since if a solution is already found, then there is no need to start the second branch of the recursion. Since the described actions are obviously correct, the correctness of the algorithm follows by induction. ∎

8 Appendix: More justifying computations for Sec. 3

To fully justify our computations, we have to show two things:

1. The root c_d as determined in Sec. 3 determines $T^0(k) = c_d^k$ and $T^1(k) = (c_d - 1)c_d^{k-1}$ such that it solves the following system of equalities:

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

In fact,

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

To see the validity of the second equation, multiply by c_d^{2-k} to get

$$(c_d - 1)c_d = c_d^2 - c_d = c_d + (d-2)(c_d - 1) = (d-1)c_d - (d-2)$$

This is true since c_d is a root of the polynomial $x^2 - dx + (d-2)$.

2. We distinguish two cases to prove that $T^0(k-1) + T^1(k-1) + T^2(k-1) + (d-4)T^3(k-1)$ is always bigger than $T^0(k-1) + T^1(k-1) + (j-2)T^2(k-1) + (d^2 - (2j+1)d + (j^2+j))T^0(k-2)$ for all $j = 2, 3, \ldots, d-2$ if $d \geq 4$: (a) we stop the analysis at T^1; (b) we stop the analysis at T^2.
In case (a), we have to show (using Eq. (2)):
for all $j = 2, \ldots, d-2$: $T^0(k-1) + (d-2)T^1(k-1) \geq$

$$T^0(k-1) + (j-1)T^1(k-1) + (d^2 - (2j+1)d + (j^2+j))T^0(k-2),$$

when $T^0(k) = c_d^k$ and $T^1(k) = (c_d - 1)c_d^{k-1}$. Canceling the term $T^0(k-1)$, we have to show that

$$(d^2 - (2j+1)d + (j^2+j))c_d^{k-2} + [j - 1 - (d-2)](c_d - 1)c_d^{k-2} \leq 0 \qquad (9)$$

Multiply by c_d^{2-k} and consider the following chain:

$$\begin{aligned} &(d^2 - (2j+1)d + (j^2+j)) + (j+1-d)(c_d - 1) \\ &\leq (d^2 - (2j+1)d + (j^2+j)) + (j-d+1)(d-2) \\ &= (d^2 - (2j+1)d + (j^2+j) + jd - d^2 + d - 2j + 2d - 2 \\ &= (2-j)d + j^2 - j - 2 \\ &= (j-2)(-d + (j+2)) - j + 2 \\ &\leq 0 \end{aligned}$$

where, for the inequalities, we used that $c_d \geq d - 1$ and that $(j - d + 2) \leq 0$ and $j \geq 2$. This therefore also shows the additional assertion of Lemma 6.
In case (b), we have to show (using Eq. (2)): for all $j = 2, \ldots, d-2$:
$T^0(k-1) + T^1(k-1) + (d-3)T^2(k-1)$ is always bigger than

$$T^0(k-1) + T^1(k-1) + (j-2)T^2(k-1) + (d^2 - (2j+1)d + (j^2+j))T^0(k-2),$$

when $T^0(k) = c_d^k$ and $T^2(k) = (c_d - 1)c_d^{k-1}$. Canceling the term $T^0(k-1) + T^1(k-1)$, we have to show that

$$(d^2 - (2j+1)d + (j^2+j))c_d^{k-2} + [j - 2 - (d-3)](c_d - 1)c_d^{k-2} \leq 0$$

As can be seen, this is exactly the same expression as Eq. (9) above, so that the claim is true. In fact, more generally, this is true for any analysis up to say T^ρ, since then a sum of ρ terms $T^0(k-1),\ldots,T^{\rho-1}(k-1)$ in both estimates would cancel out, and then the same algebraic argument applies again to show the claim.

9 Appendix: Branching behavior for Weighted 3-Hitting Set

In the following, we suppress the subscript 3, since we are only dealing with this case.

Let us first show some algebra in case that we only analyze up to $T^2(k)$, i.e., if we put $T^3(k) = T^2(k)$. What branching behavior do we observe in either case in Lemma 7 ?

