Design by Measure and Conquer: Exact algorithms for dominating set

Johan M. M. van Rooij

Hans L. Bodlaender

Technical Report UU-CS-2009-025
November 2009

Department of Information and Computing Sciences
Utrecht University, Utrecht, The Netherlands
www.cs.uu.nl

ISSN: 0924-3275

Department of Information and Computing Sciences
Utrecht University
P.O. Box 80.089
3508 TB Utrecht
The Netherlands

Design by Measure and Conquer: Exact algorithms for dominating set*

Johan M. M. van Rooij and Hans L. Bodlaender
Institute of Information and Computing Sciences, Utrecht University
P.O.Box 80.089, 3508 TB Utrecht, The Netherlands
jmmrooij@cs.uu.nl, hansb@cs.uu.nl

Abstract

The *measure and conquer* approach has proven to be a powerful tool to *analyse* exact algorithms for combinatorial problems, like Dominating Set and Independent Set. In this paper, we propose to use measure and conquer also as a tool in the *design* of algorithms. In an iterative process, we obtain a series of *branch and reduce* algorithms. A mathematical analysis of an algorithm in the series with measure and conquer results in a quasiconvex programming problem. The solution by computer to this problem not only gives a bound on the running time, but can also give a new reduction rule, thus giving a new, possibly faster algorithm. This makes *design by measure and conquer* a form of *computer aided algorithm design*.

We apply the methodology to a set cover modelling of the Dominating Set problem and obtain the currently fastest known exact algorithm for Dominating Set: an algorithm that uses $\mathcal{O}(1.4969^n)$ time and only polynomial space, while the previous fastest algorithm uses exponential space.

1 Introduction

Although the design of fast exponential time algorithms for finding exact solutions to $\mathcal{NP}$-hard problems such as Independent Set and Travelling Salesman dates back to the sixties and seventies, there has been a renewed interest in them over the last decade. See for example the results on Independent Set in the 1970s by Tarjan and Trojanowski [22, 23] and the more recent results by Robson [19, 20]. A number of different techniques have been developed to design and analyse these and other exponential time algorithms. Many examples of these can be found in a series of surveys on the topic [6, 15, 21, 28, 29].

An important paradigm for the design of exact algorithms is *branch and reduce*, pioneered in 1960 by Davis and Putnam [2]. Typically, in a branch and reduce algorithm, a collection of reduction rules and branching rules are given. Each reduction rule simplifies the instance to an equivalent, simpler instance. If no reduction rule applies, the branching rules generate a collection of two or more instances, on which the algorithm recurses.

A recent breakthrough in the analysis of branch and reduce algorithms is *measure and conquer*, which has been introduced by Fomin, Grandoni and Kratsch [5]. The measure and conquer approach helps to obtain good upper bounds on the running time of branch and reduce algorithms, often improving upon the currently best known bounds for exact algorithms. It has been used

*This paper is the first full description of our work from which a preliminary version appeared at the Symposium on Theoretical Aspects of Computer Science (STACS) 2008 [25]. In this paper, all running times are improved compared to this preliminary version by choosing a slightly better measure, similar to [8]. This leads to new and improved running times for the Dominating Set problem. Also, for the first time, we give full proofs of the running times of the algorithms obtained in the improvement series of the design by measure and conquer process.

successfully on Dominating Set [5], Independent Set [1, 5], Dominating Clique [17], Connected Dominating Set [7], Independent Dominating Set [11], Edge Dominating Set [26], the number of minimal dominating sets [8], and many others.

In this paper, we show that the measure and conquer approach can not only be used for the *analysis* of exact algorithms, but also for the *design* of such algorithms. More specifically, measure and conquer uses a non-standard size measure for instances. This measure is based on weight vectors which are computed by solving a numerical problem: a quasiconvex program. Analysis of the solution of this quasiconvex program yields not only an upper bound to the running time of the algorithm, but also shows where we could possibly improve the algorithm. This can lead to a new rule, which we add to the branch and reduce algorithm.

We apply this *design by measure and conquer* methodology to a Set Cover modelling of the Dominating Set problem, identical to the setting in which measure and conquer was first introduced. If we start with the trivial algorithm, then we can obtain in a number of steps the original algorithm of Fomin et al. [5]. With additional steps, we obtain a faster algorithm: this results in Theorem 1. This algorithm is optimal in some sense; we show that it cannot be improved straightforwardly by a similar improvement as used to obtain the algorithm.

Theorem 1 *Algorithm 4 solves the dominating set problem in $\mathcal{O}(1.4969^n)$ time while using only polynomial space.*

While for several classic combinatorial problems the first non-trivial exact algorithms date from many years ago, the first algorithms with running time faster than $O^*(2^n)$ for the Dominating Set problem are from 2004. In this year, there were three independent papers: one by Fomin, Kratsch and Woeginger [9], one by Randerath and Schiermeyer [18], and one by Grandoni [13]. Before our work, the fastest algorithm for Dominating Set is by Fomin, Grandoni, and Kratsch [5]: this algorithm uses $O(1.5260^n)$ time and polynomial space, or $O(1.5137^n)$ time and exponential space. See Table 1 for an overview of recent results on this problem.

This paper is organised in the following way. In Section 2, we start by giving some preliminaries and an introduction on measure and conquer. Then, we give a detailed step by step overview of how we design an algorithm for Dominating Set using measure and conquer in Section 3. In this section, we also prove Theorem 1. Hereafter, we give evidence for the fact that it is hard to improve our algorithm significantly using the same method in Section 4. We conclude in Section 5 by giving some remarks on our method and the associated numerical problems.

2 Preliminaries

Let $G = (V, E)$ be an n-vertex simple graph. For any $v \in V$, let $N(v) = \{u \in V \mid (v, u) \in E\}$ be the open neighbourhood of v, and let $N[v] = N(v) \cup \{v\}$ be the closed neighbourhood of v.

Authors	Polynomial space	Exponential space
F. Fomin, D. Kratsch, G. Woeginger [9]	$\mathcal{O}(1.9379^n)$	
I. Schiermeyer [18]	$\mathcal{O}(1.8899^n)$	
F. Grandoni [13]	$\mathcal{O}(1.9053^n)$	$\mathcal{O}(1.8021^n)$
F. Fomin, F. Grandoni, D. Kratsch [6]	$\mathcal{O}(1.5263^n)$	$\mathcal{O}(1.5137^n)$
J. M. M. van Rooij [24]*	$\mathcal{O}(1.5134^n)$	$\mathcal{O}(1.5086^n)$
J. M. M. van Rooij, H. L. Bodlaender [25]*	$\mathcal{O}(1.5134^n)$	$\mathcal{O}(1.5063^n)$
J. M. M. van Rooij, J. Nederlof, T. C. van Dijk [27]		$\mathcal{O}(1.5048^n)$
This paper*	$\mathcal{O}(1.4969^n)$	$\Longleftarrow$

*This paper is an improved and complete version of [24] and [25]. The difference between [24] and [25] is not the algorithm but the methodology and an improved memorisation proof. This methodology is also presented here.

Table 1: Known exact algorithms for dominating set and their running times.

A subset $D \subseteq V$ of the vertices is called a *dominating set* if every vertex $v \in V$ is either in D or adjacent to some vertex in D. I.e., a dominating set D is a set of vertices from G such that $\bigcup_{v \in D} N[v] = V$. In the Dominating Set problem, we are given a graph G and are asked to compute a dominating set in G of minimum cardinality.

This problem is long known to be $\mathcal{NP}$-complete [10], thus we cannot expect to find a polynomial time algorithm for this problem. Moreover, there exists no subexponential time algorithm for this problem unless the Exponential Time Hypothesis fails. A proof of this can be found in [9].

Exponential Time Hypothesis [14]: There exists an $\alpha > 0$ such that there is no algorithm solving 3-SAT on n variables in $\mathcal{O}(\alpha^n)$ time.
Consequence [9]: There exists an $\alpha > 0$ such that there is no algorithm solving Dominating Set on n vertices in $\mathcal{O}(\alpha^n)$ time.

Given a multiset sets $\mathcal{S}$, a *set cover* $\mathcal{C}$ of $\mathcal{S}$ is a subset $\mathcal{C} \subseteq \mathcal{S}$ such that every element in any of the sets in $\mathcal{S}$ occurs in some set in $\mathcal{C}$. I.e., a set cover $\mathcal{C}$ of $\mathcal{S}$ is a set of sets such that $\bigcup_{S \in \mathcal{C}} S = \bigcup_{S \in \mathcal{S}} S$. The universe $\mathcal{U}(\mathcal{S})$ of $\mathcal{S}$ is the set of all elements in any set in $\mathcal{S}$: $\mathcal{U}(\mathcal{S}) = \bigcup_{S \in \mathcal{S}} S$. In the Set Cover problem, we are given a multiset of sets $\mathcal{S}$ over a universe $\mathcal{U}$ and are asked to compute a set cover of minimum cardinality. We often denote a set cover instance by the tuple $(\mathcal{S},\mathcal{U})$ omitting the dependency of $\mathcal{U}$ on $\mathcal{S}$.

We can reduce the Dominating Set to the Set Cover by introducing a set for each vertex of G containing the closed neighbourhood of this vertex, i.e., $\mathcal{S} := \{N[v] \mid v \in V\}, \mathcal{U} := V$. Hence, we can solve an instance of Dominating Set on an n-vertex graph by a using a set cover algorithm running on an instance with n sets and a universe of size n. This idea was first introduced by Grandoni in [13] and is the basis of most exact exponential time algorithms for Dominating Set (all except the first two in Table 1).

Given a set cover instance $(\mathcal{S},\mathcal{U})$, let $|S|$ be the *size* or *cardinality* of a set $S \in \mathcal{S}$. Further, let $\mathcal{S}(e) = \{S \in \mathcal{S} \mid e \in S\}$ be the set of sets in $\mathcal{S}$ in which the element e occurs, and let the *frequency* $f(e)$ of an element $e \in \mathcal{U}$ be the number of sets in $\mathcal{S}$ in which this element occurs: $f(e) = |\mathcal{S}(e)|$.

A connected component component C of a graph G is an inclusion minimal, non-empty subset of the vertices of G such that, for every $v \in C$, all vertices that are reachable from v by a path in G are contained in C. Similarly, we define a connected component $\mathcal{C}$ of a set cover instance $(\mathcal{S},\mathcal{U})$ in the natural way: an inclusion minimal, non-empty subset $\mathcal{C} \subseteq \mathcal{S}$ for which all elements in the sets in $\mathcal{C}$ do not occur in $\mathcal{S} \setminus \mathcal{C}$.

In this paper, we use the $\mathcal{O}^*$ notation: $f(n)$ is $\mathcal{O}^*(g(n))$ if $f(n)$ is $\mathcal{O}(g(n)p(n))$ for some polynomial $p(n)$. This notation is often used in exponential time algorithms and suppresses all polynomial parts of the running time. We will omit the * whenever possible, especially when considering algorithms running in time $\mathcal{O}^*(\alpha^n)$ for some numerically obtained value α. In this case, we round α to $\alpha + \epsilon$ for some $\epsilon > 0$ and state that it runs in $\mathcal{O}((\alpha + \epsilon)^n)$: notice that $\alpha^n p(n) = \mathcal{O}((\alpha + \epsilon)^n)$ for any polynomial $p(n)$.

Additional notation we use is the following: let $[condition] = 1$ if the condition is true and $[condition] = 0$ otherwise.

2.1 Measure and Conquer

In the design of exact exponential time algorithms, the *branch and reduce* paradigm is one of the most prominently used approaches. A branch and reduce algorithm consists of a series of reduction rules, a series of branching rules, and a procedure to decide which branching rule to apply on a given instance. The reduction rules transform an instance with certain specific properties into an equivalent smaller instance in polynomial time. Such an algorithm first exhaustively applies its reduction rules. When the instance no longer satisfies any of the properties that fire a reduction rule, then the algorithm decides what branching rule to apply. The selected branching rule will then generate a series of smaller problem instances that are solved recursively. A solution to the current instance is then constructed from the solutions returned from the recursive calls. Some simple examples of such algorithms for 3-Sat and Independent Set can be found in [28].

A breakthrough in the analysis of branch and reduce algorithms is the *measure and conquer* technique by Fomin, Grandoni, and Kratsch [5]. This follows earlier work by Eppstein on analysing branch and reduce algorithms by multivariate recurrences [4]. In a measure and conquer analysis, a carefully chosen non-standard measure of instance or subproblem size is used. This is in contrast to classic analyses relying on simple, mostly integer measures representing the size of an instance, e.g., the number of vertices in a graph.

