Proc. 35th SOFSEM, 2009

A Problem Kernelization for Graph Packing

Hannes Moser*

Institut für Informatik, Friedrich-Schiller-Universität Jena,
Ernst-Abbe-Platz 2, D-07743 Jena, Germany
moser@minet.uni-jena.de

Abstract. For a fixed connected graph H, we consider the NP-complete H-packing problem, where, given an undirected graph G and an integer $k \geq 0$, one has to decide whether there exist k vertex-disjoint copies of H in G. We give a problem kernel of $O(k^{|V(H)|-1})$ vertices, that is, we provide a polynomial-time algorithm that reduces a given instance of H-packing to an equivalent instance with at most $O(k^{|V(H)|-1})$ vertices. In particular, this result specialized to H being a triangle improves a problem kernel for Triangle Packing from $O(k^3)$ vertices by Fellows et al. [WG 2004] to $O(k^2)$ vertices.

1 Introduction

To solve NP-hard problems, polynomial-time preprocessing is a natural approach. Problem kernelization is a preprocessing technique originating from the field of parameterized algorithmics [8,23]. A *kernelization* is an algorithm that, given a problem instance I with parameter k, replaces I by another instance I' with parameter $k' \leq k$ in polynomial time, such that I with parameter k is a yes-instance if and only if I' with parameter k' is a yes-instance, and $|I'| \leq g(k)$ for some function g. The instance I' is called the *problem kernel*. Besides its theoretical significance in parameterized complexity analysis, problem kernelization has also practical applications as a preprocessing step to get smaller problem instances. For instance, for the Vertex Cover problem, a kernel with at most $2k$ vertices can be achieved [22,5], where k is the vertex cover number of the given graph. For the (undirected) Feedback Vertex Set problem, a kernel of $O(k^3)$ vertices [2] has recently been improved to $O(k^2)$ vertices [26], where k is the feedback vertex set number of the given graph. Another "success story" for kernelization is Cluster Editing. Here, a first problem kernel had $O(k^2)$ vertices [14], where k is the number of allowed editing operations. The kernelization has been gradually improved [11,25], and the best-known kernel size is now $4k$ vertices [15]. For more about kernelization we refer to a recent survey by Guo and Niedermeier [16]. In this work, we improve a bound of $108k^3 - 73k^2 - 18k$ vertices for a problem kernel for the problem of packing h vertex-disjoint triangles [9] to a bound of $45k^2$ vertices. Moreover, we generalize our approach to a problem kernel of $O(k^{|V(H)|-1})$ vertices for the problem of packing k vertex-disjoint copies of a fixed connected graph H.

* Supported by the DFG, project AREG, NI 369/9.

The H-Packing problem is defined as follows.

H-Packing
Input: An undirected graph $G = (V, E)$ and an integer $k \geq 0$.
Question: Does G contain k vertex-disjoint copies of H?

If H is simply an edge between two vertices, then this problem is equivalent to Maximum Matching, which can be solved in polynomial time. If H is a connected graph with at least three vertices, then H-Packing becomes NP-complete [18].

H-Packing can be solved in $2^{|V(H)|k}n^{O(1)}$ time with a randomized algorithm [20] and in $16^{|V(H)|k}n^{O(1)}$ time [19] with a deterministic algorithm (see also [6] for an improved derandomization). H-Packing has been studied quite intensively for small graphs H. If H is a triangle, it is known that the problem is APX-hard and there exists a factor-1.2 polynomial-time approximation on graphs with maximum degree four [21]. Furthermore, it is NP-hard to approximate within ratio 139/138 [7]. If H is a path on three vertices, there exists a kernel with $15k$ vertices [24], which has been improved to $7k$ vertices [29]. The best-known parameterized algorithm runs in $2.48^{3k}n^{O(1)}$ time [12]. Both variants (that is, H being a triangle and H being a path on three vertices) can be easily reduced to 3-Set Packing, for which the following results exist. The currently best deterministic parameterized algorithm runs in $3.52^{3k}n^{O(1)}$ time [28] and the best randomized algorithm runs in $2^{3k}n^{O(1)}$ time [20]. A weighted variant of the problem can be solved in $10.6^{3k}n^{O(1)}$ time with a deterministic algorithm and in $7.56^{3k}n^{O(1)}$ time with a randomized algorithm [27].

An interesting question in parameterized complexity theory is the existence of lower bounds for kernel sizes. Up to now, we are aware of lower bounds on the constant factor of a linear kernel (that is, of size ck for some constant c) for Vertex Cover, Independent Set, and Dominating Set on planar graphs [4]. Moreover, it is known that several problems do not admit a polynomial-size kernel [3,13] (the existence of polynomial-size kernels for these problems would imply the collapse of the polynomial hierarchy to the third level). However, to the best of our knowledge, there is no example of a problem that does admit a polynomial-size kernel, but no linear-size kernel. Fellows et al. [9] conjectured that K_t-Packing, that is, packing cliques on t vertices, might be a candidate for a problem whose kernel cannot be smaller than $O(k^t)$ vertices. However, our result for H-Packing directly shows that an $O(k^{t-1})$-vertex kernel is possible. Our technique differs from the technique used by Fellows et al. [9] in how we analyze an initially computed greedy packing of triangles based on which the size bound of the whole kernel is derived. The drawback of their method is, as they state, that it is not obvious how to generalize it to H-Packing. Our approach combines ideas from an improved kernelization of Hitting Set [1] and from problem kernels for generalized matching and set packing problems [10] to achieve this. The $O(k^3)$-vertex kernel by Fellows et al. [9] is based on *crown decompositions*; while we also apply the idea behind crown decompositions, we will see that it is actually not necessary to compute the decomposition to derive the kernel. This does not improve on the worst-case running time of the kernel-

ization, but might be interesting for practical purposes and would probably also work similarly for other applications of the crown decomposition technique.

