Membrane computing: Brief introduction, recent results and applications

Gheorghe Păun[a,b], Mario J. Pérez-Jiménez[b,*]

[a] *Institute of Mathematics of the Romanian Academy, PO Box 1-764, 014700 Bucureşti, Romania*

[b] *Research Group on Natural Computing, Department of Computer Science and Artificial Intelligence, University of Sevilla, Avda. Reina Mercedes s/n, 41012 Sevilla, Spain*

Abstract

The internal organization and functioning of living cells, as well as their cooperation in tissues and higher order structures, can be a rich source of inspiration for computer science, not fully exploited at the present date. Membrane computing is an answer to this challenge, well developed at the theoretical (mathematical and computability theory) level, already having several applications (via usual computers), but without having yet a bio-lab implementation. After briefly discussing some general issues related to natural computing, this paper provides an informal introduction to membrane computing, focused on the main ideas, the main classes of results and of applications. Then, three recent achievements, of three different types, are briefly presented, with emphasis on the usefulness of membrane computing as a framework for devising models of interest for biological and medical research.

keywords: Natural computing; Membrane computing; Multiset processing; Universality; EGFR signalling network

1. Natural computing

In some sense, the whole history of computer science is the history of continuous attempts to discover, study, and, if possible, implement computing ideas, models, paradigms from the way nature – the humans included – computes. For instance, when defining the computing model which is currently known as *Turing machine* and which provides the standard definition of what is mechanically (that is, algorithmically) computable, Turing (in 1935) explicitly wanted to abstract and model what a clerk in a bank is doing when computing with numbers. One decade later, McCullock, Pitts, Kleene founded the finite automata theory starting from modelling the neuron and the neural nets; still later, this led to the area called now *neural computing* (see, e.g., Anderson, 1996)—whose roots can be found in unpublished papers of the same Turing (see, e.g., http://www.AlanTuring.net; Teuscher, 2001). *Genetic algorithms* and evolutionary computing/programming (Koza and Rice, 1992) are already well established (and much applied practically) areas of computer science. About 1 decade ago, the history making Adleman's experiment of computing with DNA molecules was reported (Adleman, 1994), proving that one can not only get inspired from biology for designing better algorithms for electronic computers, but one can also use a biological support (a bio-ware) for computing. In the last years, the search of computing ideas/models/paradigms in biology, in general in nature, became explicit and systematic under the general name of *natural computing*, and this research area is already well established, with new journals, series of books, research groups, etc., devoted to it.

* Corresponding author. Tel.: +34 954557952; fax: +34 954557952.
E-mail addresses: gpaun@us.es (G. Păun), marper@us.es (M.J. Pérez-Jiménez).

2. Starting from the cell

Membrane computing is a part of this general intellectual journey. Its starting point was the fact that the cell is the smallest living thing, and at the same time it is a marvellous tiny machinery, with a complex structure, an intricate inner activity self-regulated in a quite efficient way, and a well defined relationship with its environment—the neighboring cells included, all these polished during billions of years of evolution. Then, the observation is that computer science has not sufficiently considered the cell as an inspiration for computing models. For instance, the much promising area of DNA computing does not even "see" the cell, but only takes the DNA molecule as a "processor", without considering the natural environment where this "processor" evolves.

Thus, the challenge is to take the cell itself as a support for computations, to find in the structure and the functioning of the cell seen as a whole (here we have the same ambition as systems biology, Kitano, 2002; Tomita, 2001) those elements useful for computations. Computations in general, at the mathematical level, but with the hope to bring something useful to practical computing, either in the same style as genetic algorithms and neural computing, of improving the use of the existing computers, or proposing new types of electronic computers, or, possibly (but not very probably at short term) to lead to ways to use the cells themselves as computing supports (much similar to the way DNA computing uses DNA molecules as a genuinely new type of hardware). Note that there was no initial intention to provide models of real cells, models useful to biologists—although nowadays the applications in biology are one of the main directions of investigation in membrane computing (we will come back to this important issue in the end of this paper).

Staying at the level of computability theory and of existing computers, there are many things which we can learn from cells. Distribution, parallelism, nondeterminism, (de)centralization, (non)synchronization, coordination, communication, robustness, scalability, self-healing are only a few keywords related to current difficulties of computer science which have found (sometimes surprisingly efficient) solutions in biology. For instance, a problem which cannot be easily solved in terms of silicon engineering, but which was very efficiently solved by nature at the level of the cell is related to the coordination of processes, the control pathways which keep the cell alive, apparently without a high cost of coordination (in parallel computing the communication complexity, needed for coordinating parallel computing agents, is sometimes higher than the time and space complexity of the computation). Then, interesting questions appear in connection with the organization of cells into tissues, not to speak about the way the neurons cooperate in the brain.

All these were noticed several years ago in various contexts. We recall in this respect a significant paragraph from Butler et al. (1998):

"Another important feature of computation in biological systems is compartmentalization. This controls the stochastic nature of information being transferred by random walks (Hong, 1995). Compartments separate different reactions and reduce the search time for an enzyme to find a substrate. Compartments are formed from membranes which perform a dimensional reduction increasing the likelihood of a signalling molecule reaching a receptor. The membranes which form compartments can form distributed systems of control involving conformational or electronic state changes in receptors or channels. Another feature is diffusional information processing involving concentration of second messengers such as cyclic AMP or cyclic GMP acting as fields which controlling nerve impulse activity. A network of cells is a parallel distributed network which relies on both local and global interactions to perform associative learning. The highly interconnected nature of systems of communication in cells allows the cells to recognize combinations of environmental inputs and stabilizes the cell signalling pathways (Bray, 1990). In this way proteins in the cell can act as both a cellular memory and as a parallel distributed processing network. These networks perform information processing tasks such as amplification of small signals or adaptation when receptors become sensitized to large signals. To summarize, the individual components or devices used in biological systems are much smarter than digital devices. They can perform operations like counting or have internal memories. In many cases they can perform highly specific shape based recognition; in other cases fuzzy categorization. They act in stochastic networks to perform distributed processing. Based on these observations, it is possible to propose some system considerations for unconventional biological computation."

