Below All Subsets for Some Permutational Counting Problems*

Andreas Björklund

Department of Computer Science, Lund University, Lund, Sweden
andreas.bjorklund@yahoo.se

Abstract

We show that the two problems of computing the permanent of an $n \times n$ matrix of poly(n)-bit integers and counting the number of Hamiltonian cycles in a directed n-vertex multigraph with exp(poly(n)) edges can be reduced to relatively few smaller instances of themselves. In effect we derive the first deterministic algorithms for these two problems that run in $o(2^n)$ time in the worst case. Classic poly(n)2^n time algorithms for the two problems have been known since the early 1960's. Our algorithms run in $2^{n-\Omega(\sqrt{n/\log n})}$ time.

1998 ACM Subject Classification G.2.2 Graph Theory

Keywords and phrases Matrix Permanent, Hamiltonian Cycles, Asymmetric TSP

Digital Object Identifier 10.4230/LIPIcs.SWAT.2016.17

1 Introduction

We show that two well-known computationally hard counting problems defined over permutations, admit a strong form of self-reducibility. The problems are:

- Permanent: Given an $n \times n$ matrix M with poly(n)–bit integer elements, compute $\text{per}(M) = \sum_{\sigma \in S_n} \prod_i M_{i,\sigma(i)}$ where S_n is the set of all permutations on n elements.
- HamCycles: Given an n-vertex directed multigraph, compute its number of Hamiltonian cycles, i.e. the number of non-crossing spanning cycles.

For both problems, we show that the solution to an instance of size parameter n can be reduced to a weighted sum of the solutions to poly(n)2^{n-k} instances of size parameter $k < n$ of the same problem. Moreover, this reduction can be carried out in time polynomial in n per generated instance. We use this new relation to derive deterministic $2^{n-\Omega(\sqrt{n/\log n})}$ time algorithms for both Permanent and HamCycles. As a direct corollary we obtain an $Mn^2 2^{n-\Omega(\sqrt{n/log(Mn)})} + M^2 n^4$ time algorithm for Asymmetric TSP in graphs with integer arc weights in $[0, \ldots, M]$.

This is as far as the author knows the first deterministic algorithms that compute these quantities faster than explicitly inspecting at least a constant fraction of all subsets of an n-element set. In particular, no $o(2^n)$ time algorithms were previously known.

Our techniques here are elementary and the presentation is more-or-less self-contained. The main components are inclusion–exclusion counting, polynomial interpolation, and the Chinese remainder theorem. The speed-up is obtained through tabulation.

The two problems have well-known poly(n)2^n time algorithms: Ryser's algorithm based on inclusion–exclusion for the permanent [14] from 1963, and a simple variation of Bellman, Held

* This research was supported in part by the Swedish Research Council grant VR 2012-4730 Exact Exponential–Time Algorithms.

15th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT 2016).
Editor: Rasmus Pagh; Article No. 17; pp. 17:1–17:11
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

and Karp's dynamic programming algorithm for TSP [3, 9] from 1962. Later a polynomial space inclusion–exclusion algorithm in the same spirit as Ryser's for counting Hamiltonian cycles with the same running time was found [12] in 1977 (and was rediscovered twice [10, 1]).

The question of existence of $O((2-\Omega(1))^n)$ time algorithms for the two problems are well-known open problems. In comparison the recent $O(1.657^n)$ time algorithm for Hamiltonian cycle [5] is randomized, only works for undirected graphs, and cannot even approximate the number of solutions. Recently, Cygan et al. [8] gave an algorithm for Hamiltonicity detection in bipartite directed graphs in $O(1.888^n)$ time. [7] presented an algorithm for computing the parity of the number of Hamiltonian cycles in $O(1.619^n)$ time, and [6] showed that one could reduce Hamiltonicity detection in graphs with few Hamiltonian cycles to the parity problem, and thereby obtained $o(2^n)$ time algorithms when the instance is known to have few solutions. Still, not only have there been no deterministic algorithms running in $o(2^n)$ worst case time for the counting problems, it was not even known how to detect a Hamiltonian cycle in a directed n-vertex graph that fast, probabilistic algorithms included. Nor was it known how to compute the permanent of an $n \times n$ $0-1$ matrix deterministically in $o(2^n)$ time.

Moreover, Knuth asks in exercise 4.6.4.11. [M46] in [11] if it is possible to compute a real $n \times n$-matrix permanent with less than 2^n arithmetic operations. We note that reals of bounded precision can be modeled by large integers, so our algorithm here works also for them. However, a table look-up is not an arithmetic operation, so our algorithm is not exactly what Knuth solicited.