The first problem to which the measure and conquer technique has been applied is Dominating Set [5]. The authors present an algorithm for Set Cover. This algorithm is then applied to instances obtained by transforming an n-vertex Dominating Set instance into an equivalent Set Cover instance on n sets over a universe of size n. The algorithm is analysed using the following measure k, where v and w are weight functions giving an element e of frequency $f(e)$ measure $v(f(e))$ and a set S of size $|S|$ measure $w(|S|)$.

$$k = k(\mathcal{S},\mathcal{U}) = \sum_{e\in\mathcal{U}} v(f(e)) + \sum_{S\in\mathcal{S}} w(|S|) \qquad \text{with: } v, w : \mathbb{N} \to \mathbb{R}_+$$

The behaviour of the algorithm is analysed using this measure. For each branching rule, a series of recurrence relations is formulated corresponding to all possible situations the branching rule can be applied to.

Let $N(k)$ be the number of subproblems generated by branching on an instance of size k, and let R the set of cases considered by the algorithm covering all possible situations of branching. For any $r \in R$ we denoted by $\#(r)$ the number of subproblems generated when branching in case r, and by $\Delta k_{(r,i)}$ the measure by which an instance is reduced in subproblem i in case r. We thus obtain a series of recurrence relations of the form:

$$\forall r \in R \quad : \quad N(k) \leq \sum_{i=1}^{\#(r)} N\big(k - \Delta k_{(r,i)}\big)$$

We will often identify R with the set of corresponding recurrences.

A solution to this set of recurrence relations has the form α^k, for some $\alpha > 1$. This gives an upper bound on the running time of the branch and reduce algorithm expressed in the measure $k(\mathcal{S},\mathcal{U})$. Assume that $v_{\max} = \max_{n\in\mathbb{N}} v(n)$, $w_{\max} = \max_{n\in\mathbb{N}} w(n)$ are finite numbers. Then $\alpha^{(v_{\max}+w_{\max})n}$ is an upper bound on the running time of the algorithm for Dominating Set since for every input instance $k(\mathcal{S},\mathcal{U}) \leq (v_{\max} + w_{\max})n$ and thus $\alpha^k \leq \alpha^{(v_{\max}+w_{\max})n}$.

What remains is to choose ideal weight functions: weight functions that minimise the proven upper bound on the running time of the algorithm, i.e., weight functions that minimise $\alpha^{v_{\max}+w_{\max}}$. This forms a large numerical optimisation problem. Actually, under some assumptions on the weight functions (see Section 3 for more details), this gives a large but finite quasiconvex optimisation problem. Such a numerical problem can be solved by computer, see [4]. Details on the implementation of our solver can be found in Section 5.

In this way, Fomin, Grandoni and Kratsch [5] prove a running time of $\mathcal{O}(1.5263^n)$ on an algorithm that is almost identical to the $\mathcal{O}(1.9053^n)$ time algorithm in [13]. See Section 3 for examples of measure and conquer analyses.

3 Design by Measure and Conquer

In this paper, we take the measure and conquer technique one small step further. We will use measure and conquer not only to analyse branch and reduce algorithms, but also use it as a guiding tool to design these algorithms. This works in the following way.

Suppose we are given a non-standard measure of instance size using weight functions and some initial branching procedure, i.e. we are given the basic ingredient of a measure and conquer analysis and some trivial algorithm. Then, in an iterative process, we will formulate a series of branch and reduce algorithms. In this series, each algorithm is analysed by measure and conquer. Hereafter, the associated numerical optimisation problem is inspected, and the recurrence relations that

New reduction rule	Section	Running Time
Trivial algorithm	3.1	$\mathcal{O}^*(2^n)$
Unique element rule	3.2	$\mathcal{O}(1.7311^n)$
Stop when all sets have cardinality one	3.3	$\mathcal{O}(1.6411^n)$
Subset rule	3.4	$\mathcal{O}(1.5709^n)$
Stop when all sets have cardinality two	3.5	$\mathcal{O}(1.5169^n)$
Subsumption rule	3.6	$\mathcal{O}(1.5134^n)$
Counting rule	3.7	$\mathcal{O}(1.5055^n)$
Size two set with only frequency two elements rule	3.8	$\mathcal{O}(1.4969^n)$

Table 2: The iterative improvement of the algorithm

bound the current optimum (the best upper bound on the running time involving this measure) are identified. Each of these bounding recurrence relations corresponds to one or more worst case instances which can be identified easily. We can use these to formulate new reduction rules, or change the branching procedure, such that some of these worst case instances are handled more efficiently by the algorithm. The modification then gives a new, faster algorithm. This improvement series ends when we have sufficient evidence showing that improving the current worst cases is hard.

We will demonstrate this approach by example on the first problem measure and conquer has been applied to: Dominating Set. Doing so, we will design an algorithm for Set Cover instances obtained from the reduction from Dominating Set as described in Section 2, i.e. Set Cover instances consisting of n sets and n elements. This process starts with a trivial algorithm: Algorithm 1.

This section is organised as follows. In the next subsection, we set up the framework used to analyse our algorithms by measure and conquer and analyse a trivial algorithm. In each subsequent subsection, we treat the next algorithm from the series: we analyse its running time and apply an improvement step, as just described. In the last section (Section 3.8), we will prove Theorem 1. See Table 2 for a nice overview of how the trivial algorithm is improved by adding new reduction rules and analysing the corresponding algorithms.

3.1 A trivial algorithm

We start with a trivial algorithm for Set Cover: Algorithm 1. This algorithm simply selects the largest set from our instance and considers two subproblems: one in which we take the set in the set cover and one in which we discard it. In the branch where S is taken in the set cover, we remove all elements from S from the universe, and thus we remove for all $S' \in \mathcal{S} \setminus \{S\}$ all elements in S. In the other branch, we just remove S from $\mathcal{S}$. Then, the algorithm recursively solves both generated subproblems and returns the smallest set cover returned by the recursive calls. It stops when there is no set left to branch on; then, it checks whether the generated subproblem corresponds to a set cover.

We analyse this algorithm using measure and conquer. To this end, let $v, w : \mathbb{N} \to \mathbb{R}_+$ be

Algorithm 1 A trivial set cover algorithm.

Input: A set cover instance $(\mathcal{S},\mathcal{U})$
Output: A minimum set cover of $(\mathcal{S},\mathcal{U})$

MSC$(\mathcal{S},\mathcal{U})$:
1: **if** $\mathcal{S} = \emptyset$ **then return** $\emptyset$ if $\mathcal{U} = \emptyset$, or No otherwise
2: Let $S \in \mathcal{S}$ be a set of maximum cardinality
3: Recursively compute $\mathcal{C}_1 = \{S\} \cup \text{MSC}(\{S' \setminus S \mid S' \in \mathcal{S} \setminus \{S\}\}, \mathcal{U} \setminus S)$ and $\mathcal{C}_2 = \text{MSC}(\mathcal{S} \setminus \{S\}, \mathcal{U})$
4: **return** the smallest set cover from $\mathcal{C}_1$ and $\mathcal{C}_2$, or No if no set covers are returned

weight functions giving an element e of frequency $f(e)$ measure $v(f(e))$ and a set S of size $|S|$ measure $w(|S|)$, just like in Section 2.1. Furthermore, let $k = \sum_{e \in \mathcal{U}} v(f(e)) + \sum_{S \in \mathcal{S}} w(|S|)$ be our measure.

We start by defining the following very useful quantities:

$$\Delta v(i) = v(i) - v(i-1) \qquad \Delta w(i) = w(i) - w(i-1) \qquad \text{for } i \geq 1$$

We impose some constraints on these weights. We require the weights to be *monotone*, and set the weights of sets and elements that no longer play a role in the algorith to zero:

$$v(0) = 0 \quad w(0) = 0 \qquad \forall i \geq 1 \quad : \quad \Delta v(i) \geq 0 \qquad \Delta w(i) \geq 0$$

Intuitively, this corresponds to the idea that larger sets and higher frequency elements contribute more to the complexity of the problem than smaller sets and lower frequency elements, respectively. Furthermore, we impose the following non-restricting *steepness* inequalities, which we will discuss in a moment:

$$\forall i \geq 2 \quad : \quad \Delta w(i-1) \geq \Delta w(i)$$

Let r_i be the number of elements of frequency i in S. In the branch where S is taken in the set cover, the measure is reduced by $w(|S|)$ because we remove S, by $\sum_{i=1}^{\infty} r_i v(i)$ because we remove its elements, and by at least an additional $\min_{j \leq S}\{\Delta w(j)\} \sum_{i=1}^{\infty} r_i(i-1)$ because the removal of these elements reduces other sets in size. In the other branch, the measure is reduced by $w(|S|)$ because we remove S, and by an additional $\sum_{i=1}^{\infty} r_i \Delta v(i)$ because the elements in S have their frequencies reduced by one.

Let Δk_{take} and $\Delta k_{\text{discard}}$ be the reduction in the measure in the branch where we *take* S in the solution and where we *discard* S, respectively. Thus, we have derived the following lower bounds on the reductions of the measure:

$$\begin{aligned} \Delta k_{\text{take}} &\geq w(|S|) + \sum_{i=1}^{\infty} r_i v(i) + \Delta w(|S|) \sum_{i=1}^{\infty} r_i(i-1) \\ \Delta k_{\text{discard}} &\geq w(|S|) + \sum_{i=1}^{\infty} r_i \Delta v(i) \end{aligned}$$

Here, we used the steepness inequalities to replace the term $\min_{j \leq S}\{\Delta w(j)\}$ by $\Delta w(|S|)$. One can show that these steepness inequalities do not change the optimum solution of the numerical problem; they only simplify its formulation.

In this way, we find the corresponding set of recurrence relations. Let $N(k)$ be the number of subproblems generated by branching on an instance of size k.

$$\forall\, |S| \geq 1, \forall\, r_i : \sum_{i=1}^{\infty} r_i = |S| \quad : \quad N(k) \leq N(k - \Delta k_{\text{take}}) + N(k - \Delta k_{\text{discard}})$$

Finally, we compute the optimal weight functions minimising $\alpha^{v_{\max}+w_{\max}}$ where α^k is the solution to the above set of recurrence relations. To make this set of recurrence relations finite, we first set $v(i) = v_{\max}$ and $w(i) = w_{\max}$ for all $i \geq p$ for some point $p \in \mathbb{N}$. This results in the fact that all recurrences with $|S| > p+1$ are dominated by those with $|S| = p+1$. Moreover, we now only need to consider recurrences with $|S| = \sum_{i=1}^{p} r_p + r_{>p}$ where $r_{>p} = \sum_{i=p+1}^{\infty} r_i$ and $r_{>p}$ has the role of r_{p+1} in the above formulas. Notice that if all weights $v(i)$, $w(i)$ are multiplied by a positive real number, then a different value α will result from the set of recurrence relations. However, it is not hard to see that in this case the value $\alpha^{v_{\max}+w_{\max}}$ will remain the same. Hence, we can set $w_{\max} = 1$ without loss of generality[1]. We will omit these details concerning finiteness

[1]We could equally well have set $v_{\max} = 1$ giving the same upper bound on the running time for all algorithms in the improvement series except for this first algorithm. This is the case since in this analysis $v_{\max} = 0$, therefore the optimal weights cannot be multiplied by a positive real number such that $v_{\max} = 1$.

Algorithm 2 An improved version of Algorithm 1 by adding Reduction Rule 1.

Input: A set cover instance $(\mathcal{S},\mathcal{U})$
Output: A minimum set cover of $(\mathcal{S},\mathcal{U})$

MSC$(\mathcal{S},\mathcal{U})$:
1: **if** $\mathcal{S} = \emptyset$ **then return** $\emptyset$
2: Let $S \in \mathcal{S}$ be a set of maximum cardinality
3: **if** there exist an element $e \in \mathcal{U}$ of frequency one **then**
4: **return** $\{R\} \cup \text{MSC}(\{R' \setminus R \mid R' \in \mathcal{S} \setminus \{R\}\},\mathcal{U} \setminus R)$, where R is the set with $e \in R$
5: Recursively compute $\mathcal{C}_1 = \{S\} \cup \text{MSC}(\{S' \setminus S \mid S' \in \mathcal{S} \setminus \{S\}\},\mathcal{U} \setminus S)$ and $\mathcal{C}_2 = \text{MSC}(\mathcal{S} \setminus \{S\},\mathcal{U})$
6: **return** the smallest cover from $\mathcal{C}_1$ and $\mathcal{C}_2$

of the numerical problem in the analyses of the other algorithms in the improvement series in the coming subsections.