Rather convincing, with a broad panoply of suggestions for cell-inspired computability.

Membrane computing can be seen as a possible answer to this general challenge. The systematic investigations started (in (Păun, 2000)), with the paper circulated already through Internet already in 1998; a comprehensive introduction can be found in Păun (2002) and recent information at MC Website (2005); a presentation of applications can be found in Ciobanu et al. (2005)) with the

explicit goal of abstracting computing models from the *structure* and the *functioning* of a living cell. Because the membranes play a crucial role both in the cell architecture (from separating it from the environment to defining the internal compartmentalization) and in cell biochemistry, the theory developed under the name of *membrane computing*. The literature of the domain is very large (already in 2003, Thompson Institute for Scientific Information, ISI, has qualified the initial paper as "fast breaking" and the domain as "emergent research front in computer science"—see http://www.esi-topics.com) and the progresses rather rapid, so that the presentation here is general, focusing only on central aspects (notions and results) and on some recent developments, with a special emphasis on applications. Especially, we want to illustrate the fact that membrane computing is a useful framework for devising models of interest for biologists and for medical research. These applications are based on the fact that many biological processes (especially, at the level of the cell) can be successfully approached in terms of multiset processing, with results at least as relevant as those provided by the tools based on differential equations, and with a series of advantages related to scalability, understandability, modularity, etc.

We will discuss these important issues in some detail after introducing the basic notions of membrane computing.

3. Elements of membrane computing

The basic idea is to consider a distributed and parallel computing device, structured, like a cell, by means of a hierarchical arrangement of membranes which delimit compartments where various chemicals (we call them *objects*, to be free of any interpretation) evolve according to local reaction rules. These objects can also pass through membranes, under the control of specific rules. (We ignore at this stage the fact that many reactions which take place in a cell are controlled by proteins embedded in membranes, that the membranes not only delimit "protected reactors", but they are also supports for reactions. This aspect, of reactions taking place *on* membranes, central for instance to brane calculi developed in Cardelli (2005), started to be considered also in membrane computing, and we will return to this case later.) Because the chemicals from the compartments of a cell are swimming in an aqueous solution, the data structure we consider is that of a *multiset*—a set with multiplicities associated with its elements. Also, in close analogy with what happens in a cell (see, e.g., Alberts et al. (2002)), the reaction rules are applied in a parallel manner, with the objects to evolve by them and with the reactions themselves chosen in a non-deterministic manner. In this way, we can define transitions from a configuration to another configuration of our system, hence we can define computations. A computation which halts (reaches a configuration where no rule is applicable) provides a result, for instance, in the form of the number of objects present in the halting configuration in a specified compartment, or in the form of a special object, `yes` or `no`, sent to the environment in the end of the computation (thus answering a decision problem the system had to solve).

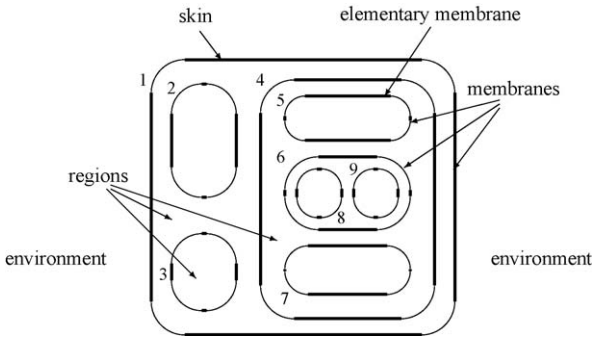

Fig. 1. A membrane structure.

There are many variants of this very basic type of a computing device, currently called *membrane system* or *P system*. In all cases, one of the fundamental ingredients is that of a *membrane structure*.

The meaning of this notion is illustrated in Fig. 1 and this is what we can see when looking to a cell through "mathematical glasses", hence abstracting as much as necessary in order to obtain a formal model, and then generalizing, for instance, considering arbitrarily many membranes arranged in arbitrarily many levels.

The typical evolution rule is of a multiset-rewriting type, hence of the form $u \to v$, where u and v are multisets of objects (this is much like a usual equation describing a chemical reaction), with several variants. A rule of the general form is called *cooperative*, when u consists of a single object the rule is called *non-cooperative* (this corresponds to context-free rules in Chomsky grammars), while an intermediate (interesting) case is that of *catalytic* rules, of the form $ca \to cv$, where c is an object which behaves like a catalyst, it never changes, but only helps object a to get transformed into multiset v. The objects from multiset v can have associated *target indications*, of the form out (meaning that the respective object has to exit the membrane where it is produced, thus becoming an element of the surrounding region—the environment, if the rule was applied in the skin region), or *in* (meaning that the object has to enter one of the immediately inner membranes, non-deterministically chosen,

if any exists, otherwise the rule cannot be applied). Thus, a typical rule is of the form $ab \to c(d, \text{out})(e, \text{in})$: a reacts with b, and as result of the reaction we produce one c (which remains in the same region where the reaction takes place), one d which goes out, and one e which goes to an inner membrane. The target indications play an important role, as they "integrate" the "computing agents" from compartments into a "computer" which behaves as a whole, as a system.

From this point of view, of ensuring the communication among parts, a very important type of rules are those which correspond to symport and antiport transmembrane processes known in biology: we write (x, in) or (x, out) for a symport moving the objects of multiset x inside, or outside a membrane, respectively, and $(x, \text{out}; y, \text{in})$ for an antiport which moves the objects of x outside at the same time with bringing the objects of y inside. (Note that we consider symport and antiport rules of a general form, passing through a membrane not only couples of objects, as usual in biology, but multisets of objects which can be arbitrarily large.)