Due to the lack of space some details are omitted.

2 Preliminaries

In this paper, all graphs are simple and undirected. For a graph $G = (V, E)$, we write $V(G)$ to denote its vertex set and $E(G)$ to denote its edge set. For a vertex set $S \subseteq V$, we write $G[S]$ to denote the graph induced by S in G, that is, $G[S] := (S, \{e \in E \mid e \subseteq S\})$. For a set of graphs $\mathcal{H}$ we define $V(\mathcal{H}) := \bigcup_{H \in \mathcal{H}} V(H)$ and $E(\mathcal{H}) := \bigcup_{H \in \mathcal{H}} E(H)$. A *triangle* ($K_3$) is the complete graph on three vertices. We say that a graph H' is a *copy of* H if H' is isomorphic to H. For a graph G and a graph H, we say that H' is a *copy of* H *in* G if H' is a subgraph of G and H' is a copy of H. Given two graphs H_1 and H_2, the *intersection* of H_1 and H_2 is defined as $V(H_1) \cap V(H_2)$.

Given a graph $G = (V, E)$, an edge subset $M \subseteq E$ is called a *matching* if the edges in M are pairwise disjoint. A matching M is *maximal* if there exists no edge $e \in (E \setminus M)$ such that $M \cup \{e\}$ is a matching. A matching M is *maximum* if there exists no larger matching. A vertex $v \in V$ is *matched* if there exists an edge in M that is incident to v. A vertex $v \in V$ is *unmatched* if it is not matched. An *M-alternating path* is a path in G, which starts with an unmatched vertex, and then contains, alternately, edges from $E \setminus M$ and M. If an M-alternating path ends with an unmatched vertex, then it is called *M-augmenting path*.

Parameterized complexity is a two-dimensional framework for studying the computational complexity of problems [8,23]. One dimension of an instance of a parameterized problem is the input size n, and the other is the *parameter* k. A parameterized problem is *fixed-parameter tractable* if it can be solved in $f(k) \cdot n^{O(1)}$ time, where f is a computable function depending only on the parameter k, not on the input size n. Problem kernelization is a core tool to develop parameterized algorithms [16,17,23]. A kernelization is often described with a set of *data reduction rules* that are applied to the instance I with parameter k of a problem and that change that instance into an smaller instance I' with parameter $k' \leq k$ in polynomial time, such that (I, k) is a yes-instance if and only if (I', k') is a yes-instance. An instance to which none of a given set of reduction rules applies is called *reduced* with respect to the rules.

Next, we show the kernelization for Triangle Packing. After that, we generalize this approach to H-Packing.

3 A Quadratic Kernel for Triangle Packing

Triangle Packing is formally defined as follows.

Triangle Packing
Input: An undirected graph $G = (V, E)$ and an integer $k \geq 0$.
Question: Does G contain k vertex-disjoint copies of K_3?

The problem kernel with $O(k^3)$ vertices by Fellows et al. [9] starts with a greedy packing $\mathcal{P}$ of triangles, which contains less than $3k$ vertices (otherwise, we already have a packing of k triangles). Then, based on the size of $\mathcal{P}$, the number of vertices in $V \setminus V(\mathcal{P})$ is bounded, which implies that the total number of vertices in the graph is bounded, yielding a problem kernel. In this sense, $\mathcal{P}$ is a witness for the number of vertices in the graph. The problem with this approach is that there is too much structure of the graph "outside" of $\mathcal{P}$. To deal with this problem, we use a different notion of witness, which contains more structure than $\mathcal{P}$, but which is still small enough in order to obtain a better bound on the number of vertices. Our kernelization is based on the same reduction rules as the kernel by Fellows et al. [9]. However, our approach applies them differently, and, most importantly, it uses a different analysis. One of the main advantages of our approach is that it is easier to generalize to H-Packing for arbitrary connected graphs H.

Our approach works with the set of all triangles in G. Since in an n-vertex graph there are at most $\binom{n}{3}$ triangles, this set can be computed in polynomial time. First, we apply the following simple data reduction rule, which is obviously correct and can be performed in polynomial time.

Reduction Rule 1 *Remove all vertices and edges that are not contained in any triangle in G.*

In the following, assume that G is reduced with respect to Reduction Rule 1. The general strategy of our kernelization algorithm is as follows. First, we compute in polynomial time a set of not necessarily disjoint triangles $\mathcal{T}$, and we show that if there are sufficiently many vertices in $V(\mathcal{T})$, then the input instance is a yes-instance, and a corresponding size-k packing can be computed. If not, then, with the size bound on $V(\mathcal{T})$, one can bound the size of $V \setminus V(\mathcal{T})$ by applying a data reduction rule based on matching techniques. In this sense, the set $\mathcal{T}$ is the basis of our kernelization and is the *witness* for the size of the kernel.