We solve the corresponding numerical program with continuous variables $v(1), v(2), \ldots, v(p)$, $w(1), w(2), \ldots, w(p)$ minimising $\alpha^{v_{\max}+w_{\max}}$ where α^k is the solution the the set of recurrence relations and obtain a solution of $N(k) \leq 2^k$ using the following weights:

i	1	2	3	4	5	6	7	> 7
$v(i)$	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
$w(i)$	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

This gives an upper bound on the running time of Algorithm 1 of $\mathcal{O}^*\big(2^{(0+1)n}\big) = \mathcal{O}^*(2^n)$. All recurrences considered contribute to the bounding (worst) case in this analysis.

It is no surprise that we prove a running time of $\mathcal{O}^*(2^n)$ on this trivial algorithm: the algorithm branches on all n sets considering two subproblems. We only formally used measure and conquer here, since the optimal weights correspond exactly to the standard measure: the total number of sets in the instance. The above analysis functions to set up our algorithm design process.

3.2 The first improvement step: unique elements

We will now give the first improvement step. To this end, we consider the bounding cases of the numerical problem associated with the previous analysis. In this case, all recurrences in the numerical problem form the set of bounding cases. At each improvement step in the design process of our algorithm, we will, as a rule of the thumb, consider the "*smallest*" worst case. With small we mean involving the smallest sets and lowest frequency elements. Thus, we consider here the worst case where $|S| = r_1 = 1$.

This case can be improved easily. Algorithm 1 considers many subsets of the input multiset $\mathcal{S}$ that will never result in a set cover. Namely, when considering a set with unique elements (elements of frequency one), Algorithm 1 still branches on this set. This, however, is not necessary since any set cover should include this set.

In the second algorithm in the series, we add a reduction rule dealing with unique elements improving, among others, the case $|S| = r_1 = 1$. This reduction rule takes any set containing a unique element in the computed set cover.

Reduction Rule 1

if there exist an element $e \in \mathcal{U}$ of frequency one **then**
return $\{R\} \cup \text{MSC}(\{R' \setminus R \mid R' \in \mathcal{S} \setminus \{R\}\},\mathcal{U} \setminus R)$, where R is the set with $e \in R$

We can change the formulation of the algorithm after adding this reduction rule to it: we no longer need to check whether every computed cover also covers all of $\mathcal{U}$. In this way, we obtain Algorithm 2.

Let us analyse Algorithm 2. To this end, we use the same measure as before; we only add some extra constraints. Since unique elements are directly removed from an instance, they do

not contribute to the (exponential) complexity of a problem instance; therefore, we set $v(1) = 0$. Notice that this results in $\Delta v(2) = v(2)$.

Next, we derive new recurrence relations for Algorithm 2. Let Algorithm 2 branch on a set S containing r_i elements of frequency i. Due to Reduction Rule 1, we now only have to consider cases with $|S| = \sum_{i=2}^{\infty} r_i$, i.e., with $r_1 = 0$. In the branch where Algorithm 2 takes S in the set cover, nothing changes to the reduction in measure. But, in the branch where it discards S, additional reductions in the measure are obtained when S contains unique elements. If $r_2 > 0$, at least one extra set is taken in the set cover. Because of the steepness inequalities, the worst case is the case where the extra set consists exactly of all frequency two elements in S, hence this set is of size r_2. This gives us an additional reduction in the measure of $[r_2 > 0]w(r_2)$.

Altogether, this gives the following set of recurrences:

$$\begin{aligned}
\forall\, |S| \geq 1, \forall\, r_i : \sum_{i=2}^{\infty} r_i = |S| \quad &: \quad N(k) \leq N(k - \Delta k_{\text{take}}) + N(k - \Delta k_{\text{discard}}) \\
\Delta k_{\text{take}} \quad &\geq \quad w(|S|) + \sum_{i=2}^{\infty} r_i v(i) + \Delta w(|S|) \sum_{i=2}^{\infty} r_i (i-1) \\
\Delta k_{\text{discard}} \quad &\geq \quad w(|S|) + \sum_{i=2}^{\infty} r_i \Delta v(i) + [r_2 > 0] w(r_2)
\end{aligned}$$

We solve the associated numerical optimisation problem minimising $\alpha^{v_{\max}+w_{\max}}$ where α^k is the solution the the set of recurrence relations and obtain a solution of $N(k) \leq 1.58143^k$ using the following set of weights:

i	1	2	3	4	5	6	7	> 7
$v(i)$	0.000000	0.011524	0.162465	0.192542	0.197195	0.197195	0.197195	0.197195
$w(i)$	0.750412	0.908050	0.968115	0.992112	1.000000	1.000000	1.000000	1.000000

This leads to an upper bound on the running time of Algorithm 2 of $\mathcal{O}\big(1.58143^{(0.197195+1)n}\big) = \mathcal{O}(1.73101^n)$. The bounding cases of the numerical problem are:

$$\begin{array}{lllllll}
|S| = r_2 = 1 & |S| = r_3 = 2 & |S| = r_4 = 2 & |S| = r_4 = 3 & |S| = r_5 = 4 & |S| = r_5 = 5 \\
|S| = r_6 = 5 & |S| = r_6 = 6 & |S| = r_{>6} = 6
\end{array}$$

This concludes the first improvement step. By applying this improvement step, we have obtained our first simple algorithm with a non-trivial upper bound on the running time.

3.3 Improvement step two: sets of cardinality one

The next step will be to inspect the case $|S| = r_2 = 1$ more closely. This case corresponds to branching on a set S containing only one element of frequency two.

We improve this case by the simple observation that we can stop branching when all sets have size one. Namely, any solution consists of a series of singleton sets: one for each remaining element. This gives us the following reduction rule.

Reduction Rule 2

Let $S \in \mathcal{S}$ be a set of maximum cardinality
if $|S| \leq 1$ **then**
  return $\{\{e\} \mid e \in \mathcal{U}\}$

We add Reduction Rule 2 to Algorithm 2 to obtain our new algorithm. For this algorithm, we derive the following set of recurrence relations; these are exactly the ones used to analyse Algorithm 2 that correspond to branching on a set of size at least two:

$$\forall\, |S| \geq 2, \forall\, r_i : \sum_{i=2}^{\infty} r_i = |S| \quad : \quad N(k) \leq N(k - \Delta k_{\text{take}}) + N(k - \Delta k_{\text{discard}})$$

$$\begin{aligned}\Delta k_{\text{take}} &\geq w(|S|) + \sum_{i=2}^{\infty} r_i v(i) + \Delta w(|S|) \sum_{i=2}^{\infty} r_i (i-1) \\ \Delta k_{\text{discard}} &\geq w(|S|) + \sum_{i=2}^{\infty} r_i \Delta v(i) + [r_2 > 0] w(r_2)\end{aligned}$$

We solve the associated numerical optimisation problem and obtain a solution of $N(k) \leq 1.42604^k$ on the recurrence relations using the following set of weights:

i	1	2	3	4	5	6	7	> 7
$v(i)$	0.000000	0.161668	0.325452	0.387900	0.395390	0.395408	0.395408	0.395408
$w(i)$	0.407320	0.814639	0.931101	0.981843	0.998998	1.000000	1.000000	1.000000

This leads to an upper bound on the running time of the algorithm of $\mathcal{O}\big(1.42604^{(0.395408+1)n}\big) = \mathcal{O}(1.64107^n)$. The bounding cases of the numerical problem are:

$$\begin{array}{llllll} |S| = r_2 = 2 & |S| = r_3 = 2 & |S| = r_3 = 3 & |S| = r_4 = 3 & |S| = r_5 = 3 & |S| = r_5 = 4 \\ |S| = r_5 = 5 & |S| = r_6 = 5 & |S| = r_6 = 6 & & & \end{array}$$

3.4 Improvement step three: subsets

Consider the new "smallest" worst case: $|S| = r_2 = 2$. For this case, the previous section uses the following values in the corresponding recurrences: $\Delta k_{\text{take}} = w(2) + 2v(2) + 2\Delta w(2)$ and $\Delta k_{\text{discard}} = w(2) + 2\Delta v(2) + w(2)$. Notice that the instance tight to these values consists of a set S containing two frequency two elements which second occurrence is in a set R which is a copy of S (notice that the final computed weights satisfy $2\Delta w(2) = w(2)$ and that by definition $v(2) = \Delta v(2)$).

The observation we use to improve the algorithm of Section 3.3 is that we never take two identical sets in any minimum set cover. This can be generalised to sets Q and R that are not identical, but where R is a subset of Q: whenever we take R in the set cover, we could equally well have taken Q, moreover, any set cover containing both R and Q can be replaced by a smaller cover by removing R. This leads to the following reduction rule:

Reduction Rule 3

if there exist sets $Q, R \in \mathcal{S}$ such that $R \subseteq Q$ **then**
  return MSC($\mathcal{S} \setminus \{R\}$,$\mathcal{U}$)

Reduction Rules 1 and 3 together remove all sets of size one: either the element in a singleton set S has frequency one and is removed by Reduction Rule 1, or it has higher frequency and thus S is a subset of another set. Consequently, Reduction Rule 2 becomes obsolete after adding Reduction Rule 3 to the algorithm.

In the analysis of this new algorithm, we set $w(1) = 0$ because sets of size one are now removed. To avoid problems with the steepness inequalities, we now only impose them for $i \geq 3$:

$$\forall i \geq 3 \quad : \quad \Delta w(i-1) \geq \Delta w(i)$$

Observe what happens when our new algorithm branches on a set S containing r_i elements of frequency i. In the branch where we take S in the set cover, we still reduce the measure by $w(|S|) + \sum_{i=2}^{\infty} r_i v(i)$ for removing S and its elements. Remind that $\Delta w(|S|)$ lower bounds the reduction in measure for reducing the size of a set by one because of the steepness inequalities. Even though $w(1) = 0$ in our new analysis, we can still use $\Delta w(|S|) \sum_{i=2}^{\infty} r_i (i-1)$ to lower bound the additional reduction in the measure due to the fact that no other set R containing elements from S can be a subset of S by Reduction Rule 3; therefore, we reduce R at most $|R| - 1$ times in size, and the steepness inequalities still make sure that $\sum_{i=2}^{|R|} \Delta w(i) \geq (|R| - 1)\Delta w(|S|)$.

In the branch where we discard S, we still reduce the measure by $w(|S|) + \sum_{i=2}^{\infty} r_i \Delta v(i)$ for removing S and reducing the frequencies of its elements. Observe what happens to frequency two elements. If $r_2 = 1$, we take at least one more set in the set cover and remove at least one more element since this set cannot be a subset of S. For every additional frequency two element in S, the size of the set that is taken in the set cover increases by one in the worst case, unless $r_2 = |S|$. In the last case, all elements cannot be in the same other set because this would make the set larger than S which cannot be the case by the branching rule. Since $w(2) = \Delta w(2)$, this leads to an additional reduction in the measure of at least $[r_2 > 0](v(2) + \sum_{i=1}^{\min(r_2,|S|-1)} \Delta w(i+1))$. Finally, if $r_2 = |S|$, then we remove at least one more set reducing the measure by at least $w(2)$ more.

