# Upper Bounds of Graph Hamiltonian Cycles with Arithmetic Computation

IFTIKHAR AHAMATH BURHANUDDIN, Adobe Research, India

The computational complexity to determine whether a graph has a Hamiltonian cycle, and the corresponding problem of counting the number of Hamiltonian cycles are conjectured to be intractable problems. We present a polynomial-time algorithm to calculate an upper bound of the number of Hamiltonian cycles of a graph. We define an algorithm that under strong conditions calculates the exact number of Hamiltonian cycles, or an additive polynomial approximation. We utilize number theory for modular arithmetic to represent combinatorial information, and additionally for node weight encodings by using generalized Sidon sets to refine the combinatorics. More generally, the larger node weights leads to a sub-exponential time algorithm, and node weights with exponential number of binary digits leads to an exact algorithm. The core algorithm uses iterative matrix multiplication and modular arithmetic to count closed paths including Hamiltonian cycles. A redefinition of the algorithm obtains upper bounds for the number of Hamiltonian paths, matrix permanents, and also obtains partial information about the cost of Traveling Salesman tours. The ideas convey some computational questions of interest about cryptography, learning, and intractability.

CCS Concepts: • **Theory of computation → Graph algorithms analysis**; **Problems, reductions and completeness**; *Random walks and Markov chains*; *Cryptographic primitives*; *Pseudorandomness and derandomization*; • **Mathematics of computing → Combinatorial optimization**; **Combinatoric problems**; **Paths and connectivity problems**.

Additional Key Words and Phrases: Algorithms, Hamiltonian Cycles, Combinatorial Optimization, Counting Complexity, Computational Complexity, Number Theory, Cryptography

**ACM Reference Format:**
Iftikhar Ahamath Burhanuddin. 2026. Upper Bounds of Graph Hamiltonian Cycles with Arithmetic Computation. In *Proceedings of (STOC 2027).* ACM, New York, NY, USA, 19 pages. https://doi.org/XXXXXXX.XXXXXXX

## 1 Introduction

The problem of determining whether a given graph has a Hamiltonian cycle, that is whether there is a simple cycle that visits each node exactly once, is known as the Hamiltonian cycle problem, and is conjectured to be a computationally intractable problem for large graphs. Several such combinatorial problems, and number theoretic problems belong to the complexity class $\mathbb{NP}$ that can be solved using polynomial time non-deterministic computation. This type of computation is equivalent to the polynomial-time verifiability of the corresponding problem instance certificates.

This paper is about a polynomial-time algorithm for the related problem of calculating an upper bound for the number of Hamiltonian cycles of a graph. We utilize iterative multiplication of the adjacency matrix of the graph, and modular arithmetic to derive combinatorial information. This random walk represents information that is an aggregate of non-deterministic combinatorial computation. When algorithms obtains a trivial upper bound, we know that the graph is not Hamiltonian, and when the upper bound is non-trivial we argue that plausible intractability is because of pseudorandonmess.

Author's Contact Information: Iftikhar Ahamath Burhanuddin, burhanud@adobe.com, Adobe Research, India.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2026 Copyright held by the owner/author(s). Publication rights licensed to ACM.
Manuscript submitted to ACM

The following are the contributions of this paper.

- A polynomial-time algorithm to calculate an upper bound of the number of Hamiltonian cycles via modular arithmetic, refer to Theorem 3.2, and using strong conditions an additive polynomial approximation, refer to Theorem 3.13. The stronger conditions of the theorems seem to be required for dense graphs because of the combinatorics of paths. The algorithm introduces a computationally simple definition of generalized Sidon sets for node weights for correct and efficient computation, refer to Definition 3.8.
- The algorithm with node weights that have sub-exponential number of binary digits, leads to a sub-exponential time algorithm with approximation guarantees, refer to Lemma 3.15, thus we are not able to disprove the Exponential Time Hypothesis. Using exponential weights, instead of aforementioned sub-exponential weights, leads to an exact algorithm, refer to Theorem 3.16.
- Though previous work has utilized powers of the multiplication of the adjacency matrix to count paths algebraically, our formulation using arithmetic computation and the use of Sidon sets as node weights to minimize the upper bound is novel. Preliminary empirical computation obtains exactly, or approximately the number of Hamiltonian cycles for random sparse graphs with up to 40 nodes, refer to Section 3.5.
- A study of combinatorial optimization and number theory computation leads to questions of interest for algorithm design, computational learning, and computational complexity, refer to Theorem 4.1, and Section 4. The main algorithm and results can be redefined to obtain upper bounds for the number of Hamiltonian paths, matrix permanents, and partial information about the cost of Traveling Salesman tours that is a weighted version of the Hamiltonian cycle problem, refer to Section A.

The paper is organized as follows: Section 2 summarizes references that have influenced our work. Section 3 presents the main results about a polynomial time algorithm to upper bound the number of Hamiltonian cycles, the correctness and exactness of the algorithm under strong specified conditions, and time complexity analysis. Section 4 suggests plausible explanations for intractability of counting complexity of Hamiltonian cycles with modular arithmetic computation by referencing the concept of pseudorandomness inherent to cryptography. Section B discusses alternative node weight settings for the algorithm that convey the difficulty of exactly counting the number of Hamiltonian cycles.

## 2 Related Work

We refer to concepts of the design and analysis of algorithms [CLRS09, AS00], number theory and cryptography [Kob94, Sil09], and computational complexity [AB09, Wig18].

**Arithmetic Computation** We have defined an algorithm to upper bound the number of Hamiltonian cycles noting that this number is invariant to the permutation of certain node weights. The ideas reference $p$-adic arithmetic geometry specialized to matrix multiplication modular arithmetic [BH05, Sil09]. Sidon sequences [VdR10, Cil17] are equivalent to Golomb rulers that are defined so that differences of elements are distinct [BG77, MP09]. Advanced results from combinatorics, and number theory about representation of positive integers using sums of squares, or sums of powers of numbers could improve our results.

**Computing the Permanent, and Random walks** Algorithm 1 calculates an upper bound of the number of Hamiltonian cycles utilizing paths of appropriate length, and the set of permutation of graph nodes are a subset of these paths, thus the computation of the matrix permanent is a related concept [Val79]. An algorithm for approximating the permanent of a matrix using the conductance of Markov Chains on the space of Hamiltonian cycles [AV77, JS89]. Several algorithms that count simple paths and cycles have been defined including utilizing powers of the adjacency

matrix and combinatorial sieves [GKW16], immanants, and by enumeration. An algebraic computation that uses matrix multiplication to represent each edge of the graph with a symbolic variable in the adjacency matrix, with the algebra being commutative, and the square of the variables being zero to avoid repeated nodes has commonality with this paper [SS11]. The algorithm to calculate the matrix permanent using the discrete Fourier Transform in exponential time [Fon26] has technical commonality with our algorithm. An alternative computational version of Algorithm 1 is to utilize one symbolic variable instead of arithmetic to obtain an upper bound with modular algebra instead of modular arithmetic.

The traces of powers of a matrix can be written in terms of the sum of the powers of the eigenvalues of the matrix [HM71]. So computation of eigenvalues is an alternative route to calculate characteristics of graph paths. The theory of stochastic processes [AS00, LP25] could be used to improve combinatorial algorithms.

**Cryptography** We will convey in Section 4 that the definitions of pseudo-randomness using unpredictability, and indistinguishability are plausible explanations for intractability of combinatorial optimization [AB09, KV94] when modular arithmetic is utilized. The concept of fixed parameter complexity for computation, and cryptography is relevant [FK92] because of the time complexity formula.

**Quantum computation** The formulation of the algorithm uses an all paths computation using linear operators, and matrix multiplication, and is inspired by non-deterministic computation, and the postulates of quantum computation [Sho94, NC26].

**Algorithms and Complexity of Counting Problems** The computational complexity of counting and the $\#\mathbb{P}$ complexity class is discussed in the context of probability, physics, graphs and networks [AB09]. The theory of counting complexity includes using the Möbius inversion formula, a generalization of the inclusion-exclusion principle to graphs to count graph homomorphisms, and subgraphs to determine the number of solutions exactly [CDM17], using prime finite field computation [BK22, LGCF21, PRS+23], and approximation algorithms [Rot26]. Our discussion of modular arithmetic is to upper bound counting to obtain the count exactly, or approximately, and not necessarily to a prime modulus.

There are connections to approximately counting the cardinality of the set of bases of a matroid using convex optimization [AOGV18]. The basis generating polynomial of a matroid is similar to the sum-product formula of Eq. 2 with each closed path representing a basis. We speculate that Algorithm 1 seems to be an arithmetic version of the aforementioned algorithm with our upper bound being calculated using integers, instead of convex optimization using probabilities of the generating polynomials of a matroid. If the ideas about log-concave polynomials or probability sequences [KKG21] could be redefined to either improve the upper bound, or prove a lower bound, then improved approximation guarantees with arithmetic computation can be obtained.

