HAL
open science

Efficient approximation of MIN SET COVER by "low-complexity" exponential algorithms

Nicolas Bourgeois, Bruno Escoffier, Vangelis Paschos

To cite this version:

Nicolas Bourgeois, Bruno Escoffier, Vangelis Paschos. Efficient approximation of MIN SET COVER by "low-complexity" exponential algorithms. 2008. hal-00906972

HAL Id: hal-00906972

https://hal.science/hal-00906972

Preprint submitted on 20 Nov 2013

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

UNIVERSITE PARIS DAUPHINE

Laboratoire d'Analyse et Modélisation de Systèmes pour l'Aide à la Décision

CNRS UMR 7024

CAHIER DU LAMSADE

278

Octobre 2008

Efficient approximation of MIN SET COVER by "low-complexity" exponential algorithms

N. Bourjeois, B. Escoffier, V. Th. Paschos

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

Efficient approximation of MIN SET COVER by "low-complexity" exponential algorithms

N. Bourgeois B. Escoffier V. Th. Paschos

LAMSADE, CNRS UMR 7024 and Université Paris-Dauphine, France

{bourgeois,escoffier,paschos}@lamsade.dauphine.fr

October 13, 2008

Abstract

We study approximation of MIN SET COVER combining ideas and results from polynomial approximation and from exact computation (with non-trivial worst case complexity upper bounds) for **NP**-hard problems. We design approximation algorithms for MIN SET COVER achieving ratios that cannot be achieved in polynomial time (unless problems in **NP** could be solved by slightly super-polynomial algorithms) with worst-case complexity much lower (though super-polynomial) than those of an exact computation.

1 Introduction

Given a ground set C of cardinality n and a system $\mathcal{S} = \{S_1, \ldots, S_m\} \subset 2^C$, MIN SET COVER consists of determining a minimum size subsystem $\mathcal{S}'$ such that $\cup_{S \in \mathcal{S}'} S = C$. MIN SET COVER is a famous **NP**-hard problem dealt in the seminal paper [22].

For the last ten years, the issue of exact resolution of **NP**-hard problems by algorithms having provably non-trivial upper time-complexity bounds has been very actively studied (see, for instance, the surveys by [16, 26, 30]). Notable results for MIN SET COVER in this area are given in the papers [16, 18, 28].

Furthermore, very active research has been also conducted around approximation of MIN SET COVER by polynomial algorithms (see, for instance, [9, 20, 17, 21, 24, 27]). More precisely, it is proved in [21, 24] that MIN SET COVER is approximable in polynomial time within tight ratio $1+\ln|S^*|$ where S^* is a maximum-cardinality set in $\mathcal{S}$ while in [9] the same upper bound is shown for the weighted version of MIN SET COVER where a nonnegative weight is associated with every set in $\mathcal{S}$ and the objective becomes to minimize the total weight of a set cover. These ratios are attained by the natural greedy algorithm which, for the unweighted case, chooses to include in the solution one of the sets of maximum residual cardinality while, for the weighted case, it chooses to include in the solution one of the sets maximizing the ratio between residual cardinality and weight. In [27], it is shown that (in the unweighted case) the greedy MIN SET COVER-algorithm achieves a tight ratio of $O(\log n)$. In [13], using semi-local optimization techniques, a $(1/2) + \ln|S^*|$-approximation algorithm is given.

On the other hand, since the beginning of 90's, and using the celebrated PCP theorem ([1]), numerous natural hard optimization problems have been proved to admit more or less pessimistic inapproximability results. For instance, MIN SET COVER is inapproximable within approximation ratio better than $(1-\varepsilon)\ln n$, for every $\varepsilon > 0$, unless $\mathbf{NP} \subset \mathbf{DTIME}(\mathbf{n}^{\log\log \mathbf{n}})$ ([15]). Similar results have been provided for numerous other paradigmatic optimization problems, as MAX INDEPENDENT SET, MIN COLORING, etc. Such results exhibit large gaps between what it is possible to do in polynomial time and what becomes possible in exponential time.

Hence, for MIN SET COVER, a natural question is: how much time takes the computation of an r-approximate solution, for $r \in]1, \log n[$? Of course, we have a lower bound to this time (any polynomial in n, unless $\mathbf{NP} \subset \mathbf{DTIME}(\mathbf{n}^{\log\log \mathbf{n}})$, thanks to the inapproximability result) and also an upper bound (the running time of exact computation). But: *can we devise, for some ratio r, a r-approximate algorithm with an improved running time located somewhere between these bounds?, is this possible for any ratio r, i.e., can we specify a global relationship between running time and approximation ratio?.*

Here we try to bring answers to these questions by matching ideas and results from exact computation and from polynomial approximation. This issue has been marginally handled by [5] for minimum coloring. It has been also handled by [7, 8, 12], though in a different setting and with different objectives oriented towards development of fixed-parameter algorithms. Also a different but very interesting kind of trade-off between exact computation and polynomial approximation is settled by [29]. Note finally that in the same setting we handle in [6] MAX INDEPENDENT SET, MIN VERTEX COVER, MAX CLIQUE, MAX BIPARTITE SUBGRAPH and MAX SET PACKING.

In what follows, we set $\Delta = |S^*|$, where $S^* = \text{argmax}\{|S| : S \in \mathcal{S}\}$. Given an element $c \in C$, we denote by f_c the quantity $|\{S : c \in S\}|$, i.e., the number of sets in $\mathcal{S}$ containing c, and we set $f = \max\{f_c : c \in C\}$. Finally, we set $d = m + n$.

Let us note that in [20] it is proved that MIN SET COVER is approximable in polynomial time within ratio f. Improvement of this ratio down to either $f - c$, or to f/c for some fixed constant $c > 0$ is very unlikely given that would entail polynomial approximation of MIN VERTEX COVER within ratio $2 - \varepsilon$, for some fixed ε, fact very highly improbable ([23]).

Let $T(\cdot)$ be a super-polynomial and $p(\cdot)$ be a polynomial, both on integers. In what follows, using notations in [30], for an integer n, we express running-time bounds of the form $p(n) \cdot T(n)$ as $O^*(T(n))$ by ignoring, for simplicity, polynomial factors. We denote by $T(n)$ the worst-case time required to solve the considered combinatorial optimization problem with n variables. We recall (see, for instance, [14]) that, if it is possible to bound above $T(n)$ by a recurrence expression of the type $T(n) \leqslant \sum T(n - r_i) + O(p(n))$, we have $\sum T(n - r_i) + O(p(n)) = O^*(\alpha(r_1, r_2, \ldots)^n)$ where $\alpha(r_1, r_2, \ldots)$ is the largest zero of the function $f(x) = 1 - \sum x^{-r_i}$.

In what follows, in Sections 2 and 3, we study efficient approximation of MIN SET COVER by low-complexity exponential algorithms. We propose several techniques that allow to achieve non trivial tradeoffs between approximation and time complexity (depending either on d, n, or m). Note that the corresponding best known exact algorithms that we are aware of have complexity $O^*(1.23^d)$ ([28]), $O^*(2^n)$ ([5]) and $O^*(2^m)$ (brute force algorithm). We first show how to devise an approximate "pruning the search tree"-based algorithm: this algorithm allows for instance to compute a 7-approximate solution in time $O^*(1.0007^d)$. We also propose a greedy approach that outperforms the previous one when n is small (respect to m), and then show that these approaches are also pertinent in particular cases such as when the frequencies of elements are upper bounded. Finally, in Section 3 we improve some of the results of Section 2 by developing and analyzing a randomized approximation algorithm for MIN SET COVER.

In Section 4 we use a standard approximability preserving reduction from MIN DOMINATING SET to MIN SET COVER and show how results of Section 2 can be transferred from MIN SET COVER to MIN DOMINATING SET. This latter problem is defined as follows: given a graph $G(V, E)$, a dominating set is a subset $H \subset V$ such that, any vertex from $V - H$ is adjacent to some vertex from H; the objective is to determine a dominating set of minimum size.