The leads to the following set of recurrence relations:

$$\forall\, |S| \geq 2, \forall\, r_i : \sum_{i=2}^{\infty} r_i = |S| \quad : \quad N(k) \leq N(k - \Delta k_{\text{take}}) + N(k - \Delta k_{\text{discard}})$$

$$\begin{aligned}
\Delta k_{\text{take}} &\geq w(|S|) + \sum_{i=2}^{\infty} r_i v(i) + \Delta w(|S|) \sum_{i=2}^{\infty} r_i (i-1) \\
\Delta k_{\text{discard}} &\geq w(|S|) + \sum_{i=2}^{\infty} r_i \Delta v(i) + [r_2 > 0] \left(v(2) + \sum_{i=1}^{\min(r_2,|S|-1)} \Delta w(i+1) \right) + [r_2 = |S|] w(2)
\end{aligned}$$

We solve the associated numerical optimisation problem and obtain a solution of $N(k) \leq 1.37787^k$ on the recurrence relations using the following set of weights:

i	1	2	3	4	5	6	7	> 7
$v(i)$	0.000000	0.089132	0.304202	0.377794	0.402590	0.408971	0.408971	0.408971
$w(i)$	0.000000	0.646647	0.853436	0.939970	0.979276	0.995872	1.000000	1.000000

This leads to an upper bound on the running time of the algorithm of $\mathcal{O}\big(1.37787^{(0.408971+1)n}\big) = \mathcal{O}(1.57087^n)$. The bounding cases of the numerical problem are:

$$\begin{array}{llllll}
|S| = r_2 = 2 & |S| = r_3 = 2 & |S| = r_3 = 3 & |S| = r_4 = 3 & |S| = r_4 = 4 & |S| = r_5 = 4 \\
|S| = r_5 = 5 & |S| = r_6 = 5 & |S| = r_6 = 6 & |S| = r_7 = 6 & |S| = r_7 = 7 &
\end{array}$$

We notice that this analysis is not tight in the following way. The quantities we use for Δk_{take} and $\Delta k_{\text{discard}}$ can both correspond to a real instance the algorithm branches on, that is, there are real instances to which the reduction in the measure is bounded tightly. However, for some considered cases, there is no real instance in which the reduction in the measure is tight to both Δk_{take} and $\Delta k_{\text{discard}}$. This even happens on the bounding case $|S| = r_2 = 2$.

We could give a better analysis of the current algorithm. However, this requires a different set up with either more automatically generated subcases or some manual case analysis. We feel that it is better to ignore this fact for the moment and continue formulating new reduction rules for instances tight to any of the individual values of Δk_{take} and $\Delta k_{\text{discard}}$. We will give such a case analysis for our final algorithm in Section 3.8.

3.5 Improvement step four: all sets have cardinality at most two

The case $|S| = r_2 = 2$ corresponds to a set S containing two frequency two elements whose second occurrences are in different sets. In this case, all sets have cardinality at most two since our algorithms only branch on sets of maximum cardinality. We observe that this case can be solved in polynomial time by computing a maximum matching.

Notice that if we construct a set cover in this situation, then we can initially pick some sets that each cover two elements until only sets containing one thus far uncovered element remain. The maximum number of sets that cover two elements per set are used in a minimum set cover.

Algorithm 3 The algorithm of Fomin, Grandoni, and Kratsch [5].

Input: A set cover instance $(\mathcal{S},\mathcal{U})$
Output: A minimum set cover of $(\mathcal{S},\mathcal{U})$

MSC$(\mathcal{S},\mathcal{U})$:
if $\mathcal{S} = \emptyset$ **then return** $\emptyset$
Let $S \in \mathcal{S}$ be a set of maximum cardinality
if there exist an element $e \in \mathcal{U}$ of frequency one **then**
    return $\{R\} \cup \text{MSC}(\{R' \setminus R \mid R' \in \mathcal{S} \setminus \{R\}\},\mathcal{U} \setminus R)$, where R is the set with $e \in R$
else if there exist sets $Q, R \in \mathcal{S}$ such that $R \subseteq Q$ **then**
    return MSC$(\mathcal{S} \setminus \{R\},\mathcal{U})$
else if $|S| \leq 2$ **then**
    return a minimum set cover computed in polynomial time by using maximum matching
Recursively compute $\mathcal{C}_1 = \{S\} \cup \text{MSC}(\{S' \setminus S \mid S' \in \mathcal{S} \setminus \{S\}\},\mathcal{U} \setminus S)$ and $\mathcal{C}_2 = \text{MSC}(\mathcal{S} \setminus \{S\},\mathcal{U})$
return the smallest cover from $\mathcal{C}_1$ and $\mathcal{C}_2$

We can find such a maximum set of disjoint size two sets by computing a maximum matching in the following graph $G = (V, E)$: introduce a vertex for every element $e \in \mathcal{U}$, and an edge (e_1, e_2) for every set of size two $\{e_1, e_2\} \in \mathcal{S}$. This maximum matching M can be computed in polynomial time [3]. Given M, we can construct a minimum set cover by taking the sets corresponding to the edges in M and add an additional set for each vertex that is not incident to an edge in M.

This leads to the following reduction rule.

Reduction Rule 4

Let $S \in \mathcal{S}$ be a set of maximum cardinality
if $|S| \leq 2$ **then**
    return a minimum set cover computed in polynomial time by using maximum matching

If we add this reduction rule to the algorithm of Section 3.4, we obtain the algorithm of Fomin, Grandoni, and Kratsch [5]: Algorithm 3.

The numerical problem associated with the computation of an upper bound on the running time of Algorithm 3 is the same as the numerical problem associated with the algorithm of Section 3.4, except for the fact that we only consider branching on sets S with $|S| \geq 3$.

$$\forall\, |S| \geq 3, \forall\, r_i : \sum_{i=2}^{\infty} r_i = |S| \quad : \quad N(k) \leq N(k - \Delta k_{\text{take}}) + N(k - \Delta k_{\text{discard}})$$

$$\Delta k_{\text{take}} \geq w(|S|) + \sum_{i=2}^{\infty} r_i v(i) + \Delta w(|S|) \sum_{i=2}^{\infty} r_i (i-1)$$

$$\Delta k_{\text{discard}} \geq w(|S|) + \sum_{i=2}^{\infty} r_i \Delta v(i) + [r_2 > 0] \left(v(2) + \sum_{i=1}^{\min(r_2, |S|-1)} \Delta w(i+1) \right) + [r_2 = |S|] w(2)$$

We again solve the associated numerical optimisation problem. We obtain a solution of $N(k) \leq 1.28505^k$ on the recurrence relations using the following set of weights:

i	1	2	3	4	5	6	7	> 7
$v(i)$	0.000000	0.261245	0.526942	0.620173	0.652549	0.661244	0.661244	0.661244
$w(i)$	0.000000	0.370314	0.740627	0.892283	0.961123	0.991053	1.000000	1.000000

This leads to an upper bound on the running time of Algorithm 3 of $\mathcal{O}\big(1.28505^{(0.661244+1)n}\big) = \mathcal{O}(1.51685^n)$. The bounding cases of the numerical problem are:

$$|S| = r_2 = 3 \quad |S| = r_3 = 3 \quad |S| = r_4 = 3 \quad |S| = r_4 = 4 \quad |S| = r_5 = 4 \quad |S| = r_5 = 5$$
$$|S| = r_6 = 5 \quad |S| = r_6 = 6 \quad |S| = r_7 = 6 \quad |S| = r_7 = 7$$

We note that the analysis in [5] gives an upper bound on the running time of Algorithm 3 of $\mathcal{O}(1.5263^n)$. The difference with our better upper bound comes from the fact that we allow $v_{\max}$ to be variable in the associated numerical optimisation problem, while $v_{\max} = 1$ in [5][2].

3.6 Improvement step five: subsumption

Again, we consider the "smallest" bounding case: $|S| = r_2 = 3$. The reduction in the measure in the formula for $\Delta k_{\text{discard}}$ in the previous section is tight to the following real instance. We have a set S of cardinality three containing three elements of frequency two, and these three frequency two elements together have their second occurrences in two sets, each containing the same extra frequency two element such that they are not subsets of S. In other words, we have $S = \{e_1, e_2, e_3\}$ existing next to $\{e_1, e_2, e_4\}, \{e_3, e_4\}$ with $f(e_1) = f(e_2) = f(e_3) = f(e_4) = 2$.

We can reduce this case by introducing the notion of subsumption. We say that an element e_1 is *subsumed* by e_2 if $\mathcal{S}(e_1) \subseteq \mathcal{S}(e_2)$, i.e., if the set of sets containing e_1 is a subset of the set of sets containing e_2. Notice that in this case, any set of sets that covers e_1 will always cover e_2 also, therefore we can safely remove e_2 from the instance.

This leads to the following reduction rule.

Reduction Rule 5

if there exist two elements e_1 and e_2 such that $\mathcal{S}(e_1) \subseteq \mathcal{S}(e_2)$ **then**
  return MSC($\{R \setminus \{e_2\} \mid R \in \mathcal{S}\}$,$\mathcal{U} \setminus \{e_2\}$)

We will now analyse our first algorithm that improves upon the algorithm by Fomin, Grandoni and Kratsch: Algorithm 3 augmented with Reduction Rule 5. In the branch where S is taken in the set cover, nothing changes. In the branch where we discard S, all sets containing the other occurrence of a frequency two element from S are taken in the set cover. These are at least r_2 sets since no two frequency two elements can have both occurrences in the same two sets by Reduction Rule 5. Hence, we reduce the measure by at least $r_2 w(2)$. The measure is further reduced because these removed sets contain elements that are removed also, moreover, this removal reduces the cardinality of other sets. We lower bound the total reduction in the measure by only considering the cases where $r_2 \in \{1, 2, 3\}$ in more detail. If $r_2 \in \{1, 2\}$, then at least one extra element is removed which we lower bound by $v(2)$; this is tight if $r_2 = 2$. If $r_2 = 3$, then the three sets can either contain the same extra element which gives a reduction of $v(3)$, or these contain more extra elements giving a reduction of at least $2v(2)$ and an additional $\Delta w(|S|)$ because these exist in other sets also.

This leads to the following set of recurrence relations:

$$\forall\, |S| \geq 3, \forall\, r_i : \sum_{i=2}^{\infty} r_i = |S| \quad : \quad N(k) \leq N(k - \Delta k_{\text{take}}) + N(k - \Delta k_{\text{discard}})$$

$$\Delta k_{\text{take}} \geq w(|S|) + \sum_{i=2}^{\infty} r_i v(i) + \Delta w(|S|) \sum_{i=2}^{\infty} r_i (i-1)$$

$$\begin{aligned} \Delta k_{\text{discard}} \geq\ & w(|S|) + \sum_{i=2}^{\infty} r_i \Delta v(i) + r_2 w(2) \\ & + [r_2 \in \{1,2\}] v(2) + [r_2 = 3] \min(v(3), 2v(2) + \Delta w(|S|)) \end{aligned}$$

We obtain a solution of $N(k) \leq 1.28886^k$ on the recurrence relations using the following set of weights:

i	1	2	3	4	5	6	7	> 7
$v(i)$	0.000000	0.218849	0.492455	0.589295	0.623401	0.632777	0.632777	0.632777
$w(i)$	0.000000	0.367292	0.734584	0.886729	0.957092	0.988945	1.000000	1.000000

[2]The constraint $v_{\max} = 1$ was also included in an earlier version of this paper [25] and in [24]. This is the reason for the better running times in this paper compared to the running times for the same algorithms in [24, 25].

This leads to an upper bound on the running time of the algorithm of $\mathcal{O}\big(1.28886^{(0.632777+1)n}\big) = \mathcal{O}(1.51335^n)$. The bounding cases of the numerical problem are:

$$|S| = r_2 = 3 \quad |S| = r_3 = 3 \quad |S| = r_4 = 3 \quad |S| = r_4 = 4 \quad |S| = r_5 = 4 \quad |S| = r_5 = 5$$
$$|S| = r_6 = 5 \quad |S| = r_6 = 6 \quad |S| = r_7 = 6 \quad |S| = r_7 = 7$$

3.7 Improvement step six: counting arguments

The new "smallest" worst case of our algorithm is $|S| = r_2 = 3$. We look at an instance in which the formula for $\Delta k_{\text{discard}}$ of the previous section is tight. This corresponds to a set S containing three elements of frequency two that all have their second occurrence in a different set of size two, and, since the optimal weights in the analysis of Section 3.6 satisfy $v(3) < v(2) + \Delta w(|S|)$, these sets all contain the same second element which is of frequency three. Thus, we have the following situation: $S = \{e_1, e_2, e_3\}$ existing next to $\{e_1, e_4\}, \{e_2, e_4\}, \{e_3, e_4\}$ with $f(e_1) = f(e_2) = f(e_3) = 2$ and $f(e_4) = 3$.

We notice that we do not have to branch on this set S. Namely, if we take S in the set cover, we cover three elements using one set, while if we discard S and thus take all other sets containing e_1, e_2 and e_3, then we cover four elements using three sets. We can cover the same four elements with only two sets if we take S and any and any of the three sets containing e_4. Therefore, we can safely take S in the set cover without branching.

