

# Looking at the Stars

Elena Prieto<sup>1</sup>      Christian Sloper<sup>2</sup>

<sup>1</sup> School of Electrical Engineering and Computer Science,  
The University of Newcastle  
NSW, Australia  
elena@cs.newcastle.edu.au

<sup>2</sup> Department of Informatics,  
University of Bergen  
Norway  
sloper@ii.uib.no

**Abstract.** The problem of packing  $k$  vertex-disjoint copies of a graph  $H$  into another graph  $G$  is NP-complete if  $H$  has more than two vertices in some connected component. In the framework of parameterized complexity we analyze a particular family of instances of this problem, namely the packing of stars. We give a quadratic kernel for packing  $k$  copies of  $H = K_{1,s}$ . When we consider the special case of  $s = 2$ , i.e.  $H$  being a star with two leaves, we give a linear kernel and an algorithm running in time  $\mathcal{O}(2^{5.301k} k^{2.5} + n^3)$ .

## 1 Introduction

The problem of MAXIMUM  $H$ -MATCHING, also called MAXIMUM  $H$ -PACKING, is of practical interest in the areas of scheduling [BM02], wireless sensor tracking [BK01], wiring-board design and code optimization [HK78] and many others.

The problem is defined as follows: Let  $G = (V, E)$  be a graph and  $H = (V_H, E_H)$  be a fixed graph with at least three vertices in some connected component. An  $H$ -packing for  $G$  is a collection of disjoint subgraphs of  $G$ , each isomorphic to  $H$ . In an optimization sense, the problem that we want to solve would be to find the maximum number of vertex disjoint copies of  $H$  in  $G$ . The problem is NP-complete [HK78] when the graph  $H$  has at least three vertices in some connected component. Note that in the case where  $H$  is the complete graph on two nodes  $H$ -packing is the very well studied (and polynomial time solvable) problem MAXIMUM MATCHING. MAXIMUM  $H$ -PACKING has been thoroughly studied in terms of approximation. The problem has been proved to be MAX-SNP-complete [K94] and approximable within  $|V_H|/2 + \varepsilon$  for any  $\varepsilon > 0$  [HS89]. Several restrictions have also been considered (planar graphs, unit disk graphs etc.) in terms of the complexity of their approximation algorithms. For a review of these we refer the reader to [AC99].

A recent result by [FHRST04] gives a general algorithm for packing an arbitrary graph  $H$  into  $G$ . Their result gives a  $2^{\mathcal{O}(|H|k \log k + k|H| \log |H|)}$  algorithm for the general case, where  $k$  is the number of copies of  $H$ . It should also be noted that it is possible to achieve a single exponential running time for this problem by adapting a result by Alon, Yuster and Zwick in [AYZ95].

**Theorem 1.** (Alon, Yuster, Zwick) *Let  $S$  be a directed or undirected graph on  $k$  vertices with treewidth  $t$ . Let  $G = (V, E)$  be a (directed or undirected) graph. A subgraph of  $G$  isomorphic to  $S$ , if one exists, can be found in  $2^{\mathcal{O}(k)} |V|^{t+1}$  expected time and in  $2^{\mathcal{O}(k)} |V|^{t+1} \log |V|$  worst case time.*

It is easy to see how to apply this problem to packing a graph  $H$ . Let the graph  $S$  in the above theorem be  $k$  copies of a graph  $H$ . Since  $S$  has treewidth at most  $|H|$ , we have a  $2^{\mathcal{O}(k)} |V|^{|H|+1}$

algorithm for the problem. Unfortunately the running time obtained by Alon et al. [AYZ95] hides a considerable constant in the exponent making this algorithm infeasible in practical terms.

We discuss the parameterized complexity of the MAXIMUM  $H$ -PACKING problem for the case when  $H$  belongs to the restricted family of graphs  $\mathcal{F} = K_{1,s}$ , a star with  $s$  leaves. More formally:

$K_{1,s}$ -PACKING

INSTANCE: Graph  $G = (V, E)$ , a positive integer  $k$

QUESTION: Are there at least  $k$  vertex disjoint instances of  $K_{1,s}$  in  $G$ ?