**Exponential Time Hypothesis** Algorithm 1 describes using polynomial Sidon weights to obtain an upper bound, and using exponential weights for exactness, refer to Section 3.4. The recent result about the refutation of the Exponential Time Hypothesis [Ope26] leads to fine-grained complexity, and the possibility of efficient algorithms for intractable problems [WW20]. The $L[n, \frac{1}{2}]$ algorithm for 3-SAT, the satisfiability of a formula of clauses with 3 variable-literals, suggests the question of whether a sub-exponential algorithm is sufficient for exact, or approximation of the number of Hamiltonian cycles.

## 3 Main Results

Let $G = (V, E)$ be a graph with the cardinality of the set of vertices $V$ equal to $\#V = n$, $\#E = m$ the cardinality of the set of edges, and each graph vertex has no self-loop. Let the graph be represented as a matrix $M_{n\times n}(\mathbb{Z}_{\geq 0})$ of non-negative

integers representing edge weights $M_{i,j} \leq O(n^{O(1)})$ with the degree of polynomial equal to a constant so that edge weights require $O(\log n)$ binary digit representation. Logarithms of positive integers will be to base 2 to present the complexity of arithmetic computation.

A Hamiltonian cycle of a directed or undirected graph is a simple cycle that contains each vertex. The problem of determining whether a graph has a Hamiltonian cycle is $\mathbb{NP}$-complete [CLRS09]. The $t^{\text{th}}$ power of the matrix $M_{i,j}^{(t)}$ calculates the sum of the path weights of length $t \geq 1$ from vertex $i$ to vertex $j$ with the path weight being the product of the edge weights of the path. This can be recursively defined as follows.

$$M_{i,j}^{(t)} = \sum_{k=1}^{n} M_{i,k}^{(t-1)} \cdot M_{k,j} \tag{1}$$

For $t = n$, this leads to a summation across paths $\gamma = (i, i_1, \ldots, i_{n-1}, j)$ from $i$ to $j$ with possibly repeated intermediate vertices $i_1, \ldots, i_{n-1}$ specific to each path.

$$M_{i,j}^{(n)} = \sum_{\gamma=(i,i_1,\ldots,i_{n-1},j)} M_{i,i_1} \cdot \ldots \cdot M_{i_{n-1},j} \tag{2}$$

This implies that the number of Hamiltonian cycles $\alpha$ of a graph with $n$ nodes is less than or equal to $(n-1)!$. A problem to solve is given an unweighted graph whether there are the node weights, and edge weights so that calculating the number of Hamiltonian cycles reduces to matrix multiplication. We make algorithm design choices for efficient computation, and discuss alternative nodes weights in Section B.

### 3.1 Counting Hamiltonian Cycles

Let a graph $G$ have node weights that are positive integers $1 < p_1 \leq p_2 \leq \ldots \leq p_n$ for $i \in V$, and let the edge weights be $M_{i,j} = \mathbb{1}_{i,j} \cdot p_i \cdot p_j$ with $\mathbb{1}_{i,j} = 1$ if $(i,j) \in E$, and equal to 0 otherwise. Consider the weight of node $i$ to be $p_k = p^{w_k}$ with $p = n^{n-1} + 1$. We will use to phrase node weight to refer to either $p^{w_i}$, or $w_i$ as per the context.

Let $\sigma(w, n) = \sum_{k=1}^{n} w_k$ denote the sum of the node weights. The details of the proof will clarify the path weight of a Hamiltonian cycle is $p^{2 \cdot \sigma(w,n)}$, and a multiple of this number that is calculated by Algorithm 1 is an upper bound of the number of Hamiltonian cycles. For instance, if $w_k = k$ for $k = 1, \ldots, n$, then $\sigma(w, n) = \frac{n(n+1)}{2}$, and the weight of the Hamiltonian cycle is $p^{2 \cdot \sigma(w,n)} = p^{n^2+n}$.

*Definition 3.1.* Let $\gamma = (i, i_1, \ldots, i_{n-1}, j)$ be a sequence of graph nodes of length $n$, then let $\deg(\gamma, k)$ denote the number of non-negative occurrences of node $k$ in the sequence of nodes $i, i_1, \ldots, i_{n-1}, j$ of the path $\gamma$. And let $w(\gamma)$ denote the weight of the path $\gamma$ with nodes weights $w_k$, that is $w(\gamma) = p^{\sum_{k=1}^{n} \deg(\gamma,k) \cdot w_k}$.

We assume that the nodes do not have loops, thus $0 \leq \deg(\gamma, j) \leq \frac{n}{2}$ for each node $j$, and because we are considering closed paths of length $n$, $\sum_{j=1}^{n} \deg(\gamma, j) = n$. And the path weights are a multiple of 2 because of the definition of edge weights.

To define a polynomial-time algorithm, we consider Generalized Sidon sets [VdR10], that is sets of integers such that each sum of $t$ elements has at most $g$ solutions up to permutation. These sets are denoted by $B_t[g]$. We consider Sidon sets for node weights $w_k$ in order to minimize the number of closed paths of weight $p^{2 \cdot \sigma(w,n)}$. The definition of Sidon sets is discussed in Section 3.2.

We summarize that the algorithm we define calculates $M_{i,i}^{(n)}$ that is a sum of the path weight as power series, and a specific coefficient of the power series gives an upper bound for $\alpha$ the number of Hamiltonian cycles.

Theorem 3.2. *Algorithm 1 is a polynomial time algorithm that calculates $\beta$ an upper bound for $\alpha$ the number of Hamiltonian cycles of a graph.*

*If there exist distinct nodes $i, j$ such that*

(1) *for each closed path $\gamma$ with $log_p(w(\gamma)) = 2 \cdot \sigma(w, n)$, $deg(\gamma, i) \neq deg(\gamma, j)$*
(2) *for each closed path $\gamma$ with $log_p(w(\gamma)) \neq 2 \cdot \sigma(w, n)$, $deg(\gamma, i) = deg(\gamma, j) \geq 0$*

*then swapping the node weights $p_i$ and $p_j$, the correct number of Hamiltonian cycles is calculated with $\beta = \alpha$.*

Proof of Theorem 3.2. To count the number of Hamiltonian cycles, we consider simple closed paths of length $n$ from an arbitrary node $i$ to itself. Thus Eq. 2 with $t = n$, and $j = i$, $M_{i,i}^{(n)}$ is the sum of weights of closed paths of length $n$ from node $i$ to $i$, and this includes the sum of the weights of Hamiltonian cycles.

$$M_{i,i}^{(n)} \quad = \quad \sum_{\gamma=(i,i_1,\ldots,i_{n-1},i)} \mathbb{1}_{i,i_1} \cdot p_i \cdot p_{i_1} \cdot \ldots \cdot \mathbb{1}_{i_{n-1},j} \cdot p_{i_{n-1}} \cdot p_i$$

The weight of a Hamiltonian cycle is $q^2$ with $q = \prod_{j=1}^{n} p_j$. If the node weights are set such that to $p_j = p^{w_j}$, then the weight of each closed path $\gamma$ of length $n$ is equal to an integer power of $p$ and is equal to

$$\prod_{j=1}^{n} p_j^{2 \cdot \deg(\gamma, j)} = p^{2 \cdot \sum_{j=1}^{n} \deg(\gamma, j) w_j}$$

and thus $q^2 = p^{2 \cdot \sigma(w,n)}$ for Hamiltonian cycles and possibly closed paths of weight $q^2$ that are not Hamiltonian cycles.

If $\gamma$ is a closed path that is not a Hamiltonian cycle, then there is a path node that is not visited, and a path node is visited more than once. Since there are no nodes with loops, the path weights are greater than or equal to $p^{n \cdot (w_1 + w_2)}$, and less than equal to $p^{n \cdot (w_{n-1} + w_n)}$ with $w_1 \leq w_2 \leq \ldots \leq w_{n-1} \leq w_n$.

