Available at
www.ComputerScienceWeb.com
POWERED BY SCIENCE DIRECT®

Theoretical Computer Science 304 (2003) 129–156

Theoretical
Computer Science

www.elsevier.com/locate/tcs

The complexity of counting self-avoiding walks in subgraphs of two-dimensional grids and hypercubes

Maciej Liśkiewicz[a,1], Mitsunori Ogihara[b,*,2], Seinosuke Toda[c]

[a] *Universität zu Lübeck, Institut für Theoretische Informatik, Wallstraße 40, D-23560, Lübeck, Germany*

[b] *Department of Computer Science, University of Rochester, P.O. Box 270226, Rochester, NY 14627-0226, USA*

[c] *Department of Applied Mathematics, Nihon University, 3-25-40 Sakurajyou-shi, Setagaya-ku, Tokyo 156, Japan*

Received 12 October 2001; received in revised form 14 November 2002; accepted 25 November 2002
Communicated by O. Watanabe

Abstract

Valiant (SIAM J. Comput. 8 (1979) 410–421) showed that the problem of computing the number of simple *s–t* paths in graphs is #P-complete both in the case of directed graphs and in the case of undirected graphs. Welsh (Complexity: Knots, Colourings and Counting, Cambridge University Press, Cambridge, 1993, p. 17) asked whether the problem of computing the number of self-avoiding walks of a given length in the complete two-dimensional grid is complete for $\#P_1$, the tally-version of #P. This paper offers a partial answer to the question of Welsh: it is #P-complete to compute the number of self-avoiding walks of a given length in a subgraph of a two-dimensional grid. Several variations of the problem are also studied and shown to be #P-complete. This paper also studies the problem of computing the number of self-avoiding walks in a subgraph of a hypercube. Similar completeness results are shown for the problem. By scaling the computation time to exponential, it is shown that computing the

* Corresponding author.
E-mail addresses: liskiewi@tcs.mu-luebeck.de (M. Liśkiewicz), ogihara@cs.rochester.edu (M. Ogihara), toda@am.chs.nihon-u.ac.jp (S. Toda).

[1] On leave from Instytut Informatyki, Uniwersytet Wrocławski, Poland.

[2] Supported in part by NSF grants EIA-0080124, EIA-0205061, and DUE-9980943 and by NIH grants RO1-AG18231 and P30-AG18254.

doi:10.1016/S0304-3975(03)00080-X

number of self-avoiding walks in hypercubes is a complete problem for #EXP in the case when a subgraph of a hypercube is specified by its dimension and a boolean circuit that accepts the nodes.

Finally, this paper studies the complexity of testing whether a given word over the four-letter alphabet $\{U, D, L, R\}$ represents a self-avoiding walk in a two-dimensional grid. A linear-space lower bound is shown for nondeterministic Turing machines with a 1-way input head to make this test.

1. Introduction

A self-avoiding walk (SAW) is a path on a graph that does not visit any node more than once. For each $n \geqslant 0$, let c_n denote the number of self-avoiding walks on the two-dimensional grid that have length n and start from the origin. By "two-dimensional grid" we mean the two-dimensional rectangular lattice Z^2 with origin (0,0). The question of whether there exists a formula for c_n has been extensively studied (see [14,27] for a survey). Although the exact formulas have been found for some special cases of the problem (see, e.g. [28]), no formulas are known for the general case. One of the key results for the general case is by Hammersley and Morton [8], which states that there exists some $\mu > 0$ such that $\lim_{n\to\infty} c_n^{1/n} = \mu$ and that, for all $n \geqslant 0$, $c_n/\mu^n \geqslant 1$. Also, Hammersley and Welsh [9] show that there exists some constant a such that $c_n/\mu^n = \mathrm{O}(a^{\sqrt{n}})$. However, the exact value of μ is yet unknown, and much work has been done towards obtaining lower and upper bounds of μ. The current best bounds are $2.62002 \leqslant \mu$ and $\mu \leqslant 2.67919$, which are respectively shown in [5,18]. A conjecture in statistical physics states that for some constants A and γ it holds that

$$c_n = A\mu^n n^{\gamma-1}(1 + \mathrm{o}(1))$$

and similar conjectures have been made for grids of higher dimension (see, e.g. [20]). Assuming that the conjecture for the two-dimensional grid is true, experiments seem to suggest that $\gamma = \frac{43}{32}$ (see [14] for more discussions). The exact value of c_n has been calculated for all n up to 51 [4]. Also, Randall and Sinclair [20] present Monte Carlo algorithms for approximating the value of c_n, and for generating SAWs of a given length almost uniformly at random.

Valiant [26] is the first to find connections between self-avoiding walks and computational complexity theory. He showed that the problem of computing the number of all simple s–t paths is #P-complete under polynomial parsimonious reductions (polynomial-time function reductions without post-computation) both in the case of directed graphs and in the case of undirected graphs. Welsh [27] asked whether the exact counting problem for the two-dimensional grid is complete for $\#\mathrm{P}_1$, i.e. the "tally" version of

#P, if the length of walks is specified by a tally string.[3] If the answer to this question is affirmative then it could mean that determining the strict value for c_n may be computationally intractable and hence that no exact formula for c_n exists (see [7] for hardness of $\#P_1$).

While it is straightforward to prove that the exact counting of SAWs for the two-dimensional grid belongs to $\#P_1$, settling the question of whether the problem is $\#P_1$-hard appears to be difficult. The difficulty seems to lie in the fact that the two-dimensional grid has a very rigid, regular structure. An approach to proving the hardness would be to embed the computation of a nondeterministic Turing machine in the grid with no holes, thereby creating a correspondence between the IDs of a Turing machine and the grid points. However, every line segment of the grid may be traversed in either direction and the graph that represents transitions between machine configurations is multidimensional. Thus, finding such an embedding seems to be hard.

This leads us to the question of whether the counting problem is hard for some complexity class if it is permitted to create holes, i.e., if the graphs in which SAWs are counted are those composed of the nodes and the edges of the two-dimensional grid. Since the number of choices for locations of the holes in a finite two-dimensional grid is exponential in the number of nodes in the grid, here we should be rather thinking about #P-hardness than $\#P_1$-hardness. Thus, we ask:

Is counting SAWs of a specified length in subgraphs of two-dimensional grids #P-complete under some polynomial-time function reductions?

Our question is very different from the question of Welsh. An answer to our question does not necessarily shed light on the original problem. Also, permitting the existence of holes has made the question easier to deal with.

In this paper, we provide a positive answer to the modified version of the question of Welsh. The problem of counting SAWs in subgraphs of two-dimensional grids is #P-complete under polynomial-time function reductions where the post-computation that is performed to recover the value of a #P-function from the count is simply right-bit-shifting the count in its binary representation, i.e., truncating the binary representation at the right end.

The proof uses as a basis a construction given by Garey et al. [6] for showing NP-completeness of the problem of testing whether a planar 3-regular graph has a Hamiltonian cycle, the problem we will denote by HamCycle-Plan3. It has been

[3] In [27, Problem 1.7.3], Welsh actually asks about #P-completeness, not about $\#P_1$-completeness. However, we believe that $\#P_1$-completeness should have been addressed. Strong evidence exists that the problem is unlikely to be #P-complete: the problem is not #P-complete unless the polynomial hierarchy collapses to the second level. Suppose that the counting problem is #P-complete. Then, by Toda's Theorem [23], the problem is hard for the polynomial hierarchy. Since the counting problem takes unary inputs, it trivially belongs to FP/poly, the class of functions computable by polynomial-size circuits. This implies that the polynomial hierarchy is included in P/poly, in particular, $NP \subseteq P/poly$. Then, by the Karp–Lipton Theorem [12], we have that $PH = \Sigma_2^p$. By noting that PP is reducible to #P, we can derive much stronger evidence: the problem is not #P-complete unless the counting polynomial hierarchy collapses to Σ_2^p. Both Toda's Theorem and the Karp–Lipton Theorem were well known at the time the book was written. Welsh even mentions Toda's Theorem [23] only a few paragraphs after Problem 1.7.3.

observed that, for many NP-complete problems Π, existing NP-completeness proofs of Π actually prove #P-completeness of the counting version of Π in the following sense. The standard method for proving NP-completeness is to construct a polynomial-time many-one reduction from a known NP-complete problem. That is, to prove that an NP problem Π is NP-complete, we find a known NP-complete problem Π' and a polynomial-time many-one reduction R from Π' to Π. In many existing NP-completeness proofs of that kind, the many-one reduction R possesses a special property that for all x the number of witnesses to the membership of x in Π' can be easily calculated from the number of witnesses to the membership of $R(x)$ in Π. Garey et al. prove NP-completeness of HamCycle-Plan3 by constructing a many-one reduction from 3SAT to HamCycle-Plan3. One might hope that this reduction exhibits such a nice property. Provan [19] claims that this is indeed the case by stating that, for each 3CNF formula φ, and for each satisfying assignment α of φ, the graph produced by the Garey–Johnson–Tarjan reduction on input φ has exactly

$$(8^7 \cdot 18)^m \cdot 8^{6a} \cdot 8^b \cdot 36.$$

Hamiltonian cycles corresponding to α, where m is the number of clauses of φ and a and b are quantities not depending on α. Provan uses this analysis to argue #P-completeness of #HamCycle-Plan3, the counting version of HamCycle-Plan3 problem, and this observation is referenced by others (e.g. [10,24]). However, we find that this analysis is erroneous. There are two factors missing in the analysis. The correct number of Hamiltonian cycles that correspond to α is

$$2^{m_2} \cdot 3^{m_3} \cdot (8^7 \cdot 18)^m \cdot 8^{6a} \cdot 8^b \cdot 36,$$

where a and b have the same meaning as before and m_2 (respectively, m_3) is the number of clauses C in φ such that α satisfies exactly two (respectively, three) literals of C. In this paper we show that the Garey–Johnson–Tarjan reduction can be modified (by considering only a certain type of formulas and by replacing one of the gadgets) to show a correct proof of #P-completeness of #HamCycle-Plan3, where the reduction required is the aforementioned "right-bit-shifting."

