Electronic Colloquium on Computational Complexity, Report No. 61 (2001)

The Complexity of Computing the Number of Self-Avoiding Walks in Two-Dimensional Grid Graphs and in Hypercube Graphs

Mitsunori Ogihara[*]
Department of Computer Science
University of Rochester

Seinosuke Toda[†]
Department of Applied Mathematics
Nihon University

July 13, 2001

Abstract

Valiant (*SIAM Journal on Computing* 8, pages 410–421) showed that the problem of counting the number of s-t paths in graphs (both in the case of directed graphs and in the case of undirected graphs) is complete for #P under polynomial-time one-Turing reductions (namely, some post-computation is needed to recover the value of a #P-function). Valiant then asked whether the problem of counting the number of self-avoiding walks of length n in the two-dimensional grid is complete for $\#\mathrm{P}_1$, i.e., the tally-version of #P. This paper offers a partial answer to the question. It is shown that a number of versions of the problem of computing the number of self-avoiding walks in two-dimensional grid graphs (graphs embedded in the two-dimensional grid) is polynomial-time one-Turing complete for #P.

This paper also studies the problem of counting the number of self-avoiding walks in graphs embedded in a hypercube. It is shown that a number of versions of the problem is polynomial-time one-Turing complete for #P. By scaling up the completeness result for #P, it is shown that the same variety of problems is polynomial-time one-Turing complete for #EXP. Here the post-computation that is required is right-bit-shift by exponentially many bits and a hypercube graph is specified by a Boolean cicuit that accepts its nodes.

1 Introduction

Self-avoiding walks are random walks that do not intersect themselves. Computing the exact number of self-avoiding walks of a given length n on the two-dimensional lattice is well-known for its difficulty and has been studied extensively (see Madras and Slade [MS93] for the details of the problem, and also Welch [Wel93] for its connection to other related problems). No simple recurrences are known for the counting problem. The exact number has been calculated only for small lengths n. The largest n for which the exact number is computed is 51, by Conway and Guttman [CG96].

Valiant [Val79b] is the first to find connections between self-avoiding walks and computational complexity theory. He showed that the problem of counting the number of all simple s-t paths in graphs is #P-complete both for the directed graphs and for the undirected graphs. He then asked

[*]Address: BOX 270226, Department of Computer Science, University of Rochester, Rochester, NY 14627-0226, USA. Email: ogihara@cs.rochester.edu. Supported in part by NSF grants CCR-9701911, CCR-9725021, DUE-9980943, INT-9726724, an NIH/NIA grant RO1-AG18231 and a DARPA grant F30602-98-2-013.

[†]Address: Department of Applied Mathematics, Nihon University, 3-25-40 Sakurajyou-shi, Setagaya-ku, Tokyo 156, Japan. Email: toda@am.chs.nihon-u.ac.jp.

ISSN 1433-8092

whether the exact counting problem for the two-dimensional grid is complete for $\#\mathrm{P}_1$, the "tally" version of #P, provided that the length of walks is specified by a tally string.

While it is very easy to prove the membership of the problem in $\#\mathrm{P}_1$, settling on the question of its hardness is difficult. The difficulty comes from the fact that the two-dimensional grid has a very rigid structure. A straightforward approach for proving the hardness would be to embed the computation of a nondeterministic Turing machine in the grid, by defining a correspondence between its configurations and the grid points. However, since every line segment of the grid can be traversed in either direction and the configuration graph of a nondeterministic Turing machine is of high dimension, such an embedding seems impossible.

This observation leads to the question of whether this counting problem is complete if one is allowed to eliminate some nodes and edges, namely counting the number of self-avoiding walks in two-dimensional grids, those composed of the nodes and the edges of the two-dimensional grid. Since elimination of edges and nodes is dependent on the input, here one should perhaps think of #P-completeness rather than $\#\mathrm{P}_1$-completeness of the problem. So, the question asked is:

> Is the problem of computing the number of self-avoiding walks in two-dimensional grid graphs #P-complete?

This paper provides a positive answer to the question. The problem is #P-complete under polynomial-time one-Turing reductions if the kind of self-avoiding walks to be counted is: (1) restricted to those between two specific nodes, (2) restricted to those starting from a specific node, or (3) totally unrestricted. However, it is unknown whether the problem is #P-complete under any of these restrictions if the length of the walks to be counted is specified. The kind of one-Turing reduction imposed here is quite simple; the only post-computation performed is to divide the number by a power of two; i.e., to shift the bits of the binary representation of the number.

Another question asked in this paper is whether the counting problem is complete if the dimension of the grid is dependent on the input; i.e., whether it is #P-complete to count the number of self-avoiding walks in hypercube graphs, the graphs consisting of nodes and edges of a hypercube. This paper gives a positive answer to the question, too. The problem is #P-complete under polynomial-time one-Turing reductions if the kind of self-avoiding walks to be counted is: (1) restricted to those between two specific nodes, (2) restricted to those starting from a specific node, and (3) totally unrestricted. Furthermore, the problem is complete even when the length of the walks to be counted is specified. When we reduce a #P-function to the counting problem of self-avoiding walks in a hypercube, the dimension of the hypercube in which the graph is embedded is $\mathcal{O}(\log n)$. It is natural to ask about the complexity of the counting problem when the dimension of the hypercube graphs increases as a polynomial in the input size. Since there are 2^n nodes in the n-dimensional hypercube, it is stipulated here that a boolean circuit is used to describe the structure of a hypercube graph. More precisely, for a boolean citcuit, C, of n inputs, the hypercube graph specified by C is the graph $G = (V, E)$ such that $V = \{w \in \{0,1\}^n \mid C(w) = 1\}$ and $E = \{(w, w') \in \{0,1\}^n \mid C(w) = C(w') = 1$ and the Hamming distance between w and w' is $1\}$. It is shown here that under that stipulation the countingproblem is #EXP-complete, the exponential-time version of #P.

