Nat Comput (2009) 8:431–447
DOI 10.1007/s11047-009-9120-5

From reaction-diffusion to Physarum computing

Andrew Adamatzky

Published online: 3 February 2009

Abstract We experimentally demonstrate that computation of spanning trees and implementation of general purpose storage-modification machines can be executed by a vegetative state of the slime mold *Physarum polycephalum*. We advance theory and practice of reaction-diffusion computing by studying a biological model of reaction-diffusion encapsulated in a membrane.

Keywords Reaction-diffusion computing · Biological computing · Spanning trees · Computational universality

1 Introduction: deficiencies of reaction-diffusion computers

In reaction-diffusion computers (Adamatzky 2001; Adamatzky et al. 2005), data is presented by an initial concentration profile or a configuration of disturbance (e.g., sites of stimulation of excitable media). The information is transferred by spreading wave patterns, computation is implemented in collisions of wave-fronts, and final concentration profile represents results of the computation. reaction-diffusion computers have been proved theoretically and experimentally capable for quite sophisticated computational tasks, including image processing and computational geometry, logics and arithmetics, and robot control (see Adamatzky et al. 2005 for detailed references, and overview of theoretical and experimental results). There is a particular feature of reaction-diffusion chemical computers: their classical, and so far commonly accepted form, the media are 'fully conductive' for chemical or excitation waves. Every point of a two- or three-dimensional medium can be involved in the propagation of chemical waves and reactions between diffusing chemical species. Once a reaction is initiated in a point, it spreads all over the computing space by target and spiral waves. Such phenomena of wave-propagation, analogues to one-to-all broadcasting in massive-parallel systems, are employed to solve problems ranging from the Voronoi diagram construction to robot navigation (Adamatzky 2001; Adamatzky et al. 2005). We could not,

A. Adamatzky (✉)
University of the West of England, Bristol BS16 1QY, UK
e-mail: andrew.adamatzky@uwe.ac.uk

however, quantize information (e.g., assign logical values to certain waves) or implement one-to-one transmission in fully reactive media.

The field of reaction-diffusion computing was started by Kuhnert, Agladze and Krinsky (Kuhnert 1986; Kuhnert et al. 1989), who, over 20 years ago, published their pioneering results on memory implementation and basic image processing in light-sensitive excitable chemical systems. Their ideas were further developed by Rambidi and colleagues (Rambidi 1998; Rambidi et al. 2002); and in Showalter and Yoshikawa's laboratories, who designed a range of chemical logical gates (Tóth and Showalter 1995; Kusumi et al. 1997). The computation of the shortest path, one of the classical optimization problems, has also been implemented in these laboratories using Belousov–Zhabotinsky media (Steinbock et al. 1995; Agladze et al. 1997; Adamatzky et al. 2005). Untill quite recently, the only way to direct and quantize information in a chemical medium was to geometrically constrain the medium. Thus, only reactive or excitable channels are made, along which waves can travel. The waves collide with other waves at the junctions between the channels and implement certain logical gates in result of the collision (see an overview in Chap. 1 of Adamatzky et al. 2005). Designs based on the geometrical constraining of the reaction-diffusion media are somewhat restricted by the conventionality of their architectures. This is because they simply re-enact standard computing architectures in non-standard 'conductive' materials.

Using sub-excitable media may be a successful way to quantize information. In sub-excitable media a local disturbance leads to the generation of mobile localization, where wave-fragments travel for a reasonably long distance without changing its shape (Sedina-Nadal et al. 2001). The presence of a wave-fragment in a given domain of space signifies logical truth, the absence of the fragment logical falsity (Adamatzky 2003). Despite being really promising candidates for collision-based computers (Adamatzky 2003), sub-excitable media are highly sensitive to experimental conditions and compact traveling wave-fragments are unstable and difficult to control.

In terms of well-established computing architectures, the following characteristics can be attributed to reaction-diffusion computers:

- massive-parallelism: there are thousands of elementary processing units, or microvolumes, in a standard Petri dish (Adamatzky et al. 2005);
- local connections: microvolumes of a non-stirred chemical medium change their states (due to diffusion and reaction) depending on states of (concentration of reactants in) their closest neighbours;
- parallel input and output: in chemical reactions with indicators concentration profiles of the reagents, one can allow for parallel optical output. There is also a range of light-sensitive chemical reactions where data can be inputted via local disturbances of illumination (Adamatzky et al. 2005);
- fault-tolerance: being in liquid phase, chemical reaction-diffusion computers do restore their architecture even after substantial part of the medium is removed, however, the topology and the dynamics of diffusive and, particularly, phase waves (e.g., excitation waves in Belousov–Zhabotinsky system) may be affected.

Reaction-diffusion computers—when implemented in chemical medium—are much slower than silicon-based massively-parallel processors. When nano-scale materials are employed, e.g., networks of single-electron circuits (Adamatzky et al. 2005), reaction-diffusion computers can however outperform even the most advanced silicon analogues.

There still remains a range of problems where chemical reaction-diffusion processors could not cope with without the external support from conventional silicon-based computing devices. The shortest path computation is one of such problems.

One can use excitable media to outline a set of all collision-free paths in a space with obstacles (Adamatzky and De Lacy Costello 2002), but to select and visualize the shortest path amongst all possible paths, one needs to use an external cellular-automaton processor, or conceptually supply the excitable chemical media with some kind of field of the local pointers (Adamatzky and De Lacy Costello 2002). Experimental setups (Steinbock et al. 1995; Agladze et al. 1997), which claim to directly compute a shortest path in chemical media, are indeed employing external computing resources to store time-lapsed snapshots of propagating wave-fronts and to analyse the dynamics of the wave-front propagation. Such usage of external resources dramatically reduces the fundamental values of the computing with propagating patterns.

Graph-theoretical computations pose even more difficulties for spatially-extended non-linear computers. For example, one can compute the Voronoi diagram of a planar set, but cannot invert this diagram (Adamatzky et al. 2005). Let us consider a spanning tree, most graph famous of classical proximity graphs. Given a set of planar points one wants to connect the points with edges, such that the resultant graph has no cycles and there is a path between any two points of the set. So far, no algorithms of spanning tree construction were experimentally implemented in spatially extended non-linear systems. This is caused mainly by uniformity of spreading wave-fronts, their inability to sharply select directions toward locations of data points, and also because excitable systems usually do not form stationary structures.

Essentially, to compute a spanning tree over a given planar set, a system must first explore the date space, then cover the data points and physically representing edges of the tree by the system's structure. This is not possible in excitable chemical systems because they are essentially memoryless, and no stationary structure can be formed. Precipitating reaction-diffusion systems are also uncapable of constructing the trees because not only they operate with uniformly expanding diffusive fronts, but the systems are incapable of altering concentration profile of the precipitate once precipitation occurred.

To overcome these difficulties, we should allow reaction-diffusion computers to be geometrically self-constrained while still capable to operate in geometrically unconstrained (architectureless or 'free') space. Encapsulating reaction-diffusion processes in membranes would a possible solution. The idea is explored in the present paper. Based on our previous results (Adamatzky 2007a, b, 2008a), we speculate that vegetative state, or plasmodium, of *Physarum polycephalum* is a reaction-diffusion system constrained by a membrane that capable for solving graph-theoretical problems (not solvable by 'classical' reaction-diffusion computers) and which is also computationally universal.