To prove #P-completeness of our SAW-counting problem, we transform #HamCycle-Plan3 to the problem of counting SAWs in subgraphs of two-dimensional grids. In this transformation an edge of the planar graph is cut in the middle and then the edges are mapped on a two-dimensional grid so that they are stretched to paths of the same length. Then the Hamiltonian cycles in the input graph can be counted by computing the number of longest paths in the grid. This establishes that the problem of counting SAWs of a given length in subgraphs of two-dimensional grids is #P-complete regardless of whether the end points of SAWs are specified.

This observation immediately raises the question of whether such flexibility exists if we need to compute the number of SAWs of *any* length. We show that this is indeed the case. That is, the problem of counting SAWs in subgraphs of two-dimensional grids is #P-complete regardless of whether the length of the SAWs is specified or not and regardless of whether the end points of the SAWs are specified.

This completeness result has an added bonus. Recall that, for each $n \geqslant 2$, an n-dimensional hypercube is the graph in which the vertices are the binary strings having

length n and in which two vertices are adjacent if and only if they differ at exactly one bit position. A polynomial-time "right-bit-shift" reduction exists from each #P function to the problem of counting SAWs in a subgraph of a hypercube, where the dimension of the hypercubes increases only logarithmically. Then we ask how complex the problem is if the dimension is allowed to grow polynomially. Since there are 2^n nodes in the n-dimensional hypercube, we assume that subgraphs of hypercubes are specified by boolean circuits (each point in a hypercube can be naturally viewed as a binary string). Furthermore, we stipulate that the subgraphs have a special property that, for each pair of neighboring nodes in the hypercube, if both of them belong to the subgraph then the edge joining them also belongs to the subgraph. Then we show that the problem of counting SAWs with this kind of specification is complete for the exponential-time version of #P.

Finally, we study computational complexity of the problem of testing whether a given string over a four-letter alphabet (consisting of U, D, L, and R, which correspond to the four directions that a path can follow on the two-dimensional grid) encodes a self-avoiding walk in the two-dimensional grid. Such study is motivated by the desire of clarifying the space complexity of the problem of counting SAWs. Here we study space complexity of the decision problem, instead of the counting problem. We show that to detect SAWs on-line every nondeterministic Turing machine needs at least linear space.

We note here that, using NP-completeness as a yardstick for measuring computational complexity of problems, Istrail [11] has recently provided negative answers to conjectures about computability of some thermodynamical quantities in two- and three-dimensional square lattices. More precisely, in [3], Barahona shows that, for the much-studied Ising model on the three-dimensional cubic lattice, the problem of computing the ground states on finite sublattices is intractable (i.e. NP-hard). Istrail extends this result and proves that the problem remains NP-hard for every non-planar lattice, including the two-dimensional non-planar lattices.

This paper is organized as follows: In Section 2, we set down notation and notions that are used throughout the paper. Our reduction from SAT to the Hamiltonian cycle problem in 3-regular planar graphs will be presented in Section 3. In Section 4, we study the complexity of computing the number of self-avoiding walks in subgraphs of two-dimensional grids and in Section 5, we consider the complexity of computing the number of self-avoiding walks in subgraphs of hypercubes. Finally, in the last section, we prove our complexity lower-bound results.

2. Preliminaries

2.1. Counting complexity classes

Let M be a nondeterministic Turing machine. By $\#acc_M$ we denote the function that maps each string x to the number of accepting computation paths of M on input x. We are interested in two classes of functions, a class #P of Valiant [25], defined as $\{\#acc_M | M$ is a polynomial-time nondeterministic Turing machine$\}$, and a class #EXP

of Papadimitriou and Yannakakis [17], defined as $\{\#acc_M | M$ is an exponential-time nondeterministic Turing machine$\}$. For a language class C, Valiant [25] proposes to define $\#C$ as the class of all functions f such that there exist some polynomial-time nondeterministic oracle Turing machine M and some language $A \in C$ such that $f = \#acc_{M^A}$. The class #EXP by Papadimitriou and Yannakakis, which we study in this paper, is strictly larger than Valiant's class #EXP. This is because each function in the Valiant version of #EXP has polynomial output length while functions in the Papadimitriou–Yannakakis version may have exponential output length.

2.2. Reductions between counting functions

Next we define polynomial-time reductions between counting functions. Let f and g be functions from Σ^* to $\mathbf{N}$. We say that f is *polynomial-time one-Turing reducible* to g, denoted by $f \leqslant^p_{1-T} g$, 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, $f(x) = R_2(x, g(R_1(x)))$. We consider two special cases of polynomial-time one-Turing reductions. We say that f is *polynomial-time parsimoniously reducible* to g, denoted by $f \leqslant^p_{\text{parsimonious}} g$, if for all x and y the above R_2 satisfies $R_2(x, y) = y$, i.e., for all x, $f(x) = g(R_1(x))$. We say that the function f is *polynomial-time right-bit-shift reducible* to g, denoted by $f \leqslant^p_{r\text{-shift}} g$, if there is a polynomial-time computable function $R_3 : \Sigma^* \to \mathbf{N} - \{0\}$ such that for all x and y it holds that $R_2(x, y) = y \operatorname{div} 2^{R_3(x)}$, i.e., for all x, $f(x) = g(R_1(x)) \operatorname{div} 2^{R_3(x)}$, where div is integer division. It is easy to see that the following proposition holds.

Proposition 1. *Both* $\leqslant^p_{r\text{-shift}}$*-reductions and* $\leqslant^p_{\text{parsimonious}}$*-reductions are transitive.*

2.3. Counting problems

Here we define the problems that we are concerned with. A 3CNF formula (respectively, a $\hat{3}$CNF formula) is a boolean formula in the conjunctive normal form such that each clause has exactly three (respectively, at most three) literals. 3SAT (respectively, $\hat{3}$SAT) is the problem of testing satisfiability of 3CNF (respectively, $\hat{3}$CNF) formulas. #SAT (respectively, #3SAT and #$\hat{3}$SAT) is the problem of computing the number of satisfying assignments of boolean formulas (respectively, 3CNF formulas and $\hat{3}$CNF formulas). By the standard reduction from nondeterministic Turing machine computations to 3CNF formulas (see, for example [16]), these three problems are each complete for #P under polynomial-time parsimonious reductions.

Proposition 2 (Valiant [26]). #SAT, #3SAT, *and* #$\hat{3}$SAT *are each complete for* #P *under polynomial-time parsimonious reductions.*

For a CNF formula φ and its satisfying assignment α, we say that α is a *not-all-equal satisfying assignment of* φ if for every clause C of φ with more than one literal it holds that α falsifies at least one literal of C. Not-All-Equal-SAT (NAE3SAT for short) [21] is a problem of testing whether a given 3CNF formula can be satisfied by a not-all-equal satisfying assignment. Schaefer [21] is the first to study this variation

of the satisfiability problem. He also proves that the problem is NP-complete. Schaefer restricted the input formulas to be 3CNF formulas, but we allow them to be $\hat{3}$CNF formulas.

The following lemma, which will play a crucial role later, states that there is a polynomial-time parsimonious reduction from #SAT to #NAE3SAT such that each formula belonging to the range of the reduction is composed of three-literal clauses and a single one-literal clause and is satisfiable only by not-all-equal satisfying assignments.

Lemma 3. *There is a polynomial time computable function f that maps each* 3*CNF formula to a* $\hat{3}$*CNF formula such that, for all integers* $n, m \geqslant 1$ *and for all formulas* φ *of* n *variables and* m *clauses,* $\psi = f(\varphi)$ *has the following properties*:

(1) *The number of variables of* ψ *is* $n+m+1$ *and the number of clauses of* ψ *is* $8m+1$. *Of the* $8m+1$ *clauses only one is a single-literal clause and the rest are three-literal clauses.*

(2) *Every satisfying assignment of* ψ *is a not-all-equal satisfying assignment. More precisely, for every satisfying assignment* α *of* ψ, α *satisfies exactly two literals for exactly* $4m$ *three-literal clauses and satisfies exactly one literal for exactly* $4m$ *three-literal clauses.*

(3) $\#\mathrm{SAT}(\varphi) = \#\mathrm{SAT}(\psi)$.

Proof. Let n and m be positive integers. Let φ be a 3CNF formula having n variables and m clauses. Let $C_1, \ldots, C_m$ be an enumeration of the clauses of φ. We construct ψ, which will be the value of our reduction on input φ, as follows: First, we introduce one new variable w and a single-literal clause $(\bar{w})$. Due to this clause, every satisfying assignment of ψ sets the value of w to false. Next, for each i, $1 \leqslant i \leqslant m$, we construct from C_i an eight-clause formula C_i' as follows: Let x, y, and z be the three literals of C_i. We introduce a new variable u_i and some clauses so that every satisfying assignment of ψ sets the value of u_i to the value of $x \vee y$. This requirement can be fulfilled by introducing three clauses:

$$(\bar{u_i} \vee x \vee y), (u_i \vee \bar{x}), (u_i \vee \bar{y}).$$

This turns C_i to

$$(\bar{u_i} \vee x \vee y) \wedge (u_i \vee \bar{x}) \wedge (u_i \vee \bar{y}) \wedge (u_i \vee z).$$

We insert the literal w to each of the last three clauses to make them three-literal clauses. Thus, we obtain

$$(\bar{u_i} \vee x \vee y) \wedge (u_i \vee \bar{x} \vee w) \wedge (u_i \vee \bar{y} \vee w) \wedge (u_i \vee z \vee w).$$

Then every satisfying assignment of this four-clause formula that also satisfies the single-literal clause $(\bar{w})$ is a not-all-equal satisfying assignment. With this current form, how many out of the four three-literal clauses in the above have exactly two satisfied literals by such a not-all-equal satisfying assignment is dependent on the satisfying assignment itself. So, we add the literal-wise complement of each of the four

clauses:

$$(u_i \vee \bar{x} \vee \bar{y}), (\bar{u_i} \vee x \vee \bar{w}), (\bar{u_i} \vee y \vee \bar{w}), (\bar{u_i} \vee \bar{z} \vee \bar{w}).$$

Note that, for all three-literal clauses D, for all assignments α, and for all integers k, $0 \leqslant k \leqslant 2$, it holds that α satisfies exactly k literals of D if and only if α satisfies exactly $3-k$ literals of the literal-wise complement of D. We denote the resulting eight-clause formula by C_i', i.e.,

$$C_i' = (\bar{u}_i \vee x \vee y) \wedge (u_i \vee \bar{x} \vee w) \wedge (u_i \vee \bar{y} \vee w) \wedge (u_i \vee z \vee w) \\ \wedge (u_i \vee \bar{x} \vee \bar{y}) \wedge (\bar{u}_i \vee x \vee \bar{w}) \wedge (\bar{u}_i \vee y \vee \bar{w}) \wedge (\bar{u}_i \vee \bar{z} \vee \bar{w}).$$

Now every satisfying assignment α of C_i' that satisfies $(\bar{w})$ satisfies exactly two literals for four three-literal clauses and exactly one literal for four three-literal clauses.

