2013 IEEE 54th Annual Symposium on Foundations of Computer Science

The Parity of Directed Hamiltonian Cycles

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

Thore Husfeldt
Department of Computer Science, Lund University, Sweden
IT University of Copenhagen, Denmark
thore.husfeldt@cs.lth.se

***Abstract*—We present a deterministic algorithm that given any directed graph on n vertices computes the parity of its number of Hamiltonian cycles in $O(1.619^n)$ time and polynomial space. For bipartite graphs, we give a $1.5^n poly(n)$ expected time algorithm.**

Our algorithms are based on a new combinatorial formula for the number of Hamiltonian cycles modulo a positive integer.

I. Introduction

It is known since the 1960s that Hamiltonian cycles in an n-vertex graph can be detected and counted in $O(2^n n^2)$ time [1], [9]. In an influential survey, Woeginger [12] asked if this could be significantly improved. Recently, two results using different techniques have met this challenge:

1) a randomized $O(1.657^n)$ time algorithm for the decision problem in undirected graphs [3],
2) a randomized $O(1.888^n)$ time algorithm for the decision problem in directed bipartite graphs [6].

For the general directed graph case, no such algorithm has yet appeared.

Intriguingly, the foundation for the construction from [6] is an algorithm for the related $\oplus$P-complete problem of computing the parity of the number of Hamiltonian cycles (in a bipartite graph). Our contribution in the present paper is to compute this value (for any directed graph) in time within a polynomial factor of $O(\phi^n)$, where $\phi = \frac{1}{2}(1+\sqrt{5}) < 1.619$ is the golden ratio.

While this does not seem to shed any direct light on the decision problem in directed graphs, it does however ask a related interesting question: Could it really be easier to solve a $\oplus$P-complete problem than its decision counterpart? There is still no known $(2-\Omega(1))^n$ time algorithm for the decision problem and even the recent decision algorithm for undirected graphs [3] is slower than our present algorithm.

In contrast, current evidence for the well-studied CNF Satisfiability problem points in the opposite direction: It is known that a fast algorithm for computing the parity of the number of satisfiable assignments to a CNF formula on n Boolean variables would disprove the Strong Exponential Time Hypothesis [5], i.e. a $(2-\Omega(1))^n$ time algorithm computing the parity of the satisfying assignments implies a $(2-\Omega(1))^n$ time algorithm for deciding if a formula has a satisfying assignment at all.

A. Results

Let $\mathscr{H}$ denote the set of (directed) Hamiltonian cycles of a directed input graph $G = (V, E)$ on $n = |V|$ vertices.

Our main result is that the parity $\oplus\mathscr{H}$ of the number of directed Hamiltonian cycles can be computed much faster than in 2^n time.

Theorem 1. *We can compute $\oplus\mathscr{H}$ for a general n-vertex digraph in time within a polynomial factor of the nth Fibonacci number $F_n \in O(1.619^n)$ and polynomial space.*

For the restricted family of bipartite graphs, we can show a stronger bound. This result is also somewhat easier to prove.

Theorem 2. *We can compute $\oplus\mathscr{H}$ for a bipartite n-vertex digraph in expected time within a polynomial factor of $O(1.5^n)$ and polynomial space.*

Both results rely on a new characterization of the number of Hamiltonian cycles in terms of a local property. We use the notation $a \equiv_K b$ to mean $a \equiv b \pmod K$

Theorem 3. *For a vertex subset X, let $d_v(X)$ denote the number of directed edges from vertex v to a vertex in X. For integer $K \geq 2$,*

$$|\mathscr{H}| \equiv_K \frac{1}{K} \sum_{Z, Y_1, \ldots, Y_K} (-1)^{|V \setminus Z|} \Big(\prod_{z \in Z} d_z(V \setminus Z) \Big) \prod_{k=1}^{K} \Big(\prod_{y \in Y_k} d_y(Y_k) \Big),$$

where the sum is over all $(K+1)$-partitions of V.

Our algorithms use this result for $K = 2$, but we state (and prove) it for general K.

B. Related results

There are nontrivial examples of hard counting problems where computing the parity is easy. In fact an important example is given by a closely related problem: computing the number $\#\mathscr{C}$ of disjoint cycle covers in a directed graph. This is a hard counting problem, equivalent to computing the permanent of the adjacency matrix. On the other hand,

0272-5428/13 $26.00 © 2013 IEEE
DOI 10.1109/FOCS.2013.83

the parity $\oplus\mathscr{C}$ equals the parity of the determinant and is therefore computable in polynomial time by Gaussian elimination. However, the problem studied in the present paper, $\oplus\mathscr{H}$, is in fact complete for the complexity class $\oplus$P, even for very restricted classes of graphs [11].

Counting the number of Hamiltonian cycles in an n-vertex directed graph can be done in $o(2^n)$ time [4]. However, the improvement over the classic algorithms [1], [9] is relatively small (within a factor of $\exp(O(\sqrt{n/\log n}))$). The parity problem appears to be easier. First, for undirected graphs it is known that when all vertex degrees are odd, there is an even number of Hamiltonian cycles passing through each (directed) edge [10]. Second, a recent paper of Cygan, Kratsch, and Nederlof [6] includes a deterministic $O(1.888^n)$ time, exponential space algorithm computing the parity of the number of Hamiltonian cycles for undirected and directed bipartite graphs. Our present algorithm is faster, uses only polynomial space, and works for unrestricted directed graphs as well. On the other hand, the algorithm from [6] works for weighted graphs. This is a crucial property because it allows the use of the Isolation lemma to construct a (randomized) decision algorithm. Our constructions do not seem to allow this extension.

C. Open questions

Our paper accentuates the fact that the exponential time complexity of counting or deciding the Hamiltonian cycles in a directed graph is not very well understood. While there are several examples of $\oplus$P-complete problems whose decision analogue is computationally easier (e.g., $\oplus$2-Satisfiability), examples of the converse are not known to the authors.

Under the Exponential Time Hypothesis, the decision problem does not allow $\exp(o(n))$-time algorithms, but there are no arguments for or against an $O(1.999^n)$-time algorithm for the decision problem.

For comparison, it is now known that the parity of the number of set covers of a given set system on n elements cannot be solved in time $(2-\Omega(1))^n$ under the Strong Exponential Time Hypothesis. [5].

D. Overview and techniques

Our algorithm is based on evaluating theorem 3 with $K = 2$, which is a summation of all 3-partitions of G.

There are two parts to our algorithm.

