MATCH
Communications in Mathematical and in Computer Chemistry

MATCH Commun. Math. Comput. Chem. **67** (2012) 761-772

ISSN 0340 - 6253

A Note Concerning the Algorithmic Analysis of Polymer Thermodynamics

J. Andres Montoya
Universidad Industrial de Santander
jmontoya@matematicas.uis.edu.co

(Received July 4, 2011)

Abstract

In this work we study a tally counting problem arising from a discrete model of polymer thermodynamics: The Model of Self-avoiding walks constrained to lattice strips. We show that the partition function of this model can be computed in time $O\left(\log^2(n)\right)$ using a polynomial number of processors

The qualitative properties of some models of statistical mechanics are completely encoded into a discrete function called *the partition function of the model* [10]. Most thermodynamical quantities describing the dynamics and structure of those models, like for example the *free energy* of the system, can be computed from their partition functions: solving a discrete model of statistical mechanics means computing its partition function. Most of the time partition functions are defined as counting problems [12]. We consider, in our research, a discrete model of statistical mechanics: the *self-avoiding walk model.* We study the computational complexity of computing the partition function of this model, we review the known facts and we prove that there exists a $O\left(\log^2(n)\right)$ parallel time algorithm that computes the number of self-avoiding walks constrained to lattice strips of fixed height.

The relation between self-avoiding walks and polymer chains is based on the conformational statistics of polymer chains, S. Edwards and his collaborators have shown that the Self-avoiding walk model can be used as the lattice model describing the relations and statements of the theory of excluded volume (much more information can be found in [3]), which is the basic thermodynamical theory of polymer space organization (see [4]).

Thus, we have that a good understanding of the partition function for self-avoiding walks on two-dimensional grids yields deep information concerning the evolution of polymer chains. We consider a restricted model of polymer chains, the model of polymer chains constrained to strips of fixed height, which is important in the study of globular proteins [5]. The self-avoiding walk model related to this physical model is the model of two-dimensional self-avoiding walks constrained to lattice strips of fixed height. We prove in this paper, that the partition function of the later model can be computed in time $O\left(\log^2(n)\right)$ employing a polynomial number of processors.

Relations to Previous Work. The self-avoiding walk model is an important model of polymer thermodynamics. This model is closely related to the counting problem $\#SAW$ defined below.

Given $n \geq 1$, the symbol $\mathcal{L}_2^n$ denotes *the square lattice of order* n, which is the graph defined by:

- $V(\mathcal{L}_2^n) = [n] \times [n]$, where $[n]$ is equal to the set $\{1, ..., n\}$.
- $E(\mathcal{L}_2^n) = \{\{(a,b),(c,d)\} : |a-c| + |b-d| = 1\}$.

The problem $\#SAW$ is the *tally counting problem* defined by

Problem 1 *($\#SAW$, the self-avoiding walk problem)*

- *input:* 1^n, *where* $n \in \mathbb{N}$.
- *Problem: compute the number of simple paths (of any length) contained in the square lattice of order* n.

In despite of the intensive work related to this problem, so few is known. We don't know of the existence of closed formulae for the counting function encoded by the problem, we do not know of the existence of efficient algorithms solving the problem and we don't know of the existence of hardness proofs that could explain, to some extent, the intractability of this problem.

Most counting problems are hard, there not exist polynomial time algorithms solving them, but many of those problems can be probabilistic approximated. Randall and Sinclair have found a $RPTAS$ (*Randomized polynomial time approximation scheme*) for $\#SAW$ [9]. Randall-Sinclair algorithm can be considered as the only important result, (related to the computational feasibility of the problem $\#SAW$), that have been obtained up to the date.

