Contents lists available at SciVerse ScienceDirect

Journal of Computer and System Sciences

www.elsevier.com/locate/jcss

A linear vertex kernel for maximum internal spanning tree ☆

Fedor V. Fomin [a], Serge Gaspers [b,*], Saket Saurabh [c], Stéphan Thomassé [d]

[a] *Department of Informatics, University of Bergen, Postboks 7803, 5020 Bergen, Norway*
[b] *Institute of Information Systems (184/3), Vienna University of Technology, Favoritenstraße 9-11, A-1040 Vienna, Austria*
[c] *Institute of Mathematical Sciences, CIT Campus, Taramani, 600 113 Chennai, India*
[d] *LIP, ENS Lyon, 46 allée d'Italie, 69364 Lyon Cedex 07, France*

ARTICLE INFO

Article history:
Received 10 December 2009
Received in revised form 3 March 2012
Accepted 16 March 2012
Available online 26 March 2012

Keywords:
Algorithm
Crown decomposition
Kernelization
Parameterized complexity
Preprocessing

ABSTRACT

We present a polynomial time algorithm that for any graph G and integer $k \geqslant 0$, either finds a spanning tree with at least k internal vertices, or outputs a new graph G_R on at most $3k$ vertices and an integer k' such that G has a spanning tree with at least k internal vertices if and only if G_R has a spanning tree with at least k' internal vertices. In other words, we show that the Maximum Internal Spanning Tree problem parameterized by the number of internal vertices k has a $3k$-vertex kernel. Our result is based on an innovative application of a classical min–max result about hypertrees in hypergraphs which states that "a hypergraph H contains a hypertree if and only if H is partition-connected."

1. Introduction

In the Maximum Internal Spanning Tree problem (MIST), we are given a graph G and the task is to find a spanning tree of G with a maximum number of internal vertices. MIST is a natural generalization of the Hamiltonian Path problem because an n-vertex graph has a Hamiltonian path if and only if it has a spanning tree with $n-2$ internal vertices.

In this paper we study a parameterized version of MIST. Parameterized decision problems are defined by specifying the input (I), the parameter (k), and the question to be answered. A parameterized problem that can be solved in time $f(k)|I|^{O(1)}$, where f is a function of k alone is said to be *fixed parameter tractable* (FPT). The natural parameter k for MIST is the number of internal vertices in the spanning tree and the parameterized version of MIST, p-Internal Spanning Tree or p-IST for short, is for a given graph G and integer k, decide if G contains a spanning tree with at least k internal vertices. It follows from Robertson and Seymour's Graph Minors theory that p-IST is FPT [13]. Indeed, the property of not having a spanning tree with at least k internal vertices is closed under taking minors, and thus such graphs can be characterized by a finite set of forbidden minors. One of the consequences of the Graph Minors theory is that every graph property characterized by a finite set of forbidden minors is FPT, and thus p-IST is FPT. These arguments are however not constructive. The first constructive algorithm for p-IST is due to Prieto and Sloper [15] and has running time $2^{4k \log k} \cdot n^{O(1)}$. Recently this result was improved by Cohen et al. [3] who solved a more general directed version of the problem in time $49.4^k \cdot n^{O(1)}$. When the input graph is cubic, p-IST can be solved in time $2.1364^k n^{O(1)}$ [4]. In this paper we study p-IST from the kernelization viewpoint.

☆ A preliminary version of this paper appeared in the Proceedings of ISAAC 2009 (Fomin et al., 2009) [5].
* Corresponding author.
E-mail addresses: fomin@ii.uib.no (F.V. Fomin), gaspers@kr.tuwien.ac.at (S. Gaspers), saket@imsc.res.in (S. Saurabh), stephan.thomasse@ens-lyon.fr (S. Thomassé).

0022-0000/$ – see front matter © 2012 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.jcss.2012.03.004

A parameterized problem is said to admit a *polynomial kernel* if there is a polynomial time algorithm (where the degree of the polynomial is independent of k), called a *kernelization* algorithm, that reduces the input instance to an instance whose size is bounded by a polynomial $p(k)$ in k, while preserving the answer. This reduced instance is called a $p(k)$ *kernel* for the problem. Let us remark that the instance size and the number of vertices in the instance may be different, and thus for bounding the number of vertices in the reduced graph, the term $p(k)$-*vertex kernel* is often used. While many problems on graphs are known to have polynomial kernels (parameterized by the solution size), there are not so many $O(k)$, or linear-vertex kernels known in the literature. Notable examples include a $2k$-vertex kernel for Vertex Cover [2], a k-vertex kernel for Set Splitting [9], and a $6k$-vertex kernel for Cluster Editing [7].