First, the vast majority of the terms in the summation can be made to vanish modulo 2. Our main algorithmic insight is the fact that any set of self-loops, and hence in particular a random set, can be added to the vertices of a Hamiltonicity instance without changing the result. This idea of randomly modifying the input instance in order to produce a large number of zero-valued terms is inspired by an algorithm for computing the permanent by Bax and Franklin [2]. After this modification, we can avoid explicitly listing the vanishing terms. The running time of the algorithm is based on a careful orchestration of a total search among linear equation systems, which happens in sections III and IV for the bipartite and the general input case, respectively.

Second, the contribution of the nonvanishing terms can be computed in polynomial time via linear algebra, see section II.

Finally, in section V we give the proof of theorem 3, the combinatorial core of our contribution. This argument at least follows a well-trodden path, if only in the beginning; the cut-and-count method [7] provides a connection between Hamiltonicity and cycle covers, and the principle of inclusion–exclusion is used to move from sums over permutations to sums of functions.

II. Algorithm

We begin by rewriting the expression from theorem 3. First, for $K = 2$, we get

$$\oplus\mathscr{H} \equiv_2 \tfrac{1}{2}\sum_{X,Y,Z}\Big(\prod_{x\in X} d_x(X)\Big)\Big(\prod_{y\in Y} d_y(Y)\Big)\Big(\prod_{z\in Z} d_z(\overline{Z})\Big), \quad (1)$$

where we have introduced $X = Y_1$, $Y = Y_2$ and let $\overline{Z}$ denote the vertex complement $\overline{Z} = V \setminus Z$. Note in particular that we removed the factor $(-1)^{|\overline{Z}|}$, because $-1 \equiv +1 \pmod 2$, and even though the mod 2 operation is applied after the division by 2, every contributing Z will be counted an even number of times as seen by changing the roles of X and Y.

It will be convenient to remove the factor $\frac{1}{2}$ altogether. To this end, consider a binary relation $\prec$ on the subsets of V such that for disjoint $X,Y \subseteq V$, not both empty, either $X \prec Y$ or $Y \prec X$. Then (1) is equivalent to

$$\oplus\mathscr{H} \equiv_2 \sum_{\substack{X,Y,Z\\ X\prec Y}}\Big(\prod_{x\in X} d_x(X)\Big)\Big(\prod_{y\in Y} d_y(Y)\Big)\Big(\prod_{z\in Z} d_z(\overline{Z})\Big). \quad (2)$$

To see this, first observe that (1) is symmetric with respect to X and Y. Furthermore, in a contributing term, the sets X and Y cannot both be empty, because for $Z = V$ the term $\prod_{z\in Z} d_z(\overline{Z})$ vanishes. Thus for every partition (X,Y,Z) with $X \prec Y$ that contributes to (2), there is a twin contribution to (1): the distinct partition (Y,X,Z), which has $Y \prec X$ and in particular $Y \neq X$.

Finally, we separate the expression into the contributions of X and $Y \cup Z$, respectively:

$$\oplus\mathscr{H} \equiv_2 \sum_{X} f(X)\prod_{x\in X} d_x(X)\,, \quad (3)$$

where

$$f(X) = \sum_{\substack{Y,Z\\ X\prec Y}}\Big(\prod_{y\in Y} d_y(Y)\Big)\prod_{z\in Z} d_z(\overline{Z})\,. \quad (4)$$

It is understood that the sum in (3) is over all $X \subseteq V$ and the sum in (4) is over all partitions Y,Z of $V \setminus X$. Still, the whole expression is over almost all 3–partitions X,Y,Z

of the vertices and thus has $\Omega(3^n)$ terms, so it does not in itself serve as a fast algorithm.

There are two parts to our algorithm. First, for many X, already the factor $\prod_{x\in X} d_x(X)$ in (3) will be zero modulo 2. Hence, these subsets contribute nothing and we will avoid explicitly listing them. Second, for any X for which it is not zero, the value of $f(X)$ modulo 2 can be computed in polynomial time in $|V|$ via linear algebra.

On the top level our algorithm is simply an evaluation of (3) and (4).

Algorithm P (*Parity.*) Given a directed graph $G = (V,E)$, computes $\oplus\mathscr{H}$.

P1 [Initialise.] Set $s = 0$.

P2 [Locate.] List every $X \subseteq V$ such that $\prod_{x\in X} d_x(X)$ is odd.

P3 [Contribute.] Compute $f(X) \pmod 2$ for every such X and add it to s.

P4 [Report.] Return $s \pmod 2$.

Note that steps P2 and P3 can be interleaved in order to avoid storing all X.

We proceed by explaining how to execute steps P2 and P3 efficiently.

A. Step P2: Locating Subsets that Contribute

We call a subset $X \subseteq V$ *contributing* if $\prod_{x\in X} d_x(X)$ is odd. Our aim is to generate the contributing subsets. We identify V with $\{1,\ldots,n\}$ and introduce the indicator variables $x_1,\ldots,x_n \in \{0,1\}$ for $X \subseteq V$ with $x_i = 1$ if and only if $i \in X$. Then the constraint that $d_i(X)$ is odd can be expressed in terms of the adjacency matrix A of G as

$$\sum_{j=1}^{n} a_{ij}x_j \equiv_2 1 \qquad \text{for all } i \in X\,,$$

equivalently,

$$x_i\Big(\sum_{j=1}^{n} a_{ij}x_j\Big) \equiv_2 x_i \qquad \text{for all } i = 1,\ldots,n\,.$$

We will view these constraints as a system of n quadratic equations over GF(2) in the variables $x_1,\ldots,x_n$:

$$\begin{aligned} x_1(a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n) &= x_1\\ x_2(a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n) &= x_2\\ &\vdots\\ x_n(a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n) &= x_n\,. \end{aligned}$$

This system can be succinctly expressed as a matrix equation over GF(2):

$$\mathbf{x} \circ A\mathbf{x} = \mathbf{x} \tag{5}$$

where $\mathbf{x} = (x_1,\ldots,x_n)$ and the operator $\circ$ denotes the coordinate-wise (or Schur, or Hadamard) product.

B. Random self-loops

A priori, listing the solutions to (5) requires 2^n steps. In fact, already the size of the solution set is easily seen to be of order 2^n for some graphs. For instance, every vector $\mathbf{x}$ with an odd number of 1s solves (5) if A is the all-1s matrix corresponding to a directed clique with self-loops.

This motivates the main algorithmic insight in our construction. Because no Hamiltonian cycle can use a self-loop, we can add self-loops to the vertices in G (or remove them) without changing $\mathscr{H}$. Algebraically, manipulation of self-loops corresponds to flipping the diagonal entries of A.

