Encapsulating Reaction-Diffusion Computers

Andrew Adamatzky

Faculty of Computing, Engineering and Mathematical Sciences,
University of the West of England, Bristol BS16 1QY, United Kingdom
andrew.adamatzky@uwe.ac.uk

Abstract. Reaction-diffusion computers employ propagation of chemical and excitation waves to transmit information; they use collisions between traveling wave-fronts to perform computation. We increase applicability domain of the reaction-diffusion computers by encapsulating them in a membrane, in a form of vegetative state, plasmodium, of true slime mold. In such form reaction-diffusion computers can also realize Kolmogorov-Uspensky machine.

1 From Reaction-Diffusion Computers to Plasmodium

In reaction-diffusion computers [2,4] data are presented by initial concentration profile or configuration of disturbance (e.g. sites of stimulation of excitable media), information is transfered by spreading wave patterns, computation is implemented in collisions of wave-fronts, and final concentration profile represents results of the computation. Reaction-diffusion computers are theoretically and experimentally proved to be capable for quite sophisticated computational tasks, including image processing, computational geometry, logics and arithmetics, and robot control, see extensive overview of theoretical and experimental results in [4].

There is a particular feature of reaction-diffusion chemical computers. In 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 propagation of chemical waves and reactions between diffusing chemical species. Once reaction is initiated in a point, it spreads all over the computing space by target and spiral waves. Such, analogues to one-to-all broadcasting in massive-parallel systems, phenomena of wave-propagation are employed to solve problems ranging from Voronoi diagram construction to robot navigation [2,4]. We could not however quantize information (e.g. assign logical values to certain waves) or implement one-to-one transmission in fully reactive media.

Till 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 wave travel. The waves collide with other waves at the junctions between the channels, and implement certain logical gates in result of the collision, see overview in Chapter 1 of e.g. [4].

J. Durand-Lose and M. Margenstern (Eds.): MCU 2007, LNCS 4664, pp. 1–11, 2007.

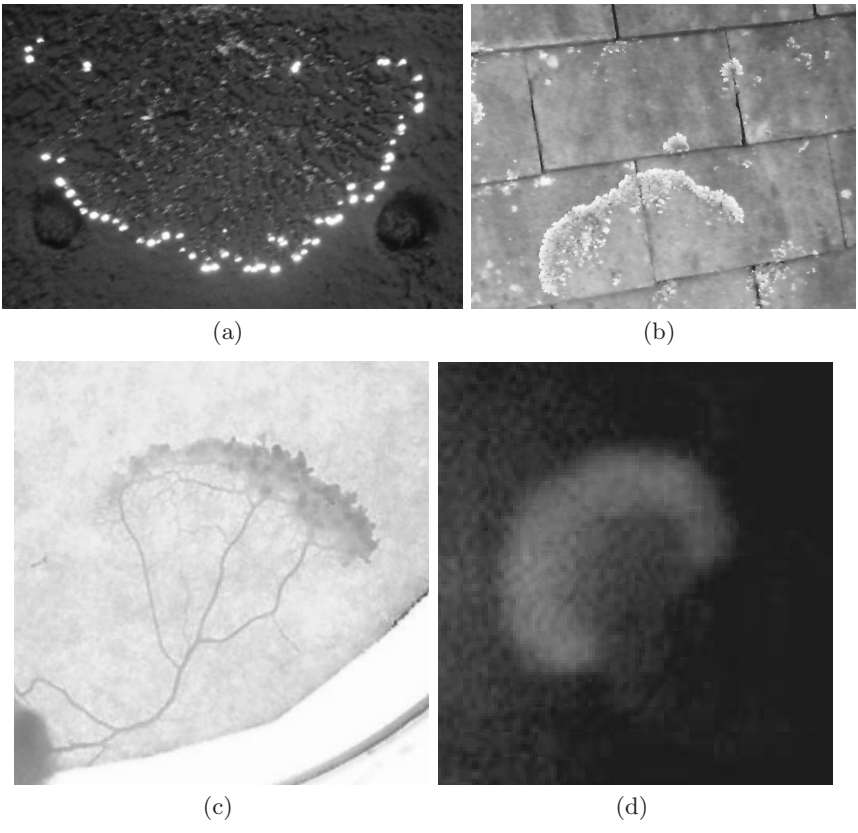

Fig. 1. Examples of localized propagations in real-world systems: (a) localized waves of combustion, (b) fragment of wave front of lichen colony, (b) propagating plasmodium, (c) wave-fragment in sub-excitable Belousov-Zhabotinsky system

