Covering and Packing in Linear Space*

Andreas Björklund[1], Thore Husfeldt[1,2], Petteri Kaski[3], and Mikko Koivisto[4]

[1] Lund University, Department of Computer Science,
P.O.Box 118, SE-22100 Lund, Sweden
andreas.bjorklund@yahoo.se,
[2] IT University of Copenhagen,
2300 Copenhagen S, Denmark
thore@itu.dk
[3] Helsinki Institute for Information Technology HIIT,
Aalto University, School of Science and Technology,
Department of Information and Computer Science
PO Box 15400, FI-00076 Aalto, Finland
petteri.kaski@tkk.fi
[4] Helsinki Institute for Information Technology HIIT,
Department of Computer Science, University of Helsinki,
P.O.Box 68, FI-00014 University of Helsinki, Finland
mikko.koivisto@cs.helsinki.fi

Abstract. Given a family of subsets of an n-element universe, the k-cover problem asks whether there are k sets in the family whose union contains the universe; in the k-packing problem the sets are required to be pairwise disjoint and their union contained in the universe. When the size of the family is exponential in n, the fastest known algorithms for these problems use inclusion–exclusion and fast zeta transform, taking time and space 2^n, up to a factor polynomial in n. Can one improve these bounds to only linear in the size of the family? Here, we answer the question in the affirmative regarding the space requirement, while not increasing the time requirement. Our key contribution is a new fast zeta transform that adapts its space usage to the support of the function to be transformed. Thus, for instance, the chromatic or domatic number of an n-vertex graph can be found in time within a polynomial factor of 2^n and space proportional to the number of maximal independent sets, $O(1.442^n)$, or minimal dominating sets, $O(1.716^n)$, respectively. Moreover, by exploiting some properties of independent sets, we reduce the space requirement for computing the chromatic polynomial to $O(1.292^n)$. Our algorithms also parallelize efficiently.

1 Introduction

Brute-force algorithms typically use lots of time but only little space. For instance, a straightforward algorithm for the traveling salesman problem (TSP)

* This research was supported in part by the Swedish Research Council, project "Exact Algorithms" (A.B., T.H.), and the Academy of Finland, Grants 117499 (P.K.) and 125637 (M.K.).

visits every possible permutation of the n cities, requiring about $n!$ computational steps and a storage for about n cities and two real numbers (in addition to the input). Designing faster algorithms is sometimes possible by trading space against time. Indeed, this is precisely what Bellman's [1, 2] and Held and Karp's [7] dynamic programming treatment of TSP does by tabulating partial solutions across all subsets of the n cities: both the runtime and the space requirement grow as 2^n. A similar story can be told about coloring n-vertex graphs, or set covering more generally, albeit the 2^n bounds were discovered only quite recently [3]. Reducing the space requirement for these problems appears challenging, and has been so far achieved only at the cost of increasing the running time [5, 9].

In this paper, we provide an input-sensitive characterization of the space–time tradeoff for the set cover problem. Regarding the time requirement, the best one can hope for is an upper bound linear in the size of the input, for all input must be read in the worst case. While no such lower bound is obvious for the space requirement, one may, again, regard a linear upper bound as a plausible goal. Since a set family over an n-set universe may contain an order of 2^n members, the known upper bounds are optimal in the worst case. For the current techniques, however, the $O^*(2^n)$ time and space bounds are tight even if the given set family is much smaller; throughout the paper the O^* notation hides a factor polynomial in n. Here, we show that the set cover problem can be solved in time $O^*(2^n)$ using space only about linear in the size of the set family. Applications to graph coloring and domatic partitioning yield space bounds of the form $O(C^n)$ with $C < 2$, as outlined in the sequel.

When making these claims we assume the general case where the set family is given explicitly in the input. It goes without saying that in many concrete problems, such as graph coloring or domatic partitioning, the set family is represented implicitly, for example in terms of a graph.