1. If $T^2(k) \leq T^1(k-1) + T^2(k-1)$, we get (by Lemma 5),

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

   which gives as a recurrence

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

   By Lemma 6,

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

   Therefore,

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

   This is resolved by $T^0(k) \leq 2.3248^k$.
2. $T^2(k) \leq T^0(k-1) + T^0(k-2)$ gives immediately the characteristic polynomial $x^2 - 2x - 1$ with largest positive real root $x = 1 + \sqrt{2} \leq 2.4143$; this is hence the worst case here.

It might be surprising at first glance that we treated the weak inequalities as if they were equalities. This is based on experience: taking this approach we were always able to come up with a solution to the envisaged system of inequalities. However, these computations should be justified by further analysis. Since this is only an (encouraging) intermediate result, we refrain from giving such validation here; we will however give it in the general case.

For WEIGHTED 3-HITTING SET, we derived the following recurrences:

$$
\begin{aligned}
T^0(k) &\leq T^0(k-1) + T^3(k) \\
T^1(k) &\leq T^0(k-1) + T^1(k-1) \\
T^2(k) &\leq \max\{T^1(k-1) + T^2(k-1), T^0(k-1) + T^0(k-2)\} \\
T^3(k) &\leq \max \left\{ \begin{array}{l} T^1(k-1) + T^0(k-2), \\ T^0(k-1) + T^0(k-3), \\ T^2(k-1) + T^3(k-1) \end{array} \right\}
\end{aligned}
$$

How do we arrive at possible solutions? Firstly, we try to solve "extreme cases" that are obtained by treating the weak inequalities as if they were equalities and by discussing all possible combinations as described by the maximum operator. In our case, this would in principle result in six systems of equations. However, since T^2 shows up only in one place in a right-hand side of a $T^3(k) \leq$-inequality, we only get four cases.

It is noteworthy to see that, when assuming $T^0(k) = c^k$, (for all situations) the first equation gives

$$T^3(k) = c^{k-1}(c-1)$$

and the second equation gives

$$T^1(k) = c^k/(c-1).$$

Notice that finally we are looking for an overall solution for all T^j, i.e., $T^j(k) = \alpha_j c_3^k$ with c_3 and the α_j still to be determined. Our considerations so far entail:

$$\alpha_0 = 1 \geq \alpha_1 = 1/(c_3 - 1) \geq \alpha_2 \geq \alpha_3 = (c_3 - 1)/c_3$$

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

Obviously, the function $T^2(k)$ does not come into play if we are primarily interested in looking for solutions for $T^0(k)$ in this case. Plugging in the expressions for $T^0(k)$, $T^1(k)$ and $T^3(k)$ in the last equation yields:

$$c^{k-1}(c-1) = c^{k-1}/(c-1) + c^{k-2}.$$

After multiplication with $(c-1)c^{2-k}$ and some reordering, this becomes the characteristic polynomial:

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

Its largest positive real root can be bounded by 2.2470 from above. As can be seen, this is the claimed worst case.

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

The same solution strategy provides:

$$c^{k-1}(c-1) = c^{k-1} + c^{k-3}.$$

Multiplication with c^{3-k} and some reordering yields:

$$0 = c^3 - 2c^2 - 1;$$

the largest positive real root can be bounded by 2.2056 from above.

In the following two cases, we have to distinguish the two different upper bounds for T^2.

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

From $T^1(k) = c^k/(c-1)$, we can deduce from the third equation:

$$T^2(k) = c^k/((c-1)^2).$$

Therefore, the last equation gives:

$$c^k(c-1) = c^{k-1}/((c-1)^2) + c^{k-1}(c-1).$$

Multiplication with $c^{1-k}(c-1)^2$ results in:

$$0 = (c-1)^4 - c = c^4 - 4c^3 + 6c^2 - 5c + 1,$$

whose largest positive real root can be estimated by 2.2208.

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

A direct plug-in yields:

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

This gives again

$$0 = c^3 - 2c^2 - 1;$$

the largest positive real root can be bounded by 2.2056 from above.

So, the worst scenario gives the characteristic polynomial $c^3 - 2c^2 - c + 1$, whose largest positive real root c_3 can be bounded from above by 2.246980.