Using sub-excitable media is yet another way of quantizing information. In sub-excitable media a local disturbance leads to generation of mobile localization, wave-fragment which travels for a reasonably long distance without changing its shape [36]. Presence of a wave-fragment in a given domain of space signifies logical truth, absence of the fragment logical falsity. A full power of collision-based computing can be applied then [3]. Such mobile localization are typical natural phenomena occurring in situations when system lacks resources to realize in full its development potential (Fig.1), e.g. experience deficiency of combustive material (Fig.1a), illumination (Fig.1b), nutrients in substate (Fig.1c), and excitability (Fig.1d).

Also there is a range of problems, where chemical processor could not cope without external support. Shortest path is one of such problems. One can use excitable medium to outline a set of all collision-free paths in a space with obstacles [4], but to select and visualize the shortest path amongst all possible one needs to use external cellular-automaton processor, or conceptually supply

excitable chemical medium with some kind of field of local pointers [4]. Experimental setups, e.g. [39] 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 calculate intersection of wave-fronts. Such usage of external resources dramatically reduce fundamental values of the computing with propagating patterns. 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 or standing waves.

Ideally, we would prefer to combine advantages of 'free space'[1] computing in fully reactive media with precision and simplicity of logical representation of geometrically constrained media. This can be done by encapsulating reaction-diffusion system in an elastic membrane.

There is a real-world system which strongly resembles encapsulated reaction-diffusion system. *P. Polycephalum* is a single cell with many nucleus which behave like amoeba, or even young neuroblast and this is why it play so perfectly role of computing substrate for our algorithm of growing spanning tree. In its main vegetative phase, called plasmodium, slime mold actively searches for nutrients. When next source of food is located plasmodium forms a vein of protoplasm between previous and current sources of food. Growing and feeding plasmodium exhibits characteristic rythmic contractions with articulated sources. The contraction waves are associated with waves of potential change, and the waves observed in plasmodium [48] are similar that found in excitable chemical systems, like Belousov-Zhabotinsky medium. The following wave phenomena were discovered experimentally [48]: undisturbed propagation of contraction wave inside the cell body, collision and annihilation of contraction waves, splitting of the waves by inhomogeneity, and formation of spiral waves of contraction. These are closely matching dynamics of pattern propagation in in excitable reaction-diffusion chemical systems.

The plasmodium has already proved to be a unique fruitful object to design various schemes of non-classical computation [10,11,45], including shortest path [31,31,33] and even design of controllers for robots [46].

In the paper we highlight novel aspects of our studies in computing with propagating localizations. Firstly, we demonstrate the spanning tree construction – the problem unsolvable in 'classical' reaction-diffusion computer without help of external hardware devices – can be solved in plasmodium of *P*hysarum polycephalum. Secondly, we outline a refreshing approach to universality of biological substrates by constructing Physarum machine, which is an experimental implementation of Kolmogorov-Uspensky machine.

2 Spanning Trees

In 1991 we proposed an algorithm of computing spanning tree of a finite planar set based on formation of a neurite tree in a development of a single neuron [1]. Our

[1] Thanks to Jonathan Mills for the term.

