Trading Geometric Realism for Efficiency in Tissue P Systems

Alberto Leporati Luca Manzoni Giancarlo Mauri
Antonio E. Porreca Claudio Zandron
Dipartimento di Informatica, Sistemistica e Comunicazione
Università degli Studi di Milano-Bicocca
Viale Sarca 336/14, 20126 Milano, Italy
{leporati,luca.manzoni,mauri,porreca,zandron}@disco.unimib.it

Abstract

It has been recently proved that polynomial-time tissue P systems with cell division are only able to solve decision problems in the complexity class \( P \) when their cell structure is embedded into the Euclidean space \( \mathbb{R}^d \), for \( d \in \mathbb{N} \). In this paper we show that if the space has an appropriate shape and is polynomial-time navigable (but not embeddable in \( \mathbb{R}^d \)), then it is possible to even solve \( \text{PSPACE} \)-complete problems. This means that the computational power of tissue P systems can be varied from \( P \) to \( \text{PSPACE} \) by just operating on the properties of the space in which they are located.

1 Introduction

Tissue P systems are biologically inspired computing devices based on communication between cells of a tissue. From a complexity theoretic perspective, tissue P systems with cell division rules [7] are able to solve \( \text{NP} \)-complete problems in polynomial time by exponentially many cells working in parallel.

However, a feature which is critically exploited to solve \( \text{NP} \)-complete problems is to assume that each cell of the tissue P system can communicate with every other cell (having a different label). While this is reasonable for small systems, it is certainly unreasonable when the number of cells grows exponentially with the input size. Therefore, it is essential for large tissue P systems to take into account the placement of the cells in the three-dimensional Euclidean space \( \mathbb{R}^3 \). For this reason, in [4] we have modelled the geometric space where the tissue P systems are located as a graph, where vertices are possible locations for cells and edges are interpreted

*This work was partially supported by Università degli Studi di Milano-Bicocca, Fondo d’Ateneo 2015: “Complessità computazionale e applicazioni crittografiche di modelli di calcolo bioispirati”.*
as proximity, that is, as the possibility of the cells located in the two end vertices to communicate. This spatial graph model is a direct generalisation of standard tissue P systems, and include them as a particular case. In order to consider only systems that can actually be constructed, in [4] we have characterised the spatial graphs that can be “embedded” in the three-dimensional Euclidean space. This embedding has only two restrictions, that can be informally stated as follows: cells that are considered “near” in the spatial graph cannot be placed too far in $\mathbb{R}^3$, and cells that are considered “distant” cannot be placed too close in $\mathbb{R}^3$. Practically speaking, the spatial graph must represent not-too-distorted real world distances.

In [4] we have thus proved that these “realistic” tissue P systems are severely limited in their ability to increase their number of cells when working in polynomial time: their growth falls from exponential to at most polynomial in $\mathbb{R}^3$. This, as a consequence, limits their computational ability to the complexity class $P$. In what follows we show that if we allow the space graph to assume an appropriately crafted shape, then it is possible to even solve PSPACE-complete problems, which are conjecturally harder than the $P^{#P}$ upper bound of standard tissue P systems [3]. This means, of course, that such a space graph does not completely satisfy the requirements imposed by the definition given in [4]: in particular it is not embeddable in $\mathbb{R}^3$, and hence it cannot be considered “realistic”. The result presented here can be seen as a further remark of the importance of the constraints given in [4], as one may easily obtain more powerful tissue P systems by escaping from just one of the imposed requirements. It also shows that the computational power of tissue P systems can be varied from $P$ to PSPACE by just operating on the properties of the space in which they are located.

2 Preliminary notions

The notion of geometrical space in tissue P systems (with cell division) has been given in [4] by means of an underlying undirected graph. This spatial graph, not to be confused with the graph implied by the communication rules of tissue P systems, is an abstract representation of the space in which tissue P systems live and operate; each vertex represents a possible location for a cell, while the edges represent spatial proximity. Only cells located at vertices connected by an edge can actually communicate, assuming a suitable communication rule actually exists; furthermore, the tissue P system can expand by division only into adjacent empty vertices. The spatial graph represents the geometrical space where a family of tissue P systems live, and as such does not depend on the individual members of the family. Examples of geometrical spaces that we may want to model as spatial graphs are a test tube, a biochemistry laboratory, or the whole Euclidean space. Therefore, we do not require the spatial graph to be finite$^1$. Here we recall the relevant definitions; for a more detailed introduction on multiset processing and tissue P systems, we refer the reader to the original paper [5].

