Comparison of Bio-Inspired and Graph-Theoretic Algorithms for Design of Fault-Tolerant Networks

Matthias Becker, Waraphan Sarasureeporn and Helena Szczerbicka
FG Simulation and Modeling
Leibniz University Hannover
Welfengarten 1, 30167 Hannover, Germany
{xmb,sarasureeporn,hsz}@sim.uni-hannover.de

***Abstract*—Recently several approaches have been presented that exploit the ability of *Physarum polycephalum* to connect several food sources via a network of pipes in order to maintain an efficient food distribution inside the organism. These approaches use the mechanisms found in nature in order to solve a technical problem, namely the design of constructing fault-tolerant and efficient connection networks. These works comprise experiments with a real slime mold *Physarum polycephalum* as well as computer simulations based on a tubular model and an agent-based approach. In this work, we study the suitability of those bio-inspired approaches and compare their performance to a graph-theoretic algorithm for construction of fault-tolerant connection networks, the (k, t)-spanner algorithm. The graph-theoretic algorithm is able to construct graphs with a certain degree of fault tolerance as well as meet a given maximal path length between two arbitrary nodes. However the definition of fault tolerance in previous bio-inspired works differs to that used in graph theory. Thus in our contribution we analyze the bio-inspired approaches as well as the graph-theoretic approach for their efficiency of designing optimal fault-tolerant graphs. We demonstrate the usability of the graph-theoretic approach despite relying on a different definition of fault tolerance. We conclude that classical efficient computational algorithms from graph theory can be adapted and applied in the same field as the bio-inspired approaches for the problem of constructing efficient fault tolerant networks. They often provide an easier to use and more direct solution than bio-inspired approaches, that need more parameter tuning before getting satisfactory results.**

***Index Terms*—slime mold, *Physarum polycephalum*, fault tolerant network, (k,t)-spanner**

I. Motivation

Recently bio-inspired computing based on slime molds raised attention in scientific renowned journals [1]–[3] as well as in popular newspapers for the slime molds' ability to solve complex problems, despite being a brainless primitive life-form [4].

Research groups use a real slime mold or different types of simulations of slime mold behavior for building networks or finding a short path through a maze. One group [1] conducted experiments with a real slime mold *Physarum polycephalum* and computer simulations based on a tube model in order to construct a fault tolerant and efficient transport network for the Tokyo rail system. The natural slime mold as well as the simulated slime mold generate networks that are similar to the existing rail system of Tokyo. While the quality of the solutions of the real slime mold shows considerable variations, the networks constructed by tubular simulations show a very regular structure in their quality that is correlated with one parameter. In another approach [5] an agent based simulation of *Physarum polycephalum* turned out to better approximate the characteristics of the real slime mold's networks.

However, existing classical algorithms for those problems have not been included in the evaluation of bio-inspired approaches in previous works.

Thus in this work we study, whether natural or simulated *Physarum polycephalum* is an efficient means for construction of fault tolerant networks at all, i.e., can networks of good quality be generated with reasonable computational effort. Although *Physarum polycephalum* simulations have also been used in the past for obtaining fault tolerant networks, the quality of such networks has only been compared to existing networks that have been historically grown and not been constructed efficiently from scratch. We demonstrate how algorithms from graph theory can be used for the design of fault-tolerant efficient networks, despite using different definitions of fault tolerance. We show that in many scenarios the graph-theoretic algorithm is a viable means to efficiently obtain reliable results without having the additional effort of parameter tuning which often is a serious disadvantage of bio-inspired algorithms.

In the following, we will describe the state of the art concerning simulation models of *Physarum polycephalum* that can be used for the construction of fault tolerant connection networks. We use the Tokyo railway network, which is the mostly used reference network in this context. We also describe the agent-based simulation model and the (k, t)-spanner algorithm from graph theory. A presentation and discussion of the resulting networks for all approaches concludes this work.

II. Physarum Polycephalum

There are basically two kinds of slime molds (cellular and acellular) which are member of a category of eukaryotic organisms that typically have some fungal-like attributes and some animal-like attributes. In this work, we are interested in *Physarum polycephalum*, a slime mold visible to the unaided eye.