The witness $\mathcal{T}$ is defined as a maximal set of triangles in G that pairwise intersect in at most one vertex. We will later show that $I := V \setminus V(\mathcal{T})$ forms an independent set. The witness can be computed by an algorithm that starts with an empty set $\mathcal{T}$ and greedily adds a triangle T to $\mathcal{T}$ if T intersects with each triangle in $\mathcal{T}$ in at most one vertex. We call this algorithm **compute_witness**. After computing $\mathcal{T}$, the following data reduction rule due to Fellows et al. [9] is applied.

Reduction Rule 2 ([9]) *If there is a vertex $u \in V(\mathcal{T})$ such that there exist at least $3k-2$ triangles in $\mathcal{T}$ that pairwise intersect exactly in u, then delete u from G and set $k := k-1$.*

Since $\mathcal{T}$ is a set of triangles that pairwise intersect in at most one vertex, the precondition of Reduction Rule 2 can be verified in polynomial time. Intuitively, this reduction rule is correct because a packing of $k-1$ triangles in the reduced graph can "hit" at most $3k-3$ triangles that pairwise intersect exactly in u in the input graph, thus there is always at least one triangle left, which can be

added to the packing, obtaining a packing of k triangles for the input graph. If Reduction Rule 2 applies, then the kernelization algorithm restarts with exhaustively applying Reduction Rule 1 and calling **compute_witness**. This is repeated until Reduction Rule 2 does not apply or until $k = 0$. The latter means that G is a yes-instance, thus the kernelization algorithm returns "yes". In the following, we can therefore assume that Reduction Rule 1 and Reduction Rule 2 do not apply and that $k > 0$.

Lemma 1. *(1) The set $I := V \setminus V(\mathcal{T})$ forms an independent set in G and (2) each triangle that contains a vertex in I shares an edge with a triangle in $\mathcal{T}$.*

Proof. (1) Suppose that I is not an independent set in G. Let e be an edge in $G[I]$. Due to Reduction Rule 1, the edge e must be contained in a triangle $T \notin \mathcal{T}$, which intersects each triangle in $\mathcal{T}$ in at most one vertex, thus T is added to $\mathcal{T}$ by **compute_witness**, contradicting $T \notin \mathcal{T}$. (2) If a triangle $T \notin \mathcal{T}$ contains a vertex in I but shares no edge with a triangle in $\mathcal{T}$, then again T intersects each triangle in $\mathcal{T}$ in at most one vertex, contradicting $T \notin \mathcal{T}$. □

Note that the graph $G[V(\mathcal{T})]$ might contain edges that are not part of any triangle in $\mathcal{T}$; however, by Lemma 1, these edges are not contained in any triangle that contains a vertex in I. This fact is crucial to obtain a quadratic bound on the number of vertices for our problem kernel. To this end, we need to bound the number of vertices in $V(\mathcal{T})$ and the number of triangles in $\mathcal{T}$.

Lemma 2. *If $|V(\mathcal{T})| > 18k^2$ or if $|\mathcal{T}| > 9k^2$, then G contains k vertex-disjoint triangles.*

Proof. Assume that there do not exist k vertex-disjoint triangles in $\mathcal{T}$. Let $\mathcal{P} \subseteq \mathcal{T}$ be a maximum-size set of vertex-disjoint triangles. Thus, $|\mathcal{P}| \leq k - 1$, and for each triangle $T \in \mathcal{T} \setminus \mathcal{P}$ we know that $V(T) \cap V(\mathcal{P}) \neq \emptyset$. By definition of **compute_witness**, the triangles in $\mathcal{T}$ pairwise intersect in a most one vertex. For each triangle $T \in \mathcal{P}$, due to Reduction Rule 2 each vertex in T is contained in at most $3k - 3$ triangles, thus for each vertex $v \in V(T)$ we have at most $6k - 6 + 1 \leq 6k$ vertices contained in triangles that contain v. Thus, in total we have at most $3|\mathcal{P}| \cdot 6k \leq 18k^2$ vertices and at most $3|\mathcal{P}| \cdot 3k \leq 9k^2$ triangles in $\mathcal{T}$. Therefore, if $|V(\mathcal{T})| > 18k^2$ or $|\mathcal{T}| > 9k^2$, then G contains k vertex-disjoint triangles. These can be found by a greedy algorithm that selects an arbitrary triangle, deletes all other intersecting triangles, and proceeds recursively with the remaining instance until it has found k triangles. □

Thus, our kernelization algorithm outputs "yes" if one of the conditions of Lemma 2 applies. If this is not the case, then it remains to upper-bound the size of I. To this end, we define an auxiliary bipartite graph $G_{\mathcal{T}}$ as follows. The vertex set consists of I as one partite set and $J := \{v_e \mid e \in E(\mathcal{T})\}$ as the other, and $G_{\mathcal{T}}$ contains an edge $\{u, v_e\}$ if $\{u\} \cup e$ induces a triangle in G. Note that by part (2) of Lemma 1 every triangle containing a vertex in I is "represented" by an edge in $G_{\mathcal{T}}$. See Figure 1 for an example. With the help of this auxiliary graph, we can state a data reduction rule to upper-bound the size of I.

