Parameterized Approximation Algorithms for Hitting Set

Ljiljana Brankovic[1] and Henning Fernau[2]

[1] School of Electrical Engineering and Computer Science
The University of Newcastle, Callaghan, NSW 2308, Australia
Ljiljana.Brankovic@newcastle.edu.au
[2] Fachbereich 4, Abteilung Informatik
Universität Trier, 54286 Trier, Germany
fernau@uni-trier.de

Abstract. We are going to analyze simple search tree algorithms for approximating d-Hitting Set, focussing on the case of factor-2 approximations for $d = 3$. We also derive several results for hypergraph instances of bounded degree, including a new polynomial-time approximation.

1 Introduction

Our approach—in general. There is now a growing body of literature concerned with combining two very natural ideas to cope with intractability: that of approximation and that of parameterized algorithms; see [4] for background information. We are putting here a recent approach of ours [3] to another test: namely, the idea of combining search tree algorithms with approximation to obtain better approximation ratios at the cost of (moderately) exponential time.

Problem statement. The d-Hitting Set (d-HS) can be viewed as a "vertex cover problem" on hypergraphs, formally stated as follows:

Given: A hypergraph $G = (V, E)$ with *edge size* bounded by d: $\forall e \in E(|e| \leq d)$
Parameter: a non-negative integer K
Question: Is there a *hitting set* C of cardinality of at most K, i.e.,
$\exists C \subseteq V \forall e \in E(C \cap e \neq \emptyset)$ and $|C| \leq K$?

We will mostly consider the case $d = 3$ in what follows.

In this paper, we follow the definition proposed in [4], which in our context can be phrased as follows. Given a hypergraph G and a parameter K such that a minimum hitting set C^* of G satisfies $|C^*| \leq K$, a *parameterized approximation algorithm with (constant) approximation ratio* ρ for minimum hitting set produces a hitting set C such that $|C| \leq \rho|C^*|$. If $K < |C^*|$, the algorithm will answer NO. Such an algorithm runs in time $\mathcal{O}^*(f(K))$ for some function f, where $\mathcal{O}^*$ notation suppresses polynomial factors.

Bibliographical notes. Despite all efforts, the best known constant factor polynomial time approximation algorithm for general (unweighted) hypergraphs with

R. Solis-Oba and G. Persiano (Eds.): WAOA 2011, LNCS 7164, pp. 63–76, 2012.

edge size d is still a factor-d approximation for fixed d, and this is even optimal under the unique games conjecture [10]. So, even the factor-2 approximation we focus on in this paper is is a considerable progress. Several exact parameterized algorithms have been developed for our problem. Let us only mention the best published results here: The best publically available algorithm for 3-HS is Wahlström's, as it appeared in his PhD Thesis [12], having a running time of $\mathcal{O}^*(2.076^K)$. The best published one has only a running time of $\mathcal{O}^*(2.179^K)$; see [7].

Why Hitting Set*?* (1) Hitting Set problems show up in many places; e.g., Reiter's ground-breaking research on *model-based diagnosis* [11] relates the automatic diagnosis of systems to Hitting Set, or HS for short. (2) vertex cover, or VC for short, is the paradigmatic test-bed problem for parameterized algorithms. As HS can be seen as a vertex cover problem on hypergraphs, it is quite a natural question to see how the ideas developed in [3] might generalize to the case of hypergraphs. It should be noted that the paper [2] can be seen as a sort of precursor of [3], although the techniques are quite different.

Why using exponential time for approximation? As we have seen when working on our VC-approximation paper, this kind of work often also gives new insights and ideas for polynomial time approximations, for instance, new reduction rules. On the other hand, we believe that this is also interesting for "classical FPT", keeping in mind that, at least with search tree algorithms, what slows them down is finding (proving) a NO answer: In that case, often the whole search tree has to be traversed. Approximation algorithms, be them polynomial time or FPT, can serve to find a quicker NO. Since we are talking exponential time anyways, a "very fast" exponential time algorithm may be worthwhile running first (or in parallel) to find this quick NO answer, even though we might be mainly interested in finding exact answers. As it is considered unlikely to find a better polynomial time approximation algorithm than the (trivial) one offering a factor of three. Whoever is interested in finding better approximation guarantees must therefore use exponential time.

The results of this paper. We show a branching algorithm that enables to approximate 3-HS within a factor of two, running in time $\mathcal{O}^*(1.29^K)$; see Sec. 3 and 4. For subcubic instances, the analysis can be improved to show an upper bound of $\mathcal{O}^*(1.26^K)$; see Sec. 5. These figures compare favorably with the ones that can be obtained by employing the best exact algorithms for approximation purposes, as explained in Sec. 2. We also give several results for approximating d-HS for general d, as well as a new polynomial-time algorithm for approximating 3-HS for instances of degree bounded by three up to a factor of $\frac{5}{2}$.

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 $\deg(v)$.

