ELSEVIER
Theoretical Computer Science 215 (1999) 225–237
Theoretical Computer Science

On the complexity of deadlock detection in families of planar nets

Bruno Durand[a,*], Anne-Cécile Fabret[b]

[a] *Laboratoire de l'Informatique du Parallélisme, ENS-Lyon CNRS, 46 Allée d'Italie, 69364 Lyon Cedex 07, France*
[b] *Laboratoire de Recherche en Informatique, Université Paris Sud, Bat. 490, 91405 Orsay Cedex, France*

Received October 1995; revised June 1997
Communicated by M. Nivat

Abstract

We are interested in some properties of massively parallel computers which we model by finite automata connected together on a 2-dimensional grid. We wonder whether it is possible to anticipate a possible appearance of a deadlock in such nets. Thus, we look for efficient algorithms to predict whether deadlocks can appear in grids of bounded size. From the point of view of worst-case complexity, we prove that this problem is NP-complete whereas it is quadratic for linear structures. The method we use is a reduction from a tiling problem.

We also prove that this problem, associated with a natural probability distribution on its instances, is RNP-complete (Random NP-complete) in the theory proposed by Levin and developed by Gurevich. Very few randomized problems are known to be RNP-complete. Under classical complexity hypotheses, this result proves that there does not exist any algorithm that solves this problem efficiently on the average case. We present other extentions of our results for different planar underlying communication graphs, and we present a Σ_2-complete problem for networks with inputs.

1. Introduction

In this paper, we are interested in some properties of massively parallel computers. These computers use a great number of processors connected together by a communication network. A large class of such machines is built in a modular way by connecting identical cards together. Hence, we can obtain infinitely many different machines from copies of a given card. These machines may only differ by the number of cards and by the topology underlying their connections. We model these machines by a 2-dimensional grid on the vertices of which we put identical copies of a nondeterministic finite automaton. These copies may communicate with each other through the edges of the grid.

* Corresponding author. Tel.: +33 472 72 85 01; fax: +33 472 72 80 80; e-mail: bdurand@ens-lyon.fr.

0304-3975/99/$ – see front matter
PII: S0304-3975(97)00185-0

We are mainly interested in the anticipation of a possible appearance of deadlock in such nets. Fabret and Petit proved in [9] that the deadlock problem is undecidable on grids: there is no algorithm that, given a finite automaton, decides whether all grids of size $n \times n$ constructed with this automaton, are deadlock free (see also [8]).

These undecidability results are a first approach to the study of the behavior of these nets; we propose here another point of view on this study: we study the algorithmic complexity of the appearance of deadlocks. Thus, we wonder if there exist efficient algorithms to predict whether deadlocks can appear in grids of bounded size. As far as worst-case complexity is concerned, our main result is the NP-completeness of this problem (better formalized in Section 3). This result can be compared with quadratic algorithms on linear structures and with particular cases studied in [1]. In order to prove this result, we construct a reduction from a tiling problem.

Even if it is interesting to know that a problem is NP-complete, this kind of result does not imply that the considered problem is practically difficult to solve since there may exist an algorithm efficient on the average to solve them. When a "natural" probability is defined on the set of instance, one obtains a randomized problem. Once randomized, many NP-complete problems are polynomial in the average case. We prove that under classical complexity hypotheses, our deadlock problem is difficult in the average. More precisely, we prove that it is RNP-complete (Random NP-complete) in the theory proposed by Levin in [13] and exposed by Gurevich in [12]. Very few randomized problems are known to be RNP-complete because conditions needed are very strict and because it is difficult to handle the notion of polynomial reductions between randomized problems. We give a hint of the proof of our RNP-completeness result in Section 5.

Our basic construction (presented in Section 3) can be slightly transformed to get analogous results for other models. A weaker version of our work is that our theorems remain true for n-dimensional grids where $n \geqslant 2$. In Section 4 we present some other possible communication graph such as torus, or hexagonal tessellations of the plane. Our result is unchanged on these structures. We also study other models where networks make their computations on an input which is given on the bottom row of the grid. In these cases, our problems are more difficult, since they are Σ_2 or Π_2 complete in the polynomial hierarchy.