In Section 5, we study MIN SET COVER but expressing the objective function as the maximization of the number of unused sets: one seeks a set cover $\mathcal{S}^*$ that maximizes $m - |\mathcal{S}^*|$. Though being obviously equivalent from an exact resolution viewpoint, approximation properties change. Polynomial approximation of this problem problem has already been studied in [10, 13, 3, 19]. In the same setting, studies have also been conducted for several other problems, for instance,

maximization of unused colors corresponding to the usual MIN COLORING problem, maximization of unused bins in the BIN PACKING problem, etc.

Maximizing the number of unused sets can be also seen as studying MIN SET COVER under the so-called *differential ratio*. Let $\mathtt{A}$ be a polynomial time approximation algorithm for an **NP**-hard problem Π, let $m(I,S)$ be the value of the solution S provided by $\mathtt{A}$ on an instance I of Π, and $\mathrm{opt}(I)$ be the value of the optimal solution for I. Finally, let $\omega(I)$ be the value of a worst solution of G defined as the value of an optimal solution for $\bar{\Pi}$, the combinatorial problem having the same constraints as Π but instead of minimizing the objective function of Π we wish to maximize it ($\omega(I) = m$ for MIN SET COVER, corresponding to the solution that takes all the sets). The *differential-approximation ratio* $\delta_{\mathtt{A}}(I)$ of an approximation algorithm $\mathtt{A}$ on I is defined as $\delta_{\mathtt{A}}(I) = (\omega(I) - m(I,S))/(\omega(I) - \mathrm{opt}(I))$. The usual ratio $\rho_{\mathtt{A}}(I) = m(I,S)/\mathrm{opt}(I)$ will sometimes be called *standard-approximation ratio* in what follows. For both ratios, the closer to 1, the better the approximability quality of an algorithm.

In general, no apparent links exist between standard and differential approximations in the case of minimization problems, in the sense that there is no evident transfer of a positive, or negative, result from one paradigm to the other. Hence a "good" differential approximation result does not signify anything for the behavior of the approximation algorithm studied when dealing with the standard framework and vice-versa.

Considering the differential approximation, it is proved in [2] (based upon a preliminary result by [11]) that MIN SET COVER is **Poly-DAPX**-hard (where **Poly-DAPX** is the class of problems approximable in polynomial time within differential ratios $1/f(|I|)$, where f is a polynomial of the instance size $|I|$). Moreover, for any $\varepsilon > 0$, there is no polynomial $m^{\varepsilon-1}$-differential approximation algorithm for MIN SET COVER unless $\mathbf{P} \neq \mathbf{NP}$.

Finally, in Section 5, we study efficient approximation of MIN SET COVER by low-complexity exponential algorithms in the differential paradigm.

2 Approximation with standard ratio

2.1 Approximately pruning the search tree

Pruning the search tree is one of the most classical techniques to get exact algorithms with non trivial exponential complexity. Here, we show that this technique can be adapted to get approximation algorithms realizing interesting tradeoffs between time complexity and approximation. The algorithm is based upon two improvements respect to an exact search tree algorithm. First, since we only seek an approximate solution, the algorithm may, when branching, make some "errors" by being "less careful" than an exact one. For instance, if one wants a 2-approximate solution, each time the algorithm branches on a set, in the case it puts it in the cover it can add another set (obtaining recursively a ratio 2): taking this additional set reduces the size of the remaining problem, thus reducing the complexity of the pruning algorithm. The second improvement consists of stoping the development of the tree before the end: if at some point the remaining instance is polynomially approximable within the ratio looked for, then no need to continue the branching.

Considering these two ways of better pruning the search tree, we propose the following algorithm `SC1`, parametrized by the ratio q we want to guarantee:

- fix $q \in \mathbb{N}^*$ and compute the largest integer p such that $\mathcal{H}(p) - 1/2 \leqslant q$, where $\mathcal{H}$ is the harmonic number sequence;
- repeat steps 1 to 4 below until C is covered:
  1. if there exists an item of C that belongs to a single subset $S \in \mathcal{S}$, add S to the solution;

2. if there exist two sets S, R in $\mathcal{S}$ such that S is included into R, remove S without branching;
3. if all the surviving subsets have cardinality at most p run the algorithm by [13] to compute a q-approximation of the optimal solution in the surviving instance;
4. determine q sets $S_1, \ldots, S_q$ from $\mathcal{S}$ such that $\cup_{i \leqslant q} S_i$ has maximum cardinality and perform the the following branching: either add every S_i to the solution (and remove $\cup_{i \leqslant q} S_i$ from C), or remove all of them.

Formally, until step 3 is executed, Algorithm SC1 returns:

$$\begin{aligned} &\texttt{SC1}(C, \mathcal{S}) = \\ &\operatorname{argmin} \left\{ |\texttt{SC1}\left(C, \mathcal{S} \setminus \{S_i : i \leqslant q\}\right)|, \left| \{S_i : i \leqslant q\} \cup \texttt{SC1}\left(C \setminus \bigcup_{i \leqslant q} S_i, \mathcal{S} \setminus \{S_i : i \leqslant q\}\right) \right| \right\} \end{aligned} \tag{1}$$

Proposition 1. *For any integer $q \geqslant 1$, Algorithm SC1 computes with running time $O^*(\alpha^d)$ a q-approximation of* MIN SET COVER*, where α is the solution of:*

$$x^{q(2+p)} - x^{q(1+p)} - 1 = 0 \tag{2}$$

and p is the largest integer such that $\mathcal{H}(p) - 1/2 \leqslant q$

Proof. The algorithm always returns a set cover, since it never removes from C an item that has not been covered yet. We now claim that the solution computed has size at most $q \times |\operatorname{opt}(I)|$, where I is the instance $(C, \mathcal{S})$ of MIN SET COVER. Obviously, this is the case if d is smaller than some bounded constant or if any set from $\mathcal{S}$ has size at most p: this corresponds to Step 3, which produces a $\mathcal{H}(p) - 1/2 \leqslant q$ approximate solution on the remaining instance.

Now, we deal with the branching step 4. Assume the ratio claimed is true for any instance such that $d < D$, consider an instance $(C, \mathcal{S})$ with $|C| + |\mathcal{S}| = D$, suppose that Algorithm SC1 branches on q sets just chosen in step 4 and fix some optimal solution $\mathcal{S}^*$. Denote finally by $\mathcal{S}^*_{\bar{q}}$ an optimal solution of the instance $(C, \mathcal{S} \setminus \{S_i : i \leqslant q\})$ and by $\mathcal{S}^*_{\bar{q},\bar{q}}$ an optimal solution of the instance $(C \setminus \cup_{i \leqslant q} S_i, \mathcal{S} \setminus \{S_i : i \leqslant q\})$. Then, two cases may occur, depending on the fact that $\mathcal{S}^*$ takes or not at least one S_i.

If, for any $i \leqslant q$, $S_i \notin \mathcal{S}^*$, then according to (1), we get:

$$|\texttt{SC1}\left(C, \mathcal{S}\right)| = |\texttt{SC1}\left(C, \mathcal{S} \setminus \{S_i, i \leqslant q\}\right)| \leqslant q \left|\mathcal{S}^*_{\bar{q}}\right| = q \left|S^*\right|$$

since the subfamily $\{S_i : i \leqslant q\}$ is not contained neither to $\mathcal{S}^*_{\bar{q}}$ nor to $\mathcal{S}^*$ and instance $(C, \mathcal{S} \setminus \{S_i, i \leqslant q\})$ has $d < D$.