2 A Simple Design for Parameterized Approximation

Most of the currently best algorithms for 3-HS are all based on a search tree algorithm, combined with the use of reduction rules. To each node n of the search tree, a set of vertices C_n of the input hypergraph $G = (V, E)$ can be associated that collects a partial hitting set, i.e., in the subtree T_n rooted at n, we are only interested in hitting set solutions S that contain C_n. C_n has been constructed on the path from the root of the search tree down to n by invoking the simple operation "`Put` x `into the solution.`" $|C_n|$ times. We can also associate a "current hypergraph" $G_n = (V_n, E_n)$ to n that can be easily obtained from G and C_n.

The technique for obtaining an approximative solution from such an algorithm is very simple through *interleaving* a step that deliberately worsens a solution in order to speed up the branching. So, we associate a hyperedge set $H_n \subseteq E$ to n that has been formed as follows (to obtain a factor-2 approximation for the ease of presentation): Whenever some vertex x is put into the solution, we pick some $e \in E_n$ and put it into H_n. The approximative solution associated to n is now $S_n := C_n \cup V(H_n)$. Of course, the "current hypergraph" $G'_n = (V_n, E_n)$ associated to n now depends on G and (C_n, H_n).

This procedure guarantees the following properties:

- If $G = (V, E)$ has a hitting set S of size K, then there is a path in the search tree of the approximative branching algorithm of length at most $K/2$ such that C_l and H_l are associated to the leaf l of that path, with $C_l \subseteq S \subseteq S_l$.
- Any valid hitting set solution C with $C_l \subseteq C \subseteq S_l$ contains at least one vertex from each edge from H_l, i.e., $|C| \geq 2|C_l|$ by construction, since $|H_l| = |C_l|$.
- Hence, $2|C_l| \leq |S|$ and $4|C_l| = |S_l|$, and S_l is an approximative solution that is at most twice as big as the optimum (constrained to sets embracing C_l).

This allows us to conclude, using Fernau's result [7]:

Proposition 1. *There is a factor-2 approximation of* 3-HS *in time* $\mathcal{O}^*(1.477^K)$.

There is a technical difficulty when trying to apply Wahlström's result [12], namely the Measure & Conquer analysis that he employs. It is not completely clear if we can always choose an edge to worsen the approximation factor whose removal reduces the measure by the same amount as the previous branching did. As we will present considerably better running times in this paper, this question is of no major concern.

The same approach can be used for other variants of d-HS, as summarized in the following table. There, the first line lists the running time estimates that we obtained in [6] for the exact solution, ρ_d shows the intended approximation factor, where $\rho_d = (d+1)/2$, since each vertex that we put into the hitting set is accompanied by one worsening step, and $T_d^{\rho_d}$ shows the obtained running times; clearly, for the chosen approximation factors, the basis of $T_d^{\rho_d}$ is just the square root of the basis for T_d.

By letting grow H_n more slowly or more rapidly along the search tree, it is straightforward to obtain similar results for other (better or worse) approximation factors with accordingly changed (slower or faster) running times. We leave

Table 1. Approximation factors and running times obtained based on [7,6]

d	3	4	5	6	10	100
$T_d(K) \leq$	2.18^K	3.12^K	4.08^K	5.05^K	9.02^K	99.0002^K
$\rho_d =$	2	2.5	3	3.5	5.5	55.5
$T_d^{\rho_d}(K) \leq$	1.51^K	1.77^K	2.02^K	2.25^K	3.01^K	9.95^K

out the according details in this extended abstract, but rather refer to similar reasonings for VC in [3].[1]

In the following, we will present branching algorithms that are simpler and simpler to analyze than those from [7,6,12], but where a more elaborated use of interleaving nonetheless leads to better running times. For the ease of presentation, we will no longer differentiate between C_n and $V(H_n)$ in what follows, but just assume that a partial hitting set C_n is associated to each node n of the search tree. Whenever clear from the context, we omit the index n. Similarly abusing notation, we also allow $G = (V, E)$ to refer to the "current hypergraph."

It is worth noticing that the approach for approximating VC of [2] (as detailed in the PhD Thesis of N. Bourgeois) that consisted in first splitting the given instance into parts and then computing exact solutions for each of the parts is dependent on some kernelization results that are not available in this case.

3 A Simple Branching for Approximation

We are now elaborating on the interleaving idea further, looking at a very simplistic-looking general branching algorithm. Due to the fact that we observe a good approximation ratio when a certain vertex is not put into the partial hitting set, even this simple branching already improves on the idea presented in the previous section. In the beginning, Algorithm 1 is called with the parameters $(G, K, \emptyset)$, where (G, K) is the original 3-HS instance that we want to solve. This original parameter K is also occasionally used by Algorithm 1, while k refers to the value of the parameter in the current situation.

