Exponential-Time Approximation of Weighted Set Cover*

Marek Cygan, Łukasz Kowalik and Mateusz Wykurz
Institute of Informatics, University of Warsaw, Poland.
cygan@mimuw.edu.pl, kowalik@mimuw.edu.pl, wykurz@gmail.com.

Abstract

The Set Cover problem belongs to a group of hard problems which are neither approximable in polynomial time (at least with a constant factor) nor fixed parameter tractable, under widely believed complexity assumptions. In recent years, many researchers design exact exponential-time algorithms for problems of that kind. The goal is getting the time complexity still of order $O(c^n)$, but with the constant c as small as possible. In this work we extend this line of research and we investigate whether the constant c can be made even smaller when one allows constant factor approximation.

In fact, we describe a kind of approximation schemes — trade-offs between approximation factor and the time complexity. We use general transformations from exponential-time exact algorithms to approximations that are faster but still exponential-time. For example, we show that for any reduction rate r, one can transform any $O^*(c^n)$-time[1] algorithm for Set Cover into a $(1+\ln r)$-approximation algorithm running in time $O^*(c^{n/r})$. We believe that results of that kind extend the applicability of exact algorithms for NP-hard problems.

1 Introduction

1.1 Motivation

One way of coping with NP-hardness is polynomial-time approximation, i.e. looking for solutions that are relatively close to optimal. Unfortunately it turns out that there are still many problems which do not allow for good approximation. Let us recall some examples. Håstad [14] showed that Independent Set cannot be approximated in polynomial time with factor $n^{1-\epsilon}$ for any $\epsilon > 0$ unless NP = ZPP. The same holds for Vertex Coloring due to Feige and Kilian [11]. By another result of Feige [10], Set Cover cannot be approximated in polynomial time with factor $(1-\epsilon)\ln n$, where n is the size of the set to cover, for any $\epsilon > 0$ unless NP $\subseteq$ DTIME($n^{\log\log n}$).

Another approach is the area of parametrized complexity (see e.g. [9]). Then the goal is to find an algorithm with time exponential only in a parameter unrelated to the instance size (then we say the problem is *fixed parameter tractable*, FPT in short). This parameter may reflect complexity of the instance – like treewidth, but then we get an efficient algorithm only for some subclass of possible instances. Another choice of the parameter is the measure of the solution quality. For example, one can verify whether in an n-vertex graph there is a

*This research is partially supported by a grant from the Polish Ministry of Science and Higher Education, project N206 005 32/0807.

[1] $O^*(f(n))$ notation suppresses polynomial factors

vertex cover of size k in $O(1.2738^k + kn)$ time [6]. Again, the parametrized approach does not succeed in some cases. Verifying whether a graph is k-colorable is NP-complete for any $k \geq 3$, while INDEPENDENT SET and SET COVER are $W[1]$- and $W[2]$-complete respectively, meaning roughly that an FPT algorithm for INDEPENDENT SET or SET COVER would imply algorithms of that kind for a host of other hard problems.

The aforementioned hardness results motivate the study of "moderately exponential time" algorithms. The goal here is to devise algorithms with exponential running time $O(2^{n/r})$ with r big enough. Indeed, a $O(2^{n/50})$-time algorithm may appear practical for some range of n, say $n \leq 1000$. Despite some progress in this area we are still far from exact algorithms with time complexity of that order. One of the most researched problems in this field is INDEPENDENT SET. Exhaustive search for that problem gives $O(2^n)$ time bound while the currently best published result [12] is $O(2^{n/3.47})$. For VERTEX COLORING the first $O^*(2^{n/0.77})$-time algorithm by Lawler was then improved in a series of papers culminating in a breakthrough $O^*(2^n)$ bound of Björklund, Husfeldt and Koivisto [3].