This counting argument can be generalised. For any set R with r_2 elements of frequency two, we let q be the number of elements in the sets containing a frequency two element from R that are not in R themselves, i.e., $q = \left|\left(\bigcup_{e \in R, f(e)=2, Q \in \mathcal{S}(e)} Q\right) \setminus R\right|$. In this case, we cover $|R|$ elements using one set if we take R in the set cover, and we cover $q + |R|$ elements using r_2 sets if we discard R. Thus, if $q < r_2$, taking R is always at least as good as discarding R since then we use $r_2 - 1$ sets less while also covering $q < r_2$, i.e., $q \leq r_2 - 1$, less elements; we can always cover these elements by picking one additional set per element.

This leads to the following reduction rule:

Reduction Rule 6
if there exist a set R with $\left|\left(\bigcup_{e \in R, f(e)=2, Q \in \mathcal{S}(e)} Q\right) \setminus R\right| < |\{e \in R \mid f(e) = 2\}|$ **then**
return $\{R\} \cup \text{MSC}(\{R' \setminus R \mid R' \in \mathcal{S} \setminus \{R\}\}, \mathcal{U} \setminus R)$

We add Reduction Rule 6 to the algorithm of Section 3.6 and analyse its behaviour. If S is discarded, we now know that at least r_2 sets are removed due to Reduction Rule 5 and that, due to Reduction Rule 6, these sets contain at least r_2 elements that are removed also. This gives a reduction of at least $r_2(v(2) + w(2))$. Furthermore, additional sets are reduced in cardinality because of the removal of these elements. The reduction in the measure can not be bounded by $r_2 \Delta w(|S|)$ because we can remove a set R completely and the steepness inequalities do not guarantee that this reduction in measure is bounded by $|R| \Delta w(|S|)$ since $w(1) = 0$. These inequalities do guarantee that this reduction is lower bounded by $\min\big(r_2 \Delta w(|S|), \lfloor \frac{r_2}{2} \rfloor\, w(2) + (r_2 \bmod 2) \Delta w(|S|)\big)$.

This leads to the following set of recurrence relations:

$$\forall\, |S| \geq 3, \forall\, r_i : \sum_{i=2}^{\infty} r_i = |S| \quad : \quad N(k) \leq N(k - \Delta k_{\text{take}}) + N(k - \Delta k_{\text{discard}})$$

$$\Delta k_{\text{take}} \geq w(|S|) + \sum_{i=2}^{\infty} r_i v(i) + \Delta w(|S|) \sum_{i=2}^{\infty} r_i (i-1)$$

$$\Delta k_{\text{discard}} \geq w(|S|) + \sum_{i=2}^{\infty} r_i \Delta v(i) + r_2(v(2) + w(2)) + \min\left(r_2 \Delta w(|S|), \left\lfloor \frac{r_2}{2} \right\rfloor w(2) + (r_2 \bmod 2) \Delta w(|S|)\right)$$

We obtain a solution of $N(k) \leq 1.29001^k$ on the recurrence relations using the following set of weights:

i	1	2	3	4	5	6	7	> 7
$v(i)$	0.000000	0.127612	0.432499	0.544653	0.587649	0.602649	0.606354	0.606354
$w(i)$	0.000000	0.364485	0.728970	0.881959	0.953804	0.987224	0.999820	1.000000

This leads to an upper bound on the running time of the algorithm of $\mathcal{O}\big(1.29001^{(0.606354+1)n}\big) = \mathcal{O}(1.50541^n)$. The bounding cases of the numerical problem are:

$$|S| = r_2 = 3 \quad |S| = r_3 = 3 \quad |S| = r_4 = 3 \quad |S| = r_4 = 4 \quad |S| = r_5 = 4 \quad |S| = r_5 = 5$$
$$|S| = r_6 = 5 \quad |S| = r_6 = 6 \quad |S| = r_7 = 6 \quad |S| = r_7 = 7 \quad |S| = r_8 = 7 \quad |S| = r_8 = 8$$

3.8 The final improvement: frequency two elements in sets of size two

We now present the final improvement step. This will prove Theorem 1 and explicitly give the associated algorithm: Algorithm 4. We again look at the case $|S| = r_2 = 3$ and closely inspect instances that are tight to the formula for $\Delta k_{\text{discard}}$ of the previous section. Currently, this corresponds to a set $S = \{e_1, e_2, e_3\}$ containing three elements of frequency two whose second occurrences form one of the following three situations:

1. $\{e_1, e_4\}, \{e_2, e_5\}, \{e_3, e_6\}$ with all elements of frequency two existing next to $\{e_4, e_5, e_6\}$.
2. $\{e_1, e_4\}, \{e_2, e_5\}, \{e_3, e_6\}$ with all elements of frequency two existing next to $\{e_4, e_5\}$ and $\{e_6, e_7, e_8\}$.
3. $\{e_1, e_4\}, \{e_2, e_5\}, \{e_3, e_4, e_6\}$ with all elements of frequency two existing next to $\{e_5, e_6\}$.

Notice that the optimal weights in the analysis of Section 3.7 satisfy $2w(2) = w(3)$, i.e., $w(2) = \Delta w(3)$. Hence, in all three cases, the measure is reduced by the same amount in the branch where S is discarded: $w(3) + 3v(2)$ for removing S plus an additional $3v(2) + 5w(2)$ due to the reduction rules.

We add the following reduction rule that removes any set of cardinality two containing two frequency two elements to our algorithm and obtain Algorithm 4.

Reduction Rule 7

if there exits a set $R \in \mathcal{S}$ of cardinality two $R = \{e_1, e_2\}$ with $f(e_1) = f(e_2) = 2$ **then**
  Let $\mathcal{S}(e_i) = \{R, R_i\}$ $(i = 1, 2)$, $Q = (R_1 \cup R_2) \setminus R$, $\mathcal{C} = \text{MSC}((\mathcal{S} \setminus \{R, R_1, R_2\}) \cup \{Q\}, \mathcal{U} \setminus R)$
  if $Q \in \mathcal{C}$ **then**
    return $(\mathcal{C} \setminus \{Q\}) \cup \{R_1, R_2\}$
  else
    return $\mathcal{C} \cup \{R\}$

If there exists a set R of cardinality two containing two frequency two elements e_1, e_2, such that e_i occurs in R and R_i, then the reduction rule transforms this instance into an instance where R, R_1 and R_2 have been replaced by the set $Q = (R_1 \cup R_2) \setminus R$.

We will now argue that Reduction Rule 7 is correct. Notice that there exist a minimum set cover of $(\mathcal{S}, \mathcal{U})$ that either contains R, or contains both R_1 and R_2. This is so because if we take only one set from R, R_1 and R_2, then this must be R since we must cover e_1 and e_2; if we take two, then it is of no use to take R since the other two cover more elements; and, if we take all three, then the set cover is not minimal. The rule postpones the choice between the first two possibilities, taking Q in the minimum set cover of the transformed problem if both R_1 and R_2 are in a minimum set cover, or taking no set in the minimum set cover of the transformed problem is R is in a minimum set cover. This works because the transformation preserves the fact that the difference in the number of sets we take in the set cover between both possibilities is one. Hence, Reduction Rule 7 is correct.

Algorithm 4 Our final algorithm for the set cover modelling of dominating set.

Input: A set cover instance $(\mathcal{S},\mathcal{U})$
Output: A minimum set cover of $(\mathcal{S},\mathcal{U})$

MSC$(\mathcal{S},\mathcal{U})$:
if $\mathcal{S} = \emptyset$ **then return** $\emptyset$
Let $S \in \mathcal{S}$ be a set of maximum cardinality
if there exist an element $e \in \mathcal{U}$ of frequency one **then**
    return $\{R\} \cup \text{MSC}(\{R' \setminus R \mid R' \in \mathcal{S} \setminus \{R\}\},\mathcal{U} \setminus R)$, where R is the set with $e \in R$
else if there exist sets $Q, R \in \mathcal{S}$ such that $R \subseteq Q$ **then**
    return $\text{MSC}(\mathcal{S} \setminus \{R\},\mathcal{U})$
else if there exist two elements e_1 and e_2 such that $\mathcal{S}(e_1) \subseteq \mathcal{S}(e_2)$ **then**
    return $\text{MSC}(\{R \setminus \{e_2\} \mid R \in \mathcal{S}\},\mathcal{U} \setminus \{e_2\})$
else if there exist a set R with $\left|\left(\bigcup_{e \in R, f(e)=2, Q \in \mathcal{S}(e)} Q\right) \setminus R\right| < |\{e \in R \mid f(e) = 2\}|$ **then**
    return $\{R\} \cup \text{MSC}(\{R' \setminus R \mid R' \in \mathcal{S} \setminus \{R\}\},\mathcal{U} \setminus R)$
else if there exits a set $R \in \mathcal{S}$ of cardinality two $R = \{e_1, e_2\}$ with $f(e_1) = f(e_2) = 2$ **then**
    Let $\mathcal{S}(e_i) = \{R, R_i\}$ $(i = 1, 2)$, $Q = (R_1 \cup R_2) \setminus R$, $\mathcal{C} = \text{MSC}((\mathcal{S} \setminus \{R, R_1, R_2\}) \cup \{Q\},\mathcal{U} \setminus R)$
    if $Q \in \mathcal{C}$ **then**
        return $(\mathcal{C} \setminus \{Q\}) \cup \{R_1, R_2\}$
    else
        return $\mathcal{C} \cup \{R\}$
else if $|S| \leq 2$ **then**
    return a minimum set cover computed in polynomial time by using maximum matching
Recursively compute $\mathcal{C}_1 = \{S\} \cup \text{MSC}(\{S' \setminus S \mid S' \in \mathcal{S} \setminus \{S\}\},\mathcal{U} \setminus S)$ and $\mathcal{C}_2 = \text{MSC}(\mathcal{S} \setminus \{S\},\mathcal{U})$
return the smallest cover from $\mathcal{C}_1$ and $\mathcal{C}_2$

Before analysing Algorithm 4, we notice that we need additional constraints on the weights for the analysis to be valid due to Reduction Rule 7. Namely, this reduction rule should never be allowed to increase the measure of an instance. For the moment, we forget that the elements e_1, e_2 and S are removed, and demand that the measure is not increased by merging the two sets S_1 and S_2. To this end, we impose the following additional constraints:

$$\forall i, j \geq 2 \quad : \quad w(i) + w(j) \geq w(i + j - 2)$$

For a proper analysis, we further subdivide our cases by differentiating between different kinds of elements of frequency two depending on the kind of set their second occurrence is in. Let r_{f3} and $r_{f\geq 4}$ be the number of elements of frequency two whose second occurrence is in a set of size two with a *frequency three* element and with a *higher frequency* element, respectively. And, let r_{s3} and $r_{s\geq 4}$ be the number of elements of frequency two whose second occurrence is in a set of *cardinality three* and a set of *greater cardinality*, respectively.

We consider branching on a set S with r_i elements of frequency i, for all $|S| \geq 3$, all $|S| = \sum_{i=2}^{\infty} r_i$, and all $r_2 = r_{f3} + r_{f\geq 4} + r_{s3} + r_{s\geq 4}$. Because S is of maximal cardinality, we only consider subcases with $r_{s4} > 0$ if $|S| \geq 4$. For these cases, we will first derive lower bounds on the reductions in the measure. In order to keep the bounding cases of the associated numerical problem corresponding to real instances the algorithm can branch on, we perform a subcase analysis dealing with the more subtle details later. Let $\mathcal{R}$ be the set of sets containing a frequency two element from S, excluding S itself: the algorithm takes the sets in $\mathcal{R}$ in the set cover if we discard S.

In the branch where S is taken in the solution, we again start with a reduction in the measure of $w(|S|) + \sum_{i=2}^{\infty} r_i v(i) + \Delta w(|S|) \sum_{i=2}^{\infty} r_i(i - 1)$ due the the removal of S, its elements, and the reduction in size of the sets containing elements from S. Additionally, for each element of frequency two occurring in a size two set, the measure is not reduced by $\Delta w(|S|)$ for reducing this set in size, but by $w(2)$ since the set will be removed by Reduction Rule 3. Similarly, for each element of frequency two occurring in a size three set, the reduction is $\Delta w(3)$ instead of $\Delta w(|S|)$ if $|S| \geq 4$. Together, this gives an extra reduction of $(r_{f3} + r_{f\geq 4})(w(2) - \Delta w(|S|)) + r_{s3}(\Delta w(|3|) - \Delta w(|S|))$.

Finally, if S contains elements of frequency two whose second occurrence is in a set of size two containing an elements of frequency three, then these frequency three elements are reduced in frequency because these sets have become singleton subsets; these elements can also be removed completely if they occur in multiple such sets. This leads to an additional reduction of the measure of at least $[r_{f3} > 0] \min(v(3), r_{f3}\Delta v(3))$.

