Modular Sieves for Directed Hamiltonian Cycles

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

Ioannis Koutis
Computer Science Department
University of Puerto Rico – Rio Piedras
Puerto Rico
ioannis.koutis@upr.edu

Abstract

In this paper we make further progress on the following problem: can we detect if an 
$n$-vertex directed graph has a Hamiltonian cycle much faster than in $2^n$ time?

We first show that the number of Hamiltonian cycles in a directed $n$-vertex graph can 
be counted modulo prime powers $p^k$ in $(2 - \Omega(1))^n$ expected time, for any constant prime 
p and $k < (1 - \lambda)n/p$ for any arbitrary fixed $\lambda > 0$. Previously a $(2 - \Omega(1))^n$ algorithm 
was known only for counting modulo two [Björklund and Husfeldt, FOCS 2013].

We then give three applications of the counting result and the techniques used.

I. We show that when we know that there are at most $d^n$ Hamiltonian cycles in a 
directed graph for some arbitrarily large known constant $d$, we can detect the presence 
of a Hamiltonian cycle by a Las Vegas algorithm running in $(2 - \Omega(1))^n$ expected time. 
Previously this was known to hold only for $d < 1.035$ [Björklund, Dell, and Husfeldt, 
ICALP 2015]. In particular we can count the Hamiltonian cycles in a directed graph of 
average outdegree $d$, in $2^{n-n/poly(d)}$ expected time. Such bounds were previously only 
known with a doubly-exponential dependency on the degree [Cygan and Pilipczuk, ICALP 
2013].

II. We show that we can count the Hamiltonian cycles modulo $2^k$ for any positive 
integer $k$ in $\sum_{i=0}^{k-1} \binom{n}{i}1.619^n$ time, deterministically and using only polynomial space. As a 
direct consequence this result gives the same bounds but without any use of randomization 
as the detection algorithm from [Björklund, Dell, and Husfeldt, ICALP 2015].

III. We show that a Hamiltonian cycle in a directed graph of in- plus outdegree at every 
vertex bounded by $d$ can be detected by a Monte Carlo algorithm running in $2^{n-\Omega(n/d^2)}$ 
time, using only polynomial space. This improves over the previously best algorithm that 
had an exponential dependency on $d$ in the denominator of the $\Omega$-expression [Björklund 
et al. TALG 2012].

1 Introduction

The problem of determining if an $n$-vertex undirected or directed graph has a Hamiltonian 
cycle is known to be NP-complete [21]. For a long time the best worst case algorithm ran in 
$2^n \text{poly}(n)$ time [2 [17]. For several other well known problems like CNFSAT, i.e. satisfying 
a conjunctive normal form Boolean formula on $n$ variables, computing the permanent of an 
$n \times n$ 0/1 matrix, or computing the chromatic number of an $n$-vertex graph, the best known 
upper bounds are all on the same form $2^{n-o(n)} \text{poly}(n)$ [21 [9] 5]. A very natural question 
to ask is whether this is indeed the best you can do for these NP-hard problems.

1In the remainder of this paper we will omit the polynomial factor in the input size in the running times to 
highlight the exponential part we are interested in. It is typically subsumed by the exponential bound given.
In the case of CNFSAT, there is even a hypothesis known as the Strong Exponential Time Hypothesis of Impagliazzo and Paturi [18], that loosely says that the running time indeed approaches $2^{n-o(n)}$ as the clause-width grows. A few problems are known to be just as hard as CNFSAT in a worst case sense. E.g. Cygan et al. [12] showed that computing the parity of the number of solutions to a set cover instance on a universe of $n$ elements, cannot be done in $(2 - \Omega(1))^n$ time unless the Strong Exponential Time Hypothesis is false. However, there are no known relations between the other problems mentioned above that are strong enough to imply that if one of the problems has a $(2 - \Omega(1))^n$ time algorithm then the others do as well.

To the contrary, the Hamiltonian cycle problem appears to be easier. In 2010, a randomized algorithm running in $O(1.657^n)$ time for the undirected Hamiltonian cycle case was found [3], but it is still open whether a $(2 - \Omega(1))^n$ time algorithm exists also for the directed case. Despite some partial success, in particular the striking $O(1.888^n)$ time algorithm for bipartite directed graphs by Cygan et al. [13], the best general algorithm known to date only guarantees a $2^n-\Omega(\sqrt{n \log n})$ running time [9]. In this paper we make further progress on this problem. We prove that unless there are really many solutions, we can detect if there is one much faster than the best known general algorithm.

1.1 Our results and related work

Our main counting result says

**Theorem 1.** There is a Las Vegas algorithm that given any $n$-vertex directed graph $G$ and a positive integer $m = \prod_{\text{prime} \leq q} p^{(1-\lambda)n/p}$ can compute the number of Hamiltonian cycles in $G$ modulo $m$, in $2^{n-\Omega(\lambda^2 n/(q \log q))}$ expected time for any positive constant $\lambda$.