Our study actually concerns the counting variant of the set cover problem. A *k-cover* over a family $\mathcal{F}$ of subsets of an n-element universe U is a tuple of k members of $\mathcal{F}$ whose union contains U. Given $\mathcal{F}$ and k, the counting problem asks the number of k-covers over $\mathcal{F}$, denoted by $c_k(\mathcal{F})$. We start with the inclusion–exclusion formula [3],

$$c_k(\mathcal{F}) = \sum_{X \subseteq U} (-1)^{|U \setminus X|} a(X)^k \,, \tag{1}$$

where $a(X)$ is the number of subsets $Y \subseteq X$ that belong to $\mathcal{F}$. The key observation is that given $\mathcal{F}$, the numbers $a(X)$, for all $X \subseteq U$, can be listed (in some order) in time $O^*(2^n)$ and space $O^*(|\mathcal{F}|)$.

It is useful to formulate this result slightly more generally in terms of the zeta transform on the subset lattice, as follows. If f is a function from the subsets of U to a ring R, then the zeta transform of f, denoted as $f\zeta$, is defined by

$$f\zeta(X) = \sum_{Y \subseteq X} f(Y), \qquad X \subseteq U \,.$$

See Fig. 1 for an illustration of the zeta transform.

Theorem 1 *Suppose f vanishes outside $\mathcal{F}$ and the members of $\mathcal{F}$ can be listed in time $O^*(2^n)$ and space $O^*(|\mathcal{F}|)$. Then the values $f\zeta(X)$, for $X \subseteq U$, can be listed in time $O^*(2^n)$ and space $O^*(|\mathcal{F}|)$.*

This result, which we will prove in Sect. 3, allows us to easily extend the result for set covers to an analogous result for set partitions and set packings; a *k-partition* (*k-packing*) over $\mathcal{F}$ is a tuple of k pairwise disjoint member of $\mathcal{F}$ whose union equals (is contained in) the universe U. Thus, given Theorem 1, we have the following.

Theorem 2 *Let $\mathcal{F}$ be a family of subsets of an n-element universe U, and let k be an integer. Suppose $\mathcal{F}$ can be listed in time $O^*(2^n)$ and space $O^*(|\mathcal{F}|)$. Then the k-covers, k-packings, and k-partitions over $\mathcal{F}$ can be counted in time $O^*(2^n)$ and space $O^*(|\mathcal{F}|)$.*

We illustrate this result with some immediate implications to graph coloring and domatic partitioning. The *chromatic number* of a graph is the smallest integer k such that there exists a proper k-coloring of the graph, that is, a mapping σ from the vertices of the graph to $\{1, 2, \ldots, k\}$ such that $\sigma(u) \neq \sigma(v)$ if u and v are adjacent in the graph. To test if the chromatic number is k or smaller, it clearly suffices to count the covers of the vertices by k independent sets of the graph; a subset of vertices is an independent set if it does not contain two adjacent vertices. In fact, it suffices to count the covers of the vertices by k *maximal* independent sets; an independent set is maximal if it is not a subset of any other independent set. The maximal independent sets can be trivially listed in time $O^*(2^n)$ and space linear in their number. Because a graph with n vertices can have at most $3^{n/3} \approx 1.44225^n$ maximal independent sets [10], Theorem 2 gives us the following.

Corollary 1 *The chromatic number of a given n-vertex graph can be found in time $O^*(2^n)$ and space $O(1.443^n)$.*

Analogous reasoning applies to domatic partitioning. The *domatic number* of a graph is the largest integer k such that the vertices of the graph be partitioned into k pairwise disjoint dominating sets; a subset of vertices D is a dominating set if every vertex not in D is adjacent to at least one vertex in D. To test if the domatic number is k or larger, it suffices to count the k-packings over the dominating sets of the graph. Again, it actually suffices to count the k-packings over the *minimal* dominating sets; a dominating set is minimal if it contains no other dominating set. Because a graph with n vertices can have at most 1.716^n minimal dominating sets, which can be listed in time $O(1.716^n)$ [6], we have the following.