The one general previous improvement over poly$(n)2^n$ time for any of the two exact counting problems we are aware of is the $2^{n-\Omega(n^{1/3}\log n)}$ *expected* time algorithm for the $0-1$ matrix version of Permanent by Bax and Franklin [2]. Their technique can be extended to work with $O(1)$-bit integers, but probably not beyond that. In contrast, besides being faster and in deterministic time, our algorithm handles poly(n)-bit integers, including negative ones.

The two known poly$(n)2^n$ time algorithms for the problems based on the principle of inclusion–exclusion, Ryser's [14] and Kohn et al.'s [12] respectively, both use only polynomial space. It is indeed very natural to ask if employing the unexploited resource of using almost as much space as time wouldn't lead to faster algorithms. The problem though with the known approaches above is that there is no evident candidate for what to tabulate. They both sum over too large and typically different combinatorial objects. In the case of Ryser's permanent it is an n-element vector, and in Kohn et al.'s Hamiltonian cycles it is an induced graph on $n/2$ vertices on average.

The key insight here enabling a speed-up from tabulation is that the two problems admit a mapping from the original instances down to a linear combination of not too many much smaller ones. So small in fact that they are bound to coincide, making tabulation worthwhile.

1.1 Overview of the Technique

Consider the Permanent case, the HamCycles is similar. The speed-up is obtained in a series of steps. First we let $k = c\sqrt{n/\log n}$ for a constant c depending on the largest absolute element in the input matrix. Next we employ the existence part of the Chinese remainder theorem to bring matrix elements down to $d \log n$ bits each for some d. That is, we compute the permanent modulo small primes p of size polynomial in n. For each such prime p, we construct poly$(n)2^{n-k}$ $k \times k$-matrices such that the permanent of the original one is equal to the sum of weighted permanents of all the matrices constructed. This reduction is in itself a two step procedure composed of a reduction to an inclusion–exclusion formula over polynomial matrices, accompanied by polynomial interpolation. We count the occurrences of each of the smaller matrices in a table. Next we compute the permanent once for each of the different smaller matrices appearing in the sum using the classic $\text{poly}(k)2^k$ time algorithm. We note that there are at most $n^{dk^2} \ll 2^n$ different such matrices of size $k \times k$. The original instance permanent is then computed as a linear combination of all the tabulated matrices' permanent values. Finally, the results for all considered primes p are assembled via the constructive part of the Chinese remainder theorem.

1.2 Organization

In Section 2 we give a self-contained description of the self-reduction, anticipating that this part of the results may be of independent interest. The main results, the $o(2^n)$ algorithms for the two counting problems, are described in Section 3.

2 The Self-Reduction

The two problems Permanent and HamCycles are closely related. At a first glance it appears that the first asks about a property of matrices and the second about graphs, but they can be expressed in the same language. For the purpose of this paper, we will redefine both the Permanent and the HamCycles problem in terms of arc-weighted complete directed graphs to stress their similarity. In the remainder of this paper, the graph $G_n = (V, A)$ will denote the complete directed graph on n vertices V labelled 1 through n.

The set of all permutations on n elements, denoted by S_n, can naturally be partitioned after the number of cycles the permutation describes: A permutation $\sigma \in S_n$ can be interpreted as a directed graph on n vertices, labeled 1 through n, with the arcs $i, \sigma(i)$ for all i. Every vertex has exactly one outgoing and one incoming arc, i.e. the graph is a set of disjoint cycles covering the vertices. We will with S_n^1 denote the subset of S_n of permutations consisting of exactly one such cycle. Hence the permanent can be viewed upon as a sum over cycle covers of a graph, and the Hamiltonian cycles a sum over cycle covers consisting of just one cycle.

In the following it will make sense to be explicitly clear about what ring the computation is over. Thus we extend our problem definitions to:

▶ **Definition 1** (R-Permanent). Given a complete directed graph $G_n = (V, A)$ and a function $f : A \to R$ mapping the arcs to some ring R, the *permanent* of (G, f) over R, denoted $\text{per}(G, f)$, is $\sum_{\sigma \in S_n} \prod_{i=1}^{n} f(i\sigma(i))$.

▶ **Definition 2** (R-HamCycles). Given a complete directed graph $G_n = (V, A)$ and a function $f : A \to R$ mapping the arcs to some ring R, the *hamcycles* of (G, f) over R, denoted $\text{hc}(G, f)$, is $\sum_{\sigma \in S_n^1} \prod_{i=1}^{n} f(i\sigma(i))$.

In the remainder of this section we will prove the following two lemmas:

▶ **Lemma 3.** *Given an instance (G_n, f) to F-Permanent with f mapping arcs to a field F having at least $(n-k)n+1$ elements, and a positive integer $k < n$, one can compute $m = ((n-k)n+1)2^{n-k}$ instances $I_i = (G_k, f_i)$ to F-Permanent and constants $a_i \in F$ for $i = 1, \ldots, m$, so that*

$$\text{per}(G_n, f) = \sum_{i=1}^{m} a_i \, \text{per}(G_k, f_i) \, .$$

Moreover, the constructed smaller instances and constants can be produced in polynomial in n arithmetic operations $+$ and $*$ over F per instance.