If we modify some of the parameters in the definition of the problem, we can get feasible versions of it. Some feasible versions of $\#SAW$ have been identified and studied. It is known, for instance, that the counting of *up-side* and *spiral self-avoiding walks* can be carried out in linear time [8]. Klein [6] and Wall-Klein [7] studied a related model, *The Model of Self-avoiding Walks constrained to Lattice Strips of Fixed Height*, this model is important in the study of globular proteins and long polymer chains [5]. We study the partition function of this problem, denoted by $\#SAW_r$, and we exhibit a parallel algorithm that computes the function $\#SAW_r$ in time $O\left(\log^2(n)\right)$.

1 Tally counting problems: Welsh's problem

A counting problem is a function $f : \Sigma^* \to \mathbb{N}$. A counting problem $f : \Sigma^* \to \mathbb{N}$ is a *tally counting problem* if and only if the size of Σ is equal to 1. If $f : \Sigma^* \to \mathbb{N}$ is a tally counting problem we assume that Σ is equal to $\{1\}$. There are many interesting tally counting problems, some of them related to mathematical chemistry, consider for instance the following problems:

1. $f_{CG} : \Sigma^* \to \mathbb{N}$ is the tally counting problem defined by

$$f_{CG}(1^n) = \#\text{ of connected planar graphs of size } n$$

2. $f_M : \Sigma^* \to \mathbb{N}$ is the tally counting problem defined by

$$f_M(1^n) = \#\text{ of perfect matchings contained in the square lattice } \mathcal{L}_2^n$$

3. $f_{ICE} : \Sigma^* \to \mathbb{N}$ is the tally counting problem defined by

$$f_{ICE}(1^n) = \#\text{ of eulerian orientations of the square lattice } \mathcal{L}_2^n$$

We are particularly interested in the tally counting problem $\#SAW$, defined above. This problem is important because it encodes the partition function of The Self-Avoiding Walk Model of polymer thermodynamics [12]. No efficient algorithms are known, and it is also unknown if this problem is hard.

Problem. ***(Welsh's problem)***

Either exhibit polynomial time algorithms solving problem $\#SAW$ or prove that $\#SAW$ is $\#P_1$ complete [12], [11].

Unfortunately we do not solve Welsh's problem. In this paper we consider a restriction of $\#SAW$ that arises from a closely related model of polymer thermodynamics and we develop efficient algorithms solving this new problem.

Remark 2 *What is special, and suspicious, about tally counting problems is the unary encoding of their instances. Consider the problem #SAW. Given $n \geq 1$, the search space associated to the instance 1^n is equal to $([n] \times [n])^n$ and its size is equal to $2^{\Omega(n \log(n))}$. It implies that the running time of the brute force algorithm for #SAW is equal to $2^{\Omega(n \log(n))}$. And it means that tally counting problems do not (necessarily) become tractable because of the unary encoding of their instances. If we would use a binary encoding of the instances, we would get a problem that can be solved in superexponential time using a naive brute force approach and which cannot be solved in polynomial time because of the length of the outputs. The quest for polynomial time algorithms (polylog time algorithms) solving the problem #SAW makes sense if we consider the unary encoding used in definition 1. Also, the unary encoding employed in the definition of tally counting problems* ***is not*** *a naive trick that makes intractable problems look easy.*

1.1 Schutzenberger–Bertoni Method

In this section we introduce the Schutzenberger–Bertoni Method.

Given Σ a finite alphabet and given $L \subseteq \Sigma^*$ a formal language, the *census function* of L is the function $f_L : \mathbb{N} \to \mathbb{N}$ defined by

$$f_L(n) = |\{w \in L : |w| = n\}|$$

Let f be a tally counting problem.

Definition 3 *We say that f is SB reducible if and only if there exists $k \geq 1$, there exist k unambiguous context-free languages $L_1, ..., L_k$, there exist k polynomial functions $p_1(X), ..., p_k(X) \in FNC^2$ and there exist $c_1, ..., c_k \in \mathbb{N}$ such that for all $n \geq 0$ the equation*

$$f(n) = \sum_{i=1}^{k} c_i f_{L_i}(p_i(n))$$

holds.