Corollary 2 *The domatic number of a given n-vertex graph can be found in time $O^*(2^n)$ and space $O(1.716^n)$.*

It should be noted that the computation of the chromatic or domatic number are, in essence, decision problems, even though the counting approach is crucial

for obtaining the reduced space requirement. If Theorem 2 is applied, for example, to the computation of the *chromatic polynomial*, which at k evaluates to the number of proper k-colorings, then the space bound $O^*(2^n)$ is tight, since it does not suffice to consider only maximal independent sets. In this light, we find it somewhat surprising that the chromatic polynomial can, however, be computed in much less space by using a variant of the linear-space zeta transform that exploits the special structure of independent sets. Indeed, in Sect. 5 we prove the following bound, which improves upon Corollary 1.

Theorem 3 *The chromatic polynomial of a given n-vertex graph can be found in time $O^*(2^n)$ and space $O(1.292^n)$.*

However, the algorithm behind Corollary 1 remains interesting for finding the chromatic number, as most graphs have a lot fewer maximal independent sets than the Moon–Moser bound $3^{n/3}$ predicts.

Apart from the space savings, our algorithms have also another feature that should be relevant for practical implementations: they admit efficient parallelization. It will be immediate from the descriptions in Sects. 3 and 4 that the algorithms can be executed in parallel on $O^*(2^n/S)$ processors, each using time and space $O^*(S)$, where S varies as given in the space bounds of Theorems 1 and 2 and Corollaries 1 and 2 ; for the chromatic polynomial some extra space is needed compared to Theorem 3: $O^*(2^{n/2}) = O(1.415^n)$ processors, each using time and space $O^*(2^{n/2})$. This capability for parallel computation is in sharp contrast to the previous algorithms [3], for which efficient parallelization to exponentially many processors seems not possible.

Remark. In the statements above and their proofs in the remainder sections, we ignore the polynomial factors for simplicity. We note, however, that the hidden factors are relatively small. For instance, in Theorem 1 the actual storage requirement is $O(|\mathcal{F}|n)$ bits and ring elements, assuming each member of $\mathcal{F}$ is represented naturally by n bits. Likewise, in Theorem 2 $O(|\mathcal{F}|n)$ bits suffice if we resort to space-efficient manipulation of the involved polynomials, that is, evaluation–interpolation, each evaluation modulo small relative primes and using the Chinese Remainder Theorem. We omit a more detailed consideration of these standard techniques in this paper.

2 Preliminaries

We adopt the following conventions. Let U be a universe of n elements. Denote by 2^U the set of all subsets of U. Let R be an algebraic ring. Let $f : 2^U \to R$ be a function. The *(down-)zeta transform* of f is the function $f\zeta : 2^U \to R$, defined for all $X \subseteq U$ by $f\zeta(X) = \sum_{Y \subseteq X} f(Y)$. The *up-zeta transform* of f is the function $f\zeta' : 2^U \to R$, defined for all $X \subseteq U$ by $f\zeta'(X) = \sum_{X \subseteq Y} f(Y)$.

We employ Iverson's bracket notation, that is, for a logical proposition P, we write $[P]$ to indicate a 1 if P is true and a 0 if P is false.

We recall that there is an algorithm, *the fast zeta transform* [8, 4], that computes the function $f\zeta$ from the function f in time and space $O^*(2^n)$, where we

assume that the arithmetic operations in the ring R take time $O^*(1)$ and each ring element takes $O^*(1)$ space.