Suppose now that there exist $k \geqslant 1$ sets S_i, $i \in \{1, \ldots, q\}$ such that $S_i \in \mathcal{S}^*$. Then, still according to (1):

$$\begin{aligned} |\texttt{SC1}(C, \mathcal{S})| &= \left| \texttt{SC1}\left(C \setminus \bigcup_{i \leqslant q} S_i, \mathcal{S} \setminus \{S_i : i \leqslant q\}\right) \right| + q \leqslant q \left|\mathcal{S}^*_{\bar{q},\bar{q}}\right| + q \\ &\leqslant q\left(|\mathcal{S}^*| - k\right) + q \leqslant q\left|\mathcal{S}^*\right| \end{aligned} \tag{3}$$

where inequality in (3) holds because instance $(C \setminus \cup_{i \leqslant q} S_i, \mathcal{S} \setminus \{S_i : i \leqslant q\})$ also has $d < D$.

Based upon the above, an easy induction on d establishes that the ratio claimed is true for any value of d.

Let us now study complexity of Algorithm SC1. Since any operation has a polynomial time execution, we only have to determine how many times it branches. The worst case complexity

of the algorithm on an instance with parameter d can be written as $p(d) \times T(d)$, where p is a fixed-degree polynomial and $T(d)$ the number of nodes of the search tree. Each time it branches, in the case the S_i's are removed, then d decreases by q. In the case the S_i's are taken in the cover, then $|\mathcal{S}|$ decreases by d. But, except maybe the last time it occurs, $\cup_{i \leqslant q} S_i$ contains at least $(p+1)q$ items, since it remains at least one subset that covers $p+1$ items or more in the surviving ground set. Then, the parameter of instance where all these sets are removed is at most $(m-q) + (n-(p+1)q) = d-(p+2)q$. So:

$$T(d) \leqslant T(d-q) + T(d-(p+2)q)$$

One can see that α^d, where α is the solution of (2) verifies this inequality. ∎

Table 1 gives complexity of Algorithm SC1 (basis of the exponential) for some values of the ratio q that one wishes to reach.

Ratio q	1	2	3	4	5	6	7	8
p	2	6	18	50	136	372	1014	2758
α	1.380	1.110	1.038	1.014	1.005	1.002	1.0007	1.0003

Table 1: Complexity of SC1 (basis of the exponential) for some values of q.

To conclude this section, let us remark that we can derive from the previous analysis a result on the running time of `SC1` with respect to m instead of d. Measuring complexity with m instead of d may be useful if m is "small", for instance if it is smaller than n.

Proposition 2. *For any integer $q \geqslant 1$, Algorithm `SC1` computes a q-approximation of* MIN SET COVER *in $O^*(2^{m/q})$*

Proof. The ratio claimed is proved in Proposition 1. For its running time just observe that, each time it branches, it removes exactly q subsets from $\mathcal{S}$, i.e., $T(m) \leqslant 2T(m-q)$ that gives $T(m) = O^*(2^{m/q})$. ∎

Table 2 deals with Proposition 2 and measures running time of `SC1` for some values of q. The exact algorithm is in $O^*(2^m)$, since there is no better algorithm that we are aware of.

Ratio q	1	2	3	4	5	6	7	8
Time	2^m	1.414^m	1.260^m	1.189^m	1.149^m	1.123^m	1.104^m	1.091^m

Table 2: Complexity of SC1 with respect to m for some values of q.

2.2 A "greedy" approach

In the previous approach, we adapted the technique of pruning the search tree to get approximation algorithms. Here, we tackle the efficient approximation of MIN SET COVER by using the classical greedy technique. In particular, this leads to good tradeoffs between approximation and complexity measured with respect to n (see for instance Corollary 1).

Suppose that an algorithm `E` solves MIN SET COVER within approximation ratio ρ, for some $\rho > 1$ and consider the following algorithm, denoted by `SC2`, that uses a greedy approach to decrease the size of MIN SET COVER-instance:

1. fix $\epsilon > 0$ and $q > 1$ and compute all collections $Z \subset \mathcal{S}$ of size at most $1/\epsilon$; if some Z are covers of C, return the smallest among them;

2. compute and store a set $S^* \subset \mathcal{S}$ of maximum cardinality; $C := C \setminus S^*$; $\mathcal{S} := \mathcal{S} \setminus \{S^*\}$;

3. repeat step 2 until the surviving set C verifies $|C| \leqslant ne^{-r+1}$; let $\mathcal{S}''$ be the the collection of sets S^* computed during all the executions of step 2;

4. output $\mathcal{S}' = \mathcal{S}'' \cup \mathtt{E}(C)$.

Proposition 3. *Assume that an algorithm* `E` *is able to compute, for some $\rho \geqslant 1$, a ρ-approximation of* MIN SET COVER *with running time $O^*(\alpha_1^n \alpha_2^m)$. Then,* `SC2` *computes for any $r \geqslant 0$ and for any $\epsilon > 0$, a $(r+\rho)$-approximation of* MIN SET COVER *with complexity $O^*(\alpha_1^{ne^{-r+\epsilon}} \alpha_2^m)$*

Proof. Let us first note that step 1 of Algorithm `SC2` is polynomial since ϵ is fixed. If a set cover of size at most $1/\epsilon$ exists in the instance, then an optimal solution is built in polynomial time; otherwise, $1/\varepsilon$ is a lower bound to the size of an optimal set cover.

For simplicity, index the surviving set C by the iteration of step 2 that has produced it ($C_0 = C$). So, execution of this step has stopped for the first i for which $|C_i| \leqslant ne^{-r+1}$. Obviously, at each step k at least $|C_{k-1}|/k^*$ elements are removed from C_{k-1}, where k^* is the size of an optimal solution. Indeed, these elements can be covered by at most k^* subsets and the algorithm includes in the solution one of the largest among them. When k subsets have been chosen, the size of the remaining set C_k to be covered is so at most $n(1-1/k^*)^k \leqslant ne^{-k/k^*}$.

Algorithm `E` completes the cover under construction by running on some instance C_k with complexity $O^*(\alpha_1^{|C_k|} \alpha_2^m)$. In all, if step 2 is executed k times, Algorithm `SC2` achieves ratio at most:

$$\frac{k+\rho k^*}{k^*} = \rho + \frac{k}{k^*} \tag{4}$$

If k is such that $|C_k| < ne^{-r+\epsilon}$ and $|C_{k-1}| \geqslant ne^{-r+\epsilon}$, then:

$$\frac{k}{k^*} \leqslant \frac{k-1}{k^*} + \epsilon \leqslant \log\left(\frac{n}{|C_{k-1}|}\right) + \epsilon \leqslant r \tag{5}$$

Combination of (4) and (5) immediately derives the approximation ratio claimed.

On the other hand, given that steps 1 and 2 are polynomial, the overall complexity of `SC2` is dominated by the complexity of `E`, that is $O^*(\alpha_1^{ne^{-r+\epsilon}} \alpha_2^m)$, as claimed. ∎

A similar approach is used by [5] in order to prove that for any $k \in \mathbb{N}$ it is possible to get a $(1 + k/opt)$-approximation of minimum coloring with running time $O^*(\gamma^n + \alpha^{ne^{-k/opt}})$, where γ^n (resp., α^n) is the running time of some exact algorithm for MAX INDEPENDENT SET (resp., minimum coloring). At first glance, our result might seem to be a generalization of the result by [5], since minimum coloring can be naturally modeled as a MIN SET COVER (see [21]). But, this is not true for the following two reasons:

- when minimum coloring problem is handled as a MIN SET COVER problem, the size of the set-system (that is the number of the independent sets of the input graph) may grow exponentially with n; in this case, the part of the complexity depending on m is dominating;

- the fact that Proposition 3 derives ratios $1+r$ for any r and not only for multiples of $1/k^*$ is due to the fact that we can solve polynomially any instance whose cover is smaller than some bounded constant; this is no more possible when m grows exponentially with n.