In the branch where S is discarded, we reduce the measure by $w(|S|) + \sum_{i=2}^{\infty} r_i \Delta v(i)$ because of removing S. The sets in $\mathcal{R}$ are taken in the set cover; this reduces the measure by at least $r_2(v(2)+w(2))$ because we have at least r_2 sets and they together contain at least r_2 other elements by Reduction Rule 6. Additionally, the r_{s3} sets of size three and the $r_{s\geq 4}$ sets of size at least four reduce the measure by at least an extra $r_{s3}\Delta w(3) + r_{s\geq 4}(w(4) - w(2))$. And, if these sets are of size two but contain elements of frequency at least three, we can add $[r_{f3} > 0]\Delta v(3) + [r_{f\geq 4} > 0]$ $(v(4) - v(2))$; notice that we cannot add $r_{f3}\Delta v(3)$ as one element may be in multiple sets in $\mathcal{R}$.

Furthermore, other sets are reduced in size because of the removal of all elements in $\bigcup \mathcal{R}$. Let $q_{\vec{r}}$ be the number element occurrences outside S and $\mathcal{R}$ that are removed after taking all sets in $\mathcal{R}$ in the set cover in the subcase corresponding to $\vec{r}$, i.e., for this subcase, this is the number of times a set is reduced in cardinality by one. By using the steepness inequalities in the same way as in the previous section, we reduce the measure by at least an additional: $\min\left(q_{\vec{r}}\Delta w(|S|), \left\lfloor \frac{q_{\vec{r}}}{2} \right\rfloor w(2) + (q_{\vec{r}} \bmod 2)\Delta w(|S|)\right)$.

We now give a lower bound on $q_{\vec{r}}$. There are at least r_2 additional elements that are removed; these are all of frequency at least two, hence at least r_2 additional element occurrences are removed. These occurrences do not all need to be outside of $\mathcal{R}$: for every set in $\mathcal{R}$ of size three, there is one empty slot that could be filled by an occurrence of these elements. Similarly, for every set in $\mathcal{R}$ of size at least four, there are $|S| - 2$ empty slots that can be filled with these elements. Furthermore, if the sets in $\mathcal{R}$ contain elements of frequency three or at least four, then the number of removed element occurrences increases by 1 or 2, respectively. Altogether, we find that $q_{\vec{r}} \geq \max\left(0, r_2 + [r_{f3} > 0] + 2[r_{f\geq 4} > 0] - r_{s3} - (|S| - 2)r_{s\geq 4}\right)$.

Finally, we split some recurrences based on Reduction Rule 7. If $r_{s3} > 0$, and a corresponding set of size three only contains elements of frequency two, then this set is removed when taking S in the set cover: this can either be done by Reduction Rule 7, or by the old Reduction Rules 1 and 3. We split the recurrence relations with $r_{s3} > 0$ into two separate cases. We identify these case by introducing yet another identifier: r_{rule7}. One subcase has $r_{\text{rule7}} = True$, and one has $r_{\text{rule7}} = False$. If $r_{\text{rule7}} = True$, we add an additional $2v(2) + w(2)$ to the formula of Δk_{take} representing the additional set and elements that are removed. Notice that we can do this because we did not take these two frequency two elements an this cardinality two set into account in the new restrictions on the weights we imposed before starting the above analysis.

This leads to the following set of recurrence relations:

$$\forall\, |S| \geq 3, \forall\, r_i : \sum_{i=2}^{\infty} r_i = |S|, \forall\, r_2 = r_{f3} + r_{f\geq 4} + r_{s3} + r_{s\geq 4}, \forall\, r_{\text{rule7}} \in \{True, False\}$$

with $r_{s\geq 4} = 0$ if $|S| = 3$, and $r_{\text{rule7}} = False$ if $r_{s3} = 0$:

$$N(k) \leq N(k - \Delta k_{\text{take}}) + N(k - \Delta k_{\text{discard}})$$

$$\begin{aligned}
\Delta k_{\text{take}} \quad \geq \quad & w(|S|) + \sum_{i=2}^{\infty} r_i v(i) + \Delta w(|S|) \sum_{i=2}^{\infty} r_i(i-1) + (r_{f3} + r_{f\geq 4})(w(2) - \Delta w(|S|)) \\
& + r_{s3}(\Delta w(|3|) - \Delta w(|S|)) + [r_{f3} > 0] \min(v(3), r_{f3}\Delta v(3)) \\
& + [r_{\text{rule7}}](2v(2) + w(2)) \\
\Delta k_{\text{discard}} \quad \geq \quad & w(|S|) + \sum_{i=2}^{\infty} r_i \Delta v(i) + r_2(v(2) + w(2)) + r_{s3}\Delta w(3) + r_{s\geq 4}(w(4) - w(2)) \\
& + [r_{f3} > 0]\Delta v(3) + [r_{f\geq 4} > 0](v(4) - v(2)) \\
& + \min\left(q_{\vec{r}}\Delta w(|S|), \left\lfloor \frac{q_{\vec{r}}}{2} \right\rfloor w(2) + (q_{\vec{r}} \bmod 2)\Delta w(|S|)\right)
\end{aligned}$$

Here, we let $q_{\vec{r}}$ be tight the above lower bound.

Unfortunately, this does not yet complete the description of the new numerical problem. For some specific cases, we will prove that we can increase the corresponding Δk_{take} and $\Delta k_{\text{discard}}$. We do this not only to improve the proven running time of the algorithm, but also to make sure that the recurrences representing bounding cases of this numerical problem represent actual instances that can be branched on. In other words, the bounding cases should not be based on non-tight lower bounds on the reductions of the measure. For one subcase, we not only increase Δk_{take} and $\Delta k_{\text{discard}}$, but also split the corresponding recurrence into two separate recurrence relations.

Consider the following cases where $S = \{e_1, e_2, e_3\}$ or $S = \{e_1, e_2, e_3, e_4\}$:

1. $|S| = 3, r_2 = r_{s3} = 3$ ($q_{\vec{r}} = 0$).

   (a) $r_{\text{rule7}} = False$.

   Remind that this means that none of the sets in $\mathcal{R}$ can consist solely of frequency two elements, i.e., $\mathcal{R}$ contains one element of frequency at least three existing in all three sets in $\mathcal{R}$, or $\mathcal{R}$ contains at least two elements of frequency at least three. In both cases, we consider what happens in the branch where the algorithm discards S and takes the sets in $\mathcal{R}$ in the set cover.

   In the first case, no frequency two element may occur in two sets in $\mathcal{R}$ by Reduction Rule 5. Thus, we need at least one more element than the three counted in the above analyses: we reduce the measure by at least an additional $\Delta v(3) + v(2)$. Furthermore, by taking the sets in $\mathcal{R}$ in the set cover, there are at least three element occurrences outside S and $\mathcal{R}$ removed. No two of these elements may exist together in a set of size two by Reduction Rule 7. Hence, this gives an at least additional $3\Delta w(3)$.

   In the second case, the measure is reduced by at least an additional $2\Delta v(3)$. Moreover, sum of the frequencies of the elements in $(\bigcup \mathcal{R}) \setminus S$ is at least eight, while there are only six open slots in $\mathcal{R}$, i.e., there are at least two element outside of S and $\mathcal{R}$ removed. If there are exactly two extra element occurrences removed which occur together in a size two set, then this set is a subset of a set in $\mathcal{R}$ because the two frequency three elements must be in some set together: this is not possible due to Reduction Rule 3. In any other case, the reduction in measure due to the removal of these element occurrences is at least $2\Delta w(3)$.

   Altogether, we add the minimum of both quantities to $\Delta k_{\text{discard}}$ by setting $\Delta k_{\text{discard}} \mathrel{+}= \min\left(\Delta v(3) + v(2) + 3\Delta w(3),\ 2\Delta v(3) + 2\Delta w(3)\right)$.

   (b) $r_{\text{rule7}} = True$.

   Notice that subcase (a) dominates every case where there exist one element of frequency at least three and one extra element in $\mathcal{R}$, or where there exist at least two elements of frequency at least three in $\mathcal{R}$. Furthermore, we can disregard the case with one element of frequency at least three and two frequency two elements because of Reduction Rule 5. Hence, we can restrict ourselves to the case where $(\bigcup \mathcal{R}) \setminus S$ consists of at least three frequency two elements.

   Consider the branch where the algorithm takes S in the set cover. If there are only three frequency two elements, i.e, $\mathcal{R} = \{\{e_1, e_4, e_5\}, \{e_2, e_4, e_6\}, \{e_3, e_5, e_6\}\}$, then Reduction Rules 7 and 1 remove all sets and elements from $\mathcal{R}$. This gives an additional reduction of $v(2) + 2w(2)$ to Δk_{take} besides the $2v(2) + w(2)$ we counted already because $r_{\text{rule7}} = True$. If there are four or more frequency two elements in $(\bigcup \mathcal{R}) \setminus S$, then Reduction Rule 7 can be applied at least twice reducing the measure by the same amount. We set $\Delta k_{\text{take}} \mathrel{+}= v(2) + 2w(2)$.

2. $|S| = 3, r_2 = r_{s3} = 2, r_3 = 1$ ($q_{\vec{r}} = 0$).

   (a) $r_{\text{rule7}} = False$.

$\mathcal{R}$ contains one element of frequency at least three existing in both sets in $\mathcal{R}$, or $\mathcal{R}$ contains at least two elements of frequency at least three. We consider the branch where S is discarded and the sets in $\mathcal{R}$ are taken in the set cover.

In the first case, $\mathcal{R} = \{\{e_1, e_4, e_5\}, \{e_2, e_4, e_6\}\}$ with $f(e_4) \geq 3$ and all other elements of frequency two. We have an extra reduction in the measure of $\Delta v(3) + v(2)$ because e_4 has higher frequency and we have an extra element. Additionally, we look at the number of element occurrences outside of S and $\mathcal{R}$ that are removed: there are at least three of these. Hence, we get an additional reduction of $\min(3\Delta w(3), w(2) + \Delta w(3))$ equivalent to setting $q_{\vec{r}} = 3$.

In the second case, we reduce the measure by at least an additional $2\Delta v(3)$ because of the higher frequency elements. Moreover, there are at least two element occurrences outside S and $\mathcal{R}$ removed: this gives an additional reduction of $\min(2\Delta w(3), w(2))$ equivalent to setting $q_{\vec{r}} = 2$.

We add the minimum of both quantities to $\Delta k_{\text{discard}}$ by setting $\Delta k_{\text{discard}}$+=$\min(\Delta v(3)+ v(2) + \min(3\Delta w(3), w(2) + \Delta w(3)),\ 2\Delta v(3) + \min(2\Delta w(3), w(2)))$.

(b) $r_{\text{rule7}} = True$.

Subcase (a) dominates every case with elements of frequency at least three in $\mathcal{R}$. Thus, we may assume that $\mathcal{R}$ contains only elements of frequency two, and there are at least three of them because of Reduction Rule 5. In the branch where S is discarded and all sets in $\mathcal{R}$ are taken in the set cover, one extra element of frequency two and at least two element occurrences outside of S and $\mathcal{R}$ are removed. Altogether, we add their contribution to the measure to $\Delta k_{\text{discard}}$ by setting $\Delta k_{\text{discard}}$+= $v(2) + w(2)$.

3. $|S| = 3, r_2 = r_{s3} = 1, r_3 = 2, r_{\text{rule7}} = False$ $(q_{\vec{r}} = 0)$.

   In the branch where S is discarded, there must be two elements in the unique set in $\mathcal{R}$ and one must be of frequency at least three. Hence, we set $\Delta k_{\text{discard}}$+= $\Delta v(3) + v(2)$.

4. $|S| = 3, r_2 = r_{s3} = 2, r_4 = 1, r_{\text{rule7}} = False$ $(q_{\vec{r}} = 0)$.

   Analogous to the case where $|S| = 3, r_2 = r_{s3} = 2, r_3 = 1, r_{\text{rule7}} = False$ (case 2a), we set $\Delta k_{\text{discard}}$+=$\min(\Delta v(3) + v(2) + \min(3\Delta w(3), w(2) + \Delta w(3)),\ 2\Delta v(3) + \min(2\Delta w(3), w(2)))$.

5. $|S| = 3, r_2 = 3, r_{f3} = 1, r_{s3} = 2, r_{\text{rule7}} = False$ $(q_{\vec{r}} = 2)$.