This means that for small primes below any positive constant, we can do the modular arithmetic in $(2 - \Omega(1))^n$ time. We can count the solutions modulo these smooth integers, i.e. integers whose prime factorization only involves small primes. This allows us to count modulo really large integers, but yet apparently escapes the possibility to always detect a Hamiltonian cycle when there are really many of them. Note that if we could do the modulo counting for just one prime of magnitude $\Omega(n)$, we could use the DeMillo-Lipton-Schwartz-Zippel lemma [15, 23] to get an efficient detection algorithm. That is, we could sample integer $O(n)$ weights for the arcs and replace each arc with that many multiple arcs. With large probability the result of the modular counting would be non-zero if there was a solution at all.

The theorem above continues a research direction set out by Björklund and Husfeldt [4]: counting the solutions modulo two was shown there to be doable in $O(1.619^n)$ time, but their technique neither seems to be able to handle powers of two, nor other small prime moduli. To be precise, they presented a new formula for counting modulo an arbitrary positive integer, but only for the case two were they able to use it to get $o(2^n)$ time algorithms. Our work here builds on another formula.

We will give three (in)direct applications of this algebraic approach. Our first and main application says

**Theorem 2.** There is a Las Vegas algorithm that given any $n$-vertex directed graph with at most $d^n$ Hamiltonian cycles, can tell if it has any at all, running in $2^{n-\Omega(n/(d^2 \log d))}$ expected time.
This means that for any constant $d$, we have a $(2 - \Omega(1))^n$ time algorithm for the decision problem. It leaves open only the case when there are really many solutions. It is plausible that such a special case is amenable to other techniques. All we need is a way to approximate the number of Hamiltonian cycles with an additive error less than $d^n$ in $(2 - \Omega(1))^n$ time. Alas, we have no idea at the moment on how to obtain such a result in general.

A weaker result in the same vein as theorem [2] was already known to hold but with a much smaller bound on the number of solutions. Björklund et al. [7], building on the algorithm of Björklund and Husfeldt [4], proved that if there are less than $1.035^n$ Hamiltonian cycles in a directed graph, you can detect if there are any at all in $(2 - \Omega(1))^n$ time. Note that this makes the directed Hamiltonian cycle detection problem different from CNFSAT in that having fewer solutions seems to help in finding one, whereas for CNFSAT solution uniqueness does not help [10]. An interesting consequence of the theorem is that in directed graphs of bounded outdegree $d$, which via the arithmetic–geometric mean inequality can be seen to have at most $d^n$ Hamiltonian cycles, we have

**Corollary 3.** There is a Las Vegas algorithm that given any $n$-vertex directed graph of average outdegree $d$ counts its Hamiltonian cycles, in $2^n - \Omega(n/(d^2 \log d))$ expected time.

We note that such bounds were previously known only with a doubly–exponential dependency on the average degree in the denominator of the $\Omega$-expression (it can be read off the algorithm for TSP in [14]). Such an improvement was in retrospect perhaps to be expected: a similar improvement for the related problem of computing the permanent of $0/1$ $n \times n$ matrices with an average of $d$ ones per row was shown recently when Izumi and Wadayama [19] found a $2^n - \Omega(n/(d \log d))$ time algorithm where the previous best was $2^n - \Omega(n/\exp(d))$. The current best for that problem is $2^n - \Omega(n/d)$ time [14].

By using the machinery of listing solutions to quadratic equation systems over GF(2) on a special form from Björklund and Husfeldt [4], we can also prove

**Theorem 4.** There is a deterministic algorithm that given any $n$-vertex directed graph $G$ and a positive integer $k$ counts the Hamiltonian cycles in $G$ modulo $2^k$, in $\sum_{i=0}^{k-1} \binom{n}{i}1.619^n$ time.

In particular, this result derandomises the Hamiltonicity detection result from Björklund et al. [7] in the sense that it gives the same running time bounds as that result when we know there are at most $2^k$ solutions.

Finally we present a fast algorithm when the in- plus outdegree is bounded. We show

**Theorem 5.** There is a Monte Carlo algorithm that given any $n$-vertex directed graph of in- plus outdegree bounded by $d$, can tell if the graph contains a Hamiltonian cycle or not, in $2^n - \Omega(n/d^2)$ time using only polynomial space.

Observe that a Las Vegas $2^n - \Omega(n/(d^2 \log d))$ expected time algorithm follows directly from corollary [3] but that algorithm uses exponential space, and the constant hidden under the $\Omega$-expression is quite small. We can get rid of the heavy space requirement and also get all the way down to $d^2$ in the denominator of the $\Omega$-expression by giving up on natural counting and instead work over a field of characteristic two and utilize the sparse regular structure of the graph. We will use the DeMillo-Lipton-Schwartz-Zippel lemma and formulate our result in terms of a Monte Carlo algorithm where the error is solely due to the bound from this lemma. The previous best results in terms of the degree follows from algorithms for TSP in undirected graphs. With small adoptions, one can see that the result for bounded $d$ degree graphs in [6] gives a $2^n - \Omega(n/d^2)$ time algorithm for directed Hamiltonicity of in- plus outdegree bounded by $d$. 