No linear-vertex kernel for p-IST was known prior to our work. Prieto and Sloper [14] provided an $O(k^3)$-kernel for the problem and then improved it to $O(k^2)$ in [15]. The main result of this paper is that p-IST has a $3k$-vertex kernel. The kernelization of Prieto and Sloper is based on the so-called "Crown Decomposition Method" [1]. Here, we use a different method, based on a min–max characterization of hypergraphs containing hypertrees by Frank et al. [6]. As a corollary of the new kernelization, we obtain an algorithm for solving p-IST running in time $8^k \cdot n^{O(1)}$.

The paper is organized as follows. In Section 2, we provide necessary definitions and facts about graphs and hypergraphs. In Section 3, we give the kernelization algorithm. Section 4 is devoted to the proof of the main combinatorial lemma, which is central to the correctness of the kernelization algorithm.

2. Preliminaries

2.1. Graphs

Let $G = (V, E)$ be an undirected simple graph with vertex set V and edge set E. For any nonempty subset $W \subseteq V$, the subgraph of G induced by W is denoted by $G[W]$. The *neighborhood* of a vertex v in G is $N_G(v) = \{u \in V\colon \{u, v\} \in E\}$, and for a vertex set $S \subseteq V$ we set $N_G(S) = \bigcup_{v\in S} N_G(v) \setminus S$. The degree of vertex v in G is $d_G(v) = |N_G(v)|$. Sometimes, when the graph is clear from the context, we omit the subscripts.

2.2. The hypergraphic matroid

Let $H = (V, E)$ be a hypergraph. A hyperedge $e \in E$ is a subset of V. A subset F of hyperedges is a *hyperforest* if $|\bigcup F'| \geqslant |F'| + 1$ for every subset F' of F, where $\bigcup F'$ denotes the union of vertices contained in the hyperedges of F'. This condition is also called the *strong Hall condition*, where *strong* stands for the extra plus one added to the usual Hall condition. A hyperforest with $|V| - 1$ edges is called a *hypertree*. Lorea proved (see [6] or [10]) that $\mathcal{M}_H = (E, \mathcal{F})$, where $\mathcal{F}$ consists of the hyperforests of H, is a matroid, called the *hypergraphic matroid*. Observe that these definitions are well-known when restricted to graphs.

Shrinking a hyperedge e means replacing it by another hyperedge $e' \subseteq e$. This operation may create multi-hyperedges. Lovász [11] proved that F is a hyperforest if and only if every hyperedge e of F can be shrunk to an edge e' (that is, $e' \subseteq e$ contains two vertices of e) in such a way that the set F' consisting of these shrunken edges forms a forest in the usual sense, i.e., a simple acyclic graph. Observe that if F is a hypertree then its set of shrunken edges F' forms a spanning tree on V.

The *border* of a partition $\mathcal{P} = \{V_1, \ldots, V_p\}$ of V is the set $\delta(\mathcal{P})$ of hyperedges of H which intersect at least two parts of $\mathcal{P}$. A hypergraph is *partition-connected* when $|\delta(\mathcal{P})| \geqslant |\mathcal{P}| - 1$ for every partition $\mathcal{P}$ of V. The following theorem can be found in [6, Corollary 2.6].

