An $O^*(2^n)$ Algorithm for Graph Coloring and Other Partitioning Problems via Inclusion–Exclusion

Mikko Koivisto
HIIT Basic Research Unit, Department of Computer Science
University of Helsinki, P.O. Box 68, FIN-00014, Finland
mikko.koivisto@cs.helsinki.fi

Abstract

We use the principle of inclusion and exclusion, combined with polynomial time segmentation and fast Möbius transform, to solve the generic problem of summing or optimizing over the partitions of n elements into a given number of weighted subsets. This problem subsumes various classical graph partitioning problems, such as graph coloring, domatic partitioning, and MAX k-CUT*, as well as machine learning problems like decision graph learning and model-based data clustering. Our algorithm runs in $O^*(2^n)$ time, thus substantially improving on the usual $O^*(3^n)$-time dynamic programming algorithm; the notation O^* suppresses factors polynomial in n. This result improves, e.g., Byskov's recent record for graph coloring from $O^*(2.4023^n)$ to $O^*(2^n)$. We note that twenty five years ago, R. M. Karp used inclusion–exclusion in a similar fashion to reduce the space requirement of the usual dynamic programming algorithms from exponential to polynomial.*

1. Introduction

Many important computational problems can be formulated as partitioning problems: Given k functions $f_1, \ldots, f_k$ from the subsets of an n-element set N to the real numbers, with $k \leq n$, compute the partition sum of $f = (f_1, \ldots, f_k)$, defined as

$$\mathrm{par}(f) := \sum_{A} \prod_{c=1}^{k} f_c(A_c)\,, \qquad (1)$$

where $A = (A_1, \ldots, A_k)$ runs through all ordered partitions of N into k disjoint subsets of N. With specific choices of the functions f_c, and possibly replacing sum–product by min–sum or max–sum, this class of partitioning problems subsumes various graph partitioning problems, including the chromatic number problem (k-colorability), the domatic number problem, and MAX k-CUT, to name a few; for other graph partitioning problems see Garey and Johnson [14]. For instance, to compute the chromatic number of a given graph we let $f_c(S) = 1$ if the vertex subset S is an independent set, and 0 otherwise: then the chromatic number is the smallest k for which $\mathrm{par}(f) > 0$. Moreover, there are machine-learning problems, such as model-based data clustering and decision graph learning, which are partitioning tasks where the input, however, is not a graph.

With the above problem specification, it is clear that computing $\mathrm{par}(f)$ requires at least $\Omega(k2^n)$ time, for this is the size of the input. But can we construct an algorithm that achieves this lower bound, say, up to factors polynomial in n?

In this paper we show that the answer is in the affirmative. Thereby, we improve the known time complexity upper bounds down to $O^*(2^n)$ for several well-known problems, including the aforementioned graph partitioning problems (throughout this paper, the notation O^* suppresses factors polynomial in n and k). We present an algorithm that is based on a novel combination of known techniques. The basic idea is to formulate the partitioning problem as a permutation–segmentation problem: Given a sequence of n elements, we can efficiently sum over its partitions into disjoint blocks of consecutive elements. Then, to sum over all permutations we use the principle of inclusion and exclusion. The key observation is that the terms in the inclusion–exclusion expression can be evaluated in $O^*(2^n)$ total time by using the fast Möbius transform algorithm (e.g., [17, 20]). Finally, the optimization variant, finding a partition A that maximizes (or minimizes) the sum $\sum_c f_c(A_c)$, can be reduced to the problem of computing a partition sum of the form (1) via an integer-coding technique and self-reducibility (for similar techniques see, e.g., [29, 19]).

Our algorithms do not rely on any intrinsic properties of the functions f_c. Thus, for specific problems, such as the chromatic number problem, even faster algorithms may exist. On the other hand, the presented algorithms are suitable

for practical use, as their running times do not involve large hidden constants (or high-degree polynomial factors).

1.1. Previous research

Karp [16] pioneered the use of inclusion–exclusion for hard problems. Karp's paper is very compact and has probably gone slightly unnoticed, as it focuses on cases where the technique offers reduction in *space* requirement, compared to a dynamic programming algorithm—typically with a slight *increase* in run time. Later, Bax and Franklin [2, 3, 4] used similar methods to count paths and cycles in general graphs. These techniques provide an $O^*(2^n)$-time and polynomial space algorithm for counting Hamiltonian paths [16, 4, 30] and solving the traveling-salesperson problem (TSP). (Karp [16] mentions TSP but does not provide details. Bax and Franklin have an unpublished paper (1997), where an $O^*(2^n)$-time and polynomial space algorithm is given for TSP (assuming integer weights). Woeginger [31] treats this as an open problem.)