Now consider the (weighted) SET COVER problem. The instance consists of a family of sets $\mathcal{S} = \{S_1, \ldots, S_m\}$, and each of the sets S_i is assigned a weight $w(S_i)$. The set $U = \bigcup \mathcal{S}$ is called *the universe* and we denote $n = |U|$. The goal is to find a subfamily $\mathcal{C} \subseteq \mathcal{S}$ such that $\bigcup \mathcal{C} = U$ so as to minimize the total weight of the sets in $\mathcal{C}$. Assume that the size of our instance is relatively small but big enough that finding an optimal solution using an exact algorithm is out of question, say $m = 150$, $n = 200$. If we use the (best known) greedy approximation algorithm (see e.g. [18]), we get the approximation ratio of $H_n < \ln n + 1$, which is roughly 6.3 in this case. A simple implementation of this algorithm runs in $O((\sum_{i=1}^{m} |S_i|) \log m)$ time, so we would get an answer very fast. A natural question is: can we get a better approximation guarantee by investing more computing time? Motivated by this question we look for approximation algorithms with good (e.g. constant) approximation guarantee and with running time exponential but substantially lower than the best known exact algorithm. Ideally, one would like to have a kind of trade-off between the running-time and approximation ratio – then one gets as much accuracy as one can afford.

1.2 Our Results

In this paper we present the following results (as in the previous section, n and m denote the size of the universe and the number of sets respectively):

- For any reduction rate r, one can transform any $O^*(c^n)$-time algorithm for (weighted) SET COVER into an $(1 + \ln r)$-approximation algorithm running in $O^*(c^{n/r})$ time.
- For any reduction rate r, one can transform any $O^*(c^m)$-time algorithm for (weighted) SET COVER into an r-approximation algorithm running in $O^*(c^{m/r})$ time.
- There is an $O^*(\min\{4^n m^{\log n}, 9^n\})$-time, polynomial space exact algorithm for (weighted) SET COVER.

Let us note that the best known exact algorithm for SET COVER with the time complexity depending exponentially only on the number of sets m is the trivial $O^*(2^m)$-time algorithm. If we are interested in time complexity depending exponentially only on the size of the universe n, a simple dynamic programming can be used to get a $O^*(2^n)$-time, $O(2^n)$-space exact algorithm. We are not aware of any previous polynomial space exact algorithm with time complexity $O^*(c^n)$, with c being a constant.

Note that by putting $r = n/\log n$ in the first result above, we get a polynomial time approximation with approximation ratio $1 + \ln n$, which roughly matches the ratio of the (essentially optimal) greedy algorithm. Thus, our approach can be viewed as a continuous scaling between the best possible polynomial time approximation and the best known exponential time algorithm. In other words, one can get as good solution as he can afford, by using as much time as is available.

1.3 Our Approach

Our approach is very natural: we transform the input instance into a number of (r times) smaller instances. Then, for each of the smaller instances, the problem is solved separately by an exact algorithm. Finally, we describe a method for building an approximate solution for the original instance, using the solutions for the smaller instances. The approximation ratio is a function of r. We will call this method a *reduction*. For some basic properties of this method, and its application to such problems as Maximum Independent Set, Vertex Coloring, Bandwidth and Semi-Metric TSP, see our technical report [7].

Note that in the case of Set Cover there are two natural measures for size of the instance: the size of the universe U and the number of sets in the family $\mathcal{S}$. As it was mentioned, we will present reductions for both measures: in sections 2 and 3 respectively.

1.4 Related Work

The concept of algorithms with a trade-off between the approximation ratio and the running-time is quite old, though researchers focused on polynomial-time algorithms – so called *polynomial-time approximation schemes (PTAS)*. Probably one of the earliest and most widely known examples of this kind is the PTAS for the knapsack problem, due to Ibarra and Kim [15], which has approximation ratio of $(1 - \varepsilon)$ and time complexity $O(n^3/\varepsilon)$, for any $\varepsilon > 0$. For more examples of PTASes see the textbook of Vazirani [18].

The first work on *exponential-time* approximation algorithms we are aware of concerns the Max Sat problem and it is due to Dantsin, Gavrilovich, Hirsch and Konev [8].

Later, Björklund and Husfeldt in [2] showed that any $O^*(c^n)$-time algorithm for the vertex coloring problem can be transformed into a $(1+\ln r)$-approximation algorithm running in time $O^*(c^{n/r} + 1.2209^n)$. In a way, they also use the method of reduction and this is probably the first application of this method in exponential-time approximation.

Very recently, exponential-time approximation algorithms for Set Cover were studied by Bourjeois, Escoffier and Paschos [5]. Independently, they used a very similar approach to ours, and they obtained approximation algorithms with the same performance as the two algorithms we present here. Additionally, they show a randomized r-approximation algorithm with time complexity smaller (by an exponential factor) than $O^*(2^{m/r})$ that returns the correct answer with high probability. However, all their algorithms apply to unweighted version of the problem only and do not extend directly to the weighted version we consider here.