We show that adding these loops at random reduces the expected solution size for all graphs.

Lemma 1. *Let A be an $n \times n$ matrix with $0,1$-entries. Choose the diagonal entries $a_{11},\ldots,a_{nn} \in \{0,1\}$ uniformly and independently at random. Then the expected number of solutions to* (5) *is* 1.5^n.

Proof: Consider the ith entry of $\mathbf{x}$. If $x_i = 0$ then the ith equation in (5) is $0 = 0$ and trivially satisfied. If $x_i = 1$ then the ith equation in (5) is satisfied if

$$1 = \sum_{j=1}^{n} a_{ij}x_j = a_{ii}\cdot 1 + \sum_{\substack{j=1\\ j\neq i}}^{n} a_{ij}x_j\,,$$

which happens with probability $\frac{1}{2}$ because a_{ii} is uniformly distributed. By independence of the choices of a_{ii}, a vector $\mathbf{x}$ with k 1s satisfies all the equations with probability 2^{-k}. By linearity of expectation, the expected number of solutions is

$$\sum_{k=0}^{n} \binom{n}{k} 2^{-k} = (1+\tfrac{1}{2})^n\,,$$

using the binomial theorem. ∎

In section III we show that these 1.5^n solutions can be efficiently listed if G is bipartite. In section IV we show that the solutions can be listed in time $O(1.619^n)$ for general graphs.

C. Gaussian elimination

We assume that the following algorithm for solving nonhomogeneous systems of linear equations is well known. We repeat it here only to recall that the solution set can be described in terms of the basis of a translated vector space (the system's *null space*). In particular, if a linear equation system has a solution at all, the number of solutions equals 2^d, where d is the dimension of the null space.

Algorithm G (*Gaussian elimination.*) Given a nonhomogeneous system $Ax = c$ of linear equations. If the system has no solution, outputs "no". Otherwise, outputs a vector v and d linearly independent vectors $b_1,\ldots,b_d$ such that the set of vectors x with $Ax = c$ equals $\{v + \operatorname{span}(b_1,\ldots,b_d)\}$.

Gaussian elimination runs in polynomial time and works over finite fields.

D. Step P3: Computing the Contributions

We return to the value $f(X)$ in (4) and show how to compute it modulo 2 in polynomial time. The salient feature of (4) is that modulo 2, we can now view $f(X)$ as the number of solutions to an equation system over GF(2).

We define the binary relation $\prec$ on the subsets of V as follows. Identify V with $\{1,\dots,n\}$ and set $X \prec Y$ if $\min X < \min Y$, with the usual convention $\min \emptyset = \infty$. (By the discussion following (2), the case $X = Y = \emptyset$ never arises.)

We will set up a system of linear equations in $y_1,\dots,y_n$, so that $f(X)$ equals the number of solutions (mod 2). We use the obvious correspondence

$$y_i = \begin{cases} 1\,, & \text{if } i \in Y\,; \\ 0\,, & \text{if } i \notin Y\,. \end{cases}$$

First, since Y belongs to the complement of X we introduce the equations

$$y_i = 0 \qquad \text{for } i \in X\,. \tag{6}$$

Second, to ensure $X \prec Y$ we introduce the equations

$$y_i = 0, \qquad \text{for } i = 1,\dots,\min X\,. \tag{7}$$

Finally, a subset Y with $X \prec Y$ contributes to $f(X)$ if all product terms in (4) are 1, so we need

$$\begin{aligned} d_i(Y) &\equiv_2 1\,, \qquad \text{if } i \in Y\,; \\ d_i(X \cup Y) &\equiv_2 1\,, \qquad \text{if } i \notin Y \end{aligned}$$

for each $i \notin X$. Since $d_i(X \cup Y) = d_i(X) + d_i(Y)$ we can rewrite the second constraint to

$$d_i(Y) \equiv_2 1 + d_i(X), \qquad \text{if } i \notin Y\,.$$

In terms of the variables $y_1,\dots,y_n$ and the adjacency matrix A of G we have the equations

$$\sum_{j=1}^{n} a_{ij} y_j = \begin{cases} 1\,, & \text{if } y_i = 1\,; \\ 1 + d_i(X)\,, & \text{if } y_i = 0\,; \end{cases}$$

for each $i \notin X$. (Note that $d_i(X)$ is a fixed value depending only on X.) The right hand side simplifies to $1 + d_i(X) + d_i(X) y_i$, so the resulting equations are

$$d_i(X) y_i + \sum_{j=1}^{n} a_{ij} y_j = 1 + d_i(X), \qquad \text{for } i \notin X\,. \tag{8}$$

In summary, the solutions to the linear equations in (6), (7), and (8) describe exactly the subsets $Y \subseteq V \setminus X$ with $X \prec Y$ that contribute 1 to $f(X)$. The number of such solutions is 2^d with d the dimension of the system's null space. It is odd if and only if the system of equations has a unique solution (and consequently $d = 0$). This is determined by Gaussian elimination in time polynomial in n.

III. Bipartite Input Graphs

We first describe an algorithm for the step P2 for the case that the input graph is bipartite. We can safely assume that the input bipartite graph is balanced on two equal sized vertex sets $V_1 \cup V_2 = V$. (Otherwise, there can be no Hamiltonian cycles.)

Let

$$B = \begin{bmatrix} 0 & B_{12} \\ B_{21} & 0 \end{bmatrix}$$

denote the adjacency matrix of the input graph. Choose random values $r_1,\dots,r_n \in \{0,1\}$ uniformly and independently and construct the diagonal matrix $R = \operatorname{diag}(r_1,\dots,r_n)$. Set $A = B + R$. The equation system (5) becomes

$$\mathbf{x} \circ (B + R)\mathbf{x} = \mathbf{x} \circ \begin{bmatrix} 0 & B_{12} \\ B_{21} & 0 \end{bmatrix} \mathbf{x} + \mathbf{x} \circ R\mathbf{x} = \mathbf{x}\,,$$

equivalently,

$$\begin{aligned} \mathbf{x}_1 \circ B_{12}\mathbf{x}_2 &= \mathbf{x}_1 - \mathbf{x}_1 \circ R_1\mathbf{x}_1 \\ \mathbf{x}_2 \circ B_{21}\mathbf{x}_1 &= \mathbf{x}_2 - \mathbf{x}_2 \circ R_2\mathbf{x}_2\,, \end{aligned}$$

where $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2)$ and $R = \operatorname{diag}(R_1, R_2)$. In this formulation it is apparent that for fixed $\mathbf{x}_1$, the equation system (in the variables $\mathbf{x}_2$) is linear; note in particular that $\mathbf{x}_2 \circ R_2\mathbf{x}_2 = R_2\mathbf{x}_2$ because R_2 is diagonal and multiplication in GF(2) is idempotent. Thus, the system can be solved by Gaussian elimination in polynomial time.