Previous research on graph partitioning is well characterized by efforts put into the graph coloring problem. Lawler [22] applied a straightforward dynamic programming algorithm, which makes use of the following simple observation. A graph G on node set N is k-colorable if and only if there is a node subset S such that S is an independent set and the restriction of G to the remaining nodes $N - S$ is $(k-1)$-colorable. By dynamic programming the subproblems for different subsets S and numbers k can be solved in $O^*(3^n)$ total time for n-vertex graphs. Lawler noticed that it is sufficient to consider maximal independent sets, and by a careful analysis of the number of maximal independent sets, he was able to reduce the run time to $O^*(2.4422^n)$ for arbitrary graphs and to $O^*(1.4422^n)$ for graphs that are 3-colorable. Since then, Lawler's 3-coloring algorithm has been improved in a sequence of papers, with the current record of $O^*(1.3289^n)$ due to Beigel and Eppstein [5]. There has been less progress in the general case. Woeginger [30, Open Problem 3.5, page 7] posed the question whether the general graph coloring problem can be solved in time around $O^*(2^n)$. Eppstein [11] was the first to improve Lawler's result, to $O^*(2.4151^n)$; the current record, based on Lawler's approach, is $O^*(2.4023^n)$ by Byskov [9]. The algorithms behind these results all require exponential memory. Recently, Bodlaender and Kratsch [8] provided a polynomial space algorithm that runs in $O^*(5.283^n)$ time.

Lawler's [22] dynamic programming algorithm has been the starting point also for other partitioning problems. Riege and Rothe [25] give an $O^*(2.9216^n)$-time algorithm for determining whether a given undirected graph can be partitioned into three dominating sets (the three domatic number problem). Koivisto and Sood [21] study an $O^*(3^n)$-time dynamic programming algorithm for partitioning small feature spaces in a machine learning context.

After submitting a preliminary version of this paper, I became aware of some very recent progress on partitioning problems. Fomin et al. [13] give an $O^*(2.8805^n)$ algorithm for finding the domatic number; Riege et al. [26] give an $O^*(2.695^n)$ algorithm for the three domatic number problem. For graph coloring Björklund and Husfeldt [6] give an $O^*(2.3236^n)$ algorithm; remarkably, this algorithm is based on the inclusion–exclusion principle.

Finally, Björklund and Husfeldt [7] (in these proceedings, independently of my work) develop their algorithm further and achieve an $O^*(2^n)$ time bound for graph coloring, domatic partitioning, and some other graph partitioning problems; they also consider polynomial space algorithms and approximation algorithms. The present paper differs from that of Björklund and Husfeldt in three major respects: we allow the subsets in a partition to have arbitrary weights (not restricted to $\{0, 1\}$); we allow the weights to depend on the component label (i.e., color); we consider optimization tasks, in addition summation (counting) problems. A minor, technical difference is also that Björklund and Husfeldt use perhaps a more direct inclusion–exclusion formulation, considering constrained set covers rather than segmentations of sequences.

1.2. Organization

The rest of this paper is organized as follows. In Section 2 we describe some basic techniques: an inclusion–exclusion formula for sums over permutations, a polynomial time segmentation algorithm, and the fast Möbius transform algorithm. In Section 3 we define the SUM WEIGHTED PARTITIONS problem and present an $O^*(2^n)$-time algorithm for it. In Section 4 we show how the optimization variant, the MAX WEIGHTED PARTITION problem, can be solved in $O^*(2^n)$ time by a reduction to SUM WEIGHTED PARTITIONS. In Section 5 we demonstrate the generality of the method by applying it to several well-known partitioning problems, including graph coloring, domatic partitioning, and MAX k-CUT. Moreover, in Section 6 we briefly describe how computation of the partition sum arises naturally in two machine learning applications: Bayesian model-based data clustering and decision graph learning. In Section 7 we conclude by pointing directions for future research.

2. Preliminaries

We begin by introducing the model of computation and some basic notation and definitions used throughout this paper. We also describe some key techniques that constitute the building blocks of the main algorithm presented in Section 3.

2.1. Model of computation

The model of computation used in this work is the random access machine with the restriction that arithmetic operations are considered unit-time only for integers of bounded size. The algorithms described in this paper rely heavily on basic arithmetic operations, and it will be obvious that the time complexity is determined by the required number of operations. To avoid ugly expressions in the running time bounds, we often find it convenient to spell out the number of arithmetic operations and the size of the operands. Accordingly, we may say that an algorithm uses $O(\alpha)$ arithmetic operations with integers of length $O(\beta)$. The corresponding time complexity is then $O(\alpha \cdot \mu(\beta))$, where $\mu(\beta) = \beta \log \beta \log\log \beta$, since multiplication of two β-bit integers takes $O(\mu(\beta))$ time, while addition and subtraction take $O(\beta)$ time [28].