Furthermore, we can have rules for handling membranes (creating, destroying, dividing, merging, etc.), the rules can have promoters or inhibitors, their use can be regulated by a priority relation, the permeability of membranes can be controlled by the used rules (no object can pass through a membrane which is made non-permeable, and then rules which ask for moving objects through such a membrane cannot be used), and so on and so forth, either with a biological or with a mathematical motivation.

In short, we abstract as much as possible/necessary, in order to obtain a mathematical model which is intended to be (i) minimalistic (as elegant as possible, containing as restricted ingredients as possible), but (ii) without losing the biological inspiration (hence remaining as "realistic" as possible), with (iii) good computability properties (as *powerful* as possible, in comparison with standard models of computing – Turing machine and its restrictions, and as *efficient* as possible – useful in solving computationally hard problems in a feasible time). All these are contradictory criteria, so that many models were proposed, focusing on one or another of these goals.

Then, when dealing with biological applications, further details should be added, especially related to the reaction rates, probabilities with which each rule is applied, and which can be statically associated with rules or can be dynamically computed depending on the concentration of reactants. In this way, quantitative ingredients are added, expressed by real numbers (thus, the basic discrete model is augmented with continuous ingredients), which leads to a hybrid model, able in this way to capture not only the multiset rewriting-like features of biochemical reactions themselves, but also quantitative features, usually handled in applications by systems of differential equations.

For details, both concerning the theoretical developments and the applications, we refer the reader to the monograph (Păun, 2002), the volume (Ciobanu et al., 2005), and to MC Website (2005).

We close this section by mentioning that besides cell-like P systems, there were considered also tissue-like and neural-like systems, inspired by the way the cells cooperate in tissues and the neurons work together in neural nets. A related notion is that of population P systems (Bernardini and Gheorghe, 2004), with inspiration in skin-like cell structures, populations of bacteria, etc. We do not enter into details here about such systems, with cells placed in the nodes of an arbitrary graph (dynamically defined in the case of population P systems), although certainly they deserve our attention (in particular, variants of these types of systems deserve to be considered as suggested by the liver organization and functioning, with appealing differences from other tissues and multi-cell structures—see the presentation of liver computing-like structures in Paton et al., 1992). Recently, population P systems were used in modelling quorum sensing in bacteria and the results are rather encouraging—see details in Terrazas et al. (2005) and Bernardini et al. (2005).

4. Computing power: easy universality

In language and automata theory, the intuition is that a computing model can reach the power of Turing machines as soon as it has (1) enough context-sensitivity, and (2) erasing capabilities. The first feature ensures the possibility to send information at any distance in the data structure used (in general, a string of abstract symbols, like on the tape of the Turing machine), thus providing a way to control the computation in a precise way, while the erasing makes possible the use of an arbitrarily large workspace. Both these features are plentifully available in a cell: the biochemical reactions are of the form $u \to v$, with several reactants in u, while the symport/antiport processes always involve at least two chemicals, which, in both cases, means context-sensitivity; then, erasing can be simulated by throwing waste products into the environment, or by storing objects in a "garbage collector", a membrane especially designed for such a job.

Consequently, in some sense, it should be no wonder that *most classes of P systems considered so far are computationally universal, equal in power with Turing machines*.

For instance, this is true for multiset-rewriting systems with catalytic rules (with two catalysts, see Freund et al., 2005), and also for symport/antiport P systems, even with rules of rather small sizes (see Alhazov et al., 2005).

The conclusion is that the cell is a very powerful "computer". In particular, *computing by communication only* (by symport/antiport rules) is already universal, which is a result with a more general significance, stressing once again the role of compartmentalization (hence of membranes) and of communication.

These results can have both pleasant (mathematical) and not so pleasant (practical) consequences. On the good side, if we are interested in devising (maybe, implementing) computing models inspired from the cell, then, it is just natural to look for powerful computers, if possible, able to do whatever a Turing machine can do; moreover, in order to have a programmable computer we need a model which has universality properties, in the same sense as proved already in 1935 by Turing for his machines (informally, a Turing machine is universal if it can simulate any other given machine, as soon as a code of the particular machine is introduced as an input to the universal machine). On the other hand, the only natural definition and proof of universality for Turing machines and their restrictions are encountered for the Turing machines themselves. Thus, as we said before, from a mathematical and computational point of view, the fact that most P systems are universal is a rather attractive result.

However, there is no immediate hope of implementing, neither in a lab, nor on an electronic support, a P system as a programmable computer. On the other hand, membrane computing is used more and more as a framework for modelling biological processes. In this case, a too powerful model has a major drawback: most decidability questions about the model evolution are not algorithmically solvable.

This conflict, between the intrinsic high power of cell-like computing systems and the practical need for decidable models, is a major motivation for research in membrane computing. For immediate applications, this conflict is avoided by looking for computer experiments, not for analytical results: computer programs are written which simulate the P systems modelling given biological processes, and the evolution of the systems is followed in silico for a large number of steps, looking for regularities, trends, interrelations, statistics.

5. Computational efficiency

One of the explicit goals of various branches of natural computing is to find ways to address computationally hard problems (typically, NP-complete problems) in order to solve them (in a strict sense or in a probabilistic sense) in a feasible time. In turn, one of the most promising approaches towards such a goal is *parallel computing*, as best illustrated by DNA computing.