Algorithm B (*Bipartite graphs.*) Given a bipartite directed graph $G = (V, E)$ with $n = |V|$, lists the vertex subsets $X \subseteq V$ for which $d_x(X)$ is odd for all $x \in X$.

B1 [Add random self loops.] Choose $r_1,\dots,r_n \in \{0,1\}$ uniformly at random and add a self-loop at vertex v_i if $r_i = 1$.

B2 [Initialise.] Set $x_1 = \cdots = x_{n/2} = 0$.

B3 [Solve.] Solve the system $\mathbf{x} \circ A\mathbf{x} = \mathbf{x}$ in the free variables $x_{n/2+1},\dots,x_n$ using algorithm **G**. If it returns "none" proceed to B5.

B4 [Report.] (Algorithm **G** returned the solution set as $v, b_1,\dots,b_d$.) Output the vectors $v + \alpha_1 b_1 + \cdots + \alpha_d b_d$ for all 2^d choices of $\alpha_1,\cdots,\alpha_d \in \{0,1\}$.

B5 [Next.] If all choices of $(x_1,\dots,x_{n/2})$ have been inspected, terminate. Otherwise generate the next choice and return to B3.

The number of choices of $\mathbf{x}_1$ is $2^{n/2}$, so step B3 takes total time $2^{n/2}\mathit{poly}(n)$. This is dominated by the overall running time of step B4, which is linear in the number of of contributing subsets X. These are 1.5^n in expectation according to lemma 1. Hence the runtime is $1.5^n\mathit{poly}(n)$ in expectation. This finishes the proof of theorem 2.

IV. General Input Graphs

For general graphs, we do not know how to list the solutions S to (5) in step P2 in time proportional to the solution set. Instead, we will efficiently list a superset S' of the solution set of size $O(1.619^n)$. We can then examine every $\mathbf{x} \in S'$ to see that it solves (5).

The superset S' of candidate solutions to (5) is defined as follows. Let $\mathbf{x}'$ denote a *prefix* vector of fixed values $\mathbf{x}' = (x_1, \ldots, x_{n-k}) \in \{0,1\}^{n-k}$ and introduce the variables $x_{n-k+1}, \ldots, x_n$. Set $\mathbf{x} = (x_1, \ldots, x_n)$. Consider the equation system

$$(\mathbf{x}', \mathbf{0}_k) \circ A\mathbf{x} = (\mathbf{x}', \mathbf{0}_k)\,. \tag{9}$$

For fixed $\mathbf{x}'$, this is nonhomogeneous set of linear equations in the variables $x_{n-k+1}, \ldots, x_n$. Every prefix of a solution to (5) is a solution to (9):

Lemma 2. *Let* $\mathbf{x} = (x_1, \ldots, x_n) \in \{0,1\}^n$ *and* $\mathbf{x}' = (x_1, \ldots, x_{n-k})$ *for some* $k \in \{0, 1, \ldots, n\}$. *If* $\mathbf{x}$ *satisfies* $\mathbf{x} \circ A\mathbf{x} = \mathbf{x}$ *then* $(\mathbf{x}', \mathbf{0}_k) \circ A\mathbf{x} = (\mathbf{x}', \mathbf{0}_k)$.

Proof: For $1 \le i \le n-k$, the ith equation of both $\mathbf{x} \circ A\mathbf{x} = \mathbf{x}$ and $(\mathbf{x}', \mathbf{0}_k) \circ A\mathbf{x} = (\mathbf{x}', \mathbf{0}_k)$ is

$$x_i \sum_{j=1}^{n} a_{ij} x_j = x_i\,.$$

For $i > n-k$, the ith equation of $(\mathbf{x}', \mathbf{0}_k) \circ A\mathbf{x} = (\mathbf{x}', \mathbf{0}_k)$ simplifies to $0 = 0$, which is trivially satisfied. ∎

We will show that we can avoid generating all prefixes.

Let $\binom{[n-k]}{k}$ denote the family of vectors $(x_1, \ldots, x_{n-k}) \in \{0,1\}^{n-k}$ for which $x_1 + \cdots + x_{n-k} = k$. Let $\binom{[n-k-1]}{k}_1$ denote the family of vectors $(x_1, \ldots, x_{n-k}) \in \{0,1\}^{n-k}$ for which $x_1 + \cdots + x_{n-k-1} = k$ and $x_{n-k} = 1$. The notation is motivated by the cardinalities,

$$\left|\binom{[n-k]}{k}\right| = \binom{n-k}{k}$$

and

$$\left|\binom{[n-k-1]}{k}_1\right| = \binom{n-k-1}{k}.$$

Lemma 3. *Let* $\mathbf{x} \in \{0,1\}^n$. *Then there is exactly one* $k \in \{0, \ldots, \lfloor n/2 \rfloor\}$ *such that the* $(n-k)$*th prefix* $\mathbf{x}' = (x_1, \ldots, x_{n-k})$ *belongs to* $\binom{[n-k]}{k}$ *or* $\binom{[n-k-1]}{k}_1$.

Proof: Let $w_k = x_1 + \cdots + x_{n-k}$. Choose the smallest $k \ge 0$ with the property

(*i*) $w_k = k$, or
(*ii*) $w_k = k+1$, $x_{n-k} = 1$.

To see that such k exists consider the pairs of values (w_k, k) for $k = 0, 1, \ldots$. The first pair is $(x_1 + \cdots + x_n, 0)$. The left value decreases monotonically by 0 or 1 down to 0. The right value increases $0, 1, 2, \ldots$. The only possibility that (*i*) lacks a solutions is that w_k and k pass each other, i.e., there exists k such that

$$w_k = k+1, w_{k+1} = k\,.$$

But in that case $x_{n-k} = 1$ so that (*ii*) is satisfied. Note also that if (*ii*) holds for k, then (*i*) does not for that k, and *vice versa*. Also note that k is at most $\lfloor n/2 \rfloor$, since w_k is at most $n-k$.

We have that either (*i*) is satisfied and $\mathbf{x}'$ belongs to $\binom{[n-k]}{k}$, or (*ii*) is satisfied and $\mathbf{x}'$ belongs to $\binom{[n-k-1]}{k}_1$.

