Trimmed Moebius Inversion and Graphs of Bounded Degree

Andreas Björklund, Thore Husfeldt, Petteri Kaski, Mikko Koivisto

Symposium on Theoretical Aspects of Computer Science 2008 (Bordeaux), pp. 85-96
www.stacs-conf.org

TRIMMED MOEBIUS INVERSION AND GRAPHS OF BOUNDED DEGREE

ANDREAS BJÖRKLUND [1], THORE HUSFELDT [1], PETTERI KASKI [2], AND MIKKO KOIVISTO [2]

[1] Lund University, Department of Computer Science, P.O.Box 118, SE-22100 Lund, Sweden
E-mail address: andreas.bjorklund@logipard.com, thore.husfeldt@cs.lu.se

[2] Helsinki Institute for Information Technology HIIT, University of Helsinki, Department of Computer Science, P.O.Box 68, FI-00014 University of Helsinki, Finland
E-mail address: {petteri.kaski,mikko.koivisto}@cs.helsinki.fi

Abstract. We study ways to expedite Yates's algorithm for computing the zeta and Moebius transforms of a function defined on the subset lattice. We develop a trimmed variant of Moebius inversion that proceeds point by point, finishing the calculation at a subset before considering its supersets. For an n-element universe U and a family $\mathcal{F}$ of its subsets, trimmed Moebius inversion allows us to compute the number of packings, coverings, and partitions of U with k sets from $\mathcal{F}$ in time within a polynomial factor (in n) of the number of supersets of the members of $\mathcal{F}$.

Relying on an intersection theorem of Chung *et al.* (1986) to bound the sizes of set families, we apply these ideas to well-studied combinatorial optimisation problems on graphs of maximum degree Δ. In particular, we show how to compute the Domatic Number in time within a polynomial factor of $(2^{\Delta+1}-2)^{n/(\Delta+1)}$ and the Chromatic Number in time within a polynomial factor of $(2^{\Delta+1}-\Delta-1)^{n/(\Delta+1)}$. For any constant Δ, these bounds are $O\big((2-\epsilon)^n\big)$ for $\epsilon > 0$ independent of the number of vertices n.

1. Introduction

Yates's algorithm from 1937 is a kind of fast Fourier transform that computes for a function $f : \{0,1\}^n \to \mathbf{R}$ and another function $\upsilon : \{0,1\} \times \{0,1\} \to \mathbf{R}$ the values

$$\widehat{f}(x_1,\ldots,x_n) \;=\; \sum_{y_1,\ldots,y_n \in \{0,1\}} \upsilon(x_1,y_1)\cdots\upsilon(x_n,y_n) f(y_1,\ldots,y_n)\,. \tag{1.1}$$

simultaneously for all $X = (x_1,\ldots,x_n) \in \{0,1\}^n$ using only $O(2^n n)$ operations, instead of the obvious $O(4^n n)$. The algorithm is textbook material in many sciences. Yet, though it appears in Knuth [13, §3.2], it has received little attention in combinatorial optimisation.

Recently, the authors [3, 4] used Yates's algorithm in combination with Moebius inversion to give algorithms for a number of canonical combinatorial optimisation problems such as Chromatic Number and Domatic Number in n-vertex graphs, and n-terminal Minimum Steiner Tree, in running times within a polynomial factor of 2^n.

This research was supported in part by the Academy of Finland, Grants 117499 (P.K.) and 109101 (M.K.).

From the way it is normally stated, Yates's algorithm seems to face an inherent 2^n lower bound, up to a polynomial factor, and it also seems to be oblivious to the structural properties of the transform it computes.

The motivation of the present investigation is to expedite the running time of Yates's algorithm for certain structures so as to get running times with a dominating factor of the form $(2-\epsilon)^n$. From the perspective of running times alone, our improvements are modest at best, but apart from providing evidence that the aesthetically appealing 2^n bound from [4] can be beaten, the combinatorial framework we present seems to be new and may present a fruitful direction for exact exponential time algorithms.

1.1. Results

In a graph $G = (V, E)$, a set $D \subseteq V$ of vertices is *dominating* if every vertex not in D has at least one neighbour in D. The *domatic number* of G is the largest k for which V can be partitioned in to k dominating sets. We show how to compute the domatic number of an n-vertex graph with maximum degree Δ in time

$$O^*\big((2^{\Delta+1} - 2)^{n/(\Delta+1)}\big)\,;$$

the O^* notation suppresses factors that are polynomial in n. For constant Δ, this bound is always better than 2^n, though not by much:

Δ	3	4	5	6	7	8	$\cdots$
$(2^{\Delta+1} - 2)^{1/(\Delta+1)}$	1.9344	1.9744	1.9895	1.9956	1.9981	1.9992	$\cdots$

The *chromatic number* of a graph is the minimum k for which the vertex set can be covered with k independent sets; a set $I \subseteq V$ is *independent* if no two vertices in I are neighbours. We show how to compute the chromatic number of an n-vertex graph with maximum degree Δ in time

$$O^*\big((2^{\Delta+1} - \Delta - 1)^{n/(\Delta+1)}\big)\,.$$

This is slightly faster than for Domatic Number:

Δ	3	4	5	6	7	8	$\cdots$
$(2^{\Delta+1} - \Delta - 1)^{1/(\Delta+1)}$	1.8613	1.9332	1.9675	1.9840	1.9921	1.9961	$\cdots$