The formula ψ is defined as

$$C_1' \wedge \cdots \wedge C_m' \wedge (\bar{w}).$$

Then ψ has $n+m+1$ variables, consists of $8m$ three-literal clauses and one single-literal clause, and has as many satisfying assignments as φ. Furthermore, for every satisfying assignment α of ψ, there are precisely $4m$ three-literal clauses such that α satisfies exactly one literal and precisely $4m$ three-literal clauses such that α satisfies exactly two literals. Clearly, the number of satisfying assignments of ψ is equal to that of φ. This proves the lemma. □

Hamiltonian Path is the problem of deciding, given a graph G and a node pair (s,t), whether there is a Hamiltonian path from s to t in G. We denote by #HamPath the problem of computing, for a given graph G and a node pair (s,t), the number of Hamiltonian paths from s to t in G. On the other hand, Hamiltonian Cycle is the problem of deciding, given a graph G, whether there is a Hamiltonian cycle in G. We define #HamCycle to be the problem of computing, given a graph G, the number of Hamiltonian cycles in G. By #HamPath-Plan$\hat{3}$ we denote the problem of counting Hamiltonian Paths where the inputs are restricted to the planar graphs having maximum-degree three and by #HamCycle-Plan3 we denote the problem of counting Hamiltonian Cycles where the inputs are restricted to the planar 3-regular graphs.

3. Polynomial time reducibility of #3SAT to #HamCycle-Plan3 and to #HamPath-Plan$\hat{3}$

To prove #P-completeness of the problem of counting SAWs in subgraphs of two-dimensional grids and the problem of counting SAWs in subgraphs of hypercubes, we use a polynomial-time right-bit-shift reduction from #3SAT to #HamCycle-Plan3. With a slight modification this reduction can be turned into a polynomial-time right-bit-shift reduction from #3SAT to #HamPath-Plan$\hat{3}$.

The reduction is a variation of the polynomial-time many-one reduction due to Garey et al. [6] from 3SAT to HamCycle-Plan3, i.e. the Hamiltonian Cycle decision problem

of planar 3-regular graphs. The reduction has a special property: for each graph G produced by the reduction, there is at least one edge traversed by all the Hamiltonian cycles of G and such an edge is easy to identify. So, by simply removing one of such common edges, the Garey–Johnson–Tarjan reduction becomes a polynomial-time many-one reduction from 3SAT to the Hamiltonian Path decision problem of planar graphs having maximum degree three.

One cannot directly use the Garey–Johnson–Tarjan reduction to prove #P-completeness of #HamCycle-Plan3 or #HamPath-Plan$\hat{3}$, because the number of Hamiltonian paths representing a satisfying assignment depends on how the assignment satisfies each clause. More precisely, let φ be a satisfiable 3CNF formula and G be the planar 3-regular graph that the Garey–Johnson–Tarjan reduction produces on input φ. Let A be the set of all satisfying assignments of φ and let P be the set of all Hamiltonian cycles in G. The Garey–Johnson–Tarjan reduction defines an onto mapping from P to A such that from each path in P the element in A that corresponds to the path with respect to the mapping can be computed in polynomial time. For each $\alpha \in A$, let P_α denote the set of all Hamiltonian cycles in P that map to α by this onto mapping. Clearly, $|P| = \sum_{\alpha \in A} |P_\alpha|$. It holds that

$$\begin{aligned} |P_\alpha| &= (8^7 \cdot 18)^{m_1} \cdot (2 \cdot 8^7 \cdot 18)^{m_2} \cdot (3 \cdot 8^7 \cdot 18)^{m_3} \cdot 8^{6a} \cdot 8^b \cdot 36 \\ &= 2^{m_2} \cdot 3^{m_3} \cdot (8^7 \cdot 18)^m \cdot 8^{6a} \cdot 8^b \cdot 36, \end{aligned}$$

where a is the number of "crossing exclusive-or" graphs, b is the number of "non-crossing exclusive-or" graphs added in the construction G, and for each i, $1 \leqslant i \leqslant 3$, m_i is the number of clauses C of φ such that α satisfies exactly i literals of C. By considering only instances of NAE3SAT and applying slight modifications to the Garey–Johnson–Tarjan reduction, we can adjust the multitude $|P_a|$ to 2^r, where r is a quantity depending only on the input.

Let φ be a 3CNF formula such that #SAT(φ) needs to be evaluated. Let n be the number of variables of φ and m be the number of clauses of φ. Let ψ be the formula generated from φ by applying the transformation in Lemma 3. The formula ψ is defined on $n + m + 1$ variables and has $8m$ three-literal clauses and one single-literal clause. We construct a graph G from ψ by applying the Garey–Johnson–Tarjan reduction with slight modifications.

Basic components of the construction are the Tutte-gadget (see Fig. 1), the XOR-gadget (see Fig. 2), and the OR-gadget (see Fig. 3). Here the first two gadgets are taken without changes from the Garey–Johnson–Tarjan reduction while the OR-gadget is our own device.

The Tutte-gadget is used to force 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 Fig. 1b).

The XOR-gadget is a ladder built using eight copies of the Tutte-gadget (see Fig. 2a). To go through all the nodes in an XOR-gadget one has to enter and exit on the same vertical axis. Moreover for each of the two vertical axes there are $(4 \cdot 2)^4 = 2^{12}$ Hamiltonian paths that traverse the gadget. XOR-gadgets can be crossed without losing

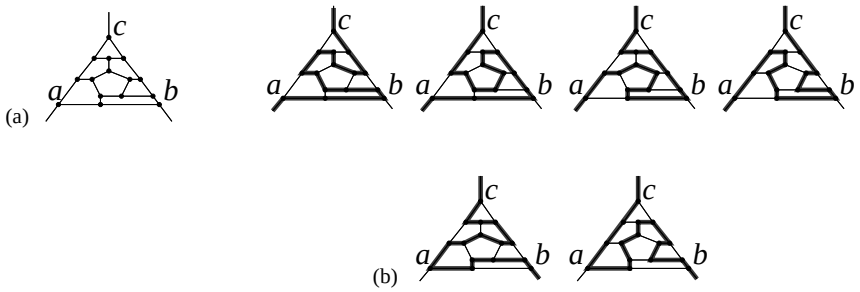

Fig. 1. The Tutte-gadget. (a) The gadget. (b) Hamiltonian traversals of the nodes in the Tutte-gadget. The top four are traversals connecting nodes a and c with b in the middle. The bottom two are traversals connecting nodes b and c with a in the middle.

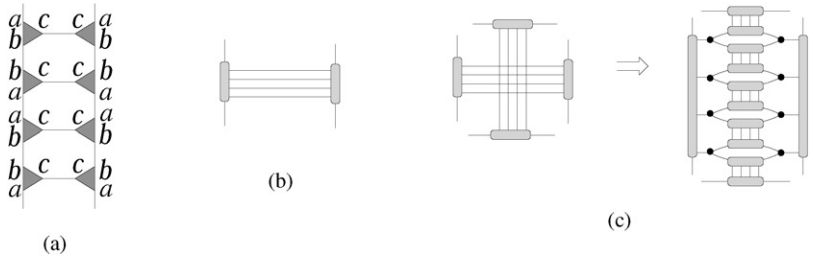

Fig. 2. The XOR-gadget. (a) The structure of the gadget: each of the eight shaded triangles represents a copy of the Tutte-gadget placed in the structure as oriented above. (b) A symbol to denote an XOR-gadget. (c) Crossing of two XOR-gadgets. On the left the four horizontal lines in one XOR-gadget and the four vertical lines in the other XOR-gadget need to be crossed. On the right the crossing of the lines are resolved by introduction of four additional XOR-gadgets.

planarity by inflating the number of Hamiltonian paths (see Fig. 2). Since four XOR-gadgets are added, the number of Hamiltonian paths is increased by a multiplicative factor of 2^{48}.

In an OR-gadget, each four-node rectangle on the right-hand side is considered to be an input. The line at the right end of an input part provides connection to the outside world. The one shown in Fig. 3 is a three-input OR-gadget. Each input line will be either entirely available or entirely unavailable. If an input line is available we think of the situation as the input being assigned true as a value. Otherwise, we think of the situation as the input being assigned false. In the former case, there are two ways to traverse the line (see Fig. 3b). In the latter there is only one way to touch the two nodes on the line (see Fig. 3c). If the number of inputs that are assigned true is one, there is only one way to traverse the nodes on the input lines (see Fig. 3d). If the number is two, there are two possibilities (see Fig. 3e). Finally, there are two ways to traverse all the four nodes on the left-hand side of an OR-gadget (see Fig. 3f).