The same team investigated also exponential-time approximation algorithms for Independent Set and Vertex Cover, see [4], getting some interesting results (also using the reduction, among other methods).

The idea of joining the worlds of approximation algorithms and "moderately" exponential algorithms appeared also in a recent work of Vassilevska, Williams and Woo [17]. However their direction of research is completely different from ours, i.e. they consider so-called hybrid

algorithms. For example, they report to have an algorithm for BANDWIDTH which for given input *either* returns an $O(\log n)$-approximate solution in polynomial time *or* returns a $(1+\epsilon)$-approximate solution in $O(2^{n/\log\log n})$ time. We see that the hybrid algorithm does not guarantee constant approximation ratio and hence cannot be directly compared with our work.

Another promising area is joining the worlds of parametrized complexity and polynomial-time approximation algorithms — see the survey paper [16].

2 Reducing the size of universe

We use the notation for SET COVER from Section 1.1. In what follows, we write $w(\mathcal{C})$ for the total weight of a family of sets $\mathcal{C}$. Any subfamily $\mathcal{C} \subseteq \mathcal{S}$ such that $\bigcup \mathcal{C} = U$ will be called a *cover* (of U).

An r-approximate solution of SET COVER can be found by dividing U into r parts, covering each of them separately by an exact algorithm and returning the union of these covers. If the exact algorithm works in $O^*(c^n)$ time for an instance with the universe of size n, this method takes $O^*(c^{n/r})$ time. In this section we describe an approach which gives much better approximation ratio within the same time bound.

Let's recall the greedy algorithm (see e.g. [18]), called Greedy from now. It selects sets to the cover one by one. Let $\mathcal{C}$ be the family of sets chosen so far. Then Greedy takes a set that covers *new* elements as cheap as possible, i.e. chooses S so as to minimize $w(S)/|S \setminus \bigcup \mathcal{C}|$. For each element $e \in S \setminus \bigcup \mathcal{C}$ the amount $w(S)/|S \setminus \bigcup \mathcal{C}|$ is called the *price* of e and denoted as $\mathrm{price}(e)$. This procedure continues until $\mathcal{C}$ covers the whole U. Let $e_1, \ldots, e_n$ be the sequence of all elements of U in the order of covering by Greedy (ties broken arbitrarily). The standard analysis of Greedy uses the following lemma (see [18] for the proof).

Lemma 2.1. *For each* $k \in 1, \ldots, n$, $\mathrm{price}(e_k) \leq w(\mathrm{OPT})/(n-k+1)$

The idea of our reduction is very simple. For example, assume we want to reduce the size of the universe twice, and n is even. Lemma 2.1 tells us that Greedy starts from covering elements very cheaply, and than pays more and more. So we just stop it before it pays much but after it covers sufficiently many elements. Note that if we manage to stop it just after $e_{n/2}$ is covered the total price of the covered elements (and hence the weight of the sets chosen) is at most $(H_n - H_{n/2})w(\mathrm{OPT}) = \ln 2 \cdot w(\mathrm{OPT})$, where H_n is the n-th harmonic number. If we cover the remaining elements, say, by exact algorithm we get a $(1+\ln 2)$-approximation. However the set that covers $e_{n/2}$ may cover many elements e_i, $i > n/2$. By Lemma 2.1 the price of each of them is at most $w(\mathrm{OPT})/(n/2) = 2w(\mathrm{OPT})/n$. Hence this last set costs us at most $w(\mathrm{OPT})$ and together we get a $(2+\ln 2)$-approximation. Luckily, it turns out that paying $w(\mathrm{OPT})$ for the last set chosen by Greedy is not necessary: below we show a refined algorithm which would yield a $(1+\ln 2)$-approximation in this particular case.

