MAX INTERNAL SPANNING TREE

Reducing to Independent Set Structure — the Case of k-INTERNAL SPANNING TREE[1]

Elena Prieto Christian Sloper

Abstract

The k-INTERNAL SPANNING TREE problem asks whether a certain graph G has a spanning tree with at least k internal vertices. Basing our work on the results presented in [PS03], we show that there exists a set of reduction rules that modify an arbitrary spanning tree of a graph into a spanning tree with no induced edges between the leaves. Thus, the rules either produce a tree with many internal vertices, effectively deciding the problem, or they identify a large independent set, the leaves, in the graph. Having a large independent set is beneficial, because then the graph allows both 'crown decompositions' and path decompositions. We show how this crown decomposition can be used to obtain a $\mathcal{O}(k^2)$ kernel for the k-INTERNAL SPANNING TREE problem, improving on the $\mathcal{O}(k^3)$ kernel presented in [PS03].

8.1 INTRODUCTION

The subject of Parameterized Complexity is motivated by an abundance of NP-complete problems that have very different behavior when parameterized. These problems includes well-known problems like DOMINATING SET, BANDWIDTH, SET SPLITTING, and INDEPENDENT SET (for definitions the reader may refer to [GJ79]). Some of the NP-complete are tractable when parameterized and admits very good parameterized algorithms. A formal definition of the class of problems which are tractable when parameterized is defined as follows:

[1]This paper has been accepted to Nordic Journal of Computing and is due to appear.

[PS03] E. Prieto, C. Sloper. Either/Or: Using Vertex Cover Structure in designing FPT-algorithms - the case of k-Internal Spanning Tree, *Proceedings of WADS 2003*, LNCS vol 2748, pp 465-483.

[PS04] E. Prieto, C. Sloper. Looking at the Stars, *To Appear in proceedings of IWPEC04*, Springer Lecture Notes in Computer Science, (2004).

[PT93] J.A.Telle and A.Proskurowski. Practical algorithms on partial k-trees with an application to domination-like problems. *Proceedings WADS'93 - Third Workshop on Algorithms and Data Structures.* Springer Verlag, Lecture Notes in Computer Science vol.709 (1993) 610-621.

[RS83] N. Robertson and P. D. Seymour, Graph minors. I. Excluding a forest, *J. Comb. Theory Series B*, 35 (1983), pp. 39-61.

[RS99] N. Robertson, PD. Seymor. Graph Minors. XX Wagner's conjecture. *To appear*.

Definition 8.1.1 *(Fixed Parameter Tractability) A parameterized problem $L \subseteq \Sigma^* \times \Sigma^*$ is* fixed-parameter tractable *if there is an algorithm that correctly decides, in time $f(k)\, n^\alpha$, for input $(x, y) \in \Sigma^* \times \Sigma^*$ whether or not $(x, y) \in L$, where n is the size of the input x, $|x| = n$, k is the parameter, α is a constant (independent of k) and f is an arbitrary function.*

The class of fixed-parameter tractable problems is denoted FPT.

It is not believed that all NP-complete problems are Fixed Parameter Tractable, the class is split into a hierarchy of classes FPT⊆W[1]⊆W[2]⊆ $\cdots$ ⊆W[P]. Here the classes W[1]⊆W[2]⊆ $\cdots$ ⊆W[P] are intractable and we justify this by a completeness-result not unlike classical complexity. In [CCDF97] Cai, Chen, Downey, and Fellows proved that k-SHORT NONDETERMISTIC TURING MACHINE ACCEPTANCE (Will a Nondetermistic Turing Machine halt in k or less steps?) is W[1]-complete thus giving strong natural evidence that $FPT \neq W[1]$.

Further background on parameterized complexity can be found in [DF98].

The problem we address in this paper concerns spanning trees, namely k-INTERNAL SPANNING TREE (Does G have a spanning tree with at most $n - k$ leaves?). The problem is NP-complete as HAMILTONIAN PATH can be considered a special case of k-INTERNAL SPANNING TREE by making $k = |V| - 2$, and HAMILTONIAN PATH is NP-complete.