We haven't yet determined α_2. From $T^2(k) \leq T^1(k-1)+T^2(k-1)$, we would get $\alpha_2 = 1/(c_3-1)^2$, and from $T^2(k) \leq T^0(k-1) + T^0(k-2)$, we arrive at $\alpha_2 = (c_3+1)/c_3^2$. However, $c_3^2 = (c_3+1)(c_3-1)^2 = (c_3^2-1)(c_3-1) = c_3^3-c_3^2-c_3+1$ is true, since c_3 is a root of the characteristic polynomial, so that both (seemingly different) values of α_2 are in fact equal. In other words, we found:
$\alpha_0 = 1 \geq \alpha_1 = 1/(c_3-1) \approx 0.80 \geq$
$\alpha_2 = (c_3+1)/c_3 \approx 0.64 \geq \alpha_3 = (c_3-1)/c_3 \approx 0.55$

After having obtained these numbers, we should verify that indeed $T^0(k) = c_3^k$, $T^1 = c_3^k/(c_3-1)$, $T^2(k) = c_3^{k-2}(c_3+1)$, and $T^3(k) = c_3^{k-1}(c_3-1)$ satisfies the whole system of inequalities. This amounts in showing that the "extreme case" we found is indeed maximizing all right-hand side maxima functions. In fact, our reasoning with determining α_2 in the preceding paragraph already shows that $T^1(k-1) + T^2(k-1) = T^0(k-1) + T^0(k-2)$ for our functions. We still have to deal with the $T^3(k) \leq$-inequalities.

1. To show: $T^1(k-1) + T^0(k-2) \geq T^0(k-1) + T^0(k-3)$. Substituting the functions we derived and multiplying with $c^{3-k}(c-1)$ leaves us to show:

$$c^2 + (c^2 - c) \geq (c^3 - c^2) + (c-1) \Leftrightarrow 0 \geq c^3 - 3c^2 + 2c - 1$$

   which is true for $c = c_3$.
2. To show: $T^1(k-1) + T^0(k-2) \geq T^2(k-1) + T^3(k-1)$. Substituting the functions we derived and dividing by c^{k-2} leaves us to show:

$$\frac{c}{c-1} + 1 \geq \frac{c^2+c}{c^2} + \frac{c^2-c}{c} = \frac{1+c^2}{c}$$

   This in turn means we have to verify (again) for $c = c_3$:

$$0 \geq c^3 - 3c^2 + 2c - 1.$$

10 Appendix: The algebra for the general case

10.1 The algebra for Lemma 9

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

Now, replace $T_d^2(k)$ by $T_d^0(k) - T_d^0(k-1)$ (and similar). Hence,

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

as claimed.

10.2 The algebra for Lemma 11

We are going to prove:

$$T_d^1(k-1)+(d-2)T_d^2(k-1) \geq T_d^0(k-1)+T_d^1(k-1)+(j-2)T_d^2(k-1)+(d-1-j)^2T_d^0(k-2)$$

with the settings as described in the Lemma. Obviously, the T_d^1-term cancels out, so we are left to show an upper bound on

$$-(d-2)T_d^2(k-1) + T_d^0(k-1) + (j-2)T_d^2(k-1) + (d-1-j)^2T_d^0(k-2).$$

By the assumptions of the lemma, this means we have to upperbound:

$$-(d-2)(c^{k-1}-c^{k-2})+c^{k-1}+(j-2)(c^{k-1}-c^{k-2})+(d^2-2(j+1)d+(j+1)^2)c^{k-2}.$$

After multiplication with c_d^{2-k}, we get the following chain:

$$\begin{aligned} &(-d+2+j-2)(c-1) + c + (d^2 - 2(j+1)d + (j+1)^2) \\ &= (j+1-d)c + d^2 + (-2j-1)d + j^2 + j + 1 \\ &\leq \underbrace{(-d+j+1)(d-1)}_{=-d^2+jd+2d-j-1} + d^2 + (-2j-1)d + j^2 + j + 1 \\ &= \overbrace{(j-2)(-d+j+2)}^{=-jd+j^2+2d-4} - d + 4 \\ &\leq 0. \end{aligned}$$

The first estimate is true, since $d-1 \leq c$ and $j \leq d-2$, and the last one from $d \geq 4$ and from $j > 1$ together with $j \leq d-2$.