2 Preliminaries

This section sets down some concepts and notation. The reader's familiarity with the basic notions in computational complexity theory is assumed; a reader unfamiliar with the subject may consult [Pap94].

Let M be a nondeterministic Turing. $\#acc_M$ denotes the function that maps each string x to the number of accepting computation paths of M on input x. Then $\#\mathrm{P} = \{\#acc_M \mid M$ is a polynomial-time nondeterministic Turing machine $\}$ and $\#\mathrm{EXP} = \{\#acc_M \mid M$ is an exponential-time nondeterministic Turing machine $\}$.[1]

Let f and g be two functions from Σ^* to $\mathbf{N}$. The function f is *polynomial-time one-Turing reducible* to g, denoted by $f \leq^p_{1\text{-}T} g$, if there is a pair of polynomial-time computable functions, $R_1 : \Sigma^* \to \Sigma^*$ and $R_2 : \Sigma^* \times \mathbf{N} \to \mathbf{N}$, such that for all x, $f(x) = R_2(x, g(R_1(x)))$. The function f is *polynomial-time right-shift reducible* to g, denoted by $f \leq^p_{r\text{-}shift} g$, if there is a pair of polynomial-time computable functions, $R_1 : \Sigma^* \to \Sigma^*$ and $R_2 : \Sigma^* \to \mathbf{N}$, such that for all x, $f(x) = g(R_1(x)) \operatorname{div} 2^{R_2(x)}$, where div is integer division.

3SAT is the problem of testing satisfiability of CNF formulas with exactly three literals per clause. #SAT is the function that counts the number of satisfying assignments of a 3CNF formula. By the standard reduction from nondeterministic Turing machine computations to 3CNF formulas, #SAT is complete for #P [Val79b].

Proposition 2.1 *For every* #P *function f, there is a polynomial-time computable function R such that for all x, $R(x)$ is a 3CNF formula and satisfies* $\#\mathrm{SAT}(R(x)) = f(x)$.

Hamilton Path (HP, for short) is the problem of deciding, given a graph G with two specified nodes s and t, whether there is a Hamilton path from s to t in G. #HP is the problem of counting, given a graph G and a node pair (s,t), the number of Hamilton paths from s to t in G.

3 Reducing #SAT to #HP

#P-completeness of the self-avoiding walk problems is proven based upon a reduction from #SAT to #HP, where the instances in #HP generated by the reduction have a low degree. More precisely, for the completeness in grid graphs, the graphs have to be planar and the maximum degree is ≤ 4, and for the hypercube graph results, the maximum degree is $\mathcal{O}(\log n)$.

To meet the requirement, modifications are given to the reduction by Garey, Johnson, and Tarjan [GJT76], which reduces SAT to HP of planar, 3-regular graphs (thereby showing that this special case of HP is NP-complete).[2] This reduction is referred to by the term GJT-reduction.

One cannot directly use their reduction to prove #P-completeness of #HP, because the number of Hamilton paths representing a satisfying assignment depends on how the assignment satisfies each clause. More precisely, let φ be a satisfiable 3CNF formula and (G,s,t) be the instance of HP generated by the GJT-reduction. Let A be the set of all satisfying assignments of φ and P be the set of all Hamilton paths from s to t in G. The GJT-reduction defines an onto mapping from P to A so that from each element path in P an element in A can be recovered in polynomial time. For each $a \in A$, let P_a denote the set of all Hamilton paths in P that map to a by this onto mapping. Clearly, $\|P\| = \sum_{a \in A} \|P_a\|$. It holds that $\|P_a\| = 2^p 3^q$, where p as well as q is a linear combination of the following three quantities: the number of clauses such that a satisfies all the three literals, the number of clauses such that a satisfies exactly two literals, and the number of clauses such that a satisfies exactly one literal.

Not-All-Equal-3-SAT [Sch78] (NAE3SAT, for short) is a special case of 3SAT in which it is asked to test whether a given CNF formula can be satisfied by an assignment that dissatisfies at

[1]The reader should be cautioned that Valiant defined #EXP as $\#\mathrm{P}^{\mathrm{EXP}}$ in [Val79a].

[2]Garey, Johnson, and Tarjan reduce 3SAT to the Hamilton Cycle problem. The same reduction can be used to show the completeness of HP, since for every graph generated by their reduction, there is an edge e that each Hamilton cycle has to traverse.

least one literal per clause. Schaefer [Sch78] shows that NAE3SAT is NP-complete. In this paper the formulas are prohibited to have clauses with only two literals but permitted to have clauses with only one literal. The following lemma states that there is a polynomial-time many-one reduction from 3SAT to NAE3SAT that preserves the number of satisfying assignments such that for all formulas ϕ produced by the reduction, for all satisfying assignments a of ϕ, and for all three-literal clauses C of ϕ, a dissatisfies at least one literal of C.