Another important components of the graph G are two-node cycles (see Fig. 4). A two-node cycle is represented as a pair of nodes that are vertically lined up and are

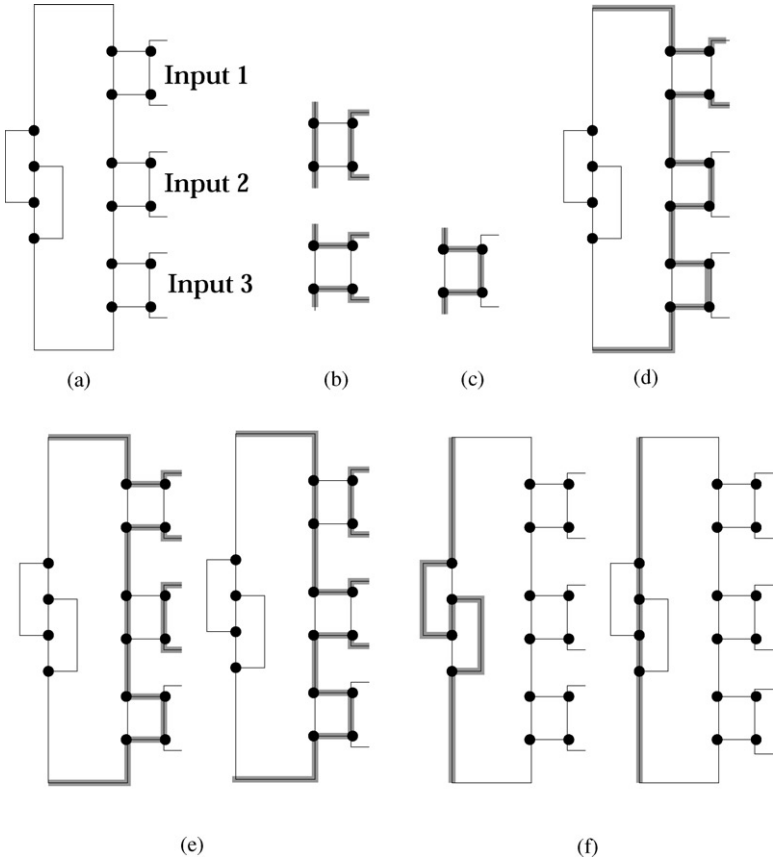

Fig. 3. The OR-gadget. (a) A three-input OR-gadget. (b) Two ways to traverse an input line whose value is true. (c) The way to traverse an input line whose value is false. (d) The traversal of input lines when only input 1 is true. (e) Two possible traversals of input lines when exactly two inputs are true. (f) Two ways to create a Hamiltonian traversal of the four nodes on the left side of the gadget.

connected by two edges. We refer to the two nodes by the *top node* and the *bottom node* and refer to the two arcs by the *right edge* and the *left edge*. The paths consisting solely of one of the two arcs are the Hamiltonian paths of a two-node cycle.

The graph G is essentially two vertical sequences of two-node cycles that are side-by-side. The right sequence is called Π and the left sequence is called Σ here. In each of the two sequences each neighboring cycle-pair is joined by an edge. The cycles in Σ correspond to the literals of ψ, so there are exactly $24m+1$ cycles in it. On the other hand, the cycles in Π correspond to truth assignments, so there are $2(n+m+1)$ cycles in it. The very bottom node of Σ and the very bottom node of Π are, respectively, called s and t. The first $24m$ cycles of Σ are divided into $8m$ three-cycle blocks, where for each i, $1 \leqslant i \leqslant 8m$, 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

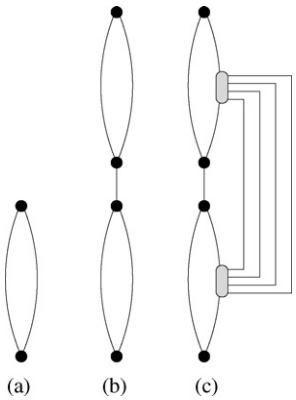

Fig. 4. (a) A two-node cycle. (b) A pair of two-node cycles joined by an edge. (c) A pair of two-node cycles joined by an edge and by an XOR-gadget.

cycle to the second literal, and the third cycle to the third literal (see Fig. 5). The three cycles in each three-cycle block are connected to each other by an OR-gadget attached on their left edges. The last cycle corresponds to the unique single-literal clause of ψ. To the left edge of the two-node cycle we attach a one-input OR-gadget. The sequence Π is divided into $n+m+1$ blocks of cycle-pairs, where for each i, $1 \leqslant i \leqslant n+m+1$, the ith pair corresponds to x_i, the ith variable. For each pair, the top cycle corresponds to $\bar{x}_i$ and the bottom to x_i. We connect the right edges of each pair by an XOR-gadget as shown in Fig. 4. This has the effect of forcing each Hamiltonian path of Π to select for each pair of two-node cycles exactly one cycle whose left edge is traversed, where the other cycle will be traversed on the right edge. Now, for each i, $1 \leqslant i \leqslant 2(n+m+1)$, and each j, $1 \leqslant j \leqslant 24m+1$, if the literal represented by the ith cycle of Π is the literal at the jth position in Σ, join the left edge of the ith cycle in Π and the right edge of the jth cycle in Σ by an XOR-gadget. Here, in the case where two XOR-gadgets connecting Σ and Π need to be crossed, we do so with the method for crossing XOR-gadgets described earlier.

Note that for every i, $1 \leqslant i \leqslant n+m+1$, traversing the XOR-gadget connecting the two right edges in the ith cycle-pair in Π corresponds to selection of a value for the ith variable. Here if the right edge of the top (respectively, the bottom) cycle is used to traverse the XOR-gadget, then it frees up the left edge of the bottom (respectively, the top) cycle, enforcing that all the XOR-gadgets attached to the left edge of the bottom (respectively, the top) cycle are traversed from the Π side and that all the XOR-gadgets attached to the left edge of the top (respectively, the bottom) cycle are traversed from the Σ side. We view this situation as the ith variable assigned the value true (respectively, false). So, each Hamiltonian path of the Π side corresponds to a truth assignment of ψ. Let a Hamiltonian path of the Π side be fixed and let α be the corresponding truth-assignment of ψ. Let j, $1 \leqslant j \leqslant 8m$, be an integer. If α does not satisfy the jth clause, then in each of the three cycles in the jth block in Σ, the right edge needs to be taken so as to traverse the XOR-gadgets attached to it, which

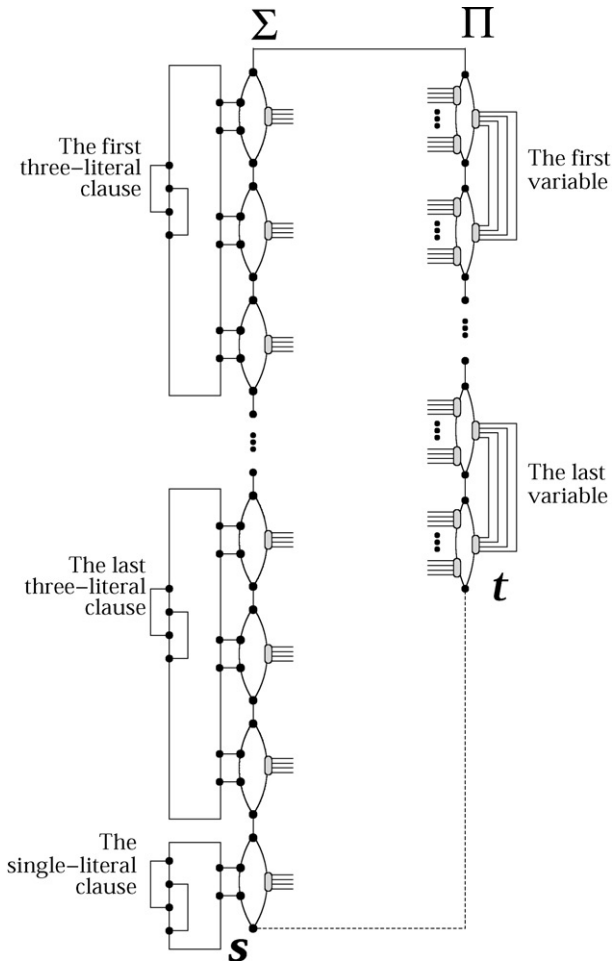

Fig. 5. The two Sequences, Σ and Π.

means that the OR-gadget attached to the block cannot be traversed. If α satisfies exactly one literal, exactly one of the three left edges of the three cycles is available, so there are two ways to traverse all the nodes in the block. If α satisfies exactly two literals, exactly two of the three left edges of the three cycles are available, so there are four ways to traverse all the nodes in the block. As to the last single-cycle with one single-input OR-gadget, if the literal is not satisfied, then there is no way to traverse the OR-gadget, and if the literal is satisfied, then there are two ways to construct a Hamiltonian path in it. The formula ψ is designed so that every satisfying assignment of ψ satisfies exactly $4m$ three-literal clauses by satisfying exactly two literals and exactly $4m$ three-literal clauses by satisfying exactly one literal. So, each Hamiltonian path of G corresponds to a satisfying assignment of ψ. Furthermore, for each satisfying

assignment α of ψ, the number of Hamiltonian paths of G that represent α is

$$2(4^{4m}2^{4m}2^{12(n+m+1)}2^{12(24m+1)}2^{48r}) = 2^{48r+12n+312m+25},$$

where r is the number of crossings of XOR-gadgets. Note that the degree of the nodes in G is all three except s and t, for both of which the degree is two.

By combining the above observations and the results in Proposition 1 and Lemma 3, we obtain the following lemma.

Lemma 4. *The problem of counting Hamiltonian paths in planar graphs of maximum degree three is* #P*-complete under* $\leqslant^{p}_{r\text{-shift}}$*-reductions.*

In the above construction, if we join s and t by an edge, then the resulting graph becomes 3-regular and the edge (s,t) is traversed by every Hamiltonian cycle of G (see the dashed line in Fig. 5). Thus, we have strengthened the NP-completeness result by Garey et al. as follows:

Lemma 5. *The Hamiltonian cycle problem of planar* 3*-regular graphs is* NP*-complete in the sense that there exists a polynomial-time many-one reduction f from* 3SAT *to the problem such that, together with another polynomial time computable function, f acts as a polynomial-time* $\leqslant^{p}_{r\text{-shift}}$*-reduction from* #3SAT *to* #HamCycle-Plan3.