▶ **Lemma 4.** *Given an instance (G_n, f) to F-HamCycles with f mapping arcs to a field F having at least $(n-k)k+1$ elements, and a positive integer $k < n$, one can compute $m = ((n-k)k+1)2^{n-k}$ instances $I_i = (G_k, f_i)$ to F-HamCycles and constants $a_i \in F$ for $i = 1, \ldots, m$, so that*

$$\mathrm{hc}(G_n, f) = \sum_{i=1}^{m} a_i \,\mathrm{hc}(G_k, f_i)\,.$$

Moreover, the constructed smaller instances and constants can be produced in polynomial in n arithmetic operations $+$ and $*$ over F per instance.

2.1 Preliminaries

In a complete directed graph G_n a *walk* of length l is a sequence of not necessarily distinct vertices $(v_0, v_1, \ldots, v_l)$. If $v_0 = v_l$ we say that the walk is a *closed* walk. For a field F and an indeterminate r, we denote by $F[r]$ the polynomial ring over F of polynomials in r with coefficients from F. For a polynomial $p(r) \in F[r]$ we denote by $[r^n]p(r)$ the coefficient of the monomial r^n in $p(r)$.

2.2 Step 1. Inclusion–exclusion

Consider an instance (G_n, f) to either F-Permanent or F-HamCycles for some field F. We fix a subset $K \subseteq V$ of the vertices of size $|K| = k$, called the *kernel* of the reduction. Without loss of generality, we let K be the vertices labeled by $1, 2, \ldots, k$, and hence $V - K$ be the vertices labelled by $k+1, k+2, \ldots, n$.

Our resulting instances will all be over the kernel K, i.e. embedded on the graph G_k. The central idea is to represent the parts of a cycle cover covering the vertices $V - K$, by arcs in G_k between the entry and exit points of the cycles in K. This approach of representing parts of a cycle cover outside a small subgraph by encoding them on the arcs of the subgraph was previously used by the author both in [4] and [5]. The novelty here, is the observation that these reductions can be seen as a mapping to a low degree univariate polynomial, that in step 2 in the next section will be efficiently brought back to the original field.

In this first step, we construct one instance per subset of $V - K$, and use the principle of inclusion–exclusion to relate them to the original instance. The resulting instances will not be over the original field F though. Instead the function f giving weights to the arcs will assign polynomials in one rank indeterminate r to them.

First we define the ranked walks in a vertex subset X. The degree of the indeterminate r counts the number of vertices visited along the walk. For any vertices $u, v \in X \subseteq V$ we let $W_{X,k}(u, v)$ be the ranked walks between vertices u and v visiting k vertices in X. We set

$$W_{X,k}(u,v) = \begin{cases} \sum_{w \in X} W_{X,k-1}(u,w) f(w,v) r & : k > 0 \\ 1 & : k = 0 \wedge u = v \\ 0 & : k = 0 \wedge u \neq v. \end{cases} \tag{1}$$

The ranked walks will be used to make sure all vertices outside the kernel K are visited by the cycle covers in the Permanent case and the Hamiltonian cycles in the HamCycles case. The principle of inclusion–exclusion makes sure crossing walks are cancelled. Since the HamCycles case is somewhat easier technically, we describe it first.

2.2.1 Inclusion–exclusion for HamCycles

We will construct instances of $F[r]$-HamCycles defined on $G_k = (K, A_K)$. We let $f_X : A_K \to F[r]$ for $X \subseteq V - K$ be defined for all $u, v \in K$ as follows

$$f_X(uv) = f(uv) + \sum_{w,z \in X} f(uw) \left(\sum_{i=0}^{n-k-1} W_{X,i}(w,z) \right) f(zv) \cdot r. \tag{2}$$

The point is that $f_X(uv)$ encodes all possible choices between either staying in K by choosing the arc uv directly or taking a detour through $V - K$ consisting of $1, 2, \ldots, n-k$ vertices starting in u and ending in v.

