Even Faster Algorithm for Set Splitting!

Daniel Lokshtanov[*] Saket Saurabh[*]

Abstract

In the p-Set Splitting problem we are given a universe U, a family $\mathcal{F}$ of subsets of U and a positive integer k and the objective is to find a partition of U into W and B such that there are at least k sets in $\mathcal{F}$ that have non-empty intersection with both B and W. In this paper we study p-Set Splitting from the view point of kernelization and parameterized algorithms. Given an instance $(U, \mathcal{F}, k)$ of p-Set Splitting, our kernelization algorithm obtains an equivalent instance with at most $2k$ sets and k elements in polynomial time. Finally, we give a fixed parameter tractable algorithm for p-Set Splitting running in time $O(1.9630^k + N)$, where N is the size of the instance. Both our kernel and our algorithm improve over the best previously known results. Our kernelization algorithm utilizes a classical duality theorem for a connectivity notion in hypergraphs. We believe that the duality theorem we make use of could become an important tool in obtaining kernelization algorithms.

1 Introduction

In the Max Cut problem we are given a graph G with vertex set $V(G)$ and edge set $E(G)$ and asked to find a partitioning of $V(G)$ into W (*white*) and B (*black*) such that the number of edges with one endpoint in W and one in B is maximized. The Max Cut problem is one of Karp's 21 NP-hard problems [13] and also the first problem for which an approximation algorithm using semi-definite programming was obtained [12]. The problem has also been studied from the viewpoint of parameterized algorithms [16, 18].

A natural generalization of Max Cut to *hypergraphs* is the Set Splitting problem, also known as Max Hypergraph 2-Coloring. A *hypergraph* $H = (\mathcal{V}, \mathcal{E})$ consists of a vertex set $\mathcal{V}$ and a set $\mathcal{E}$ of hyperedges. A *hyperedge* $e \in \mathcal{E}$ is a subset of the vertex set $\mathcal{V}$. By $V(e)$ we denote the subset of vertices corresponding to the edge e. In the Set Splitting problem we are given a family $\mathcal{F}$ of sets over a universe U. We say that a partitioning (W, B) of U *splits* a set $S \in \mathcal{F}$ if $S \cap W \neq \emptyset$ and $S \cap B \neq \emptyset$. The objective is to partition U into W and B such that the number of sets in $\mathcal{F}$ that are split is maximized. If the Set Splitting instance $(U, \mathcal{F})$ is viewed as a hypergraph $H = (U, \mathcal{F})$ the objective is to color the vertices of H black or white, maximizing the number of hyperedges containing at least one white and at least one black vertex. It should be noted that Max Cut is the special case of Set Splitting when all sets in $\mathcal{F}$ have cardinality 2. The Set Splitting and (Max) Hypergraph 2-Coloring problems have been studied intensively from a combinatorial as well as an algorithmic viewpoint [1, 2, 5, 6, 7, 15, 17, 21, 22].

We study Set Splitting from the parameterized algorithms perspective. In parameterized algorithms every instance x comes with a *parameter* k and an algorithm for the problem with running time $f(k)n^{O(1)}$ is said to be *fixed parameter tractable*. Formally a parameterized problem Π is a subset of $\Gamma^* \times \mathbb{N}$ for some finite alphabet Γ and an instance of the problem

[*]Department of Infomatics, University of Bergen, Norway. {daniello|saket.saurabh}@ii.uib.no.

History of p-Set Splitting			
Dehne, Fellows and Rosamond	WG 2003	$O(72^k N^{O(1)})$	Deterministic
Dehne, Fellows, Rosamond and Shaw	IWPEC 2004	$O(8^k N^{O(1)})$	Deterministic
Lokshtanov and Sloper	ACiD 2005	$O(2.6499^k N^{O(1)})$	Deterministic
Chen and Lu	COCOON 2007	$O(2^k + N)$	Randomized
Lokshtanov and Saurabh*	2009	$O(1.96^k + N)$	Deterministic

Table 1: List of known results about p-Set Splitting in chronological order. The row marked with $\star$ represents result in the current article.

consists of (x, k), where k is the parameter. The problem Π is said to admit a $g(k)$ *kernel* if there is a polynomial time algorithm that transforms any instance (x, k) to an equivalent instance (x', k') such that $|x'| \leq g(k)$ and $k' \leq g(k)$. If $g(k) = k^{O(1)}$ or $g(k) = O(k)$ we say that Π admits a polynomial kernel and linear kernel respectively. We remark that for most kernels, and in particular all kernels mentioned in this article k' is in fact bounded by k. In the parameterized version of Max Cut, called p-Max Cut the input is a graph G and an integer k and the objective is to partition $V(G)$ into W and B such that at least k edges have one white and one black endpoint. Similarly, in p-Set Splitting an input instance is a family $\mathcal{F}$ of sets over a universe U and an integer k. The objective is to find a partitioning (W, B) of U that splits at least k sets. Throughout this paper we denote the size of an instance $(U, \mathcal{F}, k)$ of p-Set Splitting by N.

Related Work. The fastest known parameterized algorithm for the p-Max Cut problem has running time $O(1.2418^k + |V(G)| + |E(G)|)$ [18] and the smallest kernel has k vertices and $2k$ edges [16, 18]. In fact, bounding the number of vertices by k is easy - any connected graph G has a spanning tree with $|V(G)| - 1$ edges. Since trees are bipartite we can partition $V(G)$ into (W, B) such that all edges in the spanning tree have one endpoint in W and one in B. Hence, if $|V(G)| - 1 \geq k$ we can immediately answer yes. This immediately yields a $O(2^k + |V(G)| + |E(G)|)$ time algorithm for the problem.