In Section 8.4 we use standard techniques to show that k-INTERNAL SPANNING TREE is in FPT. In Section 8.5 we describe how to use the bounded *independent set structure* to design an FPT algorithm for k-INTERNAL SPANNING TREE. We give an analysis of the running time of the algorithm generated by the method in Section 8.6. In Section 8.7 we show how the independent structure allows a pathwidth decomposition which can be useful for some problems, we illustrate this on NONBLOCKER, the dual of DOMINATING SET. We conclude with some remarks about future research. Also, as a consequence of the preprocessing of the graph necessary to create our fixed-parameter algorithm, we easily obtain a polynomial time 2-approximation algorithm for k-INTERNAL SPANNING TREE.

8.2 USING REDUCTION RULES

Currently, the main practical methods of FPT algorithm design are based on *kernelization* and *bounded search trees*. The idea of kernelization is relatively simple, and can be quickly illustrated for the VERTEX COVER problem.

In kernelization we seek to bound the size of the input instance to a function of the parameter. To achieve this we preprocess the graph using reduction rules. Two examples of reduction rules for VERTEX COVER are the *leaf-rule* and the *Buss-rule*. The leaf-rule states that given an instance (G, k) where G has a pendant vertex v of degree 1 connected to vertex u, then it is never wrong to include u in the vertex cover instead of v, as the edge uv must be covered and u possibly covers other edges as well. Thus (G, k) can be reduced to $(G', k-1)$, where $G' = G - \{u, v\}$. Another rule, the Buss-rule [B98], states that if the instance (G, k) has a vertex u of degree greater than k, then u must be in every k-vertex cover of G, since otherwise all its more than k neighbors would have to be included. Thus, (G, k) can be reduced to $(G', k-1)$ where $G' = G - u$.

The term ‘Reduction rule’ is somewhat unfortunate as it seems to imply a rule that reduces the graph in size. Although a reduction in size is a consequence, it is wrong to consider this the goal. Reduction rules should not be viewed as a ‘reduction in size’ but rather as a ‘*reduction to structure*’. In parameterized complexity the goal of the reduction process is to prove that the problem is after preprocessing trivially decidable for any ‘large’ instance, i.e., irreducible instances larger than a function $f(k)$, our kernel size. It is here that reduction rules provide us with the necessary information about the structure of the instance. In a sense, reduction rules are used to impose structure that allow us to make claims about irreducible graphs.

It is easy to be led astray by reduction rules that only offer a reduction in size, since if they do not also convey some useful structural information, then the rule is ultimately useless from the point of view of kernelization. However, such a rule could of course be very useful in practice as a preprocessing tool or in search tree algorithms.

To illustrate what we mean we again consider the leaf-rule and the Buss-rule for vertex cover. After repeated application of both we reach a graph where neither rule can be applied. We say that this graph is *irreducible* for our reduction rules. From the knowledge that the rules do not apply we can conclude that the graph has two properties. First, from the leaf-rule, we know that every vertex has degree at least 2. Second, from the Buss-rule, we know that every vertex has degree at most k.

Knowing that the minimum degree of the graph is at least two is important for ruling out cases in the search tree analysis, but it does not provide any ‘useful’ structural information as we both have arbitrarily large graphs with minimum degree at least two that have a k-Vertex Cover, and others that do not have a k-Vertex Cover. However, with the Buss-rule the situation is different. Knowing that every vertex has degree at most k combined with the fact that we can select at most k of them is enough to conclude that no irreducible yes-instance for k-Vertex Cover has more than $k(k+1)$ vertices. Thus we can trivially decide any irreducible instance of size greater than $f(k) = k(k+1)$. We have a quadratic kernel for vertex cover.