One notes that even for moderate Δ, the improvement over 2^n is minute. Moreover, the colouring results for $\Delta \leq 5$ are not even the best known: by Brooks's Theorem [5], the chromatic number of a connected graph is bounded by its maximum degree unless the graph is complete or an odd cycle, both of which are easily recognised. It remains to decide if the chromatic number is 3, 4, or 5, and with algorithms from the literature, 3- and 4-colourability can be decided in time $O(1.3289^n)$ [1] and $O(1.7504^n)$ [6], respectively. However, this approach does stop at $\Delta = 5$, since we know no $o(2^n)$ algorithm for 5-colourability. Other approaches for colouring low-degree graphs are known via pathwidth: given a path decomposition of width w the k-colourability can be decided in time $k^w n^{O(1)}$ [11]; for 6-regular graphs one can find a decomposition with $w \leq n(23+\epsilon)/45$ for any $\epsilon > 0$ and sufficiently large n [11], and for graphs with m edges one can find $w \leq m/5.769 + O(\log n)$ [12]. However, even these pathwidth based bounds fall short when $k \geq 5$—we are not aware of any previous $o(2^n)$ algorithm.

For the general case, it took 30 years and many papers to improve the constant in the bound for Chromatic Number from 2.4423 [14] via 2.4151 [9], 2.4023 [6], 2.3236 [2],

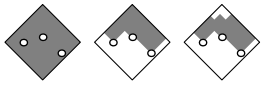


Figure 1: Trimmed evaluation. Originally, Yates's algorithm considers the entire subset lattice (left). We trim the evalation from below by considering only the supersets of 'interesting' points (middle), and from above by abandoning computation when we reach certain points (right).

to 2 [4], and a similar (if less glacial) story can be told for the Domatic Number. None of these approaches was sensitive to the density of the graph. Moreover, what interests us here is not so much the size of the constant, but the fact that it is less than 2, dispelling the tempting hypothesis that 2^n should be a 'difficult to beat' bound for computing the Chromatic Number for sparse graphs. In §4 we present some tailor-made variants for which the running time improvement from applying the ideas of the present paper are more striking.

Chromatic Number and Domatic Number are special cases of set partition problems, where the objective is to partition an n-element set U (here, the vertices of a graph) with members of a given family $\mathcal{F}$ of its subsets (here, the independent or dominating sets of the graph). In full generality, we show how to compute the covering, packing, and partition numbers of $(U, \mathcal{F})$ in time within a polynomial factor of

$$\left|\{T \subseteq U : \text{there exists an } S \in \mathcal{F} \text{ such that } S \subseteq T\}\right|, \tag{1.2}$$

the number of supersets of the members of $\mathcal{F}$. In the worst case, this bound is not better than 2^n, and the combinatorial challenge in applying the present ideas is to find good bounds on the above expression.

1.2. Techniques

The main technical contribution in this paper, sketched in Figure 1, is that Yates's algorithm can, for certain natural choices of $\upsilon : \{0,1\} \times \{0,1\} \to \mathbf{R}$, be trimmed by considering in a bottom-up fashion only those $X \in \{0,1\}^n$ that we are actually interested in, for example those X for which $f(X) \neq 0$ and their supersets. (We will understand X as a subset of $\{1, \ldots, n\}$ whenever this is convenient.) Among the transforms that are amenable to trimming are the zeta and Moebius transforms on the subset lattice.

We use the trimmed algorithms for zeta and Moebius transforms to expedite Moebius inversion, a generalisation of the principle of inclusion–exclusion, which allows us to compute the cover, packing, and partition numbers. The fact that these numbers can be computed via Moebius inversion was already used in [2, 3, 4], and those parts of the present paper contain little that is new, except for a somewhat more explicit and streamlined presentation in the framework of partial order theory.

The fact that we can evaluate both the zeta and Moebius transforms pointwise in such a way that we are done with X before we proceed to Y for every $Y \supset X$ also enables us to further trim computations from what is outlined above. For instance, if we seek a minimum set partition of sets from a family $\mathcal{F}$ of subsets of U, then it suffices to find the minimum partition of all X such that $U \setminus X = S$ for some $S \in \mathcal{F}$. In particular, we need not consider how many sets it takes to partition X for X's large enough for $U \setminus X$ not to contain any set from $\mathcal{F}$.

The main combinatorial contribution in this paper is that if $\mathcal{F}$ is the family of maximal independent sets, or the family of dominating sets in a graph, then we show how to bound (1.2) in terms of the maximum degree Δ using an intersection theorem of Chung *et al.* [8] that goes back to Shearer's Entropy Lemma. For this we merely need to observe that

the intersection of $\mathcal{F}$ and the closed neighbourhoods of the input graph excludes certain configurations.

In summary, via (1.2) the task of bounding the running time for (say) Domatic Number reduces to a combinatorial statement about the intersections of certain families of sets.

Notation. Yates's algorithm operates on the lattice of subsets of an n-element universe U, and we find it convenient to work with notation established in partial order theory.

For a family $\mathcal{F}$ of subsets of U, let $\min \mathcal{F}$ (respectively, $\max \mathcal{F}$) denote the family of minimal (respectively, maximal) elements of $\mathcal{F}$ with respect to subset inclusion. The *upper closure* (sometimes called *up-set* or *filter*) of $\mathcal{F}$ is defined as

$$\uparrow\mathcal{F} = \{\, T \subseteq U : \text{there exists an } S \in \mathcal{F} \text{ such that } S \subseteq T \,\}.$$

For a function f defined on subsets of U, the *support* of f is defined as

$$\operatorname{supp}(f) = \{\, X \subseteq U : f(X) \neq 0 \}.$$

For a graph G, we let $\mathcal{D}$ denote the family of *dominating sets* of G and $\mathcal{I}$ the family of *independent sets* of G. Also, for a subset $W \subseteq V$ of vertices, we let $G[W]$ denote the subgraph induced by W. For a proposition P, we use Iverson's bracket notation $[P]$ to mean 1 if P is true and 0 otherwise.