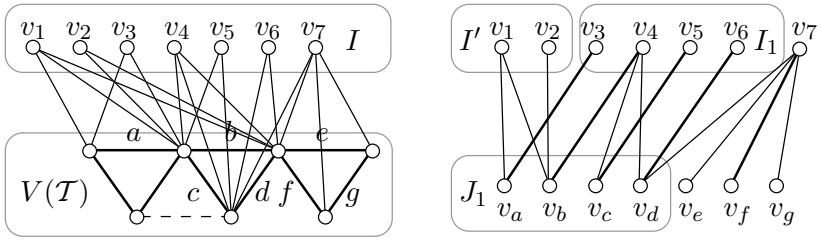

Fig. 1: Left: Graph with witness $\mathcal{T}$. The edge set $E(\mathcal{T})$ of the witness is drawn bold. The dashed edge is not contained in any triangle that contains a vertex in I nor in any triangle in the witness. Right: Corresponding auxiliary graph. Note that degree-0 vertices (corresponding to unlabeled edges of $G[V(\mathcal{T})]$) are not drawn. The bold edges are in a maximum matching. For the definition of the vertex sets see the proof of Lemma 3.

Reduction Rule 3 *Compute a maximum matching in $G_{\mathcal{T}}$. Remove all unmatched vertices in I from G.*

Lemma 3. *Reduction Rule 3 is correct, that is, G has a size-k packing of triangles if and only if the graph resulting by removing all unmatched vertices in I from G has a size-k packing of triangles.*

Proof. Let M be the computed maximum matching in $G_{\mathcal{T}}$ and let I' be all unmatched vertices in I (see Figure 1). Since M is maximum, the graph $G_{\mathcal{T}}$ contains no M-augmenting path. We have to show that G contains k vertex-disjoint triangles if and only if $G[V \setminus I']$ contains k vertex-disjoint triangles.

($\Leftarrow$) This direction is trivial, since a set of k vertex-disjoint triangles in $G[V \setminus I']$ is also contained in G.

($\Rightarrow$) Let $\mathcal{P}$ be a set of k vertex-disjoint triangles in G. If no triangle in $\mathcal{P}$ contains a vertex of I', then $\mathcal{P}$ is a set of k vertex-disjoint triangles in $G[V \setminus I']$. Therefore, suppose that there is a triangle in $\mathcal{P}$ that contains a vertex of I'. We show in the following that we can always modify $\mathcal{P}$ such that there is no triangle containing a vertex of I'.

Let $I_1 \subseteq I \setminus I'$ be the set of vertices in $I \setminus I'$ to which there exists an M-alternating path from some vertex in I' (see Figure 1). Each vertex $u \in I_1$ is an endpoint of an edge in M because there is an M-alternating path from some vertex $w \in I'$ to u, and the path begins with an edge that is not contained in M (since all vertices in I' are unmatched). Let $M' \subseteq M$ be the matching edges that have an endpoint in I_1, and let $J_1 := J \cap V(M')$ be the corresponding other endpoints of M' (see Figure 1). We claim that every triangle that contains a vertex in $I' \cup I_1$ contains an edge e corresponding to a vertex $v_e \in J_1$.

To show the claim, let T be a triangle in $\mathcal{P}$ that contains a vertex $u \in I'$. Suppose that T contains an edge e corresponding to a vertex v_e in $J \setminus J_1$. Since $v_e \notin J_1$, we know that v_e is not matched by M; otherwise, there would

be an M-alternating path (u, v_e, w) for some vertex $w \in I \setminus I'$, and this would imply $v_e \in J_1$ by the definition of J_1. Therefore, $\{u, v_e\}$ could be added to M, contradicting that M is maximum. Similarly, every triangle T in $\mathcal{P}$ that contains a vertex $u \in I_1$ contains an edge e corresponding to some $v_e \in J_1$. To see this, assume again that $v_e \in J \setminus J_1$. Then, v_e must be unmatched, but then the path in $G_{\mathcal{T}}$ consisting of the M-alternating path from some vertex $w \in I'$ to u and the edge $\{u, v_e\}$ forms an M-augmenting path, contradicting that M is maximum. This shows the claim.

Since M' is a perfect matching between I_1 and J_1 (that is, every vertex in $I_1 \cup J_1$ is matched and every matching edge has one endpoint from I_1 and the other from J_1), we can always replace all triangles in $\mathcal{P}$ that contain vertices in $I' \cup I_1$ by the same number of triangles containing only vertices in I_1. This shows that $G[V \setminus I']$ also contains k vertex-disjoint triangles. □

Lemma 4. *After applying Reduction Rule 3, at most $27k^2$ vertices of I remain.*

Proof. By Lemma 2, the witness $\mathcal{T}$ computed by **compute_witness** contains at most $9k^2$ triangles. Since $J := \{v_e \mid e \in E(\mathcal{T})\}$, we know that $|J| \leq 27k^2$. Due to Reduction Rule 3, all remaining vertices of I are matched by a maximum matching between I and J in $G_{\mathcal{T}}$. Therefore, there remain at most $27k^2$ vertices of I. □

Theorem 1. Triangle Packing *has a problem kernel with at most $45k^2$ vertices.*

Proof. By Lemma 2, we have at most $18k^2$ vertices in $V(\mathcal{T})$ and, by Lemma 4, there remain at most $27k^2$ vertices of I, thus in total we have at most $45k^2$ vertices. It is easy to verify that all the steps of the kernelization can be performed in polynomial time. □

4 Kernelization for H-Packing

In this section, we generalize the kernelization approach for Triangle Packing to H-Packing for an arbitrary connected graph H. The main difference to Triangle Packing is a new reduction rule that bounds the size of the witness and generalizes Reduction Rule 2. Let h denote the number of vertices in H. Note that h is a constant. We start with a trivial reduction rule.