Let $W_{\leq q^2}$ and $W_{>q^2}$ denote the set of closed paths of length $n$ from $i$ to itself with path weight less than equal to $q^2$, and greater than $q^2$ respectively. Then

$$M_{ii}^{(n)} \quad = \quad \alpha \cdot q^2 + \sum_{\gamma \in W_{\leq q^2}} w(\gamma) + \sum_{\gamma \in W_{>q^2}} w(\gamma)$$

Then the previous equation can be rewritten as a power series of $p$ with $\alpha_l$ denoting the $l^{\text{th}}$ coefficient of the power series.

$$\alpha \cdot q^2 \quad \leq \quad M_{ii}^{(n)} = \sum_{l=n \cdot (w_1 + w_2)}^{n \cdot (w_{n-1} + w_n)} \alpha_l \cdot p^l.$$

Therefore, we can calculate an upper bound of the number of Hamiltonian cycles using the formula

$$\alpha \leq \alpha_{2 \cdot \sigma(w,n)} \quad = \quad \min_{i=1}^{n} \frac{M_{i,i}^{(n)} \bmod p^{2 \cdot \sigma(w,n)+1} - M_{i,i}^{(n)} \bmod p^{2 \cdot \sigma(w,n)}}{p^{2 \cdot \sigma(w,n)}} \tag{3}$$

Let $\gamma$ be a closed path of length $n$ with nodes $i$ and $j$ being distinct nodes of $\gamma$. Then the logarithm to base $p$ of the weight of $\gamma$ is

$$log_p(w(\gamma)) = \deg(\gamma, i) w_i + \deg(\gamma, j) w_j + \sum_{k \neq i,j} \deg(\gamma, k) w_k.$$

If the weights of $i$ and $j$ are swapped, then this leads to the path weight $w'(\gamma)$ with $w'_i = w_j$, $w'_j = w_i$, and $w'_k = w_k$ for each node $k \neq i, j$. Because each path weight is a power of $p$, if we consider the ratio of the weight of $\gamma$ before and after the swap, the cancellation of the weight of nodes $k \neq i, j$ indicates that the paths weights $w(\gamma)$ and $w'(\gamma)$ are the

same or different depends on whether the difference of the logarithm of the path weights is equal to 0 or is a non-zero integer

$$\log_p(w(\gamma)) - \log_p(w'(\gamma)) = 2 \cdot (\deg(\gamma, i) - \deg(\gamma, j)) \cdot (w_i - w_j). \tag{4}$$

Thus because node weights are $w_i, w_j$ are distinct, this discussion illustrates that if the nodes $i$ and $j$ satisfy the theorem conditions then after the swap of weights $w_i, w_j$, we obtain $\beta = \alpha$, because of the following logic

- each closed path of weight $w(\gamma) = p^{2\cdot\sigma(w,n)}$ becomes a closed path of a different weight, since $(\deg(\gamma, i) - \deg(\gamma, j)) \neq 0$,
- each closed path of weight $w(\gamma) \neq p^{2\cdot\sigma(w,n)}$ does not become a closed path of weight equal to $p^{2\cdot\sigma(w,n)}$, since $(\deg(\gamma, i) - \deg(\gamma, j)) = 0$, and
- the weight of each closed path that is a Hamiltonian cycle $w(\gamma) = p^{2\cdot\sigma(w,n)}$ is invariant.

If such nodes $i$ and $j$ do not exist, because of the node selection loops the upper bound $\beta$ is iteratively minimized, and possibly $\beta > \alpha$. □

More generally, suppose there is a node $i$, $i \in U$ a subset of $V$, $U \subseteq V$ subset, node weights set $\{p_j \mid w(v_i) = p_i\}$, and a permutation $\pi : V \to V$ such that after swapping node weights $j \in U, p_j = p^{\pi(j)}$, and $j \in V \setminus U$, $p_j = p^j$, such that only Hamiltonian cycles are invariant to the permutation of weights, then $\alpha = \beta_{i,U,1}$. The problem of efficiently determining the aforementioned subset $U$ of graph nodes, and the permutation of node weights $\pi$ may be intractable.

### 3.2 Generalized Sidon sets as node weights

We previously used the letter $t$ to denote the power of the adjacency matrix, that is to consider paths of length $t$. For this section, the notation $t$ will denote the cardinality of a set, or multiset of node weights.

*Definition 3.3.* Let $t \geq 2$ be an integer. A sequence of positive integers $A$ is a $B_t[g]$ set if every integer $m$ has at most $g$ representations as a sum of $t$ elements of $A$ accounting for permutations. Let

$$r_{t,A}(m) = \#\{(a_1, \ldots, a_t) \mid m = a_1 + \ldots + a_t,\ a_1 \leq \ldots \leq a_t,\ a_i \in A\}.$$

That is, $A$ is defined to be a $B_t[g]$ set if $r_{t,A} \leq g$ for every integer $m$. The definition of Sidon sets from the literature [VdR10] that does not account for permutations of $t$ elements.

The $B_2[1]$ Sidon set is defined so that sums of any $t = 2$ elements are distinct. For instance, consider the following node weights.

Lemma 3.4 (Erdös-Turán). *For an odd prime number $p$,*

$$\{w_k = 1 + 2 \cdot p \cdot k + (k^2 \bmod p) \mid 0 \leq k \leq p - 1\}$$

*is a $B_2[1]$ Sidon set.*

We add 1 to the definition of the elements of Sidon sets so that the node weights are positive integer powers of $p$.

Lemma 3.5. *Given positive integer $n$, the set*

$$\{w_k = 1 + 2 \cdot n \cdot k^2 + k \mid 0 \leq k \leq n - 1\}$$

*is a $B_2[1]$ Sidon set.*

The abstract problem to solve is to set distinct node weights $p_j = p^{w_j}$, with $w_j < w_{j+1}$ for $1 \leq j \leq n-1$ so that the matrix multiplication modular arithmetic algorithm discussed exactly calculates the number of Hamiltonian cycles. Specifically, we require distinct node weights $w_k$ such that for each closed path of length $n$, $w(\gamma) = p^{2\cdot\sum_{k=1}^{n}\deg(\gamma,k)\,w_k} = p^{2\cdot\sum_{k=1}^{n} w_k} = p^{2\cdot\sigma(w,n)}$ for all closed paths. Let $x_k$ be a non-negative integer variable $\deg(\gamma, k)$ that denotes the path degree of each node $w_k$.

Let

$$T = \{(x_1, \ldots, x_n) \mid \sum_{k=1}^{n} x_k w_k = \sigma(w, n),\ 0 \leq x_k\}.$$

Theorem 3.6 (Schur). *The number of representations of $\sigma(w, n)$ as a non-negative integer linear combination of the set $\{w_1, \ldots, w_n\}$ that is a set of relatively prime positive integers is*

$$\#T = \frac{\sigma(w,n)^{n-1}}{(n-1)!\cdot\prod_{k=1}^{n} w_k} + O(L^{n-2}).$$

This problem is known as the money changing problem [Wil05]. We require a better asymptotic that the aforementioned theorem to ensure a polynomial time algorithm. Specifically, we are interested in node weights that optimize the cardinality of the set of the number of integer linear combinations with the additional conditions that the sum of the node degrees for each path sum to $n$, $\sum_{k=1}^{n} x_k = n$ and $0 \leq x_k \leq \frac{n}{2}$.

$$\operatorname{argmin}_{(w_1,\ldots,w_n)} \#\{(x_1, \ldots, x_n) \mid \sum_{k=1}^{n} x_k w_k = \sum_{k=1}^{n} w_k, \sum_{k=1}^{n} x_k = n, 0 \leq x_k \leq \frac{n}{2}\}.$$

Let us consider nodes $v_k$ with node weights $w_{v_k} = k$ for each index $k$. Let a closed path be $\gamma = (v_1, v_2, \ldots, v_n, v_1)$ of length $n$. Consider another closed path $\gamma'$ so that nodes $v_i$ and $v_j$ are not visited with $i < j$, and instead $v_{i-1}$ and $v_{j+1}$ are revisited appropriately so that $\gamma'$ is also a path of length $n$. Then because of the specific set of node weights the distinct nodes as part of a path are distinguishable by node weight, two sets of two nodes are interchangeable since the paths weights are the same. This scenario is avoided utilizing $B_2[1]$ Sidon set, because sums of any two set elements is distinct, so the number of closed paths of same weight is minimized. We utilize a generalized Sidon set to further minimize the upper bound.

We consider Sidon sets as node weights because of the guarantee of fewer paths with the same weight of Hamiltonian cycles. If a closed path $\gamma$ has $t$ unvisited nodes $u_1, \ldots, u_t$ with $\deg(\gamma, u_i) = 0$ then to be a path of length $n$ the path has to revisit $t$ nodes. We prove using the next lemma that a set of $t$ nodes can only be replaced by another set of $t$ nodes for at most $g = \rho(n, t)$ instances using Definition 3.8 of generalized Sidon sets.

Lemma 3.7. *Let $\gamma$ be a Hamiltonian cycle, and $\gamma'$ be a closed path of length $n$ that does not visit $t'$ nodes namely $W = \{v_1, \ldots, v_{t'}\}$, and $U = \{u_1, \ldots, u_{t'}\}$, is a multiset of nodes that is revisited. Suppose that the $n - t'$ many nodes $V \setminus W$ are common across $\gamma$ and $\gamma'$. Then $w(\gamma) = w(\gamma')$ if and only if $w(W) = w(U)$.*

Proof of Lemma 3.7. By the conditions specified by the lemma $\log_p(w(\gamma)) = \log_p(w(\gamma'))$ is equivalent to

$$\sum_{v\in\{v_1,\ldots,v_t\}} \deg(\gamma', v)\cdot w_v \quad = \quad \sum_{u\in\{u_1,\ldots,u_t\}} \deg(\gamma', v)\cdot w_u. \tag{5}$$

We note that for the previous equation $\deg(\gamma', v) = 1$, and $\deg(\gamma', u) \geq 1$, and the degree of the revisited nodes accounts for the multiplicity appropriately. □

*Definition 3.8.* Let $t \geq 3$ be an integer. Let $b_i = i = \sum_{k=0}^{\log_2(n)+1} b_{i,k} \cdot 2^k$ denote the binary representation of integers with $b_{i,k} = \{0, 1\}$, $i = 1, \ldots, n$. Let $\{w_i\}_{1 \leq i \leq n}$ be defined by

$$w_i = \sum_{k=0}^{\log_2(n)+1} b_{i,k} \cdot (t+1)^k.$$

Lemma 3.9. *The set $\{w_i \mid 1 \leq i \leq n\}$ according to Definition 3.8 is a $B_t[g]$ set with $g = ((\frac{3}{2} \cdot t)^{t-1} \cdot e^{-(\frac{t-1}{3})})^{\log_2(n)+2}$ accounting for permutations. For $t$ constant, $g$ is a polynomial of $n$.*

The lemma can be similarly defined for $t$ denoting positive odd integers. An alternative to the aforementioned node weights is to utilize an algorithm [Cil17] with better guarantees.