This problem has already been studied within the framework of classical complexity theory [HK86]. In their paper, Hell and Kirkpatrick studied the complexity of packing complete bipartite graphs into general graphs. We include a brief introduction to this topic in Section 2. In Section 3 we show that the general problem is tractable if parameterized, and that we can obtain a quadratic kernel. In Section 4 we show that the special case of packing  $K_{1,2}$ 's has a linear kernel, and in Section 5 we give a quick algorithm for both the general and special case. In contrast [FHRST04] obtains only a  $\mathcal{O}(k^3)$  for packing a graph with three vertices, namely  $K_3$ .

## 2 An Introduction to parameterized algorithms

A problem with main input  $x$  and parameter  $k$  is said to be fixed parameter tractable if there is an algorithm with running time  $\mathcal{O}(f(k)|x|^{\mathcal{O}(1)})$ , where  $f$  is an arbitrary function. In [F03] Mike Fellows presents a two-sided view of research on parameterized problems which he dubbed 'the two races'. Firstly, that it is interesting to obtain better running time for fixed parameter tractable problems, but also that it is also of interest to improve the size of the *kernel* even if this does not immediately lead to an improvement in running time.

**Definition 1.** *A parameterized problem  $L$  is kernelizable if there is a parametric transformation of  $L$  to itself that satisfies:*

1. *The running time of the transformation of  $(x, k)$  into  $(x', k')$ , where  $|x| = n$ , is bounded by a function  $f(k)n^{\mathcal{O}(1)}$ ,*
2.  *$k' \leq k$ , and*
3.  *$|x'| \leq h(k')$ , where  $h$  is an arbitrary function.*

Obviously the two views are not independent as improvements in the latter could give improvements in the first, but it is also important to note the following result by [DFS99], which gives a stronger link between the two races:

**Lemma 1.** *A parameterized problem  $L$  is in FPT if and only if it is kernelizable.*

The two races are worth playing as they may lead to substantial improvements on the quality of the algorithms we design and also to new strategies for practical implementations of these algorithms.

### 2.1 Preliminaries

We assume simple, undirected, connected graphs  $G = (V, E)$  where  $|V| = n$ . The neighbors of a vertex  $v$  are denoted as the set  $N(v)$ , and the neighbors of a set  $S \subseteq V$ ,  $N(S) = \bigcup_{v \in S} N(v) \setminus S$ . If  $J$  is a collection of graphs, then  $V(J)$  is the set vertices in the graphs in  $J$ .

The induced subgraph of  $S \subseteq V$  is denoted  $G[S]$ .

We use the simpler  $G \setminus v$  to denote  $G[V \setminus \{v\}]$  for a vertex  $v$  and  $G \setminus e$  to denote  $G = (V, E \setminus \{e\})$  for an edge  $e$ . Likewise  $G \setminus V'$  denotes  $G[V \setminus V']$  and  $G \setminus E'$  denotes  $G = (V, E \setminus E')$  where  $V'$  is a set of vertices and  $E'$  is a set of edges.

We say that  $K_{1,s}$  is a  $s$ -star or a star of size  $s$ .  $P_i$  denotes a path of  $i + 1$  vertices and  $i$  edges.

### 3 Parameterized complexity of STAR PACKING

In this section we prove a series of polynomial time preprocessing rules (reduction rules) and eventually show that we can obtain a kernel of quadratic size on the parameter  $k$  for the parameterized version of  $K_{1,s}$ -packing.

We use the following natural parametrization of  $K_{1,s}$ -PACKING:

$k$ - $K_{1,s}$ -PACKING

INSTANCE: Graph  $G = (V, E)$

PARAMETER:  $k$

QUESTION: Are there  $k$  vertex disjoint instances of  $K_{1,s}$  in  $G$ ?

To remove vertices of high degree and remove useless edges between vertices of low degree we introduce the following reduction rules.

**Lemma 2.** *Let  $G$  be a graph such that there exists  $v \in V, \deg(v) > k(s + 1) - 1$ .  $G$  has a  $k$ - $K_{1,s}$ -packing if and only if  $G' = G \setminus v$  has a  $(k - 1)$ - $K_{1,s}$ -packing.*

*Proof.* If  $G$  has a  $k$ - $K_{1,s}$ -packing then it is obvious that  $G'$  has a  $(k - 1)$ - $K_{1,s}$  as  $v$  cannot participate in two different stars.