▶ **Lemma 5.** *With G_n, f, K, k, G_k, f_X as above it holds that*

$$\mathrm{hc}(G_n, f) = [r^{n-k}] \sum_{X \subseteq V-K} (-1)^{|V-K-X|} \mathrm{hc}(G_k, f_X).$$

Proof. By the definition of F-HamCycles Def. 2, we have

$$\mathrm{hc}(G_k, f_X) = \sum_{\sigma \in S_k^1} \prod_{i=1}^{k} f_X(i\sigma(i)).$$

Expanding f_X via Eq. 2, we get

$$\mathrm{hc}(G_k, f_X) = \sum_{\sigma \in S_k^1} \prod_{i=1}^{k} \left(f(i\sigma(i)) + \sum_{l=1}^{n-k} r^l \sum_{v_1,\ldots,v_l \in X} f(iv_1) \left(\prod_{j=1}^{l-1} f(v_j v_{j+1}) \right) f(v_l \sigma(i)) \right).$$

From the formula above, we see that $[r^{n-k}]hc(G_k, f_X)$ is a sum with terms $\prod_{i=1}^{n} f(v_i v_{i+1})$ for each closed walk $(v_1, v_2, \ldots, v_{n+1})$ with $v_{n+1} = v_1$ where

1. Exactly $n-k$ of $v_1, \ldots, v_n$ belong to X, and
2. Each vertex in K occurs exactly once in $v_1, \ldots, v_n$.

In the inclusion–exclusion summation over $X \subseteq V - K$,

$$\mathrm{hc}(G_n, f) = \sum_{X \subseteq V-K} (-1)^{|V-K-X|} [r^{n-k}] \mathrm{hc}(G_k, f_X),$$

each walk that crosses itself, i.e. has $v_i = v_j$ for some $i < j \le n$, will be counted an even number of times. Moreover, exactly half of these times it will be added to the sum and the other half it will be subtracted, thereby canceling in the sum. To see why, let $Y = \{v_i | v_i \in V - K\}$ for a crossing walk. Clearly $Y \subset V - K$ since there are precisely $n-k$ vertices from $V - K$ on every contributing walk, and when one occurs at least twice there must be another one that is missing. Since among the subsets Z fulfilling $Y \subseteq Z \subseteq V - K$ there are as many even sized subsets as odd ones the claim follows. Contributing walks that do not cross themselves however, i.e. are Hamiltonian cycles in G, will only be counted once, for $X = V - K$. ◀

2.2.2 Inclusion–exclusion for Permanent

In addition to the ranked walks in $V - K$ we also need to keep track of ranked cycles in $V - K$ for the Permanent. We want to sum over all cycle covers of the input graph G and unlike the HamCycles case we may have vertices in $V - K$ disconnected from K in a cycle cover. Remember that the vertices in V are labelled $1, 2, \ldots, n$ and associate the natural ordering $<$ of them. We need to define cycles in a cycle cover so that they receive a unique identifier to avoid double counting in our polynomial identity. To this end, we use that every cycle has a minimum vertex under the ordering to define the ranked closed walks *anchored* at $s \in X$ as

$$C_X(s) = 1 + \sum_{i=1}^{n-k} W_{X_{\geq s},i}(s,s)\,. \tag{3}$$

where $X_{\geq s} = \{v | s \leq v \in X\}$, i.e. all vertices in X equal to or larger than s. The cycles anchored at s represents all cycles of length $1, 2, \ldots, n-k$ in $V - K$ where s is the smallest vertex on the cycle. Note in particular that self-loops through s are also included in the sum. The 1 is in the definition of Eq. 3 to take into account the possibility that no cycle is anchored at s in a contributing cycle cover.

▶ **Lemma 6.** *With G_n, f, K, k, G_k, f_X as above it holds that*

$$\mathrm{per}(G_n, f) = [r^{n-k}] \sum_{X \subseteq V-K} (-1)^{|V-K-X|}\, \mathrm{per}(G_k, f_X) \prod_{s \in X} C_X(s)\,.$$

Proof. By the definition of F-Permanent Def. 1, we have

$$\mathrm{per}(G_k, f_X) \prod_{s \in X} C_X(s) = \sum_{\sigma \in S_k} \prod_{j=1}^{k} f_X(j\sigma(j)) \prod_{i=k+1}^{n} C_X(i)\,.$$

Expanding C_X via Eq. 3 and f_X via Eq. 2, we get

$$\mathrm{per}(G_k, f_X) \prod_{s \in X} C_X(s) =$$

$$\sum_{\sigma \in S_k} \prod_{i=1}^{k} \left(f(i\sigma(i)) + \sum_{l=1}^{n-k} r^l \sum_{v_1,\ldots,v_l \in X} f(iv_1) \left(\prod_{j=1}^{l-1} f(v_j v_{j+1}) \right) f(v_l \sigma(i)) \right)$$

$$\cdot \prod_{i=k+1}^{n} \left(1 + \sum_{l=1}^{n-k} r^l \sum_{\substack{v_1,\ldots,v_l \in X_{\geq i} \\ i=v_1}} f(v_l v_1) \prod_{j=1}^{l-1} f(v_j v_{j+1}) \right)\,.$$