The paper is organized as follows: in Section 2, we explain our computational model and recall briefly the theory of tilings. Then, we explain in Section 3 our reduction and give a detailed proof of our main result. Section 4 is devoted to extensions of our result to other models, and Section 5 deals with average case complexity.

2. Models

We present below our formalization for nets of automata. We discuss a few variants and explain how they interact with our result in order to justify the robustness of our work. We then present briefly tile sets, which will be extensively used in our proofs.

2.1. Nets

Throughout this paper we consider models of parallel computers, the processors of which are placed on a grid (or on a torus, see also Section 4). We model the elementary machine by a nondeterministic communicating finite automaton. Given such a finite automaton $\mathscr{A}$, we shall denote by $\mathscr{G}_{\mathscr{A}}(n)$ the grid of size $n \times n$ formed by copies of $\mathscr{A}$ connected to their 4 neighbors. We need now to define more precisely the communication model. Each internal node of the grid has 4 neighbors (in the directions `N, S, E, W`), and thus 4 channels of communication; it can recognize which channel is used. These channels are two-ways and the automaton may perform in parallel an emission and a reception of a message. All communications are supposed to synchronize two neighbor cells, since the emission of a message does not end before the message is received, and reciprocally, if an automaton tries to receive a message, then it waits until its reception. Thus we consider two communication primitives:

- `emit(D,a)` which means that the value a is emitted in the direction D. As we only consider finite state automata, a belongs to a finite set and thus we could also consider that, between two cells, there exists a finite number of channels on which signals are emitted without carrying any value. For the sake of simplicity, we chose to consider that a message is emitted.
- `receive(D,x)` which means that the value emitted by the neighbor located in the direction D will be associated to the formal variable x. In the following, x can be tested by the automaton (e.g. `if x=a then ... else ...`).

Beware that if an automaton successively performs `emit(N,a);receive(N,x)` while its North neighbor performs `emit(S,b);receive(S,y)` then a deadlock arises: the computation is stopped forever. If one of these two automata had first tried to receive and then to emit, the computation would go on without deadlock. On the other hand, automata may perform communications in parallel (channels are two-ways). Thus `emit(N,a) // receive(N,x)` and `emit(S,b) // receive(S,y)` do not create a deadlock.

Border cells of a grid do not have 4 neighbors. We assume that, when they send an information in a "wrong" direction, then the information is lost and the communication terminates. When they wish to receive an information from this direction, we consider that it receives immediately a special message ω. This message is reserved for communication at the borders. Thus automata can test whether they are located on a border or a corner of the grid.

A global clock is not needed. We consider that there exists a local time for each automaton and thus, they are synchronized only by communications. The issue of synchonicity is often discussed in the case of cellular automata but it is of no great interest in our case.

2.2. Tilings

A tile is a square the sides of which are colored. Colors belong to a finite set C called the *color set*. A set of tiles τ is a subset of C^4. All tiles have the same (unit)

size. A tiling of the plane is *valid* if and only if all pairs of adjacent sides have the same color. Note that it is not allowed to turn tiles. Berger proved in 1966 that given a tile set, it is undecidable whether this tile set can be used to tile the plane [3]; a simplified proof was given in 1971 by Robinson [16].

We can also define *finite tilings*. We assume that the set of colors contains a special "blank" color and that the set of tiles contains a "blank" tile i.e. a tile whose sides are all blank. A finite tiling is an *almost everywhere blank* tiling of the plane. If there exist two integers i and j such that all the non-blank tiles of the tiling are located inside a rectangle of size $i \times j$, then, we say that the size of the finite tiling is lower than $i \times j$. Note that inside the $i \times j$ rectangle, there can be blank and non-blank tiles. If there is at least one non-blank tile, then the tiling is called *non-trivial*.

Another undecidability result can be proved simply by using a construction presented by Robinson in [16] which reduces the undecidability of the halting problem for Turing machines into the following: Given a tile set with a blank tile, it is undecidable whether this tile set can be used to form a valid finite non-trivial tiling of the plane.

In the following, we are mainly interested in complexity results; we shall construct our reductions from the following problem, proved NP-complete by Lewis in [14]:

FINITE-TILING

Instance: A finite set C of colors ($|C| = c$) with a blank color. A tile set $\tau \subset C^4$ containing the blank tile.