If  $G'$  has a  $(k - 1)$ - $K_{1,s}$ -packing we can create a  $k$ - $K_{1,s}$ -packing by adding  $v$ . The  $k - 1$  stars already packed cannot use more than  $(s + 1)(k - 1)$  of  $v$ 's neighbors, leaving  $s$  vertices for  $v$  to form a new star.  $\square$

**Lemma 3.** *Let  $G$  be a graph where there exists  $u, v \in V(G), uv \in E(G)$  and  $\deg(u) \leq \deg(v) < s$ .  $G$  has a  $k$ -packing if and only if  $G' = G \setminus uv$  contains a  $k$ -packing.*

*Proof.* If  $G$  has a  $k$ - $K_{1,s}$ -packing then it is obvious that  $G'$  has a  $k$ - $K_{1,s}$ -packing as  $uv$  can never participate in a  $K_{1,s}$ .

If  $G'$  has a  $k$ - $K_{1,s}$ -packing it is obvious that  $G$  has a  $k$ - $K_{1,s}$ -packing as well.  $\square$

To give a quadratic kernel for the fixed parameter version of  $k$ -STAR PACKING we will use a new technique first seen in [FM+00]. This technique borrows ideas from extremal graph theory. We will show that any graph where lemmas `refmaxdeg` and `3` no longer apply is either 'small' (less than a function  $g(k)$  vertices) or has a  $k$ - $K_{1,s}$ -PACKING. We do this by studying a 'border'-line graph  $G$ , a graph with a  $k$ - $K_{1,s}$ -packing, but no  $(k + 1)$ - $K_{1,s}$ -packing. This allows us to make claims about the structure of  $G$  and finally to prove a bound on  $|V(G)|$ .

Let a graph be *reduced* when lemmas 2 and 3 are no longer applicable. In this sense both these lemmas will be commonly referred to as *reduction rules*. As an additional reduction rule we delete vertices of degree 0, as they never participate in any star.

**Lemma 4.** (*Boundary Lemma*)

If a graph instance  $(G, k)$  is reduced and has a  $k$ - $K_{1,s}$ -packing, but no  $(k + 1)$ - $K_{1,s}$ -packing then  $|V(G)| \leq k(s^3 + ks^2 + ks + 1)$ .

*Proof.* Assume there exists a counterexample  $G$ , such that  $G$  is reduced and contains a  $k$ - $K_{1,s}$ -packing  $W$ , but no  $(k + 1)$ - $K_{1,s}$ -packing and size  $|V(G)| > k(s^3 + ks^2 + ks + 1)$ .

Let  $Q$  be  $V \setminus W$ . Let  $Q_i$  be the vertices in  $Q$  that have degree  $i$  in the subgraph induced by  $Q$ . We will now prove a series of claims that bound the number of vertices in  $Q$ .

*Claim 1.*  $\forall i \geq s, Q_i = \emptyset$

*Proof of Claim 1.* Otherwise  $W$  could not be maximal. □

*Claim 2.* A  $K_{1,s}$ -star  $S \in W$  has at most  $s^2 + k(s + 1) - 1$  neighbors in  $Q$ .

*Claim 3.*  $W$  has at most  $k \cdot (s^2 + k(s + 1) - 1)$  neighbors in  $Q$ .

This follows from Claim 2.

Let  $R = V \setminus (W \cup N(W))$  i.e. the set of vertices of  $Q$  which do not have neighbors in  $W$ .

*Claim 4.*  $R$  is an independent set in  $G$ .

Claim 4 ensures us that all vertices in  $R$  have an edge to one or more vertex in  $Q$ . By Claim 1 we know that each of the vertices in  $Q \setminus R$  have at most  $s - 1$  such neighbors and thus by Claim 3 we know that the total size of  $R$  is at most  $(s - 1) \cdot |Q \setminus R|$ .