2. Trimmed Moebius Inversion

For a family $\mathcal{F}$ of sets from $\{0,1\}^n$ and a set $X \in \{0,1\}^n$ we will consider k-tuples $(S_1, \ldots, S_k)$ with $S_i \in \mathcal{F}$ and $S_i \subseteq X$. Such a tuple is *disjoint* if $S_{i_1} \cap S_{i_2} = \emptyset$ for all $1 \leq i_1 < i_2 \leq k$, and *covering* if $S_1 \cup \cdots \cup S_k = X$. From these concepts we define for fixed k

(1) the *cover number* $c(X)$, viz. the number of covering tuples,
(2) the *packing number* $p(X)$, viz. the number of disjoint tuples,
(3) the *partition number* or *disjoint cover number* $d(X)$, viz. the number of tuples that are both disjoint and covering.

In this section we show how to compute these numbers in time $|\uparrow\mathcal{F}|n^{O(1)}$, rather than $2^n n^{O(1)}$ as in [3, 4]. The algorithms are concise but somewhat involved, and we choose to present them here starting with an explanation of Yates's algorithm. Thus, the first two subsections are primarily expository and aim to establish the new ingredients in our algorithms.

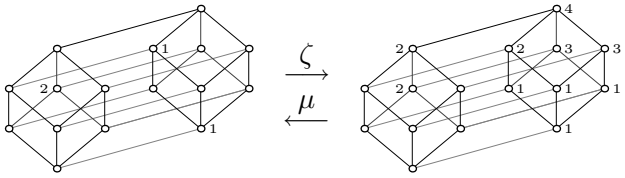
At the heart of our algorithms lie two transforms of functions $f : \{0,1\}^n \to \mathbf{R}$ on the subset lattice. The *zeta* transform $f\zeta$ is defined for all $X \in \{0,1\}^n$ by

$$(f\zeta)(X) = \sum_{Y \subseteq X} f(Y). \tag{2.1}$$

(The notation $f\zeta$ can be read either as a formal operator or as a product of the 2^n-dimensional vector f and the matrix ζ with entries $\zeta_{YX} = [Y \subseteq X]$.) The *Moebius* transform $f\mu$ is defined for all $X \in \{0,1\}^n$ by

$$(f\mu)(X) = \sum_{Y \subseteq X} (-1)^{|X \setminus Y|} f(Y). \tag{2.2}$$

These transforms are each other's inverse in the sense that $f = f\zeta\mu = f\mu\zeta$, a fundamental combinatorial principle called *Moebius inversion*. We can (just barely) draw an example in four dimensions for a function f given by $f(\{4\}) = f(\{1,2,4\}) = 1$, $f(\{1,3\}) = 2$ and $f(X) = 0$ otherwise:



Another example that we will use later is the connection between the packing number and the disjoint cover number,

$$p = d\zeta\,, \tag{2.3}$$

which is easy to verify: By definition,

$$(d\zeta)(X) \;=\; \sum_{Y\subseteq X} d(Y)\,.$$

Every disjoint k-tuple $(S_1,\ldots,S_k)$ with $S_1\cup\cdots\cup S_k \subseteq X$ appears once on the right hand side, namely for $Y = S_1\cup\cdots\cup S_k$, so this expression equals the packing number $p(X)$.

2.1. Yates's algorithm

Yates's algorithm [17] expects the transform in the form of a function $\upsilon : \{0,1\}\times\{0,1\}\to\mathbf{R}$ and computes the transformed values

$$\widehat{f}(X) \;=\; \sum_{Y\in\{0,1\}^n} \upsilon(x_1,y_1)\cdots\upsilon(x_n,y_n)f(Y)\,. \tag{2.4}$$

simultaneously for all $X\in\{0,1\}^n$. Here, we let $(x_1,\ldots,x_n)$ and $(y_1,\ldots,y_n)$ denote the binary representations (or, 'incidence vectors') of X and Y, so $x_j = [j\in X]$ and $y_j = [j\in Y]$. To obtain (2.1) set $\upsilon(x,y) = [y\le x]$ and to obtain (2.2) set $\upsilon(x,y) = [y\le x](-1)^{x-y}$.

The direct evaluation of (2.4) would take 2^n evaluations of f for each X, for a total of $O(2^n2^nn) = O(4^nn)$ operations. The zeta and Moebius transforms depend only on $Y\subseteq X$, so they would require only $\sum_X 2^{|X|} = \sum_{0\le i\le n}\binom{n}{i}2^i = 3^n$ evaluations. Yates's algorithm is faster still and computes the general form in $O(2^nn)$ operations:

Algorithm Y. (*Yates's algorithm.*) Computes $\widehat{f}(X)$ defined in (2.4) for all $X\in\{0,1\}^n$ given $f(Y)$ for all $Y\in\{0,1\}^n$ and $\upsilon(x,y)$ for all $x,y\in\{0,1\}$.

Y1: For each $X\in\{0,1\}^n$, set $g_0(X) = f(X)$.