Corollary 1. *If* `E` *stands for the inclusion-exclusion method by [5], then* `SC2` *computes for any $r \geqslant 0$ and for any $\epsilon > 0$, a $(1+r)$-approximation of* MIN SET COVER *with running time and space $O^*(2^{ne^{-r+\epsilon}})$.*

2.3 Approximation within ratio better than f

Recall that, as we mentioned in Section 1, MIN SET COVER is approximable in polynomial time within ratio f, but improvement of this ratio is very unlikely. Let us also note that Proposition 2 enables achievement of a rf-approximation for any r such that $rf \in \mathbb{N}^*$ (hence $1/f \leqslant r \leqslant 1$), with running time $O^*(2^{m/rf})$. In what follows we design another algorithm, denoted by `SC3`, that achieves the same ratio with lower complexity in the case where the maximum frequency is bounded above by a fixed constant. It works as follows:

- fix $p, q \in \mathbb{N}$ such that $p/q = r - (1/f)$ and $\lambda < r/2$;
- compute a first solution $\mathcal{S}_1$ as follows:
  1. pick some $c_1 \in C$; let F_1 be the family of sets in $\mathcal{S}$ that contain c_1 ($f_1 = |F_1| \leqslant f$); remove $\cup_{S \in F_1} S$ from C;
  2. repeat step 1 until C is empty; let t be the number of iterations of step 1; set $\mathcal{S}_1 = \cup_{i \leqslant t} F_i$, where F_i is the sub-family of $\mathcal{S}$ removed at iteration i;
- compute a second cover $\mathcal{S}_2$ as follows:
  - set $\mathcal{S}_2 = \emptyset$;
  - for any $i \leqslant t$, fix at random a partition of F_i into q sets and add to $\mathcal{S}_2$ p among these sets such that their union has maximum size;
  - run an exact algorithm `E` on the remaining instance and add the result to $\mathcal{S}_2$;
- compute a last solution $\mathcal{S}_3$ as follows:
  - fix at random an equipartition of $\mathcal{S}$ into m/rf sets $\mathcal{F}_1, \ldots, \mathcal{F}_{m/rf}$ of size rf;
  - compute all the unions of at most $\lambda m/rf$ $\mathcal{F}_i$'s; if some of them form a cover of C set $\mathcal{S}_3$ a smallest among them;
- output $\mathcal{S}' = \text{argmin}\{|\mathcal{S}_1|, |\mathcal{S}_2|, |\mathcal{S}_3|\}$.

Proposition 4. *Assume that an algorithm* `E` *is able to compute* MIN SET COVER *with running time* $O^*(\alpha_1^n \alpha_2^m)$. *Then,* `SC3` *computes, for any* $r \geqslant 0$, *an* rf*-approximation of* MIN SET COVER *with complexity* $O^*(\max(\beta^{m/rf}, \alpha_1^{(1-\mu)n} \alpha_2^{(1-\lambda\mu)m}))$, *where:*

$$\begin{aligned} \beta &= \frac{1}{\left(\frac{\lambda}{r}\right)^{\frac{\lambda}{r}} \left(1 - \frac{\lambda}{r}\right)^{1-\frac{\lambda}{r}}} < 2 \\ \mu &= r - \frac{1}{f} \end{aligned}$$

Proof. Let $\mathcal{S}^*$ be some optimal set cover. Since any $c_i \in C$ has to be covered, $|\mathcal{S}^*| \geqslant t$.

Assume first $|\mathcal{S}^*| \geqslant t/r$. Then:

$$\frac{|\mathcal{S}_1|}{|\mathcal{S}^*|} \leqslant \frac{tf}{\frac{t}{r}} = rf \tag{6}$$

On the other hand, assume $t \geqslant \lambda m/rf$ and $\sum f_i \leqslant \lambda m$. Then:

$$\frac{|\mathcal{S}_1|}{|\mathcal{S}^*|} \leqslant \frac{\lambda m}{t} \leqslant \frac{\lambda m}{\frac{\lambda m}{rf}} = rf \tag{7}$$

So, under the assumptions just made, and given that solution $\mathcal{S}_1$ is computed in polynomial time, a ratio rf is achieved in polynomial time.

We now consider that $|\mathcal{S}^*| \leqslant t/r$ and either $t \leqslant \lambda m/rf$, or $\sum f_i \geqslant \lambda m$. In the first case, we get $|\mathcal{S}^*| \leqslant \lambda m/r^2 f$. Consider the set $\mathcal{F} = \cup_{\mathcal{F}_i \cap \mathcal{S}^* \neq \emptyset} \mathcal{F}_i$. Of course $|\mathcal{F}| \leqslant rf|\mathcal{S}^*|$ and it is a cover. Since the number of $\mathcal{F}_i$ that intersects $\mathcal{S}^*$ is at most $|\mathcal{S}^*|$, an exhaustive search on all the unions of at most $\lambda m/r^2 f$, $\mathcal{F}_i$'s reaches it; hence, $|\mathcal{S}_3| \leqslant |\mathcal{F}|$. So:

$$\frac{|\mathcal{S}_3|}{|\mathcal{S}^*|} \leqslant \frac{|\mathcal{F}|}{|\mathcal{S}^*|} \leqslant rf \tag{8}$$

Using Stirling's formula, computation of $\mathcal{S}_3$ takes time at most $\binom{m/rf}{\lambda m/r^2 f} = O(\beta^{m/rf})$.

Assume finally: $t \geqslant \lambda m/rf$ and $\sum_{i \leqslant t} f_i \geqslant \lambda m$. Solution $\mathcal{S}_2$ contains at most $\sum_{i \leqslant t} f_i \times p/q + |\mathcal{S}^*|$ subsets. So:

$$\frac{|\mathcal{S}_2|}{|\mathcal{S}^*|} \leqslant \frac{\sum\limits_{i \leqslant t} f_i \left(r - \frac{1}{f}\right)}{|\mathcal{S}^*|} + 1 \leqslant \frac{ft\left(r - \frac{1}{f}\right)}{t} + 1 \leqslant rf \tag{9}$$

Computation of $\mathcal{S}_2$ requires running of the exact algorithm `E` on an instance whose size (m', n') is:

$$\begin{aligned} n' &\leqslant n\left(1 - \frac{p}{q}\right) \leqslant n\left(1 - r + \frac{1}{f}\right) \\ m' &\leqslant m - \frac{p}{q}\sum_{i \leqslant t} f_i \leqslant m\left(1 - \lambda\left(r - \frac{1}{f}\right)\right) \end{aligned}$$

where the first inequality holds since in each $\cup_{S \in F_i} S$ at least a fraction p/q of elements is covered. This, together with (6), (7), (8) and (9) conclude the proof of the proposition. ∎

For instance, if we wish to get a ratio $rf = f/2$, we can use the exact MIN SET COVER-algorithm (with running time $O^*(1.236^{n+m})$) and fix $\lambda = 1/6 < r/2$. Then the running time of `SC3` is $O^*(\max(1.89^{m/rf}, 1.236^{(1/2+1/f)n+(11/12+1/6f)m}))$. In fact, optimal value for λ obviously depends on maximum frequency f and on m/n. According to Propositions 1 and 2, `SC1` has running time $O^*(\min(2^{m/rf}, \alpha(rf)^{n+m}))$.

A comparison of the running times of Algorithms `SC1`, `SC2` and `SC3` is not easy: when n is very small w.r.t. m, then `SC2` associated with the inclusion-exclusion algorithm by [5] dominates the two other algorithms, but in the other cases, each of the three algorithms can dominate the others depending on the approximation ratio we seek and on the relative values of n and m.

3 A Randomized Algorithm

We give in this section a randomized algorithm for MIN SET COVER that improves results in Section 2. But first, consider the following (deterministic) algorithm `Divide` that will be used later:

1. group the m sets of $\mathcal{S}$ into $m/2$ groups of size 2, $\mathcal{F}_i = \{S_{2i-1}, S_{2i}\}$;
2. consider the $2^{m/2}$ subsets of $\mathcal{S}$ constituted by either none or the two sets S_{2i-1}, S_{2i} of any group $\mathcal{F}_i$;
3. for any of the subsets produced at step 2, test if this is a cover of C;
4. output the best solution computed.