2.2. Summing over permutations via inclusion–exclusion

Throughout this paper, we let N denote an n-element ground set; without loss of generality we let $N := \{1, \ldots, n\}$ for convenience. We will mainly consider subsets of N, functions from N to N, and functions from $2^N = \{S : S \subseteq N\}$ to $\mathbb{R}$ (the real line). If S is a subset of N we let $\mathcal{F}(S)$ denote the set of all mappings from N to S. A function σ in $\mathcal{F}(N)$ is a *permutation* on N if σ is a bijection; we denote the set of all permutations on N by $\mathcal{S}_n$. Note that $\mathcal{F}(S) \subset \mathcal{F}(T)$ whenever $S \subset T$.

We can represent the set of permutations on N by removing functions that are not permutations from the set of all functions:

$$\mathcal{S}_n = \mathcal{F}(N) - \bigcup_{i \in N} \mathcal{F}(N - \{i\}) \,. \tag{2}$$

Thus, the principle of inclusion and exclusion gives us the following important result, generalizing, e.g., Ryser's [27] formula for the permanent of a matrix.

Theorem 1. *Let g be a mapping from $\mathcal{F}(N)$ to the real numbers. Then*

$$\sum_{\sigma \in \mathcal{S}_n} g(\sigma) = \sum_{S \subseteq N} (-1)^{|N-S|} \sum_{\sigma \in \mathcal{F}(S)} g(\sigma) \,.$$

Proof. Use (2) and the principle of inclusion and exclusion. Note that $\bigcap_{i \in T} \mathcal{F}(N - \{i\}) = \mathcal{F}(N - T)$. □

This result is useful when direct summing over the permutations on N is hard but one can efficiently sum over all functions from N to S, for any $S \subseteq N$.

2.3. Summing over segmentations in polynomial time

Segmentations are partitions of ordered sets into disjoint blocks. Formally, we find it convenient to operate with index sets: A *k-segmentation* of a sequence $1, 2, \ldots, n$ is a vector $B = (b_0, b_1, \ldots, b_k)$ with $0 = b_0 \leq b_1 \leq \cdots \leq b_k = n$. We will denote the *block* of consecutive elements $a+1, a+2, \ldots, b$ by ab. Alternatively, we may represent the segmentation B by the vector $(B_1, \ldots, B_k)$ where $B_i = b_{i-1}b_i$. We let $\mathcal{B}_k$ denote the set of all k-segmentations of $1, 2, \ldots, n$.

Let h_c, for $c = 1, \ldots, k$, be a function from the blocks $\{ab : 0 \leq a \leq b \leq n\}$ to $\mathbb{R}$. We define the segmentation sum of $h = (h_1, \ldots, h_k)$ as

$$\mathrm{seg}(h) := \sum_{B} \prod_{c=1}^{k} h_c(B_c) \,,$$

where $B = (B_1, \ldots, B_k)$ runs through all segmentations in $\mathcal{B}_k$

We can efficiently sum over segmentations by dynamic programming (for related algorithms see, e.g., [24]):

Theorem 2. *Let n and k be integers with $1 \leq k \leq n$. Let $h_1, \ldots, h_k$ be functions from $\{ab : 0 \leq a \leq b \leq n\}$ to the integer range $[-M, M]$. Then the segmentation sum $\mathrm{seg}(h)$ can be computed in $O(kn^2)$ operations with integers of length $O(k \log(nM))$.*

Furthermore, if for each c the value $h_c(ab)$ only depends on the length $b - a$ and h_c is represented as a vector of length $n+1$, then $\mathrm{seg}(h)$ can be computed in $O(kn \log n)$ operations with integers of length $O(k \log(nM))$.

Proof (sketch). Consider the functions $g_1, \ldots, g_k$ defined at $b = 0, 1, \ldots, n$ by

$$\begin{aligned} g_1(b) &:= h_1(0b) \qquad \text{and} \\ g_c(b) &:= \sum_{a=0}^{b} g_{c-1}(a) h_c(ab) \qquad \text{for } c = 2, 3, \ldots, k \,. \end{aligned}$$

Observe that $g_k(n) = \mathrm{seg}(h)$. By using the recursion, $g_k(n)$ can be computed in $O(kn^2)$ additions and multiplications with numbers from the range $[-n^k M^k, n^k M^k]$.

For the second part, notice that each g_c (for $c \geq 2$) is obtained as the convolution of the functions g_{c-1} and h_c. Using the fast Fourier transform the number of arithmetic operations reduces to $O(kn \log n)$. □

2.4. The fast Möbius transform on subset lattices

The Möbius transform on the subset lattice $(2^N, \subseteq)$ of a set N is an operator that maps any function $f \colon 2^N \to \mathbb{R}$ to

another function $\hat{f}\colon 2^N \to \mathbb{R}$, defined by

$$\hat{f}(S) := \sum_{T \subseteq S} f(T) \qquad \text{for all } S \subseteq N\,.$$

We say that $\hat{f}$ is the Möbius transform of f.