10.3 Detailed algebra following Lemma 12 (for Thm. 4)

We are dealing with the following situation as an extreme case:

$$\begin{aligned} T_d^0(k) &= T_d^0(k-1) + T_d^2(k) \\ T_d^1(k) &= T_d^0(k-1) + T_d^1(k-1) + (d-3)T_d^2(k-1) \\ T_d^2(k) &= T_d^0(k-1) + (d-3)T_d^1(k-2) + (d^2-5d+7)T_d^0(k-2) \end{aligned} \tag{10}$$

We can multiply the second equation with $(d-3)$ and do an argument shift to obtain:

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

Subtraction of the third equation yields: $(d-3)T_d^1(k-1) - T_d^2(k) =$

$$\begin{aligned}
&= (d-3)T_d^0(k-2) + \underline{(d-3)T_d^1(k-2)} + (d-3)^2 T_d^2(k-2) \\
&\quad -(T_d^0(k-1) + \underline{(d-3)T_d^1(k-2)} + (d^2-5d+7)T_d^0(k-2)) \\
&= -T_d^0(k-1) + [(d-3) - (d^2-5d+7)]T_d^0(k-2) + (d-3)^2 T_d^2(k-2) \\
&= -T_d^0(k-1) + [(-d^2+6d-10) + (d^2-6d+9)]T_d^0(k-2) - (d^2-6d+9)T_d^0(k-3) \\
&= -T_d^0(k-1) - T_d^0(k-2) - (d^2-6d+9)T_d^0(k-3)
\end{aligned}$$

Using the first equation, this yields the following expression for $(d-3)T_d^1(k-1)$:

$$T_d^0(k) - 2T_d^0(k-1) - T_d^0(k-2) - (d^2-6d+9)T_d^0(k-3).$$

Plugging this into the third equation gives the following vanishing expression:

$$\begin{aligned}
&-T_d^0(k) + 2T_d^0(k-1) + (d^2-5d+7)T_d^0(k-2) \\
&+[T_d^0(k-1) - 2T_d^0(k-2) - T_d^0(k-3) - (d^2-6d+9)T_d^0(k-4)] \\
&= -T_d^0(k) + 3T_d^0(k-1) + (d^2-5d+5)T_d^0(k-2) \\
&\quad -T_d^0(k-3) - (d^2-6d+9)T_d^0(k-4).
\end{aligned}$$

Hence, we get the following table as estimates for c_d (which can be exactly described as the largest positive root of the characteristic polynomial $x^4 - 3x^3 - (d^2-5d+5)x^2 + x + (d^2-6d+9)$):

d	4	5	6	7	8	9	10	100
$c_d \leq$	3.1845	4.1017	5.0640	6.0439	7.0320	8.0243	9.0191	99.0002

Is is further noteworthy to see that the expressions

$$T_d^1(k) = \frac{c_d d - d - 2c_d + 3}{c_d - 1} c_d^{k-1} \tag{11}$$

can be derived from the (only) equation with left-hand side T_d^1, based on the approach that sets $T_d^0(k) = c_d^k$ and hence (with the first equation) $T_d^2(k) = (c_d - 1)c_d^{k-1}$. These expressions are correct independently of the concrete evaluation of c_d which depends on the last equation (the one for T_d^2 that involves the maximum operator). In other words, its validity only depends on the fact that we are now considering our set-up for auxiliary parameter values 0, 1, and 2. Namely,

$$\begin{aligned}
T_d^1(k) &= T_d^0(k-1) + T_d^1(k-1) + (d-3)T_d^2(k-1) \\
&= c_d^{k-1} + \frac{c_d d - d - 2c_d + 3}{c_d - 1} c_d^{k-2} + (d-3)(c_d - 1)c_d^{k-2} \\
&= \frac{c_d^2 - c_d + c_d d - d - 2c_d + 3 + (d-3)(c_d^2 - 2c_d + 1)}{c_d - 1} c_d^{k-2} \\
&= \frac{c_d^2 - c_d + c_d d - d - 2c_d + 3 + dc_d^2 - 2c_d d + d - 3c_d^2 + 6c_d - 3}{c_d - 1} c_d^{k-2} \\
&= \frac{c_d d - d - 2c_d + 3}{c_d - 1} c_d^{k-1}
\end{aligned}$$