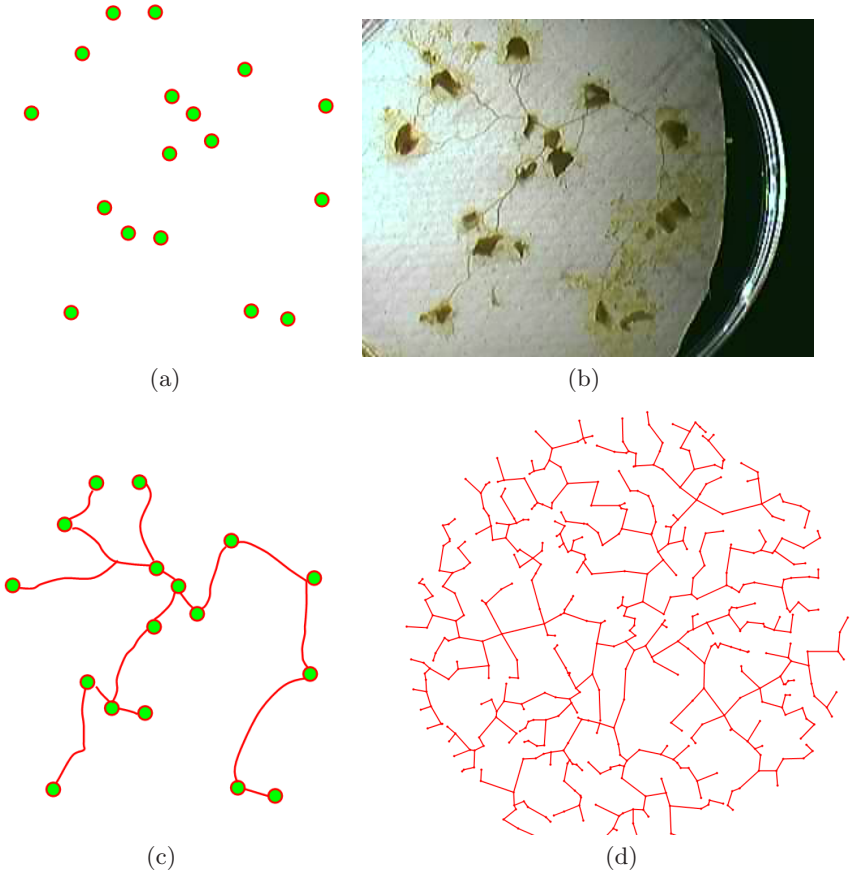

Fig. 2. Approximating spanning tree by plasmodium: (a) data set of planar points, (b) tree represented by protoplasmic strands/veins of the plasmodium, (c) extracted spanning tree, (d) spanning tree of 500 points computed by simulated plasmodium

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 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 [1]. Recent experimental developments in foraging behaviour of *P. Polycephalum* [31,31,33,46,10,11,45] convinced us that our algorithm for growing spanning tree can be implemented by living plasmodium.

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 on it, or just planted plasmodium on a bottom of bare container and fixed wet paper on the container's cover to keep humidity high. Oat flakes placed at the positions of given planar points to be spanned by a tree. The containers were stored in the dark except during periods of observation.

Once placed in the container and recovered the plasmodium starts to explore the surrounding space. Numerous pseudopodia emerge, frequently branch and proceed. The plasmodium growth 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 initial and 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 diameter of the planar set, substrate and other conditions.

Let us have a closer look at the set of 16 points (Fig. 2a) to be spanned. We represented the set by a positions of oat flakes (source of nutrients), placed flakes on the moistened filter paper, placed a piece of plasmodium at one of the flakes. In two days plasmodium spanned set of flakes. Edges of the tree are visible as dark protoplasmic strands, connecting dark irregular shapes of oat flakes (Fig. 2b). Manually enhances picture of the spanning tree is shown in Fig. 2c. Tree computed by plasmodium in our experiments satisfactory match trees computed by clasical techniques, e.g. by Jaromczyk-Supowit method [24,40], see [8]. Even when represented in simulation, the algorithm works pretty well on large data sets (Fig. 2d).

3 Phyrasum Machines

We demonstrate that plasmodium of *Physarum polycephalum* is an ideal biological sibstrate for implementation of Kolmogorov-Uspensky machines [7].

Kolmogorov-Uspensky machine (KUM) [28,29] is defined on a colored/labeled undirected graph with bounded degrees of nodes and bounded number of colors/labels. KUM operates, modify their storage, as follows. Select an active node in the storage graph. Specify local active zone, the node's neighborhood. Modify the active zone: add new node with the pair of edges, connecting the new node with the active node; delete a node with the pair of incident edges; add/delete edge between the nodes. A program for KUM specifies how to replace neighborhood of active node with new neighborhood, depending on labels of edges connected to the active node and labels of the nodes in proximity of the active node [14]. All previous and modern models of real-world computation are heirs of KUM: Knuth's linking automata [27], Tarjan's Reference Machines [41], Schönhage's storage modification machines [34,35]. When restrictions on bounded in- and out-degrees of the machine's storage graph are lifted, the machine becomes Random Access Machine.

Functions computable on Turing machines (TM) are computed in on KUM, and any sequential device are simulated by KUM [23]. KUM can simulate TM in real time, but not *vice verse* [22]. KUM's topology is much more flexible than that of TM, and KUM is stronger then any 'tree-machine' [38].