Lemma 3.1 *There is a polynomial-time computable mapping f from the set of all 3CNF formulas to the set of all CNF formulas consisting of clauses with either one or three literals such that for every 3CNF formula φ, the following conditions hold for $\psi = f(\varphi)$:*

1. *The number of variables of ψ is $n + 2m$ and the number of clauses of ψ is $9m$, out of which $8m$ are three-literal clauses and m are single-literal clauses, where n and m are respectively the number of variables and the number of clauses of φ.*

2. *$\#\mathrm{SAT}(\varphi) = \#\mathrm{SAT}(f(\varphi))$.*

3. *Every satisfying assignment a of ψ satisfies all the m single-literal clause of ψ, exactly two literals for exactly $4m$ three-literal clauses, and exactly one literal for exactly $4m$ three-literal clauses.*

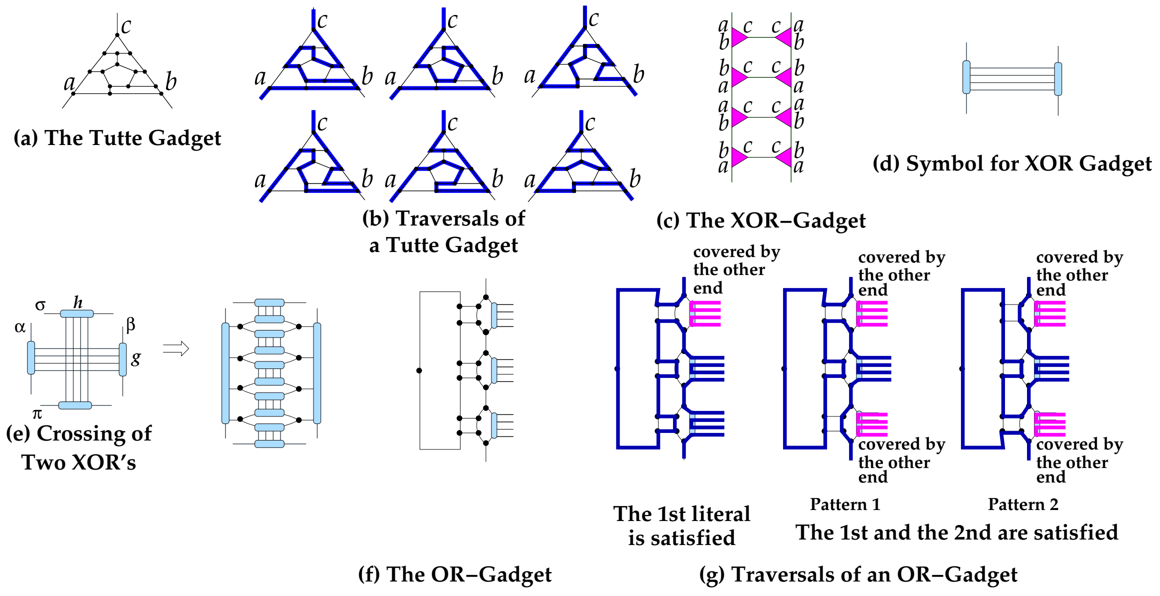
Proof. Let a 3CNF formula $\varphi = C_1 \wedge \cdots \wedge C_m$ be given. Let $1 \leq i \leq m$ and let x, y, z be the literals of C_i. Introduce a new variable u_i that is equivalent to $x \vee y$. The equivalence can be written as a CNF formula: $(\overline{u_i} \vee x \vee y) \wedge (u_i \vee \overline{x}) \wedge (u_i \vee \overline{y})$. Furthermore, introduce another new variable w_i that needs to be false by the clause $(\overline{w_i})$. Let

$$\begin{aligned} C_i' \; = \; & (\overline{u_i} \vee x \vee y) \wedge (u_i \vee \overline{x} \vee w_i) \wedge (u_i \vee \overline{y} \vee w_i) \wedge (u_i \vee z \vee w_i) \\ & \wedge (u_i \vee \overline{x} \vee \overline{y}) \wedge (\overline{u_i} \vee x \vee \overline{w_i}) \wedge (\overline{u_i} \vee y \vee \overline{w_i}) \wedge (\overline{u_i} \vee \overline{z} \vee \overline{w_i}) \wedge (\overline{w_i}). \end{aligned}$$

Then, every satisfying assignment a of C_i' satisfies exactly two literals for four three-literal clauses and exactly one literal for four three-literal clauses. Let $\psi = C_1' \wedge \ldots \wedge C_m'$. Then ψ has $n + 2m$ variables, consists of $8m$ three-literal clauses, m single-literal clauses, and has as many satisfying assignment as φ. Furthermore, for every satisfying assignment a of ψ, there are precisely $4m$ three-literal clauses such that a dissatisfies exactly one literal and precisely $4m$ three-literal clauses such that a dissatisfies exactly two literals. This proves the lemma. □

Let φ be a 3CNF formula for which the number of satisfying assignments needs to be evaluated. Suppose φ is defined over n variables and has m clauses. Let ψ be the formula generated by applying the transformation in Lemma 3.1 to φ. The GJT-reduction with slight modifications is applied to ψ.