The Fast Zeta Transform. Let the universe be $U = \{1, 2, \ldots, n\}$ and let $f : 2^U \to R$ be given as input. The algorithm pseudocode is as follows:

1. Set $f_0 \leftarrow f$.
2. For each $j = 1, 2, \ldots, n$ do:
   (a) For each $X \subseteq U$, set $f_j(X) \leftarrow [j \in X] f_{j-1}(X \setminus \{j\}) + f_{j-1}(X)$.
3. Give the output $f\zeta \leftarrow f_n$.

An analogous algorithm is easy to derive for the up-zeta transform.

3 The zeta transform in linear space and $O^*(2^n)$ time

This section proves Theorem 1.

Let us fix a bipartition of the universe U,

$$U = U_1 \cup U_2, \qquad U_1 \cap U_2 = \emptyset, \qquad |U_1| = n_1, \qquad |U_2| = n_2\,, \tag{2}$$

for integers n_1 and n_2 yet to be fixed. We now execute the following algorithm; see Fig. 2.

The Linear-Space Fast Zeta Transform. Let a family $\mathcal{F} \subseteq 2^U$ and a function $f : 2^U \to R$ that vanishes outside $\mathcal{F}$ be given as input. We execute the following algorithm to output $f\zeta(X)$ for each $X \subseteq U$, with comments delimited by double braces "{{" and "}}":

1. For each $X_1 \subseteq U_1$ do:
   (a) For each $Y_2 \subseteq U_2$, set $g(Y_2) \leftarrow 0$.
   {{ This step takes $O^*(2^{n_2})$ time and space. }}
   (b) For each $Y \in \mathcal{F}$, if $Y \cap U_1 \subseteq X_1$ then set $g(Y \cap U_2) \leftarrow g(Y \cap U_2) + f(Y)$.
   {{ This step takes $O^*(|\mathcal{F}|)$ time and $O^*(2^{n_2})$ space. }}
   (c) Compute $h \leftarrow g\zeta$ using the fast zeta transform on 2^{U_2}.
   {{ This step takes $O^*(2^{n_2})$ time and space. }}
   (d) For each $X_2 \subseteq U_2$, output the value $h(X_2)$ as the value $f\zeta(X_1 \cup X_2)$.

Note that the algorithm evaluates f only at $\mathcal{F}$. Moreover, we can iterate over the elements of $\mathcal{F}$ in arbitrary order.

We establish the correctness of the algorithm by analyzing the contents of the array h during each iteration of the loop over $X_1 \subseteq U_1$:

Lemma 1 *For each fixed $X_1 \subseteq U_1$, we have $h(X_2) = f\zeta(X_1 \cup X_2)$ for all $X_2 \subseteq U_2$.*

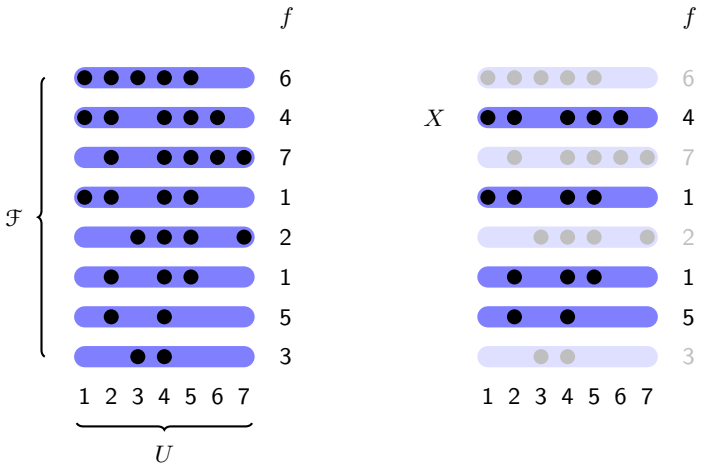


Fig. 1. Left: a set family $\mathcal{F}$ over $U = \{1, 2, \ldots, 7\}$, and values $f(Y) \neq 0$ for every $Y \in \mathcal{F}$. Right: For $X = \{1, 2, 4, 5, 6\}$, the value $f\zeta(X)$ is the sum over its subsets, $4 + 1 + 1 + 5 = 11$.