3
1.2 The ideas behind our algorithms

Our algorithm follows recent algebraic techniques to track combinatorial objects by counting through exponential sums with cancellations of unwanted terms. In particular, we develop an explicit sum of $2^n$ terms but take measures to make many of the terms vanish modulo a small prime power, and develop means to sieve out only the terms that give a non-zero contribution. This basic idea of reducing the number of contributing terms in an exponential sum goes back at least to Bax and Franklin’s paper on a $o(2^n)$ time algorithm for the $0/1$ matrix permanent \cite{bax1996}. The same idea was employed in Björklund and Husfeldt \cite{bjorklund2010}, but the result there only works for counting modulo two. Our sum here is an inclusion–exclusion formula over a matrix determinant whose value is known from the Matrix tree theorem to count spanning sink rooted trees, i.e. trees in which all vertices have a unique path to the sink. The use of the Matrix tree theorem in (parametrized) hard graph algorithms is not new, it was e.g. used recently by Gabizon et al. \cite{gabizon2018}. To efficiently list the contributing terms we use a variation of an algorithm by Björklund et al. \cite{bjorklund2010} that was used for a related problem, computing the permanent modulo small prime powers.

For our theorems \ref{thm:main} and \ref{thm:bounded}, we will also use the fast space-efficient solution listing algorithm to certain quadratic equation systems over $GF(2)$ from Björklund and Husfeldt \cite{bjorklund2010} (We will refer to the self-contained section 4 of that paper for a full description of the listing algorithm). For our result on bounded in- plus outdegree graphs, we in addition use that such regular graphs has a large set of vertices of which no pair share a neighbor. To find proof of a Hamiltonian cycle being present in this case, we use the DeMillo-Lipton-Schwartz-Zippel lemma \cite{demillo1978,lipton1980}.

1.3 Overview

Our counting result theorem \ref{thm:main} is presented in full in section \ref{sec:counting}. In section \ref{sec:overview} we prove theorem \ref{thm:main} and corollary \ref{cor:main} in subsection \ref{subsec:overview}, theorem \ref{thm:bounded} in subsection \ref{subsec:bounded} and finally theorem \ref{thm:bounded} in subsection \ref{subsec:bounded}.2

2 The counting algorithm

In this section we prove theorem \ref{thm:main}. We will describe the algorithm gradually as we develop the technical parts needed. Input to our algorithm is a directed $n$-vertex graph $\hat{G}$ and we wish to compute the number of its Hamiltonian cycles, denoted $hc(\hat{G})$ modulo a composite $m$ to be defined later. We begin by showing a relation between the Hamiltonian paths in a graph to a sum of matrix determinants.

2.1 A formula for the number of Hamiltonian paths

Given a directed loopless multigraph $G = (V, A)$, the number of Hamiltonian paths starting in vertex $s \in V$ and ending in vertex $t \in V$ is denoted by $hp(G, s, t)$. We will use an inclusion–exclusion formula over the Matrix Tree theorem for directed graphs (see e.g. \cite{jonsson1993}). In the interest of being self-contained, we provide a complete proof of the resulting formula below. First, we introduce variables $x_v$ for every vertex $v \in V$, and consider the graph’s Laplacian
matrix $L_G$, whose entries are defined by

$$L_G(u, v) = \begin{cases} m_{uv}x_v & : uv \in A \\ -\sum_{u \in A} m_{uw}x_u & : u = v \\ 0 & : \text{otherwise.} \end{cases} \quad (1)$$

Here $m_{uv}$ is the multiplicity of an arc $uv \in A$. We will think of a non-zero entry at row $u$ and column $v$ as a (weighted) arc from $u$ to $v$. Next, we form $L_G^{(t)}$ from $L_G$ for a vertex $t \in V$ by omitting column and row $t$ from $L_G$. Likewise, we let $A^{(t)}$ denote the arcs in $A$ that do not include vertex $t$. Let $L_G^{(t)}(X)$ for a subset $X \subseteq V$ be the matrix where we have equated $x_u$ with one if $u \in X$, and zero if $u \notin X$. We have

**Lemma 6.**

$$\text{hp}(G, s, t) = \sum_{\{s\} \subseteq X \subseteq V} (-1)^{|V \setminus X|} \det(L_G^{(t)}(X)).$$

**Proof.** The Leibniz formula for an $n \times n$ matrix determinant says that

$$\det(L_G^{(t)}) = \sum_{\sigma} (-1)^{\text{sgn}(\sigma)} \prod_{i=1}^{n} L_G^{(t)}(i, \sigma(i)),$$

where $\sigma$ ranges over all permutations of $n$ elements. Here it will be instructive to think of the elements of $L_G^{(t)}$ as weighted arcs, and in particular, think of the diagonal elements as several weighted arcs, one for each term in the definition $\Box$. I.e. each term on the diagonal corresponds to a unique arc in $A$ with its original weight negated. We define $A^*$ as the union of $A^{(t)}$ and another copy of the arcs for each of these diagonal terms, i.e. one extra arc for each $uv \in A$. We define the weight function $w : A^* \rightarrow Z$ as

$$w(uv) = \begin{cases} m(uv) & : uv \in A^{(t)} \\ -m(uv) & : uv \in A^* \setminus A^{(t)}. \end{cases}$$