Expanding the formula above into a sum–product formula by identifying terms, we see that

$$[r^{n-k}]\, \mathrm{per}(G_k, f_X) \prod_{s \in X} C_X(s)\,,$$

is a sum over contributions $\prod_{i=1}^{l} \prod_{uv \in O_i} f(uv)$, for $1 \leq l \leq n$ closed l-long walks $O_i = (v_{i,1}, \ldots, v_{i,m_l}, v_{i,m_l+1})$ with $v_{i,1} = v_{i,m_l+1}$ and $\sum_{i=1}^{l} m_i = n$ where

1. Exactly $n-k$ of the $v_{i,j}$ for $1 \leq i \leq n, 1 \leq j \leq m_i$ belong to X, and
2. Each vertex in K occurs exactly once in the closed walks $O_i, 1 \leq i \leq l$.

In the inclusion–exclusion summation over $X \subseteq V - K$,

$$\mathrm{per}(G_n, f) = \sum_{X \subseteq V-K} (-1)^{|V-K-X|} [r^{n-k}] \mathrm{per}(G_k, f_X) \prod_{s \in X} C_X(s)\,,$$

each set of closed walks $\{O_i\}$ that crosses itself, i.e. has $v_{i1,j1} = v_{i2,j2}$ for some $i1 \neq i2 \vee j1 \neq j2$, will be counted an even number of times. Moreover, exactly half of these times it will be added to the sum and the other half it will be subtracted, thereby canceling in the sum. To see why, again let $Y = \{v_{i,j} | v_{i,j} \in V - K\}$ for a set of closed walks with a crossing. Clearly $Y \subset V - K$ since there are precisely $n - k$ vertices from $V - K$ on every contributing set of closed walks, and when one occurs at least twice there must be another one that is missing. Since among the subsets Z fulfilling $Y \subseteq Z \subseteq V - K$ there are as many even sized subsets as odd ones the claim follows. Contributing sets of closed walks that do not cross themselves, i.e. are cycle covers in G, will only be counted once, for $X = V - K$. ◀

2.3 Step 2. Polynomial Interpolation

In the previous section we related the permanent and the Hamiltonian cycles of an arc weighted graph to smaller graphs with weights over a polynomial ring. We want to bring the small instances to map arcs to the original ring to complete the self-reduction. Unfortunately, we are only able to do this if the original ring is a field, and one that has at least polynomially many elements in the original instance size parameter. In particular, we need the following well-known result:

▶ **Lemma 7** (Lagrange interpolation). *For any set of pairs $\{(r_i, s_i)\}$ with distinct r_i's and $r_i, s_i \in F$ for $i = 1, \ldots, k+1$ where F is a field on at least $k+1$ elements, there is a unique polynomial $p(r)$ in $F[r]$ of degree at most k such that $p(r_i) = s_i$ for all i. Moreover, the polynomial is given by*

$$p(r) = \sum_{i=1}^{k+1} s_i \prod_{j \neq i} \frac{r - r_j}{r_i - r_j}\,.$$

Specifically, consider an instance (G, f) to F-HamCycles. Via Lemma 5 we see that $\mathrm{hc}(G)$ is related to a coefficient in a polynomial sum of many smaller instances (G_k, f_X) to $F[r]$-HamCycles. We use here that if we know the result in enough points over F we can reconstruct the polynomial via interpolation.

▶ **Lemma 8.** *For every polynomial term $\mathrm{hc}(G_k, f_X)$ in the outer sum in Lemma 5, it is possible to compute $(n-k)k+1$ instances (G_k, f_i) for $i = 1, \ldots, (n-k)k+1$ to the F-HamCycles on k vertices, and constants $a_i \in F$ for $i = 1, \ldots, (n-k)k+1$ so that*

$$[r^{n-k}]\,\mathrm{hc}(G_k, f_X) = \sum_{j=1}^{(n-k)k+1} a_j\,\mathrm{hc}(G_k, f_j)\,.$$

Proof. Each entry in the codomain of f_X has degree $n-k$ in r by definition of the ranked walks and the definition of f_X in Eq. 2. Since $\mathrm{hc}(G_k, f_X)$ is a sum over the product of k arcs' f_X's, the degree of $\mathrm{hc}(G_k, f_X)$ in r is $(n-k)k$.

Let $r_1, r_2, \ldots, r_m$ be m distinct elements in F and let f_j be equal to f_X evaluated in $r = r_j$. By Lagrange interpolation, it is possible to compute $\mathrm{hc}(G_k, f_X)$ and in particular the coefficient of r_{n-k} from the evaluated polynomial points $\mathrm{hc}(G_k, f_j)$. ◀

The F-Permanent case is similar: consider an instance (G_n, f). Lemma 6 states that $\mathrm{per}(G_n, f)$ is related to a coefficient in a polynomial resulting from a sum of many smaller instances (G_k, f_X) to $F[r]$-Permanent.