Y2: For each $j = 1,\ldots,n$ and $X\in\{0,1\}^n$, set

$$g_j(X) = \upsilon([j\in X],0)g_{j-1}(X\setminus\{j\}) + \upsilon([j\in X],1)g_{j-1}(X\cup\{j\})\,.$$

Y3: Output g_n.

The intuition is to compute $\widehat{f}(X)$ 'coordinate-wise' by fixing fewer and fewer bits of X in the sense that, for $j = 1,\ldots,n$,

$$g_j(X) \;=\; \sum_{y_1,\ldots,y_j\in\{0,1\}} \upsilon(x_1,y_1)\cdots\upsilon(x_j,y_j)f(y_1,\ldots,y_j,x_{j+1},\ldots,x_n)\,. \tag{2.5}$$

Indeed, the correctness proof is a straightforward verification (by induction) of the above expression.

2.2. Trimmed pointwise evaluation

To set the stage for our present contributions, observe that both the zeta and Moebius transforms 'grow upwards' in the subset lattice in the sense that $\mathrm{supp}(f\zeta), \mathrm{supp}(f\mu) \subseteq {\uparrow}\mathrm{supp}(f)$. Thus, in evaluating the two transforms, one ought to be able to trim off redundant parts of the lattice and work only with lattice points in ${\uparrow}\mathrm{supp}(f)$.

We would naturally like trimmed evaluation to occur in $O(|{\uparrow}\mathrm{supp}(f)|n)$ operations, in the spirit of Algorithm Y. However, to obtain the values at X in Step Y2 of Algorithm Y, at first sight it appears that we must both 'look up' (at $X \cup \{j\}$) and 'look down' (at $X \setminus \{j\}$). Fortunately, it suffices to only 'look down'. Indeed, for the zeta transform, setting $\upsilon(x, y) = [y \leq x]$ and simplifying Step Y2 yields

$$g_j(X) = [j \in X]g_{j-1}(X \setminus \{j\}) + g_{j-1}(X)\,. \tag{2.6}$$

For the Moebius transform, setting $\upsilon(x, y) = [y \leq x](-1)^{x-y}$ and simplifying yields

$$g_j(X) = -[j \in X]g_{j-1}(X \setminus \{j\}) + g_{j-1}(X)\,. \tag{2.7}$$

Furthermore, it is not necessary to look 'too far' down: for both transforms it is immediate from (2.5) that

$$g_j(X) = 0 \text{ holds for all } X \notin {\uparrow}\mathrm{supp}(f) \text{ and } j = 0, \ldots, n\,. \tag{2.8}$$

In what follows we tacitly employ (2.8) to limit the scope of (2.6) and (2.7) to ${\uparrow}\mathrm{supp}(f)$.

The next observation is that the lattice points in ${\uparrow}\mathrm{supp}(f)$ can be evaluated in order of their rank, using sets $\mathcal{L}(r)$ containing the points of rank r. Initially, the sets $\mathcal{L}(r)$ contain only $\mathrm{supp}(f)$, but we add elements from ${\uparrow}\mathrm{supp}(f)$ as we go along. These observations result in the following algorithm for evaluating the zeta transform; the algorithm for evaluating the Moebius transform is obtained by replacing (2.6) in Step Z3 with (2.7).

Algorithm Z. (*Trimmed pointwise fast zeta transform.*) Computes the nonzero part of $f\zeta$ given the nonzero part of f. The algorithm maintains $n+1$ families $\mathcal{L}(0), \ldots, \mathcal{L}(n)$ of subsets $X \in \{0,1\}^n$; $\mathcal{L}(r)$ contains only sets of size r. We compute auxiliary values $g_j(X)$ for all $1 \leq j \leq n$ and $X \in {\uparrow}\mathrm{supp}(f)$; it holds that $g_n(X) = (f\zeta)(X)$.

Z1: For each $X \in \mathrm{supp}(f)$, insert X into $\mathcal{L}(|X|)$. Set the current rank $r = 0$.

Z2: Select any $X \in \mathcal{L}(r)$ and remove it from $\mathcal{L}(r)$.

Z3: Set $g_0(X) = f(X)$. For each $j = 1, \ldots, n$, set

$$g_j(X) = [j \in X]g_{j-1}(X \setminus \{j\}) + g_{j-1}(X)\,.$$

[At this point $g_n(X) = (f\zeta)(X)$.]

Z4: If $g_n(X) \neq 0$, then output X and $g_n(X)$.

Z5: For each $j \notin X$, insert $X \cup \{j\}$ into $\mathcal{L}(r+1)$.

Z6: If $\mathcal{L}(r)$ is empty then increment $r \leq n$ until $\mathcal{L}(r)$ is nonempty; terminate if $r = n$ and $\mathcal{L}(n)$ is empty.

Z7: Go to Z2.

Observe that the evaluation at X is complete once Step Z3 terminates, which enables further trimming of the lattice 'from above' in case the values at lattice points with higher rank are not required.

By symmetry, the present ideas work just as well for transforms that 'grow downwards', in which case one needs to 'look up'. However, they do not work for transforms that grow in both directions, such as the Walsh–Hadamard transform.

In the applications that now follow, f will always be the indicator function of a family $\mathcal{F}$. In this case having $\operatorname{supp}(f)$ quickly available translates to $\mathcal{F}$ being efficiently listable; for example, with polynomial delay.