In this paper we show that we can learn something about the structure of the graph on a global level without reducing the graph in size. We show that there exists a set of reduction rules that modify an arbitrary spanning tree of a graph into a spanning tree with no induced edges between the leaves. Thus, the rules either produce a tree with many internal vertices, effectively deciding the problem, or they identify a large independent set, the leaves, in the graph. Having a large independent set is beneficial, because then the graph allows a 'crown decomposition'. We show how this crown decomposition can be used to obtain a $\mathcal{O}(k^2)$ kernel for the k-INTERNAL SPANNING TREE problem, improving on the $\mathcal{O}(k^3)$ kernel presented in [PS03].

8.3 PRELIMINARIES

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

We use the simpler $G \setminus v$ to denote $G[V \setminus v]$ and $G \setminus e$ to denote $G = (V, E \setminus e)$ where v and e is a vertex and an edge respectively. Likewise for sets, $G \setminus V'$ 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 a *k-internal tree* T is a subgraph of G, where T is a tree with at least k internal vertices. If $V(T) = V(G)$ we say that T is a *k-internal spanning tree* of G.

8.4 k-INTERNAL SPANNING TREE IS FPT

Using Robertson and Seymour's Graph Minor Theorem it is straightforward to prove the following membership in FPT.

Lemma 8.4.1 *The k-*INTERNAL SPANNING TREE *problem is in* FPT.

Proof. Let $\mathcal{F}_k$ denote the family of graphs that do not have spanning trees with at least k internal vertices. It is easy to observe that for each k this family is a lower ideal in the minor order. Less formally, let (G, k) be a NO-instance of k-INTERNAL SPANNING TREE, that is a graph G for which there is no spanning tree with at least k internal vertices. The local operations which configure the minor order (i.e., edge contractions, edge deletions and vertex deletions) will always transform this NO-instance into another NO-instance. By the Graph Minor Theorem of Robertson and Seymour and its companion result that order testing in the minor order is FPT [RS99] we can conclude that k-INTERNAL SPANNING TREE is also FPT. (An exposition of well-quasiordering as a method of FPT algorithm design can be found in [DF98].) □

Unfortunately, this FPT proof technique suffers from being nonuniform and nonconstructive, and gives an $\mathcal{O}(f(k)n^3)$ algorithm with a very fast-growing parameter function compared to the one we obtain in Section 8.5.

We remark that it can be shown that all fixed graphs with a vertex cover of size k are well-quasi ordered by ordinary subgraphs and have linear time order tests [F03]. The proof of this is substantially shorter than the Graph Minor Project and could be used to simplify Lemma 8.4.1.

8.5 INDEPENDENT SET STRUCTURE

In this section we show how to obtain a quadratic kernel for k-INTERNAL SPANNING TREE. We first give a set of reduction rules that either produces a spanning tree with the desired number of internal vertices or shows that the graph has a large independent set.

We will then show that this structural information is enough to prove that any irreducible instance has size at most $\mathcal{O}(k^2)$, improving the result obtained in [PS03]. Using a *crown decomposition* we are able to prove that any graph with a large independent set contain redundant vertices that can be removed, reaching the desired kernel size.

Lemma 8.5.1 *Any graph G has a spanning tree T such that all the leaves of T are independent vertices in G or G has a spanning tree T' with only two leaves.*

Proof. Given a spanning tree T of a graph G, we say that two leaves $u, v \in T$ are *in conflict* if $uv \in E(G)$. We now show that given a spanning tree with i conflicts it is possible to obtain a spanning tree with less than i conflicts using one of the rules below:

1. If x and y are in conflict and z, the parent of x has degree 3 or more, then a new spanning tree T' could be constructed using the edge xy in the spanning tree instead of xz.
2. If x and y are in conflict and both their parents have degree 2, then let x' be the first vertex on a path from x to y that has degree different from 2. If there is no such vertex x' we know that the spanning tree is a Hamiltonian path and has only two leaves. Otherwise we create a new spanning tree disconnecting the path from x to x' (leaving x') and connecting x to y, repairing the conflict between x and y. Since x' is now of degree at least 2 we have not created any new conflicts.