We have now to prove that

$$\begin{aligned}&T_d^1(k-1)+(d-2)T_d^2(k-1)\\&\leq T_d^0(k-1)+(d-3)T_d^1(k-2)+(d^2-5d+7)T_d^0(k-2)\end{aligned}$$

Having shown this, we know that the settings derived in the preceding paragraphs for T_d^0, T_d^1 and for T_d^2 are satisfying all $T_d^2(k)$ $\leq$-inequalities, because Lemma 11 is also valid in this situation.

We therefore have to derive zero as an upper bound for

$$T_d^1(k-1)+(d-2)T_d^2(k-1)-T_d^0(k-1)-(d-3)T_d^1(k-2)-(d^2-5d+7)T_d^0(k-2).$$

Plugging in the solutions c_d we found for the functions gives the next expression:

$$\begin{aligned}&\frac{c_d d-d-2c_d+3}{c_d-1}c_d^{k-2}+(d-2)(c_d-1)c_d^{k-2}-c_d^{k-1}\\&-\frac{(d-3)(c_d d-d-2c_d+3)}{c_d-1}c_d^{k-3}-(d^2-5d+7)c_d^{k-2}\end{aligned}$$

This expression is upperbounded by zero iff the following expression is upperbounded by zero (obtained by multiplication with $(c_d-1)c_d^{3-k}$):

$$\begin{aligned}&c_d^2 d-c_d d-2c_d^2+3c_d+(d-2)(c_d-1)^2c_d-c_d^2\\&-(d-3)(c_d d-d-2c_d+3)-(d^2-5d+7)(c_d-1)c_d\\=\;&c_d^2 d-c_d d-2c_d^2+3c_d+(d-2)(c_d^2-2c_d+1)c_d-c_d^2\\&-(d-3)(c_d d-d-2c_d+3)-(d^2-5d+7)(c_d-1)c_d\\=\;&c_d^2 d-c_d d-3c_d^2+3c_d\\&+dc_d^3-2dc_d^2+d-2c_d^3+4c_d^2-2c_d\\&-c_d d^2+d^2+2c_d d-3d+3c_d d-3d-6c_d+9\\&-d^2c_d^2+5dc_d^2-7c_d^2+d^2c_d-5dc_d+7c_d\\=\;&(d-2)c_d^3+(-d^2+4d-6)c_d^2+\overbrace{(-d^2+2)}^{-d+3+d-2-d^2+2d+3d-6-5d+7}c_d+d^2-6d+9\end{aligned}$$

Up to now, our goal at showing that the expression is always non-negative does look far away. We cannot simply use $d-1\leq c_d\leq d$ at this stage (and hope for a positive result), since the relation we have to show is surely not true for all $d-1\leq c_d\leq d$; i.e., somehow we must make use of the characteristic polynomial for c_d. Now, observe that the last line that we obtained is bounded by zero iff the following line is:

$$(d-2)c_d^4+(-d^2+4d-6)c_d^3+(-d^2+2)c_d^2+(d^2-6d+9)c_d \qquad (12)$$

By multiplying the characteristic polynomial with $(d-2)$, we obtain the following expression that we know will vanish:

$$\begin{aligned}
&(d-2)[c_d^4 - 3c_4^3 - (d^2-5d+5)c_d^2 + c_d + (d^2-6d+9)] \\
=\ &(d-2)c_d^4 + (-3d+6)c_d^3 \\
&+ \underbrace{(2-d)(d^2-5d+5)}_{=-d^3+7d^2-15d+10} c_d^2 + (d-2)c_d \\
&+ \underbrace{(d-2)(d^2-6d+9)}_{=d^3-8d^2+21d-18}
\end{aligned}$$