The rules of a P system are used in parallel, that is, in each membrane all objects evolve simultaneously, and, in turn, at the level of the system all membranes evolve simultaneously. This is a good degree of parallelism, which, however, is not sufficient to devise polynomial time solutions to NP-complete problems (unless P = NP, which is not at all plausible); the proof of this result can be found in Zandron et al. (2000). However, biology suggests operations with membranes which, sometimes surprisingly, make possible polynomial (often linear) solutions to NP-complete problems. Among these operations, the most investigated so far in membrane computing were membrane division and membrane creation. While membrane division brings a further level of parallelism, making possible the exponential growth of the number of membranes (hence of separated computing compartments) in polynomial time, membrane creation has a more subtle effect, also encountered in the case of membrane division: structuring the working space. For instance, by rules of the form $a \to aa$, allowed at the theoretical level, in n steps we can produce 2^n copies of a; this is an exponential space, which, according to the theorem in Zandron et al. (2000), is not sufficient for essentially speeding-up computations. However, introducing a structure/localization by means of membranes which separate the exponentially many copies of a leads to efficiency results as mentioned above. Many problems known to be NP-complete (in certain cases, even problems which are PSPACE-complete) were proven to be solvable in polynomial time by means of membrane systems provided with membrane division or membrane creation. Among them, there are both decision problems, with yes/no answers (such as SAT, the satisfiability of propositional formulas in the conjunctive normal form), and optimization numerical problem (such as CAP, the common algorithmic problem: let S be a finite set and F be a family of subsets of S; find the cardinality of a maximal subset of S which does not include any set from F).

We do not enter here into details, but we refer, e.g., to Pérez-Jiménez et al. (2002) and Pérez-Jiménez (2005), and to the chapter from Ciobanu et al. (2005) devoted to this topic. Anyway, these results are of a

clear theoretical interest (new characterizations of the Turing complexity classes were given, as well as of the P = NP problem, intriguing borderlines between efficiency and non-efficiency were found—with many challenging open problems still waiting to be considered), without having yet a practical consequence as long as no implementation of membrane systems is available.

6. A glimpse to applications

As mentioned above, membrane computing was initiated having as primary goals computability in general and natural computing in particular, without aiming to faithfully model biological facts in such a way to provide a modelling framework for the use of biologists. However, after significantly developing at the theoretical level, the domain started to be useful for biological and medical applications, in general in the following scenario: a membrane computing model is constructed for a given process taking place in the cell, a program is written (or one of the many computer programs available is used) for simulating the model, and then computer experiments are carried out, tuning certain parameters and playing with the inputs, thus generating (numerical or graphical) data of interest for the study of the process considered.

The approach has a series of features which answer several of the limitations or difficulties of models based on differential equations: modularity (intrinsic to a membrane system, crucial in biology, but not specific to systems of differential equations), scalability/extensibility (further membranes and/or evolution rules can be added to an existing system without essentially changing the way of working with the system), understandability (evolution rules directly correspond to chemical reactions, hence they are totally "transparent" for biologists), programmability (a rewriting-based model can be easily transformed into a program, with certain programming languages, such as CLIPS, perfectly adequate to such a job), while preserving other good features of differential equations models, such as non-linearity of evolution.

Of course, this does not mean at all that membrane computing should substitute differential equations in all applications. Membrane computing – in general, multiset processing by means of rewriting-like rules – is a technique complementary to systems of differential equations, in many cases as relevant as differential equations, in most cases much easier to use, and in some cases the unique technique which can be used; this last situation is met, for instance, when we deal with small populations of reactants, such that a discrete model is the only one adequate (approximating finite by infinite is useful, provided that the "finite" is large enough).

On the other hand, as mentioned also earlier, the best results are obtained when the intrinsically discrete models of membrane computing are supplemented with continuous ingredients, mainly in the form of numbers associated with rules (given in advance or computed at each step of the evolution of the system) and which control the rule applications. Reaction rates, stoichiometric coefficients, kinetic constants, probabilities, etc. were considered in this respect.

All these means that the mathematical minimalism should be forgotten when dealing with applications. Similarly, the computability-specific features should be abandoned. For instance, when modelling a biological process we are no longer interested in halting computations and in their results, but in the evolution itself of the process, which, in general, should be endless. A dynamical systems approach is then necessary, and a related theory started to be elaborated, especially by the efforts of Verona group (see Manca, 2004; Manca et al., 2004, and the corresponding chapter from Ciobanu et al. (2005)). A typical application in biology/medicine where such a hybrid P system model is involved will be discussed below.

Besides applications in biology, membrane computing was considered also in other areas, such as computer graphics (models based on compartmentalized Lindenmayer systems proved to be more powerful and more efficient than those using classic L systems), cryptography, modelling in a uniform way parallel architectures, in linguistics, economics, etc. A very promising application, in devising approximate algorithms for hard optimization problems, will be also briefly presented below.

7. Working with objects on membranes

We pass now to describing three recent results, starting with a theoretical one, which bridges brane calculi Cardelli (2005) and membrane computing. Brane calculi are somewhat dual to membrane computing, as they work with objects placed on membranes (corresponding to proteins attached to or embedded in the real membranes), with operations with membranes controlled by these objects, and trying to stay as close to the biology as possible; also the tools and the goals are different—process algebra and systems biology, respectively. However, the two areas can easily be integrated in an extended membrane computing working both with objects in the compartments and bound to membranes. In particular, the main operations with membranes considered in the two calculi developed in Cardelli (2005) can be

easily formalized in terms of P systems, and then used in building computing models as standard in membrane computing.

This was done in Cardelli and Păun (2005) for four of the six operations from Cardelli (2005): pino, exo, mate, drip. We recall here only the last two operations:

$$\text{mate} : [\,]_{ua}[\,]_v \to [\,]_{uxv}, \qquad \text{drip} : [\,]_{uav} \to [\,]_{ux}[\,]_v,$$

where the membranes are represented by square brackets, a is a protein, u, v, and x are multisets of proteins which are supposed to be placed on the respective membranes (and thus control the operations; the fact that we again consider multisets reminds of the fact that the currently accepted model of biological membranes is the fluid-mosaic one, with the possibility of proteins to migrate on the surface of membranes). The meaning of these rules should be obvious: in the first case, the two membranes merge and form a unique membrane, with all proteins on it; in the second case, the membrane is split into two membranes, with the proteins specified by u and v distributed as indicated by the rule and all other proteins of the starting membrane, not specified in the drip rule, distributed non-deterministically on the two new membranes; in each case, protein a is replaced by the proteins specified by x.