Basic components of the construction are the Tutte-gadget (Figure 1 (a)), the XOR-gadget (Figure 1 (c)), and the OR-gadget (Figure 1 (f)). Here the first two gadgets are inherited from the GJT-reduction while the OR-gadget is new. The Tutte-gadget forces branching; to visit c without missing a node, one has to either enter from a and visit b on its way or enter from b and visit a on its way. There are four ways to do the former and two ways to do the latter (see Figure 1 (b)). The XOR-gadget is a ladder built using eight copies of the Tutte-gadget. In order to go through all the nodes in an XOR-gadget one has to enter and exit on the same vertical axis. For each of the two vertical axes there are $(4 \cdot 2)^4 = 2^{12}$ Hamilton paths. XOR-gadgets can be crossed without losing planarity by inflating the number of Hamilton paths (see Figure 1 (b) and (c)). Let g be an XOR-gadget connecting two vertical lines, α and β, and h be an XOR-gadget connecting two



(a) The Tutte Gadget
(b) Traversals of a Tutte Gadget
(c) The XOR–Gadget
(d) Symbol for XOR Gadget
(e) Crossing of Two XOR's
σ
h
α
β
g
π
(f) The OR–Gadget
covered by the other end
The 1st literal is satisfied
Pattern 1
Pattern 2
The 1st and the 2nd are satisfied
(g) Traversals of an OR–Gadget

Figure 1: The Gadgets

horizontal lines, σ and π. In order to cross these two XOR-gadgets, a two-node cycle is inserted in each of the four horizontal lines of g, the edges of the inserted cycles together with σ and π, and then each pair is connected by an XOR-gadget. Since four XOR-gadgets are added, the number of Hamilton paths is increased by a multiplicative factor of 2^{48}.

The graph is built upon two vertical sequences of two-node cycles. Each two-node cycle has the *top node* and the *bottom node* and the two nodes are joined by two edges, the *right edge* and the *left edge*. In the two sequences each neighboring cycle-pair is joined by an edge. The left sequence, called Σ, has $25m$ cycles, and the right sequence, called Π, has $2n + 4m$ cycles. The top node of Σ and that of Π are joined by an edge. There are one source node s and one sink node t. The bottom of Σ is joined to s by an edge while the bottom of Π is joined to t. The first $24m$ cycles of Σ are divided into $8m$ three-cycle blocks, where for each i, $1 \leq i \leq 8m$, the ith block corresponds to the ith three-literal clause of ψ, i.e., the first of the three cycles corresponds to the first literal of the clause, the second cycle to the second literal, and the third cycle to the third literal (see Figure 2). The three cycles in each three-cycle block are connected to each other by an OR-gadget. The remaining m cycles correspond to the m single-literal clauses of ψ. Each of these m cycles has a node in the middle of the left edge of the cycle. The sequence Π is divided into $n + 2m$ blocks of cycle-pairs, where for each i, $1 \leq i \leq n + 2m$, the ith pair corresponds to x_i, the ith variable. For each pair, the top cycle corresponds to $\overline{x_i}$ and the bottom to x_i. To these sequences add a number of XOR-gadgets:

- For each i, $1 \leq i \leq n + 2m$, add an XOR-gadget to connect the right edge of the top cycle and that of the bottom cycle in the ith cycle-pair in Π.
- For each i, $1 \leq i \leq 2(n + 2m)$, and each position j, $1 \leq j \leq 25j$, if the ith cycle of Π and the jth cycle correspond to the same literal, then join them by an XOR-gadget.
- If two XOR-gadgets connecting Σ and Π need to be crossed, it is done so by addition of an XOR-gadget as described earlier.

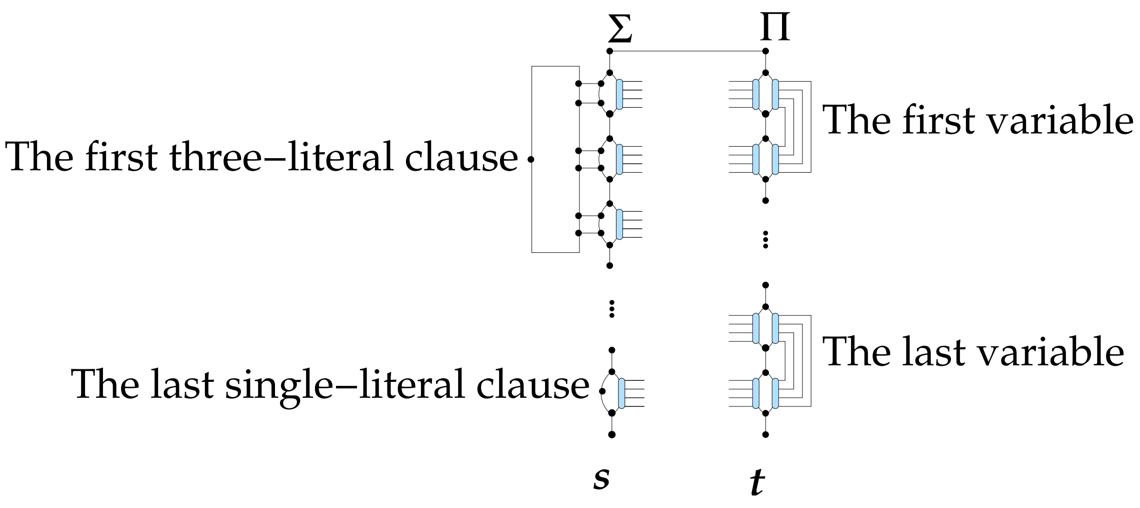

Σ
Π
The first three–literal clause
The first variable
The last single–literal clause
The last variable
s
t