2.3. Covers

The easiest application of the trimmed Moebius inversion computes for each $X \in \uparrow\mathcal{F}$ the cover number $c(X)$. This is a particularly straightforward function of the zeta transform of the indicator function f: simply raise each element of $f\zeta$ to the kth power and transform the result back using μ. To see this, observe that both sides of the equation

$$(c\zeta)(Y) = \big((f\zeta)(Y)\big)^k \tag{2.9}$$

count the number of ways to choose k-tuples $(S_1, \ldots, S_k)$ with $S_i \subseteq Y$ and $S_i \in \mathcal{F}$. By Moebius inversion, we can recover c by applying μ to both sides of (2.9).

Algorithm C. (*Cover number.*) Computes $c(X)$ for all $X \in \uparrow\mathcal{F}$ given $\mathcal{F}$. The sets $\mathcal{L}(r)$ and auxiliary values $g_j(X)$ are as in Algorithm Z; also required are auxiliary values $h_j(X)$ for Moebius transform.

C1: For each $X \in \mathcal{F}$, insert X into $\mathcal{L}(|X|)$. Set the current rank $r = 0$.

C2: Select any $X \in \mathcal{L}(r)$ and remove it from $\mathcal{L}(r)$.

C3: [Zeta transform.] Set $g_0(X) = [X \in \mathcal{F}]$. For each $j = 1, \ldots, n$, set

$$g_j(X) = [j \in X]g_{j-1}(X \setminus \{j\}) + g_{j-1}(X)\,.$$

[At this point it holds that $g_n(X) = (f\zeta)(X)$.]

C4: [Evaluate zeta transform of $c(X)$.] Set $h_0(X) = g_n(X)^k$.

C5: [Moebius transform.] For each $j = 1, \ldots, n$, set

$$h_j(X) = -[j \in X]h_{j-1}(X \setminus \{j\}) + h_{j-1}(X)\,.$$

C6: Output X and $h_n(X)$.

C7: For each $j \notin X$, insert $X \cup \{j\}$ into $\mathcal{L}(r+1)$.

C8: If $\mathcal{L}(r)$ is empty, then increment $r \leq n$ until $\mathcal{L}(r)$ is nonempty; terminate if $r = n$ and $\mathcal{L}(n)$ is empty.

C9: Go to C2.

2.4. Partitions

What makes the partition problem slightly less transparent is the fact that we need to use dynamic programming to assemble partitions from sets with different ranks. To this end, we need to compute for each rank s the 'ranked zeta transform'

$$(f\zeta^{(s)})(X) \;=\; \sum_{Y \subseteq X, |Y|=s} f(Y)\,.$$

For rank s, consider the number $d^{(s)}(Y)$ of tuples $(S_1, \ldots, S_k)$ with $S_i \in \mathcal{F}$, $S_i \subseteq Y$, $S_1 \cup \cdots \cup S_k = Y$ and $|S_1| + \cdots + |S_k| = s$. Then $d(Y) = d^{(|Y|)}(Y)$. Furthermore, the

zeta-transform $(d^{(s)}\zeta)(X)$ counts the number of ways to choose $(S_1, \ldots, S_k)$ with $S_i \subseteq X$, $S_i \in \mathcal{F}$, and $|S_1| + \cdots + |S_k| = s$. Another way to count the exact same quantity is

$$q(k, s, X) \;=\; \sum_{s_1+\cdots+s_k=s} \prod_{i=1}^{k} (f\zeta^{(s_i)})(X)\,. \tag{2.10}$$

Thus we can recover $d^{(s)}(Y)$ from $q(k, s, X)$ by Moebius inversion.

As it stands, (2.10) is time-consuming to evaluate even given all the ranked zeta transforms, but we can compute it efficiently using dynamic programming based on the recurrence

$$q(k, s, X) = \begin{cases} \sum_{t=0}^{s} q(k-1, s-t, X)(f\zeta^{(t)})(X), & \text{if } k > 1\,, \\ (f\zeta^{(s)})(X), & \text{if } k = 1\,. \end{cases}$$

This happens in Step D4.

Algorithm D. (*Disjoint cover number.*) Computes $d(X)$ for all $X \in {\uparrow}\mathcal{F}$ given $\mathcal{F}$. The sets $\mathcal{L}(r)$ are as in Algorithm Z; we also need auxiliary values $g_j^{(s)}(X)$ and $h_j^{(s)}(X)$ for all $X \in {\uparrow}\mathcal{F}$, $1 \le j \le n$, and $0 \le s \le n$; it holds that $g_n^{(s)}(X) = (f\zeta^{(s)})(X)$ and $h_n^{(s)}(X) = d^{(s)}(X)$.

D1: For each $X \in \mathcal{F}$, insert X into $\mathcal{L}(|X|)$. Set the current rank $r = 0$.

D2: Select any $X \in \mathcal{L}(r)$ and remove it from $\mathcal{L}(r)$.

D3: [Ranked zeta transform.] For each $s = 0, \ldots, n$, set $g_0^{(s)}(X) = [X \in \mathcal{F}][|X| = s]$. For each $j = 1, \ldots, n$ and $s = 0, \ldots, n$, set

$$g_j^{(s)}(X) = [j \in X] g_{j-1}^{(s)}(X \setminus \{j\}) + g_{j-1}^{(s)}(X)\,.$$

[At this point it holds that $g_n^{(s)}(X) = (f\zeta^{(s)})(X)$ for all $0 \le s \le n$.]

D4: [Evaluate zeta transform of $d^{(s)}$.] For each $s = 0, \ldots, n$, set $q(1, s) = g_n^{(s)}(X)$. For each $i = 2, \ldots, k$ and $s = 0, \ldots, n$, set $q(i, s) = \sum_{t=0}^{s} q(i-1, s-t) g_n^{(t)}(X)$.