We stress the fact that these operations are biologically realizable, they are chosen and formulated in such a way to directly correspond to processes taking place in a cell. This makes rather interesting the result obtained in Cardelli and Păun (2005): *P systems based on the mate/drip operations are Turing complete*, and the proof is obtained for systems with a reduced number of membranes (at most eleven at any step of a computation, placed in only two levels, but improvements are probably possible from this point of view), with multisets u, v, x of a reduced size (at most five elements used in total in each rule). As said before, Turing completeness is good from a computational point of view, because it brings universality/programmability, but it is bad from the modelling point of view, because it brings non-decidability. This is an aspect which entails a special difficulty to systems biology (see again Kitano, 2002 and the very title of Tomita, 2001) and should be considered when evaluating the plans and the achievements in this area.

A similar universality result is reported in Păun (2005), for another pair of operations, corresponding to pino, exo from Cardelli (2005).

The operations from Cardelli (2005) are of the form

$$\text{pino} : [\,]_{uav} \to [[\,]_{ux}]_v, \qquad \text{exo} : [[\,]_{ua}]_v \to [\,]_{uxv},$$

with the semantics as suggested by the following rules

$$\text{pino} : [P]_{uav} \to [[\,]_{ux} P]_v,$$
$$\text{exo} : [[P]_{ua} Q]_v \to P[Q]_{uxv},$$

where P and Q represent the contents of the regions where they appear.

In Păun (2005) the interpretation of these rules was slightly changed, as suggested below (because the new semantics do not correspond to the operations *pino, exo*, we change the name of operations to cre, dis, for "membrane creation" and "membrane dissolution"):

$$\text{cre} : [P]_{uav} \to [[P]_{ux}]_v,$$
$$\text{dis} : [[P]_{ua} Q]_v \to [P\, Q]_{uxv}.$$

That is, when a membrane is created inside an existing membrane, the new membrane contains all previously existing membranes, and while dissolving a membrane, its contents remains inside the membrane where it was placed before the operation. The interpretation of the latter operation is rather similar to the usual dissolution operation in membrane computing, while the membrane creation is understood as doubling the existing membrane, with the initial multiset marking the membrane being distributed to the two new membranes.

Also for this pair of operations it is possible to show that we reach the power of Turing machines—we skip the formal statement and proof of this result and we refer to the downloadable version of paper (Păun, 2005) for details.

8. Nishida's membrane algorithms

The most applied branch of natural computing is by far the evolutionary computing, genetic algorithms included, which provides unexpectedly (from a mathematical point of view) good solutions to a large number of problems, just performing a random search in the space of candidate solutions, driven by evolutionary metaphors. Roughly speaking, the solutions are encoded as "chromosomes", which are evolved by means of random crossover operations and point mutations, and selected by means of a fitness mapping. In some sense, the best "explanation" why this strategy works so well is a "bio-mystical" one: because the same strategy was used by nature in evolving species. Well, but evolution is not only based on DNA processing, but also on improving at other levels, the one which interests us here being the cell. What should we choose from the cell architecture and functioning and how can this be imitated through a software run on a usual computer in order to

obtain a class of "cell-like" algorithms as useful as genetic algorithms?

A first answer to this question was provided by Nishida (2004), in the form of *membrane algorithms*. These algorithms can be considered as a high level (distributed and dynamically evolving their structure during the computation) evolutionary algorithms. The basic variant is the following one: a (small) number of candidate solutions to an optimization problem are placed in the regions of a membrane structure of a linear shape (with the membranes embedded one in another one), together with local sub-algorithms which can improve the local solutions; after a (small) number of steps of local work, when the solutions from each membrane are evolved, the best of them is sent to the immediately lower membrane and the worst is sent to the immediately upper membrane (with exceptions to this rule in the innermost and the outermost membrane); in this way, the better solutions are moved down and the worst ones are moved up in the membrane hierarchy; this process is iterated until either a specified number of steps is reached, or no improvement of the best solution is obtained for a specified number of steps. When halting, the central membrane provides the answer, the solution to the problem. There are several variants, in terms of the number of membranes, with the initial solutions generated by a first generation of membrane algorithms (thus working in a two-stage manner, which proves to be very efficient), with the possibility to create or to destroy certain membranes during the computation, etc.

Nishida has checked this strategy for the travelling salesman problem (find the shortest path which visits all nodes of a graph, passing exactly once through each node), and the results were more than encouraging. Known benchmark problems were addressed and always the conclusions were the same: at the beginning, the convergence is very fast, so that, after a small number of steps we get a solution which is almost the best which can be provided by the algorithm, irrespective how much we continue to work; the number of membranes is rather influential on the quality of the solution, but, depending also on the dimension of the problem (the number of nodes in the graph), 50–100 membranes are sufficient for obtaining a solution which is better than that provided by simulated annealing; the method is reliable, both the average quality and the worst solutions were good enough and always better than the average and the worst solutions given by simulated annealing.

This direction of research is rather new, but it has a clear practical interest, so further efforts are necessary, e.g., checking the usefulness of this approach for other problems, comparing this method with other methods, looking for more sophisticated membrane algorithms. Trusting the cell, we are quite optimistic with this kind of strategies.