**Definition 1.** Given a (possibly infinite) connected undirected graph $G = (V_G, E_G)$, a tissue P system over $G$ is a structure $\Pi = (\Gamma, E, w_1, \ldots, w_d, R)$, where:

$^1$From a mathematical perspective, the infiniteness of the space allows us to process arbitrarily large inputs with tissue P system algorithms when the number of cells grows with the input size.
• $\Gamma$ is an alphabet, i.e., a finite non-empty set of symbols, usually called *objects*;

• $E \subseteq \Gamma$ is the alphabet of objects initially located in the external environment, in *infinitely* many copies;

• $d \geq 1$ is the *degree* of the system, i.e., the initial number of cells;

• $w_1, \ldots, w_d$ are *finite* multisets over $\Gamma$, describing the initial contents of the $d$ cells; here $1, \ldots, d$ are *labels* identifying the cells of the P system, and 0 is the label of the external environment;

• $R$ is a finite set of rules, of the following types:

  (a) *Communication rules*, denoted in this paper by $[u]_h \leftrightarrow [v]_k$ and in the literature by $(h, u/v, k)$, where $h$ and $k$ are distinct labels (including the environment), and $u$ and $v$ are multisets over $\Gamma$ (at least one of them nonempty): these rules are applicable if there exist a region with label $h$ containing $u$ as a submultiset and a region $k$ containing $v$ as a submultiset; the effect of the rule is to exchange $u$ and $v$ between the two regions. If $h = 0$ (resp., $k = 0$) then $u$ (resp., $v$) must contain at least one object from $\Gamma - E$, i.e., an object with finite multiplicity$^2$. In this paper we consider a rule $[u]_h \leftrightarrow [v]_k$ and its syntactic reverse $[v]_k \leftrightarrow [u]_h$ to be the same rule.

  (b) *Division rules*, of the form $[a]_h \rightarrow [b]_h [c]_h$, where $h \neq 0$ is the label of a cell and $a, b, c \in \Gamma$: these rules can be applied to a cell with label $h$ containing at least one copy of $a$; the effect of the rule is to divide the cell into two cells, both with label $h$; the object $a$ is replaced in the two cells by $b$ and $c$, respectively, while the rest of the original multiset contained in $h$ is replicated in both cells.

A *configuration* $C$ of a tissue P system consists of a multiset over $\Gamma - E$ describing the objects appearing with finite multiplicity in the environment, and a multiset of pairs $(h, w)$, where $h$ is a cell label and $w$ a finite multiset over $\Gamma$, describing the cells. The cells of each configuration are injectively mapped into the set of vertices $V_G$, and the initial location of the cells is given as part of the initial configuration of $\Pi$.

A *computation step* changes the current configuration in the following way

• Each object can be subject to at most one rule, and each cell can be subject to *any number* of communication rules or, *alternatively*, a *single* division rule.

• The application of rules is *maximally parallel*: each cell is subject to a maximal multiset of rules (i.e., no further rule can be applied).

• When several conflicting rules can be applied at the same time, a nondeterministic choice is performed; this implies that, in general, multiple possible configurations can be reached after a computation step.

$^2$Since communication rules are applied in a maximally parallel way, this restriction avoids the situation where infinitely many objects from the environment simultaneously enter a cell.
• Only cells located at adjacent vertices of the spatial graph can communicate.

• Communication with the environment is always possible. This corresponds to assuming the presence of an “interstitial fluid” surrounding each cell.

• Cell division can be applied to a cell located at vertex $u$ only when an adjacent empty vertex $v$ exists; the two resulting cells are nondeterministically placed at vertices $u$ and $v$.

• Except when dividing, the cells keep their position in the spatial graph.

A computation $\mathcal{C} = (C_0, C_1, \ldots)$ of the tissue P system $\Pi$ is a (possibly infinite) sequence of configurations where $C_0$ is the initial configuration, and every configuration $C_{i+1}$ is reachable from $C_i$ via a single computation step. A finite computation $\mathcal{C} = (C_0, \ldots, C_k)$ is halting if no rules are applicable in $C_k$.