In total,  $G$  has size  $|V(G)| = |W| + |Q| \leq k(s + 1) + s \cdot k \cdot (s^2 + k(s + 1) - 1) = k(s^3 + ks^2 + ks + 1)$  contradicting the assumption that the graph had more than  $k(s^3 + ks^2 + ks + 1)$  vertices. This concludes the proof of the boundary lemma. □

From this boundary lemma, follows that any reduced instance that is still ‘big’ has a  $k$ - $K_{1,s}$ -packing. Since the boundary given by the Lemma 4 does not depend on the main input, but only on the parameter and the problem in question, we can say that the reduced instance is a ‘problem-kernel’ and that the problem is in FPT .

**Lemma 5.** (*Kernelization Lemma*) If a graph  $G$  is reduced and has  $|V(G)| > k(s^3 + ks^2 + ks + 1)$ , then it contains a  $k$ - $K_{1,s}$ -packing.

*Proof.* Assume in contradiction to the stated theorem that there exists a graph  $G$  of size  $|V(G)| > k(s^3 + ks^2 + ks + 1)$ , but where  $G$  has no  $k$ - $K_{1,s}$ -packing.

Let  $k' < k$  be the largest  $k'$  for which  $G$  is a YES-instance. By the Boundary Lemma 4 we know that  $|V(G)| \leq k'(s^3 + k's^2 + k's + 1) < k(s^3 + ks^2 + ks + 1)$ . This contradicts the assumption. □

Thus for any  $k$ - $K_{1,s}$ -packing we can prove a quadratic kernel. However, for the special case  $s = 2$ , we can improve on this. This is the topic of the next section.

## 4 The special case of $P_2$ : a linear kernel

A 2-star can also be seen as a path with three vertices, denoted  $P_2$ . For this special case we can employ a different set of reduction rules to obtain a linear kernel for packing  $P_2$ 's into a graph.

$k$ - $P_2$ -PACKING

INSTANCE: Graph  $G = (V, E)$

PARAMETER:  $k$

QUESTION: Are there  $k$  vertex disjoint instances of  $P_2$  in  $G$ ?

To improve on the quadratic kernel obtained in the previous section, we will make use of a series of reduction rules based on the ideas of crown decompositions [CFJ03].

**Definition 2.** A crown decomposition  $(H, C, R)$  in a graph  $G = (V, E)$  is a partitioning of the vertices of the graph into three sets  $H$ ,  $C$ , and  $R$  that have the following properties:

1.  $H$  (the head) is a separator in  $G$  such that there are no edges in  $G$  between vertices belonging to  $C$  and vertices belonging to  $R$ .
2.  $C = C_u \cup C_m$  (the crown) is an independent set in  $G$ .
3.  $|C_m| = |H|$ , and there is a perfect matching between  $C_m$  and  $H$ .

There are several recent papers that use crown decompositions of graphs to obtain good results in parameterized complexity [CFJ03, FHRST04, F03, ACFL04, PS04]. These papers either apply the crown directly to the problem instance ([CFJ03, ACFL04]) or create an auxiliary graph where they apply crown reduction techniques.

In this paper we instead modify the crown decomposition to fit our particular problem. The first variation is 'double crown'-decomposition<sup>1</sup> where each vertex in  $H$  has two vertices from  $C$  matched to it (as opposed to only one).

**Definition 3.** A double crown decomposition  $(H, C, R)$  in a graph  $G = (V, E)$  is a partitioning of the vertices of the graph into three sets  $H$ ,  $C$ , and  $R$  that have the following properties:

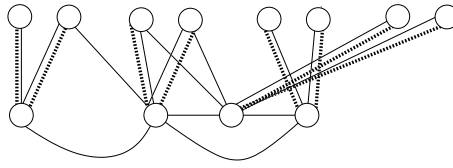
1.  $H$  (the head) is a separator in  $G$  such that there are no edges in  $G$  between vertices belonging to  $C$  and vertices belonging to  $R$ .
2.  $C = C_u \cup C_m \cup C_{m_2}$  (the crown) is an independent set in  $G$ .
3.  $|C_m| = |H|$ ,  $|C_{m_2}| = |H|$  and there is a perfect matching between  $C_m$  and  $H$ , and a perfect matching between  $C_{m_2}$  and  $H$ .

