Applications of Schutzenberger-Bertoni Method: Counting Hamiltonian Structures

Francisco Gutierrez, J. Andres Montoya, Luis E. Zambrano

Universidad Industrial de Santander, Bucaramanga, Colombia,
juamonto@uis.edu.co

Abstract. We prove in this work that the tally counting problem consisting in computing the number of hamiltonian paths included in rectangular lattices of fixed height can be solved in $O(n)$ sequential time and $O(\log(n))$ parallel time

The qualitative properties of some models of statistical mechanics are completely encoded into a discrete function called *the partition function of the model* [5]. Solving a discrete model of statistical mechanics means computing its partition function. Most of the time partition functions are defined as counting problems [5]. In this work we study a *tally counting problem* closely related to *The Self-avoiding Walk Model* of statistical mechanics [5]. The partition function of the Self-avoiding walk model can be identified with the tally counting problem $\#SAW[\mathcal{L}_2]$, defined below

Problem 1. ($\#SAW[\mathcal{L}_2]$, counting self-avoiding walks in square lattices)

- *Input:* 1^n, *where* n *is a natural number.*
- *Problem: compute the number of self-avoiding walks contained in the square lattice of height* n.

In this work we study an important restriction of $\#SAW[\mathcal{L}_2]$, we study the tally counting problem consisting in computing the number of hamiltonian paths included in a given input lattice. We prove, using Schutzenberger-Bertoni method, that the counting of hamiltonian paths in rectangular lattices of fixed height can be solved in $O(\log(n))$ parallel time. Some similar results can be found in the literature: Stoyan and Strehl [4] (implicitly)proved that the counting of hamiltonian cycles in rectangular lattices of fixed height can be carried out in $O(\log(n))$ parallel time, they used *Schutzenberger method* [2] and a completely different encoding which can not be adapted to the counting of hamiltonian paths. Stoyan's encoding is based on *Jordan's Curve Theorem*, that is: Stoyan's encoding is based on the fact that hamiltonian cycles are closed paths, and closed paths are not self-avoiding.

1 Schutzenberger-Bertoni Method

In this section we introduce Schutzenberger-Bertoni Method a powerful counting technique that can be used to solve many tally counting problems.

Let f be a tally counting problem, let $p(X)$ be a polynomial function, let L be a context-free language and let f_L be the census function of L.

Definition 1. *We say that f is SB reducible to L via $p(X)$ if and only if for all $n \geq 0$ the equation $f(n) = f_L(p(n))$ holds.*

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

Theorem 1. *(Schutzenberger-Bertoni method)*
Suppose that f is SB reducible via $p(X)$ to an unambiguous context-free language

1. *Problem f can be solved in time $O(p(n))$.*
2. *Problem f can be solved in time $O(\log(n)\log\log(n))$ using a polynomial number of processors.*
3. *If f is SB reducible to a regular language, then f can be solved in $O(\log(n))$ parallel time.*

Last theorem was proved by Bertoni, Goldwurm and Sabadini [2]. Theorem 1 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 last theorem. In next section we use Schutzenberger-Bertoni method to prove that the counting of hamiltonian paths in rectangular lattices of fixed height belongs to FP.

2 Counting hamiltonian paths in rectangular lattices

$\#SAW[\mathcal{L}_2]$ is a very important problem because of its many applications in polymer thermodynamics, percolation theory, social choice etc [5]. In this work we study an important restriction of $\#SAW[\mathcal{L}_2]$. Let r be a natural number bigger than 0, we use the symbol $\#ham[r]$ to denote the tally counting problem defined by

Problem 2. ($\#ham[r]$; *counting hamiltonian paths in rectangular lattices of height r*)

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

We prove the following theorem

Theorem 2. *Let r be a positive integer*

1. *Problem $\#ham[r]$ can be solved in time $O(n)$.*
2. *Problem $\#ham[r]$ can be solved in time $O(\log(n))$ using a polynomial number of processors.*

We observe that, to obtain a proof of theorem 2, it is sufficient to show that if we fix $r \geq 1$, we can compute in time $2^{O(r)}$ a finite state automaton $\mathcal{M}_r$ satisfying the following condition:

$$\text{for all } n \geq 1 \text{ we have that } \#ham\left(n,r\right) = f_{\mathcal{M}_r}\left(n\right)$$

where $\#ham\left(n,r\right)$ denotes the number of hamiltonian paths included in the rectangular lattice $[n] \times [r]$ and $f_{\mathcal{M}_r}$ denotes the census function of $L\left(\mathcal{M}_r\right)$, the language recognized by $\mathcal{M}_r$. The proof of this fact is based on an encoding of hamiltonian paths as words over a suitable alphabet, the key features of such an encoding are the following ones:

1. The language

   $$L_r = \{w : w \text{ is the encoding of a hamiltonian path in } [n] \times [r] \text{ for some } n\}$$

   is regular.
2. Given $n \geq 1$, all the hamiltonian paths included in $[n] \times [r]$ are encoded as words of length n.