Figure 2: The Two Sequences, Σ and Π

For every i, $1 \leq i \leq n+2m$, traversing the XOR-gadget within the ith cycle-pair in Π from the top cycle corresponds to assigning 1 to x_i, since it frees up the left edge of the bottom cycle; traversing it from the bottom corresponds to assigning 0 to x_i. Then for every i, $1 \leq i \leq n+2m$, from the "free" left edge of the ith cycle-pair, all the XOR-gadgets satisfied by the assignment correspond to the "free" left edge must be traversed. So, for every assignment a to ψ, and for every j, $1 \leq j \leq 8m$, the number of "free" left edges in the jth cycle-triple in Σ is the number of literals that a satisfies in the jth three-literal clause of ψ. In each OR-gadget, the number of ways to traverse all of its nodes is the number of literals that are satisfied (see Figure 1 (g)). Let Ξ denote the number of crossings of XOR-gadgets. Then, for every satisfying assignment a of ψ, the number of Hamilton paths (between s and t) corresponding to a is: $2^{4m} \cdot 2^{12(n+29m+4\Xi)} = 2^{12n+352m+48\Xi}$. The maximum degree of the resulting graph is 3, and the graph is planar. Combining these observations and Proposition 2.1 and Lemma 3.1 yields the following lemma.

Lemma 3.2 *Every #P function is $\leq^{p}_{r\text{-}shift}$-reducible to the problem of computing the number of Hamilton paths in planar graphs of maximum degree 3.*

4 Self-Avoiding Walks in Two-Dimensional Grid Graphs

Let f be an arbitrary #P function. Let x be a string for which $f(x)$ needs to be evaluated. Let g be the reduction from f to #HP stated in Lemma 3.2. Let $(G, s, t) = g(x)$. Let N be the number of nodes in G.

It is known that planar graphs can be embedded in the two-dimensional grid in polynomial time (for example, see [CP95]). In the case when the maximum degree is 3, the embedding can be made so that there is no edge contention. Pick one such method and apply it to G so that s is the origin. Then the resulting embedding, call it $\mathcal{E}$, is a grid graph. Let $\alpha = 8N^2$ and $\beta = N^2$. Based on $\mathcal{E}$ construct a new grid graph $\mathcal{F}$ as follows:

First, enlarge the embedding by 5α. This is done by moving each corner (a, b) of the embedding to $(5\alpha \cdot a, 5\alpha \cdot b)$. Then it is possible to allocate to each edge e of G an area of size $\alpha \times \alpha$ that is adjacent to a straight-line segment realizing e in the embedding so that no two edges share points in the allocated area.

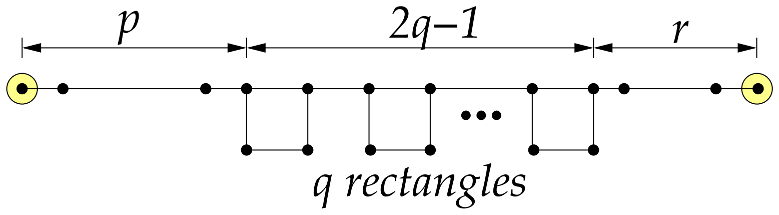


Figure 3: $S(p, q, r)$

Second, for each edge of G, attach to the path realizing it a block of β unit-size rectangles so that the rectangles do not meet with those for another edge. Such a block of rectangles is considered by $S(p, q, r)$ in Figure 3, where $p, q, r \geq 0$. In short, $S(p, q, r)$ is a path of length $p+r+2q-1$ with q rectangles attached. This second step of the conversion replaces each path π realizing an edge of G by $S(p, \beta, r)$ for some $p, r \geq 1$, such that $p+r+2\beta-1 = |\pi|$. Let L be an integer such that every path in $\mathcal{F}$ realizing an edge of G has length at most L. Then $p+r+2\beta-1 \leq L \leq N^c$ for some fixed constant $c > 0$. Call the left and the right ends of the structure u and v, respectively. For all $p, q, r \geq 1$, such that $p+2q-1+r \leq L$, the simple paths within $S(p, q, r)$ have the following properties:

1. The number of those that connect u and v is 2^q.
2. The number of those that touch u but not v is $p+1+4(2^q+\cdots+2)+2^q(r-1) = 2^q(r+7)+p-7 < L \cdot 2^q$. Similarly, the number of those that touch v but not u is less than $L \cdot 2^q$.
3. The number of those that touch neither u nor v is less than $L^2 \cdot 2^{q+1}$. This can be seen as follows:
    - The number of such paths of length zero is at most $L + 2q$.
    - The number of such paths of length one is at most $L + 3q$.
    - The number of paths within the rectangles of length two is $4q$.
    - The number of paths within the rectangles of length three is $4q$.
    - The number of paths π such that either the left end of π is a point between u and the leftmost rectangle or the right end of π is a point between v and the rightmost rectangle is less than $2(p \cdot r \cdot 2^q) < L^2 2^q$.
    - All the paths that are yet to be counted sit between the leftmost rectangle and the rightmost rectangle. For every i, $2 \leq i \leq q$, the number of such paths that touch precisely i rectangles is less than $7^2 \cdot 2^{i-2}$. The sum of the quantity for all i, $2 \leq i \leq q$, is less than $49 \cdot 2^{q-1}$.

    The grand total is less than $2L + 13q + L^2 2^q + 49 \cdot 2^{q-1}$. This is less than $L^2 \cdot 2^{q+1}$ for all but finitely many L.