Question: Does a finite non-trivial tiling by τ of the plane of size lower than $c \times c$ exists?

As usual in complexity theory, we can change the bound of the size of the tiling and thus ask for a $(P(c) \times P(c))$ tiling where P is any polynomial. We present in Section 5 an average case complexity result concerning tilings.

3. Our complexity result

We focus in this paper on the deadlock problem: given an automaton $\mathcal{A}$, can we verify that any net $\mathcal{G}_{\mathcal{A}}(n)$ is deadlock free? If we assume that parallel computers consist of a grid of Turing machines that may communicate, then with only two connected Turing machines, the deadlock problem is obviously undecidable. We prefer to work on models for SIMD computers rather than for MIMD. We also impose that the net is uniform, i.e. that all nodes contain a copy of the same machine. Of course, our results are more interesting and difficult when the model is more restrictive. They easily generalize to less restrictive models, and even to MIMD machines. Fabret and Petit proved in [9] that the following deadlock detection problem is undecidable:

Instance: A finite automaton A.

Question: Does an integer n exist such that the grid $\mathcal{G}_{\mathcal{A}}(n)$ may lead to a deadlock?

Our work consists of a complexity study on the deadlock detection in planar families of nets and is inspired by the above result. Although the reduction presented in [9] used the halting problem of Turing machines, ours is based on tilings (as in [8]) and can straight away prove their undecidability theorem. We present below our main results:

DEADLOCK

Instance: A nondeterministic finite automaton $\mathscr{A}$ (a denotes the number of its states), two integers $n < a$ and $t < a$.

Question: Can a deadlock appear in the grid $\mathscr{G}_{\mathscr{A}}(n)$ before t time steps?

DEADLOCK-FREE

Instance: A nondeterministic finite automaton $\mathscr{A}$ (a denotes the number of its states), two integers $n < a$ and $t < a$.

Question: Can the grid $\mathscr{G}_{\mathscr{A}}(n)$ compute without deadlock during at least t time steps?

Theorem 1. DEADLOCK *and* DEADLOCK-FREE *are* NP-*complete.*

The rest of this section is devoted to the proof of this theorem. We first prove that DEADLOCK is NP-complete and we then explain how to modify the reduction in order to obtain the NP-completeness of DEADLOCK-FREE.

In order to prove our result, we first consider an instance of FINITE-TILING, i.e., a tile set τ with a blank tile ($\tau \subset C^4$ where C is a set of c colors). Then we transform this tile set into a nondeterministic finite automaton $\mathscr{A}_\tau$ and two integers n_τ and t_τ such that a deadlock can appear in a grid $\mathscr{G}_{\mathscr{A}}(n_\tau)$ before t_τ time steps if, and only if there exists a finite non-trivial tiling of the plane by τ of size lower than $c \times c$. As it would be very long and tedious to give a formal specification of $\mathscr{A}_\tau$, we present it in the form of a nondeterministic algorithm depending on τ. The reader can imagine that this algorithm is performed in parallel in all cells of a grid. In this algorithm we denote the North, South, East, and West neighbors by the letters N, S, E, and W. Some special states of the automaton are denoted by a color (*black*, *red*, *pink*, *white*, and *green*). For the sake of simplicity we assume that these states are ordered: *black* > *green* > *white* > *red* > *pink*. These colors are assumed not to belong to the color set of the tile set τ.

The algorithm performed by $\mathscr{A}_\tau$.

Initialization.

Init-1: Communicate with your 4 neighbors in order to determine whether you are located in a (N, S, E, or W)-border, a (NE, SE, NW, or SW)-corner, or a central cell.

Init-2: Choose nondeterministically a tile out of τ to be associated with. If you are in a North border, then you must choose a tile the North side of which is blank (resp. for other borders). If you are in a North–East corner, then you must choose a tile the North and the East sides of which are blank (resp. for other corners).

Init-3: Look at your neighbor's associated tiles. If the side colors of your associated tile does not match the side colors of your neighbor's tiles, then enter the special

black state. Otherwise, the matching condition is locally verified for associated tiles, thus enter the special *white* state if your associated tile is the blank tile, and the special *green* state otherwise.