The algorithm uses the following ingredients:

- Several reduction rules are known for HS; we will list (some of) them below, including possibly rules that are only valid in an approximative sense.
- We still have to specify heuristic priorities that might improve our branching.
- Whenever a vertex is put into the hitting set, an edge is selected for worsening the solution. Also here, we might introduce some selection strategies to improve on the running time.
- It is well-known that (3)HS can be solved to optimality in polynomial time if each vertex belongs to at most two edges by invoking some Edge Cover algorithm on an auxiliary graph.

[1] Similar ideas have been presented at WorKer 2011 by H. Shachnai, joint work with M. Fellows and F. Rosamond.

Algorithm 1. `3HS-2-appr-general`: A 2-approximation algorithm for 3-HS

1: Input: Hypergraph $G = (V, E)$, parameter k, and a partial hitting set C
2: Output: Either NO or a hitting set C with $|C| \leq 2K$.
3: Apply all reduction rules exhaustively, possibly modifying C and k.
4: **if** $k < 0$ **then**
5: Return NO.
6: **else if** possible: choose a $v \in V$ such that $\deg(v) \geq 3$ according to heuristic priorities. **then**
7: Binary branch on v:
8: Case 1: Put $v \in C$, i.e., $C \leftarrow C \cup \{v\}$, $V \leftarrow V \setminus \{v\}$, $E \leftarrow \{e \in E \mid e \subseteq V\}$.
Select some $f \in E$ for worsening and put all vertices of f into C, i.e.,
$C \leftarrow C \cup f$, $V \leftarrow V \setminus f$, $E \leftarrow \{e \in E \mid e \subseteq V\}$, $k \leftarrow k - 2$
Recursively call `3HS-2-appr-general` with the modified parameters.
9: Case 2: Do not put v into C, i.e.,
$V \leftarrow V \setminus \{v\}$, $E \leftarrow \{e \setminus \{v\} \mid e \in E\}$.
Recursively call `3HS-2-appr-general` with the modified parameters.
10: **else**
11: Solve the remaining instance optimally using an Edge Cover algorithm.
12: Return either NO or a hitting set C with $|C| \leq 2K$.

Simple reduction rules. We first list the (well-known) reduction rules valid for (3)HS, as they can be found, e.g., in [7].

- (hyper)edge domination: A hyperedge e is *dominated* by another hyperedge f if $f \subset e$. In that case, delete e.
- tiny edges: Delete all hyperedges of size one and place the corresponding vertices into the hitting set.
- 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.

Notice that the tiny edge rule puts a vertex into the hitting set. Since this is an exact rule, we may worsen the solution in order to obtain a sufficiently approximated solution. One easy consequence of the vertex domination rule is that we can assume a minimum degree of two in an irreducible instance. Moreover, for each vertex pair (x, v) of an irreducible instance, there exists an edge pair (*irreducibility witness*) (e, f) with $x \in f, v \notin e$, but $x \notin f, v \in e$.

The following simple rule preserves the approximation factor of two.

- small edges: If e is a hyperedge of size two, i.e., $e = \{x, y\}$, then put both x and y into the hitting set.

In general, we will always first employ exact reduction rules before employing approximative reduction rules.

Analyzing a simplistic branching. In the first case of the branching, v is put into C, and afterwards, the solution is worsened by putting all vertices of an edge e

into C, as well. Since any optimum solution that contains v will also contain at least one vertex from e, while the algorithm will, altogether, put $e \cup \{v\}$ into C, with $|e \cup \{v\}| \leq 4$, this locally preserves the claimed 2-approximation factor.

In the second case of the branching, v is not put into C. Clearly, there is some edge $e \in E$ containing v. In this second case, v will be removed from e. In the recursive call, the small edges rule triggers and puts from e into C.

So, at this point of the analysis, we face one $(2, 1)$ branching vector, i.e., a branching number of 1.619, obviously worse than what we got by profiting from earlier analysis for exact algorithms. However, there is some hope that we might get better running time estimates. For instance, since we know that the vertex v is (at least) of degree three, we might find that there are edges e, f, g containing e such that $|e \cup f \cup g| = 7$. Now, in Case 2 of the branching, three small edges are produced. This alone gives already a $(2, 3)$ branch, i.e., $\mathcal{O}^*(1.325^K)$, for our factor-2 approximation. We deliver a detailed analysis in the next section.

4 A More Elaborated Analysis of a Factor-2 Approximation Algorithm for 3-HS

More approximation-preserving reduction rules.

- approximative vertex domination: Assume there is a hyperedge $e = \{x, y, z\}$ such that, whenever x belongs to some hyperedge h, then y or z also belong to h. Then, we put y and z together into the hitting set that we produce.