Corollary 6. *The counting problem* #HamCycle-Plan3 *is* #P*-complete under polynomial-time* $\leqslant^{p}_{r\text{-shift}}$*-reduction.*

4. Self-avoiding walks in subgraphs of two-dimensional grids

This section proves #P-completeness of the problem of counting the number of SAWs in subgraphs of two-dimensional grids. As mentioned in the introduction, there are the following six versions of the problem we are concerned with depending on what type of SAWs are counted:

(1) the SAWs from the origin to a specific point having a specific length,
(2) the SAWs from the origin to any point having a specific length,
(3) the SAWs between any two points having a specific length,
(4) the SAWs from the origin to a specific point having any length,
(5) the SAWs from the origin to any point having any length,
(6) the SAWs between any two points having any length.

Theorem 7. *Each of the six types of the problem of counting the number of SAWs in subgraphs of two-dimensional grids is complete for* #P *under* $\leqslant^{p}_{r\text{-shift}}$*-reductions.*

Proof. We first establish that the first three versions are each complete for #P under $\leqslant^{p}_{r\text{-shift}}$-reductions. Let f be an arbitrary #P function. Let R be the reduction from f to #HamPath stated in Lemma 4. It is known that planar graphs can be embedded in two-dimensional grids in polynomial time (for example, see [4]). In the case when

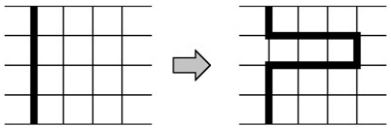

Fig. 6. Bending a vertical line.

the maximum degree is three, the embedding can be made so that there is no vertex congestion, i.e. for every two edges the paths which realize the edges in the two-dimensional grid are vertex disjoint. We pick one such method. Let x be a string for which $f(x)$ needs to be evaluated. Let (G,s,t) be the value of R on input x. Let N be the number of nodes in G. Construct G' from G by adding two nodes s' and t' and two edges (s,s') and (t,t'). We apply our embedding algorithm to G' and obtain a subgraph of a two-dimensional grid. Let U be the set of all images of the nodes in G with respect to this embedding. Since s' and t' are both exterior nodes, we may assume that s' is mapped to the node with the smallest x-coordinate among those in U having the smallest y-coordinate and t' is mapped to the node with the largest x-coordinate among those in U having the largest y-coordinate. We may also assume that s' is $(0,0)$. If not, we will shift the embedding accordingly so that s' is located on $(0,0)$. This is E_0.

We construct from E_0 a new subgraph of a two-dimensional grid, E_1, as follows: We enlarge E_0 by a factor of four. In other words, we move each point (a,b) of E_0 to $(4a,4b)$ and replace each edge $((a,b),(a',b'))$ of E_0 by the four-unit-length straight-line path between $(4a,4b)$ and $(4a',4b')$. For each edge $e=(u,v)$ of G', if the edge e is realized by a straight vertical line, then we bend the straight line as follows: Suppose that the straight line runs from (a,b) to (a,b'), where $b' \leqslant b-4$. We replace the edge $((a,b-1),(a,b-2))$ by the path:

$$(a,b-1),(a+1,b-1),(a+2,b-1),(a+3,b-1),$$

$$(a+3,b-2),(a+2,b-2),(a+1,b-2),(a,b-2).$$

(see Fig. 6). Since we have already enlarged E_0 by a factor of four, bending straight vertical line paths does not interfere with the other paths. On the other hand, if the edge e is realized by a path that includes a horizontal edge of the two-dimensional grid, since we have enlarged the embedding by a factor of four, the path realizing e contains a horizontal line of length four. This is the graph E_1. For every edge e of G', the path realizing e in E_1 contains a horizontal line of length at least three.

Furthermore, we construct a graph E_2 from E_1. Let L be the smallest integer l such that l is a power of 2, $l \geqslant N^2$, and for all edges e of G' the path realizing e in E_1 has length at most l. We enlarge E_1 by a factor of L. For each edge e of G', let $\delta(e)$ denote the length of the path that realizes e in the enlarged E_1. Then, for all edges e of G, $\delta(e)$ is a multiple of $2L$ (since the lengths of the paths realizing edges of G' in E_1 are even), is at most L^2, and is at least $2L$. Also, for all edges e of G', the path in the enlarged E_1 that realizes e contains a horizontal line having length at least $3L$.

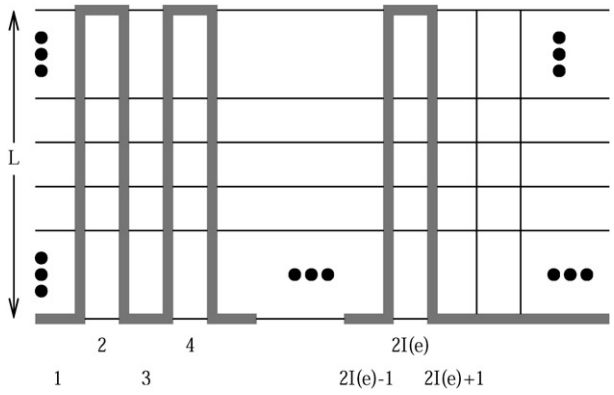

Fig. 7. Construction of towers.

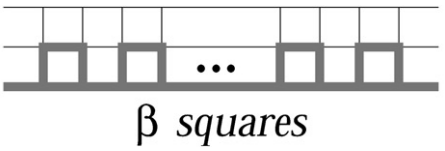

Fig. 8. Squares attached.

Now for each edge e of G' we do the following. We pick a horizontal line of length $2L+1$ that is a part of the path that realizes e, call the $2L+1$ grid edges $a(1),\ldots,a(2L+1)$ from left to right. Let $I(e)=L^2-\delta(e)/2L$. Note that $I(e)$ is an integer. We replace for each i, $1\leqslant i\leqslant I(e)$, the edge $a(2i)$ by the tower of width 1 and of height L going from $a(2i)$ (see Fig. 7). This is E_2. Each tower increases the length of the path by $2L$. So, for all edges e of G', e is realized by a path of length L^2.

Let $h=L^2(N+1)$. Since G' has $N+2$ nodes, every Hamiltonian path of G' is realized by a path in E_2 of length h. Furthermore, every path in E_2 having length h corresponds to a Hamiltonian path in G'. So, regardless of whether the end points of paths are specified or not, the number of SAWs in E_2 having length h is exactly the number of Hamiltonian paths in G'. Let τ' in E_2 be the image of t'. Define $R_1(x)=(E_2,\tau',h)$. Let $R_3(x)$ be the function that does post-computation in the $\leqslant^p_{r\text{-shift}}$-reduction from f to the problem of counting Hamiltonian paths in planar graphs of maximum degree three (see Lemma 4). Then the pair (R_1,R_3) witnesses that the types 1–3 of the counting problem of SAWs in two-dimensional grids are each #P-complete under $\leqslant^p_{r\text{-shift}}$-reductions.

Let $\alpha=4L^2$ and $\beta=L^2$. We make further modifications to E_2 and construct E_3. We enlarge E_2 by a factor of α. This is carried out by transforming each edge $((a,b),(a',b'))$ of E_2 to the straight line of length α between $(\alpha a,\alpha b)$ and $(\alpha a',\alpha b')$. Then for each edge e of G' identify a horizontal line of length α and attach above the line β unit-size squares with a gap of unit length in between (see Fig. 8). This is E_3.

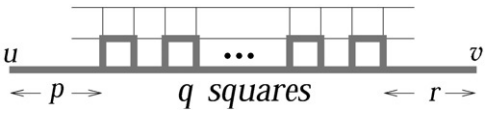

Fig. 9. $S(p,q,r)$.

Note that, for each edge e of G', now e is realized by a set of 2^β paths. So, the number of SAWs between the origin and the image τ of t is at most

$$2^{\beta(N-1)}\#\mathrm{HamPath}(G,s,t) + 2^{\beta(N-2)}\xi,$$

where ξ denotes the number of s–t non-Hamiltonian simple paths in G. Since the degree of every node in G other than s and t is exactly three, we have

$$\xi \leqslant 1 + 2 + 2^2 + \cdots + 2^{N-2} < 2^{N-1}.$$

Since $\beta \geqslant N$, the number of SAWs between the origin and the image of t in E_3 is

$$2^{\beta(N-1)}\#\mathrm{HamPath}(G,s,t) + \rho,$$

where $0 \leqslant \rho < 2^{\beta(N-2)+(N-1)} < 2^{\beta(N-1)}$. So, let $R_1'(x) = (E_3, \tau)$ and let $R_3'(x) = R_3(x) + \beta(N-1)$. Then the pair (R_1', R_3') witnesses that the fourth type is complete for #P under $\leqslant^p_{r\text{-shift}}$-reductions.

To show #P-completeness of the last two problems, let for each triple of positive integers p,q,r, $S(p,q,r)$ be a structure shown in Fig. 9. This is a sequence of q unit-size squares such that each pair of neighboring squares are connected by a unit-length edge at the bottom and a line having length p and a line having length r are respectively attached to the left and the right end of the sequence. Since each edge of e is realized by a path having length L^2 in E_2 and we have enlarged E_2 by a factor of α, the attachment of squares described in the above is equivalent to replacing each edge of e by an $S(p,\beta,r)$ for some positive integers p and r such that $p+r+2\beta-1 = \alpha L^2$.

Let p, q, r, and ℓ be positive integers such that $p + r + 2q - 1 \leqslant \ell$. Let u and v denote the left and the right ends of the structure, respectively. The simple paths within $S(p,q,r)$ have the following properties:

(1) The number of those paths that connect u and v is 2^q.

(2) The number of those paths that touch u but not v is

$$\begin{aligned} &p + 7(1 + 2 + 2^2 + \cdots + 2^{q-1}) + 2^q(r-1) \\ &\quad = 2^q(r+6) + p - 7. \end{aligned}$$

This is less than $2^q\ell$. Similarly, the number of those that touch v but not u is less than $2^q\ell$.

(3) The number of those paths that touch neither u nor v is less than $2^{q+1}\ell^2$. This can be shown as follows:

- The number of length-zero paths touching neither u nor v is at most $2q+\ell-1$.
- The number of length-one paths touching neither u nor v is at most $3q+\ell-2$.