Theorem 1. *H contains a hypertree if and only if H is partition-connected.*

The proof of Theorem 1 can be turned into a polynomial time algorithm, that is, given a hypergraph $H = (V, E)$ we can either find a hypertree or find a partition $\mathcal{P}$ of V such that $|\delta(\mathcal{P})| < |\mathcal{P}| - 1$ in polynomial time. For the sake of completeness, we briefly describe a polynomial time algorithm to do this. We do not attempt to optimize the running time of this algorithm. Recall that $\mathcal{M}_H = (E, \mathcal{F})$, where $\mathcal{F}$ consists of the hyperforests of H, is a matroid and hence we can construct a hypertree, if one exists, greedily. We start with an empty forest and iteratively try to grow our current hyperforest by adding new edges. When inspecting a new edge we either reject or accept it in our current hyperforest depending on whether by adding it we still have a hyperforest. The only question is to be able to test efficiently if a given collection of edges forms a hyperforest. In other words, we have to check whether the strong Hall condition holds. This can be done in polynomial time by simply running the well-known polynomial time algorithm for testing the usual Hall condition for every subhypergraph $H \setminus v$, where v is a vertex and $H \setminus v$ is the hypergraph containing all hyperedges $e \setminus v$ for $e \in E$.

We can also shrink the edges of a hypertree to a spanning tree in polynomial time. For this, consider any edge e of the hypertree with more than two vertices (if none exist, we already have our tree). By the result of Lovász [11] mentioned above, one of the vertices $v \in e$ can be deleted from e in such a way that we still have a hypertree. Hence we just find this vertex by checking the strong Hall condition for every choice of $e \setminus v$ where $v \in e$. This implies that we need to apply the algorithm to test the strong Hall condition at most $|V|$ times to obtain the desired spanning tree. Consequently, there exists a polynomial time algorithm which can find a spanning tree whose edges are obtained by shrinking hyperedges of a partition-connected hypergraph.

We now turn to the co-NP certificate, that is, we want to exhibit a partition $\mathcal{P}$ of V such that $|\delta(\mathcal{P})| < |\mathcal{P}| - 1$ when H is not partition-connected. The algorithm simply tries to contract every pair of vertices in $H = (V, E)$ and checks if the resulting hypergraph is partition-connected. When it is not, we contract the two vertices, and recurse. We stop when the resulting hypergraph H' is not partition-connected, and every contraction results in a partition-connected hypergraph. Observe then that if a partition $\mathcal{P}$ of H' is such that $|\delta(\mathcal{P})| < |\mathcal{P}| - 1$ and $\mathcal{P}$ has a part which is not a singleton, then contracting two vertices of this part results in a non-partition-connected hypergraph. Hence, the singleton partition is the unique partition $\mathcal{P}$ of H' such that $|\delta(\mathcal{P})| < |\mathcal{P}| - 1$. This singleton partition corresponds to the partition of H which gives our co-NP certificate.

3. Kernelization algorithm

Let $G = (V, E)$ be a connected graph on n vertices and $k \in \mathbb{N}$ be a parameter. In this section we describe an algorithm that takes G and k as an input, and in time polynomial in the size of G either solves p-IST, or produces a reduced graph G_R on at most $3k$ vertices and an integer $k' \leqslant k$, such that G has a spanning tree with at least k internal vertices if and only if G_R has a spanning tree with at least k' internal vertices. In other words, we show that p-IST has a $3k$-vertex kernel.

The algorithm is based on the following combinatorial lemma, which is interesting on its own. For two disjoint sets $X, Y \subseteq V$, we denote by $B(X, Y)$ the bipartite graph obtained from $G[X \cup Y]$ by removing all edges with both endpoints in X and all edges with both endpoints in Y.

Lemma 2. *If $n \geqslant 3$, and I is an independent set of G of cardinality at least $2n/3$, then there are nonempty subsets $S \subseteq V \setminus I$ and $L \subseteq I$ such that*

(i) *$N(L) = S$, and*
(ii) *$B(S, L)$ has a spanning tree such that all vertices of S and $|S| - 1$ vertices of L are internal.*

Moreover, given a graph on at least 3 vertices and an independent set of cardinality at least $2n/3$, such subsets can be found in time polynomial in the size of G.

The proof of Lemma 2 is postponed to Section 4. Now we give the description of the kernelization algorithm and use Lemma 2 to prove its correctness. The algorithm consists of the following reduction rules.

Rule 1 If $n \leqslant 3k$, then output G and stop. In this case G is a $3k$-vertex kernel. Otherwise proceed with Rule 2.