With this viewpoint in mind, we note that each term in the Leibniz formula, after expanding the sum–product over the diagonal elements, can be thought of as the product over a subset of $n$ arcs, where every node is the origin of exactly one arc. Call such arc subsets *non-branching*. We can thus rewrite the Leibniz formula as

$$\det(L_G^{(t)}) = \sum_{A' \subseteq A^*} (-1)^{\text{sgn}(A')} \prod_{a \in A'} w(a).$$

Here $\text{sgn}(A')$ equals $(-1)^{n-c(A')}$ where $c(A')$ is the number of components formed by the arcs in $A' \cap A^{(t)}$ plus the number of arcs in $A' \setminus A^{(t)}$ (the diagonal elements). In particular, if $A' \cap A^{(t)}$ is non-empty, there must be a cycle in $A' \cap A^{(t)}$ since every such arc must end in a vertex that is the origin of another arc in $A' \cap A^{(t)}$. We will see that any such $A'$ will be cancelled in the sum. In fact, every $A'$ in which there is a cycle will be cancelled by a term resulting from another arc subset. By the argument above, every cycle in $A'$ consists either solely of arcs in $A \cap A^{(t)}$ or arcs in $A \setminus A^{(t)}$. For each $A'$ containing a cycle, define $\pi(A') \subset A^*$ as another arc subset which is identical to $A'$ except on a smallest cycle $C$ in $A'$ (a smallest cycle can e.g. be defined as the cycle containing the smallest vertex under any fixed vertex
ordering). Instead of the arcs on \( C \), \( \pi(A') \) has the negated versions of the arcs, i.e. if the cycle was in \( A' \cap A^{(t)} \) in \( A' \) it is now in \( A' \setminus A^{(t)} \) in \( \pi(A') \), or the other way around. Note that \( \pi \) defines a fixedpoint–free involution, i.e. \( A' \neq \pi(A') \) and \( A' = \pi(\pi(A')) \). Hence all the \( A' \) with cycles are paired up uniquely. Next we will argue that the terms for \( A' \) and \( \pi(A') \) will be identical in absolute value but with opposite signs, effectively cancelling each other. If the cycle \( C \) had an odd number of arcs, \( \text{sgn}(A') = \text{sgn}(\pi(A')) \) but we have negated an odd number of weights effectively negating the product of arcs. Therefor \( A' \)'s and \( \pi(A') \)'s contributions will cancel each other. If the cycle \( C \) had an even number of arcs, \( \text{sgn}(A') = -\text{sgn}(\pi(A')) \) and we have negated an even number of weights resulting in the same product over the arcs. Therefor again \( A' \)'s and \( \pi(A') \)'s contributions will cancel each other. We are left with arc subsets \( A' \) that has no cycles, and hence no arcs in \( A \cap A^{(t)} \), but all arcs in \( A \setminus A^{(t)} \). These are subsets form a single tree oriented from the leaves to the root \( t \). We will argue that only paths from \( s \) through \( t \) survives the inclusion–exclusion summation. Since each arc \( uv \) includes the variable \( x_v \) associated with its endpoint \( v \), every tree \( A' \) can be associated with a monomial of total degree \( n \). Let \( X(A') \) denote the set of such variables in the monomial. If \( X(A') \subset V \setminus \{s\} \) the term associated with \( A' \) will be counted for every subset \( X \supset X(A') \cup \{s\} \) in the summation, and for exactly half of those subsets it will be negated. Hence such \( A' \) will not contribute to the sum. Only if \( X(A') = V \setminus \{s\} \) will the \( A' \) be counted (once for \( X = V \) in the summation), and such \( A' \) are trees with only \( s \) as a leaf, in other words describe a directed path from \( s \) to \( t \) through all the vertices.

\[ \square \]

### 2.2 From cycles to paths and getting rid of most terms

Lemma \[4\] gives a formula for the number of Hamiltonian paths, but we will aim for a formula for the Hamiltonian cycles. We will augment the input graph \( \hat{G} = (\hat{V}, \hat{A}) \) with two vertices and a few edges into a new graph \( G \), and identify two special vertices \( s \) and \( t \) so that \( h_c(\hat{G}) = \text{hp}(G, s, t) \). In fact the construction has another purpose besides reducing cycle counting to path counting, it is also needed to make certain probabilistic events used in the algorithm independent of each other. This independence will enable an easy way to argue that most of the terms in the summation will be zero modulo a small prime power.