The Schutzenberger–Bertoni method is based on the SB reducibility notion and the theorem below.

Theorem 4 ***(Schutzenberger–Bertoni method)***

1. *If f is SB reducible, problem f can be solved in time $O\left(\log^2(n)\right)$ using a polynomial number of processors.*

2. If f is SB reducible, there exists a RPTAS for f.

Item 1 can be used to design efficient parallel algorithms, item 1 was proved by Bertoni, Goldwurm and Sabadini (see [1]). Item 2 can be used to design efficient approximation algorithms, it was proved by Bertoni, Goldwurm and Santini [2]. Theorem 4 yields a robust counting technique that we call Schutzenberger–Bertoni method. From now on, when we say Schutzenberger–Bertoni method, we mean the counting method encoded in the statement of theorem 4.

2 The model, the problem and the algorithm

Klein [6], and Wall-Klein [7] studied long polymer chains constrained to infinite long lattice strips, their studies give rise to a new model of polymer thermodynamics: *The Model of Self-avoiding Walks constrained to Lattice Strips of Fixed Height*. We can think of this model as a parameterized model: if we fix a positive integer r, we get *The Model of Self-avoiding Walks constrained to Lattice Strips of Height r*, which can be considered as the r^{th} slice of the former problem. The partition function of this slice is encoded by the tally counting problem $\#SAW_r$ defined by:

Problem 5 *($\#SAW_r$: counting self-avoiding walks in lattice strips of height r)*

- *Input:* 1^n, *where n is a positive integer.*
- *Problem: compute the number of self-avoiding walks contained in the rectangular lattice* $[n] \times [r]$.

We will prove that there exists a parallel algorithm that solves the problem $\#SAW_r$ in time $O\left(\log^2(n)\right)$. Our algorithm is based on the Schutzenberger–Bertoni method.

2.1 Efficient algorithms

Let $r \geq 1$, we exhibit in this section a parallel algorithm solving the problem $\#SAW_r$. From now on we fix $r \geq 1$.

2.1.1 An encoding

Our algorithm is based on Schutzenberger–Bertoni method and a special encoding of the substructures of the rectangular lattices of height r. This encoding can be employed to solve many other counting problems related to lattices.

A *Bilateral slice* is a lattice graph isomorphic to $\{0,1\} \times [r]$. *Left slices* are the subgraphs of bilateral slices that can be obtained by eliminating all the edges of the form $\{(1,x),(1,x+1)\}$.

Left patterns (*patterns*, for short) are the isomorphism types of the subgraphs of a left slice (any two left slices of height r are isomorphic).

Lemma 6 *Given $r \geq 1$, there exist at most 2^{2r} patterns.*

The lemma above follows from the fact that any pattern is a subset of the set of edges of a left slice. It implies that we can enumerate in time $2^{O(r)}$ the set of left patterns, we use the symbol $\mathcal{P}_L$ to denote this set.

Given p and q two patterns, we can think of pq, the concatenation of p and q, as a subgraph of $\{1,2,3\} \times [r]$. The set of nodes of pq that are located on $\{2\} \times [r]$ will be called the *core* of pq (or the 2-fiber of pq), and will be denoted with the symbol $\varsigma(pq)$. Given $i \in \{0,1,2,3,4\}$ we define a function $\alpha_i : \mathcal{P}_L \times \mathcal{P}_L \rightarrow \mathbb{N}$ in the following way:

$$\alpha_i(p,q) = \left|\left\{v \in \varsigma(pq) : \deg_{pq}(v) = i\right\}\right|$$

where $\deg_{pq}(v)$ denotes the degree of v as a node of pq. We define analogous functions $\beta_3, \beta_1 : \mathcal{P}_L \rightarrow \mathbb{N}$ which count the number of degree-three and degree-one nodes located on the 0-fiber of the pattern being evaluated. We note that we can compute in time $2^{O(r)}$ the tables of each one of the functions $\alpha_0, \alpha_1, \alpha_3, \alpha_4, \beta_3$ and β_1.