Init-4: If you are in a SW corner then enter a *red* state.

Loop. According to your 4 neighbors' associated states, repeat forever the following "loop" steps (diffusion of colors):

Loop-1: This step only concerns West borders. If one of your neighbors is in the *pink* state, then enter the *pink* state too, otherwise if one of your neighbors is in the *red* state, then enter the *red* state too.

Loop-2: If you are in the NW corner, and if one of your neighbors is in the *red* state, then enter the *pink* state.

Loop-3: If you are in the SW corner, if your North neighbor is in the *pink* state, and if your East neighbor is in the *green* state, then jump to step "Gen-deadlock".

Loop-4: (general case) If you are neither in a West border nor in a NW nor a SW corner, then enter the greater of all your neighbors' states including yours ($black > green > white > red > pink$).

Deadlock generation.

Gen-deadlock: Generate a deadlock with your East neighbor.

We have decided not to give an exact description of the automaton that corresponds to this algorithm. It is almost everywhere very easy. The only non-trivial part is the deadlock generation which is rather straightforward: the corner sends to its East neighbor a special message which means "make a deadlock with me", and then one of them executes `give a; take b` and the other `give b; take a`.

Proof of Theorem 1. We first prove that both problems (DEADLOCK and DEADLOCK-FREE) are in NP. Then we explain the propagation of colors in our net and prove a lemma on this. We use it to construct our reduction that we finaly prove polynomial.

Both problems are in NP since a nondeterministic Turing machine can simulate any possible computation of the grid within a time polynomially bounded by the size of $\mathscr{A}$. The time limit t that is imposed in our problems, is not useful in our reductions since it will be clear at the end of the proof that if no deadlock appears at time $2n+4$, then there will never be any deadlock in the net. The reason for this bound is to ensure that the problems belong to NP.

Propagation of colors. Consider a grid $\mathscr{G}_{\mathscr{A},\tau}(n)$. After the step "Init-1", each cell knows whether it is a border or a corner of the grid. After "Init-2", a tile of τ is associated to each cell, and the colors of the border sides of the grid are blank. Note that the choice for a tile on a border or on a corner cell is never empty since the tile set τ contains the blank tile that can always be chosen. Then after "Init-4", the associated colors are *red* in the SW corner, *black*, *white* or *green* everywhere else. If there is a tiling mismatch in one tile, then it is *black*. It is *white* if the associated tile is blank,

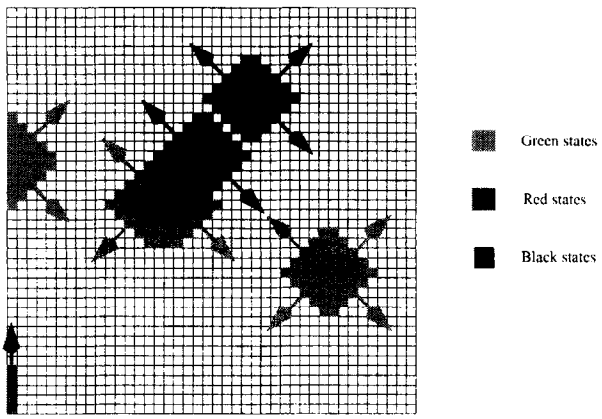

Fig. 1. The color behavior in the reduction.

else it is *green*. Since a tiling error concerns at least two tiles, then a *black* cell has always a *black* neighbor, except if the error concerns the SW corner which is *red*.

As all cells enter the same loop and are locally synchronized by the communication of the beginning of the step "Loop", then we shall use as local time the number of iteration of this loop.

From now on, only colored states will be considered in the algorithm and not the associated tiles. Let us now focus our attention to West border cells. Time step after time step, the *red* color propagates along the border until it reaches the NW corner. This corner is transformed in *pink* and this color propagates back on the West border until it arrives to the North of the SW corner $2n$ time steps later. Hence this column of cells acts as a time counter.

The behavior of the grid is illustrated by Fig. 1. Let us observe now all other cells of the grid (not in the West border). In this area, the *black* color propagates in all directions, whatever the surrounding colors may be. In a *white* surrounding, the *green* color propagates too, but if it is in competition with a *black* cell, then the *black* wins because the set of colors is ordered such that $black > green > white$. These cells are not influenced by the *red* and the *pink* of the West column since $white > red > pink$. Hence if there is a *black* cell in this area, it will spread all over the grid (except the west column). If there is no *black* cell, then the *green* spreads.