The construction selects a vertex \( \hat{s} \) in \( \hat{G} \), copies \( \hat{G}[\hat{V} \setminus \{\hat{s}\}] \) to \( G \), and inserts two vertices \( s \) and \( t \) in \( G \) instead of \( \hat{s} \). Every arc \( \hat{sw} \) in \( \hat{G} \) is replaced for an arc \( sw \) in \( G \), and every arc \( w\hat{s} \) in \( \hat{G} \) is replaced for an arc \( wt \) in \( G \). In addition, for each vertex \( v \in (\hat{V} \setminus \{\hat{s}\}) \cup \{t\} \) we pick a random number \( a_v \in \{0, \ldots, p - 1\} \), uniformly and independently, and add \( a_v \) arcs from \( v \) to \( s \) in \( G \). Note that no Hamiltonian path originating from \( s \) can use any of these randomly added arcs. Also note the Hamiltonian paths in \( G \) originating in \( s \) and ending in \( t \) is in one-to-one correspondence with the Hamiltonian cycles in \( \hat{G} \).

Let \( \varepsilon_u \) for a vertex \( u \) be the event that in a random \( G \) sampled as above and for a uniformly sampled subset \( X \subseteq V \), the \( u \)th column of \( L^{(t)}_G \) has only elements divisible by \( p \). Note that the probability \( \Pr(\varepsilon_u) = \frac{1}{2p^2} \) as off-diagonal elements are zero with probability \( 1/2 \) (depend only on \( x_u \)) and the diagonal element is divisible by \( p \) with probability \( 1/p \), and these two probabilities are independent. Also note that \( \varepsilon_u \) and \( \varepsilon_v \) for any \( u \neq v \) are independent since both \( x_u \) and \( x_v \) are, and also \( a_u \) and \( a_v \) are. Let \( Z \) be the number of columns in \( L^{(t)}_G \) identical to an all-zero vector (mod \( p \)), for a random \( G \) and uniformly chosen \( X \), then we get

\[ \mathbb{E}(Z) = \frac{n}{2p}. \]
Since two columns being zero modulo \( p \) are independent events, we have

\[ \Pr(Z \leq (1 - \lambda)\mathbb{E}(Z)) < e^{-\lambda^2\mathbb{E}(Z)/2}, \]

by a standard Chernoff bound. So in expectation there are \( 2^n e^{-\lambda^2 n/(4p)} \) \( X \)'s that contribute non-zero terms modulo \( p^{(1-\lambda)n/(2p)} \) to the sum in lemma 6. This means there is hope for an algorithm running in roughly \( 2^n - \Theta(n/p) \) time that computes the number of Hamiltonian cycles modulo \( p^{(1-\lambda)n/(2p)} \) for some \( \lambda > 0 \). All we need is to list those \( X \)'s fast and sum their contribution in polynomial time per term. We don’t know how to do it quite that fast but at least in time \( 2^n - \Theta(n/(p \log p)) \) there is a way. We describe the algorithm next.

### 2.3 Listing the non-zero terms

To efficiently list the non-zero terms in the expression for \( hp(G, s, t) \) in lemma 6, we will use the following idea: First we divide the rows in two asymmetrical parts, \( T \) for top, and \( B \) for bottom. The best balance between the two parts will depend on \( p \), but for simplicity we will assume \( T \) is holding \( n/3 \) rows. For each possible 0/1 assignment \( x_v \) for \( v \in T \), we construct one vector \( w \in [p+1]^n \) of length \( n \) indexed by the columns of the matrix \( L_G(t) \), where the element at coordinate \( c \), corresponding to column \( c \), holds the value of \( p \) if \( (c \in T) \land (x_c = 1) \), or otherwise the value

\[ \sum_{r \in T \land cr \in A(t)} m_{cr} x_r \mod p, \]

i.e. the value of the diagonal entry \( L_G(t)(c, c) \) assuming the variables \( x_v \) for \( v \in B \) are zero. Let \( X(w) \) denote the subset of \( T \) whose variables were set to 1 under the assignment. Call the resulting set of \( 2^{n/3} \) vectors \( T \).

Likewise, for each possible 0/1 assignment to the variables \( x_v \) for \( v \in B \), we construct one vector \( z \in [p+1]^n \) again indexed by the same matrix columns, where the element at coordinate \( c \) holds the value \( p \) if \( (c \in B) \land (x_c = 1) \), or otherwise the value

\[ \sum_{r \in B \land cr \in A(t)} - m_{cr} x_r \mod p. \]

Observe the negation from the top case. Again let \( X(z) \) denote the subset of \( B \) of variables set to one under the assignment and call the resulting set of \( 2^{2n/3} \) vectors \( B \).

With the above construction in mind, note that the term corresponding to \( X = X(w) \cup X(z) \) in the equation in lemma 6 will be divisible by \( p^k \) if \( w \) and \( z \) agree on \( k \) or more coordinates. Hence, to list (a superset of) the non-zero terms modulo \( p^{(1-\lambda)n/(2p)} \), we are left with the problem of listing all tuples \((w, z) \in T \times B \) that are identical in at most \((1 - \lambda)n/(2p) \) coordinates.