D5: [Ranked Moebius transform.] For each $s = 0, \ldots, n$, set $h_0^{(s)}(X) = q(k, s)$. For each $j = 1, \ldots, n$ and $s = 0, \ldots, n$, set

$$h_j^{(s)}(X) = -[j \in X] h_{j-1}^{(s)}(X \setminus \{j\}) + h_{j-1}^{(s)}(X)\,.$$

[At this point it holds that $h_n^{(s)}(X) = d^{(s)}(X)$ for all $0 \le s \le n$.]

D6: Output X and $h_n^{(|X|)}(X)$.

D7: For each $j \notin X$, insert $X \cup \{j\}$ into $\mathcal{L}(r+1)$.

D8: If $\mathcal{L}(r)$ is empty, then increment $r \le n$ until $\mathcal{L}(r)$ is nonempty; terminate if $r = n$ and $\mathcal{L}(n)$ is empty.

D9: Go to D2.

2.5. Packings

According to (2.3), to compute $p(X)$ it suffices to zeta-transform the partition number. This amounts to running Algorithm Z after Algorithm D. (For a different approach, see [4].)

3. Applications

3.1. The number of dominating sets in sparse graphs

This section is purely combinatorial. Let $\mathcal{D}$ denote the dominating sets of a graph. A complete graph has $2^n - 1$ dominating sets, and sparse graphs can have almost as many: the n-star graph has 2^{n-1} dominating sets and average degree less than 2. Thus we ask how large $|\mathcal{D}|$ can be for graphs with bounded maximum degree. An easy example is provided by the disjoint union of complete graphs of order $\Delta + 1$: every vertex subset that includes at least one vertex from each component is dominating, so $|\mathcal{D}| = (2^{\Delta+1} - 1)^{n/(\Delta+1)}$. We shall show that this is in fact the largest possible $\mathcal{D}$ for graphs of maximum degree Δ. Our analysis is based on the following intersection theorem.

Lemma 3.1 (Chung *et al.* [8])**.** *Let U be a finite set with subsets $P_1, \ldots, P_m$ such that every $u \in U$ is contained in at least δ subsets. Let $\mathcal{F}$ be a family of subsets of U. For each $1 \le \ell \le m$, define the projections $\mathcal{F}_\ell = \{ F \cap P_\ell : F \in \mathcal{F} \}$. Then*

$$|\mathcal{F}|^\delta \le \prod_{\ell=1}^{m} |\mathcal{F}_\ell| .$$

Theorem 3.2. *The number of dominating sets of an n-vertex graph with maximum degree Δ is at most $(2^{\Delta+1} - 1)^{n/(\Delta+1)}$.*

Proof. Let $G = (V, E)$ be a graph with $|V| = n$ and maximum degree Δ. For each $v \in V$, let A_v be the closed neighbourhood around vertex v,

$$A_v = \{v\} \cup \{ u \in V : uv \in E \} . \tag{3.1}$$

Next, for each $u \in V$ with degree $d(u) < \Delta$, add u to $\Delta - d(u)$ of the sets A_v not already containing u (it does not matter which). Let $a_v = |A_v|$ and note that $\sum_v a_v = (\Delta + 1)n$.

We want to apply Lemma 3.1. To this end, let $U = V$ and $m = n$. By construction, every $u \in V$ belongs to exactly $\delta = \Delta + 1$ subsets A_v. To get a nontrivial bound on $\mathcal{D}$ we need to bound the size of $\mathcal{D}_v = \{ D \cap A_v : D \in \mathcal{D} \}$. Every $D \cap A_v$ is one of the 2^{a_v} subsets of A_v, but none of the $D \cap A_v$ can be the empty set, because either v or one of its neighbours must belong to the dominating set D. Thus $|\mathcal{D}_v| \le 2^{a_v} - 1$. By Lemma 3.1, we have

$$|\mathcal{D}|^{\Delta+1} \le \prod_v (2^{a_v} - 1) . \tag{3.2}$$

Since $x \mapsto \log(2^x - 1)$ is concave, Jensen's inequality gives

$$\frac{1}{n} \sum_v \log(2^{a_v} - 1) \le \log(2^{\sum_v a_v/n} - 1) = \log(2^{\Delta+1} - 1) .$$

Taking exponentials and combining with (3.2) gives $|\mathcal{D}|^{\Delta+1} \le (2^{\Delta+1} - 1)^n$. ∎

3.2. Domatic Number

We first observe that a graph can be packed with k dominating sets if and only if it can be packed with k *minimal* dominating sets, so we can consider k-packings from $\min \mathcal{D}$ instead of $\mathcal{D}$. This has the advantage that $\min \mathcal{D}$ can be listed faster than 2^n.

Lemma 3.3 (Fomin *et al.* [10])**.** *Any n-vertex graph has at most $O^*(1.7170^n)$ minimal dominating sets, and they can be listed within that time bound.*

Theorem 3.4. *For an n-vertex graph G with maximum degree Δ we can decide in time*

$$O^*\big((2^{\Delta+1}-2)^{n/(\Delta+1)}\big)$$

whether G admits a packing with k dominating sets.