Another variation of the crown is the 'fat crown'-decomposition<sup>2</sup> where instead of independent vertices in  $C$  we have  $K_2$ 's as shown in figure 2.

**Definition 4.** A fat crown decomposition  $(H, C, R)$  in a graph  $G = (V, E)$  is a partitioning of the vertices of the graph into three sets  $H$ ,  $C$  and  $R$  that have the following properties:

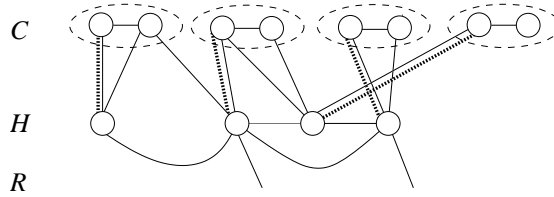
<sup>1</sup> The dashed lines in the figure indicate how each vertex in  $H$  is matched to two vertices in  $C$ .

<sup>2</sup> As in the case of the 'double crown', the dashed lines indicate the matching between  $H$  and  $C_m$  and the dashed ellipses show which  $K_2$  the vertex in  $H$  is matched to.



**Fig. 1.** Example of ‘double crown’

1.  $H$  (the head) is a separator in  $G$  such that there are no edges in  $G$  between vertices belonging to  $C$  and vertices belonging to  $R$ .
2.  $G[C]$  is a forest where each component is isomorphic to  $K_2$
3.  $|C| \geq |H|$ , and if we contract the edges in each  $K_2$  there is a perfect matching between  $C$  and  $H$ .



**Fig. 2.** Example of ‘fat crown’

Using the ‘crown’, ‘double crown’ and ‘fat crown’ we can create powerful reduction rules.

**Lemma 6.** A graph  $G = (V, E)$  that admits a ‘double crown’-decomposition  $(H, C, R)$  has a  $k$ - $P_2$ -packing if and only if  $G \setminus (H \cup C)$  has a  $(k - |H|)$ - $P_2$ -packing.

*Proof.* ( $\Leftarrow$ ): If  $G \setminus (H \cup C)$  has a  $(k - |H|)$ - $P_2$ -packing then it is obvious that  $G$  has a  $k$ - $P_2$ -packing as  $H \cup C$  has a  $|H|$ - $P_2$ -packing ( $v \in H$  and  $v$ 's matched vertices from  $C_m$  and  $C_{m2}$  form a  $P_2$ ).

( $\Rightarrow$ ): We want to prove that if  $G$  has a  $k$ - $P_2$ -packing then  $G \setminus (H \cup C)$  has a  $(k - |H|)$ - $P_2$ -packing. Assume in contradiction that there exists a graph  $G'$  that has a crown-decomposition  $(H', C', R')$  that contradicts the lemma. This implies that  $H' \cup C'$  participates in  $x > |H'|$   $P_2$ 's. Since  $H'$  is a cutset, and  $C'$  is an independent set in the graph, every  $P_2$  in  $G$  that has vertices in  $H' \cup C'$  must have at least one vertex from  $H'$ . Thus we can have at most  $|H'|$   $P_2$ 's which is a contradiction.  $\square$

**Lemma 7.** A graph  $G = (V, E)$  that admits a ‘fat crown’-decomposition  $(H, C, R)$  has a  $k$ - $P_2$ -packing if and only if  $G \setminus (H \cup C)$  has a  $(k - |H|)$ - $P_2$ -packing.

The proof of Lemma 7 is analogue to Lemma 6, thus omitted.

To apply crown-decompositions we need to know when we can expect to find one. A very useful result in this regard can be deduced from [CFJ03, page 7], and [F03, page 8]. Fortunately, the results also apply to the variations of crown decomposition described here.

**Lemma 8.** Any graph  $G$  with an independent set  $I$ , where  $|I| \geq |N(I)|$ , has a crown decomposition  $(H, C, R)$ , where  $H \subseteq N(I)$ , that can be found in  $\mathcal{O}(|V| + |E|)$  time, given  $I$ .