The straightforward way to compute the Möbius transform (i.e., $\hat{f}(S)$ at every S) uses $O(3^n)$ additions. However, the fast Möbius transform algorithm (FMT) uses only $O(n2^n)$ additions ([17, 20], [18, Chap. 3]).

Theorem 3. *Let N be a set of n elements. Then the Möbius transform on the subset lattice of N, restricted to functions to the range $[-M, M]$, can be computed in $O(n2^n)$ additions with integers of length $O(n \log M)$.*

Proof (sketch). Consider the functions $g_0, g_1, \ldots, g_n$ defined at each $S \subseteq N$ by

$$\begin{aligned} g_0(S) &:= f(S) \qquad \text{and} \\ g_i(S) &:= \sum_{T:T\cup\{i\}=S} g_{i-1}(T) \qquad \text{for } i = 1, 2, \ldots, n\,. \end{aligned}$$

One can show by induction on i that $g_i(S) = \sum_T f(T)$, where T runs through all subsets of S such that $\{j \in T : j > i\} = \{j \in S : j > i\}$. By using the recursion, each g_i can be computed in $O(2^n)$ additions. Thus, $\hat{f} = g_n$ can be computed in $O(n2^n)$ additions with integers from the range $[-2^n M, 2^n M]$. □

3. Summing over partitions

We now combine the three techniques described in the previous section to solve the following "counting" problem.

SUM WEIGHTED PARTITIONS
Input: An n-element set N and $k \leq n$ functions $f_1, \ldots, f_k$ from 2^N to the integer range $[-M, M]$.
Output: The value of the sum $\mathrm{par}(f) := \sum_A \prod_{c=1}^k f_c(A_c)$, where $A = (A_1, \ldots, A_k)$ runs through all ordered k-partitions of N.

We will show that SUM WEIGHTED PARTITIONS can be solved in $O^*(2^n)$ time, up to factors logarithmic in M. The basic idea is to sum over all ordered partitions by summing over all segmentations of all permutations (Lemma 4 below). Using inclusion–exclusion (Theorem 1), we show that the sum over all permutations can be expressed in terms of a sum over all functions from N to a subset $S \subseteq N$ (one sum for each S). We show that this sum can, in turn, be expressed as a segmentation sum, where the weight of a segment is obtained via Möbius transformation.

First we define a suitable function from $\mathcal{F}(N)$ to $\mathbb{R}$ (corresponds to the function g in Theorem 1). For any function σ in $\mathcal{F}(N)$ define

$$\mathrm{seg}(f, \sigma) := \sum_B \prod_{c=1}^k f_c(\sigma(B_c))/(|\sigma(B_c)|!)\,,$$

where B runs through the k-segmentations of $1, \ldots, n$.

Lemma 4. *We have*

$$\mathrm{par}(f) = \sum_{\sigma \in \mathcal{S}_n} \mathrm{seg}(f, \sigma)\,.$$

Proof. Consider the function ψ that maps each (σ, B) from $\mathcal{S}_n \times \mathcal{B}_k$ to the ordered k-partition $\psi(\sigma, B) = A = (A_1, \ldots, A_k)$ by $A_c = \sigma(B_c)$. We observe that the preimage $\psi^{-1}(A) \subseteq \mathcal{S}_n \times \mathcal{B}_k$ has exactly $\prod_c |A_c|!$ members. Thus, for any function g, we have

$$\sum_A g(A) = \sum_{\sigma \in \mathcal{S}_n} \sum_B g(\psi(\sigma, B)) \Big/ \prod_{c=1}^k |\sigma(B_c)|!\,.$$

Substituting $g(A) = \prod_c f_c(A_c)$ yields the claimed expression. □

Next we show that the sum of terms $\mathrm{seg}(f, \sigma)$ over all functions σ in $\mathcal{F}(S)$, for any $S \subseteq N$, can represented as a segmentation sum. To this end, for each $c = 1, \ldots, k$ and $\ell = 0, 1, \ldots, n$ define the function f_c^ℓ by

$$f_c^\ell(T) := f_c(T) \cdot \left\{ {\ell \atop |T|} \right\} \qquad \text{for } T \subseteq N\,,$$

where

$$\left\{ {p \atop q} \right\} := \frac{1}{q!} \sum_{j=0}^{q} (-1)^j \binom{q}{j} (q - j)^p$$

denotes the number of partitions of a p-element set into q nonempty subsets, the Stirling number of the second kind. We let $\hat{f}_c^\ell$ denote the Möbius transform of f_c^ℓ (on the subset lattice of N).