   Since $r_{f3} = 1$, we have a set R_1 of size two in $\mathcal{R}$ with an element of frequency three. If this element is also in the other two sets in $\mathcal{R}$, then technically we are in this case because none of the r_{e3} sets of size three consist solely of frequency two elements. However, after taking S in the solution, R_1 disappears because it has become a singleton singleton set and Reduction Rule 7 fires on the remaining sets in $\mathcal{R}$. The recurrence relation for the corresponding case with $r_{\text{rule7}} = True$ correctly represents this case. Therefore, we only have to consider the case where there is another higher frequency element that prevents the r_{s3} sets of size three from having only frequency two elements.

   For this case, consider the branch where S is discarded and the sets in $\mathcal{R}$ are taken in the set cover. Since there is another element of frequency at least three, we reduce the measure by at least an additional $\Delta v(3)$. Furthermore, there are at least eight element occurrences of the elements in $(\bigcup \mathcal{R}) \setminus S$, while $\mathcal{R}$ has only five available slots. Thus, $q_{\vec{r}}$ should be 3 instead of 2. This allows us to set $\Delta k_{\text{discard}}$+= $\Delta v(3) + \Delta w(3)$.

6. $|S| = 3, r_2 = 3, r_{f\geq 4} = 1, r_{s3} = 2, r_{\text{rule7}} = False$ $(q_{\vec{r}} = 3)$.

   In contrast to the previous case, the element of frequency at least four in the size two set in $\mathcal{R}$ can cause all other sets in $\mathcal{R}$ not to consist of frequency two elements only. We split this case into two separate recurrences: one where this element has frequency four, and one where it has higher frequency.

In the first case, we can set $\Delta k_{\text{take}} \mathrel{+}= \Delta v(4)$, because after taking S in the set cover, this element exists in a singleton set and hence its frequency is reduced by one. Notice that we could not bound this reduction before since the frequency of the element was unbounded.

In the second case, we remove at least one extra element occurrence outside $\mathcal{R}$ and S. If all other elements in $\mathcal{R}$ have frequency two, this extra element occurrence cannot be in a set with another removed element occurrence by Reduction Rule 5, and we can add $\Delta w(3)$ to the reduction (not needing to increasing $q_{\vec{r}}$ by one). If some other element in $\mathcal{R}$ has frequency at least three, then we remove at least two extra element occurrence outside $\mathcal{R}$ and S. Taking the worst case of both, we can set $\Delta k_{\text{discard}} \mathrel{+}= \Delta w(3)$.

7. $|S| = 3, r_{f3} = 2, r_{s3} = 1, r_{\text{rule7}} = False$ $(q_{\vec{r}} = 3)$.

   There are two possible situations: either there are two different elements of frequency three in the r_{f3} sets of size two, or both sets contain the same element of frequency three. In the latter case, this element cannot also be in the third set in $\mathcal{R}$ because this would trigger Reduction Rule 6 on S. Since $r_{\text{rule7}} = False$, there must be another element of frequency at least three in this third set in $\mathcal{R}$. In both cases, we have an extra element of frequency at least three; hence, we can set $\Delta k_{\text{discard}} \mathrel{+}= \Delta v(3)$.

8. $|S| = 3, r_{f\geq 4} = 2, r_{s3} = 1$ $(q_{\vec{r}} = 4)$.

   (a) $r_{\text{rule7}} = False$.

   Similar to the previous case, there are two possible situations: either there are two different elements of frequency at least four in the $r_{f\geq 4}$ sets of size two, or both sets contain the same element of frequency at least four.

   In the first case, we have an additional reduction of at least $v(4) - v(2)$ due to this element. Moreover, at least ten element occurrences of the elements in $(\bigcup \mathcal{R}) \setminus S$ are removed while there are only 4 slots available in $\mathcal{R}$ and currently $q_{\vec{r}} = 4$. This means we reduce the measure by an additional $\min(2\Delta w(3), w(2))$.

   In the second case, the element of frequency at least four cannot be in the third set in $\mathcal{R}$ because then Reduction Rule 6 applies to S; hence, there must be an element of frequency at least three in the third set in $\mathcal{R}$. The gives an additional $\Delta v(3)$, while counting the number of removed element occurrences of the elements in $(\bigcup \mathcal{R}) \setminus S$ gives us an additional $\Delta w(3)$.

   Altogether, we add the minimum of both cases to $\Delta k_{\text{discard}}$ by setting $\Delta k_{\text{discard}} \mathrel{+}= \min(\Delta v(3) + \Delta w(3), v(4) - v(2) + \min(2\Delta w(3), w(2)))$.

   (b) $r_{\text{rule7}} = True$.

   Consider the branch where we take S in the solution. After removing the elements e_1, e_2 and e_3, no sets outside of $\mathcal{R}$ are reduced in size. This is true because the two size two sets in $\mathcal{R}$ that are removed contain an element of frequency at least four: after removing their occurrences in $\mathcal{R}$, they still have frequency at least two. Moreover, $r_i = 0$ for all $i \geq 3$, so no other element occurrences outside of $\mathcal{R}$ are removed. As a result, two sets of size two or three are merged by Reduction Rule 7. Notice that we already counted the reduction due to removing a set of size two and its elements, but not the reduction due to the fact that two other sets are replaced by one larger one. This gives an additional reduction of the measure; we set $\Delta k_{\text{take}} \mathrel{+}= \min(2w(3) - w(4), w(3) + w(2) - w(3), 2w(2) - w(2))$.

9. $|S| = 3, r_{f3} = 3, r_{\text{rule7}} = False$ $(q_{\vec{r}} = 4)$.

   Consider the branch where S is discarded and the sets in $\mathcal{R}$ are taken in the set cover. Since all sets in $\mathcal{R}$ are of size two, there are three different frequency three elements in $\mathcal{R}$ by Reduction Rule 6 instead of the one we count now. These elements reduce the measure by an additional $2\Delta v(3)$. Moreover, we removed at least nine element occurrences of the elements in $(\bigcup \mathcal{R}) \setminus S$ from which only three can be in $\mathcal{R}$ and $q_{\vec{r}}$ counts only four: at least two more are removed reducing the measure by an additional $\min(2\Delta w(3), w(2))$. Altogether, we set $\Delta k_{\text{discard}} \mathrel{+}= 2\Delta v(3) + \min(2\Delta w(3), w(2))$.

10. $|S| = 3, r_{f\geq 4} = 3, r_{\text{rule7}} = False$ $(q_{\vec{r}} = 5)$.

    This case is similar to the above: there must be at least two more elements of frequency at least four reducing the measure by $2(v(4) - v(2))$ in the branch where S is discarded. And, by a counting removed element occurrences of the elements in $(\bigcup \mathcal{R}) \setminus S$, we should increase $q_{\vec{r}}$ by four. Hence, we set $\Delta k_{\text{discard}} \mathrel{+}= 2(v(4) - v(2)) + \min(4\Delta w(3), 2w(2))$.

11. $|S| = 4, r_{s\geq 4} = 4, r_{\text{rule7}} = False$ $(q_{\vec{r}} = 0)$.

    In the branch where S is discarded and the sets in $\mathcal{R}$ are taken in the solution, we only count a measure of $4v(2)$ for the removed elements in $(\bigcup \mathcal{R}) \setminus S$. There must be at least four elements in $(\bigcup \mathcal{R}) \setminus S$ by Reduction Rule 6 and there are twelve slots to fill. This can either be done by six elements of frequency two, or by using higher frequency elements. Hence, we set $\Delta k_{\text{discard}} \mathrel{+}= \min(2v(2), \Delta v(3))$.

We solve the numerical problem associated with the described set of recurrence relations and obtain a solution of $N(k) \leq 1.28935^k$ using the following set of weights:

i	1	2	3	4	5	6	7	> 7
$v(i)$	0.000000	0.011179	0.379475	0.526084	0.573797	0.591112	0.595723	0.595723
$w(i)$	0.000000	0.353012	0.706023	0.866888	0.943951	0.981278	0.997062	1.000000

This leads to an upper bound on the running time of the algorithm of $\mathcal{O}\big(1.28759^{(0.595723+1)n}\big) = \mathcal{O}(1.49684^n)$. The bounding cases of the numerical problem are:

$$|S| = r_3 = 3 \quad |S| = r_4 = 3 \quad |S| = 4, r_2 = r_{e\geq 4} = 4 \quad |S| = r_5 = 4 \quad |S| = r_5 = 5$$
$$|S| = r_6 = 5 \quad |S| = r_6 = 6 \quad |S| = r_7 = 6 \quad |S| = r_7 = 7 \quad |S| = r_8 = 7 \quad |S| = r_8 = 8$$

This proves Theorem 1.

4 Evidence That Further Improvement Is Hard

We stop with the design by measure and conquer process after obtaining Algorithm 4. We do so because it is hard to further improve the algorithm significantly by the same method.

We acknowledge that it is possible to improve on the new "smallest" worst case $|S| = 4, r_2 = r_{e\geq 4} = 4$. We tried to do so by introducing the following reduction rule that deals with each connected component separately.

Reduction Rule 8

if $\mathcal{S}$ contains multiple connected components $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_l$ **then**
return $\bigcup_{i=1}^{l} \text{MSC}(\mathcal{C}_i, \bigcup_{S\in\mathcal{C}_i} S)$

However, introducing this new reduction rule will help only marginally in obtaining better bounds with our current type of analysis. In particular, we analyse the possible gain (with our current analysis method) in some optimistic scenario where no frequency two elements appear in a worst case. Even then, the bound obtained is no better than $\mathcal{O}(1.4952^n)$. In addition, these frequency two elements cannot be expected to be reduced altogether with reduction rules; see Proposition 1 below.

We tried to close the gap between the $\mathcal{O}(1.4969^n)$ algorithm of Section 3.8 and this lower bound. This required us to perform an extensive case analysis to show the additional effects of this rule, however, we gave up after having to consider too many subcases.

First of all, it seems to be necessary to consider elements of frequency two: we do not seem to be able to either remove them completely by reduction rules, nor to solve the instance in subexponential time if all elements have frequency two. This is for the following reason.

Proposition 1 *There is no polynomial time algorithm that solves minimum set cover where all elements have frequency at most two and all sets have cardinality at most three, unless $\mathcal{P} = \mathcal{NP}$. Moreover, under the exponential time hypothesis, there is no such algorithm running in subexponential time.*

Proof: Consider an instance $G = (V, E)$ of the minimum vertex cover problem with maximum degree three. From this instance, we build an equivalent minimum set cover instance: for each edge introduce an element of frequency two, and for each vertex introduce a set containing the elements representing the edges it is incident to. Notice that the sets have cardinality at most three. It is easy to see that a minimum set cover of the constructed instance corresponds to a minimum vertex cover in G.

Therefore, a polynomial time algorithm as in the statement of the proposition would solve minimum vertex cover on graphs of maximum degree three in polynomial time, which is impossible unless $\mathcal{P} = \mathcal{NP}$. And, such an algorithm running in subexponential time would solve minimum vertex cover restricted to graphs of maximum degree three in subexponential time which is impossible unless the exponential time hypothesis fails [16]. □

Our second piece of evidence that improving our algorithm further by our method is hard is the following. Consider what we have been doing in the last few improvement steps. In each of these steps, the "smallest" worst case involved frequency two elements, and we looked for new reduction rules dealing with these elements more efficiently. We know by Proposition 1 that we cannot completely remove these frequency two elements, but what if we could formulate powerful enough reduction rules such that they never occur in any of the worst cases of the algorithm. We may not have such an algorithm, but we can analyse such an algorithm in the same way as we did in Section 3.

Using our measure, the best upper bound on the running time we can prove for such an algorithm corresponds to the solution of the numerical problem associated to the following set of recurrence relations:

$$\forall\, |S| \geq 3, \forall\, r_i : \sum_{i=2}^{\infty} r_i = |S| \quad : \quad N(k) \leq N(k - \Delta k_{\text{take}}) + N(k - \Delta k_{\text{discard}})$$

$$\Delta k_{\text{take}} \geq w(|S|) + \sum_{i=2}^{\infty} r_i v(i) + \Delta w(|S|) \sum_{i=2}^{\infty} r_i (i-1) + [r_2 > 0]\infty$$

$$\Delta k_{\text{discard}} \geq w(|S|) + \sum_{i=2}^{\infty} r_i \Delta v(i) + r_2(v(2) + w(2)) + [r_2 > 0]\infty$$

Where $[r_2 > 0]\infty = 0$ if $r_2 = 0$. These terms exist to make Δk_{take} and $\Delta k_{\text{discard}}$ large enough not to appear as a bounding case whenever $r_2 > 0$.