 ISBN: 978-1-61208-187-8

A. Biological Foundations

Physarum polycephalum is a yellow single-celled slime mold whose plasmodium is visible for the unaided human eye and can grow up to one square meter if the environmental conditions are ideal. Otherwise it usually has the size of a palm. Starting with spores of *Physarum polycephalum* mysamben with a single nucleus will be produced, which can reproduce by mitosis. Dependent on environmental conditions flagellates can evolve. If two flagellates of different sex meet they form a diploid zygote. This will grow to the final size plasmodium by division of the nucleus.

Summarizing, the large visible yellow slime mold is a large single cell with multiple nuclei. When food is used up in the area of the cell it enters the hunger phase. In this phase *Physarum polycephalum* optimizes its shape by maintaining thick pipes between food sources and by shrinking the contour where no food is available any more. This is the phase that is used by most computational slime mold inspired algorithms.

B. Computational Applications of Physarum polycephalum

New research showed that this kind of slime mold is able to construct efficient and fault tolerant connection networks between multiple food sources. This ability has been used (with real slime molds and simulations of *Physarum polycephalum*) on examples such as British and American motorways [6], [7] and for the Tokyo railway network [1]. *Physarum polycephalum* constructed networks that had similar properties as the existing networks designed by human engineers. Another application is the usage of *Physarum polycephalum* as light detector of a robot [8] and in wireless ad hoc networks [9].

C. Simulation Models for Physarum polycephalum

In the literature, several types of computer simulation models can be found.

In [10], the hunger-phase of *Physarum polycephalum* is modeled by a mesh network of tubes that can enlarge or shrink. This model is close to the natural mechanisms, where nutrition is streamed through the slime mold, so that nutrition is spread throughout the whole cell, from food sources to areas with less food. During that process the streaming channels of the slime mold enlarge, where more nutrition has to be moved, and channels shrink or disappear where little nutrition is present or is to be transported. The simulation model includes differential equations of the pressure and the movement of the fluid with time, and the changing of the size of pipes dependent on the moving fluid.

In [11], an agent based model is used which basically is a cellular automaton. Each place in the two dimensional matrix can be visited by an agent. An agent has three sensors in its front view, front right, front left and front middle. Parameters are sensor angle and sensor range. Two actions are possible: move to another cell and/or leave a trace. This approach models the distribution of the nutrition inside the *Physarum polycephalum* cell in a more abstract way, physically and quantitatively not very close to the natural mechanisms. However the phenomena of building fault tolerant short networks is captured by this model very well. The agents can be interpreted as moving nutrition that is not explicitly channeled. However channel-like streams will build up implicitly by the rules governing the agents' behavior. Furthermore the agent based approach allows fast simulations/calculations and it is not necessary to deal with costly calculation and solution of differential equations.

The approach in [12] models the expansion and shrinkage phase similarly to dilation and erosion as know from computer graphics. This algorithm is especially designed for path finding in a maze.

Subsequently, we will show how to obtain connection networks using the agent-based simulation of *Physarum polycephalum*. The resulting networks (see also [5]) will be presented and their quality will be compared to that of the networks obtained by tubular simulations and with experiments done with real *Physarum polycephalum* (see [1]). Then, it will be demonstrated how algorithms from graph theory [13] can be adapted for application in our context.

III. Network Design by Agent Based Simulation of *Physarum polycephalum*

We used our simulator, that is based on the model in [11], in order to construct a number of different fault tolerant connection networks and compared the characteristics of the found solutions with the existing real railway network and with a manually constructed fault tolerant graph (by human expertise, resulting from enhancing the Minimum Spanning Tree (MST) where fault tolerance has been introduced by manually adding some edges).

Our simulator is based on a cellular automaton. The playground is a two dimensional matrix, on which agents are placed, that observe their environment and that can move around and/or leave a chemical, dependent on the environmental conditions. The chemical is steering the movement of agents. The chemical is emitted by food sources and spreads spatially, by the same time vanishing through evaporation. Agents can reinforce this signal by emitting the chemical themselves. The value of the chemicals concentration is stored as attribute of each cell of the matrix.