We claim that there cannot be another solution $k' > k$ to either (*i*) or (*ii*). Consider first the case that k' satisfies (*i*) and k satisfies (*ii*). In particular, $k+1 = w_k = w_{k-1} + x_{n-k} = w_{k-1} + 1$, so $k = w_{k-1}$. Then,

$$k' = w_{k'} \le w_{k-1} = k\,,$$

a contradiction. The remaining three cases follow from the monotonicity of the left and right hand sides. ∎

We note in passing that the previous lemma gives a combinatorial proof of the identity

$$\sum_k \left(\binom{n-k}{k} + \binom{n-k-1}{k}\right) 2^k = 2^n\,,$$

see figure 1. For another way to verify this expression, compute

$$\sum_k \binom{N-k}{k} 2^k = \tfrac{1}{3}(2^{N+1} + (-1)^{N+1})\,,$$

which comes from evaluating a known, closed form for the generating function $\sum_k \binom{N-k}{k} z^k$ at $z = 2$ [8, exercise 1.2.9.15]. Then add these values for $N = n$ and $N = n-1$ to arrive at $\frac{1}{3}(2^{n+1} + (-1)^{n+1} + 2^n + (-1)^n) = \frac{1}{3}(2+1)2^n = 2^n$.

Finally, we need to bound the number of solutions to (9). *A priori*, this is a system in k variables each ranging over $\{0,1\}$, so the number of solutions can be 2^k. However, we will show that in expectation, the number of solutions is exactly 1:

Lemma 4. *Let* $\mathbf{x}'$ *denote a vector of fixed values* $\mathbf{x}' = (x_1, \ldots, x_{n-k})$ *from* $\binom{[n-k]}{k}$ *or* $\binom{[n-k-1]}{k}_1$ *and introduce the variables* $x_{n-k+1}, \ldots, x_n \in \{0,1\}$. *Set* $\mathbf{x} = (x_1 \ldots, x_n)$. *Let* A *be a matrix whose diagonal entries are chosen uniformly at random from* $\{0,1\}$. *Then the expected number of solutions to*

$$(\mathbf{x}', \mathbf{0}_k) \circ A\mathbf{x} = (\mathbf{x}', \mathbf{0}_k)$$

is 1.

Proof: Set $r_i = a_{ii}$. For $0 \le i \le n-k$, the ith equation has the form

$$x_i \sum_{j=1}^{n} a_{ij} x_j = x_i\,.$$

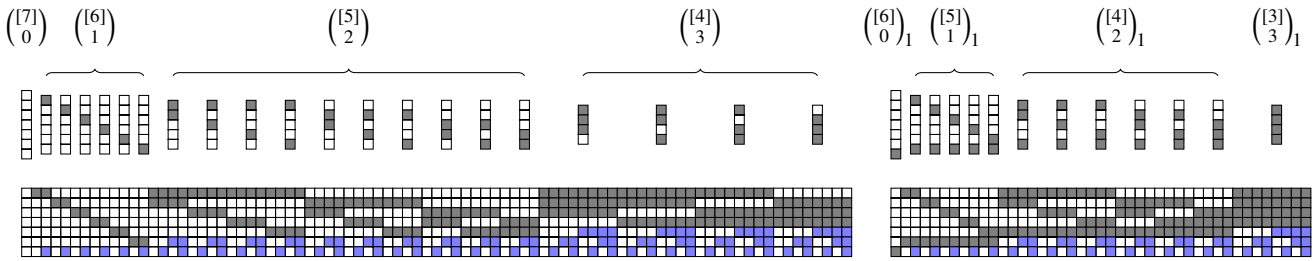

Figure 1. The sets $\binom{[n-k]}{k}$ and $\binom{[n-k-1]}{k}_1$ for $n = 7$ and $k \in \{0, \ldots, \lfloor n/2 \rfloor\}$. The bottom row shows all 128 bit patterns on 7 bits.

Since there are exactly k entries $x_i = 1$, there are exactly k of such equations that do not trivialise to $0 = 0$. We can rewrite each of these equations to

$$\sum_{j=1}^{n} a_{ij} x_j = 1$$

or, isolating the variables on the left hand side,

$$\sum_{j=n-k+1}^{n} a_{ij} x_j = 1 + r_i + \sum_{\substack{j=1 \\ j \neq i}}^{n-k} a_{ij} x_j \,.$$

Now it is clear that the system consists of k equations in the k variables $x_{n-k+1}, \ldots, x_n$. The right hand side is uniformly distributed from $\{0,1\}$, so an assignment to the variables satisfies an equation with probability $\frac{1}{2}$. By independence of the choices of r_i, all k equations are satisfied with probability 2^{-k}. There are 2^k assignments, so the expected number of solutions is $2^k 2^{-k} = 1$. ∎

Algorithm C (*Contributing subsets for general graphs.*) Given a graph $G = (V,E)$ with $n = |V|$, lists the vertex subsets $X \subseteq V$ for which $d_x(X)$ is odd for all $x \in X$.

C1 [Add random self loops.] Choose $r_1, \ldots, r_n \in \{0,1\}$ uniformly at random and add a self-loop at vertex v_i if $r_i = 1$. Let A denote the adjacency matrix of the resulting graph.

C2 [Initialize.] Set $k = 0$, $\mathbf{x}' = \mathbf{0}_n$.

C3 [Solve.] Solve the nonhomogeneous linear equation (9) in the free variables $x_{n-k+1}, \ldots, x_n$ using algorithm **G**. If there is no solution, proceed to C5.

C4 [Filter solution.] Generate every vector $\mathbf{x} = v + \alpha_1 b_1 + \cdots + \alpha_d b_d$ for all 2^d choices of $\alpha_1, \cdots, \alpha_d \in \{0,1\}$. If $\mathbf{x}$ solves the equation $\mathbf{x} \circ A\mathbf{x} = \mathbf{x}$, then output $\mathbf{x}$.

C5 [Next prefix.] Update $\mathbf{x}'$ and possibly k so that $\mathbf{x}'$ is the next element of $\binom{[n-k]}{k}$ or $\binom{[n-k-1]}{k}_1$ for $k = 0, \ldots, \frac{1}{2}n$, and return to C3. (When all these elements have been generated, the algorithm terminates.)

Lemma 5. *The running time of algorithm C is within a polynomial factor of*

$$\left(\frac{1+\sqrt{5}}{2}\right)^n \tag{10}$$

in expectation.

Proof: For the running time, first observe that the members of $\binom{[n-k]}{k}$ and $\binom{[n-k-1]}{k}_1$ can be listed with polynomial (in fact, constant) delay [8, sec. 7.2.1.3].