Lemma 1. *`Divide` is a 2-approximation algorithm for* MIN SET COVER. *Its running time is* $O^*(2^{m/2})$.

Proof. Consider an optimal solution $\mathcal{S}^*$ and the particular solution $\mathcal{S}'$ defined as follows:

- if S_{2i-1} or S_{2i} belongs to $\mathcal{S}^*$, then both S_{2i-1} and S_{2i} belong to $\mathcal{S}'$;
- otherwise, neither S_{2i-1} nor S_{2i} belong to $\mathcal{S}'$.

Note that $\mathcal{S}'$ is a particular solution built by `Divide` and, since $\mathcal{S}^* \subseteq \mathcal{S}'$, it is a feasible solution. Furthermore, its approximation ratio is 2 since in each groupe $\mathcal{F}_i$ we take at most twice the number of subsets in $\mathcal{S}^*$. ∎

Note that `Divide` can be easily generalized by partitioning in m/q groups $\mathcal{F}_i$ of size q. This would lead to a q-approximation algorithm, running in time $O^*(2^{m/q})$. This is the same bound as in Proposition 2.

We now show how to use randomization to get a q-approximation algorithm (for $q < 2$) in time much smaller than $O^*(2^{m/q})$. The idea of the randomized algorithm is the following. The worst case for `Divide` is a ratio 2, but it may perform better. Indeed, when both S_{2i-1} and S_{2i} belong to $\mathcal{S}^*$ then on the set $\mathcal{F}_i = (S_{2i-1}, S_{2i})$, `Divide` has ratio 1. If this is the case for a significant part of the $\mathcal{F}_i$'s, then the ratio achieved by `Divide` is better than 2. If we partition $\mathcal{S}$ in groups of size 2 at random, we estimate the probability that it achieves a ratio $r < 2$. Unfortunately, this probability decreases exponentially to 0. But, by repeating this partition an exponential (sufficiently large) number of times, we get an r approximation with probability 1. This is the idea leading to Algorithm `RandomDivide`. The interesting fact is that, with a suitable choice of the number of repetitions, the global running time is much smaller than the running time $O^*(2^{m/r})$ claimed in Proposition 2.

In what follows, given a partition $P = (\mathcal{F}_1, \ldots, \mathcal{F}_{m/2})$ of $\mathcal{S}$ in groups of size 2, and an optimal solution $\mathcal{S}^*$, we denote by q_1 and q_2 the number of groups $\mathcal{F}_i$ where $\mathcal{S}^*$ takes, respectively, one or two sets. Then, as shown in the proof of Lemma 1, we return a solution of size at most $2q_1 + 2q_2$. Hence, the ratio ρ of this solution verifies:

$$\rho \leqslant \frac{2q_1 + 2q_2}{q_1 + 2q_2} = 2 - \frac{2q_2}{q_1 + 2q_2} \tag{10}$$

Then, P being a random partition of $\mathcal{S}$ in groups of size 2, assume that we are able to lower bound the probability $\Pr[q_2 \geqslant km]$ that q_2 equals at least km by a function $t(k, m)$. and consider the following algorithm, denoted `RandomDivide` parameterized by the ratio $r < 2$ that we wish to guarantee:

1. set $k = 1/r - 1/2$ and apply $m/t(k, m)$ times Algorithm `Divide` using randomized (independent) partitions of $\mathcal{S}$;
2. output the best solution among the solutions computed at step 1.

Proposition 5. *`RandomDivide` computes with probability at least* $1 - e^{-m}$ *an* r*-approximate solution for* MIN SET COVER *in time* $O^*(2^{m/2}/t(1/r - 1/2, m))$.

Proof. If, for at least one random partition, q_2 is at least km, then, using (10) (and $q_1 + q_2 \leqslant m/2$), the solution computed has ratio at most $1/(k + 1/2) = r$.

Since the probability that q_2 is smaller than km is at most $1 - t(k, m)$, the probability p that it is always smaller than km verifies:

$$p \leqslant (1 - t(k, m))^{\frac{m}{t(k,m)}} \leqslant e^{-m}$$

and the result follows. ∎

Now, the main part of randomization consists of lower bounding probability $\Pr[q_2 \geqslant km]$. Before explaining how to obtain such a bound, let us present the results achieved by `RandomDivide(r)`. Figure 1 and Table 3 compare the running times (more exactly the basis of the exponential in the running time) of `RandomDivide` versus a deterministic algorithm in $2^{m/r}$. In Table 3 (the exponential basis of) the number of random partitions we use is also given.

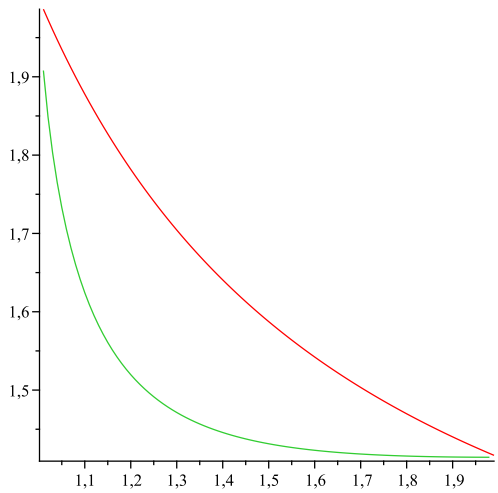

Figure 1: `RandomDivide` (lower curve) versus $2^{m/r}$

Ratio	$2^{1/r}$	`RandomDivide`	# of random partitions
1.1	1.88	1.63	1.15
1.2	1.79	1.52	1.08
1.3	1.71	1.48	1.05
1.4	1.65	1.55	1.03
1.5	1.59	1.44	1.02
1.6	1.55	1.43	1.007
1.7	1.51	1.42	1.003
1.8	1.47	1.42	1.002
1.9	1.45	1.42	1.0003

Table 3: `RandomDivide` versus $2^{m/r}$

Let us now compute the probabilities; we define the function f (for $0 < x < y$) as:

$$f(x, y) = \frac{y^y}{x^x (y-x)^{y-x}}$$

Lemma 2. *Let P be a random partition of $\mathcal{S}$ into groups of size 2. Let λ be such that $\mathcal{S}^* = \lambda m$. The probability $\Pr[q_2 = km]$ that $q_2 = km$ verifies:*

$$\Pr\left[q_2 = km\right] = \Theta\left(p(m)\alpha(k, \lambda)^m\right)$$

for some suitable polynomial p, where α is given by:

$$\alpha(k,\lambda) = \frac{f\left(k,\frac{1}{2}\right) f\left(\lambda-2k,\frac{1}{2}-k\right) 2^{\lambda-2k}}{f(\lambda,1)}$$

Proof. Consider a partition P of $\mathcal{S}$ into groups $\mathcal{F}_1,\ldots,\mathcal{F}_{m/2}$ of size 2. We have $\binom{m}{\lambda m}$ possibilities for the λm subsets of $\mathcal{S}^*$ in $\mathcal{S}$. Among these possibilities, to get $q_2 = km$, we have:

- $\binom{m/2}{km}$ choices for the km groups with two sets in the optimum solution;
- $\binom{(m/2)-km}{\lambda m-2km}$ choices for the $\lambda m - 2km$ groups with one set in the optimum solution;
- $2^{\lambda m-2km}$ choices for placing these $\lambda m - 2km$ sets, since there are two possible places in each group.

In all, the probability $\Pr[q_2 = km]$ is:

$$\Pr\left[q_2 = km\right] = \frac{\binom{m/2}{km} \times \binom{m/2-km}{\lambda m-2km} \times 2^{\lambda m-2km}}{\binom{m}{\lambda m}} \tag{11}$$

Now, using Stirling's formula, we have, for any s and t:

$$\binom{tm}{sm} = q(m) \times \left(\frac{t^t}{s^s(t-s)^{t-s}}\right)^m = q(m) \times (f(s,t))^m \tag{12}$$

for some suitable polynomial q.