Theorem 2.2. *Assume there is an exact algorithm for (weighted)* SET COVER *running in time* $O^*(c^n)$*, for some constant* c*. Then, for any* $r \in \mathbb{Q}$ *there is an algorithm with approximation ratio* $1+\ln r$ *running in* $O^*(c^{n/r})$ *time.*

Proof. Let $I = (\mathcal{S}, w)$ be an instance of SET COVER problem. Our algorithm works similarly as Greedy. However, before adding a set T to the partial cover $\mathcal{C}$ it checks whether adding T

Pseudocode 2.1 ($\ln r + 1$)-approximation algorithm for Set Cover

$\mathcal{R} \leftarrow \mathcal{S}$.
$\mathcal{C} \leftarrow \emptyset$.
while $\bigcup \mathcal{S} \cup \bigcup \mathcal{C} = U$ **do**
    Find $T \in \mathcal{S}$ so as to minimize $\frac{w(T)}{|T \setminus \bigcup \mathcal{C}|}$
    if $n - |\bigcup \mathcal{C} \cup T| > n/r$ **then**
        $\mathcal{C} \leftarrow \mathcal{C} \cup \{T\}$.
    else
        $\mathcal{C}_T \leftarrow \mathcal{C}$ (just for the analysis).
        Create an instance $I_T = (\mathcal{S}_T, w)$, such that for each $P \in \mathcal{S}$, $\mathcal{S}_T$ contains set $P \setminus (\bigcup \mathcal{C} \cup T)$, of weight $w(P)$.
        $\mathrm{OPT}_{I_T} \leftarrow$ the optimal solution for I_T.
        if $w(\mathcal{C}_T) + w(T) + w(\mathrm{OPT}_{I_T}) < w(\mathcal{R})$ **then** $\mathcal{R} \leftarrow \mathcal{C}_T \cup \{T\} \cup \mathrm{OPT}_{I_T}$.
        $\mathcal{S} \leftarrow \mathcal{S} \setminus \{T\}$
return $\mathcal{R}$.

to $\mathcal{C}$ makes the number of non-covered elements at most n/r. If so, T is called a *crossing set*. Instead of adding T to $\mathcal{C}$, the algorithm creates an instance $I_T = (\mathcal{S}_T, w)$ of Set Cover that will be used to cover the elements covered neither by $\mathcal{C}$ nor by T. Namely, for each $P \in \mathcal{S}$, $\mathcal{S}_T$ contains set $P \setminus (\bigcup \mathcal{C} \cup T)$, of weight $w(P)$. Note that in the instance I_T the size of the universe is at most n/r. Then the optimal cover OPT_{I_T} for I_T is found in $O^*(c^{n/r})$ time using the exact algorithm. Let $\mathcal{C}_T$ denote the current collection $\mathcal{C}$. Clearly, $\mathcal{C}_T \cup \{T\} \cup \mathrm{OPT}_{I_T}$ is a cover of the initial universe U. Let us denote it by $\mathcal{R}_T$. Next, set T is removed from the family of available sets $\mathcal{S}$. If it turns out that the universe cannot be covered after removing T, i.e. $\bigcup \mathcal{S} \cup \bigcup \mathcal{C} \neq U$, the algorithm stops and returns the lightest cover of U found so far. See Pseudocode 2.1 for details. Note that the algorithm finds at least 1 and at most $|\mathcal{S}|$ covers, so the claimed time complexity follows.

Now we prove the bound on the approximation ratio. Let T^* be the first crossing set found such that T^* belongs to OPT_I, some optimal solution for instance I (note that at least one crossing set is in OPT_I). We need to bound the value of $w(\mathcal{R}_{T^*}) = w(\mathcal{C}_{T^*}) + w(T^*) + w(\mathrm{OPT}_{I_{T^*}})$. Clearly $\mathrm{OPT}_I \setminus \{T^*\}$ covers $\bigcup \mathcal{S}_{T^*}$. Hence $w(\mathrm{OPT}_{I_{T^*}}) \leq w(\mathrm{OPT}_I \setminus \{T^*\})$ so $w(T^*) + w(\mathrm{OPT}_{I_{T^*}}) \leq w(\mathrm{OPT}_I)$. Since $\mathcal{C}_{T^*}$ covers less than $n - n/r$ elements, by Lemma 2.1

$$\begin{aligned} w(\mathcal{C}_{T^*}) &\leq \sum_{k=1}^{\lfloor n - n/r \rfloor} \frac{w(\mathrm{OPT}_I)}{n-k+1} = \sum_{k=1}^{n-\lceil n/r \rceil} \frac{w(\mathrm{OPT}_I)}{n-k+1} = (H_n - H_{\lceil n/r \rceil}) w(\mathrm{OPT}_I) = \\ &\leq (\ln n - \ln \lceil n/r \rceil) w(\mathrm{OPT}_I) \leq \ln r \cdot w(\mathrm{OPT}_I). \end{aligned}$$