Of course, we can subtract this complicated-looking "zero" from Eq. (12) without changing its value. We thus see that we must show that the following expression is bounded from above by zero; we will prove this in a chain of inequalities whose schematics follows the well-known Horner scheme, where we only use that $d \geq 4$ and that $d - 1 \leq c_d$.

$$\begin{aligned}
&\overbrace{(-d^2+7d-12)}^{<0} c_d^3 + (d^3-8d^2+15d-8)c_d^2 \\
&+(d^2-7d+11)c_d - d^3 + 8d^2 - 21d + 18 \\
&\leq \underbrace{[\overbrace{(-d^2+7d-12)(d-1)}^{=-d^3+8d^2-20d+12} + d^3 - 8d^2 + 15d - 8]}_{-5d+4} c_d^2 \\
&+(d^2-7d+11)c_d - d^3 + 8d^2 - 21d + 18 \\
&\leq \underbrace{[\overbrace{(-5d+4)(d-1)}^{=-20d^2+9d-4} + (d^2-7d+11)]}_{-19d^2+2d+7} c_d \\
&-d^3 + 8d^2 - 21d + 18 \\
&\leq \overbrace{(-19d^2+2d+7)(d-1)}^{=-19d^3+21d^2+5d-7} - d^3 + 8d^2 - 21d + 18 \\
&= -20d^3 + 29d^2 - 16d + 9 \\
&\leq (-80+29)d^2 - 16d + 9 \\
&\leq (-204-16)d + 9 \\
&\leq 0
\end{aligned}$$

11 Appendix: More on 4-WHS

We are claiming that Lemma 9 actually provides the worst case for 4-WHS, based on a deeper analysis and (again) slightly changed heuristic priorities (which we will not make explicit in this case but which will become clear from the analysis). So, we are going to show:

Theorem 5. *Let c_4 denote the largest positive real root of the polynomial $x^3 - 4x^2+3x-1$. Then $T_4^0(k) = c_4^k$, $T_4^1(k) = \alpha_{4,1}c_4^k$ with $\alpha_{4,1} = (c_4-2)(c_4-1)/c_4$ and $T_4^2(k) = \alpha_{4,2}c_4^k$ with $\alpha_{4,2} = (c_4-1)/c_4$ solve the system (10). Moreover, $\mathcal{O}^*(c_4^k)$ is an upper bound on the running time of our algorithm for solving* Weighted 4-Hitting Set. *We can bound c_4 from above by* 3.1479.

This also proves Cor. 1 from the main text.

As we have already seen before, the worst case (from above) we have to deal with is the case of two edges e_1, e_2 with $|e_1| = |e_2| = 3$ and $\{x\} = e_1 \cap e_2$. We will analyze two sub-cases: (a) $\exists e_1', e_2'$: $e_i' \cap (e_i \setminus \{x\}) \neq \emptyset$ but $e_i' \cap e_{3-i} = \emptyset$ for $i = 1, 2$. (b) $\forall e_1', e_2'$ with $e_i' \cap (e_i \setminus \{x\}) \neq \emptyset$: $e_i' \cap e_{3-i} \neq \emptyset$ as well, for $i = 1, 2$.

In case (a), we can branch as follows: Taking x into the hitting set gives a $T_4^0(k-1)$-branch. Otherwise, due to the condition, let us continue branching at $\{x_1\} = e_1 \cap e_1'$. (Observe that due to weighted vertex domination, not all e_1' satisfying the condition (a) may contain $\{x_1, x_2\} = e_1 \setminus \{x\}$.) Similarly, there is some $\{y_1\} = e_2 \cap e_2'$. So, if we take both x_1 and y_1 into the hitting set, we get a $T_4^0(k-2)$-branch. If we take y_1 into the hitting set but not x_1, we must select x_2. Since $y_1 \notin e_1$ by (a), this is a $T_4^1(k-2)$-branch. Similarly, taking x_1 into the hitting set but not y_1 is a $T_4^1(k-2)$-branch. If neither x_1 nor y_1 go into the hitting set, then we gain two new small edges due to condition (a), so this is even a $T_4^2(k-2)$-branch. Altogether, we have derived in case (a):