The result follows from equations 11 and 12. ∎

Of course we have to consider the worst case for the size of an optimum solution, i.e. the value λ leading to the worst value for $\Pr[q_2 \geqslant km]$. Since only the exponential part matters, the bound is:

$$g(k) = \min_{0\leqslant\lambda\leqslant 1} \max_{l\geqslant k}\{\alpha(l,\lambda)\} \tag{13}$$

Then, according to Lemma 2 and using (13), we can choose in the definition of `RandomDivide` $t(k,m) = c \times p(m)g(k)^m$, for some suitable constant c and polynomial p.

Note that, in order to simplify the computation of $g(k)$, we can try to study directly the quantity $\min_\lambda\{\alpha(k,\lambda)\}$ (for a fixed $k < 1/2$). Unfortunately, this value is 0: this is very natural since, when $\mathcal{S}^*$ is very close to m, q_1 is close 0 and q_2 close to $m/2$, hence $q_2 \geqslant km$ but $q_2 \neq km$. In other words, to get the result, we have to directly bound $\Pr[q_2 \geqslant km]$ since we cannot only lower bound $\Pr[q_2 = km]$. But, a remark allows to significantly simplify study of $\alpha(k,\lambda)$: one can show that $\alpha(k,2k) = 1$. Using this, we can easily, by a simple function minimization, determine (eventually using a computer) $g(k)$.

4 Application to MIN DOMINATING SET

Consider the following reduction from MIN DOMINATING SET to MIN SET COVER, originally proposed by [25]. Let $G(V,E)$ be an instance of MIN DOMINATING SET. We construct an instance $I(\mathcal{S},C)$ of MIN SET COVER as follows: $C = V$, $\mathcal{S} = \{S_v = \{v\}\cup\Gamma(v), v \in V\}$, where $\Gamma(v)$ is the set of neighbors of vertex v ($|\mathcal{S}| = |V|$). Consider now a cover $\mathcal{S}' = \{S_{v_1},\ldots,S_{v_k}\}$ of C. Obviously, the set $\{v_1,\ldots,v_k\}$ is a dominating set of G, since set S_{v_i} (resp., vertex v_i) covers (resp., dominates) elements corresponding to vertex v_i itself and to its adjacent vertices.

Proposition 6. *Assume there exists an r-(standard)-approximation algorithm* `A` *for* MIN SET COVER *($r \geqslant 1$) with running time $O^*(\alpha_1^n\alpha_2^m)$. Then, an r-(standard)-approximation for* MIN DOMINATING SET *can be computed with running time $O^*((\alpha_1\alpha_2)^n)$.*

To the best of our knowledge, there does not exist exact algorithm for MIN SET COVER that is fast enough to allow SC2 or SC3 to be faster than SC1 when used to solve MIN DOMINATING SET. Recall that SC2 or SC3 are faster than SC1 only when m is smaller than n and this is not the case for the reduction just described. For instance, Table 4 shows performances of SC1 and SC2 for some ratio's values and with E having running time $O^*(1.236^n)$.

Ratio	SC1	SC2
1	1.904^n	1.526^n
2	1.232^n	1.335^n
3	1.077^n	1.272^n
4	1.028^n	1.249^n
5	1.010^n	1.241^n
6	1.004^n	1.238^n
7	1.001^n	1.237^n
8	1.0006^n	1.236^n

Table 4: Complexities of SC1 and SC2 when used for solving MIN DOMINATING SET.

5 Approximation with differential ratio

In this section we propose efficient approximation algorithm for MIN SET COVER using the differential ratio (or equivalently for the maximization of unused sets problem with the standard ratio). Recall that $\omega(I) = m$ and that the differential ratio is always smaller than 1. We propose three different algorithms: a first one, quite simple, that basically performs exhaustive search on solutions of particular sizes; a second one that prunes the search tree; a third one that combines the first exhaustive approach and a greedy method.

5.1 Generating candidate solutions

Consider the following algorithm, called DSC1:

1. fix a differential ratio $r \in \mathbb{Q}$ $(0 \leqslant r \leqslant 1)$ to be achieved;
2. compute all the combinations of at most $rm/(1+r)$ subsets from $\mathcal{S}$; if some of them cover C return a minimum-size one;
3. otherwise, compute all the combinations of at least $m/(1+r)$ subsets from $\mathcal{S}$ and return a minimum-size one.

Proposition 7. *For any $r \leqslant 1$, Algorithm DSC1 returns a r-differential approximation of* MIN SET COVER *with running time $O^*(\beta^m)$, where:*

$$\beta = \beta(r) = \frac{1+r}{r^{\frac{r}{1+r}}}$$

Proof. Assume first $|\mathcal{S}^*| \leqslant rm/(1+r)$. Then step 2 of DSC1 returns an optimal solution, so does step 3 if $|\mathcal{S}^*| \geqslant m/(1+r)$.

Assume now $rm/(1+r) \leqslant |\mathcal{S}^*| \leqslant m/(1+r)$. Then the solution returned by DSC1 has been computed at step 3 and its size is bounded above by $m/(1+r)$. So, denoting by $\mathcal{S}'$ this solution:

$$\frac{m - |\mathcal{S}'|}{m - |\mathcal{S}^*|} \geqslant \frac{m\left(1 - \frac{1}{1+r}\right)}{m\left(1 - \frac{r}{1+r}\right)} = r$$

On the other hand, the algorithm has running time:

$$\sum_{i=1}^{\frac{rm}{1+r}} \binom{m}{i} + \sum_{i=\frac{m}{1+r}}^{m} \binom{m}{i} = 2\sum_{i=1}^{\frac{rm}{1+r}} \binom{m}{i} \leqslant 2m\binom{m}{rm/(1+r)} = O^*\left(m^2\beta^m\right)$$

where β is as claimed in the proposition's statement. ∎

Remark that $|\mathcal{S}'|/|\mathcal{S}^*| \leqslant 1/r$. In other words, algorithm `DSC1` also guarantees standard approximation ratio $1/r$, but is dominated in complexity by the algorithms presented in Section 2 since $2^r \leqslant \beta(r) \leqslant 2$.

Table 5 gives execution times of Algorithm `DSC1` for some ratio's values.

q	1	2	3	4	5	6	7	8
$\beta(1/q)$	2	1.89	1.755	1.649	1.569	1.507	1.458	1.417

Table 5: Execution times of `DSC1` for some values of $q = 1/r$.

5.2 A branch-and-bound-like algorithm

We now devise a tree search based algorithm, in the spirit of Algorithm `SC1` seen in Section 2, that improves the results of Section 5.1. In order to guarantee a differential ratio, we have to ensure that the output solution is far from m, which was not true in Algorithm `SC1`. This leads to the modification of the branching step; the algorithm, called `DSC2`, works as follows:

- fix $q \in \mathbb{Q}$;
- repeat steps 1 to 4 below until C is covered:
  1. if there exists an item of C that belongs to a single subset $S \in \mathcal{S}$, it adds S to the solution;
  2. if there exist two sets S, R in $\mathcal{S}$ such that S is included into R, S is removed without branching;
  3. if all the surviving subsets have cardinality at most 2 exactly solve MIN SET COVER;
  4. determine q sets $S_1, \ldots, S_q$ from $\mathcal{S}$ such that S_1 is of maximum cardinality and $\cup_{2\leqslant i\leqslant q} S_i$ has maximum cardinality in $C \setminus S_1$ and perform the the following branching: either add S_1 in the solution, or do not add it and add $\{S_i : 2 \leqslant i \leqslant q\}$.

Formally, until step 3 is executed, Algorithm `DSC2` returns:

$$\begin{aligned}&\texttt{DSC2}(C, \mathcal{S}) = \\ &\operatorname{argmin}\left\{ |S_1 \cup \texttt{DSC2}\left(C \setminus S_1, \mathcal{S} \setminus \{S_1\}\right)|, \left|\{S_2, \ldots, S_q\} \cup \texttt{DSC2}\left(C \setminus \bigcup_{2\leqslant i\leqslant q} S_i, \mathcal{S} \setminus \{S_1, \ldots, S_q\}\right)\right| \right\}\end{aligned} \tag{14}$$

On the other hand, when maximum cardinality of the instance is at most 2, MIN SET COVER is polynomial since it becomes an edge covering problem ([4]).