Rule 2 Choose an arbitrary vertex $v \in V$ and run a DFS (depth first search) from v. If the DFS tree T has at least k internal vertices, then the algorithm has found a solution and stops. Otherwise, because $n > 3k$, T has at least $2n/3 + 2$ leaves, and since all leaves but the root of the DFS tree are pairwise nonadjacent, the algorithm has found an independent set of G of cardinality at least $2n/3$. Proceed with Rule 3.

Rule 3 (reduction) Find nonempty subsets of vertices $S, L \subseteq V$ as in Lemma 2. Add a vertex v_S and make it adjacent to every vertex in $N(S) \setminus L$ and add a vertex v_L and make it adjacent to v_S. Finally, remove all vertices of $S \cup L$. Let $G_R = (V_R, E_R)$ be the new graph and $k' = k - 2|S| + 2$. Go to Rule 1 with $G := G_R$ and $k := k'$.

To prove the soundness of Rule 3, we need the following lemma. Here, S and L are as in Lemma 2. If T is a tree and X a vertex set, we denote by $\mathrm{i}_T(X)$ the number of vertices of X that are internal in T.

Lemma 3. *If G has a spanning tree with k internal vertices, then G has a spanning tree with at least k internal vertices in which all the vertices of S and exactly $|S| - 1$ vertices of L are internal.*

Proof. Let T be a spanning tree of G with k internal vertices. Denote by F the forest obtained from T by removing all edges incident to L. Then, as long as 2 vertices $u, v \in S$ are in the same connected component in F, remove an edge from F that is incident to one of these two vertices and belongs to the u–v path in F. Observe that in F, each vertex from $V \setminus (L \cup S)$ is in the same connected component as some vertex from S. Indeed, we only removed an edge uw incident to a vertex $w \in V \setminus (L \cup S)$ in case $u, v \in S$ and there was a u–v path containing w. After removing uw, w is still in the same connected component as v. Now, obtain the spanning tree T' by adding the edges of a spanning tree of $B(S, L)$ to F in which all vertices of S and $|S| - 1$ vertices of L are internal (see Lemma 2). Clearly, all vertices of S and $|S| - 1$ vertices of L are internal in T'. It remains to show that T' has at least as many internal vertices as T.

Let $U := V \setminus (S \cup L)$. Then, we have that $\mathrm{i}_T(L) \leqslant \sum_{u \in L} d_T(u) - |L|$ as every vertex in a tree has degree at least 1 and internal vertices have degree at least 2. We also have $\mathrm{i}_{T'}(U) \geqslant \mathrm{i}_T(U) - (|L| + |S| - 1 - \sum_{u \in L} d_T(u))$ as at most $|S| - 1 - (\sum_{u \in L} d_T(u) - |L|)$ edges incident to S are removed from F to separate $F \setminus L$ into $|S|$ connected components, one for each vertex of S. Thus,

$$\begin{aligned}
\mathrm{i}_{T'}(V) &= \mathrm{i}_{T'}(U) + \mathrm{i}_{T'}(S \cup L) \\
&\geqslant \mathrm{i}_T(U) - \left(|L| + |S| - 1 - \sum_{u \in L} d_T(u)\right) + \mathrm{i}_{T'}(S \cup L) \\
&= \mathrm{i}_T(U) + \left(\sum_{u \in L} d_T(u) - |L|\right) - |S| + 1 + \mathrm{i}_{T'}(S \cup L) \\
&\geqslant \mathrm{i}_T(U) + \mathrm{i}_T(L) - |S| + 1 + \mathrm{i}_{T'}(S \cup L) \\
&= \mathrm{i}_T(U) + \mathrm{i}_T(L) - \big(|S| - 1\big) + \big(|S| + |S| - 1\big) \\
&= \mathrm{i}_T(U) + \mathrm{i}_T(L) + |S| \\
&\geqslant \mathrm{i}_T(U) + \mathrm{i}_T(L) + \mathrm{i}_T(S) \\
&= \mathrm{i}_T(V).
\end{aligned}$$