Standard tissue P systems can be considered as tissue P systems whose cells are located in the complete spatial graph with a countably infinite number of vertices. In fact, the complete graph poses no restrictions to the communication between cells other than those given by the rules of the tissue P systems, and the initial position of the cells is immaterial. Since tissue P systems have a finite number of cells at each computation step, each cell always has free adjacent vertices; thus, cell division is not limited by the shape of the space but, once again, only by the rules of the system. As a consequence, the evolution of the tissue P systems is the same as in the standard model with no underlying spatial graph.

We now recall a notion of embedding for metric spaces [1]. Each connected undirected graph $G$ is a metric space with distance $d_G(u, v)$ being the length of the shortest path between vertices $u$ and $v$ [1].

**Definition 2.** Let $(X, d_1)$ and $(Y, d_2)$ be metric spaces. A map $\varphi: X \to Y$ is called a bi-Lipschitz embedding of $X$ into $Y$ if and only if there exist two positive real constants $\alpha$ and $\beta$ such that, for all $x, y \in X$, the following inequalities hold:

$$\alpha \times d_1(x, y) \leq d_2(\varphi(x), \varphi(y)) \leq \beta \times d_1(x, y).$$

These embeddings possess the two properties that we consider essential with respect to the geometry of tissue P systems: the centres of two cells have a minimum distance (as given, for instance, by the radius of the cells), and two communicating cells cannot be too far away. Bi-Lipschitz embeddings are always injective and continuous. As an example, the $d$-dimensional hypercubic grid over $V_G = \mathbb{Z}^d$ is bi-Lipschitz embeddable into the Euclidean space $\mathbb{R}^d$ via $\varphi(\vec{x}) = \vec{x}$, whereas the complete graph with a countably infinite number of vertices cannot be bi-Lipschitz embeddable into $\mathbb{R}^d$ for any $d \in \mathbb{N}$ [4].

In this paper we relax the constraint that the spatial graph be embeddable in $\mathbb{R}^d$. We are thus able to show that by choosing an appropriate shape for the space in which the tissue P systems are located, their computational power increases, passing from the ability to solve only problems in $P$ (obtained when the graph is embeddable in $\mathbb{R}^d$) to the ability to solve $PSPACE$-complete problems in polynomial time.

We also recall the notion of graph growth, that describes how many vertices are reachable at most by means of finite paths of increasing length [1].
Definition 3. A graph \( G = (V_G, E_G) \) has growth \( g: \mathbb{N} \to \mathbb{N} \) if and only if
\[
g(n) = \sup_{v \in V_G} \left| \{ u \in V_G : d_G(v, u) \leq n \} \right|.
\]

Based on this definition, the \( d \)-dimensional hypercubic grid over \( V_G = \mathbb{Z}^d \) has polynomial growth \( O(n^d) \), while the infinite \( b \)-ary tree has exponential growth \( \Omega(b^n) \). The complete graph with a countably infinite number of vertices does not have finite growth, as there are already infinitely many vertices at distance 1.

A lemma proved in [4] shows that the geometry of finite-dimensional Euclidean spaces forces any embedded graph to have at most polynomial growth. In particular, “realistic” graphs, that is, those that can be embedded into the Euclidean space \( \mathbb{R}^d \), have at most polynomial growth \( O(n^d) \). So, for example, the infinite \( b \)-ary tree cannot be considered “realistic”, as it has exponential growth \( \Omega(b^n) \) for any \( b \geq 2 \), and hence it cannot be bi-Lipschitz embedded into any finite-dimensional Euclidean space.

Since we will be interested in defining families \( \Pi = \{ \Pi_x : x \in \Sigma^* \} \) of tissue P systems over spatial graphs, where the mapping \( x \mapsto \Pi_x \) is computable in polynomial time, we want to avoid “cheating” by encoding the solution of the problem under consideration in the spatial graph itself. We thus limit spatial graphs to those with a local structure that is easy to compute.

Definition 4. A graph \( G = (V_G, E_G) \) is said to be polynomial-time navigable if and only if there exists a deterministic Turing machine \( M \) running in polynomial time such that, for all \( v \in V_G \) and \( k \in \mathbb{N} \), the output of \( M(v, k) \) is the \( k \)th vertex adjacent to \( v \) (under some fixed ordering of such set); the machine \( M \) rejects if \( v \) does not have a \( k \)th adjacent vertex (i.e., if \( k \) is too large).