▶ **Lemma 9.** *For every polynomial term* $\operatorname{per}(G_k, f_X) \prod_{i=k+1}^{n} C_X(i)$ *in the outer sum in Lemma 6, it is possible to compute* $(n-k)n+1$ *instances* (G_k, f_i) *for* $i = 1, \ldots, (n-k)n+1$ *to the* F-PERMANENT *on* k *vertices, and constants* $a_i \in F$ *for* $i = 1, \ldots, (n-k)n+1$ *so that*

$$[r^{n-k}] \operatorname{per}(G_k, f_X) \sum_{i=k+1}^{n} C_X(i) = \sum_{j=1}^{(n-k)n+1} a_j \operatorname{per}(G_k, f_j)\,.$$

Proof. Each entry in the codomain of f_X has degree $n-k$ by definition of the ranked walks and the definition of f_X in Eq. 2. Since $\operatorname{per}(G_k, f_X)$ is a sum over the product of k arcs f_X's, the degree of $\operatorname{per}(G_k, f_X)$ in r is $(n-k)k$. The degree of $\prod_{i=k+1}^{n} C_X(i)$ is $(n-k)(n-k)$ since every $C_X(i)$ has degree $n-k$ by the definition Eq. 3. Altogether, $\operatorname{per}(G_k, f_X) \prod_{i=k+1}^{n} C_X(i)$ has degree $(n-k)n$.

Let $r_1, r_2, \ldots, r_m$ be m distinct elements in F and let f_j be equal to f_X evaluated in $r = r_j$. Likewise, let b_j be equal to $\prod_{i=k+1}^{n} C_X(i)$ evaluated in $r = r_j$. By Lagrange interpolation, it is possible to compute the coefficent of r^{n-k} in $\operatorname{per}(G_k, f_X) \prod_{i=k+1}^{n} C_X(i)$ from the evaluated polynomial points $b_j \operatorname{per}(G_k, f_j)$. ◀

The self-reduction for F-PERMANENT Lemma 3 follows from the combination of Lemma 6 and Lemma 9, after observing that each $X \subseteq V - K$ and each $r \in 1, \ldots, (n-k)n+1$ corresponds to one small instance. Similarly, the self-reduction for F-HAMCYCLES Lemma 4 follows from Lemma 5 and Lemma 8 with $X \subseteq V - K$ and $r \in 1, \ldots, (n-k)k+1$. It remains to validate the runtime in terms of the number of arithmetic operations used. To compute a small instance (G_k, f_i) in Lemma 3 (Lemma 4 respectively), corresponding to a particular $X \subseteq V - K$ and $r \in 1, \ldots, (n-k)n+1$, we see from the definitions Eqs. 2 and 3 that the instance elements are computed as walks in X for a fixed r. We can compute the elements through the recursive definition of the ranked walks Eq. 1 via dynamic programming in only polynomial in n number of arithmetic operations.

3 The Algorithms

In this section we prove our main theorems:

▶ **Theorem 10.** *Any single* $n \times n$ *matrix instance of* PERMANENT *with* $\operatorname{poly}(n)$*-bit integer elements can be solved deterministically in* $2^{n-\Omega(\sqrt{n/\log n})}$ *time.*

▶ **Theorem 11.** *Any single* n*-vertex directed graph instance of* HAMCYCLES *with* $\exp(\operatorname{poly}(n))$ *number of arcs can be solved deterministically in* $2^{n-\Omega(\sqrt{n/\log n})}$ *time.*

We immediately observe that the above theorem via a standard embedding of the $(\min, +)$-semiring on the integers, can be used to count cycles by weight through polynomial interpolation. In particular, the problem of finding the length of the shortest Hamiltonian cycle, known as the Asymmetric Traveling Salesman problem can be solved by the technique.That is, we introduce yet another indeterminate z, associate an arc of weight w with z^w, and finally solve for the smallest non-zero monomial in the resulting polynomial, see e.g. [12]. Since the evaluated polynomial is of degree at most Mn^2, we get

▶ **Corollary 12.** *The shortest Asymmetric Traveling Salesman Problem route in an* n*-vertex graph with integer arc weights in* $[0, \ldots, M]$ *can be computed in* $Mn^2 2^{n-\Omega(\sqrt{n/\log(Mn)})} + M^2 n^4$ *time.*

On the top level, the idea of the algorithms is to bring the computations down to small finite fields. We next use the self-reductions from Section 2 to transform the input matrix/graph down to so small ones that several of them will be identical. By tabulating which ones of them have been constructed in this process and how often, it then suffices to compute the permanent of the small matrices/the Hamiltonian cycles of the small graphs only once. To make this precise we first need some elementary results from number theory.