To accomplish this, we build \( \ell = 3 \log p \) tables \( \Gamma_i(T) \) from \( T \), for \( i = 1, 2, \ldots, \ell \). Each \( i \) represent a disjoint interval of consecutive columns, \( I_i = \{(i-1)n/\ell + 1, \ldots, in/\ell \} \). In \( \Gamma_i(T) \) the keys are vectors \( l \in [p+1]^{n/\ell} \) and each key is associated with the list of all vectors in \( T \) that are identical in at most \((1 - \lambda)n/(2p\ell) \) coordinates on the interval \( I_i \) with the key \( l \). Note that constructing all \( \Gamma_i(T) \) takes time \( 2^{n/3}p^{n/\ell} = 2^{2n/3} \).

For every tuple \((w, z) \in T \times B \) that agree in at most \((1 - \lambda)n/(2p) \) coordinates, there must exist an interval \( I_i \) on which they agree in at most \((1 - \lambda)n/(2p\ell) \) coordinates, by an averaging argument. Hence, for each \( i = 1, \ldots, \ell \) we can loop over the vectors \( z \in B \) and use the tables...
Γ(T) to list all pairs (w, z) ∈ T × B that are equal in at most (1 − λ)n/(2pℓ) coordinates on some interval Ii, and be sure that we have at least included all the pairs we were looking for. Remembering the analysis in section 2.2 the expected number of pairs that are equal in at most (1 − λ)n/(2pℓ) coordinates on some interval Ii is ℓ2n−λ2n/(4pℓ). That is,

**Proposition 7.** We can list a superset of all vertex subsets X that give non-zero contributions modulo p(1−λ)n/(2p) in lemma 6 in expected time 2n−Ω(n/(p log p)).

### 2.4 Chinese remaindering

Equipped with a 2n−Θ(n/(p log p)) time algorithm to compute hc(G) through hp(G, s, t) modulo p(1−λ)n/(2p) for small primes p, we can compute it modulo the product of all those prime powers for every small prime by employing the Chinese remainder theorem. We know from number theory (Mertens’ first theorem, see e.g. [25], Thm. 7) that for every $q > 2$,

$$\sum_{p \leq q \text{ prime}} \ln p/p \geq \ln q - 2.$$  

Thus using proposition 7 to compute hc(G) modulo $p(1−λ)n/(2p)$ via lemma 6 for each small prime p in turn, we can assemble the modular computations to compute the Hamiltonian cycles modulo

$$m = \prod_{p \leq q \text{ prime}} p(1−λ)n/(2p) = e^{(1−λ)n/2} \sum_{p \text{ prime} \leq q} \ln p/p \geq (e^{-2}q)^{(1−λ)n/2},$$  \hspace{1cm} (2)

in $2n−Ω(n/(q log q))$ expected time. This proves our counting result theorem 1.

### 3 Applications

We give three applications through variations of the techniques behind our counting theorem 1.

#### 3.1 Detecting a Hamiltonian cycle when they are few

In this section we prove theorem 2. When we know the graph has at most $d^n$ Hamiltonian cycles for some $d$, e.g. by approximating it from above by computing the permanent of the graph’s adjacency matrix by the FPTAS of Jerrum et al. [20], we can choose enough prime powers to build a smooth number $m > d^n$ by using all primes less than $q = e^{2d^2/(1−λ)}$ according to equation 2 in section 2.4. Now we can directly apply theorem 1 with this choice of $m$ to get theorem 2.

To get corollary 3 all we need to do is bound the number of Hamiltonian cycles. This number is at most $\prod d_i$ where $d_i$ is the outdegree of vertex $i$. Since the average degree $d = \frac{1}{n} \sum d_i$, we know from the inequality of arithmetic and geometric means that there are at most $d^n$ Hamiltonian cycles.
3.2 Counting modulo powers of two

In this section we prove theorem 4. We will again use lemma 6 to compute the number of Hamiltonian cycles modulo $2^k$ for some positive $k$. However, to list the $X$’s that contribute, we will use another routine than the one outlined in section 2.3. Instead we will note that $X$’s that give rise to non-zero terms modulo 2, fulfill the quadratic equations

$$(1 + x_i) \left( m_{is} + \sum_{ij \in A(t)} x_{ij} \right) = 1 + x_i \pmod{2},$$

simultaneously for all $i = 1, \ldots, n$. That is, at most one of $x_i$ and $\sum_{ij \in A(t)} x_{ij} + m_{is}$ is allowed to be zero (otherwise the $i$th column will be zero modulo two and the determinant gets an extra factor two). Means to list all such $X$’s were given in [4] (see section 4 of that paper), where the $m_{is}$ are also set deterministically in the process to get a $1.619^n$ deterministic time bound. By looping over all subsets $S$ of at most $k - 1$ equations and adding one to the right hand side of these equations, we can in the same way list all $X$’s whose terms are non-zero mod $2^{\lfloor S \rfloor + 1}$, in total $\sum_{i=0}^{k-1} \binom{n}{i} 1.619^n$ time.