On the other hand, until this work, no deterministic algorithm with running time $O(2^k + N)$ was known for p-Set Splitting, even though the problem is quite well-studied in parameterized algorithms. Dehne, Fellows and Rosamond [4] initiated the study of p-Set Splitting and gave an algorithm running in time $O(72^k N^{O(1)})$. They also provided kernel for the problem with at most $2k$ sets in the family. Later Dehne, Fellows, Rossmand and Shaw [5] obtained an algorithm with running time $O(8^k N^{O(1)})$. Continuing this chain of improvement Lokshtanov and Sloper [14] gave an algorithm with running time $O(2.65^k N^{O(1)})$ and obtained a kernel with both universe size and family size at most $2k$. Finally, Chen and Lu [2] provided a randomized algorithm with running time $O(2^k + N)$ for a weighted version of problem. We refer to Table 1 for a quick reference on the history of the p-Set Splitting problem.

Our Results. The first part of this article is devoted to generalizing the simple kernelization algorithm for p-Max Cut to hypergraphs and giving a kernel with at most $2k$ sets and k elements for the p-Set Splitting problem. To this end, we make a detour and introduce notions of *spanning trees* and *strong cut-sets* in a hypergraph. The purpose of these notions is to be able to generalize the statement "every connected graph has a spanning tree" to "every hypergraph without a strong cut-set has a spanning tree". Making this generalization turned out to be non-trivial and required using a duality theorem for a connectivity notion in hypergraphs. Our first result is the following.

Theorem 1. *p-Set Splitting admits a kernel with $2k$ sets and k elements.*

On the face of it Theorem 1 could like a simple improvement over the previous known kernel with $2k$ sets and $2k$ elements but it is not. Observe that Theorem 1 yields as a corollary the *fastest* known deterministic algorithm for p-Set Splitting running in time $O(2^k + N)$. In the last section of this article we break the "2^k barrier" and give a $O(1.9630^k + N)$ time algorithm for problem using memoization and the *Measure & Conquer* paradigm.

Theorem 2. *There is an $O(1.9630^k + N)$ time algorithm for the p-Set Splitting problem*

The *Measure & Conquer* paradigm has been extensively applied to obtain faster exact exponential time algorithms. We refer to [9, 10] for a reference on *Measure & Conquer*. Even though *Measure & Conquer* has been applied to several problems to obtain exact exponential time algorithms, its applicability in obtaining parameterized algorithm has been limited to an algorithm for 3-Hitting Set by Wahlström [20]. Our fixed parameter algorithm for p-Set Splitting provides another example of application of *Measure & Conquer* in parameterized algorithms.

Throughout this paper for an undirected graph G by $V(G)$ we denote its vertex set and by $E(G)$ we denote its edge set. For a subset $V' \subseteq V(G)$, by $G[V']$ we mean the subgraph of G induced on V'.

2 Kernelization Algorithm

In this seciton we first give an algorithmic version of a classical duality theorem for a connectivity notion in hypergraphs. Next, we use this duality result to get a kernel for p-Set Splitting with at most $2k$ sets and k elements.

2.1 A Duality Theorem for Hypergraph Connectivity

We begin with a few definitions related to hypergraphs. With every hypergraph $H = (\mathcal{V}, \mathcal{E})$ we can associate the following graph: The *primal graph*, also called the *Gaifmann graph*, $P(H)$ has the same vertices $\mathcal{V}$ as H and, two vertices $u, v \in \mathcal{V}$ are connected by an edge in $P(H)$ if there is a hyperedge $e \in \mathcal{E}$, such that $\{u, v\} \subseteq V(e)$. We say that H is *connected* or has r components if the corresponding primal graph $P(H)$ is connected or has r components. Now we define the notions of *strong cut-sets* and forests in hypergraphs.

Definition 1 (Strong Cut-Set)**.** *A subset $X \subseteq \mathcal{E}$ is called a strong cut-set if the hypergraph $H' = (\mathcal{V}, \mathcal{E} \setminus X)$ has at least $|X| + 2$ connected components.*

Definition 2 (Hypergraph Forest)**.** *A forest $\mathcal{F}$ of a hypergraph H is a pair (F, g) where F is a forest on the vertex set $\mathcal{V}$ with edge set $E(F)$ where F is a forest in normal graph sense and $g : E(F) \to \mathcal{E}$ is an injective map such that for every $uv \in E(F)$ we have $\{u, v\} \subseteq V(g(uv))$. The number of edges in $\mathcal{F}$ is $|E(F)|$.*

Observe that if a forest $\mathcal{F}$ has $|\mathcal{V}| - 1$ edges then F is a spanning tree on $\mathcal{V}$. In this case we say that $\mathcal{F}$ is a *spanning tree* of H. Frank, Király, and Kriesell proved the following duality result relating spanning trees and strong cut-set in hypergraphs [11, Corollary 2.6].

Proposition 1 ([11])**.** *A hypergraph H contains a hypertree if and only if H does not have a strong cut-set.*

We give an algorithmic version of Proposition 1 in Theorem 3 which is central to our kernelizatio algorithm. We start with a few observations about forest in hypergraphs and a definition useful for the proof of Theorem 3. Given a forest $\mathcal{F} = (F, g)$ we classify the edges of $\mathcal{E}$ as follows. An edge $e \in \mathcal{E}$ is

- a *forest edge* if there exists an edge f in $E(F)$ such that $g(f) = e$;
- a *cut edge* if there exist two connected components C_1 and C_2 of F such that $V(e) \cap V(C_1) \neq \emptyset$ and $V(e) \cap V(C_2) \neq \emptyset$.
- an *unused edge* if there does not exist an edge f in $E(F)$ such that $g(f) = e$; that is e is not in the image of the map g.

We remark that an edge e can be a forest edge as well as a cut edge at the same time. Similarly an edge can be a cut edge as well as an unused edge at the same time.

Definition 3. *For a hypergraph $H = (\mathcal{V}, \mathcal{E})$, a forest $\mathcal{F} = (F, g)$ and $e_1, e_2 \in \mathcal{E}$, we say that an edge e_2 follows e_1 if e_1 is a forest edge of F and e_2 is a cut edge with respect to $\mathcal{F}' = (F', g')$ where $F' = (\mathcal{V}, E(F) \setminus \{g^{-1}(e_1)\})$ and $g'(f) = g(f)$ for $f \in E(F')$.*

We are now in position to state the algorithm version of Proposition 1 which will be used later in our kernelization algorithm.