In 1988 Gurevich [23] suggested that an edge of KUM is not only informational but also physical entity and reflects physical proximity of the nodes (thus e.g. even in three-dimensional space number of neighbors of each node is polynomially bounded). 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, plasmodium, of a true slime mold *Physarum polycephalum* satisfies all these requirements.

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[2] on it, or just planted plasmodium on a bottom of a bare container and fixed wet paper on the container's cover to keep humidity high. Oat flakes were distributed in the container to supply nutrients and represent part, or data-nodes, of Physarum machine. The containers were stored in the dark except during periods of observation. To color oat flakes, where required, we used SuperCook Food Colorings[3]: blue (colors E133, E122), yellow (E102, E110, E124), red (E110, E122), and green (E102, E142). Flakes were saturated with the colorings, then dried.

Nodes: 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 (Fig. 3). At the beginning of computation, stationary nodes are distributed in the computational space, and plasmodium is placed at one point of the space. Starting in the initial conditions the plasmodium exhibits foraging behavior, and occupies stationary nodes (Fig. 3).

Edges: An edge of Physarum machine is a strand, or vein, of 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 [25,30].

Data, results and halting: Program and data are represented by a spatial configuration of stationary nodes. Result of the computation over stationary data-node is presented by configuration of dynamics nodes and edges. The initial state of a Physarum machines, includes part of input string (the part which represents position of plasmodium relatively to stationary nodes), empty output string, current instruction in the program, and 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"" [29]. Physarum machine halts when all data-nodes are utilized.

Active zone: In KUM storage graph must have some active node. This is an inbuilt feature of Physarum machine. When plasmodium resides on a substrate

[2] Thanks to Prof. Soichiro Tsuda for providing me with *P. polycephalum* culture.
[3] `www.supercook.co.uk`

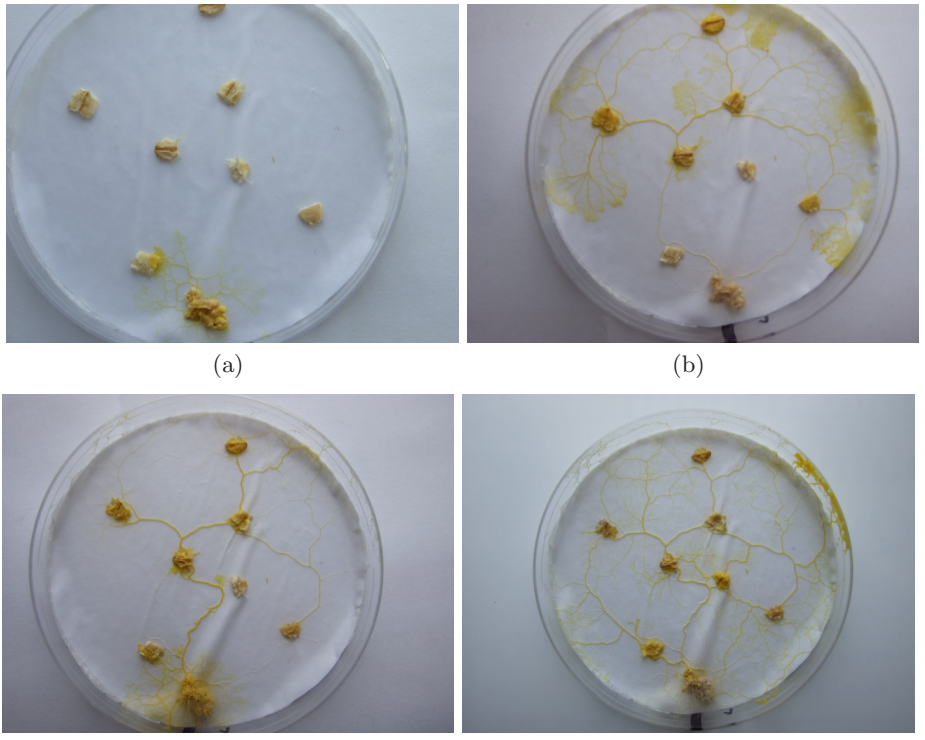

Fig. 3. An example of computational process in Physarum machine. Photographs (a)–(d) are taken with time lapse circa 24 hours.