Lemma 5. *Let S be a subset of N. Then*

$$\sum_{\sigma \in \mathcal{F}(S)} \mathrm{seg}(f, \sigma) = \sum_B \prod_{c=1}^k \hat{f}_c^{|B_i|}(S) = \mathrm{seg}(h^S)\,,$$

where $h^S := (h_1^S, \ldots, h_k^S)$ with $h_c^S(ab) := \hat{f}_c^{b-a}(S)$ for $1 \leq a \leq b \leq n$.

Proof. Write the first summation of the claim as

$$\sum_B \sum_{\sigma \in \mathcal{F}(S)} \prod_{c=1}^k f_c(\sigma(B_c))/(|\sigma(B_c)|!) = \sum_B \prod_{c=1}^k F_c(B_c)\,,$$

with the shorthand

$$F_c(B_c) := \sum_{\sigma_c\colon B_c \to S} f_c(\sigma_c(B_c))/(|\sigma(B_c)|!)\,,$$

where σ_c runs through all functions from B_c to S. We observe that the summand depends only on the image of σ_c, say $T \subseteq S$. Thus,

$$F_c(B_c) = \sum_{T\subseteq S} f_c(T) \cdot \left\{ {|B_c| \atop |T|} \right\} = \sum_{T\subseteq S} f_c^{|B_c|}(T) = \hat{f}_c^{|B_c|}(S)\,,$$

as $\left\{ {\ell \atop q} \right\} \cdot q!$ is the number of surjections from an ℓ-element set to a q-element set. This completes the proof of the first equality in the claim.

The second equality in the claim follows directly from the definitions. □

It remains to combine the above lemmas.

Theorem 6. *We have*

$$\mathrm{par}(f) = \sum_{S\subseteq N} (-1)^{|N-S|} \mathrm{seg}(h^S)\,,$$

where h^S is as defined in Lemma 5.

Proof. Use first Lemma 4, then Theorem 1, and finally Lemma 5. □

Having this representation it is easy to prove the main result.

Theorem 7. *The problem* SUM WEIGHTED PARTITIONS *can be solved in $O(kn^2 2^n)$ operations with integers of length $O(n\log(nM))$, thus, in $O^*(2^n\mu(\log M))$ time, where $\mu(\beta) = \beta\log\beta\log\log\beta$.*

Proof. Consider the following algorithm.

1. For all $c = 1,\ldots,k$ and $\ell = 0,1,\ldots,n$: compute $\hat{f}_c^\ell$.
2. For all $S \subseteq N$ and $c = 1,\ldots,k$: compute h_c^S.
3. For all $S \subseteq N$: compute $\mathrm{seg}(h^S)$.
4. Evaluate $\mathrm{par}(f)$ via the inclusion–exclusion formula given in Theorem 6.

By Theorem 6 this algorithm correctly computes $\mathrm{par}(f)$.

Let us analyze the running time of the four steps. Step 1 can be computed using the fast Möbius transform (Theorem 3 in $O(kn^2 2^n)$ operations with integers of length $O(\log(n^n M)) = O(n\log(nM))$. Step 2 is obviously no harder than Step 1, given the functions $\hat{f}_c^\ell$. Step 3 can be computed by using the polynomial time segmentation algorithm (Theorem 2) in $O(2^n kn\log n)$ arithmetic operations with integers of length $O(\log(n^k n^n M)) = O(n\log(nM))$. Step 4 is obviously no harder than Step 3, given the values $\mathrm{seg}(h^S)$ Thus, SUM WEIGHTED PARTITIONS can be solved in $O(kn^2 2^n)$ operations with integers of length $O(n\log(nM))$. □

4. Finding an optimal partition

Next we show how an optimal partition can be found by solving a sequence of related counting problems. We use rather standard techniques of coding with large integers and self-reducibility (see, e.g., [29, 19]).

Formally we consider the following problem.

MAX WEIGHTED PARTITION
Input: An n-element set N and $k \le n$ functions $f_1,\ldots,f_k$ from 2^N to the integer range $[-M, M]$.
Output: An ordered partition $A = (A_1,\ldots,A_k)$ of N such that the sum $\sum_{c=1}^k f_c(A_c)$ is maximized.

First we show that the *maximum total weight*,

$$W_*(f,N) := \max_A \sum_{c=1}^k f_c(A_c)\,,$$

can be computed by a reduction to SUM WEIGHTED PARTITIONS. We use a straightforward coding technique, which has the drawback that the time complexity will depend about linearly on the maximum weight M. A fairly detailed proof is included to support our claims concerning the polynomial factors in the time complexity.