Theorem 3. *There is a polynomial time algorithm that given a connected hypergraph $H = (\mathcal{V}, \mathcal{E})$ and a forest $\mathcal{F} = (F, g)$ of H such that $|E(F)| < |\mathcal{V}| - 1$ finds either a forest $\mathcal{F}' = (F', g')$ of H with $|E(F')| \geq |E(F)| + 1$ or a strong cut-set X of H.*

Proof. Given a hypergraph $H = (\mathcal{V}, \mathcal{E})$ and a forest $\mathcal{F} = (F, g)$ of H, a sequence of hyperedges $\mathcal{L}(H, \mathcal{F}) = e_1 e_2 \ldots e_t$ such that $e_i \in \mathcal{E}$ is called an *augmenting sequence* if (a) e_1 is a cut edge (b) e_{i+1} follows e_i for all $1 \leq i \leq t$ and (c) e_t is an unused edge.

We first prove that if there exists an augmenting sequence with respect to a forest $\mathcal{F} = (F, g)$ of H then there exists a forest $\mathcal{F}' = (F', g')$ of H with $|E(F')| \geq |E(F)| + 1$. We prove this by induction on the length of the shortest augmenting sequence t. If $t = 1$ then $\mathcal{L}(H, \mathcal{F}) = e_1$. In this case e_1 is a cut edge as well as an unused edge. Since e_1 is a cut edge there exists two connected components C_1 and C_2 of F such that $V(C_1) \cap V(e_1) \neq \emptyset$ and $V(C_2) \cap V(e_1) \neq \emptyset$. Let $u \in V(C_1) \cap V(e_1)$ and $v \in V(C_2) \cap V(e_1)$. Now define $F' = (\mathcal{V}, E(F) \cup \{uv\})$ and $g' : E(F') \rightarrow \mathcal{E}$ as $g'(f) = g(f)$ if $f \in E(F)$ and $g'(uv) = e_1$. Since we have added an edge between two distinct components of F, we have that the F' is also a forest and has one more edge than that in F.

Assume now that $t \geq 2$ and that if we are given a forest $\mathcal{F} = (F, g)$ of H and a shortest augmenting sequence $\mathcal{L}(H, \mathcal{F})$ of length at most $t' < t$ then there exists a forest $\mathcal{F}' = (F', g')$ of H with $|E(F')| \geq |E(F)| + 1$. Let $\mathcal{F} = (F, g)$ be a forest and $\mathcal{L}(H, \mathcal{F}) = e_1 e_2 \ldots e_t$ be a shortest augmenting sequence of F with length t. Observe that e_1 is a cut edge. Hence there exist two connected components C_1 and C_2 of F such that $V(C_1) \cap V(e_1) \neq \emptyset$ and $V(C_2) \cap V(e_1) \neq \emptyset$. Let $u \in V(C_1) \cap V(e_1)$ and $v \in V(C_2) \cap V(e_1)$. Furthermore e_1 is a forest edge and hence we let $f \in E(F)$ such that $g(f) = e_1$. Now we construct a forest $\mathcal{F}^* = (F^*, g')$ as follows. We let $F^* = (\mathcal{V}, E(F) \cup \{uv\} \setminus \{f\})$ and $g' : E(F^*) \rightarrow \mathcal{E}$ as $g'(f) = g(f)$ if $f \in E(F)$ and $g'(uv) = e_1$. Now we show that the $\mathcal{L}(H, \mathcal{F}^*) = e_2 \ldots e_t$ is an augmenting sequence of length at most $t - 1$ for $\mathcal{F}^*$.

To show this we will use the following properties of $\mathcal{L}(H, \mathcal{F})$, which follows by the fact that $\mathcal{L}(\mathcal{H}, \mathcal{F})$ is of shortest length:

- if $t > 1$ then e_1, e_2, …, e_{t-1} are forest edges;
- if $t > 1$ then e_2, …, e_{t-1} are not cut edges with respect to $\mathcal{F}$.

Since e_2 follows e_1 and e_2 is not a cut edge of $\mathcal{F}$ we have that e_2 is a cut edge with respect to $\mathcal{F}^*$. Since $g(E(F)) = g'(E(F^*))$ we have that e_t is an unused edge of $\mathcal{F}^*$. The only thing that remains to be proved is that e_{j+1} follows e_j for $j \in \{2, \dots, t-1\}$ with respect to $\mathcal{F}^*$. Let e_j be a hyperedge, $j \in \{2, \dots, t-1\}$, and let u and v be two vertices in $V(e_{j+1})$ that lie in different connected components of $F \setminus g^{-1}(e_j)$. We prove that u and v lie in different connected components of $F^* \setminus g'^{-1}(e_j)$. Suppose not, then there is a path P from u to v in $F^* \setminus g'^{-1}(e_j)$. If P does not contain $g'^{-1}(e_1)$ then P is a path from u to v in $F \setminus g^{-1}(e_j)$ because $g'^{-1}(e_j) = g^{-1}(e_j)$ and $g'^{-1}(e_1)$ is the only edge in F^* that is not in F. This contradicts that e_{j+1} follows e_j with respect to $\mathcal{F}$. If P contains $g'^{-1}(e_1)$ then u and v must lie in *different* connected components of $F^* \setminus g'^{-1}(e_1) = F \setminus g^{-1}(e_1)$. But e_{j+1} does not follow e_1 with respect to $\mathcal{F}$, and hence u and v must lie in the *same* connected component $F \setminus g^{-1}(e_1)$, a contradiction. Thus, we conclude that e_{j+1} follows e_j with respect to $\mathcal{F}^*$.

Hence we have shown that $\mathcal{L}(H, \mathcal{F}^*) = e_2 \dots e_t$ is an augmenting sequence of length at most $t-1$ for $\mathcal{F}^*$. This implies that the $\mathcal{F}^*$ has a shortest augmenting sequence of length at most $t-1$ and hence by the induction hypothesis this implies that there exists a forest $\mathcal{F}' = (F', g')$ of H with $|E(F')| \geq |E(F^*)| + 1 = |E(F)| + 1$.