The second last inequality follows from the fact that $H_n - \ln n$ decreases monotonically to the Euler constant. We conclude that the algorithm returns a cover of weight $\leq (1 + \ln r) w(\mathrm{OPT}_I)$. □

Clearly, to make use of the above universe-scaling reduction we need a $O^*(c^n)$ exact algorithm, where c is a constant. As noted in the Introduction, one can use a $O^*(2^n)$-time, $O(2^n)$-space algorithm by dynamic programming or the polynomial space algorithm described in Section 4.

3 Reducing the number of sets

Let us begin with a simple algorithm for the *unweighted* version of the problem (found independently by Bourjeois et al. [5]). Let $\mathcal{S} = \{S_1, \ldots, S_m\}$ be the input instance. Assume m is even. Then create a new instance $\mathcal{Z} = \{Z_1, \ldots, Z_{m/2}\}$ where $Z_i = S_{2i-1} \cup S_{2i}$. Next find an optimal solution $\mathrm{OPT}_{\mathcal{Z}}$ for $\mathcal{Z}$ using an exact algorithm. Let $\mathcal{C} = \{S_{2i-1} \mid Z_i \in \mathrm{OPT}_{\mathcal{Z}}\} \cup \{S_{2i} \mid Z_i \in \mathrm{OPT}_{\mathcal{Z}}\}$. Clearly, $|\mathrm{OPT}_{\mathcal{Z}}| \leq |\mathrm{OPT}_{\mathcal{S}}|$ and hence $|\mathcal{C}| \leq 2|\mathrm{OPT}_{\mathcal{S}}|$. Thus we get a 2-approximation in $T(m/2)$ time where $T(m)$ is the best known bound for an exact algorithm, so currently just $T(m/2) = O^*(2^{m/2})$ after applying the exhaustive search. Of course this method is scalable – similarly we get a 5-approximation in time $O^*(2^{m/5})$.

In the weighted version the above algorithm fails, basically because the sets from the optimal solution may be joined with some heavy sets. The natural thing to do is sorting the sets according to their weight and joining only neighboring sets. This simple modification does not succeed fully but with some more effort we can make it work.

Theorem 3.1. *Assume there is an exact algorithm for (weighted)* Set Cover *running in time $O^*(c^m)$, for some constant c. Then, for any $r \in \mathbb{N}$, $r > 1$, there is an r-approximation algorithm running in $O^*(c^{m/r})$ time.*

Proof. The algorithm starts from sorting the sets in $\mathcal{S}$ in the order of non-decreasing weight. So let $\mathcal{S} = \{S_1, \ldots, S_m\}$ so that $w(S_1) \leq w(S_2) \leq \ldots \leq w(S_m)$. Next it partitions this sequence into blocks $\mathcal{B}_i$, $i = 1, \ldots, \lceil m/r \rceil$, each of size at most r, namely $\mathcal{B}_i = \{S_j \in \mathcal{S} \mid (i-1)r < j \leq ir\}$. Let $U_i = \bigcup \mathcal{B}_i$ be the union of all sets in $\mathcal{B}_i$ and define its weight as the total weight of $\mathcal{B}_i$, i.e. $w(U_i) = w(\mathcal{B}_i)$. For any $k = 1, \ldots, m$ we also define $\mathcal{X}_k = \{S_j \in \mathcal{B}_{\lceil k/r \rceil} \mid j < k\}$ and $V_k = \bigcup \mathcal{X}_k$ with $w(V_k) = w(\mathcal{X}_k)$. The algorithm creates m instances, namely $\mathcal{S}_i = \{U_j \mid S_i \notin \mathcal{B}_j\} \cup \{V_i, S_i\}$ for $i = 1, \ldots, m$.

Of course any subfamily (or a cover) $\mathcal{C} \subseteq \mathcal{S}_i$ corresponds to $\widehat{\mathcal{C}}$, a subfamily of $\mathcal{S}$ with the same weight obtained from $\mathcal{C}$ by splitting the previously joined sets (we will use this denotation further). Clearly $\bigcup \mathcal{C} = \bigcup \widehat{\mathcal{C}}$, in particular if $\mathcal{C}$ is a cover, so is $\widehat{\mathcal{C}}$.