**Corollary 1.** Any graph  $G$  with a collection  $J$  of independent  $K_2$ s where such that  $|N(V(J))| \leq |J|$ , has a fat crown decomposition  $(H, C, R)$ , where  $H \subseteq N(J)$ , that can be found in linear time, given  $J$ .

*Proof.* This follows from the previous Lemma. If we replace each  $K_2$  with a single vertex, then by Lemma 8 this graph admits a crown-decomposition. We can reintroduce the  $K_2$ s to obtain a ‘fat-crown’.  $\square$

**Lemma 9.** Any graph  $G$  with an independent set  $I$ , where  $|I| \geq 2|N(I)|$ , has a double crown decomposition  $(H, C, R)$ , where  $H \subseteq N(I)$ , that can be found in linear time, given  $I$ .

*Proof.* Let  $G$  be a graph with an independent set  $I \subseteq V(G)$  such that  $2|N(I)| \leq |I|$ . Create a graph with  $G' = G$ , but for every vertex  $v \in N(I)$  add a copy  $v'$ , such that  $N(v) = N(v')$ . By Lemma 8  $G'$  has a crown-decomposition  $(H, C, R)$  such that  $H \subseteq N_{G'}(I)$ . We now claim that we can use this crown to construct a ‘double crown’  $(H', C', R')$  in  $G$ .

First observe that  $v \in H$  if and only if  $v' \in H$ . Assume in contradiction that  $v \in H$  but  $v' \notin H$ .  $v$  must be matched to some vertex  $u$  in  $C$ . Since  $N(v) = N(v')$  we have that  $v'$  cannot be in  $C$  as it would contradict that  $C$  is an independent set. Also  $v'$  is not in  $R$  as that would contradict that  $H$  is a cut-set. Thus  $v'$  must be in  $H$ , contradicting the assumption.

With this observation the result follows easily as  $H$  consists of pairs of vertices, a vertex and its copy. Each pair  $v$  and  $v'$  in  $H$  is matched to two vertices  $u_1$  and  $u_2$ . In  $G$ , let  $v$  be in  $H'$  and let it be matched to both  $u_1$  and  $u_2$ . Do this for every pair in  $H$ . It is easy to see that this forms a double crown in  $G$ .  $\square$

We will now describe a polynomial time preprocessing algorithm that reduces the graph to a kernel of size at most  $15k$ . The process below either reduces the graph or produces a packing of the appropriate size, thus we can reach a kernel by repeating the following three steps.

Step 1. Compute an arbitrary maximal  $P_2$ -packing  $W$ . Let  $Q = V \setminus W$ .

Step 2. Let  $X$  be the collection of components in  $G[Q]$  isomorphic to  $K_2$ . If  $|X| \geq |N(X)|$  in  $G$  then reduce by Lemma 7.

Step 3. Let  $I$  be the isolated vertices  $I$  in  $G[Q]$ . If  $|I| \geq 2|N(I)|$  in  $G$  then reduce by Lemma 6.

**Lemma 10.** If  $|V(G)| > 15k$  then the preprocessing algorithm will either find a  $k$ - $P_2$ -packing or it will reduce  $G$ .

*Proof.* Assume in contradiction to the stated lemma that  $|V(G)| > 15k$ , but that the algorithm produced neither a  $k$ - $P_2$ -packing nor a reduction of  $G$ .

By the assumption the maximal packing  $W$  is of size  $|W| < 3k$ . Let  $Q = V \setminus W$ . Let  $Q_i$  be the vertices in  $Q$  that have degree  $i$  in the graph induced by  $Q$ .

*Claim 5.*  $\forall i \geq 2, Q_i = \emptyset$

*Proof.* This is clear as otherwise  $W$  could not be maximal.  $\square$

*Claim 6.*  $|Q_1| \leq 6k$

*Proof.* Assume in contradiction that  $|Q_1| > 6k$ . This implies that number of  $K_2$ s  $X$  in  $Q$  is greater than  $3k$ , but then  $|X| > |W|$ . By Corollary 1  $G$  has a ‘fat crown’ and should have been reduced in step 2 of the algorithm, contradicting that no reduction took place.  $\square$