For the other direction of the proof we show that if we do not have an augmenting sequence then we have a strong cut-set. We say that a hyperedge e is *reachable* from a hyperedge $e^* \in Y$ if there exists a sequence of hyperedges $e^* e_1 \dots e_l e$ and e_1 follows e^*, e_{i+1} follows e_i for $i \in \{1, \dots, l-1\}$ and e follows e_l. Let Y be the set of cut edges with respect to $\mathcal{F}$ and X be the set of all hyperedges containing Y and all those hyperedges which are reachable from a hyperedge in X. We claim that X is the desired strong cut-set. Let $H' = (\mathcal{V}, \mathcal{E} \setminus X)$ be the hypergraph obtained from H by removing the hyperedges from X. Now we show $P(H')$ has at least $|X| + 2$ connected components. Observe that all the edges in X are forest edges, otherwise there would exist an augmenting sequence. Let $X^{-1} = \{g^{-1}(x) \mid x \in X\}$. The forest F which we started with has at least two components and hence when we remove the edges from X^{-1} it has at least $|X|+2$ connected components. We show that for every connected component C, of $F' = (\mathcal{V}, E(F) \setminus X^{-1})$, $P(H')[V(C)]$ is a connected component. Suppose not, then there exist two connected components C_1 and C_2 of F' such that there exists a hyperedge $e \notin X$ such that $V(e) \cap V(C_1) \neq \emptyset$ and $V(e) \cap V(C_2) \neq \emptyset$. Let $u \in V(e) \cap V(C_1)$ and $v \in V(e) \cap V(C_2)$. Since e is not a cut edge, u and v are in the same component of F. Since u and v are not in the same component of F' there is a hyperedge $e' \in X$ such that u and v are in different components of $F \setminus g^{-1}(e')$. Hence e follows e', a contradiction.

We have proved that for a connected hypergraph $H = (\mathcal{V}, \mathcal{E})$ and a forest $\mathcal{F} = (F, g)$ of H such that $|E(F)| < |\mathcal{V}| - 1$, either there exists a forest $\mathcal{F}' = (F', g')$ of H with $|E(F')| \geq |E(F)| + 1$ or there exists a strong cut-set X of H. We can make our proof constructive if we have a way to find a shortest augmenting path with respect to $\mathcal{F}$. In what follows we show how to find a shortest augmenting path corresponding to $\mathcal{F}$ by reducing this to finding a shortest path in an auxiliary directed graph. We make a graph G' with vertex set $V(G')$ having a vertex v_e for every hyperedge $e \in \mathcal{E}$. We make an edge from v_e to v_f if f follows e with respect to $\mathcal{F}$. Hence to find a shortest augmenting sequence it is sufficient to do a breadth first search in G' starting from $\{v_e \in V(G') \ : \ e \text{ is a cut edge}\}$ and checking whether a vertex v_f corresponding to an unused hyperedge is reached. It is clear that this procedure takes polynomial time. If we find an augmenting sequence then we can find the desired forest

$\mathcal{F}'$ in polynomial time and if we do not find an augmenting sequence then we can find the desired cut-set X as described in the proof in polynomial time. $\square$

2.2 Kernel for Set Splitting

In this section we show how to utilize Theorem 3 to give a kernel with $2k$ sets and k elements for the p-Set Splitting problem. Since Theorem 3 is phrased in terms of hypergraphs, it is useful to view the p-Set Splitting instance $(U, \mathcal{F}, k)$ as a hypergraph $H = (U, \mathcal{F})$ and integer k. We start by showing that if H contains a strong cut-set, then the instance $(U, \mathcal{F}, k)$ can be reduced.

Definition 4. *Let $f : \mathcal{V} \rightarrow \{0,1\}$ be a function from set of vertices of the hypergraph H to the set $\{0,1\}$. Then Split(f) is the set of hyperedges such that for every hyperedge $e \in Split(f)$ there exist vertices $u, v \in V(e)$ such that $f(u) = 0$ and $f(v) = 1$.*

Lemma 1. *There is a polynomial time algorithm that given a strong cut-set X of a connected hypergraph $H = (\mathcal{V}, \mathcal{E})$ finds a cut-set $X' \subseteq X$ such that $X' \neq \emptyset$ and $(H = (\mathcal{V}, \mathcal{E}), k)$ is a yes instance of* p-Set Splitting *if and only if $(H' = (\mathcal{V}, \mathcal{E} \setminus X'), k - |X'|)$ is a yes instance of* p-Set Splitting.

Proof. Let $H^* = (\mathcal{V}, \mathcal{E} \setminus X)$ and let $|X| = t$. By assumption, X is a strong cut-set and hence the primal graph $P(H^*)$ has at least $t + 2$ connected components. Let the connected components of $P(H^*)$ be $\mathcal{C} = \{C_1, \ldots, C_q\}$ where $q \geq t + 2$ and $X = \{e_1, \ldots, e_t\}$. We construct an auxiliary bipartite graph $\mathcal{B}$ with vertex set $A \cup B$ with a vertex $a_i \in A$ for every edge $e_i \in X$ and a vertex $b_i \in B$ for every connected component $C_i \in \mathcal{C}$. There is an edge $a_i b_j$ if $V(e_i) \cap V(C_j) \neq \emptyset$.

We prove the statement of the lemma by induction on $|X|$. For the base case we assume that $|X| = 1$ and $X = \{e_1\}$. In particular, we show that given any $f : \mathcal{V} \rightarrow \{0,1\}$ there exists a function $g : \mathcal{V} \rightarrow \{0,1\}$ such that $Split(g) = Split(f) \cup \{e_1\}$ which will prove the desired assertion. If $e_1 \in Split(f)$ the statement follows, so assume that $e_1 \notin Split(f)$. Since $P(H)$ is connected we have that $a_1 b_j$, $j \in \{1, \ldots, q\}$ are edges in $\mathcal{B}$. Let $g : \mathcal{V} \rightarrow \{0,1\}$ be such that $g(v) = f(v)$ if and only if $v \notin C_1$. That is, for all vertices in C_1, g flips the assignment given by f. Observe that $e_1 \in Split(g)$ since $V(e_1)$ contains a vertex $u \in C_1$ and a vertex $v \in C_2$. Since $f(u) = f(v)$, $g(u) \neq g(v)$ and hence $e_1 \in Split(g)$. For every edge in $Split(f)$ we have that $V(e)$ is completely contained in one of the components and hence, $e \in Split(f)$ implies $e \in Split(g)$. This completes the proof for the base case. So we assume that $|X| \geq 2$ and that the statement of the lemma holds for all X' satisfying the conditions of the lemma and $|X'| < |X|$. In inductive step we consider two cases:

(a) there does not exist a matching in $\mathcal{B}$ which saturates A; or

(b) there is a matching saturating A in $\mathcal{B}$.