For instance, although not embeddable in \( \mathbb{R}^d \) for any \( d \in \mathbb{N} \), the complete graph with a countably infinite number of vertices is polynomial-time navigable.

Tissue P systems can be used as language recognisers by employing two distinguished objects yes and no: we assume that all computations are halting, and that either object yes or object no (but not both) is released into the environment, and only in the last computation step, in order to signal acceptance or rejection, respectively. If all computations starting from the same initial configuration are accepting, or all are rejecting, the tissue P system is said to be confluent.

In order to solve decision problems (i.e., decide languages), we use families of recogniser tissue P systems \( \Pi = \{ \Pi_x : x \in \Sigma^* \} \) sharing the same spatial graph \( G \). Each input \( x \) is associated with a tissue P system \( \Pi_x \) that decides the membership of \( x \) in the language \( L \subseteq \Sigma^* \) by accepting or rejecting. The mapping \( x \mapsto \Pi_x \) must be efficiently computable for inputs of any length, as discussed in detail in [6].

Definition 5. A family of tissue P systems \( \Pi = \{ \Pi_x : x \in \Sigma^* \} \) over a spatial graph \( G \) is said to be (polynomial-time) uniform if the mapping \( x \mapsto \Pi_x \) can be computed by two polynomial-time deterministic Turing machines \( E \) and \( F \) as follows:

- \( F(1^n) = \Pi_n \), where \( n \) is the length of the input \( x \) and \( \Pi_n \) is a common tissue P system for all inputs of length \( n \), with a distinguished input cell.
• $E(x) = w_x$, where $w_x$ is a multiset encoding the specific input $x$.

• Finally, $\Pi_x$ is simply $\Pi_n$ with $w_x$ added to its input cell.

Instead, $\Pi$ is said to be (polynomial-time) semi-uniform if there exists a single deterministic polynomial-time Turing machine $H$ such that $H(x) = \Pi_x$ for each $x \in \Sigma^*$.

Any explicit encoding of $\Pi_x$ is allowed as output of the construction, as long as the number of cells and objects represented by it does not exceed the length of the whole description, the rules are listed one by one (this is a permissible encoding [6]), and the vertices where the cells are located in the initial configuration are specified.

The main technical result of [4] is that semi-uniform families of confluent tissue P systems over a polynomial-time navigable spatial graph having polynomial growth can be simulated by deterministic Turing machines with a polynomial slowdown. As a consequence, the sort of P systems that we may consider “realistic” cannot solve intractable problems in polynomial time: their computing power is limited to P.

3 Solving Q3SAT in polynomial time

We now show that if we drop the assumption that spatial graphs are bi-Lipschitz embeddable in $\mathbb{R}^d$, while retaining their polynomial-time navigability, then it is possible to design a uniform family $\Pi_{Q3SAT} = \{\Pi_x : x \in \Sigma^*\}$ of tissue P systems which solves any appropriately encoded instance $x$ of the problem $Q3SAT$ in polynomial time.

$Q3SAT$ is the well-known PSPACE-complete decision problem [2] which consists of determining whether the fully quantified version of a 3SAT formula (i.e., a Boolean formula in conjunctive normal form, in which each clause is composed of exactly three literals) evaluates to true or false. Formally, let $X_n = \{x_1, \ldots, x_n\}$ be a set of $n$ Boolean variables. An assignment for $X_n$ is a function $a : X_n \to \{false, true\}$ which maps each variable of $X_n$ to one of the two possible Boolean truth values. A literal over $X_n$ is either a variable $x_j \in X_n$ or its negation $\neg x_j$. A 3-clause is a disjunction ($\vee$) of three literals, where we assume that a variable or its negation cannot appear twice. Let $\mathcal{C}(n)$ be the set of all possible 3-clauses over $X_n$; since each 3-clause contains three different variables of $X_n$, and each of them may be negated or not, the set $\mathcal{C}(n)$ contains $m = 8 \binom{n}{3}$ elements. A 3SAT formula $\phi$ over $X_n$ is the conjunction ($\wedge$) of a set of 3-clauses taken from $\mathcal{C}(n)$. We denote by $3SAT(n)$ the set of all 3SAT formulas over $X_n$. Given $\phi \in 3SAT(n)$, let $\psi = \exists x_1 \exists x_2 \exists x_3 \ldots Q_n x_n \phi$ be the fully quantified version of $\phi$, where all odd-indexed (resp., even-indexed) variables are existentially (resp., universally) quantified; since $\psi$ has no free variables, it is either true or false. Let $Q3SAT(n)$ denote the fully quantified versions of formulas in 3SAT(n).