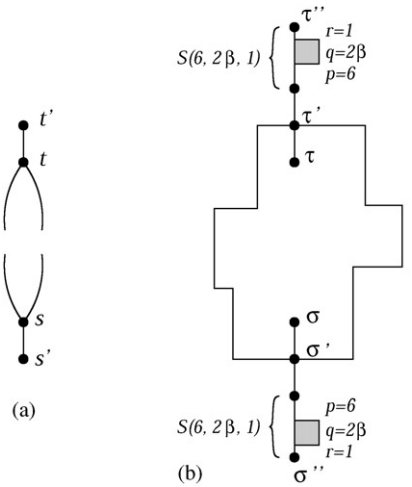

Fig. 10. (a) Graph G' and (b) the corresponding modifications to E_3.

- The number of length-two paths within single squares is $4q$.
- The number of length-three paths within single squares is $4q$.
- The number of paths π such that either the left end of π is a point between u and the leftmost square or the right end of π is a point between v and the rightmost square is less than $(2^q\ell^2)/2$.
- All the paths that are not considered in the previous cases are those that have length more than one and connect between two distinct squares. For every i, $2 \leqslant i \leqslant q$, the number of such paths that touch precisely i squares is equal to $7^2 \cdot 2^{i-2}$ (since there are seven possibilities for one end point and seven for the other end point). The sum of this for all i, $2 \leqslant i \leqslant q$, is $49(2^{q-1} - 1)$.

By summing up all the upper bounds calculated in the above, the number of simple paths that touch neither u nor v is bounded from above by

$$2\ell + 13q + (\ell^2 + 49)2^{q-1}.$$

Let σ, σ', τ, and τ' be, respectively, the images of s, s', t, and t' in E_3. Here σ' has the smallest y-coordinate and τ' has the largest y-coordinate. Then we modify E_3 attaching to σ' a vertical $S(6, 2\beta, 1)$ downwards, where the other end of the S-structure is called σ'', and attaching to τ' a vertical $S(6, 2\beta, 1)$ downwards, where the other end of the S-structure is called τ''. Let this graph be E_4 (see Fig. 10 for the whole construction).

Let Y be the number of s–t Hamiltonian paths in G. Then Y is equal to the number of s'–t' paths having length $N + 1$ in G'. For each simple path π in G', let $M(\pi)$ denote the set of all SAWs $\gamma = [v_1, \ldots, v_k]$ in E_4 such that elimination from γ of 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 cardinality of $M(\pi)$. We first establish #P-completeness of the fifth type.

Let Z_1 be the number of SAWs in E_4 from the origin (recall that the origin of E_4 is σ') to any node in E_4. We evaluate Z_1 as follows:

- Suppose π is a length $N+1$ path from s' to t'. Each SAW corresponding to π begins in σ' or in σ'' or somewhere between these nodes and ends in τ' or in τ'' or somewhere between them. By (1) and (2) in the above, the number of paths that touch σ' and any point between σ' and σ'' (including σ'') is

  $$2^{2\beta} + 7(2^{2\beta}) = 2^{2\beta+3}.$$

  The same holds for the number of paths that touch τ' and any point between τ' and τ''. So,

  $$\mu(\pi) = 2^{2\beta+3}2^{\beta(N+1)}2^{2\beta+3} = 2^{\beta(N+5)+6}.$$

- Suppose π is an s'–t' path having length at most N. Then π does not include a Hamiltonian path of G. So, by an analysis similar to the previous one,

  $$\mu(\pi) \leqslant 2^{2\beta+3}2^{\beta N}2^{2\beta+3} = 2^{\beta(N+4)+6}.$$

- Suppose that π is a path from s' having length at least one to a node other than t'. Then π traverses at most N edges of G'. Let u be the node in E_4 corresponding to the end point of π. There are two directions in E_4 to extend beyond u. By property (2) in the above, there are at most $2(2^{\beta}\alpha L^2)$ possibilities for the selection. Thus,

  $$\mu(\pi) \leqslant 2^{2\beta+3}2^{\beta N}2(2^{\beta}\alpha L^2) = 2^{\beta(N+3)+4}\alpha L^2.$$

- Suppose that π is the single node path s'. Then each SAW corresponding to π begins in σ' and either ends somewhere between σ' and σ'' (including σ'') or somewhere between σ' and σ (excluding σ). Hence by property (2) in the above

  $$\mu(\pi) < 2^{2\beta+3} + 2^{\beta}\alpha L^2.$$

Recall that Y denotes the number of s–t Hamilton paths in G. Then there are exactly Y paths of the first kind. The number of paths π of the second kind is at most

$$1 + 2 + 2^2 + \cdots + 2^{N-2} < 2^{N-1}.$$

The number of paths π of the third kind is at most

$$1 + 2 + 2^2 + \cdots + 2^{N-2} + 2^{N-1} < 2^N.$$

The number of paths π of the last kind is one. Thus, the sum

$$(2^{\beta(N+4)+6})2^{N-1} + (2^{\beta(N+3)+4}\alpha L^2)2^N + (2^{2\beta+3} + 2^{\beta}\alpha L^2)$$

is an upper bound for the number of SAWs in E_4 which correspond to the paths of the last three kinds. Since $\alpha = 4\beta$, $\beta = L^2$, and $L \geqslant N^2$, for all but finitely many N, this sum is less than $2^{\beta(N+5)+6}$. So,

$$Z_1 = (2^{\beta(N+5)+6})Y + \rho_1,$$

where $0 \leqslant \rho_1 \leqslant (2^{\beta(N+5)+6}) - 1$. Thus, Y can be recovered from Z_1 by right-bit-shifting. This proves that the fifth type is #P-complete.

Finally, let Z_2 be the number of any SAWs in E_4. As in the previous cases, we evaluate Z_2 as follows:

- Suppose π is a length $N+1$ path from s' to t'. Then

$$\mu(\pi) = 2^{\beta(N+5)+6}.$$

There are Y paths of this kind.

- Suppose π is an s'–t' path having length at most N. Then

$$\mu(\pi) = 2^{\beta(N+4)+6}.$$

The number of paths of this kind is at most 2^N.

- Suppose that π is a path from s' having length at least one to a node other than t' or a path from t' having length at least one to a node other than s'. By the previous analysis for the third type it holds

$$\mu(\pi) \leqslant 2^{\beta(N+3)+4}\alpha L^2,$$

and there are at most $2(2^N) = 2^{N+1}$ paths of this kind.

- Suppose that π is the single node path s' or the single node path t'. Then

$$\mu(\pi) < 2^{2\beta+3} + 2^{\beta}\alpha L^2.$$

- Suppose that π is a path in G. Then

$$\mu(\pi) \leqslant (2(2^{\beta}\alpha L^2))^2 2^{\beta(N-1)} = 2^{\beta(N+1)+2}\alpha^2 L^4$$

and there are at most $N(2^N)$ paths of this kind.

- Finally, suppose that π is the empty path (i.e. it has no node at all). Then since both s' and t' have degree 1 and the remaining nodes of G' have degree 3, G' has exactly $3N/2+1$ edges, and therefore in E_4 we have $3N/2+1$ structures $S(p,\beta,r)$, with $p+r+2\beta-1=\alpha L^2$, corresponding to the edges. Recall that in E_4 we have additionally 2 structures $S(6,2\beta,1)$. Then, we conclude

$$\begin{aligned}\mu(\pi) &\leqslant (3N/2+1)(2^{\beta}\alpha L^2) + 2(2^{2\beta+3}) \\ &\leqslant (3N/2+3)(2^{2\beta+3}\alpha L^2).\end{aligned}$$

It is not hard to see that

$$Z_2 = 2^{\beta(N+5)+6}Y + \rho_2,$$

where $0 \leqslant \rho_2 \leqslant 2^{\beta(N+5)+6} - 1$. So, Y can be recovered from Z_2 by right-bit-shifting. Thus, the sixth type is #P-complete. □

5. Self-avoiding walks in subgraphs of hypercubes

5.1. #P-completeness

Let f be an arbitrary #P function. Let x be an input to f and let E_4 be the subgraph of a two-dimensional grid that is produced on input x by the reduction defined in the

proof of Theorem 7. We show that the graph E_4 can be embedded in an $O(\log n)$-dimensional hypercube so that there is an integer d such that every edge of E_4 is realized by a path having length d. Then the longest simple path in E_4 has the length $d \cdot ((N+1)(\alpha L^2 + 2\beta) + 2(6+4\beta))$ and it is easy to see that an analysis similar to the one presented in the proof of Theorem 7 can be made. So, it remains to show that such an embedding is indeed possible. In [13,15], methods for embedding a graph in a hypercube with congestion 1 are shown. However, in these methods the dilation of an edge (i.e., the length of the path encoding the edge) is dependent on the Hamming distance between the names of the two end points and thus cannot be used for our purpose.

Lemma 8. *Let $d \geqslant 1$ be an integer. There exists a polynomial time computable embedding Ψ and a pair of polynomial time computable functions $s, \ell : \mathbf{N} \to \mathbf{N}$ such that the following properties hold*:

(1) *$s(n) = O(\log n)$ and $\ell(n) = O(\log \log n)$.*

(2) *For every $n \geqslant 1$, and for every undirected graph $G = (V,E)$ of n nodes having maximum degree d, $\Psi(G)$ is an embedding of G in the $s(n)$-dimensional hypercube $HC^{(s(n))}$ with the following property: Let μ and ν be respectively the embedding of nodes and the embedding of edges specified by $\Psi(G)$. Then,*
   - *for all nodes u and v in G, $u \neq v$ implies $\mu(u) \neq \mu(v)$, and*
   - *for every edge $e = (u,v)$ in G, the dilation of $\nu(e)$ (i.e. its length) is $2^{\ell(n)}$ and the path visits no $\mu(w)$ between $\mu(u)$ and $\mu(v)$.*

Proof. Let d, n, and m be positive integers. Let G be a graph of maximum degree d having n nodes and m edges. It holds 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 $q \geqslant \lceil \log n \rceil$. 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 each nonempty binary string u, let $\bar{u}$ 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 four components.