A brief introduction to computing with *Physarum polycephalum* is presented in Sect. 2. Section 3 introduces our experimental findings on constructing spanning trees of finite planar sets by plasmodium of *Physarum polycephalum*. In Sect. 4 we demonstrate that plasmodium of *Physarum polycephalum* is an ideal biological substrate for the implementation of Kolmogorov–Uspensky machines (Adamatzky 2007b). Directions of further studies are outlined in Sect. 5.

2 Physarum computing

There is a real-world system which strongly resembles encapsulated reaction-diffusion system. *Physarum polycephalum* is a single cell with many nuclei which behave like amoeba. In its vegetative phase, called plasmodium, slime mold actively searches for nutrients. When another source of food is located, plasmodium forms a vein of protoplasm between previous and current food-sources.

Why the plasmodium of Physarum is an analog of an excitable reaction-diffusion system enclosed in a membrane? Growing and feeding plasmodium exhibits characteristic rhytmic contractions with articulated sources. The contraction waves are associated with waves of potential change, and the waves observed in plasmodium (Matsumoto et al. 1986; Matsumoto et al. 1988; Yamada et al. 2007) are similar to the waves found in excitable chemical systems, like Belousov–Zhabotinsky medium. The following wave phenomena were discovered experimentally (Yamada et al. 2007): undisturbed propagation of contraction wave inside the cell body, collision and annihilation of contraction waves, splitting of the waves by inhomogeneity, and the formation of spiral waves of contraction (see Fig. 6c–f in Yamada et al. 2007). These are closely matching dynamics of pattern propagation in excitable reaction-diffusion chemical systems.

Yamada et al. (2007) indicate a possibility for interaction between the contraction force generating system with oscillating chemical reactions of calcium, ATP and associated pH (Ridgway and Durham 1976; Nakamura et al. 1982; Ogihara 1982). Chemical oscillations can be seen as primary oscillations—contraction waves are guided by chemical oscillation waves—because chemical oscillations can be recorded in absence of contractions (Oster and Odel 1984; Nakamura and Kamiya 1985; Nakagaki et al. 1999).

Nakagaki, Aono, Tsuda (Aono and Gunji 2001, 2004; Nakagaki 2001; Nakagaki et al. 2001; Tsuda et al. 2004, 2006) and others have been exploring a power of Physarum computing from 2000 (Nakagaki et al. 2000b). They proved experimentally that the plasmodium is a unique fruitful object to design various schemes of non-classical computation (Aono and Gunji 2001, 2004; Tsuda et al. 2004), including Voronoi diagram (Shirakawa and Gunji 2009) and shortest path (Nakagaki 2001; Nakagaki et al. 2001; Shirakawa and Gunji 2009), and even design of robot controllers (Tsuda et al. 2006). In present we paper we focus on one specialized instance of Physarum computing—approximation of spanning trees, and also implementation of a general purpose storage-modification machine.

The scoping experiments were designed as follows. We either covered container's bottom with a piece of wet filter paper and placed a piece of living plasmodium[1] on it, or we just planted plasmodium on a bottom of a bare container and fixed wet paper on the container's cover to keep the humidity high. Oat flakes were distributed in the container to supply nutrients and represent set of nodes to be spanned by a tree (Sect. 3) or to represent data-nodes of Physarum machine (Sect. 4). The containers were stored in the dark except during periods of observation. To color oat flakes, where required, we used SuperCook Food Colorings[2]: blue (colors E133, E122), yellow (E102, E110, E124), red (E110, E122), and green (E102, E142). The flakes were saturated with the colorings, then dried.

3 Approximation of spanning trees

The spanning tree of a finite planar set is a connected, undirected, acyclic planar graph, which vertices are points of the planar set; every point of the given planar set is connected to the tree (but no cycles or loops are formed). The tree is a minimal spanning tree where the sum of edges' lengths is minimal. Original algorithms for computing minimum

[1] Thanks to Dr. Soichiro Tsuda for providing me with *P. polycephalum* culture.

[2] http://www.supercook.co.uk.

spanning trees are described in Kruskal (1956), Prim (1957) and Dijkstra (1959). Hundreds if not thousands of papers were published in last 50 years, mostly improving the original algorithms, or adapting them to multi-processor computing systems (Gallager et al. 1983; Ahuja and Zhu 1989; Huang 1990).

Non-classical and nature-inspired computing models brought their own solutions to the spanning tree problem. Spanning tree can be approximated by random walks, electrical fields, and even social insects (Chong 1993; Lyons and Peres 1997; Adamatzky 2001; Adamatzky and Holland 2002). However neither of these non-classical algorithms offer an experimental realization.

In 1991 we proposed an algorithm of computing the spanning tree of a finite planar set based on the formation of a neurite tree in a development of a single neuron (Adamatzky 1991). Our idea was to place a neuroblast somewhere on the plane amongst drops of chemical attractants, positions of which represent points of a given planar set. Then, the neurite tree starts to grow and spans the given planar set of chemo-attractants with acyclic graph of axonal and dendritic branches. Due to lateral circumstances experimental implementation of the algorithm was not possible at the time of its theoretical investigation (Adamatzky 1991). Recent experimental developments in foraging behaviour of *P. Polycephalum* (Aono and Gunji 2001, 2004; Nakagaki 2001; Nakagaki et al. 2001; Tsuda et al. 2004, 2006) convinced us that our original algorithm for morphological growing of spanning trees can be implemented by living plasmodium.

When computing the spanning tree, the plasmodium acts as follows: once placed in the container, where oat flakes represent given planar set to be spanned by a tree, and recovered, the plasmodium starts to explore the surrounding space. Numerous pseudopodia emerge, frequently branch and proceed. The plasmodium grows from its initial position by protoplasmic pseudopodia detecting, by chemotaxis, relative locations of closest sources of nutrients. When another source of nutrients, element of the given planar set, is reached, the relevant part of the plasmodium reshapes and shrinks to a protoplasmic strand, or a tube. This tube connects the initial and the newly acquired sites. This protoplasmic strand represents an edge of the computed spanning tree. Planar points distributed in a Petri dish are usually spanned by a protoplasmic vein tree in 1–3 days, depending on the diameter of the planar set, substrate and other conditions. An example of a spanning tree approximated by plasmodium is shown in Fig. 1.

The tree computed by plasmodium in our experiments (Adamatzky 2008a) satisfactory match trees computed by classical techniques, e.g., by Jaromczyk–Supowit method (Jaromczyk and Kowaluk 1987; Supowit 1988, see Adamatzky 2008a). Even when represented in simulation, the algorithm works pretty well on large data sets (Adamatzky 2008a).

We would like to refer those eager for details to our previous papers (Adamatzky 2007a, 2008a), where the advantages of computing the spanning tree by pladmodium are discussed. In the present paper we will mention two speculative points of the approximation.

Plasmodium almost never computes the same (including exact location of protoplasmic edges) trees from the same data planar points. Not only the locations and configurations of the edges can be different, but also the topologies of the trees. An example is provided in Fig. 2. In one experiment plasmodium spans eastern and western data points while spreading North (Fig. 2a, c), while in another experiment plasmodium relocates to the northern part of the data set and then spreads back South (Fig. 2b, d).