$$T_4^2(k) \leq T_4^0(k-1) + T_4^0(k-2) + 2T_4^1(k-2) + T_4^2(k-2). \tag{13}$$

In case (b), there must be an edge e with $(e_1 \setminus \{x\}) \cap e \neq \emptyset$ and $(e_2 \setminus \{x\}) \cap e \neq \emptyset$, since otherwise (i.e., if the "forall condition" is vacuously satisfied) the weighted vertex domination rule would trigger and result in the following branching:

$$T_4^2(k) \leq T_4^0(k-1) + 2T_4^0(k-2).$$

We will see that the case we are going to consider will result in a branching that is strictly worse, so that we can neglect this case. Moreover, we can also assume that $|(e_1 \setminus \{x\}) \cap e| = 1$, for if not, the weighted vertex domination rule would trigger in the case that x is not taken into the hitting set. This means that one of the two vertices from $(e_1 \setminus \{x\}) \cap e$ must go into the hitting set. Moreover, since e is now hit, two sub-cases arise: either both vertices from $e_1 \setminus \{x\}$ are contained in e and there are no edges that contain vertices from $e_2 \setminus \{x\}$ apart from e_2 and possibly e; then, the weighted vertex domination rule would trigger once more and altogether yield the branch we already observed before, namely:

$$T_4^2(k) \leq T_4^0(k-1) + 2T_4^0(k-2);$$

or, say $y_1 \in e_2 \setminus \{x\}$ is contained in one other edge e_y besides e_2 and e, and no vertex from $e_1 \setminus \{x\}$ is contained in e_y. Branching at y_1 would hence gain us one small edge at least in the situation that y_1 is not going into the hitting set. Altogether, we get as estimate:

$$T_4^2(k) \leq T_4^0(k-1) + T_4^0(k-2) + T_4^1(k-2).$$

This is again always better than the general estimate that we derive now. So, we can assume now that $\{x_1\} = (e_1 \setminus \{x\}) \cap e$ and that $\{y_1\} = (e_2 \setminus \{x\}) \cap e$. Assume we start branching at x_1. Let us call $\{x_2\} = e_1 \setminus \{x, x_1\}$. We distinguish two sub-cases regarding $\{y_2\} = (e_2 \setminus \{x, y_1\})$: (i) $\delta(y_2) = 1$ and (ii) $\delta(y_2) \geq 2$. Case (i) is again split into two cases: (ia) $|\{e \in E \mid y_1 \in e, x_1 \notin e\}| = 1$ and (ib) $|\{e \in E \mid y_1 \in e, x_1 \notin e\}| > 1$.

In case (ia), if x_1 is taken into the hitting set, we will delete either y_1 or y_2 due to the weighted vertex domination rule, and then this edge is resolved by the small edge rule. This gives a $T_4^0(k-2)$-branch. If x_1 is not taken into the hitting set, x_2 must be in. Moreover, if we continue branching at y_2, we get a $T_4^0(k-2)$-branch when y_1 goes into the hitting set and two $T_4^0(k-3)$-branches when not y_1 but y_2 is in the hitting set, since then one of the two remaining vertices from e must be in the hitting set, too. Overall, we get in this case:

$$T_4^2(k) \leq T_4^0(k-1) + 2T_4^0(k-2) + 2T_4^0(k-3).$$

(Notice that this is strictly speaking a tight analysis for $\delta(y_1) = 2$.) Again, this is not the worst case to consider. In case (ib), if x_1 is taken into the hitting set, we might take y_1 into the hitting set. This gives a $T_4^0(k-2)$-branch. If y_1 does not go into the hitting set, then y_2 will, giving a $T_4^1(k-2)$-branch (gaining a small edge by the case assumption). If x_1 is not going into the hitting set but x_2, we get one $T_4^0(k-2)$-branch and two $T_4^1(k-3)$-branches. This yields:

$$T_4^2(k) \leq T_4^0(k-1) + 2T_4^0(k-2) + T_4^1(k-2) + 2T_4^1(k-3);$$

again, this is not the worst case to consider.