Lemma 8. *Let N and $f_1,\ldots,f_k$ be as above. The maximum $W_*(f,N)$ can be computed in $O(kn^2 2^n)$ operations with integers of length $O(Mn^2\log n)$.*

Proof. Suppose first that each f_c gets only nonnegative values. Denote $W_* := W_*(f,N)$ for short. Consider the following construction. Choose the smallest integer $\beta = 2^d$ such that β is greater than the number of all ordered k-partitions of N; clearly, $d \le n\log_2 n$ suffices. For $c = 1,\ldots,k$, define $g_c\colon 2^N \to \{0,1,\ldots,\}$ by $g_c(S) := \beta^{f_c(S)}$, for $S \subseteq N$. We observe that

$$\mathrm{par}(g) = \alpha_0 + \alpha_1\beta + \cdots + \alpha_{kM}\beta^{kM}\,,$$

where α_r is the number of k-partitions of N for which the sum $\sum_{c=1}^k f_c(A_c)$ equals r. Consequently W_* equals the largest r for which $\alpha_r > 0$. Given $\mathrm{par}(g)$ the coefficients α_r can be computed in time $O(dkM)$. So, the time complexity of computing W_* is determined by the complexity of computing $\mathrm{par}(g)$. By Theorem 7, $\mathrm{par}(g)$ can be computed in $O(kn^2 2^n)$ operations with integers of length $O(n\log(nM'))$, where $M' = O(\beta^M) = O(n^{nM})$, implying the claimed bound.

In the case that some $f_c(S)$ is negative, shift the values of $f_1,\ldots,f_k$ by adding the constant M, operate on the range $[0, 2M]$, and finally subtract the total shift, kM. □

We can then use self-reducibility to find an optimal partition by choosing a "color" $c \in \{1, \ldots, k\}$ for each element in N, one by one, such that a suitably defined smaller subproblem still has the same maximum. This leads to only a k-fold increase in the total run time, as the smaller subproblems are exponentially easier. We omit the details.

Theorem 9. *The problem* MAX WEIGHTED PARTITION *can be solved in $O(k^2 n^2 2^n)$ operations with integers of length $O(Mn^2 \log n)$, thus, in $O^*(2^n \mu(M))$ time, where $\mu(\beta) = \beta \log \beta \log\log \beta$.*

5. Application to graph partitioning

Many classical graph partitioning problems take an undirected graph G with n vertices as the input. In MINIMUM GRAPH COLORING the task is to find a partition of the vertices into a minimum number k of disjoint sets $A_1, \ldots, A_k$ such that each A_c is an independent set of G. The minimum k is the chromatic number of G. The MAXIMUM DOMATIC PARTITION problem is to find a partition of the vertex set into a maximum number k of disjoint sets $A_1, \ldots, A_k$ such that each A_c is a dominating set of G. The maximum k is the domatic number of G. MINIMUM COLOR SUM is yet another problem where the task is find a partition $A_1, \ldots, A_k$ such that each A_c is an independent set of G and the weighted sum $|A_1| + 2|A_2| + \cdots + k|A_k|$ is minimized. Clearly, these problems can be efficiently reduced to MAX WEIGHTED PARTITION.

Theorem 10. *The problems* MINIMUM GRAPH COLORING, MAXIMUM DOMATIC PARTITION, *and* MINIMUM COLOR SUM *can be solved in $O^*(2^n)$ time.*

Some graph partitioning problems are defined on edge-weighted graphs. The MIN k-PARTITION problem is to find a partition $A_1, \ldots, A_k$ such that the total weight of edges within the sets A_c is minimized. Its dual, the MAX k-CUT problem, is to find a partition $A_1, \ldots, A_k$ such that the total weight of cutting edges (edges between two sets A_c and $A_{c'}$) is maximized. Again, these problems can be reduced to MAX WEIGHTED PARTITION.

Theorem 11. *The problems* MAX k-PARTITION *and* MAX k-CUT, *restricted to integer weights from the range $[-M, M]$, can be solved in $O^*(2^n \mu(M))$ time, where $\mu(\beta) = \beta \log \beta \log\log \beta$.*

These problems, of course, do not form an exhaustive list of problems that reduce to MAX WEIGHTED PARTITION or SUM WEIGHTED PARTITIONS. The reader can easily come up with other examples (e.g., from [14]; see also [7]).

6. Application to data clustering and decision graph learning

Let us consider Bayesian model-based clustering (see, e.g., [1, 12, 15, 23]). Let $y_1, \ldots, y_m$ be data points. We let A be a partition of the index set $\{1, \ldots, m\}$ into clusters $A_1, \ldots, A_k$. Each cluster A_c is associated with a component model parameterized by θ_c, such that the distribution of y_j given that j belongs to A_c is $p(y_j | \theta_c)$. Then the total probability of the data $\mathbf{y}$ is given by

$$p(\mathbf{y}) = \sum_A \prod_{c=1}^{k} f_c(A_c) = \mathrm{par}(f)\,,$$

where

$$f_c(A_c) := \rho_c(A_c) \int \prod_{j \in A_c} p(y_j | \theta_c) q_c(\theta_c) \mathrm{d}\theta_c\,, \qquad (3)$$

with some prior distributions ρ_c and q_c. Computing $p(\mathbf{y})$ for different numbers k is useful, e.g., for selecting a plausible number of clusters. For finding an optimal clustering, one usually considers the corresponding max–sum expression.