Experimental results shown in Fig. 3 demonstrate that (1) tree can be constructed via other kinds of proximity graphs, or *k*-skeletons, and (2) plasmodium never stops 'computing', at some stage a tree is built, but then it is transformed to a planar graph with cycles. This experimental finding amazingly similar to how are spanning trees constructed

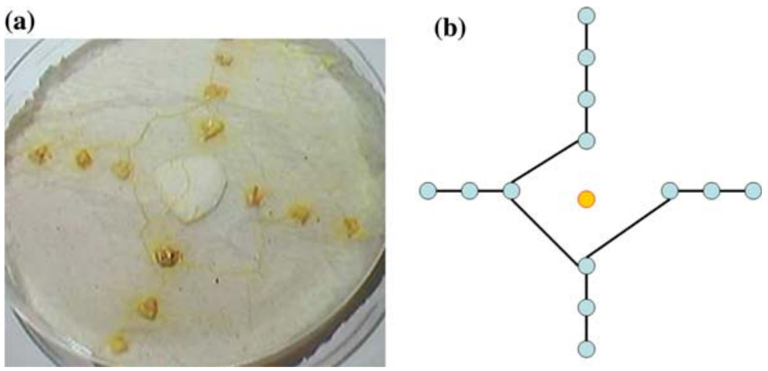

Fig. 1 Approximating spanning tree by plasmodium (**a**) photograph of living plasmodium in a container, where oat flakes represent the nodes of the tree; (**b**) scheme of the tree computed by plasmodium

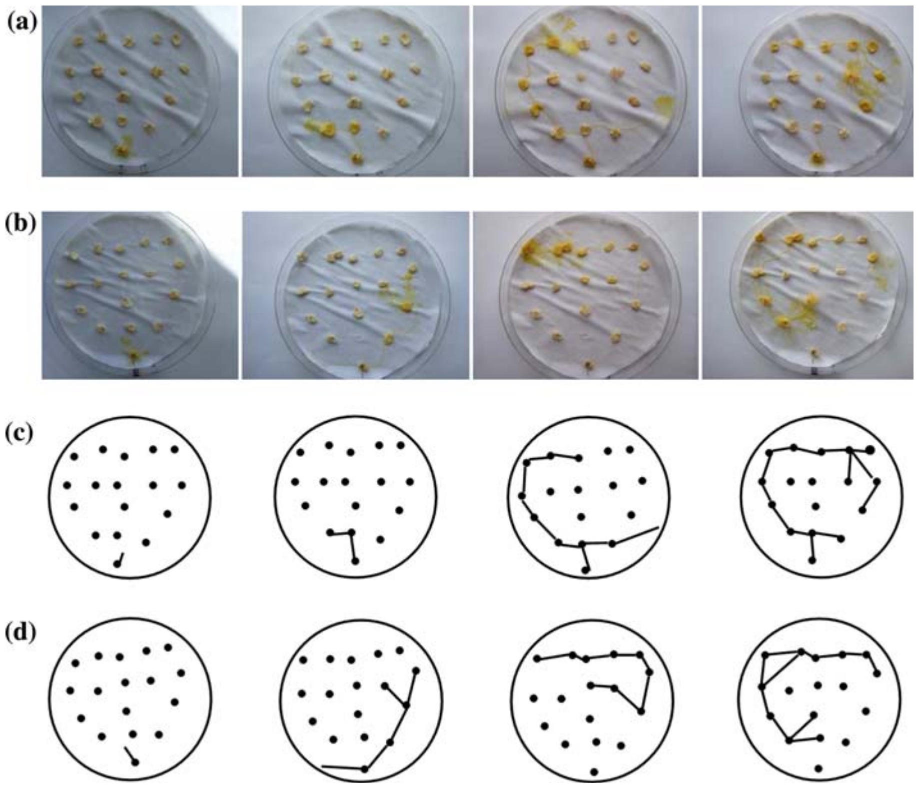

Fig. 2 Two scenarios of computing the spanning tree from the same planar data-points: (**a**, **b**) show photographs of the living plasmodium spanning oat flakes, which represent data nodes; (**c**, **d**) schemes of the trees approximated. At the beginning of both experiments, plasmodium was placed at the Southmost oat flake

on conventional computers—first a relative neighbourhood graph is computed, then some edges are deleted, and thus the graph is transformed to a minimum spanning tree (Jaromczyk and Kowaluk 1987; Supowit 1988).

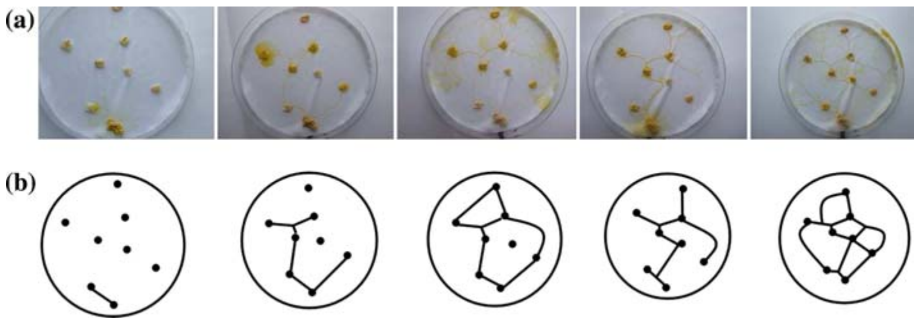

Fig. 3 Particular results of spanning planar data points by plasmodium (from the *left* to the *right*): first incomplete spanning tree is formed, then planar graph with cycles, then complete spanning tree; plasmodium continues its development after the tree is computed by transforming the tree again to a cyclic planar graph; (**a**) photographs of living plasmodium; (**b**) schemes of the graphs constructed

4 Universal Physarum machines

In the late 1940s and early 1950s, while developing his ideas on recursive functions and recursively enumerable sets (which are fundamentals of algorithm theory) (Uspensky 1992), Kolmogorov (Kolmogorov 1953; Kolmogorov and Uspensky 1958) established a formalism for algorithmic process realizable in physical time and space. He proposed that each state of an algorithm process is comprised of a finite number of elements and connections amongst them. Elements and connections belong to some types, and total number of types are bounded. Each connection has a fixed number of elements, and every element has a restricted number of connections. A restricted number of connections means locality in a sense that graphs connectivity is several orders less then the size of the graph. The state has a local active zone (i.e., specified elements) and connections amongst the elements can be updated dynamically. In computer science, Kolmogorov machine is treated as a computational device whose storage can change its topology. Later Kolmogorov's formalism was enriched by Uspensky, thus the name of the final abstract computational device.

A Kolmogorov–Uspensky machine (KUM) (Kolmogorov 1953; Kolmogorov and Uspensky 1958) is defined on a colored/labeled undirected graph with bounded degrees of nodes and bounded number of colors/labels. As Uspenski poetically said an algorithmic process "... can be regarded as a finite first-order structure of a finite signature, the signature being fixed for every particular algorithm" (Uspensky 1992).

KUM operates, and modifies its storage as following: select an active node in the storage graph. Specify local active zone, the node's neighborhood. Modify the active zone, i.e., add a new node with the pair of edges, then connect the new node with the active node; delete a node with the pair of incident edges; add or delete edges between the nodes.