This finishes the proof of the lemma. □

Lemma 4. *Rule 3 is sound, $|V_R| < |V|$, and $k' \leqslant k$.*

Proof. We claim first that the resulting graph $G_R = (V_R, E_R)$ has a spanning tree with at least $k' = k - 2|S| + 2$ internal vertices if and only if the original graph G has a spanning tree with at least k internal vertices. Indeed, assume G has a spanning tree with $\ell \geqslant k$ internal vertices. Then, let $B(S, L)$ be as in Lemma 2 and T be a spanning tree of G with ℓ internal vertices such that all vertices of S and $|S| - 1$ vertices of L are internal (which exists by Lemma 3). Because $T[S \cup L]$ is connected, every two distinct vertices $u, v \in N_T(S) \setminus L$ are in different connected components of $T \setminus (L \cup S)$. But this means that the graph T' obtained from $T \setminus (L \cup S)$ by connecting v_S to all neighbors of S in $T \setminus (S \cup L)$ is also a tree in which the degree of every vertex in $N_G(S) \setminus L$ is unchanged. The graph T'' obtained from T' by adding v_L and connecting v_L to v_S is also a tree. Then T'' has exactly $\ell - 2|S| + 2$ internal vertices.

In the opposite direction, if G_R has a tree T'' with $\ell - 2|S| + 2$ internal vertices, then all neighbors of v_S in T'' are in different components of $T'' \setminus \{v_S\}$. By Lemma 2 we know that $B(S, L)$ has a spanning tree T_{SL} such that all the vertices of S and $|S| - 1$ vertices of L are internal. We obtain a spanning tree T of G by considering the forest $T^* = T'' \setminus \{v_S, v_L\} \cup T_{SL}$ and adding edges between different components to make it connected. For each vertex $u \in N_{T''}(v_S) \setminus \{v_L\}$, add an edge uv to T^*, where uv is an edge of G and $v \in S$. By construction we know that such an edge always exists. Moreover, the degrees of the vertices in $N_G(S) \setminus L$ are the same in T as in T''. Thus T is a spanning tree with ℓ internal vertices.

Finally, as $|S| \geqslant 1$ and $|L \cup S| \geqslant 3$, we have that $|V_R| < |V|$ and $k' \leqslant k$. □

Thus Rule 3 compresses the graph and we conclude with the following theorem.

Theorem 5. *p-IST has a $3k$-vertex kernel.*

Corollary 6. *p-IST can be solved in time $8^k \cdot n^{O(1)}$.*

Proof. Obtain a $3k$-vertex kernel for the input graph G in polynomial time using Theorem 5 and run the $2^n n^{O(1)}$ time algorithm of Nederlof [12] on the kernel. □

4. Proof of Lemma 2

In this section we provide the postponed proof of Lemma 2. Let $G = (V, E)$ be a connected graph on n vertices, I be an independent set of G of cardinality at least $2n/3$ and $C := V \setminus I$.

Let Y be a subset of V. A subset $X \subseteq (V \setminus Y)$ has *Y-expansion* c, for some $c > 0$, if for each subset Z of X, $|N(Z) \cap Y| \geqslant c \cdot |Z|$. We first find an independent set $L \subseteq I$ whose neighborhood has L-expansion 2. For this, we need the following result.

Lemma 7. *(See [16].) Let B be a nonempty bipartite graph with vertex bipartition (X, Y) with $|Y| \geqslant 2|X|$ and such that every vertex of Y has at least one neighbor in X. Then there exist nonempty subsets $X' \subseteq X$ and $Y' \subseteq Y$ such that the set of neighbors of Y' in B is exactly X', and such that X' has Y'-expansion 2. Moreover, such subsets X', Y' can be found in time polynomial in the size of B.*

By using Lemma 7, we find nonempty sets of vertices $S' \subseteq C$ and $L' \subseteq I$ such that $N(L') = S'$ and S' has L'-expansion 2.

Lemma 8. *Let $G = (V, E)$ be a connected graph and $L' \subseteq V$ be an independent set such that $S' = N(L')$ has L'-expansion 2. Then there exist nonempty subsets $S \subseteq S'$ and $L \subseteq L'$ such that*

- *$B(S, L)$ has a spanning tree in which all the vertices of L have degree at most 2,*
- *S has L-expansion 2, and*
- *$N(L) = S$.*

Moreover, such sets S and L can be found in time polynomial in the size of G.