Lemma 1. *Consider the tiling obtained on the grid after step "Init-2". After* $2n$ *time steps, the right neighbor of the SW corner is* black *if, and only if there is a tiling error; it is* white *if the tiling is everywhere blank, it is* green *if the tiling is valid and non-trivial.*

Proof of Lemma 1. Let us call the right neighbor of the SW corner the "special" cell. Assume that after step "Init-2" there is somewhere a tiling error. If the error is not in the West border, then it is clear that the *black* color will spread in the grid. The distance between the special cell and all other cells of the grid is $2n - 1$. Thus this

cell becomes *black* before the step $2n$. Note also that the farther cell is the NE corner, and if this corner is *black*, then so is one of its neighbors that is 1 step nearer. If after step "Init-2", the error is on the left border, then it will reach the second West column before being erased with the propagation of the *red* and *pink* colors. Thus the special cell is "informed" on time of any tiling error.

Assume now that the tiling is correct. Then, the *green* color will spread in the grid exactly as was spreading the *black* in the previous case. Thus after $2n$ steps, the special cell is green if and only if the tiling is valid and non-trivial. □

Reduction. Let us prove now that our construction is a reduction of FINITE-TILING into DEADLOCK. The automaton $\mathscr{A}_\tau$ has just been constructed, we choose $n_\tau = c$ for the size of the grid (c is the number of colors), we choose a time $t_\tau = 2c + 4$. Imagine that a finite non-trivial tiling of the plane of size $c \times c$ exists. Then a possible evolution for the grid $\mathscr{G}_{\mathscr{A}_\tau}$ is that it chooses the tiles of this tiling. Then, after at most $2n$ time steps in the loop and at most 2 more communications for the initialization process, the special cell is *green*, the SW corner is *pink*, hence after 2 more communications, a deadlock appears between these two cells.

Reciprocally, if a deadlock appears in the net, it can only appear between the SW cell and the special one, since obviously the loop cannot generate a deadlock. Hence, the SW corner has a *pink* North neighbor and a *green* East one (the special cell). The North neighbor cannot be pink after less than $2n$ steps (see above). Thus if there had been a tiling error in the associated tiles, the special tile would have had time to become black. Hence the tiling formed by the chosen tiles is valid. It is also non-trivial, otherwise the special cell would have remained *white*. The obtained tiling is bordered by blank sides because the border cells have chosen their tiles under this restriction. Hence we can extend the tiling obtained on the grid into a finite non-trivial tiling of the plane of size at most $c \times c$.

The last point to prove is that the reduction is polynomial. The construction of $\mathscr{A}_\tau$ does not require any difficult computation, and its size is clearly a polynomial in the number of tiles which is itself bounded by c^4.

Concerning the NP-completeness of DEADLOCK-FREE, we have just to change step "Loop-3" and replace it by "Loop-3-bis". It is clear that a finite non-trivial tiling of the plane by τ exists iff there exist a deadlock free computation by $\mathscr{G}_{\mathscr{A}_\tau}$.

Loop-3-bis: If you are in the SW corner, if your North neighbor is in the *pink* state, and if your East neighbor is either in the *black* state or in the *white* state, then jump to step "Gen-deadlock". □

4. Improvements

4.1. Other nets

The condition that automata should be placed on a grid may seem very restrictive since it is possible to construct other regular planar structures of communication. For

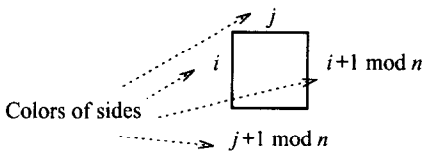

Fig. 2. Tiles of λ.

instance, hexagonal tessellations of the plane may be considered. But we can generalize our result to any archimedean net. An archimedean net is a planar structure, forming a regular tiling of the plane. These tilings have been first studied by Kepler in the 16th century. If copies of the same automaton are placed on the nodes of such structures, then a strong equivalence result between all these structures and the grid has been proved by Róka in [17]. As a corollary, our results still hold.