A program for KUM specifies how to replace the neighborhood of an active node with new neighborhood, depending on labels of edges connected to the active node and labels of the nodes in the proximity of the active node (Blass and Gurevich 2003). All previous and modern models of real-world computation are heirs of KUM: Knuth's linking automata (Knuth 1968), Tarjan's Reference Machines (Tarjan 1977), Schönhage's storage modification machines (Schönhage 1973, 1980) (Fig. 4). When the restrictions on bounded in- and out-degrees of the machine's storage graph are lifted, the machine becomes Random Access Machine.

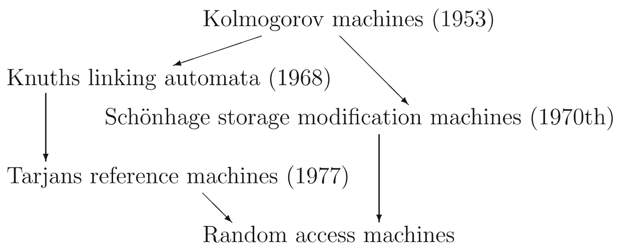
Kolmogorov machines (1953)
Knuths linking automata (1968)
Schönhage storage modification machines (1970th)
Tarjans reference machines (1977)
Random access machines

Fig. 4 Development of storage modification machines

Functions computable on Turing machines (TM) are also computed on KUM, and any sequential device are simulated by KUM (Gurevich 1988). KUM can simulate TM in real time, but not vice versa (Grigoriev 1976). KUM's topology is much more flexible than that of TM, and KUM is also stronger than any 'tree-machine' (Shvachko 1991).

In 1988 Gurevich (1988), suggested that an edge of KUM is not only an informational but also a physical entity and reflects the physical proximity of the nodes (thus e.g., even in three-dimensional space number of neighbors of each node is polynomially bounded). A TM formalizes computation as performed by *humans* (Turing 1936), whereas KUM formalizes computation as performed by *physical process* (Blass and Gurevich 2003).

What would be the best natural implementation of KUM? A potential candidate should be capable for growing, unfolding, graph-like storage structure, dynamically manipulating nodes and edges, and should have a wide range of functioning parameters. Vegetative stage, i.e., plasmodium, of a true slime mold *Physarum polycephalum* satisfies all these requirements.

Physarum machine has two types of nodes: stationary nodes, presented by sources of nutrients (oat flakes), and dynamic nodes, sites where two or more protoplasmic veins originate. At the beginning of the computation, the stationary nodes are distributed in the computational space, and the plasmodium is placed at one point of the space. Starting in the initial conditions, the plasmodium exhibits foraging behavior, and occupies stationary nodes.

An edge of Physarum machine is a strand, or vein, of a protoplasm connecting stationary and/or dynamic nodes. KUM machine is an undirected graph, i.e., if nodes x and y are connected, then they are connected by two edges (xy) and (yx). In Physarum machine this is implemented by a single edge but with periodically reversing flow of protoplasm (Kamiya 1950; Nakagakia et al. 2000a).

Program and data are represented by a spatial configuration of stationary nodes. Result of the computation over stationary data-node is presented by a configuration of dynamical nodes and edges. The initial state of a Physarum machine, includes part of input string (the part which represents position of plasmodium relatively to stationary nodes), an empty output string, a current instruction in the program, and a storage structure consists of one isolated node. That is, the whole graph structure developed by plasmodium is the result of its computation, "if S is a terminal state, then the connected component of the initial vertex is considered to be the 'solution'" (Kolmogorov and Uspensky 1958). Physarum machine halts when all data-nodes are utilized.

In KUM, a storage graph must have at least one active node. This is an inherent feature of Physarum machines. When the plasmodium resides on a substrate with poor or no nutrients, then just one or few nodes generate actively spreading protoplasmic waves. In these cases, the protoplasm spreads as mobile localizations similar to wave-fragments in sub-excitable Belousov–Zhabotinsky media (Sedina-Nadal et al. 2001). An example of a single active node, which has just started to develop its active zone, is shown in Fig. 5. At every step of computation there is an active node and an active zone, usually nodes neighboring to active node. The active zone has limited complexity, in a sense that all elements of the zone are connected by some chain of edges to the initial node. In general, the size of an active zone may vary depending on the computational task. In Physarum machine an active node is a trigger of contraction/excitation waves, which spread all over the plasmodium tree and cause pseudopodia to propagate, change their shape, and even protoplasmic veins to annihilate. An active zone is comprised of stationary or dynamic nodes connected to an active node with veins of protoplasm.

KUM, in its original form, have a single control device (Gurevich 1988; Blass and Gurevich 2003). Plasmodium acts as a unit in a long-term, i.e., it can change position, or retract some processes in one place to form new ones in another place. However, periodic contractions of the protoplasm are usually initiated from a single source of contraction waves (Nakagakia et al. 2000a; Tero et al. 2005). The source of the waves can be interpreted as a single control unit. In some cases we experimentally observed (Fig. 6) presence of a single active zone in the growing plasmodium. However, during foraging behavior, several branches or processes of plasmodium can act independently and almost autonomously.

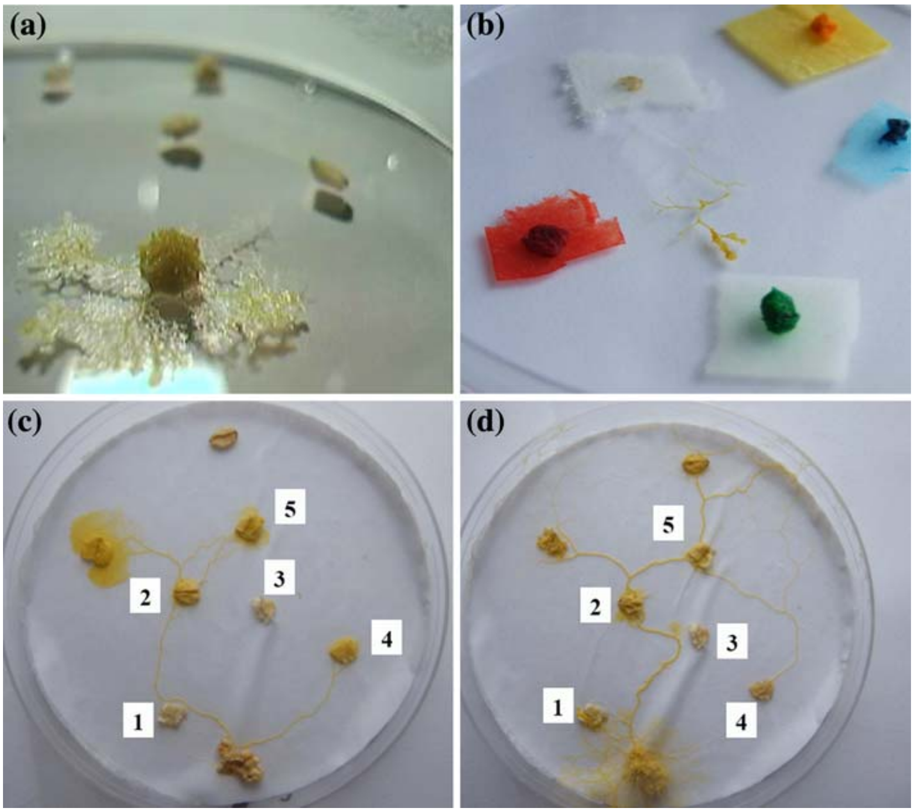

Fig. 5 Basic operations of Physarum machine: (**a**) a single active node generates an active zone at the beginning of computation; (**b**) addressing of a green-coloured data-node; (**c**, **d**) implementation of ADD NODE (nodes 3), ADD EDGE (edge (5,4)), REMOVE EDGE (edge (*route*, 4)) operations