Lemma 1. *The approximative vertex domination rule is correct for an algorithm aiming at a factor-2 approximation.*

Proof. Namely, assume that an optimum solution contains x. Then, we can replace x in that solution by y and z, losing a factor of two, but still having a valid hitting set. If x is not in any optimum solution C, then, in order to hit e, y or z (or both) must be in C, so our rule loses again at most a factor of two. □

This rule alone is already very powerful. Consider any vertex v that Algorithm 1 chooses for branching. By approximative vertex domination, v must belong to two edges e, f with $|e \cup f| = 5$, as otherwise, fixing some e where v belongs to, all edges f that contain v would also contain some other vertex of e. In particular, this reasoning shows that there is for any vertex v of degree two, the two edges containing v will together host five vertices. This yields a branching vector of $(2, 2)$, which already improves on the much more sophisticated exact algorithms that were (ab)used to produce approximative solutions in Proposition 1.

- small triangle situation: Assume there are three small hyperedges $e = \{y, z\}$, $f = \{x, y\}$, $g = \{x, z\}$. This describes a triangle situation (e, f, g). Then, we put $\{x, y, z\}$ together into the hitting set, and we can even choose another hyperedge of size three to worsen the ratio.

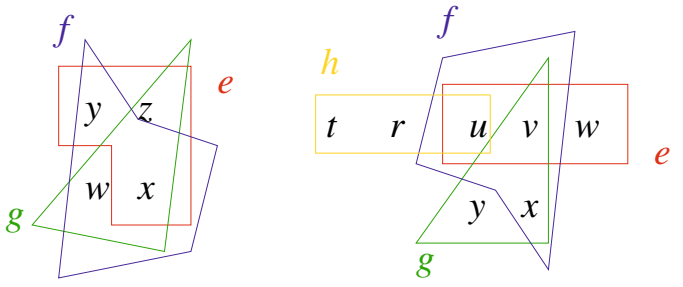

Fig. 1. Special situations for better branching

Notice that it is clear that two of the three vertices $\{x, y, z\}$ must be in any optimum solution. So, if we put $\{x, y, z\}$ together with a whole edge h into the hitting set, out of these six vertices, three must be in any optimum solution. This shows the validity of this rule. To improve our branching, we will always first look for small triangle situations before applying the small edges rule.

- approximative double vertex domination: Assume there is a hyperedge $e = \{x, y, a\}$ and another vertex b such that, whenever x or y belong to some hyperedge h, then a or b also belong to h. Then, we put a and b together into the hitting set that we produce.

Lemma 2. *The approximative double vertex domination rule is correct for an algorithm aiming at a factor-2 approximation.*

Proof. Assume that an optimum solution contains at least one of the vertices x and y. Then, we can replace The vertices x and y in that solution by a and b, losing at most a factor of two, but still having a valid hitting set. If neither x nor y is in any optimum solution C, then, in order to hit e, a must be in C, so our rule loses again at most a factor of two. □

We will now describe how this new reduction rule can help in certain branching scenarios. Assume there are three hyperedges $e = \{x, y, z\}$, $f = \{x, y, w\}$, $g = \{x, w, z\}$. This describes a potential triangle situation $(x; e, f, g)$ in the sense that, when branching at vertex x, in the branch that does not put x into the hitting set, a small triangle situation will be produced. Hence, we can reduce the parameter k by three in that branch.

We can experience a similar profit from vertex domination. We assume there are hyperedges $e = \{u, v, w\}$, $f = \{u, v, x\}$ and a further edge $g \neq \{v, w, x\}$ containing v but not u; we can further assume $\deg(u) \geq 3$ (as otherwise v dominates u), and that none of the further edges h_i containing u also contain v. Finally, we consider edge $h = \{u, r, t\}$ and assume that all edges h_i containing u, also contain r or t. We then consider a binary branch at v. In the branch when v is put into the hitting set, the approximative vertex domination rule triggers on vertex u. Hence, we can reduce the parameter by at least three in this branch

(including the worsening step). In the case when v is not put into the hitting set, then the small edges $e' = e \setminus \{v\} = \{u, w\}$, $f' = f \setminus \{v\} = \{u, x\}$ and $g' = g \setminus \{v\} \neq \{x, w\}$ (with $u \notin g'$) are produced, so that either $e' \cap g' = \emptyset$ or $f' \cap g' = \emptyset$. Hence, the small edge rule can be performed twice, yielding a parameter reduction of two. We will term $(v; e, f, g)$ a potential vertex domination situation.

Algorithm 2. `3HS-2-appr`: A more specific 2-approximation algorithm for 3-HS

1: Input: Hypergraph $G = (V, E)$, parameter k, and a partial hitting set C
2: Output: Either NO or a hitting set C with $|C| \leq 2K$.
3: Apply all reduction rules exhaustively, possibly modifying C and k.
4: **if** $k < 0$ **then**
5: Return NO.
6: **else if** possible: choose (a) an internal or (b) [if (a) fails] an external branching pair (u, v) **then**
7: Binary branch on $\{u, v\}$ as follows:
8: Case 1: Put $u, v \in C$, i.e.,
$C \leftarrow C \cup \{u, v\}$, $V \leftarrow V \setminus \{u, v\}$, $E \leftarrow \{e \in E \mid e \subseteq V\}$, $k \leftarrow k - 1$
Recursively call `3HS-2-appr` with the modified parameters.
9: Case 2: Do not put u, v into C, i.e.,
$V \leftarrow V \setminus \{u, v\}$, $E \leftarrow \{e \setminus \{u, v\} \mid e \in E\}$.
Recursively call `3HS-2-appr` with the modified parameters.
10: **else**
11: Solve the remaining instance optimally using an Edge Cover algorithm.
12: Return either NO or a hitting set C with $|C| \leq 2K$.