In what follows we denote by $\Pi_n$ the common tissue P system (without input) which is used to solve all possible instances of $Q3SAT(n)$, expressed in the form $\psi = \exists x_1 \forall x_2 \exists x_3 \ldots Q_n x_n \phi$. The actual P system $\Pi_x$ that solves the instance $\psi$ of $Q3SAT$ is obtained by encoding the Boolean formula $\phi$ as informally described below, and putting the resulting multiset into the input membrane of $\Pi_n$, in the initial configuration. The relevant graph of cells of $\Pi_x$ that will be used to solve $\psi$ will have
the structure of a full binary tree, as illustrated in Figure 1; in what follows we refer to it as the formula tree. The depth of such a tree will be $n$: each level $\ell \in \{1, \ldots, n\}$ (with the root being considered at level 1) will be associated with the Boolean variable $x_\ell$, and will be used to evaluate the corresponding universal or existential quantifier $Q_\ell$; the last level, indicated by $\phi$, will contain all possible evaluations of the Boolean formula $\phi$, for all possible assignments to variables $x_1, \ldots, x_n$.

The computation of $\Pi_x$ will consist of five phases. During the first phase, a particular cell of $\Pi_x$ that occurs at each used level of the spatial graph, called the controller, accumulates an appropriate number of “token” objects needed to drive the subsequent cell division process. In the second phase of computation, the cells of $\Pi_x$ divide according to the shape and free vertices of the spatial graph, provided that they possess one copy of the token object. During the third phase, the objects that will generate all possible assignments of variables $x_1, \ldots, x_n$, are sent from the root to the leaves of the formula tree. There, during the fourth phase of computation, the evaluation of $\phi$ over all possible assignments is performed, and the results are propagated back to the root of the tree, combined according to the universal and existential quantifiers associated to each level. Finally, the output object yes or no is sent to the environment, in the last step of computation.

The first four levels of the infinite spatial graph in which the P systems of $\Pi_{\text{Q3SAT}}$ live is illustrated in Figure 1. Each level is a full binary tree in which a “gadget” (illustrated in Figure 2, for the fourth level of the spatial graph) is attached to all internal vertices. The full binary tree at level $\ell$ has depth $\ell - 1$, and hence $2^{\ell-1}$ leaves. At level 1 there are no internal vertices, as the root is also a leaf, and thus there is no gadget. The gadget is used to drive the division process during the second phase of computation of $\Pi_x$, exploiting the token objects produced during the first phase.

The circled vertices in the spatial graph illustrated in the left side of Figure 1 constitute the initial cell structure of $\Pi_x$: for each level $\ell = 1, \ldots, n$ of the spatial graph there is one cell with label $h_\ell$, and one controller cell with label $k_\ell$ (but for the first level, which does not have a gadget and hence a controller). Cells $h_1, \ldots, h_n$ are located in the root vertex of the corresponding gadgets. Further, one copy of cell $h_\phi$ in
and a corresponding controller cell $k_\phi$ occur in the last level of the cell structure that will be generated. Each controller $k_{\ell}$, for $2 \leq \ell \leq n$, contains one copy of object $t_{\ell-1}$, while the controller $k_\phi$ contains one copy of object $t_n$.

In the first phase of computation, the controller of each level gathers from the environment enough copies of the token object $t$ that will be needed to perform the subsequent cell divisions:

\[
[t_i]_{k_{\ell}} \leftrightarrow t_{i-1}t_{i-1} \quad \text{for } \ell \in \{2, \ldots, n\} \text{ and } 2 \leq i \leq \ell - 1
\]

\[
[t_i]_{k_{\ell}} \leftrightarrow tt \quad \text{for } \ell \in \{2, \ldots, n\}
\]

\[
[t_i]_{k_\phi} \leftrightarrow t_{i-1}t_{i-1} \quad \text{for } 2 \leq i \leq \ell
\]

\[
[t_i]_{k_\phi} \leftrightarrow tt
\]

So doing, after $\ell - 1$ time steps cell $k_{\ell}$ will contain $2^\ell$ copies of object $t$, for $\ell = 2, \ldots, n$; similarly, cell $k_\phi$ will contain $2^{n+1}$ copies of $t$.