3.1 Preliminaries on Modular Arithmetic

The well-known Chinese remainder theorem has two parts, an existence and a constructive one. The existence part states that an integer solution to a set of linear modular equations is uniquely defined in the range between zero and the least common multiple of the moduli. The constructive part describes how to recover the solution given the modular equations. We state them here in a slightly modified form as we will need them

▶ **Lemma 13** (CRT). *Given m distinct primes p_i, and residues $0 \leq a_i < p_i$, $1 \leq i \leq m$,*

- *Existence: There is a unique integer n in $\left\lfloor -\frac{\prod_{i=1}^m p_i}{2} \right\rfloor \leq n < \left\lceil \frac{\prod_{i=1}^m p_i}{2} \right\rceil$ fulfilling $n \equiv a_i(\mod p_i), 1 \leq i \leq m$.*
- *Construction: The integer n can be computed by evaluating $n_+ = \sum_{i=1}^m a_i r_i$ where $r_i = \prod_{j \neq i} p_j \left(\left(\prod_{j \neq i} p_j \right)^{-1} (\mod p_i) \right)$ and then setting $n = n_+$ if $n_+ < \frac{\prod_{i=1}^m p_i}{2}$, and $n = n_+ - \prod_{i=1}^m p$ otherwise.*

We also use the following bound of the prime number theorem to answer how many and large primes we will need to break down a computation using the CRT:

▶ **Lemma 14** (Rosser [13]). *For every integer $n \geq 55$ the number of primes $\pi(n)$ less than or equal to n obey $n/(ln(n)+2) < \pi(n) < n/(ln(n)-4)$.*

3.2 The Algorithm

We will first describe the algorithm for the Permanent case Thm. 10 , and then point out the few changes needed for the HamCycles case Thm. 11. We begin by describing the algorithm in pseudo-code below. Next we will explain the steps in more detail.

Permanent per(G_n, f)
1. Let M be the largest absolute value in the image of f.
2. Let P be the smallest set of primes $> n^2$ such that $\prod_{p \in P} p > 2M^n n!$.
3. Let $k = \lfloor \sqrt{.99n/\log_2 p_{\max}} \rfloor$ where $p_{\max} = \max_{p \in P} p$.
4. For each prime $p \in P$
5. Construct a table T from all $\mathbb{Z}_p^{k \times k}$ matrices to the positive integers, initially set to all zeros.
6. Evaluate $f_{(p)} = f(\mod p)$.
7. Compute $m = (n-k)n2^{n-k}$ instances (G_k, f_j) and constants a_j for $j = 1, \ldots, m$ to $\mathbb{Z}_p$-Permanent such that per$(G_n, f_{(p)}) = \sum_{j=1}^m a_i$ per(G_k, f_j).
8. For $j = 1, \ldots, m$
9. Let $T(f_j) = T(f_j) + a_j$.
10. Set $sum = 0$.
11. For each g with non-zero table entry $T(g)$

12. Compute $\operatorname{per}(G_k, g)$ using Ryser's permanent algorithm.

13. Let $sum = sum + T(g)\operatorname{per}(G_k, g)(\mod p)$.

14. Store $per(G_n, f_{(p)}) = sum$

15. Compute the permanent over $\mathbb{Z}$ using the stored $\operatorname{per}(G_n, f_{(p)})$ for all $p \in P$ using the constructive part of CRT.

The existence part of CRT Lemma 13 makes it clear that to compute an integer function solely with the operations $+$ and $*$ over the integers, one can just as well compute it modulo several primes and assemble the result in the end. Both the Permanent and the HamCycles problems are defined as sum–products, so to compute their quantities modulo a prime p, we can replace the input integers with their residues modulo p. Steps 2–6 of the algorithm do precisely that, transform the input integer Permanent instance to instances of $\mathbb{Z}_p$-Permanent for primes p. Step 7 next generates $(n-k)n2^{n-k}$ instances of the $\mathbb{Z}_p$-Permanent problem using the constructive proof for Lemma 3. Steps 8–9 counts the occurrences of each of the different matrices in $\mathbb{Z}_p^{k \times k}$ by keeping track of the total coefficients of each of the smaller matrices' permanents in Lemma 3. Steps 10–14 computes the solution to the $n \times n$-matrix permanent $\operatorname{per}(G_n, f_{(p)})$, and finally step 15 assembles the modular results using the constructive part of the CRT Lemma 13. The correctness of the algorithm follows from Lemma 13 and the self-reduction Lemma 3, after noting that enough primes are chosen in step 2.