*Claim 7.*  $|Q_0| \leq 6k$

*Proof.* Assume in contradiction that  $|Q_0| > 6k$ , but then  $|Q_0|$  is more than  $2|W|$  and by Lemma 9  $G$  has a ‘double crown’ and by Lemma 6 should have been reduced in step 3 of the algorithm, contradicting that no reduction took place.  $\square$

Thus the total size  $|V(G)| = |W| + |Q_0| + |Q_1| + |Q_2| + \dots \leq 3k + 6k + 6k + 0 = 15k$ . This contradicts the assumption that  $|V(G)| > 15k$ .  $\square$

**Corollary 2.** Any instance  $(G, k)$  of  $P_2$ -packing can be reduced to a problem kernel of size  $\mathcal{O}(k)$ .

*Proof.* This follows from the Lemma, as we can run the preprocessing algorithm until it fails to reduce  $G$ . By Lemma 10 the size is then at most  $15k$ .  $\square$

## 5 Running Time

To compute the kernel we will run the preprocessing algorithm  $\mathcal{O}(n)$  times. Since a maximal  $k$ -packing of  $P_2$ 's can be computed in  $\mathcal{O}(kn)$  the most time consuming part of the preprocessing algorithm is the  $\mathcal{O}(|V|+|E|)$  time needed to compute a crown decomposition. Thus the kernelization process can be completed in  $\mathcal{O}(n^3)$  time.

We will apply a straightforward brute-force algorithm on the kernels to find the optimal solution. In the case of  $P_2$ -packing we will select the center-vertices of the  $P_2$ s in a *brute force* manner. There are  $\binom{15k}{k}$  ways to do this. By Stirling's formula this expression is bounded by  $2^{5.301k}$ . With  $k$  center vertices already selected the problem reduces to a problem on bipartite graphs where the question is if the left hand side each can have 2 neighbors assigned to it. This can easily be transformed to MAXIMUM BIPARTITE MATCHING by making 2 copies of each vertex on the left hand side. MAXIMUM BIPARTITE MATCHING can be solved in time  $\mathcal{O}(\sqrt{|V|}|E|)$  [HK73]. We now have  $15k + k$  vertices, and thus  $\mathcal{O}(k^2)$  edges. We can solve each of these in time  $\mathcal{O}(k^{2.5})$ , giving a running time of  $\mathcal{O}(2^{5.301k} k^{2.5})$  for the kernel. In total we can decide the  $P_2$ -packing problem in time  $\mathcal{O}(2^{5.301k} k^{2.5} + n^3)$ .

Applying the same technique for the  $s$ -stars we will achieve  $\mathcal{O}(2^{\mathcal{O}(k \log k)} k^{\mathcal{O}(1)} n^{\mathcal{O}(1)})$ , asymptotically worse due to the quadratic kernel.

## 6 Conclusions and Further Research

Packing vertex-disjoint copies of a graph  $H$  into another graph  $G$  is NP-complete as long as  $H$  has more than two vertices [HK78]. We have analyzed within the framework of parameterized complexity a specific instance of this problem, the packing of vertex-disjoint stars with  $s$  leaves. We have proved that packing  $K_{1,2}$ s in a graph  $G$ , equivalently  $k$ - $P_2$ -PACKING has a linear kernel.

Our algorithm for  $k$ - $P_2$ -Packing runs in time  $\mathcal{O}(2^{5.301k} k^{2.5} + n^3)$ . This running time arises from reducing the problem to a kernel of size  $15k$ . We believe that this kernel can be further improved and thus the running time substantially decreased, however, it is already much better than  $2^{\mathcal{O}(|H|k \log k + k|H| \log |H|)}$ , the running time of the general algorithm in [FHRST04].

We have also proved that  $s$ -Star Packing ( $K_{1,s}$ -Packing) is in general fixed-parameter tractable with a quadratic kernel size. We also gave an algorithm for the general case with running time  $\mathcal{O}^*(2^{\mathcal{O}(k \log k)})$ , but this is not an improvement to [FHRST04] or [AYZ95].

There are several related problems that could be considered in the light of the techniques used in Section 3. The most obvious one is the following:

$k$ - $K_{1,s}$ -PACKING  
 INSTANCE: Graph  $G = (V, E)$   
 PARAMETER:  $k$   
 QUESTION: Are there  $k$  edge-disjoint instances of  $K_{1,s}$  in  $G$ ?