We can see any subgraph of $[n] \times [r]$ as a word written with patterns. Observe that given γ, a subgraph of the rectangular lattice $[n] \times [r]$, subgraph γ is the concatenation of a sequence $p_1 ... p_n$ of n patterns such that all the edges contained in p_n are located on its left column. Also, we can try to encode subgraphs of $[n] \times [r]$ as words of length n whose characters are patterns.

2.1.2 A technical lemma

Let $r \geq 1$ and let $\mathcal{A}_r$ be the set of *finite acyclic subgraphs* of $\mathbb{N} \times [r]$ whose total degree is upperbounded by 2 (i.e. given $H \in \mathcal{A}_r$ and given v a node of H, the degree of v, as a node of H, is bounded by 2).

Lemma 7 *Given $r \geq 1$, one can compute in time $2^{O(r)}$ a finite state automaton $\mathcal{M}_r$ that recognizes the set $\mathcal{A}_r$.*

Proof. We want to prove that the language

$$\Omega_r = \{p_1 ... p_n \in \Sigma^* : \Psi\}$$

is regular, where Ψ is equal to the conjunction of the following constraints:

1. All the edges occurring in p_n are vertical edges located on its left column and for any $i \leq n-1$ we have that the pair (p_i, p_{i+1}) is an admissible pair.

2. $\beta_1(p_1) + \sum_{i \leq n-1} \alpha_1(p_i, p_{i+1}) = 2$ and the equations $\beta_0(p_1) = \beta_3(p_1) = 0$ hold.

3. The pattern chain $p_1 p_2 ... p_n$ forbids the creation of cycles.

We can write Ψ as $\Psi_1 \wedge \Psi_2$, where Ψ_1 is the conjunction of the first two constraints, and Ψ_2 is equal to:

The pattern chain $p_1 p_2 ... p_n$ forbids the creation of cycles and there are not degree-three nodes contained in $p_1 p_2 ... p_n$

We define two languages

$$\Omega_1 = \{p_1 ... p_n \in \Sigma^* : \Psi_1\}$$

and

$$\Omega_2 = \{p_1 ... p_n \in \Sigma^* : \Psi_2\}$$

We observe that $\Omega_r = \Omega_1 \cap \Omega_2$. Recall that the intersection of two regular languages is regular. Also, it is sufficient to show that both languages, Ω_1 and Ω_2, are regular. First at all we show that Ω_1 is a regular language, and we also show that a finite state automaton $\mathcal{M}_1$ recognizing Ω_1 can be computed in time $2^{O(r)}$.

Recall that we can compute the sets $\mathcal{P}_L$ and $\mathcal{I}$; and the tables of α_0, α_1, α_3, α_4, β_0, β_1 and β_3 in time $2^{O(r)}$. Also, before computing the automaton we compute all these objects which are used by $\mathcal{M}_1$ as lookup tables: the tables of α_0, α_1, α_3, α_4, β_0, β_1 and β_3 are incorporated into the transition function of $\mathcal{M}_1$. Let $\mathcal{M}_1$ be the finite state automaton (Q, q_0, F, δ) defined by:

1. $Q = \{(q, i) : q \in \Sigma \ \& \ 0 \leq i \leq 2\ \} \cup \{q_0, q_r, q_a\}$.

2. $F = Q - \{q_r\}$.

3. Let $p_1 ... p_n$ be an input of $\mathcal{M}_1$. The transition function δ is defined in the following way:

   (a) $\delta(q_0, p_1) = (p_1, \beta_1(p_1))$ if $\beta_0(p_1) = \beta_3(p_1) = 0$ and $\beta_1(p_1) \leq 2$, otherwise $\delta(q_0, p_1) = q_r$

(b) Given $i \leq n-1$, we have that $\delta\left(\left(p_i, k\right), p_{i+1}\right) = q_r$, whenever one of the following conditions is satisfied:

- $\left(p_i, p_{i+1}\right) \notin \mathcal{I}$.
- $k + \alpha_1\left(p_i, p_{i+1}\right) \geq 3$.