To bound the runtime, the only question is how many and how large primes are required, and indirectly, how large tables will be used? The permanent is a sum of $n!$ products of n elements from the input function f. In step 2 of the algorithm we measure the absolute max over all elements used to conclude that $|\operatorname{per}(G_n, f)| \leq M^n n! < 2^{n^c}$ for some positive constant c when the input entries have $\operatorname{poly}(n)$ bits. From Lemma 14 we see that there are at least $m = n^d/(dln(n)+2) - n^2/(2ln(n)-4)$ primes larger than n^2 but smaller than n^d for $n \geq 55$. We want the product of the first m primes larger than n^2, the set of primes P in step 4 of the algorithm, to be larger than $2 \cdot 2^{n^c}$, i.e. $n^{2m} > 2^{n^c+1}$. It is straightforward to note that a constant d depending on c will suffice, in fact using $d = c+3$ is more than enough. Hence $p_{\max}$ in step 3 is bounded by n^d for d constant and k is $\Omega(\sqrt{n/\log n})$.

For each prime $p \in P$ in step 4, we use a table T in step 5–13 with one entry per matrix in $\mathbb{Z}_p^{k \times k}$. An upper bound on the number of matrices in $\mathbb{Z}_p^{k \times k}$ using $p_{\max}$ from step 3 and k from step 4 of the algorithm is $(p_{\max})^{k^2} < 2^{0.99n}$. The runtime of steps 5–9 is easily seen to be $O((n-k)n2^{n-k})$ from the bound on the table T's size and Lemma 3. Computing the permanent of each of the matrices is a $O(k2^k)$ time task with Ryser's algorithm [14], so the total runtime of steps 10–15 is $o(2^{n-k})$. Altogether, the loop at steps 4–14 is run a polynomial number of times, and step 15 is polynomial time, so we get $\operatorname{poly}(n)2^{n-k}$ time in total which is $2^{n-\Omega(\sqrt{n/\log n})}$ time as claimed.

To adjust the algorithm and the proof to counting HamCycles, all we need to do is to replace Lemma 3 for Lemma 4 in step 7 of the algorithm and the analysis, and exchange Ryser's algorithm for the permanent in step 12 for e.g. Bax's [1] Hamiltonian cycle counting algorithm.

Acknowledgments. I thank Thore Husfeldt, Alexander Golovnev, Petteri Kaski, Mikko Koivisto, and Ryan Williams and several anonymous referees for comments on an earlier draft of the paper and stimulating discussions on the subject.

References

1. E. T. Bax. Inclusion and exclusion algorithm for the hamiltonian path problem. *Inform. Process. Lett*, 47:203–207, 1993.
2. E. T. Bax and J. Franklin. A permanent algorithm with $exp[\omega(n^{1/3}/2ln(n))]$ expected speedup for 0-1 matrices. *Algorithmica*, 32:157–162, 2002.
3. R. Bellman. Dynamic programming treatment of the travelling salesman problem. *J. Assoc. Comput. Mach*, 9:61–63, 1962.
4. A. Björklund. Counting perfect matchings as fast as ryser. In *Proceedings of the ACM-SIAM SODA*, pages 914–921, 2012.
5. A. Björklund. Determinant sums for undirected hamiltonicity. *SIAM J. Comput.*, 43:280–299, 2014.
6. A. Björklund, H. Dell, and T. Husfeldt. The parity of set systems under random restrictions with applications to exponential time problems. In *Proceedings of ICALP,*, pages 231–242, 2015.
7. A. Björklund and T. Husfeldt. The parity of directed hamiltonian cycles. In *Proceedings of the IEEE FOCS,*, pages 724–735, 2013.
8. M. Cygan, S. Kratsch, and J. Nederlof. Fast hamiltonicity checking via bases of perfect matchings. In *Proceedings of the ACM STOC*, pages 301–310, 2013.
9. M. Held and R. M. Karp. A dynamic programming approach to sequencing problems. *J. Soc. Indust. Appl. Math.*, 10:196–210, 1962.
10. R. M. Karp. Dynamic programming meets the principle of inclusion and exclusion. *Oper. Res. Lett.*, 1:49–51, 1982.
11. D. E. Knuth. *The Art of Computer Programming. Vol. 2: Seminumerical Algorithms.* Addison-Wesley, third edition, 1997.
12. S. Kohn, A. Gottlieb, and M. Kohn. A generating function approach to the traveling salesman problem. In *Proceedings of the Annual Conference (ACM'77), Association for Computing Machinery*, pages 294–300, 1977.
13. B. Rosser. Explicit bounds for some functions of prime numbers. *American Journal of Mathematics*, 63:211–232, 1941.
14. H. J. Ryser. *Combinatorial Mathematics.* The Carus mathematical monographs, The Mathematical Association of America, 1963.