Next, for each instance $\mathcal{S}_i$, $i = 1, \ldots, m$, the optimum solution $\mathcal{C}_i$ is found in $O^*(c^{m/r})$ time. Finally, the algorithm chooses the lightest of them, say $\mathcal{C}_q$, and returns $\widehat{\mathcal{C}_q}$, a cover of U.

Now it suffices to show that one of the instances has a cover that is light enough. Let $i^* = \max\{i \mid S_i \in \mathrm{OPT}\}$. We focus on instance $\mathcal{S}_{i^*}$. If $\mathcal{X}_{i^*} \cap \mathrm{OPT} = \emptyset$ we choose its cover $\mathcal{R} = \{U_j \in \mathcal{S}_{i^*} \mid \mathcal{B}_j \cap \mathrm{OPT} \neq \emptyset\} \cup \{S_{i^*}\}$, otherwise $\mathcal{R} = \{U_j \in \mathcal{S}_{i^*} \mid \mathcal{B}_j \cap \mathrm{OPT} \neq \emptyset\} \cup \{V_{i^*}, S_{i^*}\}$ Clearly it suffices to show that $w(\widehat{\mathcal{R}} \setminus \mathrm{OPT}) \leq (r-1)w(\mathrm{OPT})$. Consider any $S_i \in \widehat{\mathcal{R}} \setminus \mathrm{OPT}$. If $S_i \notin \mathcal{X}_{i^*}$ we put $f(i) = \min\{j \mid S_j \in \mathrm{OPT} \text{ and } \lceil j/r \rceil > \lceil i/r \rceil\}$, otherwise $f(i) = i^*$. Then $w(S_i) \leq w(S_{f(i)})$. We see that f maps at most $r-1$ elements to a single index j of a set from OPT, so indeed $w(\widehat{\mathcal{R}} \setminus \mathrm{OPT}) \leq (r-1)w(\mathrm{OPT})$ and hence $w(\widehat{\mathcal{R}}) \leq rw(\mathrm{OPT})$. Since $w(\mathcal{R}) = w(\widehat{\mathcal{R}})$, it follows that $w(\mathrm{OPT}_{\mathcal{S}_{i^*}}) \leq w(\mathcal{R}) \leq rw(\mathrm{OPT})$ so $w(\mathcal{C}_{i^*}) \leq rw(\mathrm{OPT})$ and finally $w(\widehat{\mathcal{C}_q}) \leq rw(\mathrm{OPT})$. □

4 $O^*(c^n)$-time polynomial space exact algorithms for Set Cover

In this section we present the first $O^*(c^n)$-time polynomial space exact algorithms for Set Cover. We follow the divide-and-conquer approach of Gurevich and Shelah [13] rediscovered

recently by Björklund and Husfeldt [1] and we get a $O^*(4^n m^{\log n})$-time algorithm. If m is big we can use another, $O^*(9^n)$-time version of it.

Let us note that for the unweighted case there is an $O(2^n mn)$-time polynomial space algorithm by Björklund et al. [3] using the inclusion-exclusion principle.

Theorem 4.1. *There is a $O^*(\min\{4^n m^{\log n}, 9^n\})$-time polynomial-time algorithm that finds a minimum-weight cover of the universe of size n by a family consisting of m sets.*

Proof. The algorithm is as follows. For an instance with universe U of size n we recurse on an exponential number of instances, each with universe of size smaller than $n/2$. Namely, we choose one of m sets S and we divide the remaining elements, i.e. $U \setminus S$ into two parts, each of size at most $n/2$. We consider all choices of sets and all such partitions – there are $O(m2^n)$ of them. For each such set S and partition U_1, U_2 we find recursively $\mathcal{C}_1$, an optimal cover of U_1 and $\mathcal{C}_2$, an optimal cover of U_2. Clearly $\mathcal{C}_1 \cup \mathcal{C}_2 \cup \{S\}$ forms a cover of U. We choose the best cover out of the $O^*(m2^n)$ covers obtained like this.