9. Applications in cancer-related research

Because the processes related to cancer are at the same time intricate and well investigated, with a large amount of data accumulated and publicly available, various types of models were attempted in this area—membrane computing models included. We only mention the simulation of p53 protein pathways control (the interaction between proteins p53 and MdM2) through a P system, as carried out by Suzuki and his co-workers (details can be found in Ciobanu et al., 2005), and the modelling of avascular tumor growth, carried out in Gutiérrez-Naranjo and Pérez-Jiménez (2005), and we briefly present the modelling of EGFR (epidermal growth factor receptor) signalling network from Pérez-Jiménez and Romero-Campero (2005). The EGFR has an important role in cell growth, survival, proliferation and differentiation, hence it is a key target for the development of anticancer therapies. The signaling network involved in EGFR activity is rather complex. In Pérez-Jiménez and Romero-Campero (2005), a P system with three regions, 60 proteins, and 160 reactions among these proteins is considered. The picture of the system is given in Fig. 2, where both the membranes are the reactions are indicated (the nucleus was not considered yet, but ongoing investigations will take it into account, hence one further region will be used). These reactions were expressed as multiset rewriting rules, with reaction rates computed according to the law of mass action and to Michaelis law. The model was simulated by means of a CLIPS program developed by the authors, and the evolution in time of various proteins was graphically represented. The results, proving the robustness of the EGFR signalling network, looks like those in Figs. 3 and 4, and they are in full accordance with biological data (e.g., from Klodenko et al., 1999) obtained experimentally. This proves that the model is reliable, hence lab experiments can be replaced by computer experiments based on this model (and its supporting software).

10. Membrane computing software

Besides the theoretical developments, mainly related to computability power and efficiency, and the applications, a significant direction of research in membrane computing concerns implementations (in fact, simulations) on the usual computer. There are about two dozens of programs able to simulate various types of P systems, some of them with a didactic purpose (used, e.g., as

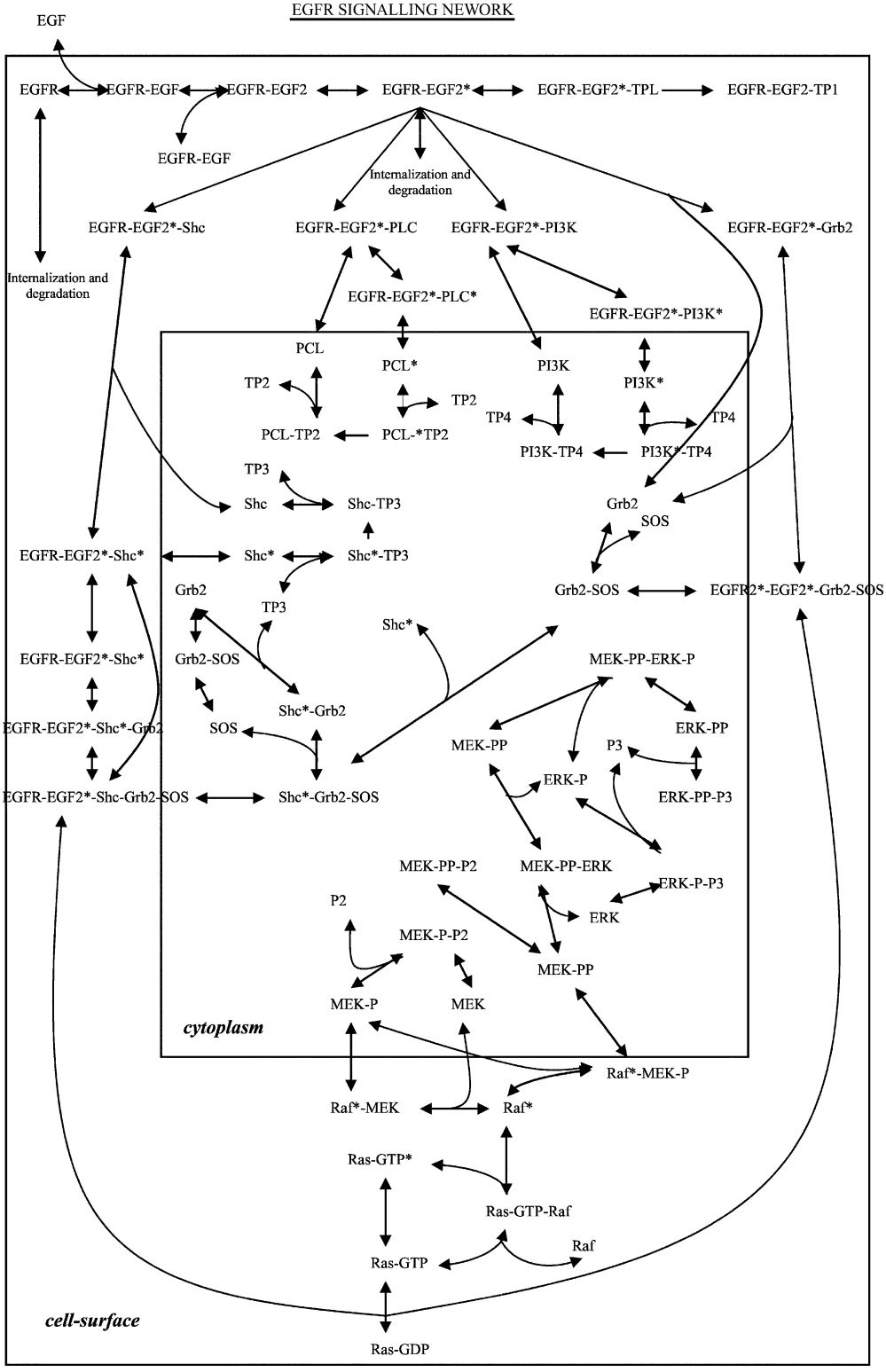

Fig. 2. The EGF signalling network.

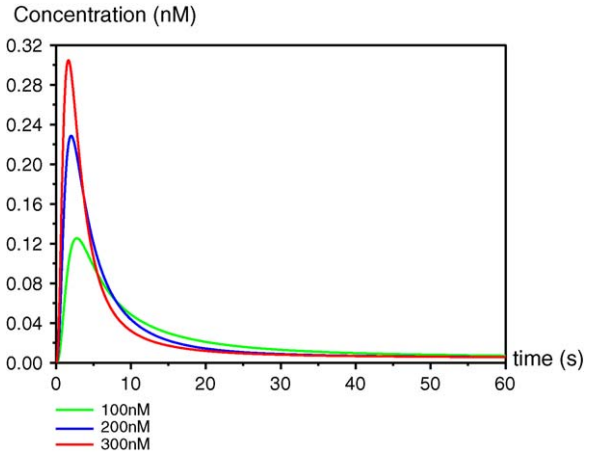

Fig. 3. The EGF receptor activation by auto-phosphorylation (with a rapid decay after a high peak in the first 5 s).

teaching or research assistants, for instance, in checking the correctness of constructions involved in proofs), others devoted to applications; there also are several distributed implementations, trying to be closer to the intrinsic distributed and parallel character of P systems. We refer to MC Website (2005) and to the corresponding chapter from Ciobanu et al. (2005) for details.