(c) Let $i \leq n-1$. If $\left(p_i, p_{i+1}\right) \in \mathcal{I}$ and $k + \alpha_1\left(p_i, p_{i+1}\right) \leq 2$, then

$$\delta\left(\left(p_i, k\right), p_{i+1}\right) = \left(p_{i+1}, k + \alpha_1\left(p_i, p_{i+1}\right)\right)$$

(d) If $\alpha_1\left(p_{n-1}, p_n\right) + k = 2$ and p_n does not contain horizontal edges, then

$$\delta\left(\left(p_n, k\right), \square\right) = q_a$$

(e) If either $\alpha_1\left(p_{n-1}, p_{n+1}\right) + k \neq 2$ or p_n contains horizontal edges, then

$$\delta\left(\left(p_n, k\right), \square\right) = q_r$$

Note that automaton $\mathcal{M}_1$ simply checks that $p_1 ... p_n$ encodes a sequence of compatible patterns, such that the number of one-degree nodes is equal to 2 and such that all the edges occurring in p_n are vertical edges located on its left column. It is easy (but tedious) to check that automaton $\mathcal{M}_1$ recognizes Ω_1. We have to estimate the computing time required to construct automaton $\mathcal{M}_1$. The computing time is upperbounded by $2^{O(r)}$, since the lookup tables can be computed in time $2^{O(r)}$ and $|Q| \in 2^{O(r)}$. Thus, we can compute $\mathcal{M}_1$ in time $2^{O(r)}$.

To finish with the proof we show that Ω_2 is a regular language. We give a very brief description of an automaton $\mathcal{M}_2$ recognizing Ω_2. We show that if we fix $n \geq 1$, we can use a regular automaton $\mathcal{M}_2$ to recognize the cyclic subgraphs of $[n] \times [r]$.

A *c-structure* is a subgraph of $[n] \times [r]$ constituted by a connected set (array) of vertical edges on the left, and two horizontal edges pointing to the right, one of them located on the bottom and the other one located on the top of the vertical array. We observe that:

1. Any cycle begins with a *c*-structure, that is: the leftmost pattern of a cycle necessarily contains at least one *c*-structure.

2. Any *c*-structure can be detected by $\mathcal{M}_2$, when the automaton is scanning the corresponding left-slice (pattern).

A *c*-structure is like an alert, which warn us of the possible emergence of a cycle. Also, we have to save information concerning the *c*-structures that haven been already

observed. To this end, we keep track of the evolution of the trajectories that arise from the endpoints of those c-structures. It is possible to keep track of the evolution of those trajectories given that:

1. Given w an input of $\mathcal{M}_2$ and given $t \leq |w|$, there are at most $\frac{r}{2}$ pairs of trajectories at time t, originated in previously observed c-structures and threatening of giving rise to cycles.

2. The only information that we have to save, in order to control the evolution of a pair of *dangerous* trajectories, are the y-coordinates of the nodes where those trajectories meet the right column of the left-slice being scanned.

Given i a positive integer lesser than $\frac{r}{2}$, we use the symbol P_i to denote the set

$$\{((a_1, b_1), ..., (a_i, b_i)) : \ \Phi\}$$

where Φ is the condition:

$a_1, ..., a_i, b_1, ..., b_i \in \{1, ..., r\}$ are pairwise different and $a_1 \lneqq b_1; ...; a_i \lneqq b_i$.