The validity of the rules is easy to verify and it is obvious that they can be executed in polynomial time. Lemma 8.5.1 then follows by recursively applying the rules until no conflicts exist. □

Observe that any application of the rules on a spanning tree produces a spanning tree with more internal vertices, thus the reduction rules above are used less than k times.

For the remainder of the paper we assume that we obtained a spanning tree T where the leaves are independent and we define the set A as the internal vertices of T and B as the leaves of T. Observe that A is a connected set and B an independent set.

Several corollaries follow easily from this Lemma. One of them gives an approximation for k-INTERNAL SPANNING TREE, the others relate the problem to the well-studied INDEPENDENT SET.

Corollary 8.5.1 k-INTERNAL SPANNING TREE *has a 2-approximation algorithm.*

Proof. Note that since B is an independent set it is impossible to include more than $|A|$ elements of B as internals in the optimal spanning tree, as otherwise the spanning tree would contain a loop. The maximum number of internal vertices is at most $2|A|$, and since the spanning tree generated by the algorithm in Lemma 8.5.1 has $|A|$ internal vertices, it is a 2-approximation for k-INTERNAL SPANNING TREE. □

Corollary 8.5.2 *If a graph $G = (V, E)$ is a* NO*-instance for $(n-k)$*-INDEPENDENT SET *then G is a* YES*-instance for k*-INTERNAL SPANNING TREE.

Proof. If a graph does not have an independent set of size greater then $(n - k)$, then $|B| < (n - k)$ and $|A| \geq k$, a YES-instance of k-INTERNAL SPANNING TREE. □

Corollary 8.5.3 *If a graph $G = (V, E)$ is a* YES*-instance for $(n - k)$*-INDEPENDENT SET *then G is a* NO*-instance for $(2k + 1)$*-INTERNAL SPANNING TREE.

Proof. If G has an $(n-k)$-INDEPENDENT SET I then for each vertex in I that we include as an internal in the spanning tree we must include at least one other vertex in $V - I$. Thus at most $2k$ vertices can be internal in the spanning tree and therefore G is a NO-instance for $(2k + 1)$-INTERNAL SPANNING TREE. □

We now know that if a graph does not have an $(n - k)$-INDEPENDENT SET then it is a YES-instance for k-INTERNAL SPANNING TREE. We will now show how we can use this structural information to give a bound on the size of the kernel. To reduce the large independent set we will use the crown-reduction technique seen in [CFJ03, FHRST04, F03, ACFL04] to reduce the size of the independence set.

Definition 8.5.1 *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 in C and vertices in 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.*

Although being a recently introduced idea, some theory about the existence of crowns can be found in literature.

The following theorem can be deduced from [CFJ03, page 7], and [F03, page 8].

Theorem 8.5.1 *Any graph G with an independent set I, where $|I| \geq n/2$, has a crown decomposition (H, C, R), where $H \subseteq N(I)$ and $C \subseteq I$, and this crown decomposition can be found in time $\mathcal{O}(|V| + |E|)$, given I.*

In [FHRST04] the following is observed:

Lemma 8.5.2 *If a bipartite graph $G = (V \cup V', E)$ has two crown decompositions (H, C, R) and (H', C', R') where $H \subseteq V$ and $H' \subseteq V$, then G has a crown decomposition $(H'' = H \cup H', C'' = C \cup C', R'' = R \cap R')$.*

From these two results we can deduce that if the independent set is sufficiently large then there exists a crown-decomposition where $C_u \neq \emptyset$.