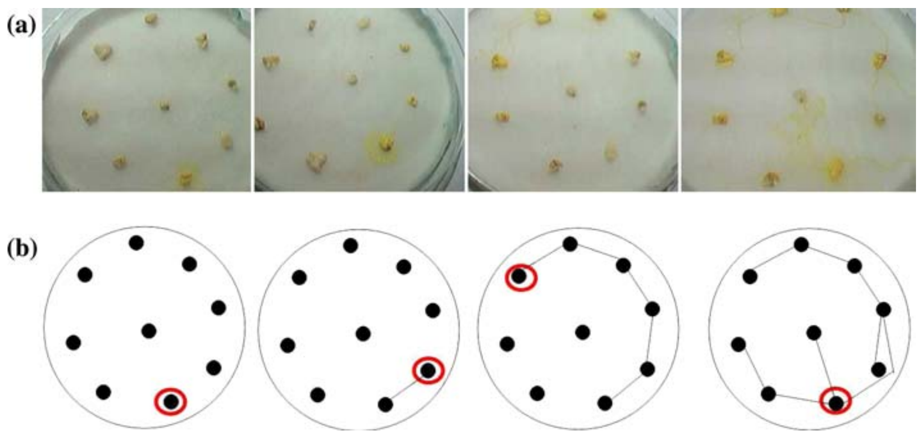

Fig. 6 Serial approximation of spanning tree by plasmodium: (**a**) snapshots of the living plasmodium, from the *left* to the *right*, made with 6 h intervals; (**b**) scheme of the graph. The active zone at each step of computation is encircled

In contrast to Schönhage machine, KUM has bounded in- and out-degree of the storage graph. Graphs developed by Physarum are predominantly planar graphs. Moreover, if we put a piece of vein of protoplasm on top of another vein of protoplasm, the veins fuse (Shirakawa 2007). Usually, not more than three protoplasmic strands join each other in one given point of space. Therefore we can assume that the average degree of the storage graph in Physarum machines is slightly higher then the degree of a spanning tree (average degree of 1.9 as reported in Cartigny et al. 2005) but smaller than the average degree of a random planar graph (degree 4, Alber et al. 2001).

Every node of KUM must be uniquely addressable and nodes and edges must be labeled (Kolmogorov and Uspensky 1958). There is no direct implementation of such addressing in Physarum machine. With stationary nodes this can be implemented, for example, by coloring the oat flakes. An example of such experimental implementation of a unique node addressing is shown in Fig. 5b.

A possible set of instructions for Physarum machine could be as follows: common instructions would include INPUT, OUTPUT, GO, HALT, and internal instructions: NEW, SET, IF (Dexter et al. 1997). At the present state of the experimental implementation, we assume that INPUT is done via distribution of sources of nutrients, while OUTPUT is recorded optically. The instruction SET causes pointers redirection, and can be realized by a placing a fresh source of nutrients in the experimental container, preferably on top of one of the old sources of nutrients. When a new node is created, all pointers can be redirected from the old node to the new node. Let us look at the experimental implementation of the core instructions.

To add a stationary node *b* to node *a*'s neighborhood, the plasmodium must propagate from *a* to *b*, and form a protoplasmic vein representing the edge (*ab*). To form a dynamic node, propagating the pseudopodia must branch into two or more pseudopodia, and the site of branching will represent the newly formed node. We have also obtained experimental evidence that dynamic nodes can be formed when a tip of growing pseudopodia collides with existing protoplasmic strand. In some cases merging of protoplasmic veins occur.

To remove the stationary node from Physarum machine, the plasmodium leaves the node. Annihilating protoplasmic strands, which form a dynamic node at their intersection, remove the dynamic node from the storage structure of Physarum machine.

To add an edge to a neighborhood, an active node generates propagating processes, which establish a protoplasm vein with one or more neighboring nodes.

When a protoplasmic vein annihilates, e.g., depending on the global state or when source of nutrients exhausted, the edge represented by the vein is removed from Physarum machine (Fig. 5c, d). The following sequence of operations is demonstrated in Fig. 5c, d: node 3 is added to the structure by removing edge (12) and forming two new edges (13) and (23).

Let us consider an example of a task solvable by Physarum machine. Given nodes labeled Red, Green, Blue, Yellow, connect only Green, Blue and Yellow nodes in a single chain. Physarum machine solves the task by first exploring the whole data space, then connecting required nodes (Fig. 7).

A possible compromise between the original theoretical framework and the partly parallel execution in experiments could be reached by proposing two levels of 'biological' commands executed by Physarum machine's elements. There would have to be high-level commands, e.g., search for nutrients, escape light, form sclerotium, fructify, and low-level commands, e.g., form process, propagate in direction of, occupy source of nutrients, retract process, branch. Global commands are executed by plasmodium as a whole at once, i.e., in a given time step the plasmodium executes only one high-level command.

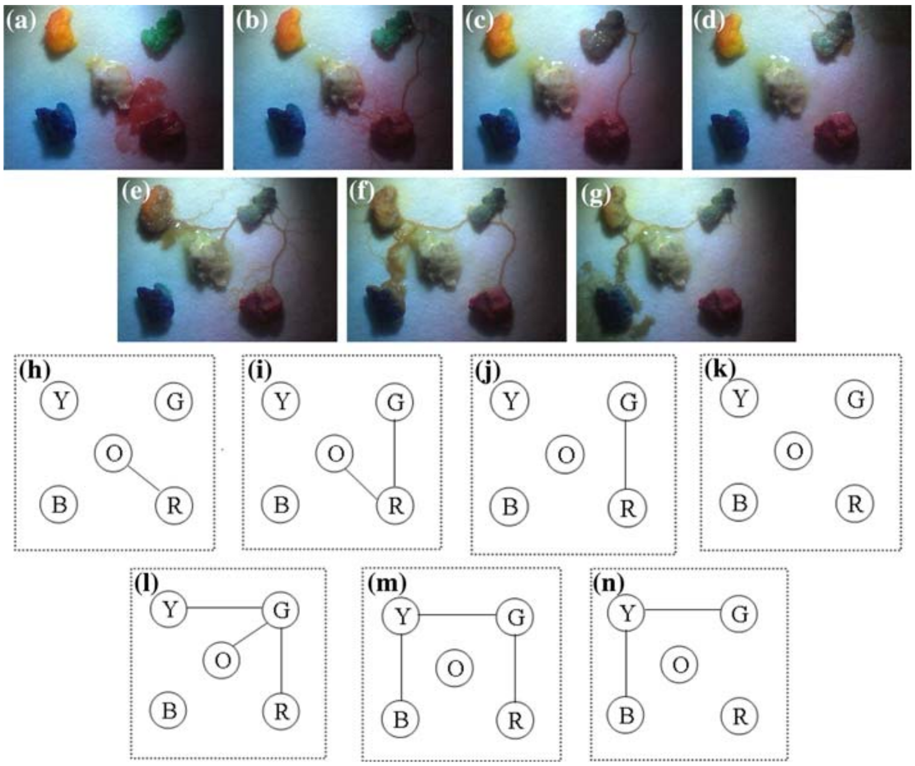

Fig. 7 Implementation of a simple task of connecting coloured nodes by Physarum machine: (**a**–**g**) shows a sequence of photographs of plasmodium (magnification ×10) placed in a small container in the centre of a rectangle, which corners represented by oat flakes that are coloured in yellow, green, red, and blue; the scheme of the computation is shown in (**h**–**n**)