Proof. The proof is by induction on $|S'|$. If $|S'| = 1$, the lemma holds with $S := S'$ and $L := L'$. Let $H = (S', E')$ be the hypergraph with edge set $E' = \{N(v) \mid v \in L'\}$. If H contains a hypertree, then it has $|S'| - 1$ hyperedges and we can obtain a tree $T_{S'}$ on S' by shrinking edges. We use this to find a subtree T' of $B(S', L')$ spanning S' as follows: for every edge $e = uv$ of $T_{S'}$ there exists a hyperedge corresponding to it and hence a unique vertex, say w, in L'; we delete the edge $e = uv$ from $T_{S'}$ and add the edges wu and wv to $T_{S'}$. Observe that the resulting subtree T' of $B(S', L')$ has the property that every vertex in T' which is in L' has degree 2 in it. Finally, we extend T' to a spanning tree of $B(S', L')$ by adding the remaining vertices of L' as pending vertices. All this can be done in polynomial time using the algorithm in Section 2.2. Thus S' and L' are the sets of vertices we are looking for. Otherwise, if H does not contain a hypertree, then H is not partition-connected by Theorem 1. Then we can find a partition $\mathcal{P} = \{P_1, P_2, \ldots, P_\ell\}$ of S' such that its border $\delta(\mathcal{P})$ contains at most $\ell - 2$ hyperedges of H in polynomial time. Let b_i be the number of hyperedges completely contained in P_i, where $1 \leqslant i \leqslant \ell$. Then there exists an i, $1 \leqslant i \leqslant \ell$, such that $b_i \geqslant 2|P_i|$. Indeed, otherwise $|L'| \leqslant (\ell - 2) + \sum_{i=1}^{\ell}(2|P_i| - 1) < 2|S'|$, which contradicts that S' has L'-expansion 2. Let $X := P_j$ and $Y := \{w \in L' \mid N(w) \subseteq P_j\}$. We know that $|Y| \geqslant 2|X|$ and hence by Lemma 7 there exists a $S^* \subseteq X$ and $L^* \subseteq Y$ such that S^* has L^*-expansion 2 and $N(L^*) = S^*$. Furthermore, S^* and L^* can be computed from X and Y in polynomial time. Thus, by the induction assumption, there exist $S \subseteq S^*$ and $L \subseteq L^*$ with the desired properties. □

Let S and L, be as in Lemma 8. We will prove in the following that there exists a spanning tree of $B(S, L)$ such that all the vertices of S and exactly $|S| - 1$ vertices of L are internal. Note that there cannot be more than $2|S| - 1$ internal vertices in a spanning tree of $B(S, L)$ without creating cycles. By Lemma 8, we know that there exists a spanning tree of $B(S, L)$ in which $|S| - 1$ vertices of L have degree exactly 2.

Consider the bipartite graph B_2 obtained from $B(S, L)$ by adding a copy S_c of S (each vertex in S has the same neighborhood as its copy in S_c and no vertex of S_c is adjacent to a vertex in S). As $|L| \geqslant |S \cup S_c|$ and each subset Z of $S \cup S_c$ has at least $|Z|$ neighbors in L, by Hall's theorem, there exists a matching in B_2 saturating $S \cup S_c$. This means that in $B(S, L)$, there exist two edge-disjoint matchings M_1 and M_2, both saturating S. We refer to the edges from $M_1 \cup M_2$ as the *favorite* edges.

Lemma 9. *$B(S, L)$ has a spanning tree T such that all the vertices of S and $|S| - 1$ vertices of L are internal in T.*

Proof. Let T be a spanning tree of $B(S, L)$ in which all vertices of L have degree at most 2, obtained using Lemma 8. As T is a tree, exactly $|S| - 1$ vertices of L have degree 2 in T. As long as a vertex $v \in S$ is not internal in T, add a favorite edge uv to T which was not yet in T ($u \in L$), and remove an appropriate edge from the tree which is incident to u so that T remains a spanning tree. Vertex v becomes internal and the degree of u in T remains unchanged. As u is only incident to one favorite edge, this rule increases the number of favorite edges in T even though it is possible that some other vertex in S would have become a leaf. We apply this rule until no longer possible. We know that this rule can only be applied at most $|S|$ times. In the end, all the vertices of S are internal and $|S| - 1$ vertices among L are internal as their degrees remain the same. □

To conclude with the proof of Lemma 2, we observe that $S \subseteq C$, $L \subseteq I$ and $N(L) = S$ by the construction of S and L, and by Lemma 9, $B(S, L)$ has a spanning tree in which all the vertices of S and $|S| - 1$ vertices of L are internal.