Reduction Rule 4 *Remove all vertices and edges that are not contained in any copy of H in G.*

Lemma 5. *Reduction Rule 4 is correct and can be performed in polynomial time.*

Proof. The correctness is trivial. By looking at the at most $\binom{n}{h}$ many copies of H in the input graph and marking all vertices and edges that are contained in some copy, the vertices and edges not contained in any copy can be found in polynomial time. □

In the following, we assume that G is reduced with respect to Reduction Rule 4. Let $\mathcal{H}$ be the set of all copies of H in G. Due to Reduction Rule 4, every vertex in G is contained in at least one copy of H in $\mathcal{H}$. The set $\mathcal{H}$ can be computed in polynomial time by simply trying all $\binom{n}{h}$ vertex subsets. As for Triangle Packing, we define a witness. The witness definition for H-Packing is slightly more complicated. A witness has to be defined with respect to a subset of $\mathcal{H}$, since in the course of the witness computation some elements of $\mathcal{H}$ will be removed.

Definition 1. *Let $\mathcal{H}$ be the set of all copies of H in G. A* witness *with respect to a set $\mathcal{H}' \subseteq \mathcal{H}$ for H-Packing is a maximal subset $\mathcal{W} \subseteq \mathcal{H}'$ such that the copies of H in $\mathcal{W}$ pairwise intersect in at most $h-2$ vertices.*

The algorithm **compute_witness**, given in Figure 2, computes a witness $\mathcal{W}$ with respect to $\mathcal{H} \setminus \mathcal{R}$, where $\mathcal{R}$ is a set of "unnecessary" copies of H in $\mathcal{H}$, that is, if there exists a size-k H-packing, then there is a size-k H-packing that does not use any element of $\mathcal{R}$. The identification of unnecessary copies of H is derived from a combination of ideas for data reduction rules for Hitting Set [1] and generalized matching and set cover problems [10]. The basic idea is that if there are many copies of H in $\mathcal{W}$ that intersect in the same vertex subset S, then some of them do not need to be considered for a maximum H-packing in G and can therefore be removed from the graph. The algorithm uses an iterative approach, which starts with empty sets $\mathcal{R}$ and $\mathcal{W}$. In line 3 of Figure 2, it computes a witness $\mathcal{W}$ with respect to $\mathcal{H} \setminus \mathcal{R}$. This can be done in polynomial time by an iterative approach that adds an element H' from $(\mathcal{H} \setminus \mathcal{R}) \setminus \mathcal{W}$ to $\mathcal{W}$ if H' intersects with each element in $\mathcal{W}$ in at most $h-2$ vertices. In lines 5–11, the algorithm identifies unnecessary copies and adds them to $\mathcal{R}$; the correctness of this part will be shown with Lemma 6. After having identified and removed unnecessary copies of H from $\mathcal{W}$, the set $\mathcal{W}$ might not be a witness with respect to $\mathcal{H} \setminus \mathcal{R}$; therefore, the algorithm repeats until no more unnecessary copies of H can be found. Then, the resulting set $\mathcal{W}$ is a witness with respect to $\mathcal{H} \setminus \mathcal{R}$ due to line 3.

Let $\mathcal{W}$ be the witness that is returned by **compute_witness**$(\mathcal{H})$. The following lemma shows that one can remove the copies of H in $\mathcal{C}'$ in line 11 of **compute_witness** without changing the size of a maximum H-packing in G. After executing **compute_witness**, the set $\mathcal{R}$ contains all the removed copies of H.

Lemma 6. *If there exists an H-packing $\mathcal{P}$ of size k in G, then there exists an H-packing $\mathcal{P}'$ of size k in G that does not contain any element of $\mathcal{R}$.*

Proof. If $\mathcal{R} \cap \mathcal{P} = \emptyset$, then $\mathcal{P}' := \mathcal{P}$. Otherwise, we show that we can replace each copy of H in $\mathcal{R} \cap \mathcal{P}$ by another copy of H not contained in $\mathcal{R}$.

We show the claim by induction on i (line 5 in Figure 2). Intuitively, i determines the size of the set S computed in line 7; for $i = 0$, S contains $h-2$ vertices, thus all copies of H whose vertex sets are supersets of S intersect exactly in S due to Definition 1, and the number of these copies can be bounded easily. For $i > 0$, the set S contains less than $h-2$ vertices, and the copies of H whose vertex sets are supersets of S might also intersect outside of S, but then we can bound their number based on the induction hypothesis.