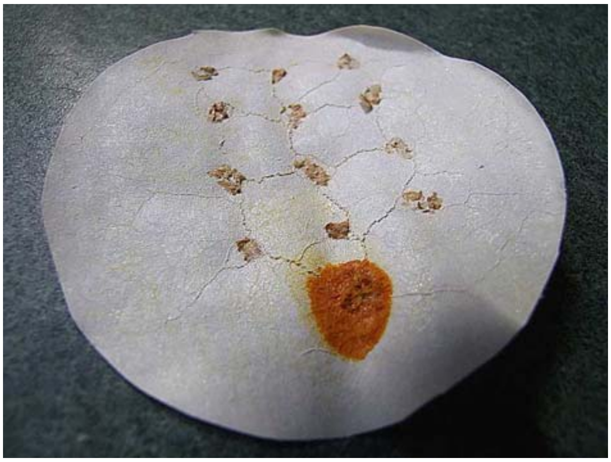

Fig. 8 Computation in Physarum is canceled by lowering humidity. Dark-brown coloured sclerotium is formed. You can also see empty and dead protoplasmic tubes, which formed previously active proximity graph spanning food sources

Local commands are executed by local parts of the plasmodium. Two spatially distant sites can execute different low-level commands at the same time.

One of the referees questioned what would be the implementation of the HALT command. So far we do not have any. The plasmodium continues its developed and colonizes spaces even when e.g., a spanning tree is completed. We can however 'froze' the computation by depriving plasmodium from water. In a low humidity conditions the plasmodiums stops its foraging behaviour and forms a sclerotium, a compact mass of hardened protoplasmic mass, see Fig. 8. Results of the computation are not destroyed and remain detectable as 'empty/dead' protoplasmic tubes. In the state of sclerotium the Physarum machine is ready for further deployment.

5 Discussions

Up to date, there were three types of reaction-diffusion computers with respect to the geometry of reactor space and reaction wave propagation.

First, unconstrained reaction-diffusion computers: once locally perturbed target and spiral waves initiated, and they propagate everywhere. Computation can be performed at any point of the space, where travelling waves interact with each other. Such systems are massive parallel and successfully solve NP-complete problems of computational geometry, e.g., plane tessellation (Voronoi diagram), as well as robot guidance and navigation. The designs were also implemented in large-scale integrated circuits and possible nano-scale implementation in networks of single-electron oscillators were studied, see overview in Adamatzky et al. (2005). The drawback of unconstrained reaction-diffusion computers is that they cannot complete graph optimization tasks, such as spanning tree or shortest path, without help of external computing devices.

Second, geometrically constrained reaction-diffusion computers: the chemical medium resides only in channels and logical circuits are made of the channels, where computation happens at junctions between two or more channels, see e.g., Tóth and Showalter (1995), Sielewiesiuk and Górecki (2001), Górecki et al. (2003) and Motoike and Adamatzky (2005). The geometrically constrained systems can implement Boolean and multiple-valued logical operations, as well as count and process sequences of signals. There are two deficiencies: (1) geometrically constrained reaction-diffusion computers are essentially only implementations of conventional computing architectures with wires and logical switches in novel chemical materials; and (2) the intrinsic parallelism of the medium is not utilized properly.

Third, reaction constrained reaction-diffusion computers: travelling waves can be initiated and then propagate anywhere in the reaction space. However due to low reactivity of the system, no classical (e.g., target) waves are formed. This is typical for sub-excitable Belousov–Zhabotinsky systems. Local perturbation leads to formation of compact wave-fragments which can travel for reasonable distance preserving its shape, see e.g., Sedina-Nadal et al. (2001). The system is an ideal implementation of collision-based computing schemes (Adamatzky 2003). The deficiency of the system is that travelling compact wave-fragment are very sensitive to conditions of the medium, and therefore are cumbersome to control.

To overcome all these deficiencies of reaction-diffusion computers, we suggested to encapsulate the reaction-diffusion systems in a membrane because this seems to be a good combination of unconstrained physical space and membrane-constrained geometry of propagating patterns.[3] Also, a system encapsulated in an elastic or contractable membrane would be capable of reversible shape changing, a feature unavailable in reaction-diffusion chemical systems. We demonstrated that the vegetative state, the plasmodium, of slime mold *Physarum polycephalum*, is an ideal biological medium for suggested implementation. We have provided first experimental evidences that plasmodium can compute spanning trees of finite planar sets and implement Kolmogorov–Uspensky machines. Thus, Physarum computers can solve graph-theoretic problems and are capable for universal computation.

Physarum computers will be particularly efficient in solving large-scale graph and network (including telecommunication and road traffic networks) optimization tasks, and can also be used as embedded controllers for non-silicon (e.g., gel-based) reconfigurable robots and manipulators. They are reasonably robust (they live on almost any non-aggressive substrate including plastic, glass and metal foil, in a wide range of temperatures, they do not require special substrates or sophisticated equipment for maintenance) and are programmable (plasmodium exhibits negative phototaxis, and can follow gradients of humidity and some chemo-attractants) spatially extended and, distributed, computing devices.

In contrast to ‘classical’ chemical reaction-diffusion computers (Adamatzky et al. 2005), Physarym machines can function on virtually any biologically non-agressive substrate, include metal and glass. Moreover, the substrate does not have to be static. For example, to implement Physarum machines with mobile data nodes, one can use a container with water, place the plasmodium of Physarum on one floating object, and the oat flakes (data) on several other floating objects. Plasmodium will then explore the physical space, travelling on the surface of the water, and eventually set up connections between

[3] Recently we have demonstrated in chemical and biological laboratory experiments that plasmodoium of *Physarun polycephalum* behaves almost exactly the same, apart of leaving a ‘trace’, as excitation patterns in *sub-excitable* Belousov–Zhabotinsky medium, see details in Adamatzky et al. (2009). Thus plasmodium is also proved to be capable for collision-based universal computation (Adamatzky 2003).