5. Conclusion

We have designed a polynomial time algorithm transforming an instance (G, k) for MIST into an equivalent instance (G', k') for MIST such that G' has at most $3k$ vertices. Moreover, a spanning tree of G with k internal vertices can be obtained from a spanning tree of G' on k' internal vertices in polynomial time, by replacing vertices v_S and v_L as described by Lemma 4. Since, for an instance (G, k), the reduction rule can be executed whenever a DFS tree of G has at most $|V(G)|/3$ internal nodes, our algorithm also implies a 3-approximation in polynomial time. The best known approximation ratio that can be achieved in polynomial time for this problem is currently 5/3 [8].

We finish by asking whether the kernel size can be improved.

References

[1] F.N. Abu-Khzam, M.R. Fellows, M.A. Langston, W.H. Suters, Crown structures for vertex cover kernelization, Theory Comput. Syst. 41 (3) (2007) 411–430.

[2] J. Chen, I.A. Kanj, W. Jia, Vertex cover: Further observations and further improvements, J. Algorithms 41 (2) (2001) 280–301.

[3] N. Cohen, F.V. Fomin, G. Gutin, E.J. Kim, S. Saurabh, A. Yeo, Algorithm for finding k-vertex out-trees and its application to k-internal out-branching problem, J. Comput. System Sci. 76 (7) (2010) 650–662.

[4] H. Fernau, S. Gaspers, D. Raible, Exact and parameterized algorithms for max internal spanning tree, in: Proceedings of WG 2009, in: Lecture Notes in Comput. Sci., vol. 5911, Springer, 2009, pp. 100–111.

[5] F.V. Fomin, S. Gaspers, S. Saurabh, S. Thomassé, A linear vertex kernel for maximum internal spanning tree, in: Proceedings of ISAAC 2009, in: Lecture Notes in Comput. Sci., vol. 5878, Springer, 2009, pp. 275–282.

[6] A. Frank, T. Király, M. Kriesell, On decomposing a hypergraph into k connected sub-hypergraphs, Discrete Appl. Math. 131 (2003) 373–383.

[7] J. Guo, A more effective linear kernelization for cluster editing, Theory Comput. Sci. 410 (8–10) (2009) 718–726.

[8] M. Knauer, J. Spoerhase, Better approximation algorithms for the maximum internal spanning tree problem, in: Proceedings of WADS 2009, in: Lecture Notes in Comput. Sci., vol. 5664, Springer, 2009, pp. 459–470.

[9] D. Lokshtanov, S. Saurabh, Even faster algorithm for set splitting!, in: Proceedings of IWPEC 2009, in: Lecture Notes in Comput. Sci., vol. 5917, Springer, 2009, pp. 288–299.

[10] M. Lorea, Hypergraphes et matroides, Cah. Cent. étud. Rech. Opér. 17 (1975) 289–291.

[11] L. Lovász, A generalization of König's theorem, Acta Math. Acad. Sci. Hungar. 21 (1970) 443–446.

[12] J. Nederlof, Fast polynomial-space algorithms using Möbius inversion: Improving on Steiner tree and related problems, in: Proceedings of ICALP 2009, in: Lecture Notes in Comput. Sci., vol. 5555, Springer, 2009, pp. 713–725.

[13] N. Robertson, P.D. Seymour, Graph minors—a survey, in: I. Anderson (Ed.), Surveys in Combinatorics, Cambridge Univ. Press, 1985, pp. 153–171.

[14] E. Prieto, C. Sloper, Either/or: Using vertex cover structure in designing FPT-algorithms – the case of k-internal spanning tree, in: Proceedings of WADS 2003, in: Lecture Notes in Comput. Sci., vol. 2748, Springer, 2003, pp. 465–483.

[15] E. Prieto, C. Sloper, Reducing to independent set structure – the case of k-internal spanning tree, Nordic J. Comput. 12 (3) (2005) 308–318.

[16] S. Thomassé, A $4k^2$ kernel for feedback vertex set, ACM Trans. Algorithms 6 (2) (2010) 32:1–32:8.