The number of iterations is given by the sizes of $\binom{[n-k]}{k}$ and $\binom{[n-k-1]}{k}_1$, summed over all $k = 0, \ldots, \lfloor n/2 \rfloor$. These are well-studied quantities, and it is known [8, ex. 1.2.8.16] that

$$\sum_k \binom{n-k}{k} + \sum_k \binom{n-k-1}{k} = F_{n+1} + F_n = F_{n+2} \,,$$

where F_n is the nth Fibonacci number, bounded by (10).

The running time is dominated by the time spent in steps C3 and C4. Step C3 takes polynomial time every time it is executed thanks to the polynomial running time of algorithm G. Step C4 looks more difficult to bound, because it may exhaust the entire solution space to an equation system in k unknowns. Naively, this would lead to a total running time of

$$\sum_k \binom{n-k}{k} 2^k > 2^{n-1} \,.$$

However, by lemma 4, the expected size of this solution space is 1. Thus, we expect to generate and verify only one vector $\mathbf{x}$, so the time spent in C4 is polynomial in expectation in each iteration. By linearity of expectation, the total time spent in C4 is within a polynomial factor of (10). ∎

A. Derandomization

We use the method of conditional expectations to derandomize algorithm C. Our randomness is over the choices of $r_1, \ldots, r_n$, the self-loop indicators. The expected number of candidate solutions S given a specific choice of values for the first k choices satisfies

$$\mathbf{E}[\, S \mid (r_1, \ldots, r_k) \,] = \frac{1}{2}\mathbf{E}[\, S \mid (r_1, \ldots, r_k, 0) \,] + \frac{1}{2}\mathbf{E}[\, S \mid (r_1, \ldots, r_k, 1) \,]$$

for all $k \geq 0$. In particular, there is a specific assignment to $r_1, \ldots, r_n$ such that $\mathbf{E}[\, S \mid (r_1, \ldots, r_n) \,] \leq \mathbf{E}[S]$. Thus, by mimicking the behaviour of algorithm C, but computing the expected solution sizes of the systems (9) instead of solving

them, we can choose the assignment greedily one variable at the time, until we get rid of all the randomness. The final assignment has no more candidate solutions than the original expected number.

The next lemma shows that we can efficiently compute the expectations $\mathbf{E}[\, S \mid (r_1,\ldots,r_k,0)\,]$ and $\mathbf{E}[\, S \mid (r_1,\ldots,r_k,1)\,]$.

Lemma 6. *Let* $\mathbf{x}'$ *and* $\mathbf{x}$ *be an in lemma 4. Let* $l \in \{0,\ldots,n\}$. *Let* A *be a matrix whose diagonal entries* $a_{ii} = r_i$ *for* $l < i \le n$ *are chosen uniformly at random from* $\{0,1\}$. *Then the expected number of solutions to*

$$(\mathbf{x}', \mathbf{0}_k) \circ A\mathbf{x} = (\mathbf{x}', \mathbf{0}_k)$$

is given by a polynomial size linear equation system in the variables $x_{n-k+1},\ldots,x_n$ *and* $r_{l+1},\ldots,r_n$.

Proof: The only contributing equations are those for which $x_i = 1$, of the form

$$1 \cdot \sum_{j=1}^{n} a_{ij}x_j = 1\,.$$

We can rewrite these equations as

$$\sum_{j=n-k+1}^{n} a_{ij}x_j = 1 + \sum_{j=1}^{n-k} a_{ij}x_j$$

for $i \le l$, and

$$r_i + \sum_{j=n-k+1}^{n} a_{ij}x_j = 1 + \sum_{\substack{j=1\\ j\neq i}}^{n-k} a_{ij}x_j$$

for $i > l$. Now it is clear that the system consists of k equations in the $k + (n-l)$ variables $x_{n-k+1},\ldots,x_n$ and $r_{l+1},\ldots,r_n$.

If this system has S solutions then the expected number of solutions to (9) is $S/2^{n-l+1}$. ∎

Algorithm D (*Derandomization of C.*) Given a graph $G = (V,E)$ with $n = |V|$, determines values $r_1,\ldots,r_n \in \{0,1\}$ for algorithm C. Let A denote the adjacency matrix of G.

D1 [Initialize.] Set $l = 1$.

D2 [Initialize.] Set $k = 0$, $\mathbf{x}' = \mathbf{0}_n$, $N_0 = N_1 = 0$.

D3 [Solve for each choice of r_l.] For $b = 0$, tentatively set $a_{ll} = b$ and solve the nonhomogeneous linear equation (9) in the free variables $x_{n-k+1},\ldots,x_n$ and $r_{l+1},\ldots,r_n$ using algorithm **G**. Let S_b denote the size of the solution space. Repeat step D3 for $b = 1$.

D4 [Tally solutions.] Increase N_b by $S_b/2^{n-l+1}$ for $b = 0,1$.

D5 [Next prefix.] Update $\mathbf{x}'$ and possibly k so that $\mathbf{x}'$ is the next element of $\binom{[n-k]}{k}$ or $\binom{[n-k-1]}{k}_1$ for $k = 0,\ldots,\frac{1}{2}n$, and return to D3. (When all these elements have been generated, proceed to D6.)

D6 If $N_0 > N_1$ then fix $a_{ll} = 1$, otherwise $a_{ll} = 0$. Increase l. If $l \le n$ return to D2. Otherwise output $a_{11},\ldots,a_{nn}$.

V. Proof of Theorem 3

Let $\mathscr{C} \subseteq 2^E$ denote the set of (directed) cycle covers of the directed graph G. We use Iverson's bracket notation: for a proposition P, we write

$$[P] = \begin{cases} 1, & \text{if } P\,;\\ 0, & \text{otherwise}\,. \end{cases}$$

$G[Y]$ is the graph induced by $Y \subseteq V$. We sometimes write $e \in G[Y]$ to refer to a directed edge e in the induced graph $G[Y]$. We will operate on sums of partitions of vertex sets, and manipulate these sums. For this purpose, we introduce the notation

$$\sum_{Y_1,\ldots,Y_K}^{V}$$

for the sum over all ordered K-partitions $Y_1,\ldots,Y_K$ of V. That is, $Y_1 \cup \cdots \cup Y_K = V$ and $Y_i \cap Y_j = \emptyset$ $(1 \le i < j \le k)$. Parts may be empty.

For a subset $T \subseteq E$ of directed edges define

$$h(T) = \frac{1}{K} \sum_{Y_1,\ldots,Y_K}^{V} \prod_{e\in T} [e \in G[Y_1] \cup \cdots \cup G[Y_K]]\,.$$

We will show that the residue modulo K of the function h serves as an indicator variable for Hamiltonicity on $\mathscr{C}$.