In the final step of the conversion, add two new nodes s' and t' and join s and s' by $S(7, 2\beta, 1)$ and join t and t' by $S(7, 2\beta, 1)$. For each simple path π in G, let $M(\pi)$ denote the set of all simple paths $\sigma = [v_1, \ldots, v_k]$ such that eliminating from σ all the nodes not corresponding to the nodes of G produces π. Also, for each simple path π in G, let $\mu(\pi)$ denote the size of $M(\pi)$. Let $B = 2^{(N+3)\beta+6}$. The value of $\mu(\pi)$ is evaluated as follows:

(Case 1) π is a Hamilton path: $\mu(\pi) = 2^{\beta(N-1)}(8 \cdot 2^{2\beta})^2 = 2^{(N+3)\beta+6}$.

(Case 2) π is nonempty, non-Hamiltonian, and connects s and t: Here $1 \leq |\pi| \leq N-2$. So, $\mu(\pi) = 2^{\beta|\pi|}(8 \cdot 2^{2\beta})^2 \leq 2^{(N+2)\beta+6}$.

(Case 3) π touches exactly one of s and t: Here $1 \leq |\pi| \leq N-2$. There are at most two edges attached to the end node of π that is neither s nor t. By (2) in the above, there are $2L \cdot 2^{\beta}$ paths that can grow out of the end points of π along each of the two edge, so $\mu(\pi) \leq 2^{|\pi|\beta} \cdot (8 \cdot 2^{2\beta}) \cdot (2L \cdot 2^{\beta}) < 16L \cdot 2^{\beta(N+1)}$.

(Case 4) π touches neither s nor t: Here $1 \leq |\pi| \leq N-3$. There are at most two edges attached to each end node of π. So, $\mu(\pi) \leq 2^{|\pi|\beta} \cdot (2L \cdot 2^{\beta})^2 < 4L^2 \cdot 2^{\beta(N-1)}$.

(Case 5) π is empty: Since G has maximum degree 3. There are at most $3N/2$ edges in N. By (3) in the above, $\mu(\pi) < L^2 2^{\beta+1}$.

Note that the largest of the upper bounds in Cases 2 through 5 is $2^{(N+2)\beta+6}$. The number of non-Hamiltonian paths in G is bounded by $3N \cdot (2^{N-2} + 2^{N-3} + \cdots + 1) < 3N \cdot 2^{N-1}$. This is because there are N choices for one end point, there are at most three possible directions for the first edge, there are at most two possibilities there after, and the length may vary from 1 to $N-2$. Thus, the sum of $\mu(\pi)$ for all non-Hamiltonian paths is less than $3N \cdot 2^{N-1} \cdot 2^{(N+2)\beta+6} < B$. So:

Fact 1 *The number of simple paths in $\mathcal{F}$ is $\#\mathrm{HP}(G, s, t) \cdot B + R$, where $0 \leq R \leq B - 1$.*

Let $B' = 2^{(N+1)\beta+3}$. By an analysis similar to the above the following fact can be proven.

Fact 2 *The number of simple paths in $\mathcal{F}$ starting from the origin (the image of s) is $\#\mathrm{HP}(G, s, t) \cdot B' + R$, where $0 \leq R \leq B' - 1$.*

Also, let $B'' = 2^{(N-1)\beta}$. By an analysis similar to the above, the following fact holds.

Fact 3 *The number of simple paths in $\mathcal{F}$ between s' and t' is $\#\mathrm{HP}(G, s, t) \cdot B'' + R$, where $0 \leq R \leq B'' - 1$.*

Hence, the following theorem has been proven.

Theorem 4.1 *The problem of counting the number of self-avoiding walks in two-dimensional grid graphs is #P-complete under $\leq^{p}_{r\text{-}shift}$-reductions in each of the three possible definitions of the counting problem:*

- *the number of self-avoiding walks between two specified nodes needs to be counted;*
- *the number of self-avoiding walks from a specified node needs to be counted;*
- *the number of all self-avoiding walks needs to be counted.*

5 Self-Avoiding Walks in Hypercube Graphs

5.1 #P-Completeness

Let f be an arbitrary #P function. Let x be a string for which $f(x)$ needs to be evaluated. Let g be the reduction from f to the #HP as stated in Lemma 3.2. Let $(G, s, t) = g(x)$. Let $\alpha = 8N^2$ and $\beta = N^2$ as in the previous proof. Construct from G a graph H by replacing each edge by $S(1, \beta, \alpha - 2\beta)$ and attach $S(1, 2\beta, 7)$ to s and t. Call the end of the two copies of $S(1, 2\beta, 7)$ by s' and t'. Let $L = \alpha$. This graph H is still planar and has maximum degree 3. Exactly the same analysis that holds for the two-dimensional grid graph can be applied to H. Namely, Facts 1, 2, and 2 hold with H in place of $\mathcal{F}$. Furthermore, let $\Lambda = 2\alpha(N-1) + 2(8 + 2\beta)$. Then the number of paths in H having length Λ is equal to the number of Hamilton paths in G. Since the maximum degree of the graph is 3, H can be embedded as a hypercube graph $\mathcal{E}$ of dimension $\mathcal{O}(\log N)$ so that all paths realizing an edge of G have length d, where $d = 2^e$ for some integer e. Let $\mathcal{H}$ be the resulting embedding. Since $d = \mathcal{O}(\log N)$, the analysis is not affected much. The three facts hold with $\mathcal{H}$ in place of $\mathcal{F}$. Also, the following fact holds.