The main rules of the agent's behavior according to [11] (slightly different rules can be found in [14], [15]) are:

Movement:

```
Step 1: 'Attempt to move forward in current direction'
Step 2:
.....IF ('move forward successful')
.........THEN 'Deposit trail in new location'
.........ELSE 'Choose random new orientation'
```

Given that each agent faces into a certain direction and can sense the matrix for the concentration of the chemical in front direction (F), front right (FL) and front left (FL), the agent maintains its direction, rotates a certain angle (RA) to the left or right, or randomly:

```
Step 1: 'Sample trail map sensor values F, FL, FR'
```

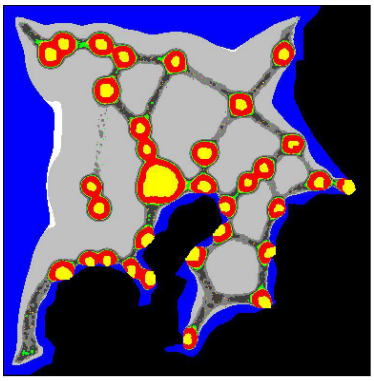

Fig. 1. Screenshot of simulation

```
Step 2:
IF (F > FL) AND (F > FR)
.....'Stay facing same direction'
ELSEIF (F < FL) AND (F < FR)
.....'Rotate randomly left or right by RA'
ELSEIF (FL < FR)
.....'Rotate right by RA'
ELSEIF (FR < FL)
.....'Rotate left by RA'
ELSE
.....'Continue facing same direction'
```

The simulation starts with a matrix which can be represented as bitmap where the different colors stand for the various states of a cell (cf. Fig. 1): The background representing empty cells is white. A food source is yellow. Yellow food sources represent the nodes of a graph (cities) which have to be connected by the slime mold (connections representing railways). Black and blue is forbidden terrain, red to gray is the intensity of the trail signal.

In nature, food is the attractor for *Physarum polycephalum* and where there is more food there *Physarum polycephalum* moves to. In nature, food flows through *Physarum polycephalum* and influences the movement and shape of *Physarum polycephalum*. In the agent model, the food flow is modeled by a trace that agents leave on their trail, the trace signaling that food is near, or that many agents are heading in this direction, probably because someone found food there. This behavior is similar to ant algorithms, where ants also emit pheromones on their path [16].

The optimization problem to be solved when constructing a fault tolerant graph is to find a solution between the Minimum Spanning Tree (minimum cost, that is sum of length of edges between nodes, but no fault tolerance) and a fully connected graph (maximal fault tolerance but also maximal cost). The slime molds tries to connect all food sources thereby shortening all of its connections as much as it can. Fault tolerance is not a direct goal of the slime mold, fault tolerance is a side effect since the slime mold just connects nodes that are near each other so that automatically several paths are established when a higher number of nodes are crowded in an area. Only isolated nodes will not be connected in a fault tolerant way. Note that the graph constructed by *Physarum polycephalum* will not only consist of direct links between nodes but may also have Steiner tree like connections. As can be observed from the picture in Fig. 1, the agents/chemicals not necessarily 'draw' an easily automatically measurable line between food sources. Thus the resulting graph has to be determined manually by drawing an edge where are 'enough' agents between two food sources. The graph constructed that way can then be analyzed for its quality measures. Because there is not always a state of clear convergence a stopping criteria has to be defined, i.e., when the simulation has to be stopped and the graph has to be defined and analyzed. We did measurements in two ways to account for the dynamics of the simulation: First experiments stopped after a fixed number of iterations (5000), the resulting network was measured then. In the second row of experiments, the simulation was stopped periodically (every 250 iterations), the graph was measured and analyzed, and then the simulation has been continued.

Both experiments were repeated with an additionally activated 3x3 filter, that smoothens the chemical values around each agent by averaging the values, which makes four different experimental setups which will be shown and discussed in Fig. 3 in the next section.

Before presenting the results, the measures describing the quality of the found graph will be introduced. In order to make this comparable to recent results the definition of the quality measures are taken from [1]:

- Graph G
- S: Set of nodes of G
- E: Set of edges e of G
- length(e): weight of edge e representing the distance of two nodes
- MST(G): Minimum Spanning Tree of G
- N: Number of nodes of G
- $SP(v_i, v_j)$: Shortest Path from node i to node j

The evaluation of the found networks is analogous to [1], where the definitions for the following measures of the graph are taken from:

Total length of all connections:

$$\mathrm{TL}(G) = \sum_{e \in E} \mathrm{length}(e)$$

Cost of the network relative to the minimum cost (of the MST):

$$\mathrm{Cost}(G) = \frac{\mathrm{TL}(G)}{\mathrm{TL}(MST(G))}$$

Fault Tolerance FT that takes into account the length of edges, since a long connection is more prone to fault:

$$\mathrm{FT}(G) = \frac{\sum_{e \in E | (G \setminus \{e\}) \text{ isConnected}} \mathrm{length(e)}}{\mathrm{TL}(G)}$$