The aforementioned definition of $B_t[g]$ accounts for the permutation of less than or equal to $t$ nodes for the purpose of establishing the approximation guarantee $\rho(n, t)$ that will be defined subsequently.

Proof of Lemma 3.9. For closed path $\gamma$ of length $n$, suppose there are $t$ unvisited nodes $W = \{v_1, \ldots, v_t\}$, and instead a multiset of $t$ nodes $U = \{u_1, \ldots, u_t\}$ are revisited with possible repetition to ensure that $\gamma$ is a closed path with length $n$.

We will consider the number of instances of sets $U$ such that $w(W) = w(U)$ that could lead to the possibility of same weight closed paths of length $n$. And by our choice of node weights the number of instances of closed paths of length $n$ and weight $p^{\sigma(w,n)}$ are partially minimized. To simplify the notation, let $w_i$ denote the node weight of $u_i$, and $b_{i,k}$ the corresponding $(t+1)$-adic binary coefficients, that is

$$w_i = \sum_{k=0}^{\log_2(n)+1} b_{i,k}(t+1)^k,$$

and let $z$ denote $\sum_{i=1}^{t} w_i$, and $z_k$ denote the corresponding $(t+1)$-adic coefficients of the sum $z$, thus we obtain

$$z = \sum_{i=1}^{t} w_i = \sum_{i=1}^{t} \sum_{k=0}^{\log_2(n)+1} b_{i,k}(t+1)^k = \sum_{k=0}^{\log_2(n)+1} z_k(t+1)^k.$$

As discussed in Definition 3.8, the node weights are numbers in base $(t+1)$ with binary coefficients so that each coefficient in base $(t+1)$ of the sum of $t$ node weights results is $\leq t$, that is

$$0 \leq z_k = \sum_{i=1}^{t} b_{i,k} \leq t.$$

If $z = w(W) = w(U) = z'$, then $z_k = z'_k$ for $k = 0, \ldots, \log_2(n) + 1$. Let $s$ denote $z_k$. Then the number of possibilities for the binary multiset $\{b_{1,k}, \ldots, b_{t,k}\}$ that sum to $0 \leq s \leq t$, accounting for permutations is

$$\begin{aligned} \binom{s+t-1}{t-1} \cdot (t-1)! &= \frac{(s+t-1)!}{(s)!} \\ &= (s+t-1) \cdot (s+t-2) \cdot \ldots \cdot (s+1) \\ &= (s+t)^{t-1} \cdot \prod_{j=1}^{t-1} (1 - \frac{j}{s+t}) \\ &\leq (s+t)^{t-1} \cdot e^{-\frac{(t-1)\cdot t}{2\cdot(s+t)}}, \end{aligned} \tag{6}$$

using the approximation $1 + x \leq e^x$.

The maximum value is obtained when $s = \frac{t}{2}$. Thus, for constant $t$, for large $n$ that across $0 \leq k \leq \log_2(n) + 1$, there is independence across $(t+1)$-adic coefficients, and there are at most

$$((\frac{3}{2} \cdot t)^{t-1} \cdot e^{-(\frac{t-1}{3})})^{\log_2(n)+2}$$

possible instances of the multiset $U$ that can replace the set $W$ of cardinality $t$. □

We note that $w_i \leq w_n = O((t+1)^{\log_2(n)+1}) = O((t+1) \cdot n^{\log_2(t+1)})$, and $\sigma(w,n) = O((t+1) \cdot n^{1+\log_2(t+1)})$.

Let us define the sets of closed paths based on the number of unvisited nodes in each path.

*Definition 3.10.* Let the number of unvisited nodes of a closed path $\gamma$ be denoted by $\text{undeg}(\gamma)$.

*Definition 3.11.* Let $1 \leq t' \leq n-2$,

$$\mathcal{B}_{t'} = \{\gamma \mid w(\gamma) = p^{2\cdot\sigma(w,n)}, \text{undeg}(\gamma) = t'\}$$

We know that $\text{undeg}(\gamma) = 0$ for Hamiltonian cycles, and more generally $0 \leq \text{undeg}(\gamma) \leq n-2$. There are no closed paths with $\text{undeg}(\gamma) = n-1$, or $n$, because we consider graphs without self-loops.

A combinatorial argument implies

$$\#\mathcal{B}_{t'} \leq (n-1-t')^{n-1} \cdot \binom{n-1}{t'}$$

because starting from a node without loops and $t'$ unvisited nodes, there are at most $n-1-t'$ possibilities of edges for the $n-1$ intermediates nodes for closed paths of length $n$. $\#\mathcal{B}_0 = \alpha$, and $\#\mathcal{B}_1 = 0$ because node weights are distinct, and $\#\mathcal{B}_2 = 0$ using $B_2[1]$ Sidon weights. If a closed path from node $i$ does not visit $n-2$ nodes, then the path is a traversal of $(i,k)$ and $(k,i)$ of length $\frac{n}{2}$ for each node $k$, thus $\mathcal{B}_{n-2} \leq n-1$.

Let us recall the definitions of set $W$ and $U$ from Lemma 3.7. Suppose we consider a closed path of length $n$ of weight $p^{2\cdot\sigma(w,n)}$ that does not visit a set of $t'$ many nodes $W$. If we consider the closed paths from node $i$, then there are $\binom{n-1}{t'}$ many such sets. For each such set $W$, the number of multisets $U \subset V \setminus W$ of cardinality $t'$ that can be possibly swapped with $W$ is $\binom{n-t'+t'-1}{t'}$. Thus there are at most many instances of interchangeability

$$\sum_{t'=2}^{n-1} \binom{n-1}{t'}^2 \tag{7}$$

without assuming the constraint of $w(W) = w(U)$, and not considering the permutations of the other $n-t'$ nodes of the closed path of length $n$. If for each of these instances the set weights ensure non-interchangeability, then $\beta = \alpha$ since $\mathcal{B}_{t'} = 0$ for $t' > 0$. This could be because of graph structure, sparsity, and the node weights.

For each closed path, the weights of the unvisited nodes $W$ reindexed to be in increasing order, $w_{v_1} < \ldots < w_{v_{t'}}$ partition the sequence of node weights in to $t'+1$ intervals. The non-decreasing sorted list of weights of the revisited nodes $w_{u_1} \leq \ldots \leq w_{u_{t'}}$ could result in a closed path of length $n$ and weight equal to $p^{2\cdot\sigma(w,n)}$. Let us consider the assumption of Algorithm 1 of an ordering of nodes with respect to non-decreasing node degrees, and for simplification let the nodes be categorized as high degree nodes and low degree nodes. Then for each closed path, if the unvisited nodes $W$ are low degree nodes, and the revisited nodes are high degree nodes, then this closed path will not be of weight equal to $p^{2\cdot\sigma(w,n)}$, and not contribute to $\mathcal{B}_{t'}$.

Lemma 3.12. *The calculation of $\beta$ the upper bound of the number of Hamiltonian cycles by Algorithm 1 can be rewritten as a sum of closed paths of length n of weight $p^{2\cdot\sigma(w,n)}$ according to the number of unvisited nodes of the closed path as*

*follows.*

$$\beta \quad = \quad \alpha + \sum_{t'=2}^{n-2} \#\mathcal{B}_{t'}$$

To obtain an additive polynomial approximation, we require a strong condition on the cardinality of the sets $\mathcal{B}_{t'}$ that is probably true for sparse graphs, and when generalized Sidon node weights results in few closed paths of weight $p^{2\cdot\sigma(w,n)}$ with greater than $t$ number of unvisited nodes.