Explaining a more refined branching. In Algorithm 2, we follow another branching strategy in a hypergraph $G = (V, E)$ by selecting branching pairs. More specifically, a pair (u, v) of vertices, where $\deg(u), \deg(v) \geq 3$, is called a *branching pair* if one of two conditions is met:

- If there are two hyperedges $e = \{u, v, w\}$, $f = \{u, v, x\}$, then (u, v) is an *internal* branching pair. We branch on this case with preference.
- If in the hypergraph $G' = (V', E')$ that is obtained from G by deleting u and all the edges u belongs to, v has still degree at least three, then (u, v) is an *external* branching pair.

In the penultimate line of the algorithm, it might still be that there are vertices of degree at least three. However, as we do not find any branching pair, these vertices will disappear in a single branch that will afterwards allow the use of an Edge Cover algorithm to solve the Hitting Set instance with maximum degree of two.

Clearly, an optimum solution either contains u or v, or neither u nor v. We are worsening this case distinction by a factor of two if we consider the case when u or v is in an optimum solution together, so taking $\{u, v\}$ into the solution. This is done in the first branch. If this is not the case, neither u nor v are in an

optimum solution. However, in the second branch, when u, v are removed from the instance, reduction rules will apply We are giving a more refined analysis in the following theorem. The following lemma is crucial to show the running time of `3HS-2-appr`.

Lemma 3. *If the maximum degree in an irreducible hypergraph G is at least three, then there must exist a vertex v with* $\deg(v) \geq 3$*, called* preferred vertex*, that satisfies one of the following cases.*

1st branching scenario *There is another vertex $u \in V$ and three edges e, f, g such that $\{u, v\} = e \cap f \cap g$.*

2nd branching scenario *There are 3 edges e, f, g s.t. (A) $\{v\} = e \cap f \cap g$ and $|e \cup f \cup g| = 7$ or $(v; e, f, g)$ describes (B) a potential triangle situation or (C) a potential vertex domination situation.*

Proof. Consider an irreducible hypergraph $G = (V, E)$ with maximum degree at least 3, and consider a vertex $v \in V$ with degree at least 3. Then one of the following three cases must be satisfied:

1. There are three edges e, f, g containing v such that $e \cap f \cap g = \{u, v\}$; then we have the first branching scenario.
2. There are no three edges e, f, g containing v such that $e \cap f \cap g = \{u, v\}$, but there are two edges containing v such that $e \cap f = \{u, v\}$, say $e = \{u, v, w\}$ and $f = \{u, v, x\}$.
   - If there exists an edge $h = \{v, w, x\}$ or $h = \{u, w, x\}$ then $\{v; e, f, h\}$ (or $\{u; e, f, h\}$) is a potential triangle situation (case (B) in the second branching scenario). In what follows we assume that such edge h does not exist.
   - Since the hypergraph G is irreducible, the vertex u also obeys $\deg(u) \geq 3$.

     Claim: There is an edge g such that $g \cap (e \cup f) = \{v\}$ or $g \cap (e \cup f) = \{u\}$. Namely, any edge h ($h \notin \{e, f\}$) containing v [or u, resp.] will not contain u [or v, resp.] to avoid the first branching scenario. To falsify the claim, $h \cap \{w, x\} \neq \emptyset$. To avoid the potential triangle situation described in the previous item, $|h \cap \{w, x\}| = 1$. Still, any hyperedge h containing u or v also contains w or x, which is not possible, as the approximative double vertex domination rule would have dealt with this situation. ◊

     Without loss of generality, assume that $g \cap (e \cup f) = \{v\}$, and let j be another edge containing u (but not v).
     - If there are no more edges containing v, then $\{u; e, f, j\}$ describes a potential vertex domination situation.
     - If there is another edge ℓ containing v, then in order to avoid the potential triangle situation $\ell \neq \{v, w, x\}$ and thus $|\ell \cap \{w, x\}| \leq 1$. Hence, $\{v; e, g, \ell\}$ or $\{v; f, g, \ell\}$ describe the second branching scenario, case (A).
3. There are no two edges e, f containing v such that $e \cap f = \{u, v\}$ for some vertex u; then we have the second branching scenario, case (A). □