Theorem 8.5.2 *Any graph G with an independent set I, where $|I| \geq 2n/3$, has a crown decomposition (H, C, R), where $H \subseteq N(I)$, $C \subseteq I$ and $C_u \neq \emptyset$, that can be found in time $\mathcal{O}(|V||E|)$ given I.*

Proof. First observe that $|C_m| \leq |N(I)|$. By Theorem 8.5.1, G has a crown decomposition (H, C, R), where $H \subseteq N(I)$. If $|C| \geq \frac{n}{3}$ then $|C| > N(I)$ and the result follows, otherwise $|I \setminus C| \geq n/3$ and by Theorem 8.5.1 $G \setminus C$ has a crown decomposition (H'', C', R'). By Lemma 8.5.2 these crown-decompositions can be combined to a crown-decomposition (H'', C'', R''). This process can be repeated until the combined crown-decomposition $(\hat{H}, \hat{C}, \hat{R})$ no longer satisfies $|I \setminus \hat{C}| > \frac{n}{3}$, thus $|\hat{C}| > |N(I)|$ and the result follows. The algorithm in Theorem 8.5.1 is executed at most n times, giving the bound of $\mathcal{O}(|V||E|)$. □

Using an approach similar to the one in [FHRST04], we create an auxiliary graph model where a crown decomposition in the auxiliary graph infer reductions in the original graph.

Observe that vertices in the independent set B can only participate in a spanning tree in two ways. Either they are leaves, or they are internal vertices between two or more vertices in A.

We will define the model as the bipartite graph $G_I = (A' \cup B, E_I)$ where: $A' = A \cup (A \times A)$, i.e., A and a vertex vv' for every pair v and v' in A. The edges of G_I are the original edges E and an edge between a vertex $b \in B$ and a pair vertex if b has edges to both vertices of the pair. $E_I = E \cup \{(vv')b \mid vv' \in A', b \in B, \{vb, v'b\} \subseteq E\}$.

We now prove the following reduction rule.

Reduction Rule 3 *If G_I has a crown decomposition $(H, C_m \cup C_u, R)$ where $H \subseteq A'$ then G has a k-internal spanning tree if and only if $G \setminus C_u$ has a k-internal spanning tree.*

Proof. One direction is trivial, if $G \setminus C_u$ has a k-internal spanning tree then G obviously has one, as we cannot get fewer internals by adding vertices to the graph.

We prove the other direction by construction. Let S^* be a k-internal spanning tree in G. $S^* - C$ is a forest F. We will show that we can construct a k-internal spanning tree from F by using vertices from C_m to connect the components in F, showing that C_u is redundant.

Let Q be the components of F. Observe that at most $|Q| - 1$ vertices from C connected the components in S^* and that all these vertices are internal.

Let Q_i and Q_j be two arbitrary components in Q that were connected by a vertex $c \in C$. Let u_i and u_j be the vertices in Q_i and Q_j respectively of which c is a neighbor in S^*. Connect these vertices using the vertex in C_m matched to the pair-vertex $u_i u_j$. Because of the matching in the crown decomposition, this vertex is uniquely determined and never used elsewhere in the construction. The number of components have decreased by one. Repeat this process until all components in Q are connected. Note that we added $|Q| - 1$ internal vertices, thus we used at least as many vertices to connect F as the optimal solution did. F is now a tree.

For every leaf u_i in F which is not a leaf in S^*, append the vertex matched to $u_i \in C_m$. As above, the vertex matched to u_i is uniquely determined and not used elsewhere in the construction.

Note that the construction of the k-internal spanning tree never depends on C_u, thus C_u is redundant. □

Lemma 8.5.3 *If G is reduced and $|V(G)| > k^2 + 2k$ then G has a k-Internal Spanning Tree.*

Proof. Assume in contradiction to the stated lemma that G is reduced and $|V(G)| > k^2 + 2k$, but that G has no k-Internal Spanning Tree.

By assumption $|A| < k$, otherwise the tree produced in Lemma 8.5.1 would have k internal vertices. Hence, $|B| = |V(G) - A| > k^2 + k$. In G_I we have that is $|A'| < k(k+1)/2$, i.e., $|B| > 2|A'|$. Thus by Lemma 9.4.3, G_I has a crown with at least one vertex in C_u, contradicting the assumption that G was reduced.