Proposition 8. *For any integer $q \geqslant 1$, algorithm `DSC2` computes an $1/q$-differential approximation of* MIN SET COVER *with running time $O^*(\alpha^d)$, where α is the solution of:*

$$x^{4q-3} - x^{4q-7} - 1 = 0 \tag{15}$$

Proof. Algorithm DSC2 obviously returns a set cover. Assume that $|\mathcal{S}| - |\operatorname{opt}(C,\mathcal{S})| \leqslant q(|\mathcal{S}| - |\texttt{DSC2}(C,\mathcal{S})|)$ for any instance $(C,\mathcal{S})$ such that $|C|+|\mathcal{S}| < D$, consider some instance $(C,\mathcal{S})$, where $|C|+|\mathcal{S}| = D$, set $|\mathcal{S}| = m$, fix some optimal solution $\mathcal{S}^*$, and denote by $\mathcal{S}^*_{\bar{1}}$ an optimal solution of the instance $(C\setminus S_1, \mathcal{S}\setminus\{S_1\})$ and by $\mathcal{S}^*_{\bar{q},\bar{q}}$ an optimal solution of the instance $(C\setminus\cup_{i\leqslant q}S_i, \mathcal{S}\setminus\{S_i : i \leqslant q\})$.

If $S_1 \in \mathcal{S}^*$, then according to (14), we get:

$$m - |\texttt{DSC2}(C,\mathcal{S})| = m - 1 - |\texttt{DSC2}\,(C\setminus S_1, \mathcal{S}\setminus\{S_1\})| \geqslant \frac{m-1-\left|\mathcal{S}^*_{\bar{1}}\right|}{q} \geqslant \frac{m-|\mathcal{S}^*|}{q}$$

since instance $(C\setminus S_1, \mathcal{S}\setminus\{S_1\})$ under consideration has $d < D$.

If, on the other hand, $S_1 \notin \mathcal{S}^*$, let $k \leqslant q-1$ be the number of subsets that belong to $\mathcal{S}^*$ and our algorithm added to the solution. Then, always according to (14):

$$\begin{aligned} m - |\texttt{DSC2}(C,\mathcal{S})| &= m - q - |\texttt{DSC2}\,(C\setminus\cup_{i\leqslant q}S_i, \mathcal{S}\setminus\{S_i : i\leqslant q\})| + 1 \\ &\geqslant \frac{m-q-\left|\mathcal{S}^*_{\bar{q},\bar{q}}\right|}{q} + 1 \qquad (16) \\ &\geqslant \frac{m-q-|\mathcal{S}^*|+k}{q} + 1 \;\geqslant\; \frac{m-|\mathcal{S}^*|}{q} \end{aligned}$$

where inequality in (16) holds because instance $(C\setminus\cup_{i\leqslant q}S_i, \mathcal{S}\setminus\{S_i : i\leqslant q\})$ also has $d < D$.

Based upon the above, an easy induction on d establishes that the ratio claimed is true for any value of d.

Let us now study complexity of Algorithm DSC2. Any time it branches, every set $S_i \cap (C\setminus\cup_{j<i}S_j), i \leqslant q$ contains at least 3 items (otherwise the surviving instance would be polynomial and would met step 3). Then, $|\cup_{i\leqslant q} S_i| \geqslant 3q$, and the parameter of instance were all these sets except S_1 are taken is at most $(m-q)+(n-3(q-1)) = d-4q+3$. Consequently, $T(d) \leqslant T(d-4)+T(d-4q+3)$ and α^d, where α is the solution of (15) verifies this inequality. ∎

Table 6 presents examples of running time (basis of the exponential) for Algorithm DSC2 for some values of q (ratio achieved $1/q$).

q	1	2	3	4	5	6	7	8
α	1.380	1.167	1.119	1.095	1.081	1.071	1.063	1.057

Table 6: Complexity (basis of the exponential) of DSC2 for some values of $q = 1/r$.

As in the case of standard approximation paradigm (Proposition 2), the following analogous result holds here.

Proposition 9. *For any integer $q \geqslant 1$, Algorithm DSC2 computes an $1/q$-differential approximation for* MIN SET COVER *with running time $O^*(\gamma^m)$, where γ is solution of:*

$$\gamma^q - \gamma^{q-1} - 1 = 0 \qquad (17)$$

In order to show complexity claimed in Proposition 9 just observe that each time DSC2 branches, either 1 (if taken) or q (if dismissed) subsets are fixed. So, $T(m) \leqslant T(m-1)+T(m-q)$ that leads to (17).

In Table 7 examples of running times are given for Algorithm DSC2 are given for some values of q (ratio $1/q$).

q	1	2	3	4	5	6	7	8
Time	2^m	1.618^m	1.466^m	1.380^m	1.345^m	1.285^m	1.255^m	1.232^m

Table 7: Complexity of DSC2 for some values of $q = 1/r$.

5.3 Combining greedy and exhaustive approaches

We now propose an algorithm, called `DSC3` that combines greedy and exhaustive methods in the same spirit as in Section 2.2. Assume in what follows an exact Algorithm `E` that is able to solve (exactly) MIN SET COVER with running time $O^*(\alpha_1^m \alpha_2^n)$.

1. fix: a ratio $r \in \mathbb{Q}$ $(0 \leqslant r \leqslant 1)$ to be achieved, a $\lambda < 1$ and $\epsilon > 0$;
2. compute all the combinations of at least $(r\lambda + 1 - r)m$ subsets from $\mathcal{S}$; let $\mathcal{S}_1$ be one of the minimum-size covers so computed;
3. compute and store a set $S^* \subset \mathcal{S}$ of maximum cardinality; $C := C \setminus S^*$; $\mathcal{S} := \mathcal{S} \setminus \{S^*\}$;
4. repeat step 3 until the surviving set C verifies $|C| \leqslant ne^{-(1-r)((1/\lambda)-1)}$; $\mathcal{S}_2 := \mathcal{S}'' \cup \mathtt{E}(C)$;
5. output $\mathcal{S}' = \text{argmin}\{|\mathcal{S}_1|, |\mathcal{S}_2|\}$.

Proposition 10. *Assume that an Algorithm `E` exactly solves* MIN SET COVER *with running time $O^*(\alpha_1^n \alpha_2^m)$. Then, `DSC3` computes, for any $r \geqslant 0$ and for any $\epsilon > 0$, an r-differential approximation of* MIN SET COVER *with complexity $O^*(\max\{\beta^m, \alpha_1^{ne^{-(1-r)(1/\lambda)+\epsilon}} \alpha_2^m\})$, where:*

$$\beta = \frac{1}{(r\lambda + 1 - r)^{r\lambda+1-r}(r - r\lambda)^{r-r\lambda}} < 2$$

Proof. Assume first $|\mathcal{S}^*| \geqslant (r\lambda + 1 - r)m$. Then, Algorithm `DSC3` is optimal. Furthermore, if $\lambda m \leqslant |\mathcal{S}^*| \leqslant (r\lambda + 1 - r)m$, then $\mathcal{S}_1$ is a cover of size $(r\lambda + 1 - r)m$ and:

$$\frac{m - |\mathcal{S}_1|}{m - |\mathcal{S}^*|} \geqslant \frac{m - m(r\lambda + 1 - r)}{m - \lambda m} = r$$

Assume now $\lambda m \geqslant |\mathcal{S}^*|$ and, as in the proof of Proposition 3 (Section 2.2), index the surviving set C by the iteration of step 3 that has produced it. At each iteration k, at least $|C_{k-1}|/|\mathcal{S}^*|$ elements are removed from C_{k-1}. Indeed, C_{k-1} can be covered by at most $|\mathcal{S}^*|$ subsets and `DSC3` adds the largest one to solution $\mathcal{S}_2$. When k subsets have been chosen, the size of the remaining set C_k to be covered is at most $n(1 - (1/|\mathcal{S}^*|))^k \leqslant ne^{-k/|\mathcal{S}^*|}$. Fix k such that $|C_k| < ne^{-(1-r)((1/\lambda)-1)+\epsilon}$ and $|C_{k-1}| \geqslant ne^{-(1-r)((1/\lambda)-1)+\epsilon}$. Note that $k \leqslant (1 - r)((1/\lambda) - 1)|\mathcal{S}^*|$.