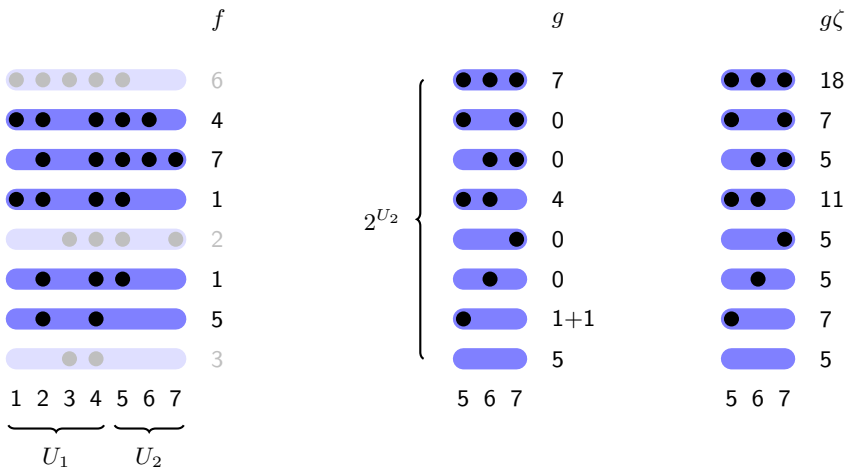


Fig. 2. The linear space zeta transform. U is partitioned into U_1 and U_2 with $|U_2| = \log |\mathcal{F}|$. We iterate over all $X_1 \subseteq U_1$; here we show $X_1 = \{1, 2, 4\}$. Left: We consider only the sets $Y \in \mathcal{F}$ with $Y \cap U_1 \subseteq X_1$. Middle: For each of these sets, add its value $f(Y)$ to $g(Y \cap U_2)$. Right: Compute the zeta transform $g\zeta$ on U_2. The result contains the values of $f\zeta$ for all X for which $X \cap U_1 = X_1$. For example, $f\zeta(\{1, 2, 4, 5, 6\}) = 11$. The central point of the analysis is that the most space-expensive operation, the exponential-space fast zeta transform, operates only on U_2, so the whole algorithm runs in space $2^{|U_2|} = |\mathcal{F}|$.

Proof. Expanding the assignments in the algorithm, we have

$$\begin{aligned} h(X_2) &= \sum_{Y_2 \subseteq X_2} g(Y_2) \\ &= \sum_{Y_2 \subseteq X_2} \sum_{Y \in \mathcal{F}} [Y \cap U_1 \subseteq X_1][Y \cap U_2 = Y_2] f(Y) \\ &= \sum_{Y \in \mathcal{F}} [Y \cap U_1 \subseteq X_1][Y \cap U_2 \subseteq X_2] f(Y) \\ &= \sum_{\substack{Y \in \mathcal{F} \\ Y \subseteq X_1 \cup X_2}} f(Y) \\ &= \sum_{Y \subseteq X_1 \cup X_2} f(Y) \\ &= f\zeta(X_1 \cup X_2)\,. \end{aligned}$$

□

Observe that the algorithm runs in $O^*(2^{n_1}(2^{n_2} + |\mathcal{F}|))$ time and $O^*(2^{n_2})$ space. We now fix the values n_1 and n_2 to $n_2 = \lceil \log_2 |\mathcal{F}| \rceil$ and $n_1 = n - n_2$. The algorithm thus runs in $O^*(2^n)$ time and $O^*(|\mathcal{F}|)$ space. Note also that because the computations are independent for each $X_1 \subseteq U_1$, they can be executed in parallel on $O^*(2^n/|\mathcal{F}|)$ processors.

4 Coverings, packings, and partitions

This section proves Theorem 2.