Consider an optimal cover OPT. For each element e of U assign a unique set S_e from OPT such that $e \in S_e$. For each $S \in \text{OPT}$ let $S^* = \{e \in U : S = S_e\}$. Let $\mathcal{P} = \{S^* : S \in \text{OPT}\}$. $\mathcal{P}$ is a partition of U. Clearly, after removing the biggest set $\hat{S}$ from OPT we can divide all the sets in $\mathcal{P}$ into two groups, $\mathcal{P}_1$ and $\mathcal{P}_2$, each covering less than $n/2$ elements from $U \setminus \hat{S}$. It follows that one of the $O^*(m2^n)$ covers found by the algorithm has weight $w(\text{OPT})$, namely the cover obtained for set $\hat{S}$ and partition $(\bigcup \mathcal{P}_1 \setminus \hat{S}, \bigcup \mathcal{P}_2 \setminus \hat{S})$.

It is clear that the above algorithm works in time $O^*(4^n m^{\log n})$. Similarly we can also get a $O^*(9^n)$ bound – instead of at most $m2^n$ instances we recurse on at most $2 \cdot 3^n$ instances: we consider all partitions of U into three sets A, B, C. Let $|A| \geq |B| \geq |C|$. If $|A| \leq n/2$ we recurse on A, B and C and otherwise we check whether $A \in \mathcal{S}$ and if so, we recurse on B and C. □

References

[1] A. Björklund and T. Husfeldt. Exact algorithms for exact satisfiability and number of perfect matchings. In *Proc. ICALP'06*, pages 548–559, 2006.

[2] A. Björklund and T. Husfeldt. Inclusion–exclusion algorithms for counting set partitions. In *Proc. FOCS'06*, pages 575–582, 2006.

[3] A. Björklund, T. Husfeldt, and M. Koivisto. Set partitioning via inclusion-exclusion. *SIAM J. Comput., Special Issue for FOCS 2006.* To appear.

[4] N. Bourjeois, B. Escoffier, and V. T. Paschos. Efficient approximation by "low-complexity" exponential algorithms. Technical Report 271, LAMSADE, Universite Paris Dauphine, 2008.

[5] N. Bourjeois, B. Escoffier, and V. T. Paschos. Efficient approximation of MIN SET COVER by "low-complexity" exponential algorithms. Technical Report 278, LAMSADE, Universite Paris Dauphine, 2008.

[6] J. Chen, I. A. Kanj, and G. Xia. Improved parameterized upper bounds for vertex cover. In *Proc. MFCS'06*, pages 238–249, 2006.

[7] M. Cygan, Łukasz Kowalik, M. Pilipczuk, and M. Wykurz. Exponential-time approximation of hard problems, 2008. arXiv:0810.4934, http://arxiv.org/abs/0810.4934.

[8] E. Dantsin, M. Gavrilovich, E. A. Hirsch, and B. Konev. MAX SAT approximation beyond the limits of polynomial-time approximation. *Ann. Pure Appl. Logic*, 113(1-3):81–94, 2001.

[9] R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer, 1999.

[10] U. Feige. A threshold of $\ln n$ for approximating set cover. *J. ACM*, 45(4):634–652, 1998.

[11] U. Feige and J. Kilian. Zero knowledge and the chromatic number. *J. Comput. Syst. Sci.*, 57(2):187–199, 1998.

[12] F. V. Fomin, F. Grandoni, and D. Kratsch. Measure and conquer: a simple $O(2^{0.288n})$ independent set algorithm. In *Proc. SODA'06*, pages 18–25, 2006.

[13] Y. Gurevich and S. Shelah. Expected computation time for hamiltonian path problem. *SIAM J. Comput.*, 16(3):486–502, 1987.

[14] J. Håstad. Clique is hard to approximate within $n^{1-\epsilon}$. *Acta Mathematica*, 182(1):105–142, 1999.

[15] O. H. Ibarra and C. E. Kim. Fast approximation algorithms for the knapsack and sum of subset problems. *J. ACM*, 22(4):463–468, 1975.

[16] D. Marx. Parameterized complexity and approximation algorithms. *The Computer Journal*, 51(1):60–78, 2008.

[17] V. Vassilevska, R. Williams, and S. L. M. Woo. Confronting hardness using a hybrid approach. In *Proc. SODA'06*, pages 1–10, 2006.

[18] V. V. Vazirani. *Approximation Algorithms*. Springer, 2001.