In classification and related tasks, one often seeks clusters in a feature space. Decision graphs generalize the popular decision tree models in that they can represent arbitrary (not only tree-structured) partitions of a feature space (e.g., [10]). A probabilistic decision graph defines a conditional distribution of a class variable y given a feature vector x. A decision graph specifies a partitioning of the feature space X into disjoint subsets $A_1, \ldots, A_k$. Each partition A_c is assigned a simple model parametrized by θ_c, such that the distribution of y given that x belongs to A_c is $p(y | x, \theta_c)$.

As many different decision graphs can specify the same partition A, we consider the problem of evaluating the overall goodness of a class of decision graph models in the light of m data points $(x_1, y_1), \ldots, (x_m, y_m)$. Suppose that X is a finite set of n elements. In a Bayesian approach [21], the partition and the parameters $\theta = (\theta_1, \ldots, \theta_k)$ are assigned prior distributions, $p(A) = \prod_c \rho_c(A_c)$ and $p(\theta) = \prod_c q_c(\theta_c)$. Then the conditional probability of $\mathbf{y} = (y_1, \ldots, y_m)$ given $\mathbf{x} = (x_1, \ldots, x_m)$ can be written as

$$p(\mathbf{y} | \mathbf{x}) = \sum_A \prod_{c=1}^{k} f_c(A_c) = \mathrm{par}(f)\,,$$

where

$$f_c(A_c) := \rho_c(A_c) \int \prod_{j=1 : x_j \in A_c}^{m} p(y_j | x_j, \theta_c) q_c(\theta_c) \mathrm{d}\theta_c\,. \quad (4)$$

The quantity $p(\mathbf{y} | \mathbf{x})$ is useful, for instance, in feature selection: one computes $p(\mathbf{y} | \mathbf{x})$ for different feature variables x

and selects the one that gives the largest value. If one is also interested in the best partition, the sum–product expression is replaced by the obvious max–sum version.

For both data clustering and decision graph learning the parametric models and the priors ρ_c and q_c are usually chosen such that the terms (3) and (4) can be efficiently computed [10, 21]. Then $p(\mathbf{y})$ and $p(\mathbf{y}|\mathbf{x})$ can be computed in $O^*(2^m)$ and $O^*(2^n)$ arithmetic operations, respectively, using the algorithm for SUM WEIGHTED PARTITIONS. However, it should be noted that then the weights are typically rational numbers rather than small integers; it is not clear whether fixed precision computation produces numerically stable results, or whether it is better to operate with large integers to get accurate results. Likewise, it seems that our algorithm for MAX WEIGHTED PARTITION is not suitable for solving the optimization versions of these problems.

7. Concluding remarks

While the presented algorithms for SUM WEIGHTED PARTITIONS and MAX WEIGHTED PARTITION are near optimal in a certain sense, there remain some interesting open questions. Can we reduce the time complexity of MINIMUM GRAPH COLORING to $o(2^n)$ by exploiting the very special properties of independent sets? For MAX WEIGHTED PARTITION, can we reduce the dependence of the maximum weight M from about linear to about logarithmic?

The memory requirement of the presented algorithms is exponential in n even when the input functions have a polynomial representation, e.g., as a weighted graph; this is because we use the fast Möbius transform algorithm. However, by replacing FMT by a straightforward algorithm, which for every set S simply goes through all its subsets in a brute-force manner, we obtain the result below. Can we find faster polynomial space algorithms? (For positive examples see Björklund and Husfeldt [7] in these proceedings.)

Theorem 12. MINIMUM GRAPH COLORING, MAXIMUM DOMATIC PARTITION, MINIMUM COLOR SUM, MIN k-PARTITION, *and* MAX k-CUT *can be solved in polynomial space in* $O^*(3^n\mu(M))$ *time, provided that the input weights are from the integer range* $[-M, M]$*; here* $\mu(\beta) = \beta \log \beta \log\log \beta$.

Acknowledgements

I am grateful to Heikki Mannila for valuable conversations on this work.

References

[1] J. D. Banfield and A. E. Raftery. Model-based Gaussian and non Gaussian clustering. *Biometrics*, 49:803–821, 1993.

[2] E. T. Bax. Inclusion and exclusion algorithm for the Hamiltonian path problem. *Information Processing Letters*, 47(4):203–207, 1993.