In Case (a) by Hall's theorem we know that there exists a subset $A' \subseteq A$, $A' \neq \emptyset$ such that $|A'| > |N(A')|$ and such a set can be found in polynomial time. We claim that $X' = X \setminus \{e_j \mid a_j \in A'\}$ is a strong cut-set and is of smaller size than X. It is clear that $|X'| < |X|$ as $A' \neq \emptyset$. We now show that X' is indeed a strong cut-set. Let $\mathcal{C}' = \mathcal{C} \setminus \{C_j \mid b_j \in N(A')\}$. Observe that in $H' = (\mathcal{V}, \mathcal{E} \setminus X')$, every $C_i \in \mathcal{C}'$ is a connected component. The size of $\mathcal{C}'$ is bounded as follows

$$|\mathcal{C}'| = |\mathcal{C}| - |N(A')| \geq (t + 2) - |N(A')| > (t + 2) - |A'| = t - |A'| + 2 = |X'| + 2,$$

and hence X' is indeed a strong cut-set. In this case the statement of the lemma follows from the induction hypothesis as $|X'| < |X|$.

For Case (b) we assume that we have a matching M saturating A. Without loss of generality let M be $a_1b_1, \ldots, a_tb_t$. Let $U = \{b_{t+1}, \ldots, b_q\}$ be the set of vertices in B that are unsaturated by M. Iteratively we construct a set U' containing U as follows. Initially we set $U' := U$ and $\tilde{A} = A$.

- Check whether there exists a neighbor of a vertex in U' in $\tilde{A}$; if yes go to the next step. Otherwise, output U'.
- Let a_j be a vertex in $\tilde{A}$ having a neighbor in U'. Set $U' := U' \cup \{b_j\}$ (b_j is the matching end point of a_j in B), $\tilde{A} := \tilde{A} \setminus \{a_j\}$ and go to the first step.

Let U' be the set returned by the iterative process above. Observe that $U \subsetneq U'$ and $\tilde{A} \subsetneq A$ as $P(H)$ is connected and hence there exists at least one vertex a_j having a neighbor in U and hence the above iteration does not stop in the first round. Let $A' = A \setminus \tilde{A}$ and let $X' = \{e_j \mid a_j \in A'\}$. In what follows we prove that X' is the desired subset of X' mentioned in the statement of the lemma.

We first show that X' is a strong cut-set. Let $\mathcal{C}' = \{C_j \mid b_j \in U'\}$. From the construction it follows that every $C_i \in \mathcal{C}'$ is a connected component of $H' = (\mathcal{V}, \mathcal{E} \setminus X')$. The size of C' is bounded as follows

$$|\mathcal{C}'| = |U'| = |U| + |A'| \geq |A'| + 2 = |X'| + 2,$$

and hence X' is a strong cut set.

We show that given any $f : \mathcal{V} \to \{0,1\}$ there exists a function $g : \mathcal{V} \to \{0,1\}$ such that $\mathit{Split}(g) = \mathit{Split}(f) \cup X'$. This will complete the proof of the lemma. Let $U' \setminus U = \{b_{1'}, b_{2'}, \ldots, b_{r'}\}$ and without loss of generality assume that $b_{1'}$, $b_{2'}$, ..., $b_{r'}$ is the order in which these elements are included in the set U'. Let $\mathcal{B}_i = \mathcal{B}[U \cup \{b_{1'}, \ldots, b_{i'}\} \cup \{a_{1'}, \ldots, a_{i'}\}]$. Iteratively we construct the function $g : \mathcal{V} \to \{0,1\}$ as follows. Initially we set $g := f$ and $i := 1$ and repeat the following until $i = r$:

> Check whether $e_{i'} \in \mathit{Split}(g)$. If yes $i := i+1$ and repeat. Otherwise let $C_{i'}$ be the connected component corresponding to $b_{i'}$ having vertex set $V(C_{i'})$. Now for every vertex $u \in V(C_{i'})$ change $g(u)$ to $1 - f(u)$. Basically, we flip the assignment of 0 and 1 in the vertex set $V(C_{i'})$. Set $i := i+1$ and repeat the procedure.

Now we show that $\mathit{Split}(g) = \mathit{Split}(f) \cup X'$. Observe that when we flip the assignment of the vertex set $V(C_{i'})$ the only hyperedges which could go out of the set $\mathit{Split}(g)$ are $\{e_{i'}, e_{(i+1)'}, \ldots, e_{r'}\}$. The reason we flip the assignment is because $e_{i'} \notin \mathit{Split}(g)$ at that point. Also notice that by construction there exists a $b_{j'} \in \{b_{1'}, b_{2'} \ldots, b_{(i-1)'}\}$ such that $V(e) \cap V(C_{j'}) \neq \emptyset$. Hence after we flip the assignment of the vertex set $V(C_{i'})$ we have that $e_{i'} \in \mathit{Split}(g)$. Hence after the r^{th} step of the procedure we have that $\mathit{Split}(g) = \mathit{Split}(f) \cup X'$. This concludes the proof. □

Lemma 1 naturally gives rise to a reduction rule for the p-SET SPLITTING problem. Given a strong cut set X, a strong cut set X' obtained by the Lemma 1 is called *reducible strong cut-set*. This brings us to the following reduction rule.