Theorem 3.13. *Given a graph with $n$ nodes, and an integer $t \geq 2$, and a polynomial $\rho(n,t)$ defined by Lemma 3.9. Algorithm 1 is a polynomial-time approximation algorithm for $\alpha$ the number of Hamiltonian cycles of a graph with generalized Sidon node weights defined by Definition 3.8 with the strong condition that if there are at most $\rho(n,t)$ many closed paths $\gamma$ of length $n$ with $w(\gamma) = p^{2\cdot\sigma(w,n)}$, and $undeg(\gamma) > t$ denoted by*

$$\sum_{t'=t+1}^{n-2} \#\mathcal{B}_{t'} \leq (n-1)\cdot\rho(n,t)$$

*then*

$$\alpha \;\leq\; \beta \;\leq\; \alpha + n\cdot\rho(n.t)$$

*Without the aforementioned assumptions, the cardinality of $\mathcal{B}_{t'}$ is*

$$\#\mathcal{B}_{t'} \leq (n-1-t')^{n-1}\cdot\binom{n-1}{t'}.$$

Proof of Theorem 3.13. Rewriting $\beta$ as

$$\beta \quad = \quad \alpha + \sum_{t'=2}^{t} \#\mathcal{B}_{t'} + \sum_{t'=t+1}^{n-2} \#\mathcal{B}_{t'},$$

Lemma 3.9, and $\#\mathcal{B}_{t'}$ by Eq. 7 imply the theorem. □

A multiplicative approximation guarantee that is $\alpha \leq \beta \leq \rho(n)\cdot\alpha$ with $1 \leq \rho(n)$ seems difficult to prove, because Section A.1 describes a non-Hamiltonian graph with $(n-2)!$ closed paths of weight $p^{2\cdot\sigma(w,n)}$. Though if the graph is Hamiltonian, each Hamiltonian cycle utilizing up to $\binom{n}{t'}$ subsets of $t'$ many unvisited nodes could lead to closed paths of path weight $p^{2\cdot\sigma(w,n)}$.

A question of interest is for what proportion of graphs does Algorithm 1 calculate the correct number of Hamiltonian cycles with Sidon node weights.

### 3.3 Algorithm Time Complexity

Given the adjacency matrix of a graph with $n$ nodes, calculating the square the adjacency matrix requires at most $n^3$ integer multiplications and $n^2\cdot(n-1)$ integer additions. Calculating a positive integer power of a matrix can be further optimized by the exponentiation-by-squaring algorithm, using the binary digits of the integer power exponent.

We note that algorithms for multiplication, division and modular arithmetic of two $k$ bit numbers require about $O(k\cdot\log k\cdot\log\log k)$ bit operations, and there are more efficient algorithms. So for this discussion, we do not considering sub-linear terms, and assume that modular multiplication, and division are almost linear.

Theorem 3.14. *The time complexity of Algorithm 1 is $O((t+1)^2\cdot n^{2+\log_2(t+1)}\cdot\log_2(n))$ bit operations, for $t \geq 2$.*

**Algorithm 1** Algorithm to calculate an upper bound for the number of Hamiltonian cycles of a given graph

Let the nodes of graph $G$ be denoted as $(1, \ldots, n)$ such that $\deg(G, i) \leq \deg(G, i+1)$ for $1 \leq i \leq n-1$ with $\deg(G, i)$ denoting the degree of node $i$. That is, the nodes are sorted so that the graph degree sequence is non-decreasing
Let node weights be either $w_i = i$, or $w_i \in B_t[g]$, for $1 \leq i \leq n$ constructed using definitions of Lemma 3.5 for $t = 2$, and Lemma 3.9 for $t > 2$
Let $p = (n^{n-1} + 1)$, $p_i = p^{w_i}$
Let $\sigma(w, n) = \sum_{i=1}^{n} w_i$, and the modulus $q^2 = \prod_{i=1}^{n} p_i^2 = p^{2 \cdot \sigma(w,n)}$
$M_{i,j} = \mathbb{1}_{i,j} \cdot p_i \cdot p_j$ that is equal to $p_i \cdot p_j$ if $(i, j)$ is a graph edge, and 0 otherwise
**for** $i = 1, \ldots, n$ **do**
  **for** $j = i+1, \ldots, n$ **do**
    Swap $p_i, p_j$
    Calculate $\beta$ using efficient matrix multiplication and modular arithmetic

$$\beta_{i,j} = \frac{M_{i,i}^{(n)} \bmod p^{2 \cdot \sigma(w,n)+1} - M_{i,i}^{(n)} \bmod p^{2 \cdot \sigma(w,n)}}{p^{2 \cdot \sigma(w,n)}}$$

    **if** $\beta_{i,j} = 0$ **then return** $[0, \pi]$
    **end if**
    **if** $\beta_{i,j} < \beta$ **then** $\beta = \beta_{i,j}$
    **end if**
    Swap $p_i, p_j$
  **end for**
**end for**
**return** $[\beta, \pi]$

Proof of Theorem 3.14. Let the node weight $p_n = f(n)$ be a function of $n$. Then by Eq. 1, and mathematical induction we obtain,

$$M_{i,j}^{(t)} \leq n^{t-1} f(n)^{2 \cdot t}.$$

The $n^{t-1}$ factor of the inequality is because of the $n$ terms of addition, and the $f(n)^{2 \cdot t}$ factor is because of the multiplicative terms, since $p_i \leq p_n = f(n)$. Thus, $M_{i,j}^{(n)} < n^{n-1} \cdot p_n^{2 \cdot n}$.

We note for a graph with $n$ nodes that the maximum number of closed paths of a graph with $n-1$ intermediate nodes is at most $n^{n-1}$, and the maximum number of Hamiltonian cycles is $(n-1)!$. Thus, we initialize node weights for the algorithm to calculate the number of Hamiltonian cycles so that $p = (n^{n-1} + 1)$ is greater than the number of all paths of length $n$. The weights of each node are $p_i = p^{w_i}$. Thus, we obtain $M_{i,j}^{(n)} \leq n^{n-1} \cdot p_n^{2 \cdot n} = n^{n-1} \cdot p^{2 \cdot n \cdot w_n}$.

Utilizing Definition 3.8 of $B_t[g]$ generalized Sidon set,

$$w_i \leq w_n = (t+1)^{\log_2(n)+2} = (t+1)^2 \cdot n^{\log_2(t+1)}.$$

Additionally, $2 \cdot \sigma(w, n) = 2 \cdot \sum_{k=1}^{n} w_k = O(n \cdot (t+1)^{\log_2(n)+1})$ because the $(t+1)$-adic digits of the sum of the node weights $w_k$ is $\frac{n}{2}$, we obtain

$$p^{2 \cdot \sigma(w,n)} = p^{(t+1) \cdot n^{\log_2(t+1)+1}}.$$

Since $p = (n^{n-1} + 1)$, the number of bits to represent $M_{i,i}^{(n)}$ is at most

$$\left| M_{i,i}^{(n)} \right| \leq (n-1) \cdot \log_2(n) + 2 \cdot n \cdot w_n \cdot \log_2(p) = (n-1) \cdot \log_2(n) + 2 \cdot (t+1)^2 \cdot n^{2+\log_2(t+1)} \cdot \log_2(n).$$

Thus we have proved the time complexity of the algorithm. □

With $t'$ tending from a constant to a function of $n$, the time complexity can be formulated as fixed parameter complexity [FK92].

### 3.4 Sub-exponential and exponential time algorithms

The discussion of the previous subsection implies a sub-exponential time algorithm with $(n+1)$-adic Sidon weights. The sub-exponential time complexity notation is discussed in Section B.1.

Lemma 3.15. *The time complexity of Algorithm 1 using $(n+1)$-adic node weights is $O(n^{4+\log_2(n+1)} \cdot \log_2(n))$ bit operations, that is a quasi-polynomial time algorithm, or sub-exponential time algorithm with parameter $\frac{\log_2(4+\log_2(n+1))}{\log_2(\frac{n}{\log_2(n)})}$, and at most $(\frac{3}{2} \cdot n)^{n-1} \cdot e^{-\frac{(n-1)}{3}}$ closed paths of length n, and of weight $p^{2\cdot\sigma(w,n)}$.*

Proof of Lemma 3.15. Using the Definition 3.8 of $B_t[g]$ node weights, the weight of a closed path of length $n$ to consider is $p^{2\cdot\sigma(w,n)}$, with $2 \cdot \sigma(w,n) = n \cdot (n+1)^{\log_2(n)+2}$, since each of the $(t+1)$-adic coefficients are equal to $\frac{n}{2}$. Using Eq. 6 with $s = \frac{t}{2}$, and $t = n$ leads to a correct calculation of the upper bound of $\alpha$ using $(t+1)$-adic coefficient of the sum of $n$ node weights with $s \leq \frac{n}{2}$, because the coefficient equality conditions are conjunctive. And this argument proves the lemma. □

For a sub-exponential algorithm, we will have to account for the sub-linear factors of the time complexity of integer multiplication, or utilize advanced algorithms [HvdH21].

Lemma 3.15 implies that the number of Hamiltonian cycles $\alpha \leq (n-1)! \leq (\frac{3}{2} \cdot n)^{n-1} \cdot e^{-\frac{(n-1)}{3}}$, and thus able to calculate an upper bound for $\alpha$. Algorithm 1 with sub-exponential instead of polynomial node weights should obtain a better upper bound, though we are not able to quantify the approximation. We note that for a sum of $n$ arbitrary numbers from the set of node weights $\{w_i\}$ to form a closed path of length $n$, so that each $\log_2(n)$ many $(n+1)$-adic coefficients of the closed path is equal to $\frac{n}{2}$ will be less probable for sparse graphs, unless there is a Hamiltonian cycle.