3.3 A better Monte Carlo algorithm for bounded degree graphs

In this section we prove theorem 5. We will again employ lemma 6 but instead of counting modulo small primes, we will operate in a field $\mathbb{GF}(2^k)$ of characteristic 2. Here $k$ will be no larger than logarithmic in $n$.

Lemma 8. In every directed graph $G = (V, A)$ of in- plus outdegree bounded by $d$, there is a subset $W \subseteq V$ such that 1. the size $|W| \geq \frac{n}{d^2 + 1}$, and 2. for no pair $u \neq v \in W$, both $uw$ and $vw$ belong to $A$ for any $w \in V$. 

Furthermore, such a set $W$ can be found in polynomial time in $|V|$.

Proof. We consider the corresponding undirected graph $H$, where we have replaced each arc with an edge (if there are multiple edges we drop all except one). If we take the square $H^2$ of this graph, i.e. the graph where there is an edge between $u$ and $v$ if there exists $w \in V$ such that $(uw \in A) \land (wv \in A)$, we see that an independent set in $H^2$ obey the second requirement for a $W$. Since $H_2$ has degree bounded by $d^2$, we know from Turan’s theorem on the complement of the graph that there is an independent set in $H^2$ of size at least $n/(d^2 + 1)$, obeying the first requirement for a $W$. We can find it by greedily adding vertices to $W$, repeatedly choosing the one with smallest degree in the remaining graph, and for each chosen vertex discarding all its neighbors.

Our algorithm works as follows. First we find such an independent set $W$. Next we run the reduction in [22] with $a_i \in \{0, 1\}$ for all $i \in W$ to be set later but let the remaining $a_i = 0$ as they will play no role in this algorithm. Instead we add random “multiplicities” $m_{uv} \in \mathbb{GF}(2^k)$ for every arc origin in a vertex $u \in V \setminus W$, and let $m_{uv} = 1$ otherwise. Instead of listing the non-zero terms through the machinery in section 2.3, we will again use quadratic equation systems to describe the terms we are looking for. We construct

$$(1 + x_i) \left( m_{is} + \sum_{ij \in A(t)} x_{ij} \right) = 1 + x_i \pmod{2},$$
for all \(i \in W\), and use the deterministic 1.619\(|W|\) time algorithm from [4] to list the \(X\)'s that may be non-zero. In the process the multiplicities \(m_{is} = a_i\) will be set deterministically to values that make sure the number of listed terms is at most equal to the expected number of terms under a random assignment. This step turns the Las Vegas algorithm into deterministic one, but we will soon use another source of randomness leading to a Monte Carlo algorithm.

We next note that the sum equals

\[
\sum_{H \subseteq A(t)} \prod_{uv \in H} m_{uv} = \sum_{H \subseteq A(t)} \prod_{u \in V \setminus W} m_{uv}.
\]

Furthermore, each term on the right hand side above is unique when seen as a monomial in the variables \(\{m_{uv} : u \in V \setminus W\}\). This is because the properties of \(W\) guarantee that for any Hamiltonian cycle, there cannot be another Hamiltonian cycle going through exactly the same arcs in \(\{m_{uv} : u \in V \setminus W\}\). If there was another one, the vertices in \(W\) would have to change outgoing neighbors along the cycle in some permutation still leading to a single cycle, but there cannot be any other permutation of the neighbors since no pair of vertices in \(W\) shares any.

We use the fingerprint lemma known as the DeMillo-Lipton-Schwartz-Zippel lemma [15, 23]:

**Lemma 9.** For a non-zero multivariate polynomial \(p(x_1, \ldots, x_n)\) of degree \(d\) over a finite field \(F\) on \(|F|\) elements, the probability of a uniformly sampled point \((x_1, \ldots, x_n) \in F^n\) obeying \(p(x_1, \ldots, x_n) = 0\), is at most \(\frac{d}{|F|}\).

Hence, if we use random numbers from GF\((2^k)\) in place of the variables \(\{m_{uv} : u \in V \setminus W\}\) as we described above, from the lemma above it follows that with probability at least \(1 - \frac{n}{2^k}\) we will get a non-zero result iff there are Hamiltonian cycles in the graph. The running time is \(1.619\{|W|2^{n-|W|} = 2^n - \Omega(n/d^2)\}\) as claimed.

**Acknowledgments**

This work started when the authors met at the workshop “Satisfiability Lower Bounds and Tight Results for Parameterized and Exponential-Time Algorithm” hosted by the Simons institute for the theory of computing at Berkeley in November 2015. A.B. is supported by the Swedish Research Council grant VR 2012-4730 “Exact Exponential-time Algorithms”, and I.K. is supported by the NSF CAREER award CCF-1149048.

**References**

[1] E. Bax and J. Franklin. A permanent algorithm with \(\exp[\Omega(n^{1/3}/2\ln n)]\) expected speedup for 0–1 matrices. Algorithmica, Volume 32(1), pp. 157–162, 2002.