[3] E. T. Bax. Algorithms to count paths and cycles. *Information Processing Letters*, 52(5):249–252, 1994.

[4] E. T. Bax and J. Franklin. A finite-difference sieve to count paths and cycles by length. *Information Processing Letters*, 60(4):171–176, 1996.

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

[6] A. Björklund and T. Husfeldt. Exact algorithms for exact satisfiability and number of perfect matchings. In *Proceedings of the 33rd International Colloquium on Automata, Languages and Programming (ICALP 2006), LNCS 4051*, pages 548–559. Springer, 2006.

[7] A. Björklund and T. Husfeldt. Inclusion–exclusion algrithms for counting set partitions. In *Proceedings of the 47th IEEE Symposium on Foundations of Computer Science (FOCS 2006)*. 2006. these proceedings.

[8] H. L. Bodlaender and D. Kratsch. An exact algorithm for graph coloring with polynomial memory. Technical Report UU-CS-2006-015, Utrecht University, 2006.

[9] J. M. Byskov. *Exact Algorithms for Graph Colouring and Exact Satisfiability*. PhD thesis, University of Aarhus, August 2005.

[10] D. M. Chickering, D. Heckerman, and C. Meek. A Bayesian approach to learning Bayesian networks with local structure. In *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence (UAI 1997)*, pages 80–89, 1997.

[11] D. Eppstein. Small maximal independent sets and faster exact graph coloring. *J. Graph Algorithms Appl.*, 7(2):131–140, 2003.

[12] M. A. T. Figueiredo and A. K. Jain. Unsupervised learning of finite mixture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(3):381–396, 2002.

[13] F. V. Fomin, A. V. P. F. Grandoni, and A. A. Stepanov. Bounding the number of minimal dominating sets: A measure and conquer approach. In *Algorithms and Computation: 16th International Symposium (ISAAC 2005), LNCS 3827*, pages 573–582. Springer, 2006.

[14] M. Garey and D. Johnson. *Computers and Intractability - A Guide to the Theory of NP-completeness*. W. H. Freeman & Co., San Fransisco, CA, 1979.

[15] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323, 1999.

[16] R. M. Karp. Dynamic programming meets the principle of inclusion and exclusion. *Operations Research Letters*, 1(2):49–51, 1982.

[17] R. Kennes. Computational aspects of the Moebius transform of a graph. *IEEE Transactions on Systems, Man, and Cybernetics*, 22:201–223, 1991.

[18] M. Koivisto. *Sum–Product Algorithms for the Analysis of Genetic Risks*. PhD thesis, University of Helsinki, January 2004.

[19] M. Koivisto. Optimal 2-constraint satisfaction via sum–product algorithms. *Information Processing Letters*, 98(1):24–28, 2006.

[20] M. Koivisto and K. Sood. Exact Bayesian structure discovery in Bayesian networks. *Journal of Machine Learning Research*, 5:549–573, 2004.

[21] M. Koivisto and K. Sood. Computational aspects of Bayesian partition models. In *International Conference on Machine Learning (ICML 2005)*, pages 433–440, 2005.

[22] E. Lawler. A note on the complexity of the chromatic number problem. *Information Processing Letters*, 5:66–67, 1976.

[23] F. A. Quintana and P. L. Iglesias. Bayesian clustering and product partition models. *Journal of the Royal Statistical Society B*, 65(2):557–574, 2003.

[24] L. R. Rabiner. A tutorial on Hidden Markov Models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[25] T. Riege and J. Rothe. An exact 2.9416^n algorithm for the three domatic number problem. In *Proceedings of the 30th International Symposium on Mathematical Foundations of Computer Science (MFCS 2005)*, pages 733–744, 2005.

[26] T. Riege, J. Rothe, H. Spakowski, and M. Yamamoto. An improved exact algorithm for the domatic number problem. In *Proceedings of the Second IEEE International Conference on Information & Communication Technologies: From Theory to Applications (ICTTA)*, pages 1021–1022, 2006.

[27] H. J. Ryser. *Combinatorial Mathematics*. Carus Math. Monographs, No. 14, Wiley, New York, 1963.

[28] A. Schönhage and V. Strassen. Schnelle Multiplikation großer Zahlen. *Computing*, 7:281–292, 1971.

[29] R. Williams. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theoretical Computer Science*, 348(2–3):257–265, 2005.

[30] G. J. Woeginger. Exact algorithms for NP-hard problems: a survey. In *Combinatorial optimization – eureka, you shrink!, LNCS 3142*, pages 185–207. Springer, 2003.

[31] G. J. Woeginger. Space and time complexity of exact algorithms: Some open problems. In *Proceedings of the First International Workshop on Parametrized and Exact Computation (IWPEC 2004), LNCS 3162*, pages 281–290. Springer, 2004.