Algorithm: compute_witness ($\mathcal{H}$)
Input: A set $\mathcal{H}$ of copies of H in G.
Output: A set $\mathcal{R}$ of unnecessary copies of H and a witness $\mathcal{W}$ with respect to $\mathcal{H} \setminus \mathcal{R}$.

```
1  R ← ∅; W ← ∅
2  repeat
3      Greedily add elements from H \ (W ∪ R) to W such that W is a witness.
4      C' ← ∅
5      for i ← 0 to h − 3 do
6          for each H' ∈ W do
7              for each S ⊊ V(H'), |S| = h − 2 − i do
8                  C ← {H'' ∈ W | V(H'') ⊋ S}
9                  if |C| > Σ_{t=0}^{i+1} (h · (k − 1))^t then
10                     choose any set C' ⊊ C of size |C| − Σ_{t=0}^{i+1} (h · (k − 1))^t.
11                     W ← W \ C'; R ← R ∪ C'
12 until C' = ∅
13 return W, R
```

Fig. 2: Pseudo-code of the algorithm to compute the witness $\mathcal{W}$.

Let $i = 0$ and let S be a size-$(h-2)$ vertex subset such that $|\mathcal{C}| > 1+h\cdot(k-1)$ (line 9), and let $\mathcal{C}' \subseteq \mathcal{C}$ be as in line 10. Clearly, $|\mathcal{C} \setminus \mathcal{C}'| = 1 + h \cdot (k-1)$. Since the copies of H in $\mathcal{C}$ pairwise intersect (due to the construction of $\mathcal{C}$ in line 8), at most one of them can be in $\mathcal{P}$. Let H_1 be that copy and assume that $H_1 \in \mathcal{C}'$. The remaining $k-1$ copies of H in $\mathcal{P} \setminus \{H_1\}$ can intersect with at most $h \cdot (k-1)$ copies of H in $\mathcal{C} \setminus \mathcal{C}'$, since the copies of H in $\mathcal{C}$ pairwise intersect exactly in S (because $\mathcal{W}$ is a maximal set of copies of H that pairwise intersect in at most $h-2$ vertices and $|S| = h-2$). Therefore, there is at least one $H_2 \in \mathcal{C} \setminus \mathcal{C}'$ such that $V(H_2) \cap V(\mathcal{P}) = V(H_2) \cap V(H_1) = S$. We remove H_1 from $\mathcal{P}$ and add H_2 to it. As a consequence, $\mathcal{P}$ contains no copy of H from $\mathcal{C}'$.

For $i > 0$, let S be a size-$(h-2-i)$ vertex subset such that $|\mathcal{C}| > \sum_{t=0}^{i+1}(h \cdot (k-1))^t$ (line 9), and let $\mathcal{C}' \subseteq \mathcal{C}$ be as in line 10. Again, we may assume that there exists an $H_1 \in \mathcal{P} \cap \mathcal{C}'$. We count the number of copies of H in $\mathcal{C} \setminus \mathcal{C}'$ that can intersect with $\mathcal{P} \setminus \{H_1\}$. Let $W := V(\mathcal{P} \setminus \{H_1\})$. Obviously, $|W| \leq h \cdot (k-1)$. Each vertex $v \in W$ can "hit" at most $\sum_{t=0}^{i}(h \cdot (k-1))^t$ copies of H in $\mathcal{C} \setminus \mathcal{C}'$, since by the induction hypothesis, there are at most $\sum_{t=0}^{i}(h \cdot (k-1))^t$ copies of H whose vertex sets are supersets of $S \cup \{v\}$. Therefore, at most $h \cdot (k-1) \sum_{t=0}^{i}(h \cdot (k-1))^t = \sum_{t=1}^{i+1}(h \cdot (k-1))^t$ copies of H in $\mathcal{C} \setminus \mathcal{C}'$ intersect with $\mathcal{P} \setminus \{H_1\}$, and thus there is at least one left in order to replace H_1 with (recall that $|\mathcal{C} \setminus \mathcal{C}'| = \sum_{t=0}^{i+1}(h \cdot (k-1))^t$).

Thus, eventually we obtain a set $\mathcal{P}'$ of k vertex-disjoint copies of H such that $\mathcal{P}' \cap \mathcal{R} = \emptyset$. □

Lemma 7. *Algorithm* ***compute_witness*** *runs in polynomial time.*

With the help of **compute_witness** we can state the following reduction rule.

Reduction Rule 5 *Run* ***compute_witness*** *to get a witness $\mathcal{W}$ and the set $\mathcal{R}$. Then, replace G by $G[V(\mathcal{H} \setminus \mathcal{R})]$.*

Lemma 8. *Reduction Rule 5 is correct, that is, G has a size-k H-packing if and only if $G[V(\mathcal{H} \setminus \mathcal{R})]$ has a size-k H-packing.*

In the following, we assume that the graph G is reduced with respect to Reduction Rule 4 and Reduction Rule 5, that $\mathcal{H}$ is the set of all copies of H in G, and that $\mathcal{W}$ is a witness with respect to $\mathcal{H}$.