Table 2. A list of branching vectors, keeping track of the worsening steps

Situation	Branching vector	subcubic case?	with $w = 1$	$w = 1.5$	$w = 2$
1st branching sc.	$(1, 3 + 3w)$	No	1.2852	1.2431	(No)
2nd b.s., Case (A)	$(1 + w, 3)$	Yes	1.3196	1.2600	1.2356
2nd b.s., Case (B)	$(1 + w, 2 + w)$	No	$\leq$ Case (A)	$\leq$ Case (A)	$\leq$ Case (A)
2nd b.s., Case (C)	$(2 + w, 2)$	Yes	1.3196	1.2600	1.255

Remark 1. As an aside, let us mention that there is a simple variant of Algorithm 1 that branches on preferred vertices, if possible internal branching pairs, and would obtain the branching numbers listed in Table 2, referring to the analysis of the previous lemma. The parameter w refers to a worsening step; usually, $w = 1$. Our new analysis profits from external branching pairs, as we will see.

Theorem 1. *`3HS-2-appr` can be used to find a 2-approximation* 3-HS *of a 3HS-instance $G = (V, E)$ in time $\mathcal{O}^*(1.2852^K)$.*

Proof. The correctness of the Algorithm 2 has been discussed before.

Now we turn to the running time analysis. The first branching scenario is encountered if the algorithm branches on an internal branching pair. It assumes the existence of two vertices u and v and three edges e, f, g such that $e \cap f \cap g = \{v, u\}$. Hence, the recursive call faces three tiny edges: $e' = e \setminus \{v, u\}$, $f' = f \setminus \{v, u\}$, and $g' = g \setminus \{v, u\}$. The tiny edges rule will then put three vertices into the hitting set, but since this is an exact reduction rule, three additional independent edges can be selected and put into the hitting set. Altogether, this yields a branching vector of $(1, 6)$ and a branching number of 1.2852.

If no internal branching pair exists, then we face the 2nd branching scenario described in Lemma 3. Let v be some preferred vertex. Let V_v and E_v collect all vertices and edges that are directly affected by branching at v. This means: (a) If $\{v\} = e \cap f \cap g$ and $|e \cup f \cup g| = 7$ or if $(v; e, f, g)$ describes a potential triangle situation, then $V_v = e \cup f \cup g$, $E_v = \{e, f, g\}$; (b) if $(v; e, f, g)$ describes a potential vertex domination situation, then $V_v = e \cup f \cup g \cup h$, where h is the edge where the (approximative) vertex domination rule will apply to, see the discussion leading to Fig. 1, and E_v collects all edges containing vertices from V_v. Consider $G' = (V', E')$, where $V' = V \setminus V_v$, $E' = E \setminus E_v$. If the maximum degree in G' is at most two, then by branching at v alone according to the 2nd branching scenario, possibly followed by branching at other vertices from E_v, we will produce a hypergraph of maximum degree two that can be solved in polynomial time. The finitely many branches indicated in the previous sentence do not affect the overall running time.

Note that there is no preferred vertex in G' that corresponds to the 1st branching scenario, as such vertex would have been previously selected by the algorithm instead of v as part of an internal branching pair. Hence, we will find a vertex v' that fits into one of the cases of the second branching scenario. For the purpose of analyzing this part of the algorithm, we assign parameter a to vertices v and v', where $a(v) = 1$ if $\{v; e, g, f\}$ describes a potential vertex domination situation, and $a(v) = 0$ otherwise; similarly, $a(v') = 1$ if $\{v'; e', g', f'\}$ describes a

potential vertex domination situation, and $a(v') = 0$ otherwise. Then we branch as follows:

Case 1. At least one of the vertices v and v' is in a minimum hitting set respecting previous choices; then we put $\{v, v'\}$ in C and have a parameter reduction of 1. Additionally, if $\{v; e, g, f\}$ and/or $\{v'; e', g', f'\}$ describes a potential vertex domination situation, we add additional vertices to C, as described above. Thus the total parameter reduction is $1 + a(v) + a(v')$.

Case 2. None of the vertices v and v' is in any minimum hitting set respecting previous choices; then we simply remove v and v' from the vertex set V and from all the edges that contain v or v'. The total parameter reduction is $6 - a(v) - a(v')$.

For the claimed parameter reductions to be true, it is crucial to observe that the branching at v and v' is done independently, as it is guaranteed by the construction of V_v and E_v. Hence, the reductions follow from what we collected for branching at a single vertex in Table 2. In total, we have a branching vector $(1 + a(v) + a(v'), 6 - a(v) - a(v')$. Depending on the values $a(v)$ and $a(v')$, the branching vector can be $(1, 6)$, $(2, 5)$ or $(3, 4)$. Out of the 3 corresponding branching numbers, the largest one is 1.2852, corresponding to $(1, 6)$. □

5 Approximating 3-HS with Degree Constraints

In the related case of Minimum VC, quite some research was undertaken to find better approximations for the degree-restricted case, e.g., for the case of cubic graphs. Surprisingly, to the best knowledge of the author, no such results are known for 3-HS. Also for the problem of finding smaller kernels, only relatively small progress was reported in [9], though that paper is far from trivial. Here, we are going to report on several results, focussing on consequences of running Algorithm 2 on subcubic instances, i.e., instances where each vertex belongs to at most three hyperedges. Proofs are omitted due to space restrictions.