Lemma 7. *For* $C \in \mathscr{C}$,

$$[C \in \mathscr{H}] \equiv_K h(C)\,. \tag{11}$$

Proof: First, we show that for $C \in \mathscr{H}$ we have $h(C) \equiv_K 1$. Consider a Hamiltonian cycle C and a partition $Y_1,\ldots,Y_K$ of V. First, every partition that sets $Y_k = V$ for some $k \in \{1,\ldots,K\}$ (and all other parts empty) has $e \in G[Y_k]$ for all $e \in C$, so its contribution is 1. On the other hand, consider a partition where some Y_k $(1 \le k \le K)$ is neither $\emptyset$ nor V. Since C is Hamiltonian, there is a directed edge $uv \in C$ with $u \in Y_k$ and $v \notin Y_k$. This directed edge belongs to none of the $G[Y_k]$, so the product vanishes and the partition does not contribute to the sum. Thus, the total contribution of C to the sum in $h(C)$ is K, and $h(C) \equiv_K 1$.

Second, we show for each $C \notin \mathscr{H}$ that $h(C)$ is 0 (mod K). In particular, we show

$$\sum_{Y_1,\ldots,Y_K}^{V} \prod_{e\in C} [e \in G[Y_1] \cup \cdots \cup G[Y_K]] \equiv_{K^2} 0\,.$$

Partition the non-Hamiltonian cycle cover C into cycles

$$C = C_1 \cup \cdots \cup C_r\,,$$

such that each C_i is a simple cycle. Note that $r > 1$ because $C \notin \mathscr{H}$.

Construct the corresponding partition of vertices

$$V = V_1 \cup \ldots \cup V_r\,,$$

such that V_i are the vertices visited by the cycle C_i.

Let $Y_1, \ldots, Y_K$ be a vertex partition such that

$$\prod_{e \in C} [e \in G[Y_1] \cup \cdots \cup G[Y_K]] = 1 \,. \tag{12}$$

We first need to observe that the cycle-induced vertex partition $V_1, \ldots, V_r$ refines $Y_1, \ldots, Y_K$. Indeed, assume that $V_i \cap Y_j$ is neither empty nor V_i. Then there is a directed edge $e = uv \in C_i$ with $u \in V_i \cap Y_j$ but $v \notin V_i \cap Y_j$. In particular, the directed edge e belongs to neither $G[Y_j]$ nor to any other of the $G[Y_k]$, contradicting (12). Hence the partition $Y_1, \ldots, Y_K$ consists of the r parts of $V_1, \ldots, V_r$. There are K^r ways to pick these parts, each amounting to one 1 in the summation. Since $r > 1$ this shows that the total contribution of $C \notin \mathscr{H}$ is a multiple of K^2 and after division of K still vanishes modulo K. ∎

In particular, we can count the number of Hamiltonian cycles modulo K as

$$|\mathscr{H}| \equiv_K \sum_{C \in \mathscr{C}} h(C) \,. \tag{13}$$

We next rewrite the right hand side by an application of inclusion–exclusion.

For a vertex set $Z \subseteq V$ let $\mathscr{F}(Z)$ be the family of edge subsets in which every vertex has outdegree 1 and all terminals are in Z. In other words, $\mathscr{F}(Z)$ is the family of total functions $f \colon V \to Z$ where $vf(v)$ is a directed edge in the graph for all $v \in V$.

Lemma 8.

$$\sum_{C \in \mathscr{C}} h(C) \equiv_K \sum_{Z \subseteq V} (-1)^{|V \setminus Z|} \sum_{F \in \mathscr{F}(Z)} h(F) \,.$$

Proof: Consider a cycle cover $C \in \mathscr{C}$. It belongs to $\mathscr{F}(Z)$ exactly when $Z = V$, so its total contribution to the right hand side is $h(C)$.

Consider now an edge subset $F \in \mathscr{F}(V) \setminus \mathscr{C}$ that is not a cycle cover. Let X denote the set of vertices appearing as terminals for the directed edges in F. Then F belongs to $\mathscr{F}(Z)$ for every Z with $X \subseteq Z \subseteq V$. By the principle of inclusion–exclusion, the number of Z with $X \subseteq Z \subseteq V$ is even (using that $X \subsetneq V$ because F is not a cover). In particular there are as many odd Z as even ones. Thus, the contributions of all $F \in \mathscr{F}(V) \setminus \mathscr{C}$ cancel. ∎

Recall that $d_v(X)$ denotes the number of directed edges from v to a vertex in X. For pairwise disjoint subsets $Y_1, \ldots, Y_K$ of V whose union includes all of Z, let $d_v(Z; Y_1, \ldots, Y_K)$ denote the number of directed edges from v to a vertex in Z that stay in the same part as v, formally

$$d_v(Z; Y_1, \ldots, Y_K) = N(Z \cap Y_k) \,, \quad \text{where } v \in Y_k \,.$$

Lemma 9. *For $Z \subseteq V$,*

$$\sum_{F \in \mathscr{F}(Z)} h(F) \equiv_K \frac{1}{K} \prod_{v \in V \setminus Z} d_v(Z) \sum_{Y_1, \ldots, Y_K}^{Z} \prod_{v \in Z} d_v(Z; Y_1, \ldots, Y_K) \,.$$

Proof: Expanding $h(F)$ and rearranging, we have

$$\begin{aligned} \sum_{F \in \mathscr{F}(Z)} h(F) &= \\ &\frac{1}{K} \sum_{F \in \mathscr{F}(Z)} \sum_{Y_1, \ldots, Y_K}^{V} \prod_{e \in F} [e \in G[Y_1] \cup \cdots \cup G[Y_K]] = \\ &\frac{1}{K} \sum_{Y_1, \ldots, Y_K}^{V} \sum_{F \in \mathscr{F}(Z)} \prod_{e \in F} [e \in G[Y_1] \cup \cdots \cup G[Y_K]] = \\ &\frac{1}{K} \sum_{Y_1, \ldots, Y_K}^{V} \prod_{v \in V} d_v(Z; Y_1, \ldots, Y_K) \,, \end{aligned}$$

where the last step is based on counting in two different ways the number of ways that every vertex in V can choose another vertex in Z in the same part of $Y_1, \ldots, Y_K$.