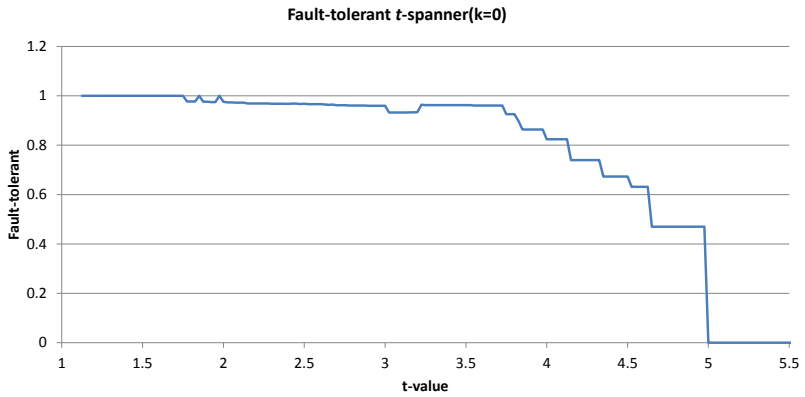

Fig. 2. Fault tolerance of spanner algorithm with varying value of t

The performance of the network as average distance between two arbitrary nodes relative to MST:

$$\text{Performance(G)} = \frac{\text{avgDistance}(G)}{\text{avgDistance}(MST)}$$

IV. Geometrical Spanners

In this section we explain the algorithm from graph theory, that has as input a set of points, and delivers a graph with a certain degree of fault tolerance and a maximal path length between each pair of nodes as result.

For a given set of nodes, a spanner is a graph that connects all nodes. In this context, the notion of fault tolerance is defined independently of the length of edges: a graph is said to be k-fault tolerant when k arbitrary edges can be removed and the graph still remains fully connected.

Obviously, the Minimum Spanning Tree is a spanner with zero fault tolerance.

A. Definition and Construction of a (k,t)-Spanner

Taken from [17], the necessary definitions and the algorithm for construction of a (k,t) spanner, are presented in this section. Note that it has also be proven that vertex and edge fault tolerance are corresponding concepts.

Let S be the set of N points in $\mathbf{R}^2$ as defined above. Let $t > 1$ be a real number, let $k \geq 0$ be an integer, and let $G = (S, E)$ be an undirected Euclidean graph with vertex set S. The spanner with stretch factor t is called t-spanner, if $\delta(p,q) < t|pq|$ for any two points of S. The notation $|pq|$ to denote the Euclidean distance and $\delta(p,q)$ to denote the shortest Euclidean length of a path in a geometric graph G between p and q. The approach is a greedy algorithm. Each edge of the complete graph is considered for construction of the spanner. The edges are processed in increasing order of the edge length, for that purpose they are stored sorted in list L. For each edge the intermediate result graph will be checked, whether already the fault tolerant criterion between the two points connected by that edge is fulfilled. If not, the edge is added to the graph. If yes, but the t criterion is not fulfilled (i.e. the existing paths are too long) then the edge is also added. Formally, that is, if the graph G does not contains $k+1$ vertex-disjoint t-spanner paths between p and q, the edge (p,q) will be included in the set of edges E. The output of algorithm is a t-spanner for S with k fault tolerance.

Altogether, the algorithm can be formalized as shown in the following algorithm 1 (excerpted from [17]).

Algorithm 1 (k,t)-spanner algorithm

Input: A set S of N points in $\mathbf{R}^2$, an integer $k \geq 0$, and a real number $t > 1$
Output: A (k,t)-spanner for S
Initialisation: $G := (S, E)$ with $E := \emptyset$

for each $\{p,q\} \epsilon L$ considered pairs in nondecreasing order
do if G does not contain $k+1$ vertex-disjoint t-spanner paths between p and q
then $E := E \cup \{\{p,q\}\}$
$G := (S, E)$
endif
endfor

Note that there are different definitions of fault tolerance. In the following we will use the definition that includes the length of edges. This has two reasons; first, the networks produced by bio-inspired algorithms nearly never show fault tolerance greater than one. Second, for application in real networks, the definition including the lengths of edges is more realistic, since