□

8.6 ANALYSIS OF THE RUNNING TIME

Our algorithm is similar to that found in [PS03] and works in several stages. It first calls a regular spanning tree algorithm and then modifies it to make the leaves independent. Then, if the spanning tree does not contain enough internals, we know that the spanning tree's leaves form an independent set. We use our crown reduction rule to reduce the independent set, after which the graph is reduced in size to $\mathcal{O}(k^2)$. Finally, we employ a brute-force spanning tree algorithm to find an optimal solution for the reduced instance.

We can use a simple breadth-first search algorithm to obtain any spanning tree in G. This spanning tree can thus be obtained in time $\mathcal{O}(|V| + |E|)$ [CLR90]. The conflicts (i.e., the leaves in the tree which are not independent) can be detected in time $\mathcal{O}(|E|)$ and repaired in time $\mathcal{O}(|V|)$.

Given a large independent set, a crown can be found in linear time. A maximal crown can be found in time $\mathcal{O}(|V||E|)$. We have then identified the redundant vertices and we can reduce the graph to a $\mathcal{O}(k^2)$ kernel.

We now want to find k vertices in the kernel that can form the internals of a spanning tree. We will in a brute force manner test every such k-set, there are at most $\binom{k^2}{k}$ such sets. By Stirling's observation that $n^{\frac{n}{2}} < n! < n^n$ we have that $\binom{k^2}{k}$ is less than $k^{\frac{3}{2}k}$. Note that this can be rewritten as $2^{1.5k \log k}$. We now have to verify if these k vertices can be used as the internal vertices of a spanning tree. To do this we try every possible construction of a tree T with these k vertices, by Cayley's formula there are no more than k^{k-2} such trees. This, again, can be rewritten as $(2^{k \log k - 2 \log k})$. Then we test whether or not each leaf in T can be assigned at least one vertex in the remaining kernel as its leaf. This is equivalent to testing if the leaves and the remaining kernel have a perfect bipartite matching, which can be done in time $\mathcal{O}(\sqrt{|V|} \cdot |E|)$. In this particular bipartite subset there are not more than $\mathcal{O}(k^3)$ edges giving us a total of $\mathcal{O}(k^4)$ for the matching. Thus for each k-set we can verify if it is a valid solution in $2^{k \log k} \cdot k^2$ time.

The total running time of the algorithm is $\mathcal{O}(2^{2.5k \log k} k^2 + |V||E|)$.

8.7 ANOTHER PATH(WIDTH) TO SUCCESS

If we cannot use crown-decompositions to reduce the graph efficiently, we can sometimes make use of the fact that the independent structure allows an easy path-decomposition as well. The notion of pathwidth was introduced by Robertson and Seymour [RS83].

Definition 8.7.1 *A* path decomposition *of a graph $G = (V, E)$ is a sequence $(X_1, X_2, \dots, X_r)$ of subsets of V such that:*

1. $\bigcup_{1 \le i \le r} X_i = V$.
2. *For all $vw \in E$, there is an i such that $1 \le i \le r$ and $v, w \in X_i$.*
3. *For all $1 \le i_0 \le i_1 \le i_2 \le r$, we have $X_{i_0} \cap X_{i_2} \subseteq X_{i_1}$.*

The width *of a path decomposition $(X_1, X_2, \dots, X_r)$ is $\max_{1 \le i \le r} |X_i| - 1$. The* pathwidth *of a graph is the minimum width over its path decompositions.*

If we have an independent set I of size $n - g(k)$ we can create a path decomposition with width $g(k)$ in the following manner. Let $I_1, I_2, \dots$ be an arbitrary ordering of I. The path decomposition is then the sequence of subsets $B_j = \overline{I} \cup I_j$. It is easy to convince oneself that this construction satisfies the requirements of a path decomposition.

To give an example where this is useful, consider the parametric dual of k-DOMINATING SET, namely k-NONBLOCKER. (Does $G = (V, E)$ have a subset V' of size k, such that every element of V' has at least one neighbor in $V \setminus V'$?).