To compute the number of k-coverings we need to evaluate (1). Note that a equals $f\zeta$, where f is $\mathcal{F}$'s characteristic function, $f(Y) = [Y \in \mathcal{F}]$. By Theorem 1 we can list all values $a(X)$ for $X \subseteq U$ in some order within the desired time and space bounds; while doing so we accumulate their kth powers with the sign given by (1).

We turn to counting the k-partitions. The idea is to modify the function a above so that it controls the size of the counted members of the set family [3]. This can be handily formulated [4] by replacing $a(X)$ by a polynomial over an indeterminate z:

$$d_k(\mathcal{F}) = \sum_{X \subseteq U} (-1)^{|U \setminus X|} \Big(a_0(X) + a_1(X) z + \cdots + a_n(X) z^n \Big)^k ,$$

where the coefficient $a_j(X)$ is the number of subsets $Y \subseteq X$ that belong to $\mathcal{F}$ and are of size j. Now $d_k(\mathcal{F})$ is a polynomial whose coefficient of the monomial z^n is the number of k-partitions [4]. To evaluate this expression, we note that the polynomial $a(X)$ equals $g\zeta(X)$, where $g(Y) = [Y \in \mathcal{F}] z^{|Y|}$. Now, the linear-

space fast zeta transform (Theorem 1, Sect. 3) operates in a ring of polynomials and lists the polynomials $a(X)$ for all $X \subseteq U$ in the desired time and space.[5]

Finally, the number of k-packings can be viewed as the number of $(k+1)$-partitions with k members from $\mathcal{F}$, the $(k+1)$th member being an arbirary subset of U. The expression corresponding to (1) becomes

$$p_k(\mathcal{F}) = \sum_{X \subseteq U} (-1)^{|U \setminus X|} (1+z)^{|X|} \Bigg(\sum_{j=0}^{n} a_j(X) z^j \Bigg)^k ,$$

and the coefficient of z^n gives the number of k-packings.

5 The chromatic polynomial in time $O^*(2^n)$ and space $O(1.2916^n)$

This section proves Theorem 3.

Let G be a graph with vertex set V, $|V| = n$. For a positive integer k, denote by $\chi_G(k)$ the number of proper k-colorings of the vertices of G, that is, the number of mappings σ from V to $\{1, 2, \ldots, k\}$ so that $\sigma(u) \neq \sigma(v)$ holds whenever u and v are adjacent in G. It is well known that the integer function $\chi_G(t)$ admits representation as a polynomial of degree n with integer coefficients. In particular, $\chi_G(t)$ is called the *chromatic polynomial* of G.

To compute $\chi_G(t)$ from a given G, it suffices to evaluate $\chi_G(k)$ for $n+1$ distinct positive integers k and then recover the polynomial in t by interpolation.

Thus, our task reduces to counting the number of proper k-coloring of G. Equivalently, our task is to count the number of ordered partitions (with empty parts permitted) of the vertices of G into k independent sets. Put otherwise, it suffices to compute for every $X \subseteq V$ the z-polynomial

$$i(X) = \sum_{Y \subseteq X} z^{|Y|} [Y \text{ is independent in } G] ,$$

which enables us to recover the number of k-colorings of G as the coefficient of z^n in the polynomial

$$r = \sum_{X \subseteq V} (-1)^{n-|X|} i(X)^k .$$

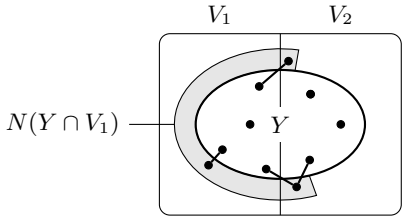
Our improved space requirement stems from an algorithm that evaluates the polynomials $i(X)$ in two parts. To this end, partition the vertex set V into two disjoint sets, V_1 and V_2, with $|V_1| = n_1$ and $|V_2| = n_2$. Let $n_1 = \lceil n(\log 2)/(\log 3) \rceil$ and $n_2 = n - n_1$. Observe that $2^{n_2} = O(1.29153^n)$.