From now on we fix $r \geq 1$. A *Bilateral slice (block)* is a lattice graph isomorphic to $\{1,2\} \times [r]$. *Left slices* are the subgraphs of fundamental slices that can be obtained by eliminating all the edges of the form $\{(2,x),(2,x+1)\}$. It is clear that $[n] \times [r]$ can be partitioned into a finite set of left slices and a single bilateral slice. *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).

We note that given γ, a hamiltonian path in the rectangular lattice $[n] \times [r]$, path γ is the concatenation of a sequence $p_1 ... p_n$ of n patterns such that p_n does not contain horizontal edges. Also, we can try to codify hamiltonian paths as words, whose characters are patterns.

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

Last proposition follows from the fact that any pattern is a subset of the set of edges of a left slice, last proposition implies that we can enumerate the set of patterns in time $2^{O(r)}$.

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 located in $\{2\} \times [r]$ will be called the *core* of pq (or the 2-fiber of pq), and will be denoted with the symbol $\varsigma\left(pq\right)$. We use the symbol $\mathcal{P}_L$ to denote the set of patterns. 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:

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

where $\deg_{pq}\left(v\right)$ denotes the degree of v as a node of the graph pq. We define three analogous functions $\beta_0, \beta_1, \beta_3 : \mathcal{P}_L \rightarrow \mathbb{N}$ which count the number of degree-zero, degree-one and degree-three nodes (respectively) located on the 1-fiber of the left-pattern being evaluated. We note that we can compute in time $2^{O(r)}$

the tables of each one of the functions $\alpha_0, \alpha_1, , \alpha_3, \alpha_4, \beta_0, \beta_1$ and β_3. Given p, q two patterns, we say that pair (p, q) is *admissible* if and only if the conditions $\alpha_1(p,q) \leq 2$ and $\alpha_0(p,q) = \alpha_3(p,q) = \alpha_4(p,q) = 0$ are satisfied. We use the symbol $\mathcal{I}$ to denote the set of admissible pairs, and we note that $\mathcal{I}$ can be computed in time $2^{O(r)}$.

A spanning subgraph of $[n] \times [r]$, say γ, is a hamiltonian path if and only if γ is connected, there exist exactly two degree-one nodes in γ and the remaining nodes are degree-two nodes. We note that the *degree constraint* encodes a local property which can be easily checked by a finite state automaton. On the other hand, we have that the *connectivity constraint* is global, and it is not clear if we can check it using a finite state automaton. Note that: given γ a spanning subgraph of $[n] \times [r]$, subgraph γ is a hamiltonian path if and only if γ is acyclic, there exist exactly two degree-one nodes in γ and for all $v \in V([n] \times [r])$ the condition $1 \leq \deg_\gamma(v) \leq 2$ holds, (where $\deg_\gamma(v)$ denotes the degree of v as a node of γ).

Lemma 2. *A sequence of patterns $p_1 p_2 ... p_n$ encodes a hamiltonian path in $[n] \times [r]$ if and only if:*

1. *All the edges occurring in p_n are vertical edges located on its left column and for any $i \leq n-1$ the pair (p_i, p_{i+1}) is admissible.*
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. *$p_1 p_2 ... p_n$ forbids the creation of cycles, (i.e. the graph γ, encoded by the chain $p_1 p_2 ... p_n$, is an acyclic graph).*

The proof of last lemma is quite easy. Given $n \geq 2$, a *n-pattern chain* is a sequence $p_1 p_2 ... p_n$ satisfying the conditions imposed in the statement of the above lemma. Also, we have that there exists a bijection between the set of n-pattern chains and the set of hamiltonian paths included in $[n] \times [r]$.

We are ready to define our encoding of hamiltonian paths, to this end we fix the alphabet Σ defined by

$$\Sigma = \{p : p \in \mathcal{P}_L\}$$

Let L_r be the language

$$\{p_1 ... p_n \in \Sigma^* : p_1 ... p_n \text{ is a } n\text{-pattern chain}\}$$

Next lemma is the key of our proof, (a *detailed proof is included in the appendix*).