Fact 4 *The total number of simple paths in $\mathcal{H}$ having length Λ is $\#\mathrm{HP}(G, s, t)$.*

Thus, the following theorem has been proven.

Theorem 5.1 *The problem of counting the number of self-avoiding walks in hypercube graphs is #P-complete under $\leq^p_{r\text{-}shift}$-reductions in each of the following four cases:*

- *the number of self-avoiding walks of a specified length needs to counted;*
- *the number of self-avoiding walks between two specified nodes needs to be counted;*
- *the number of self-avoiding walks from a specified node needs to be counted;*
- *the number of all self-avoiding walks needs to be counted.*

Now it remains to show that the embedding is indeed possible.

Lemma 5.2 *Let $d \geq 1$ be an integer. There exists a polynomial-time computable embedding $\mathcal{E}$ and a pair of polynomial-time computable functions $s, \ell : \mathbf{N} \to \mathbf{N}$ such that the following properties hold:*

1. *$s(n) = \mathcal{O}(\log n)$ and $\ell(n) = \mathcal{O}(\log \log n)$.*
2. *For every $n \geq 1$, and every undirected graph $G = (V, E)$ of n nodes having maximum degree d, $\mathcal{E}(G)$ is an embedding of G in the $s(n)$-dimensional hypercube $HC^{(s(n))}$ such that*
   - *for all $u \neq v$ in G, $\mu(u) \neq \mu(v)$, and*
   - *for every edge $e = (u, v)$ in G, $\nu(e)$ has dilation $2^{\ell(n)}$ and visits no $\mu(w)$ between $\mu(u)$ and $\mu(v)$.*

   Here μ and ν are respectively the embedding of nodes and the embedding of edges specified by $\mathcal{E}(G)$.

Proof. Let $d, n, m \geq 1$ be integers. Let G be a degree d graph with n nodes and m edges. Note that $m < n^2/2$. Identify the nodes in G with the set of integers from 0 to $n-1$ and identify the edges in G with the set of integers from 0 to $m-1$. For each node u of G, fix an enumeration of its neighbors. For every edge (u, v) of G, let $I_u(v)$ denote the order of v in the enumeration of the neighbors of u. Let q be the smallest integer such that $3q+4$ is a power of 2 and such that $q \geq \lceil \log n \rceil$. Let $s = 6q + d + 2$. Let H denote the s-dimensional hypercube. Each node of G will be viewed as a q-bit binary number and each edge of G as a $2q$-bit binary number. For a binary string u, let $\overline{u}$ denote the string constructed from u by flipping all of its bits.

The s-bit representation of a node in H is divided into the following five components.

- The Node Part: This part has length $2q$. Here for each node $u = u_1 \cdots u_q$ of G, $\overline{u}u$ encodes u. Note that this encoding has exactly q many 1s in it.
- The Edge Part: This part has length $4q$ and is used to encode the edge to be traversed. Here each edge $e = e_1 \cdots e_{2q}$ is encoded as $\overline{e}e$.
- The Neighbor Part: This part has length d and is used to encode the value of $I_u(v)$ or the value of $I_v(u)$ when the edge (u, v) is traversed.
- The Switch Part: This part has length 2.

Let u, v, w, y be binary strings of length $2q$, $4q$, d, and 2, respectively. Then, $\langle u, v, w, y \rangle$ denotes the node $uvwy$ in $HC^{(s)}$.

The embedding $\mathcal{E}(G) = (\mu, \nu)$ is defined as follows: For each node $u = u_1 \cdots u_q$,

$$\mu(u) = \langle \overline{u}u, 0^{2q}, 0^d, 000 \rangle.$$

As for ν, let $e = (u, v)$ be an edge in G such that $u < v$. Let $A = a_1 \cdots a_{2q} = \overline{u}u$, $B = b_1 \cdots b_{2q} = \overline{v}v$, $C = c_1 \cdots c_{4q} = \overline{e}e$, $W_1 = 0^{I_u(v)-1}10^{d-I_u(v)}$, and $W_2 = 0^{I_v(u)-1}10^{d-I_v(u)}$. Let $i_1, \ldots, i_q$ be the enumeration of all the positions at which A has a bit 1 in increasing order. Let $j_1, \ldots, j_q$ be the enumeration of all the positions at which B has a bit 1 in increasing order. Let $k_1, \ldots, k_{2q}$ be the enumeration of all the positions at which C has a bit 1 in increasing order. For each t, $1 \leq t \leq q$, let $A_t = 0^{i_t} a_{i_t+1} \cdots a_{2q}$ and $B_t = 0^{j_t} b_{j_t+1} \cdots b_{2q}$. Also, for each t, $1 \leq t \leq 2q$, let $C_t = c_1 \cdots c_{k_t} 0^{4q-k_t}$. Note that $A_q = B_q = 0^q$ and $C_{2q} = C$. The edge e is represented by a path from $\langle A, 0^{4q}, 0^d, 00 \rangle$ to $\langle 0^q, C, 0^d, 11 \rangle$ and a path $\langle B, 0^{4q}, 0^d, 00 \rangle$ to $\langle 0^q, C, 0^d, 11 \rangle$, each of length $3q+4$. The first path is defined by:

$$\langle A, 0^{4q}, 0^d, 00 \rangle, \langle A, 0^{4q}, W_1, 00 \rangle, \langle A, 0^{4q}, W_1, 10 \rangle, \langle A, C_1, W_1, 10 \rangle, \cdots, \langle A, C_{2q}, W_1, 10 \rangle,$$
$$\langle A_1, C, W_1, 10 \rangle, \cdots, \langle A_q, C, W_1, 10 \rangle, \langle 0^q, C, 0^d, 10 \rangle, \langle 0^q, C, 0^d, 11 \rangle.$$