REDUCTION RULE 1. : *Let $(H = (\mathcal{V}, \mathcal{E}), k)$ be an instance of p-*SET SPLITTING *and X' be a reducible strong cut-set of H. Remove X' from the set of hyperedges and reduce k to $k - |X'|$, that is, obtain an instance $(H' = (\mathcal{V}, \mathcal{E} \setminus X), k - |X|)$.*

When the hypergraph H is disconnected we can give a simple reduction rule.

Reduction Rule 2. : *Let $(H = (\mathcal{V}, \mathcal{E}), k)$ be an instance of p-Set Splitting such that $P(H)$ has connected components $P(H)[C_1], \dots, P(H)[C_t]$. Let $v_1, \dots, v_t$ be vertices such that $v_i \in C_i$. Construct a hypergraph $H' = (\mathcal{V}', \mathcal{E}')$ from H by unifying the vertices $v_1, \dots, v_t$. In particular $\mathcal{V}' = \mathcal{V} \setminus \{v_2, \dots, v_t\}$ and for every hyperedge $e \in \mathcal{E}$ make the edge $e' \in \mathcal{E}'$ where $e' = e$ if $v_i \notin e$ for every i and $e' = (V(e) \setminus \{v_2, \dots, v_t\}) \cup \{v_1\}$ otherwise. We obtain the instance (H', k).*

The correctness proof for this reduction rule is simple, and given for example in [19] for the case of p-Max Cut.

Proof of Theorem 1. Given an instance (H, k) of p-Set Splitting we first obtain an equivalent instance with at most $2k$ sets and at most $2k^2$ elements by applying the kernelization algorithm of Chen and Lu [2], given in Theorem 1 of their paper. We then apply Reduction Rules 1 and 2 extensively. Let $(H' = (\mathcal{V}', \mathcal{E}'), k')$ be the reduced instance. Since both our rules and the rules of Chen and Lu [2] only reduce k we have that $k' \leq k$. Let H' have n elements and $m \leq 2k$ sets. We show that if $n > k'$ then (H', k') is a yes-instance. In particular, since Reduction Rule 2 does not apply, H' is connected. Since Reduction Rule 1 does not apply, H' does not have a strong cut-set. By Theorem 3 we can find in polynomial time a forest $\mathcal{F} = (F, g)$ of H' with $n - 1$ edges. Since F is a forest, F is bipartite. Let $W \uplus B$ be bipartitions of $\mathcal{V}'$. By the definition of a forest in a hypergraph, the bipartitions (W, B) splits all sets corresponding to hyperedges in $\mathcal{F}$. Since F has $n - 1$ edges, at least $n - 1 \geq k$ hyperedges are split and hence (H', k') is a yes-instance. Thus if $n > k$ for a reduced instance, the kernelization algorithm outputs that (H', k') is a yes-instance. Hence any unresolved reduced instance has at most $k' \leq k$ elements. This concludes the proof. □

3 Faster Parameterized Algorithm for p-Set Splitting

Theorem 1 yields a simple $O(2^k k^2 + N)$ time algorithm for the p-Set Splitting problem by looping over all possible bipartitions of set of elements into (W, B) and for each checking whether they split at least k edges. Previously, only a randomized $O(2^k k^2 + N)$ time algorithm [2] and a deterministic $O(2.65^k + N)$ time algorithm [14] was known. In this section we give an algorithm for for the p-Set Splitting problem running in $O(1.9630^k + N)$ time. Our algorithm first obtains a kernel with $2k$ sets and at most k elements using Theorem 1. Then the algorithm proceeds to solve the small instance recursively.

The subcases generated by the algorithm are naturally phrased as a colored version of the p-Set Splitting. In this version of the problem the sets in $\mathcal{F}$ are either *uncolored* or colored *white* or *black*. A black set S is split by a partitioning of U into W and B if $S \cap W \neq \emptyset$. Similarly a white set S is split if $S \cap B \neq \emptyset$. Hence, an instance to the Colored p-Set Splitting (p-CSS) problem is a universe U, a family $\mathcal{F} = \mathcal{F}_u \uplus \mathcal{F}_w \uplus \mathcal{F}_b$ over U and an integer k. The families $\mathcal{F}_u$, $\mathcal{F}_w$, and $\mathcal{F}_b$ denote the families of uncolored, white and black sets respectively.

Our algorithm is based on a single branching step. For a particular element v of U we try putting v in W or in B. If v is inserted into W, all sets in $\mathcal{F}_b$ containing v are split and all sets in $\mathcal{F}_u$ containing v are put into $\mathcal{F}_w$ instead. The sets that are split are removed from $\mathcal{F}_b$ and k is decreased by the number of newly split sets. Finally v is removed from the universe U and from all sets containing v. Similarly, if v is inserted into B then all sets in $\mathcal{F}_w$ containing v are split and all sets in $\mathcal{F}_u$ containing v are put into $\mathcal{F}_b$ instead. For a vertex v let $\mathcal{N}_u(v)$, $\mathcal{N}_b(v)$ and $\mathcal{N}_w(v)$ be the set of uncolored, black and white sets containing

v respectively. We call $d_u(v) = |\mathcal{N}_u(v)|$, $d_b(v) = |\mathcal{N}_b(v)|$ and $d_w(v) = |\mathcal{N}_w(v)|$ the uncolored, black and white degree of v. The degree of v is $d(v) = d_u(v) + d_w(v) + d_b(v)$. Formalizing the discussion above we obtain the following recurrence.

$$\begin{gathered}(U, \mathcal{F}_u, \mathcal{F}_w, \mathcal{F}_b, k) \in p\text{-CSS} \\ \Longleftrightarrow \\ (U \setminus \{v\}, \mathcal{F}_u \setminus \mathcal{N}_u(v), \mathcal{F}_w \cup \mathcal{N}_u(v), \mathcal{F}_b \setminus \mathcal{N}_b(v), k - d_b(v)) \in p\text{-CSS} \\ \bigvee \\ (U \setminus \{v\}, \mathcal{F}_u \setminus \mathcal{N}_u(v), \mathcal{F}_w \setminus \mathcal{N}_w(v), \mathcal{F}_b \cup \mathcal{N}_u(v), k - d_w(v)) \in p\text{-CSS}\end{gathered} \tag{1}$$

We now describe the algorithm for p-Set Splitting using Recurrence 1. We first formulate the p-Set Splitting instance $(U\mathcal{F}, k)$ as a p-CSS instance $(U, \mathcal{F}_u, \emptyset, \emptyset, k)$ where $\mathcal{F}_u = \mathcal{F}$. We fix $K = k$, and fix $\alpha = 0.027$, $\beta = 0.31$, and $\gamma = 0.13$.