Proof. We use Algorithm D with $\mathcal{F} = \min \mathcal{D}$. By the above lemma, we can complete Step D1 in time $O^*(1.7170^n)$. The rest of the algorithm requires time $O^*(|{\uparrow}\min \mathcal{D}|)$. Since every superset of a dominating set is itself dominating, ${\uparrow}\min \mathcal{D}$ is a sub-family of $\mathcal{D}$ (in fact, it is exactly $\mathcal{D}$), so Theorem 3.2 bounds the total running time by

$$O^*\big((2^{\Delta+1}-1)^{n/(\Delta+1)}\big)\,.$$

We can do slightly better if we modify Algorithm D in Step D7 to insert $X \cup \{j\}$ only if it excludes at least one vertex for each closed neighbourhood. Put otherwise, we insert $X \cup \{j\}$ only if the set $V \setminus (X \cup \{j\})$ dominates the graph G. The graph then has Domatic Number at least $k+1$ if and only if the algorithm reports some X for which $d(X)$ is nonzero. The running time can again be bounded as in Theorem 3.2 but now $D \cap A_v$ can neither be the empty set, nor be equal to A_v. Thus the application of Lemma 3.1 can be strengthened to yield the claimed result. ∎

3.3. Chromatic Number

Our first argument for Chromatic Number is similar; we give a stronger and slightly more complicated argument in §3.4.

We consider the independent sets $\mathcal{I}$ of a graph. An independent set is not necessarily dominating, but it is easy to see that a *maximal* independent set is dominating. Moreover, the Moon–Moser bound tells us they are few, and Tsukiyama *et al.* tell us how to list them with polynomial delay:

Lemma 3.5 (Moon and Moser [15]; Tsukiyama *et al.* [16])**.** *Any n-vertex graph has at most $O^*(1.4423^n)$ maximal independent sets, and they can be listed within that bound.*

Theorem 3.6. *For an n-vertex graph G with maximum degree Δ we can decide in time*

$$O^*\big((2^{\Delta+1}-1)^{n/(\Delta+1)}\big)$$

whether G admits a covering with k independent sets.

Proof. It is easy to see that G can be covered with k independent sets if and only if it can be covered with k *maximal* independent sets, so we will use Algorithm C on $\max \mathcal{I}$. Step C1 is completed in time $O^*(1.4423^n)$, and the rest of the algorithm considers only the points in ${\uparrow}\max \mathcal{I}$, which all belong to $\mathcal{D}$. Again, Theorem 3.2 bounds the total running time. ∎

3.4. Chromatic Number via bipartite subgraphs

We can do somewhat better by considering the family $\mathcal{B}$ of vertex sets of induced *bipartite* subgraphs, that is, the family of sets $B \subseteq V$ for which the induced subgraph $G[B]$ is bipartite. As before, the literature provides us with a nontrivial listing algorithm:

Lemma 3.7 (Byskov and Eppstein [7])**.** *Any n-vertex graph has at most $O^*(1.7724^n)$ maximal induced bipartite subgraphs, and they can be listed within that bound.*

The family $\max \mathcal{B}$ is more than just dominating, which allows us to use Lemma 3.1 in a stronger way.

Theorem 3.8. *For an n-vertex graph of maximum degree Δ it holds that*

$$|{\uparrow}\max \mathcal{B}| \leq (2^{\Delta+1} - \Delta - 1)^{n/(\Delta+1)} .$$

Proof. Let $G = (V, E)$ be a graph with $|V| = n$ and maximum degree Δ. Let $\mathcal{F} = {\uparrow}\max \mathcal{B}$. Let A_v be as in (3.1). With the objective of applying Lemma 3.1, we need to bound the number of sets in $\mathcal{F}_v = \{ F \cap A_v : F \in \mathcal{F} \}$.

Assume first that G is Δ-regular. Let $A_v = \{v, u_1, \ldots, u_\Delta\}$. We will rule out $\Delta + 1$ candidates for $F \cap A_v$, namely

$$\emptyset, \{u_1\}, \ldots, \{u_\Delta\} \notin \mathcal{F}_v . \tag{3.3}$$

This then shows that $|\mathcal{F}_v| \leq 2^{\Delta+1} - \Delta - 1$ and thus the bound follows from Lemma 3.1.

To see that (3.3) holds, observe that $F \in \mathcal{F}$ contains a $B \subseteq F$ such that the induced subgraph $G[B]$ is maximal bipartite. To reach a contradiction, assume that there exists a $v \in V$ with $F \cap A_v \subseteq \{u_\ell\}$. Since $B \subseteq F$, we have $B \cap A_v \subseteq \{u_\ell\}$, implying that v does not belong to B, and that at most one of its neighbours does. Consequently, $G[B \cup \{v\}]$ is also bipartite, and v belongs to a partite set opposite to any of its neighbours. This contradicts the fact that $G[B]$ is maximal bipartite.

To establish the non-regular case, we can proceed as in the proof of Theorem 3.2, adding each $u \in V$ with $d(u) < \Delta$ to some $\Delta - d(u)$ of the sets A_v not already including u. Note that by adding y new vertices to A_v originally containing x vertices, we get $|\{F \cap A_v : F \in \mathcal{F}\}| \leq 2^y(2^x - x - 1)$. Next, since $2^y(2^x - x - 1) \leq 2^{y+x} - (y + x) - 1$ for all non-negative integers y, x and $\log(2^x - x - 1)$ is a concave function, the bound follows as before via Jensen's inequality. ∎

Theorem 3.9. *For an n-vertex graph G with maximum degree Δ we can decide in time*

$$O\big((2^{\Delta+1} - \Delta - 1)^{n/(\Delta+1)}\big)$$

whether G admits a covering with k independent sets.

Proof. When k is even, it is easy to see that G can be covered by k independent sets if and only if it can be covered by $k' = k/2$ maximal bipartite sets, so we will use Algorithm C on $\max \mathcal{B}$ and investigate whether $c(V) \neq 0$.