In case (ii), we can assume that there is an edge e_y that contains y_2 but none of the vertices from $\{x, y_1, x_1\}$. Namely, since $\delta(y_2) \geq 2$, there is at least one edge e_y besides e_2 that contains y_2. As can be seen, $\delta(y_2) = 2$ is the worst case we assume henceforth. If $y_1 \in e_y$, the weighted edge domination rule would have triggered, yielding a better branching as analyzed before. If $x_1 \in e_y$, we have a situation analyzed under case (i) above. We branch as follows: If x_1 goes into the hitting set, we can gain a small edge in the case that y_2 does not go into the hitting set (assume we continue branching at y_2); hence, this is one $T_4^0(k-2)$-branch and one $T_4^1(k-2)$-branch. If x_1 is not in the hitting set, let us continue branching at y_1. By a simple analysis, we get one $T_4^0(k-2)$ and two $T_4^0(k-3)$-branches (similar as in the previous cases). This means:

$$T_4^2(k) \leq T_4^0(k-1) + 2T_4^0(k-2) + T_4^1(k-2) + 2T_4^0(k-3). \tag{14}$$

We will show now that the case that $e_1 \cap e_2 = \emptyset$ is in fact the worst case that yields the estimate $T_4^2(k) = T_4^1(k-1) + 2T_4^2(k-1)$.

We first consider Eq. (13): we have to find an upper bound for

$$-T_4^2(k) + T_4^0(k-1) + T_4^0(k-2) + 2T_4^1(k-2) + T_4^2(k-2).$$

With the settings of the functions as formulated in Theorem 5, this expression means:

$$-c_4^k + 2c_4^{k-1} + c_4^{k-2} + 2(c_4-2)(c_4-1)c_4^{k-3} + c_4^{k-2} - c_4^{k-3}.$$

After multiplication with c_4^{3-k}, we get the following chain:

$$\begin{aligned}
&-c_4^3 + 2c_4^2 + 2c_4^1 + 2(c_4-2)(c_4-1) - 1 \\
&= -c_4^3 + 2c_4^2 + 2c_4^1 - 1 + 2c_4^2 - 6c_4 + 4 \\
&= [-c_4^3 + 4c_4^2 - 3c_4 + 1] - c_4 + 2 \\
&\leq 0
\end{aligned}$$

The expression in square brackets vanishes, since c_4 is a root of the characteristic polynomial mentioned in Theorem 5, and the inequality follow from $3 \leq c_4$.

Let us consider Eq. (14): we will find an upper bound for

$$-T_4^2(k) + T_4^0(k-1) + 2T_4^0(k-2) + T_4^1(k-2) + 2T_4^0(k-3)$$

under the settings of the functions as formulated in Theorem 5. This means we have to upperbound:

$$-c_4^k + 2c_4^{k-1} + 2c_4^{k-2} + (c_4-2)(c_4-1)c_4^{k-3} + 2c_4^{k-3}.$$

After multiplication with c_4^{3-k}, we get the following chain:

$$\begin{aligned}
&-c_4^3 + 2c_4^2 + 2c_4^1 + (c_4-2)(c_4-1) + 2 \\
&= -c_4^3 + 3c_4^2 - c_4 + 4 \\
&= [-c_4^3 + 4c_4^2 - 3c_4 + 1] - c_4^2 + 2c_4 + 3 \\
&\leq -c_4^2 + 3c_4 \\
&= c_4(3 - c_4) \\
&\leq 0
\end{aligned}$$

The expression in square brackets vanishes, since c_4 is a root of the characteristic polynomial mentioned in Theorem 5, and the two inequalities follow from $3 \leq c_4$.

12 A general final remark on the analysis

Remark 1. The astute reader might have noticed that there are possible situations say with two or more edges of size $d-1$ where the reduction rules actually destroy *all* of them and replace them by *one* edge of size $d-2$ or smaller. However, it can be easily verified that, for all $d \geq 3$ and $\ell \geq 2$—as far as analyzed—, $T_d^\ell(k) \geq (d-2)^k$. This is based on the assumption $c_d \geq d-1$. Therefore, this omission in the analysis will never affect the worst case running time claimed in the paper.