Preprocessing. The algorithm computes a table for all subproblems $(U', \mathcal{F}'_u, \mathcal{F}'_w, \mathcal{F}'_b, k')$ where

- $U' \subseteq U$ and $|U'| \leq \alpha k$;
- $\mathcal{F}'_u \uplus \mathcal{F}'_w \uplus \mathcal{F}'_b \subseteq \mathcal{F}_u$ and $|\mathcal{F}'_u \uplus \mathcal{F}'_w \uplus \mathcal{F}'_b| \leq 4\alpha k$;
- $k' \leq k$.

An entry of the table contains true if the corresponding instance $(U', \mathcal{F}'_u, \mathcal{F}'_w, \mathcal{F}'_b, k')$ is in p-CSS. The table can be filled using Recurrence 1 and bottom up dynamic programming in time linear in the number of table entries. Thus the total time required to perform the preprocessing step is

$$O\left(\binom{k}{\alpha k} \cdot \binom{2k}{4\alpha k} \cdot 3^{4\alpha k} \cdot k\right).$$

Rewriting $\binom{a}{b}$ as $\frac{a!}{b!(a-b)!}$, using Stirling's approximation for $n!$ and plugging in the value of $\alpha = 0.027$ yields that the preprocessing step is done in $O(1.9630^k)$ time.

Branching. The algorithm selects an element $v \in U$ of highest degree and branches on this vertex using Recurrence 1. If the algorithm generates a subcase for which the answer is already stored in the preprocessing table, the algorithm returns this answer. If k reaches 0 or a negative number the algorithm returns "yes" and if k is positive and $U = \emptyset$ the algorithm returns "no". While k might not be the same in different recursive calls, the value of K fixed initially remains the same throughout the algorithm. Correctness of the algorithm follows directly from Recurrence 1. We now proceed to analyze the running time of the algorithm.

Running Time Analysis. We use the *Measure & Conquer* paradigm to analyze the running time of the algorithm. For two constants β and γ we define

$$\mu = \mu(U, \mathcal{F}_u, \mathcal{F}_w, \mathcal{F}_b, k) = |U| + \beta k + \gamma|\mathcal{F}_u|.$$

The running time of the algorithm is bounded from above by the number of leaves in the search tree, modulo a polynomial in k. Let $T(\mu, |U|)$ be an upper bound on the number of leaves in the search tree of the algorithm on an instance with measure μ and universe size $|U|$. We first need an auxiliary claim about the size of the search tree when the degree of any element is at most 4.

Claim 1. *Let K and α be fixed as in the discussion above and $(U, \mathcal{F}_u, \mathcal{F}_w, \mathcal{F}_b, k)$ be an instance of p-CSS such that $|U| \geq \alpha K$ generated during a recursive call such that the degree of any element is at most 4. Then $T(\mu, |U|) \leq 2^{|U|-\alpha K}$.*

Proof. We prove the claim by induction on $|U|$. If $|U| = \alpha K$ then the algorithm solves the instance by looking up in the preprocessing table as $|\mathcal{F}| \leq 4\alpha K$ and hence $T(\mu, |U|) = 1$. Assume now that the statement holds whenever $|U| = t$ for some fixed $t \geq \alpha K$ and consider an instance with $|U| = t+1$. The algorithm makes two recursive calls applying Recurrence 1. In each recursive call all elements have degree at most 4 and the size of $|U|$ is exactly t. By the induction hypothesis the number of leaves in the search tree of the two subinstances is at most $2^{t-\alpha K}$. Hence the total number of leaves in the search tree is bounded from above by $2 \cdot 2^{t-\alpha K} = 2^{|U|-\alpha K}$. $\square$

We now extend the analysis in Claim 1 to instances with no constraints on element degree.

Claim 2. *Let K and α be fixed as in the discussion above and $(U, \mathcal{F}_u, \mathcal{F}_w, \mathcal{F}_b, k)$ be an instance of p-CSS generated during a recursive call. Then $T(\mu, |U|) \leq 2^{|U|-\alpha K} + 1.5222^{\mu}$.*

Proof. We prove the claim by induction on $|U|$. If there are no elements of degree at least 5 and $|U| \geq 4\alpha K$ then the statement of the claim holds by Claim 1. If there are no elements of degree at least 5 and $|U| < 4\alpha K$ then $T(\mu, |U|) = 1 \leq 1.5222^{\mu}$. Assume now that there are elements of degree at least 5 and let v be the element on which the algorithm branches. Since the algorithm picks an element of largest degree, v has degree at least 5. If the uncolored, white and black degrees of v are $d_u(v), d_w(v), d_b(v)$ we let d'_u, d'_w and d'_b be non-negative integers such that $d'_u + d'_w + d'_b = 5$ and $d'_u \leq d_u(v)$, $d'_w \leq d_w(v)$ and $d'_b \leq d_b(v)$. When we apply Recurrence 1 to branch on an element v we get the following recurrence for T

$$T(\mu, |U|) \leq T(\mu - 1 - \beta d'_b - \gamma d'_u, |U| - 1) + T(\mu - 1 - \beta d'_w - \gamma d'_u, |U| - 1).$$