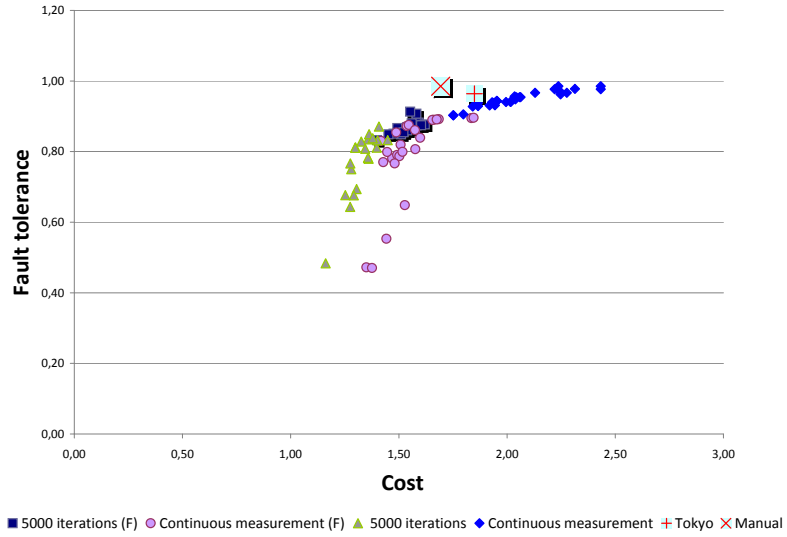

Fig. 3. Agent-based slime mold simulation results for the trade-off fault tolerance vs cost (both normalized to MST)

the fault probability of longer connections usually is higher in reality, be it communication lines or transport networks.

In Fig. 2, the characteristics of the generated networks can be observed for changing values of t (for $k = 0$). For values of t approaching one, the generated networks converge towards the complete graph. Since for large values of t the resulting networks converge towards the Minimum Spanning Tree, the fault tolerance of the resulting spanners initially is one, and falls down on a convex curve towards zero. We conclude that by variation of parameter t a wide range of networks can be generated. In the next section, we will analyse these networks for their properties concerning length and fault tolerance and compare them to those generated by the slime mold simulation.

V. Results

A. Fault Tolerance versus Costs

Fig. 3 presents the results obtained by agent-based simulations of *Physarum polycephalum*, showing the degree of fault tolerance versus the costs of each produced network. The diversity of networks has its origin in the stochastic nature of the agent-based simulation. Additionally, the real Tokyo network is included, as well as a network which has been designed manually, taking the MST and adding several connections manually, where obviously necessary. As first observation, it can clearly be seen that the fault tolerance of the Tokyo network as well as the manually constructed network is better than the networks produced by the slime mold algorithm. The Tokyo network is a bit less fault tolerant and has marginally less costs than the manually constructed network.

This is not surprising since the Tokyo network is grown historically and has to cope with geographical, political and other constraints that are not represented in the graph.

Since fault tolerance is not a primary goal of the slime mold algorithm the results for performance do not reach top values. However, a number of solutions show a good trade off: Especially the simulation results after 5000 iterations without filter reach values for fault tolerance around 0.84 while having costs of only 1.4 (the Tokyo and manually constructed networks have fault tolerance of around 0.98 and costs of around 1.75).

Fig. 3 directly relates to Fig. 3 (A) in [1]. For reference, the results for the natural slime mold and the tubular simulations from [1] are given in Fig. 4 here. It can be seen that the graph for the tubular simulation results starts with zero fault tolerance and cost of one (obviously it found the MST), raising quickly in a straight line to a fault tolerance of approximately 0.9 at normalized cost of 1.5 and from there on converging to one, at raising cost. Contrary to the tubular simulation, the results for the natural *Physarum polycephalum* barely find a non-fault tolerant network, nearly all networks having fault tolerance over 0.6. The convergence to fault tolerant networks at high costs cannot be observed, several networks with costs