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 similarly to wave-fragments in sub-excitable Belousov-Zhabotinsky medium [36]. An example of single active node, which is just started to develop its active zone, is shown in (Fig. 4). At every step of computation in KUM there is an active node and some 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, size of active zone may vary depending on 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, shape to change and even protoplasmic veins to annihilate. Active zone is comprised of stationary or dynamic nodes connected to active node with veins of protoplasm.

Bounded connectivity: 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 [37]. Usually, not more

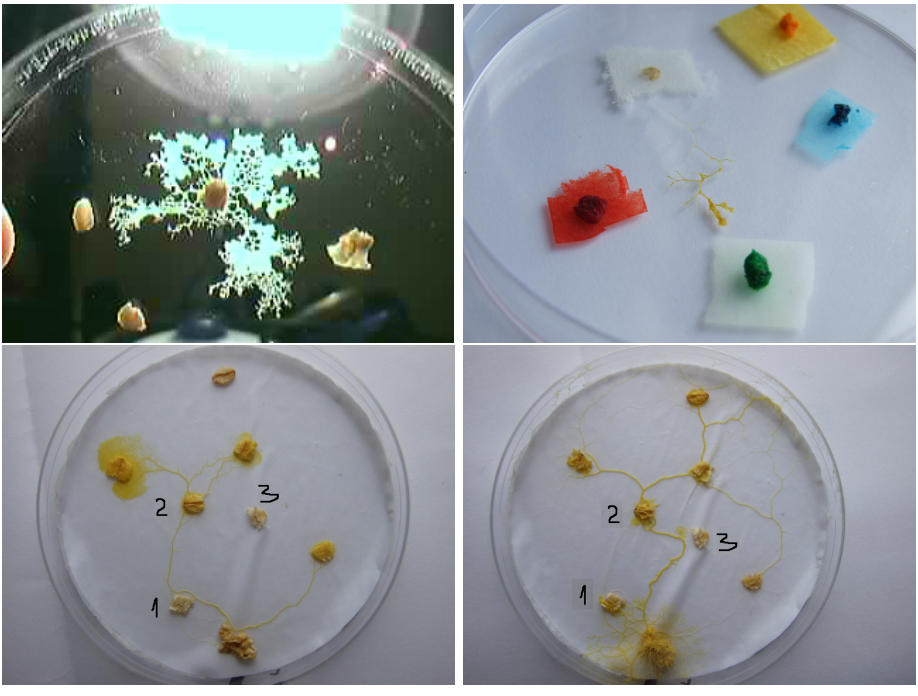

Fig. 4. Basic operations: (a) single active node is generating active zone at the beginning of computation, (b) addressing of green-coloured data-node, (c) and (d) implementation of ADD NODE, ADD EDGE, REMOVE EDGE operations

then three protoplasmic strands join each other in one given point of space. It is reported that average degree of minimum spanning tree is around 1.99, and of relative neighborhood graph around 2.6 [17]. Graphs produced by standard procedures for generating combinatorial random planar graphs show a limited growth of average degree with number of nodes or edges, the degree stays around 4 when number of edges increase from 100 to 4000 [9]. We could assume that average degree of storage graph in Physarum machines is a bit higher then degree of spanning trees but less then degree of random planar graphs.

Addressing and labeling: Every node of KUM must be uniquely addressable and nodes and edges labeled [29]. There is no direct implementation of such addressing in Physarum machine. With stationary nodes this can be implemented either by coloring the nodes, or by tuning humidity of the oat flakes. Coloring the stationary nodes could be another solution. An example of experimental implementation of addressing is shown in Fig. 4b.

Basic operations: A possible set of instructions for Physarum machine could be as follows. Common instruction would include INPUT, OUTPUT, GO, HALT, and internal instructions: NEW, SET, IF [19]. At present state of experimental implementation we assume that INPUT is done via distribution of sources of

nutrients, while OUTPUT is recorded optically. Instruction SET causes pointers redirection, and can be realized by placing fresh source of nutrients in the experimental container, preferably on top of one of the old sources of nutrients. When new node is created all pointers from the old node point to the new node. Let us look at the experimental implementation of core instructions.

ADD NODE: To add a stationary node b to node a's neighborhood, plasmodium must propagate from a to b, and form a protoplasmic vein representing edge (ab). To form a dynamic node, propagating pseudopodia must branch into two or more pseudopodia, and the site of branching will represent newly formed node.