Thus, we are not able to prove that Algorithm 1 refutes the Exponential Time Hypothesis. Using exponential weights, instead of aforementioned sub-exponential weights, leads to an exact algorithm.

Theorem 3.16. *If $p_i = p^{w_i}$ with either $w_i = (\frac{n}{2}+j)!$ or $(\frac{n}{2}+1)^i$, then the algorithm calculates $\alpha$ exactly in time that is exponential of n.*

Proof of Theorem 3.16. The set $\{w_j\}$ is a $B_n[1]$ Sidon set, and with each path consisting of up to $\frac{n}{2}$ repeated nodes all path weights are distinct, because so are subset sums for closed paths of length $n$. We note that $\sigma(w,n) = \sum_{i=1}^{n}(\frac{n}{2}+j)!$, or $\sigma(w,n) = \sum_{i=1}^{n}(\frac{n}{2}+1)^i$ respectively. Since $w_i$ is exponential, and $p_i$ are doubly exponential, the algorithm is not polynomial time. □

### 3.5 Computational verification

Algorithmically counting the correct number of Hamiltonian cycles exactly to verify the upper bound of the polynomial time algorithm is computationally expensive for large graphs. The upper bound algorithm implementation gives exact, or strict upper bounds for varying density of edges for random graphs, and these results have been verified up to $n = 20$ nodes for graphs of varying density, and up to $n = 40$ for sparse non-Hamiltonian graphs. This computation is with $B_2[1]$ Sidon sets, and for power series of $p = (n^{n-1}+1)$, and for power series of symbolic univariate variable using the SageMath computational algebra system [Tea26].

## 4 Pseudorandomness, Learnability, and Computation

The previous section illustrates that a plausible reason for intractability is because we are given a number that is the sum of two variables, the first variable is the required answer, and this has to be distinguished from the second summand, and enumeration is computationally expensive. We note this phenomenon also occurs in the context of cryptography, and this identifies easy integer factoring instances that can be solved efficiently utilizing a polynomial time reduction to the discrete logarithm problem.

We note that a discussion of computational time complexity is in terms of the instance size, for counting the number of Hamiltonian cycles this is the number of graph nodes and edges, whereas this is the number of binary digits of the number to be factorized, and the number of binary digits to represent the cardinality of the prime finite field, and the elements for the discrete logarithm problem respectively.

The definitions of the computational problems discussed in this section are as follows.

- $\mathbb{INTFACT}_{r \cdot q}$ Given an integer $y = r \cdot q$ that is a product of two prime numbers, to determine $r$ and $q$.
- $\mathbb{DISLOG}$ Given an element of a prime finite field $y = g^x \in \mathbb{F}_p^*$ to determine $x$ with $g$ as generator of the multiplicative group.

Lemma 4.1. *Given $y = r \cdot q = g^x \in \mathbb{F}_p^*$, a product of primes $r$ and $q$ with $r = g^{x_r}$, $q = g^{x_q}$, if the binary addition of $x = x_r + x_q \bmod p - 1$ does not require binary digit carries, and if the sum of binary digits of $x$ is a polynomial of $\log \log p$, then*

$$\mathbb{INTFACT}_{r \cdot q} \leq \mathbb{DISLOG}$$

More generally, the reduction is exponential of $\log_2 p$ as $O(2^{\log_2 p})$ enumerations of $x_p$ are required to factorize.

Proof of Lemma 4.1. We are given a number $y = r \cdot q$ that is a product of two prime numbers and an algorithm for $\mathbb{DISLOG}$. We select a prime finite field $\mathbb{F}_p^*$ such that $y < p$. Candidate factors of $y$ are calculated as follows. If the binary digit of $x$ is 0, then the corresponding binary digit of $x_r$ is 0, otherwise the corresponding binary digit of $x_r$'s are either 0 or 1. If the sum of binary digits of $x$ is a polynomial of $\log \log p$, then by enumeration of possible $x_r$, the prime factors can be calculated in polynomial time. □

To determine that the graph is Hamiltonian could be better understood when a sequence of random graphs phase transitions from being a non-Hamiltonian to a Hamiltonian graph depending on the sparsity of graph edges [FK15]. A phase transition could be indicative of the learnability of a multi-layer perceptron or a graph neural network that is represented by iterative matrix multiplication with appropriate activation functions. Solving computational tasks by setting node weights instead of learning neural network parameters could be consequential to better optimization of their parameter, and sample complexity. Previous work [KV94], and the aforementioned theorems suggest that pseudorandomness is a plausible reason for the intractability of learning, counting and computation.

## 5 Conclusions

We have presented polynomial, and sub-exponential time algorithms to calculate an upper bound for the number of Hamiltonian cycles of a graph, and approximation guarantees. The idea of non-deterministic computation to calculate aggregate information using matrix multiplication could be used to improve algorithm design using the theory of computational complexity. This leads to questions of interest connecting arithmetic computation, cryptography, stochastic processes, and computational learning.

## Acknowledgments

AI tools were used sparingly for literature survey only.

## Ethics and Privacy Statement

We remark that this paper presents algorithm innovation and has minimal societal risks.

## References

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, Cambridge, UK, 2009.

[AKS04] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. Primes is in p. *Annals of Mathematics*, 160(2):781–793, 2004. doi:10.4007/annals.2004.160.781.

[AOGV18] Nima Anari, Shayan Oveis Gharan, and Cynthia Vinzant. Log-concave polynomials, entropy, and a deterministic approximation algorithm for counting bases of matroids. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 35–46. IEEE, 2018. doi:10.1109/focs.2018.00013.

[AS00] Noga Alon and Joel Spencer. *The Probabilistic Method*. John Wiley & Sons, Ltd, 2000. doi:10.1002/0471722154.

[AV77] Dana Angluin and Leslie G. Valiant. Fast probabilistic algorithms for hamiltonian circuits and matchings. In *Proceedings of the Ninth Annual ACM Symposium on Theory of Computing*, STOC '77, page 30–41, New York, NY, USA, 1977. Association for Computing Machinery. doi:10.1145/800105.803393.

[BG77] Gary S. Bloom and Solomon W. Golomb. Applications of numbered undirected graphs. *Proceedings of the IEEE*, 65:562–570, 1977. URL: https://api.semanticscholar.org/CorpusID:20782388.

[BH05] Iftikhar A. Burhanuddin and Ming-Deh A. Huang. Elliptic curve torsion points and division polynomials. In *Computational Aspects of Algebraic Curves*, pages 13–37, 2005. doi:10.1142/9789812701640_0002.

[BK22] Andrei A. Bulatov and Amirhossein Kazeminia. Complexity classification of counting graph homomorphisms modulo a prime number. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2022, page 1024–1037, New York, NY, USA, 2022. Association for Computing Machinery. doi:10.1145/3519935.3520075.

[CDM17] Radu Curticapean, Holger Dell, and Dániel Marx. Homomorphisms are a good basis for counting small subgraphs. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2017, page 210–223, New York, NY, USA, 2017. Association for Computing Machinery. doi:10.1145/3055399.3055502.

[Cil17] Javier Cilleruelo. A greedy algorithm for bh[g] sequences. *Journal of Combinatorial Theory, Series A*, 150:323–327, 2017. URL: https://www.sciencedirect.com/science/article/pii/S0097316517300377, doi:10.1016/j.jcta.2017.03.010.

[CLRS09] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, 3rd Edition*. MIT Press, 2009. URL: https://mitpress.mit.edu/books/introduction-algorithms.

[FK92] Michael Fellows and Neal Koblitz. Fixed-parameter complexity and cryptography. *International Symposium on Applied Algebra, Algebraic Algorithms, and Error-Correcting Codes*, 1992.

[FK15] Alan Frieze and Michał Karoński. *Random Graphs*. Cambridge University Press, 2015.

[Fon26] José A. R. Fonollosa. Ryser, glynn, and the discrete fourier transform: orthogonal schemes for the permanent. *Arxiv 2607.09949*, 2026. URL: https://arxiv.org/abs/2607.09949.

[GKW16] Pierre-Louis Giscard, Nils M. Kriege, and Richard C. Wilson. A general purpose algorithm for counting simple cycles and simple paths of any length. *Algorithmica*, 81:2716 – 2737, 2016. URL: https://api.semanticscholar.org/CorpusID:15676701.

[HM71] Frank Harary and Bennet Manvel. On the number of cycles in a graph. *Matematický časopis*, 21(1):55–63, 1971.

[HvdH21] David Harvey and Joris van der Hoeven. Integer multiplication in time o(n log n ). *Annals of Mathematics*, 2021.

[JS89] Mark Jerrum and Alistair Sinclair. Approximating the permanent. *SIAM Journal on Computing*, 18(6):1149–1178, 1989. doi:10.1137/0218077.

[Kaw01] Ken-ichi Kawarabayashi. A survey on hamiltonian cycles. *Interdisciplinary Information Sciences*, 7(1):25–39, 2001. doi:10.4036/iis.2001.25.