 ISBN: 978-1-61208-187-8

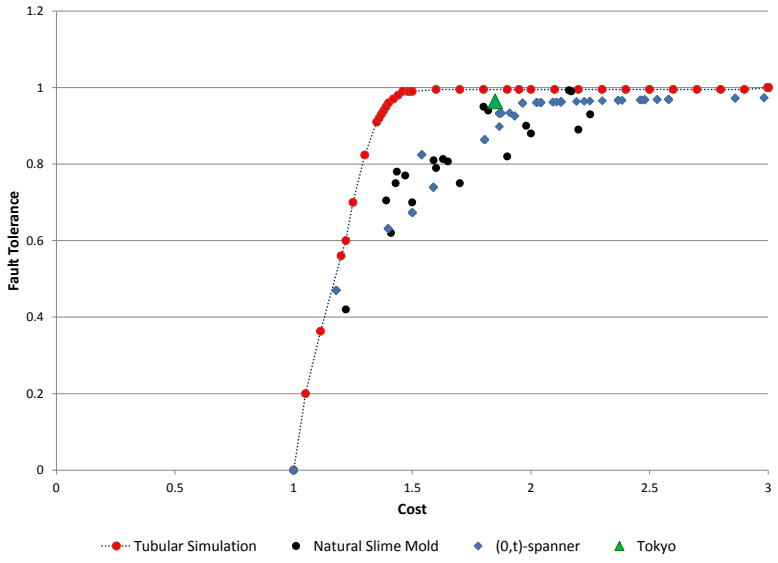

Fig. 4. Fault tolerance versus costs for tubular model, (k,t)-spanner and natural slime mold

of around 1.5 have a fault tolerance of less than 0.8.

The agent-based simulation results reproduce much better the variety of the results obtained by the natural slime mold as can be seen by comparing Fig. 3 and Fig. 4. Taking the results for 5000 iterations without filter and continuous measurements with filter together, the results resemble the natural results much more in their variance, and also not showing less fault tolerance than 0.4. The results for continuous measurement without filter seem to be similar to the tube model, the trend to convergence for fault tolerant networks at higher cost can be clearly observed.

Fig. 4 also contains the results for the (k,t)-spanner. The data points for the spanner algorithm show a good accordance to the naturally produced variety of networks regarding non-extremal values for t. For the extremal values of t the results of course converge against the complete graph and MST, similarly as for the tubular model.

Contrary to the tubular model, the spanner algorithm reproduces the variety of naturally produced networks much better for non-extremal values for t, resulting in networks with costs in the interval of 1.25 and 2, and fault tolerance between 0.5 and 0.95.

Altogether, it can be stated that the (k,t)-spanner algorithm is applicable despite the different definition of fault tolerance and that it is well capable of producing networks whose characteristics show the same variety as networks constructed by real slime molds. This is achieved by changing the value of parameter t.

VI. Discussion and Conclusion

The results in the last section showed that slime mold inspired construction of fault tolerant optimized connection networks is a working approach.

However, it is the question, whether dedicated classical algorithms might be a better choice for this task.

Algorithms for that purpose belong to the class of algorithms for geometrical spanners [13], [17]. It has been shown in this paper that by varying the parameter t in the (k,t)-spanner algorithm, networks with the desired trade-off between fault tolerance and performance can be generated. It has also been shown how to overcome the different definitions of fault tolerance, used in graph theory and in the bio-inspired approaches.

Thus, the (k,t)-spanner algorithm can well be used in order to construct a connection network tailored to the needs, be it fault tolerance, length, etc.

For the Tokyo network, the spanner algorithm worked quite satisfactory and faster than the bio-inspired approaches. However the complexity is $O(|L| \cdot \log|L|)$ stemming from the sorting of $|L|$ possible edges. Note that the problem size should be characterized by the number of points N to be connected, resulting in $\binom{N}{2}$ possible edges. For future work it should be investigated, for which N the (k,t)-spanner algorithm is not applicable anymore and whether the bio-inspired approaches still work for that problem complexity. In order to do this an approach is needed to automatically construct a graph out of the simulation results, since manually marking the result graph in the image of the final agent distribution is not a viable approach for larger problems.

It also showed that the approach of using human intuition, i.e., constructing a Minimum Spanning Tree and adding manually some edges for fault tolerance, is a cheap solution that leads to good results. However, this approach is not viable for larger graphs.

Summarizing, it can be said that naturally inspired algorithms is of course an interesting field that also lead to valuable

 ISBN: 978-1-61208-187-8

new algorithmic approaches. However, not everything that works in nature can be transferred to technical solutions easily. As we show in this work, slime mold inspired algorithms are capable to construct fault tolerant connection networks. Drawbacks are found in the analysis of the simulation results and in the runtime for the application of the Tokyo railway system. Furthermore, relatively high effort has to be put in the parameter tuning of the complex agent-based algorithm, until it delivers satisfactory results. Additionally, it turned out that among the different modeling approaches for *Physarum polycephalum*, the agent based model the best one in the context of constructing networks between food sources.