Lemma 8.7.1 k-NONBLOCKER *can be solved in time $\mathcal{O}(3^k + n^{\mathcal{O}(1)})$.*

To show this observation, we first compute a *maximal* independent set I. The complement of I, $\overline{I} = V \setminus I$ is a nonblocking set. Thus either $|\overline{I}| < k$ or G has a k-NONBLOCKER. We can then compute a path decomposition with pathwidth k. Now, using the algorithm introduced by Telle and Proskurowski [PT93] and further improved by Alber and Niedermeier [AN02] we can compute a minimum dominating set (and thus maximal nonblocking set) in time $\mathcal{O}(3^k + n^\alpha)$. The above algorithm actually solves the problem for the more general treewidth decomposition in time $\mathcal{O}(4^k + n^\alpha)$, but since this is a path decomposition we can avoid the costly functions combining subtrees of the decompositions. This result improves on the running time of McCartin's algorithm [McC03], which obtains a $\mathcal{O}(4^k + n^\alpha)$ algorithm by using a very different technique.

8.8 CONCLUSIONS AND FURTHER APPLICATIONS TO INDEPENDENT SET STRUCTURES

In this paper we have given a fixed parameter algorithm for k-INTERNAL SPANNING TREE. The algorithm runs in time $\mathcal{O}(2^{2.5k \log k} \cdot k^2 + |V||E|)$, which is the best currently known for this problem. A natural question is whether or not the there is a $2^{\mathcal{O}(k)}$ algorithm for the problem.

We also give a 2-approximation algorithm for the problem. This could be further improved, and the same idea could be used to find more approximation algorithms for other related problems. We would like to note that a limited number of experiments suggest that this algorithm is a very good heuristic.

We have shown the remarkable structural bindings between k-INTERNAL SPANNING TREE and $(n-k)$-INDEPENDENT SET in Corollaries 8.5.2 and 8.5.3. We believe that similar structural bindings exist between INDEPENDENT SET/VERTEX COVER (k-Vertex Cover is of course equivalent to $(n-k)$-INDEPENDENT SET) and other fixed-parameter tractable problems. We are confident that this inherent structure can be used to design potent algorithms for these problems, especially when combined with constructive polynomial time algorithms that produce either an independent set or a solution for the problem in question. Crown decompositions seem to be a natural companion as it has shown itself useful in reducing independent sets in a range of problem [FHRST04, PS04, MPS04, DFRS04, CFJ04].

We also show how the independent set structure allows an easy path decomposition and show that this is useful for k-NONBLOCKER where we improve upon the existing FPT-algorithms.

If large independent sets are the targets, but no such polynomial *either/or* algorithm can be found, we may still use the quite practical FPT VERTEX COVER-algorithm to find the vertex cover structure. The current state of the art algorithm for VERTEX COVER runs in time $\mathcal{O}(1.286^k + n)$ [CKJ01] and has been proven useful in implementations by groups at Carleton University in Ottawa and the University of Tennessee in Knoxville for exact solutions for values of n and k up to 2,500 [L03]. We believe that exploiting vertex cover/independent set structure may be a powerful tool for designing algorithms for other fixed parameter tractable problems for which structural bindings with INDEPENDENT SET exist. For example, we suspect that the parameterized versions of MAX LEAF SPANNING TREE, MINIMUM INDEPENDENT DOMINATING SET and MINIMUM PERFECT CODE are very likely to fall into this class of problems.

BIBLIOGRAPHY

[A03] F. Abu-Khzam. Private communication.

[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.

[AN02] J. Alber and R. Niedermeier. Improved tree decomposition based algorithms for domination-like problems. *Proceedings of the 5th Latin American Theoretical INformatics (LATIN 2002)*, number 2286 in Lecture Notes in Computer Science, pages 613–627, Springer (2002).

[B98] S. Buss, *listed as private communication in the book Parameterized Complexity*

[CFJ03] B. Chor, M. Fellows, D. Juedes. Private communication concerning manuscript in preparation.