11. Hopes and limits

At about 7 years since the domain was initiated, the question arises whether membrane computing succeeded in its attempt to "learn computing paradigms from living cells". The answer is a qualified yes: from a theoretical point of view, a powerful branch of natural computing was developed, with a continuous afflux of new ideas,

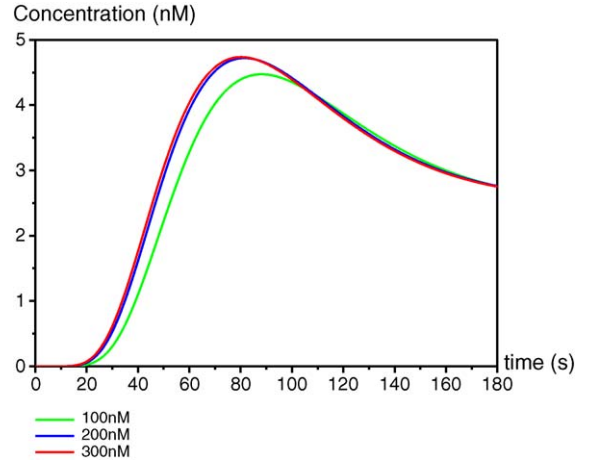

Fig. 4. The evolution of the kinase MEK (proving a surprising robustness of the signalling cascade).

notions, problems, having a series of applications, especially in modelling biological phenomena. No lab implementation was intended, and no such implementation is known to be planned for the near future. Then, except for Nishida's membrane algorithms (and, partially, the computer graphics and cryptography), no "real" application in practical computer science was reported, at least not in the direction which is more plausible: imitating the cell in devising new types of algorithms and/or of computers. For instance, at the theoretical level, polynomial solutions to computationally hard (presumably intractable) problems can be devised in terms of P systems, but there is no way to implement these solutions on the existing computers, sequential or with a limited parallelism as they are. In several places, it was advocated that membrane computing can be a general, unifying framework for investigating distributed and parallel computing, many parallel architectures were formalized/simulated in terms of P systems, but also this promising idea still waits to be fully materialized.

However, not planned in the beginning, membrane computing turned out to be a useful framework for building models with biological relevance, and the number of applications of this type is steadily increasing and becoming more and more advanced and elaborated. Multiset processing has already convincingly proved the usefulness and the advantages with respect to tools based on differential equations—at least in a series of circumstances where compartmentalization is essential, or the concentration of chemicals is too small in order to legitimate the use of continuous mathematical tools. Easy scalability, programmability, and understandability are other attractive features of membrane computing as a modelling framework for the use of biology.

On the other hand, coming back to the initial idea of learning from the cell/biology for the use of computability (which is the general goal of natural computing), we have to be aware that not all processes, tools, ideas which work so marvellously in nature can be imitated in silico. This remark is a warning for natural computing in general, not only for membrane computing. Nature has (in a certain sense) unlimited time and resources, nature is cruel, kills what is not fit (all these are difficult to incorporate in computers, let them be based on electronic hardware or on a hypothetic bio-ware); nature has other goals than computing; many bio-chemical processes have a degree of non-determinism which we cannot afford in our computations; the life processes are complex, with a high degree of redundancy; biology seems to deal with non-crisp mathematics, with probabilities, with fuzzy estimations, which are not fully manageable in computations.

With respect to cell-inspired computing, the big hopes are related to distribution, parallelism, decentralization, self-healing, and other similar dreams of computer science, but we have to note that the cell is not fully decentralized, not maximally parallel, not fully non-deterministic, and so on. The right questions concerns then the *right degree* of centralization, parallelism, etc.—and in this respect further efforts should be paid.

Last but not least, maybe we dream too much even from a theoretical point of view. First, the space–time trade-off specific to molecular computing, cannot redefine complexity classes, and it is sometimes too costly in space (in the size of used bio-ware). Then, Conrad (1988) warned us that programmability (universality), efficiency, and evolvability are three contradictory features of any computing model, of any type, and we cannot simultaneously improve on all of these features. Both these last observations indicate that there is no free lunch in computer science, even in the bio-inspired one.

References

Adleman, L.M., 1994. Molecular Computation of Solutions to Combinatorial Problems. Science 226, 1021–1024.

Alberts, B., Johnson, A., Lewis, J., Raff, M., Roberts, K., Walter, P., 2002. Molecular Biology of the Cell, 4th ed. Garland Science, New York.

Alhazov, A., Margenstern, M., Rogozhin, V., Rogozhin, Y., Verlan, S., 2005. Communicative P systems with minimal cooperation. In: Membrane Computing. International Workshop WMC5. Revised Papers, LNCS 3365. Milan, Italy, 2004. Springer, Berlin, pp. 162–178.

Anderson, J.A., 1996. An Introduction to Neural Networks. The MIT Press, Cambridge, MA.

Bernardini, F., Gheorghe, M., 2004. Population P systems. J. Universal Comput. Sci. 10 5, 509–539.

Bernardini, F., Gheorghe, M., Krasnogor, N., Muniyandi, R.C., Pérez-Jiménez, M.J., Romero-Campero, F.J., 2005. On P systems as a modelling tool for biological systems. Sixth Workshop on Membrane Computing, Vienna.

Bray, D., 1990. Intracellular signalling as a parallel distributed process. J. Theor. Biol. 143, 215–231.