Lemma 9. *(1) The set $I := V \setminus V(\mathcal{W})$ forms an independent set in G and (2) each copy of H in $\mathcal{H} \setminus \mathcal{W}$ contains a vertex in I and $h-1$ vertices of some copy of H in $\mathcal{W}$.*

Proof. Similar to the proof of Lemma 1. (1) If $G[I]$ contains an edge, which has to be part of some copy of H due to Reduction Rule 4, then at most $h-2$ vertices of that copy intersect with each $H \in \mathcal{W}$, contradicting the fact that $\mathcal{W}$ is maximal. (2) If a copy of H in $\mathcal{H} \setminus \mathcal{W}$ shares at most $h-2$ vertices with each copy of H in $\mathcal{W}$, then we again have a contradiction to the fact that $\mathcal{W}$ is maximal. □

It follows directly from Lemma 9 that each copy of H contains at most one vertex of I. Now, analogously to Triangle Packing, we bound the number of vertices in $V(\mathcal{W})$ and the number of copies of H in $\mathcal{W}$.

Lemma 10. *If $|V(\mathcal{W})| > 2h(h\cdot(k-1))^{h-1}$ or if $|\mathcal{W}| > 2(h\cdot(k-1))^{h-1}$, then G contains k vertex-disjoint copies of H.*

Proof. We use the same proof strategy as in the proof of Lemma 2. Assume that G does not contain a size-k H-packing. Let $\mathcal{P}$ be an H-packing of maximum size. Since $|\mathcal{P}| \leq k-1$, there are at most $h \cdot (k-1)$ vertices in $V(\mathcal{P})$. Each of these vertices is contained in at most $\sum_{t=0}^{h-2}(h\cdot(k-1))^t \leq 2(h\cdot(k-1))^{h-2}$ copies of H (geometric series, with $h \geq 3$ and $k \geq 2$; recall that for $h \leq 2$ H-Packing is polynomial-time solvable and $k = 1$ implies $\mathcal{P} = \emptyset$, and therefore the graph cannot contain any triangle, thus $|\mathcal{W}| = 0$), since for $i = h-3$ each set S in line 7 of **compute_witness** (Figure 2) contains one vertex, and the number of copies of H containing S directly follows from the condition in line 9. Hence, $|\mathcal{W}| \leq 2(h\cdot(k-1))^{h-1}$ and $|V(\mathcal{W})| \leq 2h(h\cdot(k-1))^{h-1}$. □

It remains to bound the size of $I := V \setminus V(\mathcal{W})$. To this end, as for Triangle Packing, we define a bipartite auxiliary graph $G_{\mathcal{W}}$ as follows. The vertex set consists of I as one partite set and a set J as the other, where J contains a vertex v_X for each set $X \in \{V(H) \cap V(\mathcal{W}) \mid H \in (\mathcal{H} \setminus \mathcal{W})\}$ (that is, we have a vertex for each possible intersection of the copies of H in $\mathcal{H} \setminus \mathcal{W}$ with the vertex set $V(\mathcal{W})$). For each $H \in (\mathcal{H} \setminus \mathcal{W})$ there is an edge between the vertex in the set $V(H) \cap I$ and the vertex v_X with $X := V(H) \cap V(\mathcal{W})$. Note that for

each $H' \in \mathcal{W}$ there are at most h sets $X \subset V(H')$ in $\{V(H) \cap V(\mathcal{W}) \mid H \in (\mathcal{H} \setminus \mathcal{W})\}$, and therefore the size bound of $\mathcal{W}$ from Lemma 10 together with Lemma 9 part (2) yields $|J| < 2h(h \cdot (k-1))^{h-1}$. The size of the independent set I then can be bounded exactly as for Triangle Packing with the following reduction rule; the proof of correctness is almost the same as the proof of Lemma 3 (replacing $G_{\mathcal{T}}$ with $G_{\mathcal{W}}$ and "triangle" with "copy of H").

Reduction Rule 6 *Compute a maximum matching in $G_{\mathcal{W}}$. Remove all unmatched vertices in I from G.*

The number of remaining vertices then can be bounded by the size of J, that is, there are at most $2h(h \cdot (k-1))^{h-1}$ vertices of I remaining. Together with the at most $2h(h \cdot (k-1))^{h-1}$ vertices in $V(\mathcal{W})$ (Lemma 10), we obtain the following.

Theorem 2. *H-Packing has a problem kernel with $O(k^{|H|-1})$ vertices.*

Further Remarks. We believe that our approach also works for Set Packing. However, this only gives a better kernel with respect to the number of elements, not with respect to the number of sets and would therefore not improve the known kernelization results [10]. One of the key ingredients for obtaining a kernel with $O(k^{|H|-1})$ vertices instead of $O(k^{|H|})$ vertices is the matching technique to bound the number of vertices in the remaining independent set. It would be interesting to know whether it is possible to bound structures different from independent sets by similar techniques. This way, a witness with less vertices and edges could be possible, which could make a better kernel size possible.

Acknowledgements. I thank Jiong Guo (Jena) for inspiring discussions and Rolf Niedermeier (Jena) for helpful comments improving the presentation.