For a set $Y \subseteq V$, denote by $N(Y) \subseteq V$ the set of vertices that are adjacent to at least one vertex in Y.

[5] To save some polynomial factors in both time and space, the arithmetic should actually not be carried out directly with polynomials. Instead, the polynomials can be evaluated at sufficiently many small integers, and finally the coefficients of $d_k(\mathcal{F})$ are recovered via interpolation and the Chinese Remainder Theorem.

Our strategy is to count each independent set Y via its parts $Y \cap V_1$ and $Y \cap V_2$ using the following elementary observation:

Lemma 2 *A set of vertices $Y \subseteq V$ is independent in G if and only if $Y \cap V_2 \subseteq V_2 \setminus N(Y \cap V_1)$ and both $Y \cap V_1$ and $Y \cap V_2$ are independent in G.*



In particular, we observe that $Y \cap V_2$ is oblivious to $Y \cap V_1$ except for the requirement $Y \cap V_2 \subseteq V_2 \setminus N(Y \cap V_1)$. Thus, in terms of space usage, we can collect together all eligible independent sets $Y_1 \subseteq V_1$ that share the set $Z_2 = V_2 \setminus N(Y_1)$. The function h in the following algorithm collects such sets Y_1 by their size $|Y_1|$ into the coefficients of the polynomial $h(Z_2)$. The function h requires only $O^*(2^{n_2})$ space, which in effect enables us to compute $i(X)$ for all $X \subseteq V$ in time $O^*(2^n)$ and less space than the number of independent sets in G. That is, using Lemma 2 we can beat the bound in Theorem 1.

In precise terms, we execute the following algorithm to obtain the number of proper k-colorings of G:

1. Set $r \leftarrow 0$.
2. For each $X_1 \subseteq V_1$ do:
   (a) For each $Z_2 \subseteq V_2$, set $h(Z_2) \leftarrow 0$.
   {{ The array h takes space $O^*(2^{n_2})$. }}
   (b) For each $Y_1 \subseteq X_1$, set

$$h(V_2 \setminus N(Y_1)) \leftarrow h(V_2 \setminus N(Y_1)) + z^{|Y_1|}[Y_1 \text{ is independent in } G].$$

   {{ This loop takes time $O^*(2^{|X_1|})$. }}
   (c) For each $Y_2 \subseteq V_2$, set $\ell(Y_2) \leftarrow z^{|Y_2|}[Y_2$ is independent in $G]$.
   {{ The array ℓ takes space $O^*(2^{n_2})$. }}
   (d) Set $g \leftarrow (h\zeta') \cdot \ell$.
   {{ Here "$\cdot$" denotes the elementwise product of two arrays.
   Time and space $O^*(2^{n_2})$ using the fast up-zeta transform on 2^{V_2}. }}
   (e) Set $j \leftarrow g\zeta$.
   {{ Time and space $O^*(2^{n_2})$ using the fast zeta transform on 2^{V_2}. }}
   (f) For each $X_2 \subseteq V_2$, set $r \leftarrow r + (-1)^{n-|X_1|-|X_2|} j(X_2)^k$.
3. Return the coefficient of z^n in r.

We establish the correctness of the algorithm by analyzing the contents of the array i during each iteration of the loop over $X_1 \subseteq V_1$:

Lemma 3 *For each fixed $X_1 \subseteq V_1$, we have $j(X_2) = i(X_1 \cup X_2)$ for all $X_2 \subseteq V_2$.*