The next step is to establish

$$\sum_{Y_1, \ldots, Y_K}^{V} \prod_{v \in V} d_v(Z; Y_1, \ldots, Y_K) = \Big(\prod_{v \in V \setminus Z} d_v(Z) \Big) \sum_{Y_1, \ldots, Y_K}^{Z} \prod_{v \in Z} d_v(Z; Y_1, \ldots, Y_K) \tag{14}$$

by induction in $|V \setminus Z|$.

For $V = Z$ the claim is vacuous. Otherwise, select a vertex $w \in V \setminus Z$ and split the sum on the left hand side into K sums according to which part includes the vertex w.

$$\sum_{Y_1, \ldots, Y_K}^{V} \prod_{v \in V} d_v(Z; Y_1, \ldots, Y_K) = S_1 + \cdots + S_K \,,$$

where

$$S_k = \sum_{Y_1, \ldots, Y_K}^{V \setminus \{w\}} \prod_{v \in V} d_v(Z; Y_1, \ldots, Y_k \cup \{w\}, \ldots, Y_K) \,.$$

In the product, the factor contributed by the term corresponding to $v = w$ is

$$\begin{aligned} d_w(Z; Y_1, \ldots, Y_k \cup \{w\}, \ldots, Y_K) &= \\ d_w(Z \cap (Y_k \cup \{w\})) &= d_w(Z \cap Y_k) \,, \end{aligned}$$

because w does not belong to Z. Thus,

$$S_k = \sum_{Y_1, \ldots, Y_K}^{V \setminus \{w\}} d_w(Z \cap Y_k) \prod_{v \in V \setminus \{w\}} d_v(Z; Y_1, \ldots, Y_K) \,.$$

Summing all the S_k we arrive at

$$S_1 + \cdots + S_K = \sum_{Y_1,\ldots,Y_K}^{V\setminus\{w\}} \Big\{ \Big(\sum_{k=1}^{K} d_w(Z \cap Y_k) \Big) \prod_{v\in V\setminus\{w\}} d_v(Z; Y_1, \ldots, Y_K) \Big\} = \sum_{Y_1,\ldots,Y_K}^{V\setminus\{w\}} d_w(Z) \prod_{v\in V\setminus\{w\}} d_v(Z; Y_1, \ldots, Y_K) = d_w(Z) \sum_{Y_1,\ldots,Y_K}^{V\setminus\{w\}} \prod_{v\in V\setminus\{w\}} d_v(Z; Y_1, \ldots, Y_K) = \Big(\prod_{w\in V\setminus Z} d_w(Z) \Big) \sum_{Y_1,\ldots,Y_K}^{Z} \prod_{v\in Z} d_v(Z; Y_1, \cdots, Y_K),$$

where the last identity follows by induction over w, establishing (14). ∎

We are ready to establish theorem 3, namely,

$$|\mathscr{H}| \equiv_K \frac{1}{K} \sum_{Z,Y_1,\ldots,Y_K}^{V} (-1)^{|V\setminus Z|} \Big(\prod_{z\in Z} d_z(V \setminus Z) \Big) \prod_{k=1}^{K} \prod_{y\in Y_k} d_y(Y_k).$$

Proof of theorem 3: Write $\overline{Z} = V \setminus Z$. Combining (13) and lemmas 8 and 9 we get:

$$|\mathscr{H}| \equiv_K \frac{1}{K} \sum_{Z\subseteq V} \sum_{Y_1,\ldots,Y_K}^{Z} (-1)^{|\overline{Z}|} \prod_{v\in\overline{Z}} d_v(Z) \prod_{v\in Z} d_v(Z; Y_1, \ldots, Y_K) \equiv_K \frac{1}{K} \sum_{Z,Y_1,\ldots,Y_K}^{V} (-1)^{|Z|} \prod_{v\in Z} d_v(\overline{Z}) \prod_{v\in\overline{Z}} d_v(\overline{Z}; Y_1, \ldots, Y_K),$$

where we just changed the original summand Z to its complement $\overline{Z}$.

Finally, every $y \in \overline{Z}$ belongs to exactly one Y_k, and for this value of k we have $d_y(\overline{Z}; Y_1, \ldots, Y_K) = d_y(Y_k)$. Thus,

$$\prod_{y\in\overline{Z}} d_y(\overline{Z}; Y_1, \ldots, Y_K) = \prod_{k=1}^{K} \prod_{y\in Y_k} d_y(Y_k).$$

∎

Acknowledgements

This work is supported by the Swedish Research Council, grant VR 2012-4730: Exact Exponential-time Algorithms.

References

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

[2] E. Bax and J. Franklin. A permanent algorithm with $\exp[\Omega(n^{1/3}/2\ln n)]$ expected speedup for 0-1 matrices. Algorithmica 32, pp. 157–172, 2002.

[3] A. Björklund. Determinant sums for undirected Hamiltonicity. In proceedings of the 51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23–26, 2010, Las Vegas, Nevada, USA. IEEE Computer Society 2010, 51st FOCS, pages 173–182.

[4] A. Björklund. Below all subsets for permutational counting problems. arXiv:1211.0391, 2012.

[5] 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 27th Conference on Computational Complexity, CCC 2012, Porto, Portugal, June 26-29, 2012, pages 74–84.

[6] M. Cygan, S. Kratsch, and J. Nederlof. Fast Hamiltonicity checking via bases of perfect matchings. arXiv:1211.1506, 2012. Accepted to 45th ACM Symposium on the Theory of Computing, STOC 2013, June, 1–4, 2013, Palo Alto, USA.

[7] 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, CA, USA, October 22-25, 2011. IEEE Computer Society 2011, 52nd FOCS, pages 150–159.

[8] D. E. Knuth, *The Art of Computer Programming*, 3rd edition, Addison–Wesley, 1997.

[9] M. Held and R. M. Karp. A dynamic programming approach to sequencing problems, J. Soc. Indust. Appl. Math. 10, pp. 196–210, 1962.

[10] A. G. Thomason. Hamiltonian cycles and uniquely edge colourable graphs, Advances in Graph Theory (Cambridge Combinatorial Conf., Trinity College, Cambridge, 1977), Annals of Discrete Mathematics, 3, pp. 259–268, 1978.

[11] L. G. Valiant. Completeness for parity problems. In proceedings of Computing and Combinatorics, 11th Annual International Conference, COCOON 2005, Kunming, China, August 16-29, 2005. Lecture Notes in Computer Science 3595, Springer 2005, pages 1–8.

[12] G. J. Woeginger. Exact algorithms for NP-hard problems: A survey. In Combinatorial Optimization – Eureka! You shrink!, M. Juenger, G. Reinelt and G. Rinaldi (eds.). Lecture Notes in Computer Science 2570, Springer 2003, pages 185–207.