In the initial configuration of $\Pi_x$, each cell $h_{\ell}$ – but the one located in the root of the spatial graph – also contains one copy of object $d$. Such an object is used to request one copy of the token, than in the next time step activates cell division:

\[
[d]_{h_{\ell}} \leftrightarrow [t]_{k_{\ell}} \quad \text{for } \ell \in \{2, \ldots, n\} \cup \{\phi\}
\]

\[
[t]_{h_{\ell}} \rightarrow [d]_{h_{\ell}} [d]_{h_{\ell}} \quad \text{for } \ell \in \{2, \ldots, n\} \cup \{\phi\}
\]

Cells $h_{\ell}$ which are not connected with the controller $k_{\ell}$ will not be able to obtain the token object $t$ which is needed to divide. On the other hand, the cells $h_{\ell}$ which cannot divide because there are no free adjacent vertices in the spatial graph, will keep forever one copy of object $t$. This is the reason why, during the first phase of computation, we have produced two copies of the token object $t$ for each internal vertex at each layer of the cell structure. Since $n$ is the depth of the generated formula tree, in at most $2n$ steps the division process will terminate: in fact, at each step every cell occupies one of the free child vertices of the spatial graph, and at the next step it occupies the other one. At the end of the division process the formula tree of $\Pi_x$ is obtained,
These rules make use of objects $t_i$, occurring in all cells $h_j$, since the beginning of the computation, to ask for the object representing variable $x_i$ (currently located at the previous level of the formula tree) or its truth value ($f_i$ or $t_i$) if this has already been generated. Like before, object $f_{i,j}$ (resp., $t_{i,j}$) denotes the truth value false (resp., true) assigned to variable $x_i$, located at the $j$-th level of the formula tree.

When an object representing variable $x_i$ reaches the $i$-th level of the tree, the objects denoting its two possible truth values are generated:

$$[x_{i,i-1}]_{h_i} \leftrightarrow f_{i,i}t_{i,i}$$

for $1 \leq i \leq n$

On the other hand, the objects which represent truth values and still unassigned variables (i.e., variables not having yet reached the corresponding level) are duplicated, and the second index is updated according to the current level of the formula tree:

$$[x_{i,j-1}]_{h_j} \leftrightarrow x_{i,j}x_{i,j}$$
$$[t_{i,j-1}]_{h_j} \leftrightarrow t_{i,j}t_{i,j}$$
$$[f_{i,j-1}]_{h_j} \leftrightarrow f_{i,j}f_{i,j}$$

for all $1 \leq i, j \leq n$ such that $i > j$

for all $1 \leq i, j \leq n$ such that $i < j$
Finally, when the truth value objects reach the \( n \)-th level, the following rules bring them into the leaf cells of \( \Pi_x \) (which are also the leaves of the formula tree):

\[
\begin{align*}
  [\text{token}]_{i, \phi} & \leftrightarrow [f_{i,n}]_{h_\phi} \quad \text{for } 1 \leq i \leq n \\
  [\text{token}]_{t_{i,n}} & \leftrightarrow [t_{i,n}]_{h_\phi} \quad \text{for } 1 \leq i \leq n
\end{align*}
\]

In the next phase of the computation, the aim is to evaluate the Boolean formula \( \phi \) in all the leaf cells of \( \Pi_x \). The second index of the truth value objects occurring in those cells is updated, and simultaneously a single \textit{timer} object for each cell \( h_\phi \) is created:

\[
\begin{align*}
  [f_{i,n}]_{h_\phi} & \leftrightarrow f_{i,\phi} \quad \text{for } 1 \leq i < n \\
  [t_{i,n}]_{h_\phi} & \leftrightarrow t_{i,\phi} \quad \text{for } 1 \leq i < n \\
  [f_{n,n}]_{h_\phi} & \leftrightarrow f_{n,\phi} \quad \text{timer} \\
  [t_{n,n}]_{h_\phi} & \leftrightarrow t_{n,\phi} \quad \text{timer}
\end{align*}
\]