[KKG21] Anna R. Karlin, Nathan Klein, and Shayan Oveis Gharan. A (slightly) improved approximation algorithm for metric tsp. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2021, page 32–45, New York, NY, USA, 2021. Association for Computing Machinery. doi:10.1145/3406325.3451009.

[Kob94] Neal Koblitz. *A Course in Number Theory and Cryptography*. Springer, 1994.

[KV94] Michael Kearns and Leslie Valiant. Cryptographic limitations on learning boolean formulae and finite automata. *Journal of ACM*, 1994.

[LGCF21] J. A. Gregor Lagodzinski, Andreas Göbel, Katrin Casel, and Tobias Friedrich. On counting (quantum-)graph homomorphisms in finite fields of prime order. In *48th International Colloquium on Automata, Languages, and Programming (ICALP 2021)*, volume 198 of *LIPIcs*, pages 91:1–91:15, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. URL: https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.ICALP.2021.91, doi:10.4230/LIPIcs.ICALP.2021.91.

[LP25] Aditi Laddha and Madhusudhan Reddy Pittu. An algorithmic upper bound for permanents via a permanental schur inequality, 2025. URL: https://arxiv.org/abs/2509.08121.

[MP09] Christophe Meyer and Periklis A. Papakonstantinou. On the complexity of constructing golomb rulers. *Discrete Applied Mathematics*, 157(4):738–748, 2009. URL: https://www.sciencedirect.com/science/article/pii/S0166218X08003053, doi:10.1016/j.dam.2008.07.006.

[NC26] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2026.

[Ope26] OpenAI. The exponential time hypothesis is false. *Preprint*, 2026. URL: https://github.com/openai/math/blob/main/preprints/The-Exponential-Time-Hypothesis-is-False-October-25-2026/paper.pdf.

[Pom96] Carl Pomerance. A tale of two sieves. *Notices of the AMS*, 43(12):1473–1485, 1996.

[PRS+23] Norbert Peyerimhoff, Marc Roth, Johannes Schmitt, Jakob Stix, Alina Vdovina, and Philip Wellnitz. Parameterized counting and cayley graph expanders. *SIAM Journal on Discrete Mathematics*, 37(2):405–486, 2023. doi:10.1137/22M1479804.

[Rot26] Marc Roth. Parameterised counting complexity theory. *Computer Science Review*, 59:100837, 2026. URL: https://www.sciencedirect.com/science/article/pii/S1574013725001133, doi:10.1016/j.cosrev.2025.100837.

[Sho94] Peter W Shor. Algorithms for quantum computation: discrete logarithms and factoring. *Proceedings 35th annual symposium on foundations of computer science*, pages 124–134, 1994.

[Sil09] Joseph Silverman. *The Arithmetic of Elliptic Curves*. Springer, 2009.

[SS11] René Schott and G. Stacey Staples. Complexity of counting cycles using zeons. *Computers & Mathematics with Applications*, 62(4):1828–1837, 2011. doi:10.1016/j.camwa.2011.06.026.

[Tea26] SageMath Team. Sagemath computer algebra system v10.8, 2026. URL: https://www.sagemath.org.

[Val79] Leslie G. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8(2):189–201, 1979. doi:10.1016/0304-3975(79)90044-6.

[VdR10] Carlos Vinuesa del Río. *Generalized Sidon Sets*. Phd dissertation, Universidad Autónoma de Madrid, 2010.

[Wig18] Avi Wigderson. *Mathematics and Computation: A Theory Revolutionizing Technology and Science*. Princeton University Press, 2018.

[Wil05] Herbert Wilf. *generatingfunctionology*. A K Peters/CRC Press, 2005. doi:10.1201/b10576.

[WW20] Virginia Vassilevska Williams and Ryan Williams. Fine-grained algorithms and complexity. *Lecture Notes*, 2020. URL: https://people.csail.mit.edu/virgi/6.s078/.

## A Appendix - Related Computational Problems

This section presents related computational problems such as determining whether the graph is Hamiltonian, or finding a Hamiltonian cycle, and those problems that could be solved partially, or such as vertex cover, traveling salesman, and matrix permanent using similar algorithms.

### A.1 Determining whether a graph is Hamiltonian

The upper bound calculated by Algorithm 1 is informative about whether the graph is non-Hamiltonian for instance when the upper bound $\beta = 0$. The node degree sequence of a graph is informative about Hamiltonian cycles, for instance if each node degree is at least $\frac{n}{2}$ [Kaw01]. Additionally, if the density of edges is $O(n \log n)$ then a random graph is Hamiltonian [FK15]. These results about node degrees, edge sparsity or density, and combinatorial random walks could be used to improve Algorithm 1.

We note that a graph with $n + 1$ nodes with the first node with an edge to a second node that is part of a complete graph of $n$ nodes is not Hamiltonian. This example would result in an upper bound that is distinct from the number of Hamiltonian cycles. The degree of the first node is 1, it is known that if a graph has a Hamiltonian cycle, then the node degrees have to be greater than 1, so this graph instance can be labeled non-Hamiltonian without using the algorithm.

### A.2 Finding a Hamiltonian cycle

Algorithm 1 optimizes the complexity of counting Hamiltonian cycles, and closed paths of the same weight, and so does not compute a certificate path that can be verified. To identify a Hamiltonian cycle, consider a graph with more than $n$ number of edges, and that is probably Hamiltonian. If the graph is sparsified, and $\alpha$ is calculated, then this sequence of graphs generated will eventually result in a graph with $n - 1$ edges and no Hamiltonian cycle. Thus, the transition in

the graph sequence from a graph that is probably Hamiltonian to a graph that is not Hamiltonian will be informative about the graph being Hamiltonian using the sequence of graphs.

### A.3 Vertex Cover

There is a reduction of the vertex cover problem to the Hamiltonian cycle problem by constructing a graph using widget subgraphs and $k$ selector nodes so that the given graph has a vertex cover of size $k$ if and only if the constructed graph is Hamiltonian [CLRS09]. Thus, an algorithm for the Hamiltonian cycle problem is useful to solve the vertex cover problem.

### A.4 Traveling Salesman Problem

To generalize Algorithm 1 to weighted graphs requires obtaining information from the sum of the product of edge weights of tours. Let the $\mathbb{TSP}$ problem for a graph with edge weights $q_{i,j}$ that are prime numbers be denoted by $\mathbb{TSP}_{\prod}$. For $\mathbb{TSP}_{\prod}$, let $M_{i,j} = \mathbb{1}_{i,j} \cdot q_{i,j} \cdot p_i \cdot p_j$, with $q_{1,1} < \ldots < q_{n,n} < p_1 < \ldots < p_n$. Iteratively dividing $\beta$ by the edge weights $q_{i,j}$ or calculating $\gcd(\beta, \prod q_{i,j})$ will determine the edges that are part of all tours up to modular arithmetic.

### A.5 Hamiltonian Paths and Matrix Permanent

A redefinition of Algorithm 1 obtains upper bounds for the number of $s-t$ Hamiltonian paths from node $s$ to $t$ of length $n$. Summing across all source-sink nodes calculates the total number of Hamiltonian paths.

The permanent of a matrix $A = [A_{i,j}]$ is defined as

$$\text{perm}(M) = \sum_{\sigma \in S_n} \prod_{i=1}^{n} A_{i,\sigma(i)}$$

with $S_n$ denoting the symmetric group of $n$ elements consisting of $n!$ permutations of $\{1, 2, \ldots, n\}$. The permanent computation is similar to the determinant of a matrix formula without considering the signs of the permutations of the nodes. The computation of the permanent of a $(0,1)$-matrix is $\#\mathbb{P}$-complete. The computation of $M_{i,i}^{(n)}$ denoted the sum of weighted closed paths of length $n$ including Hamiltonian paths starting at a specific node $i$ to $i$, while the permanent computation considers sums of weighted $s-t$ Hamiltonian paths of length $n$, with a summation across all nodes. The algorithm for matrices with the generalized Sidon node weights 3.2 will obtain a better approximation of their permanent.

## B Appendix - Sub-exponential complexity, alternative node weights, and modular arithmetic

### B.1 Time complexity notation

The $L$-notation is used to denote the asymptotic time complexity of algorithms from polynomial to exponential time with the problem instance input size appropriately parameterizing the notation [Kob94, Pom96]. Let $exp(x)$ denote the exponential function $e^x$, or $2^x$, and the respective logarithm bases.

*B.1.1 Number theoretic time complexity.* If $n$ is the integer to be factorized, or whose discrete logarithm is to be computed, then the input requires $\log_2 n$ number of binary digits representation, and the corresponding $L$-notation is

$$L[n, d] = exp(O(1) \cdot (\log n)^d \cdot (\log \log n)^{1-d}), \ with \ d \geq 0.$$

The time complexity to find a target integer in a list of sorted integers with the binary search algorithm is

$$L[n, 0] = exp(\log \log n) = O(\log n)$$

With the assumption that comparing the magnitude of two integers for equality is $O(1)$ units of computation. The number of divisions required to determine if a number is prime with iterative division is

$$L[n, 1] = exp(\frac{1}{2}(\log n)) = O(\sqrt{n})$$

So an algorithm that represents a number $n$ using log$n$ bits, and $d = 0$ for the L-notation represents a polynomial time of $\log n$ algorithm, and $d = 1$ represents an exponential time of $\log n$ algorithm.