REMOVE NODE: To remove stationary node from Physarum machine, plasmodium leaves the node. Annihilating protoplasmic strands forming a dynamic node at their intersection, remove the dynamic node.

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

REMOVE EDGE: When protoplasmic vein annihilates, e.g. depending on global state or when source of nutrients exhausted, edge represented by the vein is removed from Physarum machine (Fig. 4cd). The following sequence of operations is demonstrated in Fig. 4cd: node 3 is added to the structure by removing edge (12) and forming two new edges (13) and (23).

References

1. Adamatzky, A.: Neural algorithm for constructing minimal spanning tree. Neural Network World 6, 335–339 (1991)
2. Adamatzky, A.: Computing in non-linear media and automata collectives. IoP, Bristol (2001)
3. Adamatzky, A. (ed.): Collision-Based Computing. Springer, Heidelberg (2003)
4. Adamatzky, A., De Lacy Costello, B., Asai, T.: Reaction-Diffusion Computers. Elsevier, Amsterdam (2005)
5. Adamatzky, A., Teuscher, C.: From Utopian to Genuine Unconventional Computers. Luniver Press (2006)
6. Adamatzky, A.: Physarum machines: encapsulating reaction-diffusion to compute spanning tree (submitted)
7. Adamatzky, A.: Physarum machine: implementation of Kolmogorov-Uspensky machine in biological substrate. Parallel Processing Letters (in press, 2007)
8. Adamatzky, A.: Growing spanning trees in plasmodium machines, Kybernetes (in press, 2007)
9. Alber, J., Dornm, F., Niedermeier, R.: Experiments on Optimally Solving NP-complete Problems on Planar Graphs. Manuscript (2001), `http://www.ii.uib.no/~frederic/ADN01.ps`
10. Aono, M., Gunji, Y.-P.: 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 (2001)
11. Aono, M., Gunji, Y.-P.: Material implementation of hyper-incursive field on slime mold computer. In: Computing Anticiaptory Systems. AIP Conference Proceedings, vol. 718, pp. 188–203 (2004)

12. Bardzin's, J.M.: On universality problems in theory of growing automata. Doklady Akademii Nauk SSSR 157, 542–545 (1964)
13. Barzdin', J.M., Kalnins, J.: A universal automaton with variable structure. Automatic Control and Computing Sciences 8, 6–12 (1974)
14. Blass, A., Gurevich, Y.: Algorithms: a quest for absolute definitions. Bull. Europ. Assoc.TCS 81, 195–225 (2003)
15. van Emde Boas, P.: Space measures for storage modification machines. Information Process. Lett. 30, 103–110 (1989)
16. Calude, C.S., Dinneen, M.J., Păun, G., Rozenberg, G., Stepney, S.: UC 2006. LNCS, vol. 4135. Springer, Heidelberg (2006)
17. Cartigny, J., Ingelrest, F., Simplot-Ryl, D., Stojmenovic, I.: Localized LMST and RNG based minimum-energy broadcast protocols in ad hoc networks. Ad Hoc Networks 3, 1–16 (2005)
18. Cloteaux, B., Rajan, D.: Some separation results between classes of pointer algorithms. In: DCFS '06: Proceedings of the Eighth Workshop on Descriptional Complexity of Formal Systems, pp. 232–240 (2006)
19. Dexter, S., Doyle, P., Gurevich, Y.: Gurevich abstract state machines and Schönhage storage modification machines. J. Universal Computer Science 3, 279–303 (1997)
20. Dijkstra, E.A.: A note on two problems in connection with graphs. Numer. Math. 1, 269–271 (1959)
21. Gacs, P., Leving, L.A.: Casual nets or what is a deterministic computation, STAN-CS-80-768 (1980)
22. Grigoriev, D.: Kolmogorov algorithms are stronger than Turing machines. Notes of Scientific Seminars of LOMI (in Russian) 60, 29–37 (1976) (English translation in J. Soviet Math. 14(5) 1445–1450 (1980))
23. Gurevich, Y.: On Kolmogorov machines and related issues. Bull. EATCS 35, 71–82 (1988)
24. Jaromczyk, J.W., Kowaluk, M.: A note on relative neighborhood graphs. In: Proc. 3rd Ann. Symp. Computational Geometry, pp. 233–241 (1987)
25. Kamiya, N.: The protoplasmic flow in the myxomycete plasmodium as revealed by a volumetric analysis. Protoplasma 39, 3 (1950)
26. Kirkpatrick, D.G., Radke, J.D.: A framework for computational morphology. In: Toussaint, G.T. (ed.) Computational Geometry, pp. 217–248. North-Holland, Amsterdam (1985)
27. Knuth, D.E.: The Art of Computer Programming. Fundamental Algorithms, vol. 1. Addison-Wesley, Reading, Mass (1968)
28. Kolmogorov, A.N.: On the concept of algorithm. Uspekhi Mat. Nauk 8(4), 175–176 (1953)
29. Kolmogorov, A.N., Uspensky, V.A.: On the definition of an algorithm. Uspekhi Mat. Nauk (in Russian), 13, 3–28 (1958) (English translation: ASM Translations 21(2), 217–245 (1963))
30. Nakagakia, T., Yamada, H., Ueda, T.: Interaction between cell shape and contraction pattern in the *Physarum plasmodium*. Biophysical Chemistry 84, 195–204 (2000)
31. Nakagaki, T.: Smart behavior of true slime mold in a labyrinth. Research in Microbiology 152, 767–770 (2001)
32. Nakagaki, T., Yamada, H., Toth, A.: Maze-solving by an amoeboid organism. Nature 407, 470 (2000)
33. Nakagaki, T., Yamada, H., Toth, A.: Path finding by tube morphogenesis in an amoeboid organism. Biophysical Chemistry 92, 47–52 (2001)