When k is odd, we again use Algorithm C with $k' = (k - 1)/2$ maximal bipartite sets, but this time we check whether an X is output such that both $c(X) \neq 0$ and $V \setminus X$ is independent in G.

In both cases the running time bound follows from Theorem 3.8. ∎

4. Concluding Remarks

Since the presented improvements on running time bounds are modest, one can ask whether this is because of weak bounds or because of inherent limitations of the technique. We observe that the running time bounds in Theorems 3.4, 3.6, and 3.9 are met by a disjoint union of complete graphs of order $\Delta + 1$. Thus, either further trimming or splitting into connected components is required for improved algorithms in this context.

We chose to demonstrate the technique for Chromatic and Domatic Number since these are well-known and well-studied. To briefly demonstrate some further application potential, more artificial problem variants such as determining if a Δ-regular graph has domatic number at least $\Delta/2$, or if the square of a Δ-regular graph has chromatic number at most

$3\Delta/2$, admit stronger bounds. For example, if G has domatic number at least d, d even, then its vertices can be partitioned into two sets, both of which contain $d/2$ dominating sets. This suggests the following meet-in-the-middle strategy. Run Algorithm D with $\mathcal{F}$ equal to all dominating sets and $k = d/2$, but modify Step D7 to insert $X \cup \{j\}$ only if $|A_v \setminus (X \cup \{j\})| \geq d/2$ holds for all vertices v. At termination, we check whether the algorithm has output two sets X and Y such that $X \cup Y = V$ and $d(X), d(Y) > 0$. (For example, one can check for duplicates in a table with entry $\{X, V \setminus X\}$ for each output X with $d(X) > 0$.) This algorithm variant considers only sets with many forbidden intersections with the neighbourhoods of vertices, which translates into stronger bounds via Lemma 3.1.

References

[1] R. Beigel and D. Eppstein, 3-*coloring in time* $O(1.3289^n)$, J. Algorithms **54** (2005), 168–204.

[2] A. Björklund and T. Husfeldt, *Exact algorithms for exact satisfiability and number of perfect matchings*, Algorithmica, to appear. Prelim. version in Proc. 33rd International Colloquium on Automata, Languages and Programming, Lect. Notes in Comp. Science, Vol. 4051, Springer, Berlin, 2006, pp. 548–559.

[3] A. Björklund, T. Husfeldt, P. Kaski, and M. Koivisto, *Fourier meets Möbius: fast subset convolution*, Proc. 39th ACM Symposium on Theory of Computing, Association for Computing Machinery, New York, NY, 2007, pp. 67–74.

[4] A. Björklund, T. Husfeldt, and M. Koivisto, *Set partitioning via inclusion–exclusion*, SIAM J. Comput., to appear. Prelim. versions in Proc. 47th IEEE Symposium on Foundations of Computer Science, IEEE Computer Society, Los Alamitos, CA, 2006, pp. 575–582, 583–590.

[5] R. L. Brooks, *On colouring the nodes of a network*, Proc. Cambridge Philos. Soc. **37** (1941), 194–197.

[6] J. M. Byskov, *Enumerating maximal independent sets with applications to graph colouring*, Oper. Res. Lett. **32** (2004), 547–556.

[7] ———, *Exact algorithms for graph colouring and exact satisfiability*, Ph.D. Thesis, Univ. Aarhus, 2004.

[8] F. R. K. Chung, P. Frankl, R. L. Graham, and J. B. Shearer, *Some intersection theorems for ordered sets and graphs*, J. Combin. Theory Ser. A **43** (1986), 23–37.

[9] D. Eppstein, *Small maximal independent sets and faster exact graph coloring*, J. Graph Algorithms Appl. **7** (2003), 131–140.

[10] F. V. Fomin, F. Grandoni, A. V. Pyatkin, and A. A. Stepanov, *Bounding the number of minimal dominating sets: a measure and conquer approach*, Proc. 16th Intern. Symposium on Algorithms and Computation, Lect. Notes in Comp. Science, Vol. 3827, Springer, Berlin, 2005, pp. 573–582.

[11] F. V. Fomin, S. Gaspers, S. Saurabh, and A. A. Stepanov, *On two techniques of combining branching and treewidth*, Algorithmica, to appear. Reports in Informatics, no. 337, Department of Informatics, University of Bergen, 2006.

[12] J. Kneis, D. Mölle, S. Richter, and P. Rossmanith, *Algorithms based on the treewidth of sparse graphs*, Revised Selected Papers from the 31st Intern. Workshop on Graph-Theoretic Concepts in Computer Science, Lect. Notes in Comp. Science, Vol. 3787, Springer, Berlin, 2005, pp. 385–396.

[13] D. E. Knuth, *The Art of Computer Programming, Vol.* 2: *Seminumerical Algorithms*, 3rd ed., Addison–Wesley, Reading, MA, 1997.

[14] E. L. Lawler, *A note on the complexity of the chromatic number problem*, Inform. Process. Lett. **5** (1976), 66–67.

[15] J. W. Moon and L. Moser, *On cliques in graphs*, Israel J. Math. **3** (1965), 23–28.

[16] S. Tsukiyama, M. Ide, H. Ariyoshi, and I. Shirakawa, *A new algorithm for generating all the maximal independent sets*, SIAM J. Comput. **6** (1977), 505–517.

[17] F. Yates, *The design and analysis of factorial experiments*, Technical Communication 35, Commonwealth Bureau of Soils, Harpenden, U.K., 1937.