[2] R. Bellman. Dynamic programming treatment of the travelling salesman problem, J. Assoc. Comput. Mach. 9, pp. 61–63, 1962.

[3] A. Björklund. Determinant Sums for Undirected Hamiltonicity. SIAM J. Comp. 2014. Volume 43(1), pp. 280–299, 2014. Preliminary version in proceedings of IEEE FOCS, 2010.
[4] A. Björklund and T. Husfeldt. The Parity of Directed Hamiltonian Cycles. In proceedings of the 54th Annual Symposium on Foundations of Computer Science, FOCS 2013, Berkeley. IEEE Society, pp. 727–735, 2013.

[5] A. Björklund, T. Husfeldt, and M. Koivisto. Set partitioning via inclusion–exclusion. SIAM J. Comput. 2009, Volume 39(2), pp. 546–563, 2009.

[6] A. Björklund, T. Husfeldt, P. Kaski, and M. Koivisto. The traveling salesman problem in bounded degree graphs. ACM Transactions on Algorithms 8(2), 18, 2012.

[7] A. Björklund, H. Dell, and T. Husfeldt. The Parity of Set Systems under Random Restrictions with Applications to Exponential Time Problems. In proceedings of Automata, Languages, and Programming, volume 9134 of Lecture Notes in Computer Science, pp 231-243, 2015.

[8] A. Björklund, T. Husfeldt, and I. Lyckberg. Computing the Permanent modulo a Prime Power. Unpublished manuscript, 2015.

[9] A. Björklund. Below All Subsets for Some Permutational Counting Problems. In proceedings of SWAT, 2016.

[10] C. Calabro, R. Impagliazzo, V. Kabanets, and R. Paturi. The Complexity of Unique $k$-SAT: An Isolation Lemma for $k$-CNFs. Journal of Computer and System Sciences, vol 74, issue 3, pp. 386–393, 2008.

[11] M. Cygan, J. Nederlof, M. Pilipczuk, M. Pilipczuk, J. M. M. van Rooij, and J. O. Wojtaszczyk. Solving connectivity problems parameterized by treewidth in single exponential time. In proceedings of the 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs. IEEE Computer Society 2011, pp. 150–159, 2011.

[12] M. Cygan, H. Dell, D. Lokshtanov, D. Marx, J. Nederlof, Y. Okamoto, R. Paturi, S. Saurabh, M. Wahlström. On Problems as Hard as CNF–SAT. In Proceedings of the 2012 IEEE Conference on Computational Complexity, pp. 74–84, 2012.

[13] M. Cygan, S. Kratsch, and J. Nederlof. Fast hamiltonicity checking via bases of perfect matchings. In proceedings of the 45th annual ACM Symposium on Theory of computing, pp. 301–310, 2013.

[14] M. Cygan and M. Pilipczuk. Faster exponential-time algorithms in graphs of bounded average degree. In proceedings of Automata, Languages, and Programming, volume 7965 of Lecture Notes in Computer Science, pp. 364–375, 2013.

[15] R. A. DeMillo and R. J. Lipton, A probabilistic remark on algebraic program testing. Inform. Process Lett. 7, pp. 193–195, 1978.

[16] A. Gabizon, D. Lokshtanov, and M. Pilipczuk. Fast Algorithms for Parameterized Problems with Relaxed Disjointness Constraints. In proceedings of the 23rd Annual European Symposium on Algorithms, volume 9294 of Lecture Notes in Computer Science, pp. 545–556, 2015.

[17] M. Held and R. M. Karp. A dynamic programming approach to sequencing problems, J. Soc. Indust. Appl. Math. 10, pp. 196–210, 1962.
[18] R. Impagliazzo and R. Paturi. On the complexity of k-SAT. Journal of Computer and System Sciences, vol. 62, no. 2, pp. 367–375, 2001.

[19] T. Izumi, and T. Wadayama. A new direction for counting perfect matchings. In proceedings of the 53rd Annual Symposium on Foundations of Computer Science, FOCS 2012. New Brunswick. IEEE Computer Society, pp. 591–598, 2012.

[20] M. Jerrum, A. Sinclair, and E. Vigoda. A polynomial-time approximation algorithm for the permanent of a matrix with non-negative entries. In proceedings of the 33rd ACM Symposium on Theory of Computing, pp. 712–721, doi:10.1145/380752.380877, (also as ECCC TR00-079), 2001.

[21] R. M. Karp. Reducibility Among Combinatorial Problems. Complexity of Computer Computations, pp. 85–103, 1972.

[22] G. Royle and C. Godsil. Algebraic Graph Theory. Graduate Texts in Mathematics. Springer Verlag, 1997.

[23] J. T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. J. Assoc. Comput. Mach. 27, pp 701–717, 1980.

[24] R. Schuler. An algorithm for the satisfiability problem of formulas in conjunctive normal form. Journal of Algorithms, Vol. 54, no. 1, pp. 40–44, 2005.

[25] G. Tenenbaum. Introduction to Analytic and Probabilistic Number Theory. Cambridge University Press, 1995.