References

1. F. N. Abu-Khzam. Kernelization algorithms for d-Hitting Set problems. In *Proc. 10th WADS*, volume 4619 of *LNCS*, pages 434–445. Springer, 2007.
2. H. L. Bodlaender. A cubic kernel for feedback vertex set. In *Proc. 24th STACS*, volume 4393 of *LNCS*, pages 320–331. Springer, 2007.
3. H. L. Bodlaender, R. G. Downey, M. R. Fellows, and D. Hermelin. On problems without polynomial kernels. In *Proc. 35th ICALP*, volume 5125 of *LNCS*, pages 563–574. Springer, 2008.
4. J. Chen, H. Fernau, I. A. Kanj, and G. Xia. Parametric duality and kernelization: Lower bounds and upper bounds on kernel size. *SIAM J. Comput.*, 37(4):1077–1106, 2007.
5. J. Chen, I. A. Kanj, and W. Jia. Vertex cover: Further observations and further improvements. *J. Algorithms*, 41(2):280–301, 2001.
6. J. Chen, S. Lu, S.-H. Sze, and F. Zhang. Improved algorithms for path, matching, and packing problems. In *Proc. 18th SODA*, pages 298–307. ACM/SIAM, 2007.
7. M. Chlebík and J. Chlebíková. Approximation hardness for small occurrence instances of NP-hard problems. In *Proc. 5th CIAC*, volume 2653 of *LNCS*, pages 152–164. Springer, 2003.

8. R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer, 1999.
9. M. R. Fellows, P. Heggernes, F. A. Rosamond, C. Sloper, and J. A. Telle. Finding k disjoint triangles in an arbitrary graph. In *Proc. 30th WG*, volume 3353 of *LNCS*, pages 235–244. Springer, 2004.
10. M. R. Fellows, C. Knauer, N. Nishimura, P. Ragde, F. A. Rosamond, U. Stege, D. M. Thilikos, and S. Whitesides. Faster fixed-parameter tractable algorithms for matching and packing problems. *Algorithmica*, 52(2):167–176, 2007.
11. M. R. Fellows, M. A. Langston, F. A. Rosamond, and P. Shaw. Efficient parameterized preprocessing for Cluster Editing. In *Proc. 16th FCT*, volume 4639 of *LNCS*, pages 312–321. Springer, 2007.
12. H. Fernau and D. Raible. A parameterized perspective on packing paths of length two. In *Proc. 2nd COCOA*, volume 5165 of *LNCS*, pages 54–63. Springer, 2008.
13. L. Fortnow and R. Santhanam. Infeasibility of instance compression and succinct PCPs for NP. In *Proc. 40th STOC*, pages 133–142. ACM Press, 2008.
14. J. Gramm, J. Guo, F. Hüffner, and R. Niedermeier. Graph-modeled data clustering: Exact algorithms for clique generation. *Theory Comput. Syst.*, 38(4):373–392, 2005.
15. J. Guo. A more effective linear kernelization for Cluster Editing. In *Proc. 1st ESCAPE*, volume 4614 of *LNCS*, pages 36–47. Springer, 2007.
16. J. Guo and R. Niedermeier. Invitation to data reduction and problem kernelization. *ACM SIGACT News*, 38(1):31–45, 2007.
17. F. Hüffner, R. Niedermeier, and S. Wernicke. Techniques for practical fixed-parameter algorithms. *The Computer Journal*, 51(1):7–25, 2008.
18. D. G. Kirkpatrick and P. Hell. On the completeness of a generalized matching problem. In *Proc. 10th STOC*, pages 240–245. ACM Press, 1978.
19. J. Kneis, D. Mölle, S. Richter, and P. Rossmanith. Divide-and-color. In *Proc. 32nd WG*, volume 4271 of *LNCS*, pages 58–67. Springer, 2006.
20. I. Koutis. Faster algebraic algorithms for path and packing problems. In *Proc. 35th ICALP*, volume 5125 of *LNCS*, pages 575–586. Springer, 2008.
21. G. Manic and Y. Wakabayashi. Packing triangles in low degree graphs and indifference graphs. *Discrete Math.*, 308(8):1455–1471, 2008.
22. G. L. Nemhauser and L. E. Trotter. Vertex packings: Structural properties and algorithms. *Math. Program.*, 8:232–248, 1975.
23. R. Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, 2006.
24. E. Prieto and C. Sloper. Looking at the stars. *Theor. Comput. Sci.*, 351(3):437–445, 2006.
25. F. Protti, M. D. da Silva, and J. L. Szwarcfiter. Applying modular decomposition to parameterized cluster editing problems. *Theory Comput. Syst.*, 2008.
26. S. Thomassé. A quadratic kernel for feedback vertex set. In *Proc. 20th SODA*. ACM/SIAM, 2009. To appear.
27. J. Wang and Q. Feng. Improved parameterized algorithms for weighted 3-set packing. In *Proc. 14th COCOON*, volume 5092 of *LNCS*, pages 130–139. Springer, 2008.
28. J. Wang and Q. Feng. An $O^*(3.523k)$ parameterized algorithm for 3-set packing. In *Proc. 5th TAMC*, volume 4978 of *LNCS*, pages 82–93. Springer, 2008.
29. J. Wang, D. Ning, Q. Feng, and J. Chen. An improved parameterized algorithm for a generalized matching problem. In *Proc. 5th TAMC*, volume 4978 of *LNCS*, pages 212–222. Springer, 2008.