The factor-base sieve algorithms for DISLOG the discrete logarithm problem for prime finite fields and INTFACT the integer factoring problem are heuristically $L[n, \frac{1}{3}]$ sub-exponential complexity.

*B.1.2 Combinatorial algorithm time complexity.* For combinatorial graph problems, if $n$ is the number of vertices of the graph, the $L$-notation is

$$L[n, d] = exp(O(1) \cdot n^d \cdot (\log n)^{1-d}), \ with \ d \geq 0.$$

We note that the aforementioned L-notation to denote the magnitude of integers can be rewritten as follows $L[n, d] = exp(O(1) \cdot (\frac{n}{\log n})^d \cdot \log n)$. This leads to a combinatorial formulation of time complexity

$$L[n, d] = n^{O(1) \cdot (\frac{n}{\log n})^d}.$$

### B.2 Alternative node weights

This section discusses alternative node weights that can be used to calculate upper bounds of the number of Hamiltonian cycles. There are several choices of encoding, for instance powers of integers, prime numbers, product of prime numbers, and polynomial symbolic variables with appropriate modular arithmetic, or modular polynomial algebraic computation. For different choices of nodes weights $p_i$ and the modulus $q$, Algorithm 1 calculates

$$\beta = \frac{M_{i,i}^{(n)} - M_{i,i}^{(n)} \bmod q^2}{q^2}.$$

For sparse graph instances the algorithm calculates the exact number of Hamiltonian cycles, otherwise a strict upper bound is obtained indicating computational intractability. We note that the algorithm could use several polynomial number of choices of node weights to minimize the upper bound.

For efficiency, the node weights $p_i$ can be optimized instead of optimizing by setting $p_i = p^{w_i}$ and using Sidon weights for $w_i$ if the modular arithmetic logic guarantees algorithm correctness.

### B.3 Identical node weights

A simple upper bound is obtained using $p_j = p$ is a positive integer greater than 1 for each node $j$, then $q = \prod_{j=1}^{n} = p^n$, and

$$\beta \quad = \quad \frac{M_{i,i}^{(n)} - M_{i,i}^{(n)} \bmod q^2}{q^2} = \#\{\gamma \mid \text{closed paths from } i \text{ to } i \text{ of length } n\}.$$

And this calculation includes Hamiltonian cycles and other closed paths from $i$ of length $n$.

### B.4 Node weights are prime numbers

Let the node weights $p_j$ be distinct prime numbers such that $n < p_1 < p_2 < \ldots < p_n$. And $q = \prod_{i=1}^{n} p_j$. Then

$$\begin{aligned} M_{i,i}^{(n)} &= x_0 + x_1 \cdot q + \sum_{j \geq 2} x_j \cdot q^j = \alpha \cdot q^2 + z_0 + z_1 \cdot q + \sum_{j \geq 2} z_j \cdot q^j \\ &= z_0 + z_1 \cdot q + (\alpha + z_2) \cdot q^2 + z_3 \cdot q^3 + O(q^4). \end{aligned}$$

with $0 \leq x_j, z_j < q$. The notation $O(q^4)$ denotes terms greater than the third power of $q$. To define algorithm computation, the previous equations can be written

$$\begin{aligned} \beta &= \frac{M_{i,i}^{(n)} - M_{i,i}^{(n)} \bmod q^2}{q^2} \\ &= \sum_{j \geq 2} x_j \cdot q^{j-2} = \alpha + z_2 + z_3 \cdot q + \sum_{j \geq 4} z_j \cdot q^{j-2} \\ &= \beta_0 + (\gamma + z_3) \cdot q + \sum_{j \geq 4} z_j \cdot q^{j-2} \end{aligned}$$

Here $\beta_0 = \beta \bmod q = x_2 = (\alpha + z_2) \bmod q$. And $\gamma = \lfloor \frac{\alpha + z_2}{q} \rfloor$. The values of $\gamma$ are equal to 0, or 1, since $0 \leq \alpha, z_2 < q$, and $\alpha + z_2 < 2 \cdot q$. Specifically, this discussion implies the following scenarios, if $\beta = 0$, then $\alpha = 0$; $0 < \beta = \beta_0 = \alpha + z_2 < q$; and $\beta \geq q$.

*B.4.1 Arithmetic modulo products of primes.* If the nodes weights are the first $n$ consecutive primes $p_1 = 2, p_2 = 3, \ldots$, and the $n^{\text{th}}$ node weight is $p_n$. That could be obtained by sieving, or using a primality algorithm [AKS04]. By the Prime Number Theorem, $p_n \sim n \cdot \log n$ asymptotically. Simple bounds for the primorial function are $2^n \leq q = \prod_{j=1}^{n} p_j < p_n^n = (n \cdot \log n)^n$. Thus

$$M_{i,j}^{(n)} < n^{n-1} \cdot (n \cdot \log n)^{2 \cdot n} = n^{3 \cdot n - 1} \cdot (\log n)^{2 \cdot n}.$$

This implies that

$$\beta = \frac{M_{i,i}^{(n)} - M_{i,i}^{(n)} \bmod q^2}{q^2} < \frac{n^{3 \cdot n - 1} \cdot (\log n)^{2 \cdot n}}{2^{2 \cdot n}} = n^{3 \cdot n - 1 + 2 \cdot n \cdot (\frac{\log \log n - 1}{\log n})} < q^3.$$

The list of primes can be obtained efficiently by sieving the integers. The first $n$ consecutive primes can be obtained using the list of the first $O(n \log n)$ consecutive integers, and by dividing subsequent numbers that are multiples. This determines the prime and composite numbers in the list.

### B.5 Node weights are consecutive integers

Alternatively, for efficient computation, node weights can be positive integers greater than 1, and less than or equal to $n + 1$, thus $q = (n + 1)!$. Using Stirling's formula of

$$n! \sim \sqrt{2\pi n} \cdot (\frac{n}{e})^n$$

the approximate upper bounds of the algorithm variables are $M_{i,i}^{(n)} < n^{n-1} \cdot (n + 1)^{2 \cdot n}$, and

$$\beta < n^{n-1} \cdot \frac{(n+1)^{2 \cdot n}}{((n+1)!)^2} < (n+1)^{(n+1) \cdot (1 + \frac{2 \cdot \log e}{\log (n+1)})} = q \cdot e^{3 \cdot (n+1)} < q^2.$$

### B.6 Node weights are consecutive integers

For graphs with $n = p - 2$ nodes with prime $p$, the node weights $p_i$ can be defined to be elements of cyclic group $\mathbb{F}_p^*$ excluding 1. Given that the set of node weights $\{g, g^2, \ldots, g^{p-2}\}$ is bijective to the set $\{2, 3, \ldots, p-1\}$, we obtain information about

$$\left[\alpha \cdot ((p-1)!)^2 + \sum_{\gamma,\ w(\gamma) \neq ((p-1)!)^2} w(\gamma)\right] \bmod p^j$$

with $\alpha \leq (n-1)! < (p-2)!$.

Since the weight of a Hamiltonian cycle modulo $p$ is $((p-1)!)^2 = \prod_{j=1}^{p-2}(g^j)^2 = (g^{\frac{(p-2)\cdot(p-1)}{2}})^2 = 1 \bmod p$, the computation modulo $p$ is $\alpha$ plus the contribution of the weight of the closed paths. And for calculations modulo $p^2$, since the Euler totient function of $p^2$ is $\phi(p^2) = p(p-1)$, the weight of a Hamiltonian cycle is $((p-1)!)^2 = \prod_{j=1}^{p-2}(g^j)^2 = (g^{\frac{(p-2)\cdot(p-1)}{2}})^2 = g^{(p-2)\cdot(p-1)} = g^{(2-2\cdot p)} \bmod p^2$. If conditions can be obtained so that there is cancellation of the sum of weights of closed paths, then good upper bound of $\alpha$ can be calculated.

### B.7 Algebraic Computation

Algorithm 1 with base $p = n^{n-1} + 1$ can be redefined for base $p \cdot x$, with computation defined by the univariate polynomial ring $\mathbb{Z}[x]/(x^4 + 1)\mathbb{Z}[x]$, or another irreducible polynomial of constant degree. Considering the powers of $x$, the coefficient of $(p \cdot x)^{2\cdot\sigma(w,n)} = \pm p^{2\cdot\sigma(w,n)} \cdot x^k$ is an upper bound with the sign, and $k = \{0, 2\}$ depending on whether $\sigma(w, n)$ is even or odd. Because the closed paths of same weight are further distinguishable, this leads to better bounds for the number of Hamiltonian cycles.