We also have the same result if the grid is replaced by a torus. In this case, there are no borders and all cells are identical. The tiling problem on a torus is NP-complete but references proposed by [10] are not correct. A proof can be found in [6]. The construction of our automaton has to be modified according to the following idea: we shall make all cells to elect a "pseudo-corner" that will act as the SW corner of our basic reduction. In order to do that, each cell will choose nondeterministically a tile of τ *and* a tile of a special tile set λ defined in Fig. 2, for which we call "position" of a tile of τ_2 the pair (i,j). Note that this tile set depends on the size of the net. Now our algorithm is very similar to the proof of Theorem 1 beginning at step "Init-3". The differences are that both the tiling components corresponding to τ and to λ are checked; the cell $(0,0)$ in the tiling λ will act as a the SW corner, and the border cells will be the $(0,i)$-cells of the λ component. The NW corner will be the cell $(0,n)$. The end of the proof is exactly the same.

4.2. *Nets with inputs and a Σ_2-complete problem*

It is often natural to model the behavior of a parallel computer by a net of automata with an input given to the first row of the grid. In our first model, there was no input. Now we consider an input as a word over the binary alphabet, and at the beginning of the computation, its first bit is sent to the leftmost cell of the first row, the second bit to the second cell, etc. For this model, we give complexity results on the two following deadlock problems with inputs. Beware that the second problem is not the negation of the first one.

DEADLOCK-2

Instance: A nondeterministic finite automaton $\mathcal{A}$ (a denotes the number of its states), two integers $n<a$ and $t<a$.

Question: Is it true that for all input word w, $|w|<n$, there exists an integer k, $|w|<k<n$ such that the net $\mathcal{G}_{\mathcal{A}}(k)$ may lead to a deadlock?

DEADLOCK-FREE-2

Instance: A nondeterministic finite automaton $\mathscr{A}$ (a denotes the number of its states), two integers $n<a$ and $t<a$.

Question: Does an input word w exists ($|w|<n$) such that all nets $\mathscr{G}_{\mathscr{A}}(k)$, $|w|<k<n$ lead to a deadlock?

Theorem 2. DEADLOCK-2 *is* Π_2*-complete and* DEADLOCK-FREE-2 *is* Σ_2*-complete in the polynomial hierarchy.*

In order to prove this result, we need to come back to tilings. The following problem is Σ_2-complete in the polynomial hierarchy (see [18]).

FINITE-TILING-2

Instance: A finite set C of colors ($|C|=c$) with a blank color, and a tile set $\tau \subset C^4$ containing the blank tile.

Question: Does a correctly tiled row of at most c tiles exists (non-blank but with blank sides below and on both ends) such that it cannot be extended in a finite non-trivial tiling of the plane?

Proof of Theorem 2. The reduction is almost the same as for Theorem 1. We reduce FINITE-TILING-2 into DEADLOCK-FREE-2 using the second part of Theorem 1 (see the end of the proof). Concerning DEADLOCK-2, we start from the negation of FINITE-TILING-2 with the first reduction of Theorem 1. The only change needed is that the choices of tiles in the bottom row will be made accordingly to the input word. □

These result can be generalized when the input is given bit after bit on the leftmost cell of the bottom row. The needed transformation can be found in [15].

5. Average case complexity results

After Cook and Levin's presentation of NP-completeness theory in 1971, many problems have been proved NP-complete. In order to cope with these problems, people use to look for algorithms that are not polynomial in the worst case, but which are polynomial for most instances. This led to the notion of average-case polynomial algorithms (AP); to define this notion, a probability function μ is defined on the set of instances. Very soon, it seemed that, if for some NP-complete problems it was possible to find such average polynomial time algorithms (hence that these problems belong to AP), it was a very difficult task for others.

The goal of this section is to prove that the deadlock detection problem in families of planar nets is not only difficult in the worst case but also in the average. Thus in this section, we use the notion of randomized problem where a probability distribution μ is associated to the inputs. The complexity class "Random NP"

(RNP for short) was introduced by Leonid Levin in 1986 [13]. He proposed a notion of reduction from a randomized problem into another which allowed him to present an RNP-complete problem in the same paper. These notions have been discussed in more details and some other problems have been proved RNP-complete in [11, 12, 4, 5].