Let P be equal to the set of states of automaton $\mathcal{M}_2$, the set P is essentially equal to $\bigcup_{0 \leq i \leq \frac{r}{2}} P_i$. Let $p = ((a_1, b_1), ..., (a_i, b_i))$ be an element of P, and suppose that p is the state of $\mathcal{M}_2$ at time t. State p is informing us that i pairs of dangerous trajectories are exiting the left-slice being scanned at time t, through the pair of nodes $(a_1, b_1), ..., (a_i, b_i)$. The transition function of $\mathcal{M}_2$, denoted by ρ, is defined in such a way that it allows us to keep track of the evolution of those trajectories. Consider, as an example, the following situation:

1. The state of automaton $\mathcal{M}_2$, at time t, is equal to $((a_1, b_1), (a_2, b_2), (a_3, b_3))$.

2. $a_1 \lneqq b_1 \lneqq a_2 \lneqq b_2 \lneqq a_3 \lneqq b_3$.

3. The trajectory whose y-coordinate is equal to $a_1 \geq 2$, is extended with a vertical edge pointing down, and then with a horizontal edge.

4. The trajectory whose y-coordinate is equal to b_1 is extended with a vertical edge pointing up, and then with a horizontal edge.

5. The trajectory whose y-coordinate is equal to a_2 is extended with a vertical edge pointing down, and then with a horizontal edge. Moreover, we suppose that $a_2 - b_1 = 3$.

6. The trajectory whose y-coordinate is equal to b_2 is extended with a vertical edge pointing up, and then with an horizontal edge.

7. The remaining trajectories are extended with horizontal edges. Moreover, we suppose that $a_3 - b_2 \geq 2$.

8. A new c-structure is observed, it is constituted by two horizontal edges, whose y-coordinates are equal to $k+3$ and $k \gneqq b_3$, and by three vertical edges on the left joining the nodes located at heights k and $k+3$.

Then, given all this information, we have that the inner state of automaton $\mathcal{M}_2$, at time $t+1$, is equal to

$$\left(\left(a_1 - 1, b_1 + 1\right), \left(a_2 - 1, b_2 + 1\right)\left(a_3, b_3\right), \left(k, k+3\right)\right)$$

We identify an evolving c-structure with its corresponding tracking pair. Suppose that automaton $\mathcal{M}_2$ is scanning word w, and suppose that the pair (a,b) belongs to the inner state of automaton $\mathcal{M}_2$, at time t. There are three possibilities for pair (a,b), (at time $t+1$): pair (a,b) merges with another pair occurring at time t; pair (a,b) survives or pair (a,b) vanishes. We say that (a,b) vanishes a time $t+1$ if and only if w_{t+1}, the $t+1$-th pattern of the word being scanned, contains a chain of vertical edges joining the nodes located at heights a and b. If a pair vanishes, a cycle has been created (detected). On the other hand, if two pairs merge with each other, this merging gives rise to a cycle if and only the pairs are nested, that is: if pairs (a,b) and (c,d) merge with each other at time $t+1$, and the inequality $a \lneqq c \lneqq d \lneqq b$ holds, then a cycle has been created (detected). We use the term *pair-vectors* to denote the inner states of automaton $\mathcal{M}_2$, we note that the evolution of pairs defines an algebra of pair-vectors which, being finite, can be precomputed and encoded into the transition function of $\mathcal{M}_2$.

We have already discussed the key features of $\mathcal{M}_2$. At this point, the reader should be completely convinced that Ω_2 is a regular language, and that a finite state automaton recognizing Ω_2 can be computed in time $2^{O(r)}$.

We can now claim that a finite state automaton recognizing Ω_r can be computed in time $2^{O(r)}$. Thus, we have finished with the proof of the lemma. ■

2.1.3 An algorithm

Let 1^n be an instance of $\#SAW_r$. A self-avoiding walk contained in the lattice $[n] \times [r]$ can be encoded as a pattern chain $p_1 ... p_n$ of length n.