Lemma 3. *L_r is a regular language. Furthermore, we can compute in time $2^{O(r)}$ a finite state automaton $\mathcal{M}_r$ recognizing L_r.*

Theorem 2, the main theorem of this section, is an easy corollary of lemma 3.

3 Concluding remarks

We have proven that for all $r \geq 1$ the problem $\#ham[r]$ can be solved in $O(\log(n)\log\log(n))$ parallel time. It is important to remark that the linear time tractability of $\#ham[r]$ could be obtained via a general theorem of Arnborg, Lagergren and Seese [1]. It is not the case with the parallel algorithm derived by means of Shutzenberger-Bertoni method. Furthermore, many similar results can be obtained by means of Schutzenberger-Bertoni method and a *block-coding* similar to the one used in the proof of theorem 2. So, for example, we can prove that each one of the following problems can be solved in $O(\log(n)\log\log(n))$ parallel time: the counting of hamiltonian cycles in rectangular lattices of fixed height, the counting of self-avoiding walks in rectangular lattices of fixed height, the counting of matchings in rectangular lattices of fixed height, the counting of polyominoes and animals when restricted to rectangular lattices of fixed height [3].

Acknowledgement. This research was developed with the financial support of VIE-UIS and Colciencias research project 111518925292.

References

[1] S. Arnborg, J. Lagergren, D. Seese. Easy problems for tree-decomposable graphs. Journal of Algorithms 12 (2): 308-340, 1991.

[2] A. Bertoni, M. Goldwurm, N. Sabadini. The Complexity of Computing the Number of Strings of a Given Length in Context Free Languages. Theoretical Computer Science 86:325-342, 1991.

[3] F. Gutierrez, J. A. Montoya, L. Zambrano. Applications of Schuzenberger-Bertoni method: Counting Polyominoes. Submitted LAGOS 2011.

[4] R. Stoyan, V. Strehl. Enumeration of Hamiltonian Circuits in Rectangular Grids. Journal of Combinatorial Mathematics and Combin. Computing, 21:197-127, 1996.

[5] D. Welsh. *Complexity: Knots, Colourings and Counting.* Cambridge University Press, Cambridge (UK), 1993.

APPENDIX: Proof of lemma 3

Proof. We want to prove that the language

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

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. $\beta_L(p_1) + \sum_{i \leq n-1} \alpha_1(p_i, p_{i+1}) = 2$ 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 the third constraint. We define two languages

$$L_1 = \{p_1...p_n \in \Sigma^* : \Psi_1\}$$

and

$$L_2 = \{p_1...p_n \in \Sigma^* : \Psi_2\}$$

We observe that $L_r = L_1 \cap L_2$. Recall that the intersection of two regular languages is regular. Also, it is sufficient to show that both languages, L_1 and L_2, are regular. First at all we show that L_1 is a regular language, and we also show that a finite state automaton $\mathcal{M}_1$ recognizing L_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 will be used by $\mathcal{M}_1$ as lookup tables: these objects will be incorporated in 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 $\delta((p_i, k), p_{i+1}) = q_r$, whenever one of the following conditions is satisfied:
   - $(p_i, p_{i+1}) \notin \mathcal{I}$.
   - $k + \alpha_1(p_i, p_{i+1}) \geq 3$.

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

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

(d) If $\beta_1(p_n) + k = 2$ and p_n does not contain horizontal edges, then

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

(e) If either $\beta_1(p_i) + k \neq 2$ or p_i contains horizontal edges, then

$$\delta((p_i, k), \square) = q_r$$

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 L_1. We have to estimate the computing time required to construct automaton $\mathcal{M}_1$. This 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 L_2 is a regular language. We give a very brief description of an automaton $\mathcal{M}_2$ recognizing L_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 (character).

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

$$\left\{((a_1, b_1), ..., (a_i, b_i)) : i \leq \frac{r}{2} \ \& \ \Phi\right\}$$

where Φ is the condition:

$a_1, ..., a_i, b_1, ..., b_i \in \{1, ..., r\}$ are pairwise different and $a_1 \lneqq b_1; a_2 \lneqq b_2; ...; a_i \lneqq b_i$.

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 $a_2 - b_1 = 2$.
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 $k \gneqq b_i$, 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

$$((a_1 - 1, b_2 + 1), (a_3, b_3), (k, k + 3))$$

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 $a \lneqq c \lneqq d \lneqq b$ 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 L_2 is a regular language, and that a finite state automaton recognizing L_2 can be computed in time $2^{O(r)}$.

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