Each object \( f_{i,\phi} \) (resp., \( t_{i,\phi} \)) then produces the set \( C_{f_i} \) (resp., \( C_{t_i} \)) of objects which represent the 3-clauses that are satisfied by setting \( x_i = \text{true} \) (resp., \( x_i = \text{false} \)), independent from the values assumed by the other variables. Stated otherwise, \( C_{t_i} \) (resp., \( C_{f_i} \)) is the set of objects corresponding to the 3-clauses which contain the literal \( x_i \) (resp., \( \neg x_i \)). Meanwhile, the \textit{timer} object waits for one step, becoming \( \text{timer}' \):

\[
\begin{align*}
  [f_{i,\phi}]_{h_\phi} & \leftrightarrow C_{f_i} \\
  [t_{i,\phi}]_{h_\phi} & \leftrightarrow C_{t_i} \\
  [\text{timer}]_{h_\phi} & \leftrightarrow \text{timer}'
\end{align*}
\]

The evaluation of the Boolean formula \( \phi \) is now performed as follows. An assignment to variables \( x_1, \ldots, x_n \) satisfies \( \phi \) if and only if it satisfies all the 3-clauses that compose it. Given that cells \( h_\phi \) contain since the beginning of the computation all the objects that correspond to the 3-clauses which are \textit{not} part of \( \phi \), the assignment corresponding to a given leaf cell \( h_\phi \) will satisfy \( \phi \) if and only if the cell contains all the objects representing all possible clauses \( c_1, \ldots, c_m \). If this happens, then all these objects – together with the object \( \bullet \), which is present in each cell \( h_\phi \) since the beginning – are exchanged with object \( T_{\phi,\phi} \), which represents the truth value \textit{true} for \( \phi \), located in the last level of the formula tree:

\[
\bullet \{ c_1 c_2 \cdots c_m \}_{h_\phi} \leftrightarrow T_{\phi,\phi}
\]

Meanwhile, the timer object waits one more step, becoming \( \text{timer}'' \):

\[
[\text{timer}'']_{h_\phi} \leftrightarrow \text{timer}''
\]

Now \( \text{timer}'' \) seize the object \( \bullet \) if it is still present in cell \( h_\phi \), and brings in the object \( F_{\phi,\phi} \) which represents the truth value \textit{false} for \( \phi \) located in the last level of the formula tree. As a result, object \( F_{\phi,\phi} \) enters into the cell if and only if the corresponding assignment does \textit{not} satisfy the Boolean formula \( \phi \):

\[
\bullet \{ \text{timer}'' \}_{h_\phi} \leftrightarrow F_{\phi,\phi}
\]

Next, the following rules evaluate the universal and existential quantifiers associated with the boolean variables \( x_1, \ldots, x_n \), thus preparing the answer of the tissue P system.
Here and in what follows, $\lambda$ denotes the empty multiset. Precisely, the next two rules send all copies of objects $T_{\phi, 0}$ and $F_{\phi, 0}$ to the upper cells $h_n$:

$$[T_{\phi, 0}]_{h_0} \leftrightarrow [\lambda]_{h_n} \quad [F_{\phi, 0}]_{h_0} \leftrightarrow [\lambda]_{h_n}$$

There, the quantifier $Q_n$ is evaluated by the following rules, which basically compute a logical AND or a logical OR, depending upon whether $Q_n = \forall$ or $Q_n = \exists$, respectively:

$$[F_{\phi, 0} F_{\phi, 0}]_{h_n} \leftrightarrow F_{\phi, n} \quad \text{if } Q_n = \forall$$
$$[F_{\phi, 0} T_{\phi, 0}]_{h_n} \leftrightarrow F_{\phi, n} \quad \text{if } Q_n = \exists$$
$$[F_{\phi, 0} T_{\phi, 0}]_{h_n} \leftrightarrow T_{\phi, n}$$
$$[T_{\phi, 0} T_{\phi, 0}]_{h_n} \leftrightarrow T_{\phi, n}$$