Proof. Expanding the assignments in the algorithm and using Lemma 2, we have

$$\begin{aligned}
j(X_2) &= \sum_{Y_2 \subseteq X_2} g(Y_2) \\
&= \sum_{Y_2 \subseteq X_2} h\zeta'(Y_2)\ell(Y_2) \\
&= \sum_{Y_2 \subseteq X_2} \sum_{Y_2 \subseteq Z_2} h(Z_2)\ell(Y_2) \\
&= \sum_{Y_2 \subseteq X_2} \sum_{Y_2 \subseteq Z_2} \sum_{Y_1 \subseteq X_1} z^{|Y_1|}[Y_1 \text{ is independent in } G \text{ and } Z_2 = V_2 \setminus N(Y_1)] \\
&\qquad\qquad \times z^{|Y_2|}[Y_2 \text{ is independent in } G] \\
&= \sum_{Y_1 \subseteq X_1} \sum_{Y_2 \subseteq X_2} z^{|Y_1 \cup Y_2|}[Y_1 \cup Y_2 \text{ is independent in } G] \\
&= i(X_1 \cup X_2)\,.
\end{aligned}$$

□

To analyze the running time, we observe that the total running time (over all iterations $X_1 \subseteq V_1$) spent in Step 2(b) of the algorithm is within a polynomial factor of

$$\sum_{X_1 \subseteq V_1} 2^{|X_1|} = \sum_{j=0}^{n_1} \binom{n_1}{j} 2^j = 3^{n_1} = O(2^n)\,.$$

Thus, the total running time is $O^*(2^n)$, using space $O^*(2^{n_2}) = O(1.2916^n)$.

Because the computations are independent for each $X_1 \subseteq V_1$, they can be executed in parallel on $O^*(2^{n_1}) = O(1.5486^n)$ processors. While the space requirement per processor is $O(1.2916^n)$ and the total running time remains $O^*(2^n)$, the time requirement per processor varies, ranging from $O(1.5486^n)$ to $O(1.2916^n)$. This bottleneck can be removed by taking a more balanced scheme with n_1 and n_2 about equal, yielding time and space $O^*(2^{n/2})$ for each of the $O^*(2^{n/2})$ processors.

References

1. Bellman, R.: Combinatorial Processes and Dynamic Programming. In: Bellman, R., Hall, M., Jr. (eds.) Combinatorial Analysis, Proceedings of Symposia in Applied Mathematics 10, pp. 217–249. American Mathematical Society (1960)
2. Bellman, R.: Dynamic Programming Treatment of the Travelling Salesman Problem. J. Assoc. Comput. Mach. 9, 61–63 (1962)
3. Björklund, A., Husfeldt, T., Koivisto, M.: Set partitioning via inclusion–exclusion. SIAM J. Comput. 39, Special Issue for FOCS 2006, 546–563 (2009)

4. Björklund, A., Husfeldt, T., Kaski, P., Koivisto, M.: Fourier meets Möbius: fast subset convolution. In: 39th ACM Symposium on Theory of Computing (STOC 2007), pp. 67–74. ACM Press (2007)
5. Björklund, A., Husfeldt, T., Kaski, P., Koivisto, M.: Computing the Tutte polynomial in vertex-exponential time. In: 49th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2008), pp. 677–686. IEEE Computer Society (2008)
6. Fomin, F.V., Grandoni, F., Pyatkin, A., Stepanov, A.: Combinatorial bounds via measure and conquer: Bounding minimal dominating sets and applications. ACM Transactions on Algorithms 5, 1–17 (2008)
7. Held, M., Karp, R.M.: A Dynamic Programming Approach to Sequencing Problems. J. Soc. Indust. Appl. Math. 10, 196–210 (1962)
8. Kennes, R.: Computational aspects of the Moebius transform of a graph. IEEE Transactions on Systems, Man, and Cybernetics 22, 201–223 (1991)
9. Koivisto, M., Parviainen, P: A space-time tradeoff for permutation problems. In: 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2010), pp. 484–492. (2010)
10. Moon, J.W., Moser, L.: On cliques in graphs. Israel Journal of Mathematics 3, 23–28 (1965)