The second path is defined with B in place of A. The total length of the join of the two paths is $6q + 8$ and this is a power of 2 by assumption. Note that every node in the entire path contains one of the following: (i) C in the edge part, (ii) A in the node part and W_1 in the neighbor part, or (iii) B in the node part and W_2 in the neighbor part. So, no two edges share internal nodes in their path representation. It is not hard to see that the embedding can be computed in logarithmic space. This proves the lemma. □

5.2 Scaling up to exponential time

Let f be a function belonging to #EXP. Then there is a one-tape nondeterministic exponential-time machine M such that $f = \#acc_M$. It can be assumed that there is a polynomial p such that for all strings x, M on input x halts at step $2^{p(|x|)}$. By applying the tableau method to the computation of M and then by reducing the number of occurrences of each variable by introducing its copies, it can be shown that the computation of f can be transformed to #SAT by a polynomial-time local computation.

Lemma 5.3 *There is a reduction R from the evaluation problem of f to* #SAT *that satisfies the following conditions.*

1. *For every string x, $R(x)$ is a 3CNF formula.*
2. *For every string x, $f(x) = \#\mathrm{SAT}(R(x))$.*
3. *For every string x, $R(x)$ can be locally computed in polynomial time in the following sense:*
   (a) *For each variable y of $R(x)$, y or $\overline{y}$ appears in exactly three clauses.*
   (b) *For each string x, let $\nu(x)$ denote the number of variables in $R(x)$. Then ν is polynomial-time computable.*
   (c) *For each string x, let $\mu(x)$ denote the number of variables in $R(x)$. Then ν is polynomial-time computable.*
   (d) *For each string x and each i, $1 \leq i \leq \mu(x)$, let $C(x,i)$ be the ith clause in $R(x)$. Then C is polynomial-time computable.*
   (e) *For each string x and each i, $1 \leq i \leq \nu(x)$, let $S(x,i)$ be the set of all indices j such that the ith variable appears in $C(x,j)$. Then S is polynomial-time computable.*

Now apply conversion of Lemma 3.1 to each formula generated by R. Denote the resulting transformation from the set of strings to NAE3SAT by R'. Since R is locally polynomial-time computable, so is R'. Modify the construction for Lemma 3.2 so that crossing of XOR-gadgets is allowed. Then the maximum degree remains three and the scaling factor becomes $2^{12n+352m}$. The connectivity of the gadgets in the graph essentially depends only on the occurrences of corresponding variables, so the mapping can be computed locally in polynomial time. Now apply the embedding described above. Denote the resulting transformation from the set of strings to the set of hypercube graphs by H. The length of the path is proportional to the logarithm of the number of nodes, so they are polynomially bounded. For each string x, let $d(x)$ be the dimension of $H(x)$, $N(x)$ denote the set of all strings of length $d(x)$ that encode a node in $H(x)$, and $E(x)$ denote the set of all strings ww' having length $2d(x)$ such that (w,w') is an edge of $H(x)$. Note that, due to the construction of the paths, if w and w' are both members of $N(x)$ and ww' is a nonmember of $E(x)$, then w and w' are not neighbors. Thus, the graph $H(x)$ can be specified only by the set $N(x)$: For all w and w' in $E(x)$, there is an edge between w and w' if and only if the Hamming distance between w and w' is one. This observation yields the following fact.

Fact 5 *The function d is polynomial-time computable. There exists a polynomial-time computable function C such that $C(x)$ is a polynomial-size boolean circuit that decides $N(x)$.*

This gives the following theorem.

Theorem 5.4 *The problem of counting the number of simple paths in hypercube graphs is* #EXP-*complete under $\leq^{p}_{r\text{-}shift}$-reductions if the graphs are specified by circuits.*

Acknowledgement

The authors would like to thank Joerg Rothe for careful reading of the manuscript.

References

[CG96] A. R. Conway and A. J. Guttmann. Square lattice self-avoiding walks and corrections-to-scaling. *Physical Review Letters*, 77:5284–5287, 1996.

[CP95] M. Chrobak and T. H. Payne. A linear time algorithm for drawing a planar graph on a grid. *Information Processing Letters*, pages 241–246, 1995.

[GJT76] M. Garey, D. Johnson, and E. Tarjan. The planar Hamiltonian circuit problem is NP-complete. *SIAM Journal on Computing*, 5(4):704–714, 1976.

[MS93] N. Madras and G. Slade. *The Self-Avoiding Walk*. Birkháuser, Boston, MA, 1993.

[Pap94] C. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[Sch78] T. Schaefer. The complexity of satisfiability problem. In *Proceedings of 10th Symposium on Theory of Computing*, pages 216–226. ACM Press, 1978.

[Val79a] L. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8:189–201, 1979.

[Val79b] L. Valiant. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8(3):410–421, 1979.

[Wel93] D. Welsh. *Complexity: Knots, Colourings and Counting*. Cambridge University Press, 1993.