- The node part: This part has length $2q$. For each node $u = u_1 \cdots u_q$ of G, u is encoded as $\bar{u}u$, which has exactly q 1's.
- The edge part: This part has length $4q$ and is used to encode the edge to be traversed. For each edge $e = e_1 \cdots e_{2q}$, e is encoded as $\bar{e}e$, which has exactly $2q$ 1's.
- The neighbor part: This part has length d. When an edge (u,v) is traversed, this part is used to encode either the value of $I_u(v)$ or the value of $I_v(u)$.
- The switch part: This part has length 2.

Let u, v, w, y be binary strings having length $2q$, $4q$, d, and 2, respectively. Then, $\langle u,v,w,y \rangle$ denotes the node $uvwy$ in H.

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

$$\mu(u) = \langle \bar{u}u, 0^{4q}, 0^d, 00 \rangle.$$

As for v, let $e = (u, v)$ be an edge in G such that $u < v$. Let $A = a_1 \cdots a_{2q} = \bar{u}u$, $B = b_1 \cdots b_{2q} = \bar{v}v$, $C = c_1 \cdots c_{4q} = \bar{e}e$, $W_1 = 0^{I_u(v)-1}10^{d-I_u(v)}$, 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, $1 \leqslant t \leqslant q$, 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, $1 \leqslant t \leqslant 2q$, let $C_t = c_1 \cdots c_{k_t} 0^{4q-k_t}$. Note that $A_q = B_q = 0^{2q}$ and $C_{2q} = C$. The edge e is realized by joining a path from $\langle A, 0^{4q}, 0^d, 00\rangle$ to $\langle 0^{2q}, C, 0^d, 11\rangle$ and a path from $\langle B, 0^{4q}, 0^d, 00\rangle$ to $\langle 0^{2q}, C, 0^d, 11\rangle$, each having length $3q + 4$. So, the total length is $6q + 8$, which is a power of 2 by assumption. The first path is defined by:

$$\langle A, 0^{4q}, 0^d, 00\rangle, \langle A, 0^{4q}, W_1, 00\rangle, \langle A, 0^{4q}, W_1, 10\rangle, \langle A, C_1, W_1, 10\rangle, \ldots, \langle A, C_{2q}, W_1, 10\rangle,$$

$$\langle A_1, C, W_1, 10\rangle, \ldots, \langle A_q, C, W_1, 10\rangle, \langle 0^{2q}, C, 0^d, 10\rangle, \langle 0^{2q}, C, 0^d, 11\rangle.$$

The second path is defined similarly with B in place of A. Every node appearing on the two paths satisfies one of the following conditions:

(i) C appears in the edge part.
(ii) A appears in the node part and W_1 appears in the neighbor part.
(iii) B appears in the node part and W_2 appears in the neighbor part.

This implies that 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. This property can be expressed formally as follows:

Lemma 9. *For each $f \in$ #EXP, there is a function R 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 $R(x)$, there are exactly three clauses in which one of y and $\bar{y}$ appears.*
   (b) *For each string x, let $v(x)$ denote the number of variables in $R(x)$. Then v is polynomial time computable.*

(c) *For each string* x, *let* $\mu(x)$ *denote the number of clauses in* $R(x)$. *Then* μ *is polynomial time computable.*
(d) *For each string* x *and each* i, $1 \leqslant i \leqslant \mu(x)$, *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, $1 \leqslant i \leqslant v(x)$, *let* $S(x,i)$ *be the set of all indices* j *such that the* i*th variable appears in* $C(x,j)$. *Then* S *is polynomial time computable.*

Proof. A proof of this result can be found in [16, Chapter 20]. □

Now apply the conversion given in Lemma 3 to each formula generated by function R given in the lemma above. Let R' denote the resulting transformation from the set of strings to NAE3SAT. Since R is locally polynomial-time computable, so is R'. Modify the construction from the proof of Lemma 4 so that the crossing of XOR-gadgets is allowed. Then the maximum degree remains three and the scaling factor becomes $2^{12n+312m+25}$. The connectivity of the gadgets in the graph essentially depends only on the occurrences of the 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 subgraphs of hypercubes by H. 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 $H(x)$.

Lemma 10. *The function* d *defined above is polynomial-time computable. There are polynomial-time computable functions* C_N *and* C_E *such that* $C_N(x)$ *and* $C_E(x)$ *are both polynomial-size boolean circuits and accept* $N(x)$ *and* $E(x)$, *respectively.*

This lemma gives the following theorem.

Theorem 11. *Each of the six types of the problem of counting the number of SAWs in subgraphs of the hypercubes is* #EXP-*complete under* $\leqslant^{p}_{r\text{-shift}}$-*reductions if the subgraphs are specified by circuits.*

6. Complexity of checking the self-avoiding property

This section studies the problem of testing whether a path on the two-dimensional grid is a SAW and it concentrates on the space complexity for this problem. A motivating question to our study is whether counting of SAWs on the two-dimensional grid belongs to the class #L [1,2]. Though we leave this question open, the main result of this section (Theorem 13) could suggest a negative answer to this problem. As an application we give at the end of the section an automata-theoretic proof of the formula for the number of the self-avoiding walks that move up and sideways but not down.

We encode SAWs using strings over the alphabet $\{U,D,L,R\}$ in a natural way: symbols of such strings will describe in an obvious way (i.e. U stays for up-move,

D for down, and so on) successive unit-length parts of a given SAW. Let L_{SAW} denote the set of all strings over $\{U,D,L,R\}$ which encode correct SAWs. We use the standard notation for logspace-bounded complexity classes. By L we denote the set of all languages decided by deterministic Turing machines working in logarithmic space. In the case of nondeterministic machines we use the notation NL. Moreover, if we restrict the nondeterministic TMs to machines with 1-way input heads then we denote the corresponding class of languages by 1-NL. Let co-1-NL denote the set of all languages whose complements belong to 1-NL. It is easy to see that L_{SAW} belongs to L and to co-1-NL.

Proposition 12. $L_{\mathrm{SAW}} \in \mathrm{L} \cap$ co-1-NL.

Proof. Let $n \geqslant 1$ be an integer and $w = w_1 \cdots w_n$ be a word of length n over the alphabet $\Gamma = \{U,D,L,R\}$. Then $w \notin L_{\mathrm{SAW}}$ if and only if there exist some i and j, $1 \leqslant i < j \leqslant n$ such that the path $w_i \cdots w_j$ is a cycle. Let $y = y_1 \cdots y_m$ be a nonempty path in Γ^*. Then y is a cycle if and only if

$$|\{i \mid 1 \leqslant i \leqslant m \wedge y_i = U\}| - |\{i \mid 1 \leqslant i \leqslant m \wedge y_i = D\}| = 0$$

and

$$|\{i \mid 1 \leqslant i \leqslant m \wedge y_i = L\}| - |\{i \mid 1 \leqslant i \leqslant m \wedge y_i = R\}| = 0.$$

It is easy to see that each of these equalities can be tested using a deterministic logspace Turing machine with 1-way input head. To test non-membership in L_{SAW}, a 1-way nondeterministic logspace machine has only to select nondeterministically i and j while scanning the input and test whether the portion of the input is a cycle. Thus, $\overline{L_{\mathrm{SAW}}} \in$ 1-NL, so $L_{\mathrm{SAW}} \in$ co-1-NL. This test can be carried out deterministically in logspace if the input head is two-way. Thus, $L_{\mathrm{SAW}} \in \mathrm{L}$. □

Proposition 12 immediately raises the question of whether $L_{\mathrm{SAW}} \in$ 1-NL. How likely is it that this containment holds? In the theorem below we prove that the containment does not hold. In fact we prove even more:

Theorem 13. *No* 1*-way nondeterministic Turing machine recognizes* L_{SAW} *in space* $\mathrm{o}(n)$.

Proof. Let us assume that a nondeterministic Turing machine M with a 1-way input head recognizes the language L_{SAW} in a sublinear space. We will construct below, for all but finitely many n, a non-self-avoiding walk having length $3(9n)+5$ whose encoding is accepted by M.

Let $n > 0$ be fixed. First we construct a family of 2^n SAWs each having length $27n+5$. Central to the construction are the gadgets G_0 and G_1 shown in Fig. 11. Both G_0 and G_1 consist of three SAWs that are vertically lined up and have length nine. In each of the two gadgets, of the three SAWs, we call the top one the *upper border*, the middle one the *backbone*, and the bottom one the *lower border*.

The SAWs are constructed assuming that the upper border and the backbone are traversed from left to right and the lower border from right to left. Let A_0 (respectively,

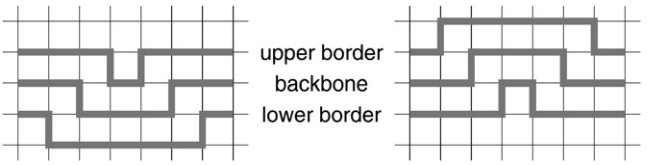

Fig. 11. The SAWs of the gadget G_0 (left) and G_1 (right).

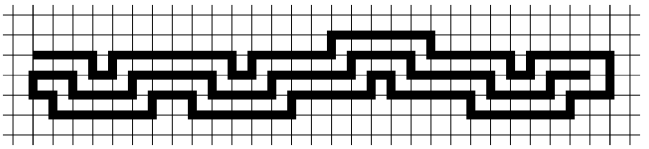

Fig. 12. The SAW encoded by the string $W(0010)$.

B_0 and C_0) be the string that encodes the traversal of the upper border (respectively, the lower border and the backbone) of G_0. Similarly, define A_1, B_1, and C_1 for G_1. We have:

$$A_0 = RRRDRURRR,$$

$$B_0 = LDLLLLLUL,$$

$$C_0 = RRDRRRURR,$$

$$A_1 = RURRRRRDR,$$

$$B_1 = LLLULDLLL, \text{ and}$$

$$C_1 = RRURRRDRR.$$

Let $X = b_1 b_2 \ldots b_n$ be an arbitrary binary string of length n. Define

$$W(X) = A(X)\ RDDL\ B(X)\ U\ C(X),$$

where $A(X) = A_{b_1} A_{b_2} \ldots A_{b_n}$, $B(X) = B_{b_n} \ldots B_{b_2} B_{b_1}$, and $C(X) = C_{b_1} C_{b_2} \ldots C_{b_n}$.