One can verify that if we pick $c = 1.5222$ then for any choice of (d'_u, d'_w, d'_b) such that $d'_u + d'_w + d'_b = 5$ we have

$$\begin{aligned}
T(\mu, |U|) &\leq T(\mu - 1 - \beta d'_b - \gamma d'_u, |U| - 1) + T(\mu - 1 - \beta d'_w - \gamma d'_u, |U| - 1) \\
&\leq c^{\mu-1-\beta d'_b-\gamma d'_u} + 2^{|U|-1-\alpha K} + c^{\mu-1-\beta d'_w-\gamma d'_u} + 2^{|U|-1-\alpha K} \\
&= c^{\mu} \cdot (c^{-1-\beta d'_b-\gamma d'_u} + c^{-1-\beta d'_w-\gamma d'_u}) + 2 \cdot 2^{|U|-1-\alpha K} \\
&\leq c^{\mu} + 2^{|U|-\alpha K}.
\end{aligned}$$

Hence $T(\mu, |U|) \leq 2^{|U|-\alpha K} + 1.5222^{\mu}$, concluding the proof. $\square$

Summing up the above analysis, noticing that $\mu \leq K + \beta K + \gamma 2K$ in a reduced instance, and inserting this into the bound for $T(\mu, |U|)$ from Claim 2 yields an upper bound of $O(1.9630^k)$ for the running time of the branching part of the algorithm. Since both parts of the algorithm take $O(1.9630^k)$ time, this completes the proof of Theorem 2.

4 Conclusion and Discussions

In this paper we gave a smaller kernel and a faster algorithm for the p-Set Splitting problem improving over the previosuly known results. The number of elements and sets in our kernel matches the number of vertices and edges in the best known kernel for p-Max Cut. It should be noted that both the kernel and the algorithm for p-Set Splitting presented here also work for the p-Not All Equal SAT problem. The reduction rule we use to handle instances with strong cut-sets has similarities with reduction rules based on *crown decompositions* [3, 8, 19], and it seems that crown decompositions and strong cut-sets are closely related. This similarity also makes us believe that the duality theorem we made us of in our kenrelization algorithm will be a useful tool in the filed of kernelization.

Acknowledgments.

We would like to thank Stéphan Thomassé and Fedor V. Fomin for valuable suggestions and insightful discussions. We especially thank Stéphan Thomassé for pointing us to the Proposition 1 and the reference [11].

References

[1] G. Andersson and L. Engebretsen. Better approximation algorithms for SET SPLITTING and NOT-ALL-EQUAL SAT. *Inf. Process. Lett.*, 65(6):305–311, 1998.

[2] J. Chen and S. Lu. Improved algorithms for weighted and unweighted set splitting problems. In *COCOON*, volume 4598 of *Lecture Notes in Computer Science*, pages 537–547, 2007.

[3] M. Chlebík and J. Chlebíková. Crown reductions for the minimum weighted vertex cover problem. *Discrete Applied Mathematics*, 156(3):292–312, 2008.

[4] F. K. H. A. Dehne, M. R. Fellows, and F. A. Rosamond. An FPT algorithm for set splitting. In *WG*, volume 2880 of *Lecture Notes in Computer Science*, pages 180–191, 2003.

[5] F. K. H. A. Dehne, M. R. Fellows, F. A. Rosamond, and P. Shaw. Greedy localization, iterative compression, modeled crown reductions: New fpt techniques, an improved algorithm for set splitting, and a novel $2k$ kernelization for vertex cover. In *IWPEC*, volume 3162 of *Lecture Notes in Computer Science*, pages 271–280, 2004.

[6] P. Erdős. On a combinatorial problem, I. *Nordisk Mat. Tidskrift*, 11:5–10, 1963.

[7] P. Erdős. On a combinatorial problem, II. *Acta Math, Hungary*, 15:445–447, 1964.

[8] M. R. Fellows. Blow-ups, win/win's, and crown rules: Some new directions in FPT. In *WG*, volume 2880 of *Lecture Notes in Computer Science*, pages 1–12, 2003.

[9] F. V. Fomin, F. Grandoni, and D. Kratsch. Measure and conquer: Domination - a case study. In *ICALP*, volume 3580 of *Lecture Notes in Computer Science*, pages 191–203, 2005.

[10] F. V. Fomin, F. Grandoni, and D. Kratsch. Measure and conquer: a simple $o(2^{0.288})$ independent set algorithm. In *SODA*, pages 18–25, 2006.

[11] A. Frank, T. Király, and M. Kriesell. On decomposing a hypergraph into k connected sub-hypergraphs. *Discrete Applied Mathematics*, 131(2):373–383, 2003.

[12] M. X. Goemans and D. P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM*, 42(6):1115–1145, 1995.

[13] R. M. Karp. Reducibility among combinatorial problems. *Complexity of Computer Computations*, pages 85–103, 1972.

[14] D. Lokshtanov and C. Sloper. Fixed parameter set splitting, linear kernel and improved running time. In *ACiD*, volume 4 of *Texts in Algorithmics*, pages 105–113, 2005.

[15] L. Lovász. Covering and coloring of hypergraphs. In *Proceedings of the 4th Southeastern Conference on Combinatorics, Graph Theory and Computing*, Utilitas Mathematica Publishing, pages 3–12, 1973.

[16] E. Prieto. The method of extremal structure on the k-maximum cut problem. In *CATS*, pages 119–126, 2005.

[17] J. Radhakrishnan and A. Srinivasan. Improved bounds and algorithms for hypergraph 2-coloring. *Random Struct. Algorithms*, 16(1):4–32, 2000.

[18] V. Raman and S. Saurabh. Improved fixed parameter tractable algorithms for two "edge" problems: MAXCUT and MAXDAG. *Inf. Process. Lett.*, 104(2):65–72, 2007.

[19] C. Sloper. Techniques in parameterized algorithm design. PhD thesis, University of Bergen, 2005.

[20] M. Wahlström. Algorithms, measures, and upper bounds for satisfiability and related problems. PhD thesis, Linkp̌ing University, 2007.

[21] J. Zhang, Y. Ye, and Q. Han. Improved approximations for max set splitting and max NAE SAT. *Discrete Applied Mathematics*, 142(1-3):133–149, 2004.

[22] U. Zwick. Outward rotations: A tool for rounding solutions of semidefinite programming relaxations, with applications to max cut and other problems. In *STOC*, pages 679–687, 1999.