Butler, M.H., Paton, R.C., Leng, P.H., 1998. Unconventional approaches for biologically inspired computing. In: Calude, C.S., Casti, J., Dinneen, M.J. (Eds.), Unconventional Models of Computation. Springer, Singapore, pp. 118–130.

Cardelli, L., 2005. Brane Calculi. Interactions of biological membranes. In: Computational methods in systems biology. International conference CMSB 2004. Revised Selected Papers, LNCS 3082. Paris, France, May 2004. Springer, Berlin, pp. 257–280.

Cardelli, L., Păun, Gh., 2005. An universality result for a (mem)brane calculus based on mate/drip operations. In: Gutiérrez-Naranjo, M.A., Păun, Gh., Pérez-Jiménez, M.J. (Eds.), Cellular Computing. Complexity Aspects. Fenix Editora, Sevilla, pp. 75–94.

Ciobanu, G., Păun, Gh., Pérez-Jiménez, M.J. (Eds.), 2005. Applications of Membrane Computing. Springer, Berlin.

Conrad, M., 1988. The price of programmability. In: Herken, R. (Ed.), The Universal Turing Machine: A Half-Century Survey. Kammerer and Unverzagt, Hamburg, pp. 285–307.

Freund, R., Kari, L., Oswald, M., Sosik, P., 2005. Computationally universal P systems without priorities: two catalysts are sufficient. Theor. Comput. Sci. 330 2, 251–266.

Gutiérrez-Naranjo, M.A., Pérez-Jiménez, M.J., Romero-Campero, F.J., 2005. Simulating avascular tumors with membrane systems. In: Proceedings of the Third Brainstorming Week on Membrane Computing, Sevilla. RGNC Report 01/2005, Sevilla University, pp. 185–196.

Hong, F.T., 1995. Mesoscopic processes in biocomputing. The role of randomness and determinism. In: Extended Abstracts of the Fifth International Symposium on Biomolecular and Molecular Electronic Devoces and the Sixth International Conference on Molecular Electronics and Biocomputing, Tokyo, pp. 184–281.

Kitano, H., Nov 2002. Computational systems biology. Nature 420, 206–210.

Klodenko, B.G., et al. 1999., Quantification of short term signalling by the epidermal growth factor receptor. J. Biol. Chem. 274, 30169–30181.

Koza, J.H., Rice, J.P., 1992. Genetic Algorithms: The Movie. MIT Press, Cambridge, MA.

Manca, V., 2004. On the dynamics of P systems. In: Pre-proceedings of the Fifth Workshop on Membrane Computing, WMC5. Milano, Italy, pp. 29–43.

Manca, V., Franco, G., Scollo, G., 2004. State transition dynamics. basic concepts and molecular computing perspectives. In: Gheorghe, M. (Ed.), Molecular Computational Models. Unconventional Approaches. Idea-Group, London, pp. 32–54.

MC Website, 2005. The membrane computing website: http://psystems.disco.unimib.it.

Nishida, T.Y., 2004. An approximate algorithm for NP-complete optimization problems expoiting P systems. In: Proceedings of the Workshop on Uncertainty in Membrane Computing. Palma de Mallorca, pp. 185–192.

Paton, R.C., Nwana, H.S., Shave, M.J.R., Bench-Capon, T.J.M., 1992. Computing at the tissue/organ level (with particular reference to the liver). In: Varola, F.J., Bourgine, P. (Eds.), Towards a Practice of Autonomous Systems. MIT, Bradford, Cambridge, pp. 411–420.

Păun, Gh., 2000. Computing with membranes. J. Comput. Syst. Sci. 61 (1), 108–143 (and Turku Center for Computer Science—TUCS Report 208, November 1998. http://www.tucs.fi).

Păun, Gh., 2002. Membrane Computing. An Introduction. Springer, Berlin.

Păun, Gh., 2005. One more universality result for P systems with objects on membranes. In: Proceedings of the Third Brainstorming Week on Membrane Computing, February 2005. Sevilla, Spain, Technical Report RGNC 01/05 (http://www.gcn.us.es/).

Pérez-Jiménez, M.J., 2005. An approach to computational complexity in membrane computing. In: Mauri, G., Păun, Gh., Pérez-Jiménez, M.J., Rozenberg, G., Salomaa, A. (Eds.), Membrane Computing. International Workshop WMC5, Milano, Italy, 2004. Revised and Invited Papers. LNCS 3365, Springer, Berlin, pp. 85–109.

Pérez-Jiménez, M.J., Romero-Campero, F.J., 2005. A study of the robustness of the EGFR signalling cascade using continuous membrane systems. In: Mira, J., Alvarez, J.R. (Eds.), Mechanisms, Symbols, and Models Underlying Cognition. First International Work-Conference on the Interplay between Natural and Artificial Computation, IWINAC. LNCS 3561, Springer, Berlin, pp. 268–278.

Pérez-Jiménez, M., Romero-Jiménez, A., Sancho-Caparrini, F., 2002. Teoría de la Complejidad en Modelos de Computatión Celular con Membranas. Editorial Kronos, Sevilla.

Terrazas, G., Krasnogor, N., Gheorghe, M., Bernardini, F., Diggle, S., Camara, M., 2005. An environment aware P system model of quorum sensing. In: Barry Cooper, S., Löwe, B., Torenvliet, L. (Eds.), New Computational Paradigms. First Conference on Computability in Europe, CiE 2005, Amsterdam. LNCS 3536, Springer, pp. 479–485.

Teuscher, C., 2001. Alan Turing. Life and Legacy of a Great Thinker. Springer, Berlin.

Tomita, M., 2001. Whole-cell simulation: a grand challenge of the 21st century. Trends Biotechnol. 19, 205–210.

Zandron, C., Ferretti, C., Mauri, G., 2000. Solving NP-complete problems using P systems with active membranes. In: Antoniou, I., Calude, C.S., Dinneen, M.J. (Eds.), Unconventional Models of Computation. Springer, London, pp. 289–301.