According to our assumptions, Algorithm `E` runs on C_k with complexity $O^*(\alpha_1^{|C_k|} \alpha_2^m)$. Putting this together with the complexity of step 2 derives the overall complexity claimed.

In order to conclude the proof, remark that the differential ratio achieved by steps 3 and 4 is:

$$\begin{aligned}\frac{m - |\mathcal{S}_2|}{m - |\mathcal{S}^*|} &\geqslant \frac{m - |\mathcal{S}^*| - (1 - r)\left(\frac{1}{\lambda} - 1\right)|\mathcal{S}^*|}{m - |\mathcal{S}^*|} \geqslant \frac{m - \left(r + \frac{1}{\lambda} - \frac{r}{\lambda}\right)|\mathcal{S}^*|}{m - |\mathcal{S}^*|} \\ &\geqslant \frac{1 - (r\lambda + 1 - r)}{1 - \lambda} = r\end{aligned}$$

as claimed. ∎

For instance, if one wishes to attain ratio $r = 1/2$, one can use the (best known) exact algorithm for MIN SET COVER, whose running time is $O^*(1.236^{n+m})$ and one can fix $\lambda = 4/5$. Then, running time of DSC3 is $O^*(max(1.384^m, 1.236^{0.535n+m}))$.

Once again, optimal value for λ depends on quantity m/n. Also, it is easy to see that when ratio m/n is small enough, it is always possible to determine some λ such that DSC3 is faster than both DSC1 and DSC2.

References

[1] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and intractability of approximation problems. *J. Assoc. Comput. Mach.*, 45(3):501–555, 1998.

[2] C. Bazgan, B. Escoffier, and V. Th. Paschos. Completeness in standard and differential approximation classes: Poly-(D)APX- and (D)PTAS-completeness. *Theoret. Comput. Sci.*, 339:272–292, 2005.

[3] C. Bazgan, J. Monnot, V. Th. Paschos, and F. Serrière. On the differential approximation of MIN SET COVER. *Theoret. Comput. Sci.*, 332:497–513, 2005.

[4] C. Berge. *Graphs and hypergraphs.* North Holland, Amsterdam, 1973.

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

[6] N. Bourgeois, B. Escoffier, and V. Th. Paschos. Efficient approximation by "low-complexity" exponential algorithms. Cahier du LAMSADE 271, LAMSADE, Université Paris-Dauphine, 2008. Available at `http://www.lamsade.dauphine.fr/cahiers/PDF/cahierLamsade271.pdf`.

[7] L. Cai and X. Huang. Fixed-parameter approximation: conceptual framework and approximability results. In H. L. Bodlaender and M. A. Langston, editors, *Proc. International Workshop on Parameterized and Exact Computation, IWPEC'06*, volume 4169 of *Lecture Notes in Computer Science*, pages 96–108. Springer-Verlag, 2006.

[8] Y. Chen, M. Grohe, and M. Grüber. On parameterized approximability. In H. L. Bodlaender and M. A. Langston, editors, *Proc. International Workshop on Parameterized and Exact Computation, IWPEC'06*, volume 4169 of *Lecture Notes in Computer Science*, pages 109–120. Springer-Verlag, 2006.

[9] V. Chvátal. A greedy-heuristic for the set covering problem. *Math. Oper. Res.*, 4:233–235, 1979.

[10] M. Demange, P. Grisoni, and V. Th. Paschos. Differential approximation algorithms for some combinatorial optimization problems. *Theoret. Comput. Sci.*, 209:107–122, 1998.

[11] M. Demange and V. Th. Paschos. On an approximation measure founded on the links between optimization and polynomial approximation theory. *Theoret. Comput. Sci.*, 158:117–141, 1996.

[12] R. G. Downey, M. R. Fellows, and C. McCartin. Parameterized approximation problems. In H. L. Bodlaender and M. A. Langston, editors, *Proc. International Workshop on Parameterized and Exact Computation, IWPEC'06*, volume 4169 of *Lecture Notes in Computer Science*, pages 121–129. Springer-Verlag, 2006.

[13] R. Duh and M. Fürer. Approximation of k-set cover by semi-local optimization. In *Proc. STOC'97*, pages 256–265, 1997.

[14] D. Eppstein. Improved algorithms for 3-coloring, 3-edge-coloring, and constraint satisfaction. In *Proc. Symposium on Discrete Algorithms, SODA'01*, pages 329–337, 2001.

[15] U. Feige. A threshold of $\ln n$ for approximating set cover. *J. Assoc. Comput. Mach.*, 45:634–652, 1998.

[16] F. V. Fomin, F. Grandoni, and D. Kratsch. Some new techniques in design and analysis of exact (exponential) algorithms. Bulletin of the European Association for Theoretical Computer Science 87, pp. 47–77, 2005.

[17] O. Goldsmidt, D. S. Hochbaum, and G. Yu. A modified greedy heuristic for the set covering problem with improved worst case bound. *Inform. Process. Lett.*, 48:305–310, 1993.

[18] F. Grandoni. A note on the complexity of minimum dominating set. *J. Discr. Algorithms*, 4(2):209–214, 2006.

[19] R. Hassin and S. Khuller. z-approximations. *J. Algorithms*, 41:429–442, 2001.

[20] D. S. Hochbaum. Approximation algorithms for the set covering and vertex cover problems. *SIAM J. Comput.*, 11(3):555–556, 1982.

[21] D. S. Johnson. Approximation algorithms for combinatorial problems. *J. Comput. System Sci.*, 9:256–278, 1974.

[22] R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of computer computations*, pages 85–103. Plenum Press, New York, 1972.

[23] S. Khot and O. Regev. Vertex cover might be hard to approximate to within $2-\varepsilon$. In *Proc. Annual Conference on Computational Complexity, CCC'03*, pages 379–386, 2003.

[24] L. Lovász. On the ratio of optimal integral and fractional covers. *Discrete Math.*, 13:383–390, 1975.

[25] A. Paz and S. Moran. Non deterministic polynomial optimization problems and their approximations. *Theoret. Comput. Sci.*, 15:251–277, 1981.

[26] U. Schöning. Algorithmics in exponential time. In Volker Diekert and Bruno Durand, editors, *Proc. Symposium on Theoretical Aspects of Computer Science, STACS'05*, volume 3404 of *Lecture Notes in Computer Science*, pages 36–43. Springer-Verlag, 2005.

[27] P. Slavík. A tight analysis of the greedy algorithm for set cover. In *Proc. STOC'96*, pages 435–441, 1996.

[28] J. M. M. van Rooij and H. L. Bodlaender. Design by measure and conquer, a faster exact algorithm for dominating set. In S. Albers and P. Weil, editors, *Proc. International Symposium on Theoretical Aspects of Computer Science, STACS'08*, pages 657–668. Internationales Begegnungs- und Forschungszentrum f"ur Informatik (IBFI), Schloss Dagstuhl, Germany, 2008. Available at `http://drops.dagstuhl.de/opus/volltexte/2008/1329`.

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

[30] G. J. Wœginger. Exact algorithms for NP-hard problems: a survey. In M. Juenger, G. Reinelt, and G. Rinaldi, editors, *Combinatorial Optimization - Eureka! You shrink!*, volume 2570 of *Lecture Notes in Computer Science*, pages 185–207. Springer-Verlag, 2003.