Lemma 4. *If $G = (V, E)$ is a subcubic irreducible 3-HS instance, then we know: If there are two edges $e, f \in E$ with $|e \cap f| = 2$, then for any further edge g with $v \in g$, $e \cap g = e \cap f = \{v\}$.*

Lemma 5. *If $G = (V, E)$ is a subcubic irreducible 3-HS instance, then no potential triangle situation occurs.*

In the following, we assume (in addition), that hyperedge components with at most 27 vertices are solved (exactly) due to table look-up. This will be called the small component rule. This rule, as well as the tiny edge and the small triangle rule are exact rules that put a vertex into the hitting set, so that a worsening step triggers. Hence, these three rules are summarized as *trigger rules*. The following auxiliary results turns out to be useful for proving the crucial Lemma 8.

Lemma 6. *Whenever a hypergraph component completely disappears when applying reduction rules, then the last reduction rule applied was a trigger rule.*

Lemma 7. *Let $G = (V, E)$ be a subcubic irreducible 3-HS instance. The removal of $H \subseteq V$, $|H| \leq 3$, cannot destroy any hypergraph component.*

Lemma 8. *After branching on a subcubic instance or after performing a worsening step, i.e., after the corresponding vertices were put into the hitting set, we find a vertex of degree two or a yet unaccounted small edge, unless we have entered the final polynomial-time phase.*

Theorem 2. *Algorithm 2 can be implemented to find a 2-approximation for* 3-Hitting Set *on subcubic instances in time $\mathcal{O}^*(1.2555^K)$.*

Proof. Due to Lemmas 4 and 5, only the cases marked with "Yes" in Table 2 may occur when running Algorithm 1, modified towards branching on preferred vertices as indicated in Remark 1. A yet unaccounted small edge will first trigger the small edge or the small triangle rule, clearly allowing to add one to each component of the branching vector, yielding, in particular, $w \geq 2$. Otherwise, let v be a vertex of degree two, as it exists due to Lemma 8, pertaining to hyperedges e and f. The idea is to exploit the worsening step as follows. If any hyperedge h containing some vertex from $X = (e \cup f) \setminus \{v\})$, with $h \notin \{e, f\}$, has two vertices from X, then X induces a small hyperedge component, again allowing to reduce the parameter by at least one. So, there is a hyperedge h containing exactly one vertex from $X = (e \cup f) \setminus \{v\}$, with $h \notin \{e, f\}$. Slightly modifying our algorithm, we will always pick such a hyperedge h in the worsening step. This will put (at least) two more vertices in the approximative hitting set compared to what we already accounted for, due to the vertex domination and small edge rules. So, we can always rely on $w \geq 2$ in the branching vectors. Hence, we are facing as worst-case branching vectors: $(3, 3)$ and $(4, 2)$. □

We can make use of the same idea for subcubic instances of d-HS in general. However, we must be careful with the interplay between the intended approximation factor and the corresponding small edges rule. We give some details for the case $d = 4$ in the following. If we put a hyperedge of size 3 into the hitting set, then this gives us an approximation factor of three (only). We show two ways how to deal with this problem: either, we aim at an approximation factor of three only, or we have to set up recurrences that allow for an improved factor-2.5 approximation. The figures are based on Table 1.

Theorem 3. Minimum 4-HS *can be approximated in time $\mathcal{O}^*(1.4613^K)$ up to a factor of three in general and in time $\mathcal{O}^*(1.2556^K)$ on subcubic hypergraphs.*

Theorem 4. Minimum 4-HS *can be approximated in time $\mathcal{O}^*(1.7650^K)$ up to a factor of* 2.5 *in general and in time $\mathcal{O}^*(1.5754^K)$ on subcubic hypergraphs.*

Let us move back to the 3-HS case again, but now applying the idea mentioned last to hypergraphs of maximum degree four. Similar to Lemma 8, we can now assume that, before applying any worsening step, we might find a vertex of degree three in the graph. So, with the help of two subsequent worsening steps, we can produce a small hyperedge due to vertex domination. This means (again) that we can assume $w = 3/2$ in Table 2. Evaluations the corresponding branching vectors are also shown there prove:

Theorem 5. Minimum 3-Hitting Set *can be approximated in time* $\mathcal{O}^*(1.3196^K)$ *up to a factor of two in hypergraphs of maximum degree four.*

We conclude with a new polynomial-time approximation algorithm:

Algorithm 3. `3HS3-2.5-appr`: A 2.5-approximation algorithm for 3-HS3

```
1: Input: Hypergraph G = (V, E) of maximum degree three
2: Output: a hitting set C with |C| ≤ 2.5|C*|, where C* is an optimum solution
3: Initially branch on an arbitrary hyperedge (if it exists).
4: while E ≠ ∅ do
5:    Apply all reduction rules exhaustively.
6:    if the hypergraph is 3-regular then
7:       Pick a hyperedge h and put it into C.
8:    Pick a hyperedge e s.t. some neighbor x ∉ e of some vertex v ∈ e obeys deg(x) =
      2. // Irreducibility witness.
9:    Put e into the hitting set C.
10: end while
11: Return C.
```

Theorem 6. Minimum 3-Hitting Set *can be approximated in polynomial time up to a factor of* 2.5 *in hypergraphs of maximum degree three.*

Proof. 1. As long as Line 7 in Algorithm `3HS3-2.5-appr` is not executed, the algorithm works fine: It would put a hyperedge e into C, reduce the degree of x to one, so that vertex domination triggers, followed by the small edge rule on $f \setminus \{x\}$, putting in total 5 vertices in C. As at least one vertex of f and of e must be in C^*, by a local ratio argument, we achieve a factor of 2.5.
2. We can afford that the tiny edge, the small triangle and the small (non-trivial) component rule are followed by a worsening step, still staying within the promised approximation factor.
3. We claim that Line 10 either creates a new vertex of degree two (see 1.), or it triggers one of the three mentioned rules, so that Line 7 can be executed as a worsening step (see 2.). □

6 Further Questions

(1) Can we employ other forms of exact parameterized algorithms to obtain parameterized approximation algorithms, for instance, those relying on a Measure & Conquer analysis (see [12]) or on iterative compression, (see [8]). (2) Can we further improve on the running time analysis, for instance, by making use of a Measure & Conquer style analysis in the cubic case? (3) Can the techniques presented be extended to work for the weighted case, as known for moderately exponential-time approximation algorithms [5]? (4) The method that we employed for obtaining approximation algorithms is reminiscent of the well-known local ratio method [1]. This deserves further exploration. (5) Can other prominent techniques from polynomial-time approximation be employed for exponential-time approximation?

References

1. Bar-Yehuda, R.: One for the price of two: a unified approach for approximating covering problems. Algorithmica 27, 131–144 (2000)
2. Bourgeois, N., Escoffier, B., Paschos, V.T.: Efficient Approximation of Combinatorial Problems by Moderately Exponential Algorithms. In: Dehne, F., Gavrilova, M., Sack, J.-R., Tóth, C.D. (eds.) WADS 2009. LNCS, vol. 5664, pp. 507–518. Springer, Heidelberg (2009)
3. Brankovic, L., Fernau, H.: Combining Two Worlds: Parameterised Approximation for Vertex Cover. In: Cheong, O., Chwa, K.-Y., Park, K. (eds.) ISAAC 2010, Part I. LNCS, vol. 6506, pp. 390–402. Springer, Heidelberg (2010)
4. Chen, Y., Grohe, M., Grüber, M.: On Parameterized Approximability. In: Bodlaender, H.L., Langston, M.A. (eds.) IWPEC 2006. LNCS, vol. 4169, pp. 109–120. Springer, Heidelberg (2006)
5. Cygan, M., Kowalik, Ł., Wykurz, M.: Exponential-time approximation of weighted set cover. Information Processing Letters 109, 957–961 (2009)
6. Fernau, H.: Parameterized algorithmics for d-hitting set. International Journal of Computer Mathematics 87(14), 3157–3174 (2010)
7. Fernau, H.: A top-down approach to search trees: Improved algorithmics for 3-HITTING SET. Algorithmica 57, 97–118 (2010)
8. Fomin, F.V., Gaspers, S., Kratsch, D., Liedloff, M., Saurabh, S.: Iterative Compression and Exact Algorithms. In: Ochmański, E., Tyszkiewicz, J. (eds.) MFCS 2008. LNCS, vol. 5162, pp. 335–346. Springer, Heidelberg (2008)
9. Kanj, I.A., Zhang, F.: 3-HITTING SET on Bounded Degree Hypergraphs: Upper and Lower Bounds on the Kernel Size. In: Marchetti-Spaccamela, A., Segal, M. (eds.) TAPAS 2011. LNCS, vol. 6595, pp. 163–174. Springer, Heidelberg (2011)
10. Khot, S., Regev, O.: Vertex cover might be hard to approximate to within $2-\varepsilon$. Journal of Computer and System Sciences 74, 335–349 (2008)
11. Reiter, R.: A theory of diagnosis from first principles. Artificial Intelligence 32, 57–95 (1987)
12. Wahlström, M.: Algorithms, Measures and Upper Bounds for Satisfiability and Related Problems. PhD thesis, Department of Computer and Information Science, Linköpings universitet, Sweden (2007)