Future work will include evaluation of both bio-inspired and classical approaches for a number of small to large benchmark problems in order to decide for which problem complexity classical or bio-inspired approaches are superior.

REFERENCES

[1] A. Tero, S. Takagi, T. Saigusa, K. Ito, D. Bebber, M. Fricker, K. Yumiki, R. Kobayashi, and T. Nakagaki, "Rules for biologically inspired adaptive network design," *Science*, vol. 327, no. 5964, p. 439, 2010.

[2] W. Marwan, "Amoeba-Inspired Network Design," *Science*, vol. 327, no. 5964, p. 419, 2010.

[3] T. Nakagaki, H. Yamada, and A. Toth, "Intelligence: Maze-solving by an amoeboid organism," *Nature*, vol. 407, 2000.

[4] J. T. Bonner, "Brainless behavior: A myxomycete chooses a balanced diet," *PNAS*, vol. 107, no. 12, pp. 5267–5268, 2010.

[5] M. Becker, "Design of fault tolerant networks with agent-based simulation of physarum polycephalum," in *Evolutionary Computation (CEC), 2011 IEEE Congress on*, june 2011, pp. 285 –291.

[6] A. Tero, R. Kobayashi, and T. Nakagaki, "Physarum solver: A biologically inspired method of road-network navigation," *Physica A: Statistical Mechanics and its Applications*, vol. 363, no. 1, pp. 115 – 119, 2006.

[7] A. Adamatzky and J. Jones, "Road planning with slime mould: If physarum built motorways it would route m6/m74 through newcastle," *International Journal of Bifurcation and Chaos (IJBC)*, vol. 20, no. 10, pp. 3065–3084, 2010.

[8] S. Tsuda, K. Zauner, and Y. Gunji, "Robot control with biological cells," *BioSystems*, vol. 87, no. 2-3, pp. 215–223, 2007.

[9] K. Li, K. Thomas, L. Rossi, and C. Shen, "Slime Mold Inspired Protocol for Wireless Sensor Networks," in *Self-Adaptive and Self-Organizing Systems, 2008. SASO'08. Second IEEE International Conference on*. IEEE, 2008, pp. 319–328.

[10] A. Tero, K. Yumiki, R. Kobayashi, T. Saigusa, and T. Nakagaki, "Flow-network adaptation in physarum amoebae," *Theory in Biosciences*, vol. 127, pp. 89–94, 2008.

[11] J. Jones, "The emergence and dynamical evolution of complex transport networks from simple low-level behaviours," *International Journal of Unconventional Computing*, vol. 6, no. 2, pp. 125–144, 2010.

[12] M. Ikebe and Y. Kitauchi, "Evaluation of a multi-path maze-solving cellular automata by using a virtual slime-mold model," *Unconventional Computing 2007*, p. 238, 2007.

[13] J. Gudmundsson, G. Narasimhan, and M. Smid, *Encyclopedia of Algorithms*. Berlin: Springer-Verlag, 2008, ch. Geometric spanners, pp. 360–364.

[14] A. Adamatzky and J. Jones, "Programmable reconfiguration of Physarum machines," *Natural Computing*, vol. 9, no. 1, pp. 219–237, 2010.

[15] J. Jones, "Approximating the Behaviours of Physarum polycephalum for the Construction and Minimisation of Synthetic Transport Networks, Unconventional Computation 2009," *Springer LNCS*, vol. 5715, pp. 191–208, 2009.

[16] M. Dorigo and L. Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem," *Evolutionary Computation, IEEE Transactions on*, vol. 1, no. 1, pp. 53 –66, Apr. 1997.

[17] G. Narasimhan and M. Smid, *Geometric Spanner Networks*. New York, NY, USA: Cambridge University Press, 2007.

[18] P. Bose, J. Gudmundsson, and M. H. M. Smid, "Constructing plane spanners of bounded degree and low weight," in *Proceedings of the 10th Annual European Symposium on Algorithms*, ser. ESA '02. London, UK: Springer-Verlag, 2002, pp. 234–246.

 ISBN: 978-1-61208-187-8