If an RNP-complete problem were in AP, then such would be all problems of RNP, hence RNP would be included in AP which is very unlikely. It would imply that problems solved in non-deterministic exponential time (NEXP) can be solved in deterministic exponential (DEXP) time (see [12]). Furthermore, there exists a subclass of problems in RNP (called *flat* problems) that cannot be complete unless NEXP and DEXP are equal (see [2]). A problem is *flat* if $\mu(x) \leqslant 2^{-n^\alpha}$ $(\alpha > 0)$ for all instances x of sufficiently large size n. This class of problems includes most probabilistic graph problems and many others, such as a randomized version of SAT. In most NP-complete problems, the number of positive and negative instances for inputs of great size are not comparable. If a problem is RNP-complete, then there are roughly as many positive and negative instance for inputs of a given size.

We do not present here the theory of RNP-completeness. A short introduction can be read in [7]; the reference paper is [12]. The basic idea is that a reduction between randomized problems should not diminish too much the probability of a given instance. A sufficient method for proving the RNP-completeness of a problem is the following:

1. Prove that your problem is in NP and that its probability distribution is P-time computable.
2. Take a RNP-complete problem with a probability function μ_1.
3. Consider this problem only as an NP-complete problem and reduce it polynomially to your problem: your problem is proved NP-complete. Call the reduction function f.
4. Prove that the image of μ_1 by f is P-dominated by the probability function of your problem.

To understand the previous method, we need the following definition:

Definition 1. Assume that A is our instance set, μ_1, μ_2 two probability distributions on A, and that there exists a polynomial function p such that

$$\forall x \in A, \quad \mu_2(x) . p(|x|) > \mu_1(x).$$

Then we say that μ_2 P-dominates μ_1.

A function f from a set A to a set B transform the probability distribution μ into ν iff $\nu(y) = \sum_{f(x)=y} \mu(x)$.

The following tiling problem has been proved RNP-complete by Leonid Levin in 1986 [13]:

RAND-FINITE-TILING

Instance: A finite set C of colors $(|C| = c)$ with a blank color, a collection $\tau \in C^4$ of tiles including a blank tile; a row R of non-blank matching tiles, with blank sides

above and on both ends of the row; an integer n coded in unary, $n > |\tau|$, $n > |R|$ (i.e. the word $\tau.R.1^n$).

Question: Is there an extension of the row R forming a finite non-trivial tiling of the plane of size at most $|R| \times n$?

Probability: Consider any reasonable P-time computable encoding of tile sets. We take a probability function corresponding to the following experiment: "choose n; choose a number of tiles lower than n, choose n tiles according to your favorite probability function; choose a length for the row lower than n, choose each tile one after another such that the current one matches the already chosen ones".

RAND-DEADLOCK

Instance: A nondeterministic finite automaton $\mathscr{A}$ (a denotes the number of its states); a word w over the alphabet $\{0,1\}$, an integer n coded in unary, $n > |w|$ and $n > |\mathscr{A}|$ (i.e. the word $\mathscr{A}.\omega.1^n$).

Question: Can a deadlock appear in the grid $\mathscr{G}_{\mathscr{A}}(n)$ on the input word w before a time steps?

Probability: Consider any reasonable P-time computable encoding automata denoted by $c(\mathscr{A})$. We take a probability function proportional to $1/(n^4 2^{|w|} 2^{|\mathscr{A}|})$. This probability function corresponding to the following experiment: "choose an integer n according to the standard probability function $\mu(n) = (6/\pi^2)(1/n^2)$. Then choose independently two integers m and k lower than n – your probability function is now proportional to $1/n^4$. Choose two words of length m and k representing, respectively, the coding of w and $\mathscr{A}$". This experiment defines a function proportional to $1/(n^4 2^{|w|} 2^{|\mathscr{A}|})$.