Let L_r be the language

$$\{p_1p_2..p_n : \psi(p_1p_2..p_n)\}$$

where $\psi(p_1p_2..p_n)$ is the sentence: the pattern chain $p_1p_2..p_n$ encodes a self-avoiding walk. Observe that $\#SAW_r(1^n)$ is equal to $f_{L_r}(1^n)$. If we want to show that $\#SAW_r$ can be solved in time $O\left(\log^2(n)\right)$ it is sufficient to show that one can compute in time $2^{O(r)}$ an unambiguous pushdown automaton $\mathcal{M}_r$ such that

$$f_{L_r}(1^n) = f_{L(\mathcal{M}_r)}(1^n)$$

Next lemma is straightforward

Lemma 8 *$p_1p_2..p_n$ belongs to L_r if and only if the following conditions are satisfied*

1. *For all $i \leq n-1$ we have that the pair (p_i, p_{i+1}) is an admissible pair.*
2. *$\beta_1(p_1) + \sum_{i \leq n-1} \alpha_1(p_i, p_{i+1}) = 2$ and the equations $\beta_0(p_1) = \beta_3(p_1) = 0$ hold.*
3. *The pattern chain $p_1p_2...p_n$ forbids the creation of cycles*

It follows from lemma 8 that the language L_r can be defined as

$$L_r = \{p_1p_2..p_n : \Psi_1 \wedge \Psi_2\}$$

where Ψ_1 is the conjunction of the first two constraints in the statement of lemma 8 and Ψ_2 is the third constraint.

Theorem 9 *L_r is a regular language and we can compute in time $2^{O(r)}$ a finite state automaton $\mathcal{M}_r$ recognizing L_r.*

Proof. It is time to put all the pieces together. We want to prove that

$$L_r = \{p_1...p_n \in \Sigma^* : \Psi_1 \wedge \Psi_2\}$$

is a regular language (recall that any regular language is an unambiguous context-free language). We define two languages H_1 and H_2, where given $i \leq 2$

$$H_i = \{p_1...p_n \in \Sigma^* : \Psi_i\}$$

We observe that $L_r = H_1 \cap H_2$. Recall that the intersection of a finite number of regular languages is a regular language as well. Also, it is sufficient to show that we can compute two finite state automata recognizing the languages H_1 and H_2. Lemma 7 estates that one can compute, in time $2^{O(r)}$, both automata. ■

Acknowledgment. This research was developed with the financial support of VIE-UIS.

References

[1] A. Bertoni, M. Goldwurm, N. Sabadini, The complexity of computing the number of strings of a given length in context free languages, *Theor. Comput. Sci.* **86** (1991) 325–342.

[2] A. Bertoni, M. Goldwurm, M. Santini, Random generation and approximate counting of ambiguously described combinatorial structures, *Proc. STACS Conf.* (2000) 567–580.

[3] M. Doi, S. Edwards, *The Theory of Polymer Dynamics*,Oxford Univ. Press, Oxford, 1986.

[4] P. Flory, *Principles of Polymer Chemistry*, Cornell Univ. Press, Itaca, 1953.

[5] I. Jensen, Enumeration of compact self–avoiding walks, *Comput. Phys. Commun.* **142** (2001) 109–113.

[6] D. Klein, Asymptotic distribution for self–avoiding walks constrained to strips, cylinders and tubes, *Jour. Stat. Phys.* **23** (1980) 561–586.

[7] D. Klein, F. Wall, Self–avoiding random walks on lattice strips, *Proc. Nat. Acad. Sci. USA (Chemistry)* **76** (1979) 1529–1531.

[8] M. Liskiewicz , M. Ogihara, S. Toda, The complexity of counting self–avoiding walks in two–dimensional grid graphs and in hypercube graphs, *Elec. Colloq. Comput. Compl.* **Report No. 61** (2001).

[9] D. Randall, A. Sinclair, Testable algorithms for self–avoiding walks, *Proc. SODA* (1994) 593–602.

[10] C. Thompson, *Mathematical Statistical Mechanics*, Princ. Univ. Press, Princeton, 1972.

[11] L. Valiant, The complexity of enumeration and reliability problems, *SIAM Jour. Comput.* **8** (1979) 410–421.

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