It is easy to see that for all $X \in \{0,1\}^n$, $W(X)$ encodes a self-avoiding walk (see the illustration in Fig. 12).

Clearly, for all pairs of distinct length-n binary strings, X and Y, $W(X) \neq W(Y)$, so $F_n = \{W(X) \mid X \in \{0,1\}^n\}$ has cardinality 2^n. This is the family of the SAWs for integer n. Now using a counting argument one can show that if M recognizes L_{SAW} in a sublinear space then there exist two different binary strings $X, Y \in \{0,1\}^n$ such that M cannot distinguish between the prefixes $A(X)RDDLB(X)$ and $A(Y)RDDLB(Y)$. More precisely, we argue as follows: For each $X \in \{0,1\}^n$, select one accepting computation path of M on input $W(X)$. Let $\alpha(X)$ denote the path. For each $X \in \{0,1\}^n$, let $\sigma(X)$ be the configuration of M on input $W(X)$ along the path $\alpha(X)$ immediately after the last symbol of $B(X)$ is read. Since the head of M is one-way and M is sublinear space

bounded, there are $2^{o(n)}$ possibilities for $\sigma(X)$. Thus, for n sufficiently large, there exist two strings X and Y, $X \neq Y$, such that $\sigma(X) = \sigma(Y)$. Pick such a combination of X and Y. Since $X \neq Y$, there exists an i, $1 \leqslant i \leqslant n$, such that X and Y disagree at the ith bit position. Pick such an i. Without loss of generality, we can assume that the ith bit of X is a 0 and that of Y is a 1. Since both $\alpha(X)$ and $\alpha(Y)$ are accepting computation paths and go through the same configuration immediately after the B-part has been read, M accepts on input Z, defined by

$$A(X)\ RDDL\ B(X)\ U\ C(Y).$$

The three SAWs comprising the ith component of the SAW encoded by Z are the lower and upper borders of G_0 and the backbone of G_1. So, in the ith component the upper border crosses the backbone. This implies that Z does not encode an SAW. Thus, we have a contradiction. □

The situation changes drastically if one restricts the walks on the two-dimensional grid to the "up-side" SAWs, i.e. the self-avoiding walks that move up and sideways but do not move down. Let us denote by $L_{\text{up-side}}$ the set of all strings over $\{U, L, R\}$ that encode up-side SAWs. Then obviously this language can be recognized even by a deterministic 1-way finite automaton. Below we show how using this fact one can deduce a formula for the number of up-side SAWs of a given length n. Let us fix the following automaton M for $L_{\text{up-side}}$, with the states $Q = \{q_U, q_L, q_R\}$, where all the three states are accepting and q_U is the initial state. The transition function $\delta : Q \times \{U, L, R\} \to 2^Q$ is defined as follows: for all $x, y \in \{U, L, R\}$

$$\delta(q_x, y) = \begin{cases} \emptyset & \text{if } (x, y) \in \{(L, R), (R, L)\}, \\ \{q_y\} & \text{otherwise.} \end{cases}$$

Now consider the transition matrix A defined as follows:

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}.$$

The matrix A can be thought of as encoding the following rule: If the states q_U, q_L, q_R correspond, respectively, to the positions $1, 2, 3$ then $A(i, j) = 1$ if and only if M, being in the state corresponding to i, can reach in one step the state corresponding to j. Define sequences $\{f_n\}_{n \geqslant 1}$, $\{g_n\}_{n \geqslant 1}$, and $\{h_n\}_{n \geqslant 1}$ as follows: For all $n \geqslant 1$,

$$\begin{bmatrix} f_n \\ g_n \\ h_n \end{bmatrix} = A^n \cdot \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}.$$

Then it is easy to see that, for all integers $n \geqslant 1$, f_n (respectively, g_n and h_n) is equal to the number of words having length n that are accepted by M when it starts in state

q_U (respectively, q_L and q_R). Thus, f_n is equal to the number of up-side SAWs having length n. To obtain a formula for f_n, observe that

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 3 \\ 2 \\ 2 \end{bmatrix}$$

and that, for all $n \geqslant 1$, if

$$A^n \cdot \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} a_{2n} \\ a_{2n-1} \\ a_{2n-1} \end{bmatrix}$$

then

$$\begin{bmatrix} a_{2(n+1)} \\ a_{2(n+1)-1} \\ a_{2(n+1)-1} \end{bmatrix} = A^{n+1} \cdot \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} a_{2n} \\ a_{2n-1} \\ a_{2n-1} \end{bmatrix} = \begin{bmatrix} a_{2n} + 2a_{2n-1} \\ a_{2n-1} + a_{2n-1} \\ a_{2n-1} + a_{2n-1} \end{bmatrix}.$$

It is easy to check that the integer sequence $\{a_n\}_{n \geqslant 1}$ defined above has the property that for all integer $n \geqslant 1$

$$a_{2n-1} = \frac{1}{2\sqrt{2}} [(1+\sqrt{2})^{n+1} - (1-\sqrt{2})^{n+1}] \quad \text{and}$$

$$a_{2n} = \frac{1}{2} [(1+\sqrt{2})^{n+1} + (1-\sqrt{2})^{n+1}]$$

(for a generating function of this sequence see e.g. [22]). Thus, the formula for a_{2n} gives the number of up-side SAWs having length n on the two-dimensional grid. This is an alternative method for obtaining the formula by Williams [28].

Acknowledgements

The authors are grateful to Jörg Rothe for his helpful comments and his careful reading of a draft to detect errors in an early version of the paper and to Andreas Jakoby and Dick Lipton for stimulating discussions. Also, the authors thank the anonymous referees for their remarks, which helped the authors in improving the presentation.

References

[1] E. Allender, M. Ogihara, Relationships among PL, #L, and the determinant, RAIRO—Theoret. Inform. Appl. 30 (1996) 1–21.

[2] C. Álvarez, B. Jenner, A very hard Log-space counting classes, Theoret. Comput. Sci. 107 (1993) 3–30.

[3] F. Barahona, On the computational complexity of Ising spin glass models, J. Phys. A 15 (3) (1982) 3241–3245.
[4] M. Chrobak, T.H. Payne, A linear time algorithm for drawing a planar graph on a grid, Inform. Process. Lett. 54 (4) (1995) 241–246.
[5] A.R. Conway, A.J. Guttmann, Square lattice self-avoiding walks and corrections-to-scaling, Phys. Rev. Lett. 77 (26) (1996) 5284–5287.
[6] M. Garey, D. Johnson, E. Tarjan, The planar Hamiltonian circuit problem is NP-complete, SIAM J. Comput. 5 (4) (1976) 704–714.
[7] J. Goldsmith, M. Ogihara, J. Rothe, Tally NP sets and easy census functions, Inform. Comput. 158 (2000) 29–52.
[8] J.M. Hammersley, K.W. Morton, Poor man's Monte Carlo, J. Roy. Statist. Soc. B 16 (1954) 23–38.
[9] J.M. Hammersley, D.J.A. Welsh, Further results on the rate of convergence to the connective constant of the hypercubical lattice, Quart. J. Math. Oxford 13 (2) (1962) 108–110.
[10] H.B. Hunt III, M.V. Marathe, V. Radhakrishnan, R.E. Stearns, The complexity of planar counting problems, SIAM J. Comput. 27 (4) (1998) 1142–1167.
[11] S. Istrail, Statistical mechanics, three-dimensionality and NP-completeness: I. Universality of intractability for the partition function of the Ising model across non-planar surfaces, in: Proc. 31st Symp. on Theory of Computing, ACM Press, New York, 2000, pp. 87–96.
[12] R.M. Karp, R.J. Lipton, Some connections between nonuniform and uniform complexity classes, in: Proc. 12th Symp. on Theory of Computing, ACM Press, New York, 1980. pp. 302–309.
[13] Y.M. Kim, T.-H. Lai, The complexity of congestion-1 embedding in a hypercube, J. Algorithms 12 (1991) 246–280.
[14] N. Madras, G. Slade, The Self-Avoiding Walk, Birkhäuser, Boston, MA, 1993.
[15] A. Matsubayashi, S. Ueno, Small congestion embedding of graphs into hypercubes, Networks 33 (1999) 71–77.
[16] C. Papadimitriou, Computational Complexity, Addison-Wesley, Reading, MA, 1994.
[17] C. Papadimitriou, M. Yannakakis, A note on succinct representations of graphs, Inform. Control 71 (3) (1986) 181–185.
[18] A. Pönitz, P. Tittman, Improved upper bounds for self-avoiding walks in Z^d, Electron. J. Combinatorics 7 (1) (2000) #R21. http://www.combinatorics.org/Volume_7/v7i1toc.html.
[19] J.S. Provan, The complexity of reliability computations in planar and acyclic graphs, SIAM J. Comput. 15 (3) (1986) 694–702.
[20] D. Randall, A. Sinclair, Self-testing algorithms for self-avoiding walks, J. Math. Phys. 41 (3) (2000) 1570–1584.
[21] T. Schaefer, The complexity of satisfiability problems, in: Proc. 10th Symp. on Theory of Computing, ACM Press, New York, 1978, pp. 216–226.
[22] N.J.A. Sloane, S. Plouffe, The Encyclopedia of Integer Sequences, Academic Press, San Diego, CA, 1995.
[23] S. Toda, PP is as hard as the polynomial time hierarchy, SIAM J. Comput. 20 (5) (1991) 865–877.
[24] S.P. Vadhan, The complexity of counting in sparse, regular, and planar graphs, SIAM J. Comput. 31 (2) (2001) 398–427.
[25] L. Valiant, The complexity of computing the permanent, Theoret. Comput. Sci. 8 (2) (1979) 189–201.
[26] L. Valiant, The complexity of enumeration and reliability problems, SIAM J. Comput. 8 (3) (1979) 410–421.
[27] D. Welsh, Complexity: Knots, Colourings and Counting, Cambridge University Press, Cambridge, 1993.
[28] L. Williams, Enumerating up-side self-avoiding walks on integer lattices, Electron. J. Combinatorics 3 (1) (1996), Research Paper 31.