This problem is fixed-parameter tractable when  $s = 2, 3$  using Robertson and Seymour's Graph Minor Theorem [RS99] since it can be easily proved that its NO-instances are closed under the minor operations. The issue here is that this method is non-constructive and carries a fast growing function  $f(k)$ . Possibly, applying similar arguments as those in Section 4 would lead to a much better running time.

**Acknowledgements.** We would like to thank Mike Fellows for all the inspiring conversations leading to the completion of this paper.

## References

- [ACFL04] F. Abu-Khzam, R. Collins, M. Fellows and M. Langston. Kernelization Algorithms for the Vertex Cover Problem: Theory and Experiments. *Proceedings ALENEX 2004*, Springer-Verlag, *Lecture Notes in Computer Science* (2004), to appear.
- [AC99] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, M. Protasi. Complexity and Approximation *Springer Verlag* (1999).
- [AYZ95] N. Alon, R. Yuster, U. Zwick. Color-Coding, *Journal of the ACM*, Volume 42(4), pages 844–856 (1995).
- [BM02] R. Bar-Yehuda, M. Halldórsson, J. Naor, H. Shachnai, I. Shapira. Scheduling Split Intervals. *Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 732-741 (2002).
- [BK01] R. Bejar, B. Krishnamachari, C. Gomes, and B. Selman. Distributed constraint satisfaction in a wireless sensor tracking system. *Workshop on Distributed Constraint Reasoning, International Joint Conference on Artificial Intelligence*, 2001
- [CFJ03] B. Chor, M. Fellows, D. Juedes. An Efficient FPT Algorithm for Saving  $k$  colors. *Manuscript* (2003).
- [DFS99] R. Downey, M. Fellows, U. Stege. Parameterized Complexity: A Framework for Systematically Confronting Computational Intractability. *AMS-DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, Volume 49, pages 49-99 (1999).
- [F03] M. Fellows. Blow-Ups, Win/Win's, and Crown Rules: Some new Directions in FPT. *Proceedings 29th Workshop on Graph Theoretic Concepts in Computer Science*, LNCS 2880(2003), pages 1-12.
- [FHRST04] M.Fellows, P.Heggernes, F.Rosamond, C. Sloper, J.A.Telle, Exact algorithms for finding  $k$  disjoint triangles in an arbitrary graph, *Proceedings 30th Workshop on Graph Theoretic Concepts in Computer Science* (2004) LNCS 3353, pages 235-244

- [FM+00] M.R. Fellows, C. McCartin, F. Rosamond, and U. Stege. Coordinatized Kernels and Catalytic Reductions: An Improved FPT Algorithm for Max Leaf Spanning Tree and Other Problems, *Foundations of Software Technology and Theoretical Computer Science*, LNCS 1974 (2000), page 240.
- [HK73] J. Hopcroft and R. Karp. An  $n^{5/2}$  Algorithm for Maximum Matchings in Bipartite Graphs. *SIAM Journal on Computing*, 2 pages 225–231 (1973).
- [HK78] P. Hell and D. Kirkpatrick. On the complexity of a generalized matching problem. *Proceedings of 10th ACM Symposium on theory of computing*, pages 309–318 (1978).
- [HK86] P. Hell and D. Kirkpatrick. Packings by complete bipartite graphs. *SIAM Journal of Algebraic Discrete Methods*, number 7, pages 199–209 (1986).
- [HS89] C. Hurkens and A. Schrijver. On the size of systems of sets every  $t$  of which have an SDR, with application to worst case ratio of Heuristics for packing problems. *SIAM Journal of Discrete Mathematics*, number 2, pages 68–72 (1989).
- [K94] V. Kann. Maximum bounded H-matching is MAX-SNP-complete. *Information Processing Letters*, number 49, pages 309–318 (1994).
- [PS04] E. Prieto, C. Sloper. Creating Crown Structure — The case of Max Internal Spanning Tree. *Submitted*
- [RS99] N. Robertson, P.D. Seymour. *Graph Minors XX. Wagner’s conjecture*, to appear.