The resulting objects $T_{\phi, n}$ and $F_{\phi, n}$ are sent to the previous level of the formula tree:

$$[T_{\phi, i+1}]_{h_{i+1}} \leftrightarrow [\lambda]_{h_i} \quad \text{for } 1 \leq i < n$$
$$[F_{\phi, i+1}]_{h_{i+1}} \leftrightarrow [\lambda]_{h_i} \quad \text{for } 1 \leq i < n$$

and the evaluation of quantifier $Q_i$ is performed as before by the following rules:

$$[F_{\phi, i+1} F_{\phi, i+1}]_{h_i} \leftrightarrow F_{\phi, i} \quad \text{for } 1 \leq i < n$$
$$[F_{\phi, i+1} T_{\phi, i+1}]_{h_i} \leftrightarrow F_{\phi, i} \quad \text{for all } 1 \leq i < n \text{ such that } Q_i = \forall$$
$$[F_{\phi, i+1} T_{\phi, i+1}]_{h_i} \leftrightarrow T_{\phi, i} \quad \text{for all } 1 \leq i < n \text{ such that } Q_i = \exists$$
$$[T_{\phi, i+1} T_{\phi, i+1}]_{h_i} \leftrightarrow T_{\phi, i} \quad \text{for } 1 \leq i < n$$

The communication and evaluation process is repeated until object $T_{\phi, 1}$ or object $F_{\phi, 1}$ is obtained in cell $h_1$; this object determines the answer of $\Pi_x$, expelled to the environment as one copy of object yes or one copy of object no, in the last computation step, by one of the following rules:

$$[F_{\phi, 1} \text{ no}]_{h_1} \leftrightarrow \lambda \quad \text{for } Q_n = \forall$$
$$[F_{\phi, 1} \text{ yes}]_{h_1} \leftrightarrow \lambda$$

The objects yes and no are given in the initial configuration of $\Pi_x$, and hence occur in cell $h_1$ since the beginning of the computation.

The spatial graph used by $\Pi_x$ is polynomial-time navigable. In fact, consider the labelling of vertices given in the right side of Figure 2, where the first element of the label indicates the level of the spatial graph where the vertex under consideration is located, whereas the second element provides a numbering of the vertices of the same level, according to the following criteria. Given the label $(\ell, p)$, with $p \neq k$ (that is, we are considering a vertex which does not contain a controller), then the parent vertex has index $\lfloor \frac{p}{2} \rfloor$ and the two children vertices have indexes $2p$ and $2p + 1$. The parent vertex exists if and only if $p > 1$, and the children vertices exist if and only if the vertex is not a leaf of the tree, that is, if and only if $p < 2^\ell - 1$; in such a case, the controller (having index $k$) is also an adjacent. The leaves $(\ell, p)$ of the gadget at level $\ell$, for which $2^\ell - 1 \leq p < 2^\ell$, are not connected to the controller $(\ell, k)$ but are instead connected (provided that $\ell > 1$) with one leaf vertex of the gadget at the previous level, having label $(\ell - 1, \lfloor \frac{k}{2} \rfloor)$, and with two leaf vertices of the next level, having labels $(\ell + 1, 2p)$ and $(\ell + 1, 2p + 1)$. In all the other cases, the Turing machine $M$ that navigates the graph rejects.
4 Conclusions

In [4] it has been proved that, when tissue P systems live in a spatial graph which is polynomial-time navigable and can be embedded into the Euclidean space \( \mathbb{R}^d \), only decision problems in \( \mathcal{P} \) can be solved in polynomial time. In this paper we have proved that, by dropping the embeddability condition, it is possible to design a uniform family \( \Pi_{\text{q3sat}} \) of tissue P systems which live in a polynomial-time navigable spatial graph, and solve the \text{PSPACE}-complete problem q3sat. This also proves that, by just changing the properties of the space in which the tissue P systems live, we can affect the computational power of the P systems themselves, going from \( \mathcal{P} \) (for spaces embeddable in \( \mathbb{R}^d \)) to \text{PSPACE} \ (for spaces non-embeddable in \( \mathbb{R}^d \)), thus identifying another parameter for obtaining frontiers of computational efficiency.

Interesting open questions, which ask for further research, are the following: what kind of features or constraints do we have to impose on spatial graphs to obtain tissue P systems precisely characterising intermediate complexity classes, such as \( \mathcal{P}^{\text{NP}} \) or \( \mathcal{P}^{\#\mathcal{P}} \)? What happens if the spatial graph is embeddable in the Euclidean space \( \mathbb{R}^d \), for some \( d \in \mathbb{N} \), but not polynomial-time navigable? Finally, we hypothesize that by using appropriately crafted spatial graphs one can design families of tissue P systems that solve even non recursively enumerable decision problems.

References