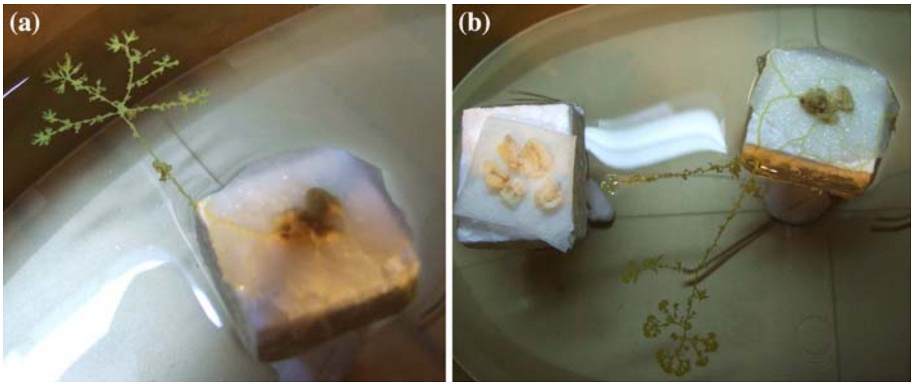

Fig. 9 A floating Physarum machine: (**a**) an active zone of Physarum machine travelling on water surface; (**b**) a connection is formed between the active zone and one of the data points

data points (Fig. 9). First steps towards Physarum robots are reported at Adamatzky et al. (2008c).

Future research will concentrate on expanding the domain of graph-theoretic tasks solvable by Physarum computers, developing programming language for Physarum machines, design and experimental implementation of plasmodium-based intelligent manipulators, and general purpose logical and arithmetical circuits.

6 Breaking news

The present paper was written almost one year and a half ago. During this time exciting developments occurred and significant findings related to Physarum computers and robots were established. We have no chance to discuss new results in details. So below we just provide a brief sketch of hot new findings:

- In Adamatzky (2008b) we experimentally evaluated a possibility that the plasmodium constructs a series of proximity graphs: nearest-neighbour graph (NNG), minimum spanning tree (MST), relative neighborhood graph (RNG), Gabriel graph (GG) and Delaunay triangulation (DT). The graphs can be arranged in the inclusion hierarchy (Toussaint hierarchy): NNG $\subseteq$ MST $\subseteq$ RNG $\subseteq$ GG $\subseteq$ DT. We verified that these graphs—when nodes are sources of nutrients and edges are protoplasmic tubes—appear in the development of the plasmodium in the order NNG $\rightarrow$ MST $\rightarrow$ RNG $\rightarrow$ GG $\rightarrow$ DT, corresponding to inclusion of the proximity graphs.
- In Shirakawa and Gunji (2009) we experimentally demonstrated that both Voronoi diagram and its dual graph Delaunay triangulation are simultaneously constructed—in certain conditions—by the plasmodium. Every point of given planar data set is represented by a tiny mass of plasmodium. The plasmodia spread from their initial locations but, in certain conditions, stop spreading when encounter plasmodia originated from different locations. Thus space loci not occupied by the plasmodia represent edges of Voronoi diagram of the given planar set. At the same time, the plasmodia originating at neighbouring locations form merging protoplasmic tubes, where the strongest tubes approximate Delaunay triangulation of the given planar set.

- In laboratory experiments and simulation we demonstrate how the plasmodium-based storage modification machine can be programmed (Adamatzky and Jones 2009). We shown execution of the following operations with active zone (where computation occurs): merge two active zones, multiple active zone, translate active zone from one data site to another, direct active zone.
- In Adamatzky (2009) we designed and tested in real-world experiments first ever plasmodium robot. We shown that when adhered to light-weight object resting on a water surface the plasmodium can propel the object by oscillating its protoplasmic pseudopodia. In experimental laboratory conditions and computational experiments we studied phenomenology of the plasmodium-floater system, and possible mechanisms of controlling motion of objects propelled by on board plasmodium.

Acknowledgments Many thanks to Dr. Christof Teuscher (Los Alamos Labs, US) for editing the manuscript. I am grateful to Dr. Soichiro Tsuda (Southmapton Univ, UK) for providing me with the culture of *Physarum polycephalum* and subsequent fruitful discussions.

References

Adamatzky A (1991) Neural algorithm for constructing minimal spanning tree. Neural Netw World 6: 335–339

Adamatzky A (2001) Computing in non-linear media and automata collectives. IoP Publishing, Bristol, 401 pp

Adamatzky A (ed) (2003) Collision-based computing. Springer, London

Adamatzky A (2007a) Physarum machines: encapsulating reaction-diffusion to compute spanning tree. Naturwisseschaften 94:975–980

Adamatzky A (2007b) Physarum machine: implementation of a Kolmogorov–Uspensky machine on a biological substrate. Parallel Process Lett 17:455-467

Adamatzky A (2008a) Growing spanning trees in plasmodium machines. Kybern Int J Syst Cybern 37:258–264

Adamatzky A (2008b) Developing proximity graphs by *Physarum polycephalum*: does the plasmodium follow Toussaint hierarchy? Parallel Process Lett 19:105–127

Adamatzky A (2008c) Towards Physarum robots: computing and manipulating on water surface. arXiv:0804.2036v1 [cs.RO]

Adamatzky A (2009) Physarum boats: if plasmodium sailed it would never leave a port (submitted)

Adamatzky A, De Lacy Costello BPJ (2002) Collision-free path planning in the Belousov–Zhabotinsky medium assisted by a cellular automaton. Naturwissenschaften 89:474–478

Adamatzky A, Holland O (2002) Reaction-diffusion and ant-based load balancing of communication networks. Kybernetes 31:667–681

Adamatzky A, Jones J (2009) Programmable reconfiguration of Physarum machines (submitted)

Adamatzky A, De Lacy Costello B, Asai T (2005) Reaction-Diffusion Computers. Elsevier, Amsterdam

Adamatzky A, De Lacy Costello B, Shirakawa T (2009) Universal computation with limited resources: Belousov–Zhabotinsky and Physarum computers. Int J Bifurcat Chaos (in press)

Agladze K, Magome N, Aliev R, Yamaguchi T, Yoshikawa K (1997) Finding the optimal path with the aid of chemical wave. Physica D 106:247–254

Ahuja M, Zhu Y (1989) A distributed algorithm for minimum weight spanning tree based on echo algorithms. In: Proceedings of the international conference on distributed computing system, pp 2–8

Alber J, Dorn F, Niedermeier R (2001) Experiments on optimally solving NP-complete problems on planar graphs, manuscript. http://www.ii.uib.no/~frederic/ADN01.ps

Aono M, Gunji Y-P (2001) Resolution of infinite-loop in hyperincursive and nonlocal cellular automata: introduction to slime mold computing. in: Computing anticiaptory systems, AIP conference proceedings, vol 718, pp 177–187

Aono M, Gunji Y-P (2004) Material implementation of hyper-incursive field on slime mold computer. In: Computing anticipatory systems, AIP conference proceedings, vol 718, pp 188–203

Blass A, Gurevich Y (2003) Algorithms: a quest for absolute definitions. Bull Eur Assoc Theor Comput Sci 81:195–225

Cartigny J, Ingelrest F, Simplot-Ryl D, Stojmenovic I (2005) Localized LMST and RNG based minimum-energy broadcast protocols in ad hoc networks. Ad Hoc Netw 3:1–16

Chong F (1993) Analog techniques for adaptive routing on interconnection networks. MIT transit note no. 14

Dexter S, Doyle P, Gurevich Yu (1997) Gurevich abstract state machines and Schönhage storage modification machines. J Univers Comput Sci 3:279–303

Dijkstra EA (1959) A note on two problems in connection with graphs. Numer Math 1:269–271

Gallager RG, Humblet PA, Spira PM (1983) A distributed algorithm for minimum-weight spanning tree. ACM Trans Program Lang Syst 5:66–77