We obtain a solution of $N(k) \leq 1.26853^k$ on the recurrence relations using the following set of weights:

i	1	2	3	4	5	6	7	> 7
$v(i)$	0.000000	0.000000	0.426641	0.607747	0.671526	0.687122	0.690966	0.690966
$w(i)$	0.000000	0.353283	0.706566	0.866101	0.948043	0.985899	1.000000	1.000000

This leads to an upper bound on the running time of such a hypothetical algorithm proven by these methods of $\mathcal{O}\big(1.26853^{(0.690966+1)n}\big) = \mathcal{O}(1.49513^n)$. The bounding cases of the numerical problem are:

$$|S| = r_3 = 3 \quad |S| = r_4 = 3 \quad |S| = r_5 = 4 \quad |S| = r_6 = 4 \quad |S| = r_6 = 5 \quad |S| = r_6 = 6$$
$$|S| = r_7 = 6 \quad |S| = r_7 = 7 \quad |S| = r_{>7} = 7 \quad |S| = r_{>7} = 8$$

Of course, this is not a lower bound on the complexity of dominating set. This is a lower bound on the upper bounds on running times we can get by using measure and conquer in this way on algorithms using the set cover formulation of dominating set that only considers to branch on a single set. It shows that it is not worth the effort to try and improve Algorithm 4 with the purpose of finding an algorithm with a lower upper bound on its running time by doing an extensive case analysis since it can only improve the running time marginally.

5 Conclusion

We have shown that measure and conquer can not only be used for the analysis of exact exponential time algorithms, but also as a guiding tool in the design of these algorithms. As an example problem, we applied our method to the set cover modelling of DOMINATING SET: the same problem used to introduce measure and conquer. In this way, we have obtained an algorithm for DOMINATING SET running in time $\mathcal{O}(1.4969^n)$ and using polynomial space.

In another paper [26], we applied the same approach in a slightly different manner to a series of edge domination problems: EDGE DOMINATING SET, MINIMUM MAXIMAL MATCHING, MATRIX DOMINATING SET, and weighted version of these problems. In this setting, we did not iteratively introduce new reduction rules, but iteratively changed the branching rule on specific local structures to improve the worst case behaviour of the algorithm. In this paper, we did not feel the need to work with modifications to the branching rule. This is because every worst case we tried to improve could be improved by introducing a new reduction rule. Also, branching on a set S of maximal cardinality has two advantages: a set of maximal cardinality has maximal impact on the instance, the maximum number of other sets and elements are removed or reduced, and it allows us to use the cardinality of S as an upper bound on the cardinality of any other set in the instance. However, some of our arguments in Section 4 do not apply to algorithms using modified branching strategies.

Practical issues related to solving the numerical problems. We notice that in order to apply our method to any problem, it is vital to have an efficient solver for numerical problems associated with a measure and conquer analysis: having to wait more than an hour for an analysis to be completed and for the new bounding cases to be known is not an option. Such a solver can be obtained in many ways, for example by means of random search. However, random search does not converge to the optimum quickly enough to handle larger problems involving more than ten variables and hundreds of thousands of subcases, much like the problem associated with Algorithm 4. For a general treatment of such problems, see [4]. Before going into more details on our implementation, we note that an alternative way to solve these problems was found by Gaspers and Sorkin in [12] by rewriting the measure in such a way that a convex program is obtained.

We used a variation of Eppstein's smooth quasiconvex programming algorithm [4]. We modified this algorithm in two important ways. Let $\vec{w} \in D$ be the finite dimensional vector of weights in the feasible polytope $D = \{\vec{x} \in \mathbb{R}^d \mid \mathbf{A}\vec{x} \leq \vec{b}\}$ defined by the constraints $\mathbf{A}\vec{w} \leq \vec{b}$. We denote row i of $\mathbf{A}$ by $\vec{a}_i$, and the ith component of $\vec{b}$ by b_i. Let $\alpha_r(w)$ be the function mapping the weight vector $\vec{w}$ to the solution to recurrence $r \in R$, where R is the set of recurrence relations; we are minimising $\alpha(\vec{r}) = \max_{r \in R} \alpha_r(\vec{w})$ over all $\vec{w} \in D$. For some initial tolerance, and until some desired tolerance level is obtained, it repeats the following steps.

1. For all $\alpha_r(\vec{w})$ that are within a tolerance interval from $\alpha(\vec{w})$, compute the gradient $\nabla\alpha_r(\vec{w})$.
2. Compute the set of constraints with $\vec{a}_i \cdot \vec{w}$ within a fraction of the tolerance interval from b_i.
3. Compute the vector $\vec{v}$ such that γ is maximal and satisfies the following set of constraints: $\vec{v} \cdot \nabla\alpha_r(\vec{w}) \geq \gamma$ for all $\nabla\alpha_r(\vec{w})$ in the computed set; $\vec{a}_i \cdot \vec{v} \leq 0$ for all constraints i in the computed set; $\vec{v}$ lies in the unit ball in $\mathbb{R}^d$.
4. If no such vector exists, lower the tolerance, and restart from step 1.
5. Minimise $\alpha(\vec{w}+\epsilon\vec{v})$ by a line search using standard one dimensional optimisation techniques.
6. Repeat from step 1 using $\vec{w} := \vec{w} + \epsilon\vec{v}$ and lower the tolerance if ϵ is very small.

Compared to Eppstein, we introduced the following two modifications. First of all, we included the feasible set of weights D in the algorithm instead of incorporating them in the functions (setting $\alpha(\vec{w}) = \infty$ if $w \notin D$). This allows the algorithm to smoothly follow these boundaries without getting stuck.

Second of all, we do not just compute a direction in which we can find a better point, but we compute the unit length direction $\vec{v}$ with the property that its minimum inner product with a gradient $\alpha_r(\vec{w})$ of a function which value is within a tolerance interval form $\alpha(\vec{w})$ is maximised. This gives a direction in which a line search is most likely to give a better result. And, if the result is not satisfactory, then this must be because the tolerance level is still to high, in which case, we lower it in step 6. We note that the gradients can be found by means of implicit differation, and that we can compute the vector $\vec{v}$ by solving a small quadratically constrained program for which many solvers are available. Both modifications greatly improve the algorithm in practical situations.

Acknowledgements

We would like to thank Thomas C. van Dijk for his continued assistance in the C++ programming related to solving the numerical problems used in this paper.

References

[1] N. Bourgeois, B. Escoffier, V. Th. Paschos, J. M. M. van Rooij. Fast algorithms for max independent set in graphs of small average degree. Technical report. Cahier du Lamsade no 277, Lamsade, Université Paris-Dauphine, 2008.

[2] M. Davis and H. Putnam. A computing procedure for quantification theory. *Journal of the ACM*, 7:201–215, 1960.

[3] J. Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17:445–467, 1965.

[4] D. Eppstein. Quasiconvex analysis of multivariate recurrence equations for backtracking algorithms. *ACM Transactions on Algorithms*, 2:492–509, 2006.

[5] F. V. Fomin, F. Grandoni, and D. Kratsch. A measure & conquer approach for the analysis of exact algorithms. *Journal of the ACM*, 56, 2009.

[6] F. V. Fomin, F. Grandoni, and D. Kratsch. Some new techniques in design and analysis of exact (exponential) algorithms. *Bulletin of the EATCS*, 87:47–77, 2005.

[7] F. V. Fomin, F. Grandoni, and D. Kratsch. Solving connected dominating set faster than 2^n. In *Proceedings 26th International Comference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2006*, pages 152–163. Springer Verlag, Lecture Notes in Computer Science, vol. 4337, 2006.

[8] F. V. Fomin, F. Grandoni, A. Pyatkin, and A. A. Stepanov. Combinatorial bounds via measure and conquer: bounding minimal dominating sets and applications. *ACM Transactions on Algorithms*, 5, 2008.

[9] F. V. Fomin, D. Kratsch, and G. J. Woeginger. Exact (exponential) algorithms for the dominating set problem. In *Proceedings 30th International Workshop on Graph-Theoretic Concepts in Computer Science, WG 2004*, pages 245–256. Springer Verlag, Lecture Notes in Computer Science, vol. 3353, 2004.

[10] M. R. Garey and D. S. Johnson. *Computers and Intractability, A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, New York, 1979.

[11] S. Gaspers and M. Liedloff. A branch-and-reduce algorithm for finding a minimum independent dominating set in graphs. In *Proceedings 32nd International Workshop on Graph-Theoretic Concepts in Computer Science, WG 2006*, pages 78–89. Springer Verlag, Lecture Notes in Computer Science, vol. 4271, 2006.

[12] S. Gaspers and G. B. Sorkin. A universally fastest algorithm for max 2-sat, max 2-csp, and everything in between. In *Proceedings 20th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2009*, pages 606-615. SIAM, 2009.

[13] F. Grandoni. A note on the complexity of minimum dominating set. *Journal of Discrete Algorithms*, 4:209–214, 2006.

[14] R. Impagliazzo, R. Paturi, and F. Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63:512-530, 2001.

[15] K. Iwama. Worst-case upper bounds for k-sat. *Bulletin of the EATCS*, 82:61-71, 2004.

[16] D. S. Johnson and M. Szegedy. What are the least tractable instances of max independent set? In *Proceedings of the 10th ACM-SIAM Symposium on Discrete Algorithms, SODA 1999*, pages 927–928. SIAM, 1999.

[17] D. Kratsch and M. Liedloff. An exact algorithm for the minimum dominating clique problem. In *Proceedings 2nd International Workshop on Parameterized and Exact Computation, IWPEC 2006*, pages 128–139. Springer Verlag, Lecture Notes in Computer Science, vol. 4169, 2006.

[18] I. Schiermeyer. Efficiency in exponential time for domination-type problems. *Discrete Applied Mathematics*, 156:3291–3297, 2008.

[19] J. M. Robson. Algorithms for maximum independent sets. *Journal of Algorithms*, 7:425–440, 1986.

[20] J. M. Robson. Finding a maximum independent set in time $O(2^{n/4})$. Technical report 1251-01, LaBRI, Université Bordeaux I, 2001.

[21] U. Schöning. Algorithmics in exponential time. In *Proceedings 10th Scandinavian Workshop on Algorithm Theory, SWAT 2005*, pages 36–43, Springer Verlag, Lecture Notes in Computer Science, vol. 4059, 2006.

[22] R. E. Tarjan. Finding a maximum clique. Technical report, Department of Computer Science, Cornell University, Ithaca, NY, 1972.

[23] R. E. Tarjan and A. Trojanowski. Finding a maximum independent set. *SIAM J. Comput.*, 6:537–546, 1977.

[24] J. M. M. van Rooij. Design by measure and conquer: an $O(1.5086^n)$ algorithm for dominating set and similar problems. Utrecht University, Master Thesis, INF/SCR.06.05, 2006.

[25] J. M. M. van Rooij and H. L. Bodlaender. Design by measure and conquer: a faster exact algorithm for dominating set. In *Proceedings 25st Symposium on Theoretical Aspects of Computer Science, STACS 2008*, pages 657–668, 2008.

[26] J. M. M. van Rooij and H. L. Bodlaender. Exact algorithms for edge domination. In *Proceedings 3th International Workshop on Parameterized and Exact Computation, IWPEC 2008*, pages 214–225. Springer Verlag, Lecture Notes in Computer Science, vol. 5018, 2008.

[27] J. M. M. van Rooij, J. Nederlof, and T. C. van Dijk. Inclusion/exclusion meets measure and conquer: exact algorithms for counting dominating sets. In *Proceedings 17th Annual European Symposium on Algorithms, ESA 2009*, pages 554–565. Springer Verlag, Advanced Research in Computing and Software Science, Lecture Notes in Computer Science, vol. 5757, 2009.

[28] G. J. Woeginger. Exact algorithms for NP-hard problems: a survey. In *Combinatorial Optimization: "Eureka, you shrink"*, pages 185–207, Springer Verlag, Lecture Notes in Computer Science, vol. 2570, 2003.

[29] G. J. Woeginger. Space and time complexity of exact algorithms: some open problems. In *Proceedings 1st International Workshop on Parameterized and Exact Computation, IWPEC 2004*, pages 281–290, Springer Verlag, Lecture Notes in Computer Science, vol. 3162, 2004.