Theorem 3. RAND-DEADLOCK *is* RNP-*complete.*

Proof. We use the same reduction as for the first part of Theorem 1 and use the input word as in Theorem 2. The part of the proof that is specific to RNP-completeness is the following: first remark that RAND-FINITE-TILING is RNP-complete even if the number of choices for each tile of the row is bounded by 2 (see [12]). Then remark that a choice of a letter of the input word w corresponds exactly through the reduction to the choice of a tile in the row. This remark is sufficient to prove that our reduction does not diminish too much the probability of a given instance: observe that in the probability distribution, the most "brutal" part is $1/2^{|w|}$ because the other factors correspond to the size of the automaton and the bounding integers that are polynomially dependent on the size of the tile set and the associated integer. Thus the critical part is the probability associated to the row of tiles and to the word w which in the reduction depends on the row; if there is a choice for 2 different tiles in the row, then through the reduction, there are exactly 2 choices for the letter of w. Hence in both probability distributions, the factor corresponding for these choices is $1/2^{|w|}$. Thus item 5 of the method is verified and RAND-DEADLOCK is RNP-complete. □

6. Conclusion

Our results reinforce the point of view that it is distinctly different to organize parallel computers on a linear structure (such as $\mathbb{Z}$) than on a planar one ($\mathbb{Z}^2$). The second dimension gives more power and more unpredictability to phenomena. A question naturally arises: is the space $\mathbb{Z}^3$ more complicated than the plane? In other words, we wonder whether there exist "natural" problems that are decidable on $\mathbb{Z}^2$ but not on $\mathbb{Z}^3$. An idea to solve positively this question is to use the fact that planar graphs are simpler than graphs in $\mathbb{R}^3$.

Acknowledgements

We are very grateful to Yuri Gurevich for a discussion on average case completeness. We would like to thank Antoine Petit for his interesting remarks and comments.

References

[1] J. Beauquier, A. Choquet, A. Petit, G. Vidal-Naquet, Detection of deadlocks in an infinite family of nets, in: STACS'91, Lecture Notes in Computer Science, vol. 480, Springer, Berlin, 1991, pp. 334–347.
[2] S. Ben-David, B. Chor, O. Goldreich, M. Luby, On the theory of average case complexity, in: STOC, 1989, pp. 204–216.
[3] R. Berger, The undecidability of the domino problem, Memoirs Amer. Math. Soc. 66 (1966).
[4] A. Blass, Y. Gurevich, Randomizing reductions of search problems, SIAM J. Comput. 22 (5) (1993) 949–975.
[5] A. Blass, Y. Gurevich, Matrix transformation is complete for the average case, SIAM J. Comput., to appear.
[6] B. Durand, Automates cellulaires: réversibilité et complexité, Ph.D. thesis, Ecole Normale Supérieure de Lyon, 1994.
[7] B. Durand, A random NP-complete problem for inversion of 2D cellular automata, Theoret. Comput. Sci. 148 (1) (1995) 19–32.
[8] A.-C. Fabret, Etude de quelques propriétés de familles de réseaux modulaires, Ph.D. thesis, Université Paris-Sud, 1996.
[9] A-C. Fabret, A. Petit, On the undecidability of deadlock detection in families of nets, in: STACS'95, Lecture Notes in Computer Science, vol. 900, Springer, Berlin, 1995.
[10] M. Garey, D. Johnson, Computers and Intractability, Freeman, New York, 1983 edition, 1979.
[11] Y. Gurevich, Complete and incomplete randomized NP problems, in: FOCS, 1987.
[12] Y. Gurevich, Average case completeness, J. Comput. System Sci. 42 (1991) 346–398.
[13] L. Levin, Average case complete problems, SIAM J. Comput. 15 (1) (1986) 285–286.
[14] H. R. Lewis, Complexity of solvable cases of the decision problem for the predicate calculus, in: FOCS, vol. 19, 1978, pp. 35–47.
[15] J. Mazoyer, N. Reimen, A linear speed-up theorem for cellular automata, Theoret. Comput. Sci. 101 (1992) 59–98.
[16] R.M. Robinson, Undecidability and nonperiodicity for tilings of the plane, Invent. Math. 12 (1971) 177–209.
[17] Zs. Róka, Simulations between cellular automata on Cayley graphs, in: LATIN'95, Lecture Notes in Computer. Science, Springer, Berlin, 1995.
[18] P. van Embde Boas, Dominoes are forever, Research report 83-04, University of Amsterdam, Department of Mathematics, 1983.