[CFJ04] B. Chor, M. Fellows, D. Juedes. Linear Kernels in Linear Time, or How to Save k Colors in $\mathcal{O}(n^2)$ steps. To appear in proceedings 30th Workshop on Graph Theoretic Concepts in Computer Science (WG '04), Springer Lecture Notes in Computer Science, (2004).

[CCDF97] Liming Cai, J. Chen, R. Downey and M. Fellows. The parameterized complexity of short computation and factorization. *Archive for Mathematical Logic* 36 (1997), 321-338.

[CKJ01] J. Chen, I. Kanj, and W. Jia. Vertex cover: Further Observations and Further Improvements. *Journal of Algorithms* Volume 41, 280-301 (2001).

[CLR90] T.H.Cormen, C.E.Leierson, R.L.Rivest, *Introduction to Algorithms,* MIT Press.

[DF98] R. Downey and M. Fellows. *Parameterized Complexity* Springer-Verlag (1998).

[DFRS04] F. Dehne, M. Fellows, F. Rosamond, P.Shaw. Greedy Localization, Iterative Compression and Modeled Crown Reductions: New FPT Techniques and Improved Algorithms for Max Set Splitting and Vertex Cover. *To Appear at IWPEC04* Springer Lecture Notes in Computer Science, (2004).

[DFS99] R. Downey, M. Fellows and U. Stege. Parameterized complexity: a framework for systematically confronting computational intractability. *Contemporary Trends in Discrete Mathematics* (R. Graham, J. Kratochvil, J. Nesetril and F. Roberts, eds.), *AMS-DIMACS Series in Discrete Mathematics and Theoretical Computer Science* 49 (1999), 49-99.

[F03] M.Fellows. Blow-ups, Win/Wins and Crown Rules: Some New Directions in *FPT*. *Proceedings WG 2003*, Springer Verlag LNCS 2880, pages 1-12, 2003.

[FHRST04] M.Fellows, P.Heggernes, F.Rosamond, C. Sloper, J.A.Telle, Finding k disjoint triangles in an arbitrary graph. To appear in proceedings *30th Workshop on Graph Theoretic Concepts in Computer Science (WG ’04)*, Springer Lecture Notes in Computer Science, (2004).

[FMRS01] M. Fellows, C. McCartin. F. Rosamond and U. Stege. Spanning Trees with Few and Many Leaves. *To appear*

[GMM94] G. Galbiati, F. Maffioli, and A. Morzenti. A Short Note on the Approximability of the Maximum Leaves Spanning Tree Problem. *Information Processing Letters* 52 (1994), 45–49.

[GMM97] G. Galbiati, A. Morzenti and F. Maffioli. On the Approximability of some Maximum Spanning Tree Problems. *Theoretical Computer Science* 181 (1997), 107–118.

[GJ79] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.*W.H. Freeman, San Francisco, 1979.

[KR00] Subhash Khot and Venkatesh Raman. *Parameterized Complexity of Finding Hereditary Properties*. Proceedings of COCOON. Theoretical Computer Science (COCOON 2000 special issue)

[L03] M. Langston. Private communication.

[LR98] H.-I. Lu and R. Ravi. Approximating Maximum Leaf Spanning Trees in Almost Linear Time. *Journal of Algorithms* 29 (1998), 132–141.

[MPS04] L. Mathieson, E. Prieto, P. Shaw. Packing Edge Disjoint Triangles: A Parameterized View. *To Appear IWPEC 04*, Springer Lecture Notes in Computer Science, (2004).

[McC03] Catherine McCartin. Ph.D. dissertation in Computer Science, Victoria University, Wellington, New Zealand, (2003).

[NR99b] R. Niedermeier and P. Rossmanith. Upper Bounds for Vertex Cover Further Improved. In C. Meinel and S. Tison, editors, *Proceedings of the 16th Symposium on Theoretical Aspects of Computer Science*, number 1563 in Lecture Notes in Computer Science, Springer-Verlag (1999), 561–570.