34. Schönhage, A.: Real-time simulation of multi-dimensional Turing machines by storage modification machines. Project MAC Technical Memorandum, vol. 37. MIT, Cambridge (1973)
35. Schönhage, A.: Storage modification machines. SIAM J. Comp. 9, 490–508 (1980)
36. Sedina-Nadal, I., Mihaliuk, E., Wang, J., Perez-Munuzuri, V., Showalter, K.: Wave propagation in subexcitable media with periodically modulated excitability. Phys. Rev. Lett. 86, 1646–1649 (2001)
37. Shirakawa, T.: Private communication (February 2007)
38. Shvachko, K.V.: Different modifications of pointer machines and their computational power. In: Tarlecki, A. (ed.) Mathematical Foundations of Computer Science 1991. LNCS, vol. 520, pp. 426–435. Springer, Heidelberg (1991)
39. Steinbock, O., Tóth, A., Showalter, K.: Navigating complex labyrinths: optimal paths from chemical waves. Science 267, 868–871 (1995)
40. Supowit, K.J.: The relative neighbourhood graph, with application to minimum spanning tree. J. ACM 30, 428–448 (1988)
41. Tarjan, R.E.: Reference machines require non-linear time to maintain disjoint sets, STAN-CS-77-603 (March 1977)
42. Tero, A., Kobayashi, R., Nakagaki, T.: A coupled-oscillator model with a conservation law for the rhythmic amoeboid movements of plasmodial slime molds. Physica D 205, 125–135 (2005)
43. Teuscher, C., Adamatzky, A. (eds.): Unconventional Computing 2005: From Cellular Automata to Wetware. Luniver Press (2005)
44. Tirosh, R., Oplatka, A., Chet, I.: Motility in a "cell sap" of the slime mold *Physarum Polycephalum*. FEBS Letters 34, 40–42 (1973)
45. Tsuda, S., Aono, M., Gunji, Y.-P.: Robust and emergent Physarum-computing. BioSystems 73, 45–55 (2004)
46. Tsuda, S., Zauner, K.P., Gunji, Y.P.: Robot Control: From Silicon Circuitry to Cells. In: Ijspeert, A.J., Masuzawa, T., Kusumoto, S. (eds.) BioADIT 2006. LNCS, vol. 3853, pp. 20–32. Springer, Heidelberg (2006)
47. Uspensky, V.A.: Kolmogorov and mathematical logic. The Journal of Symbolic Logic 57, 385–412 (1992)
48. Yamada, H., Nakagaki, T., Baker, R.E., Maini, P.K.: Dispersion relation in oscillatory reaction-diffusion systems with self-consistent flow in true slime mold. J. Math. Biol. (2007)