Górecki J, Yoshikawa K, Igarashi Y (2003) On chemical reactors that can count. J Phys Chem A 107:1664–1669

Grigoriev D (1976) Kolmogorov algorithms are stronger than turing machines. Notes Sci Semin LOMI 60:29–37 (in Russian). English translation: J Sov Math 14:1445–1450 (1980)

Gurevich Y (1988) On Kolmogorov machines and related issues. Bull Eue Assoc Theor Comput Sci 35:71–82

Huang ST (1990) A fully pipelined minimum spanning tree constructor. J Parallel Distrib Comput 9: 55–62

Jaromczyk JW, Kowaluk M (1987) A note on relative neighborhood graphs. In: Proceedings of the 3rd annual symposium on computational geometry, pp 233–241

Kamiya N (1950) The protoplasmic flow in the myxomycete plasmodium as revealed by a volumetric analysis. Protoplasma 39:3

Knuth DE (1968) The art of computer programming, vol 1: fundamental algorithms. Addison-Wesley, Reading

Kolmogorov AN (1953) On the concept of algorithm. Usp Mat Nauk 8:175–176

Kolmogorov AN, Uspensky VA (1958) On the definition of an algorithm. Uspekhi Mat Nauk 13:3–28 (in Russian). English translation: ASM Transl 21:217–245 (1963)

Kruskal JB (1956) On the shortest subtree of a graph and the traveling problem. Proc Am Math Soc 7:48–50

Kuhnert L (1986) A new photochemical memory device in a light sensitive active medium. Nature 319:393

Kuhnert L, Agladze KL, Krinsky VI (1989) Image processing using light-sensitive chemical waves. Nature 337:244–247

Kusumi T, Yamaguchi T, Aliev R, Amemiya T, Ohmori T, Hashimoto H, Yoshikawa K (1997) Numerical study on time delay for chemical wave transmission via an inactive gap. Chem Phys Lett 271:355–360

Lyons R, Peres Y (1997) Probability on trees and networks. http://www.mypage.iu.edu/~rdlyons/prbtree/prbtree.html

Matsumoto K, Ueda T, Kobatake Y (1986) Propagation of phase wave in relation to tactic responses by the plasmodium of *Physarum polycephalum.* J Theor Biol 122:339–345

Matsumoto K, Ueda T, Kobatake Y (1988) Reversal of thermotaxis with oscillatory stimulation in the plasmodium of *Physarum polycephalum.* J Theor Biol 131:175–182

Motoike I, Adamatzky A (2005) Three-valued logic gates in reaction-diffusion excitable media. Chaos Solitons Fractals 24:107–114

Nakagaki T (2001) Smart behavior of true slime mold in a labyrinth. Res Microbiol 152:767–770

Nakagaki T, Yamada H, Ito M (1999) reaction-diffusion advection model for pattern formation of rhythmic contraction in a giant amoeboid cell of the *Physarum plasmodium.* J Theor Biol 197:497–506

Nakagaki T, Yamada H, Ueda T (2000a) Interaction between cell shape and contraction pattern in the *Physarum plasmodium.* Biophys Chem 84:195–204

Nakagaki T, Yamada H, Toth A (2000b) Maze-solving by an amoeboid organism. Nature 407:470

Nakagaki T, Yamada H, Toth A (2001) Path finding by tube morphogenesis in an amoeboid organism. Biophys Chem 92:47–52

Nakamura S, Kamiya N (1985) Regional difference in oscillatory characteristics of *Physarum plasmodium* as revealed by surface pH. Cell Struct Funct 10:173–176

Nakamura S, Yoshimoto Y, Kamiya N (1982) Oscillation in surface pH of the *Physarum plasmodium.* Proc Jpn Acad 58:270–273

Ogihara S (1982) Calcium and ATP regulation of the oscillatory torsional movement in a triton model of Physarum plasmodial strands. Exp Cell Res 138:377–384

Oster GF, Odel GM (1984) Mechanics of cytogels I: oscillations in Physarum. Cell Motil 4:469–503

Prim RC (1957) Shortest connection networks and some generalizations. Bell Syst Tech J 36:1389–1401

Rambidi NG (1998) Neural network devices based on reaction-duffusion media: an approach to artificial retina. Supramol Sci 5:765–767

Rambidi NG, Shamayaev KR, Peshkov GYu (2002) Image processing using light-sensitive chemical waves. Phys Lett A 298:375–382

Ridgway EB, Durham ACH (1976) Oscillations of calcium ion concentration in *Physarum plasmodia.* Protoplasma 100:167–177

Schönhage A (1973) Real-time simulation of multi-dimensional Turing machines by storage modification machines. Project MAC technical memorandum 37, MIT

Schönhage A (1980) Storage modification machines. SIAM J Comput 9:490–508

Sedina-Nadal I, Mihaliuk E, Wang J, Perez-Munuzuri V, Showalter K (2001) Wave propagation in subexcitable media with periodically modulated excitability. Phys Rev Lett 86:1646–1649

Shirakawa T (2007) Private communication, Feb 2007

Shirakawa T, Gunji Y-P (2009) Computation of Voronoi diagram and collision-free path using the Plasmodium of *Physarum polycephalum.* Int J Unconv Comput (in press)

Shirakawa T, Adamatzky A, Gunji Y-P, Miyake Y (2009) On simultaneous construction of Voronoi diagram and Delaunay triangulation by *Physarum polycephalum.* Int J Bifurcat Chaos (in press)

Shvachko KV (1991) Different modifications of pointer machines and their computational power. In: Proceedings of symposium of the mathematical foundations of computer science MFCS. Lect Notes Comput Sci, vol 520, pp 426–435

Sielewiesiuk J, Górecki J (2001) Logical functions of a cross junction of excitable chemical media. J Phys Chem A105:8189

Steinbock O, Toth A, Showalter K (1995) Navigating complex labyrinths: optimal paths from chemical waves. Science 267:868–871

Supowit KJ (1988) The relative neighbourhood graph, with application to minimum spanning tree. J ACM 3:428–448

Tarjan RE (1977) Reference machines require non-linear time to maintain disjoint sets, STAN-CS-77-603, March

Tero A, Kobayashi R, Nakagaki T (2005) A coupled-oscillator model with a conservation law for the rhythmic amoeboid movements of plasmodial slime molds. Physica D 205:125–135

Tóth A, Showalter K (1995) Logic gates in excitable media. J Chem Phys 103:2058–2066

Tsuda S, Aono M, Gunji Y-P (2004) Robust and emergent Physarum-computing. BioSystems 73:45–55

Tsuda S, Zauner KP, Gunji YP (2006) Robot control: from silicon circuitry to cells. In: Ijspeert AJ, Masuzawa T, Kusumoto S (eds) Biologically inspired approaches to advanced information technology. Springer, New York, pp 20–32

Turing A (1936) On computable numbers, with an application to the Entscheidungs-problem. Proc Lond Math Soc 42:230–265

Uspensky VA (1992) Kolmogorov and mathematical logic. J Symb Log 57:385–412

Yamada H, Nakagaki T, Baker RE, Maini PK (2007) Dispersion relation in oscillatory reaction-diffusion systems with self-consistent flow in true slime mold. J Math Biol 54:745–760