Nat Comput (2013) 12:277–289
DOI 10.1007/s11047-012-9344-7

Fault tolerant network design inspired by *Physarum polycephalum*

Maarten Houbraken · Sofie Demeyer · Dimitri Staessens · Pieter Audenaert · Didier Colle · Mario Pickavet

Published online: 28 August 2012

Abstract *Physarum polycephalum*, a true slime mould, is a primitive, unicellular organism that creates networks to transport nutrients while foraging. The design of these natural networks proved to be advanced, e.g. the slime mould was able to find the shortest path through a maze. The underlying principles of this design have been mathematically modelled in literature. As in real life the slime mould can design fault tolerant networks, its principles can be applied to the design of man-made networks. In this paper, an existing model and algorithm are adapted and extended with stimulation and migration mechanisms which encourage formation of alternative paths, optimize edge positioning and allow for automated design. The extended model can then be used to better design fault tolerant networks. The extended algorithm is applied to several national and international network configurations. Results show that the extensions allow the model to capture the fault tolerance requirements more accurately. The resulting extended algorithm overcomes weaknesses in geometric graph design and can be used to design fault tolerant networks such as telecommunication networks with varying fault tolerance requirements.

Keywords Bio-inspired algorithm · Fault tolerant network design · Mathematical modelling · Network optimization · *Physarum polycephalum*

M. Houbraken (✉) · S. Demeyer · D. Staessens · P. Audenaert · D. Colle · M. Pickavet
Department of Information Technology, Ghent University, Gaston Crommenlaan 8 (Bus 201), 9050 Ghent, Belgium
e-mail: maarten.houbraken@intec.ugent.be

1 Introduction

Transport networks are used to minimize the total effort that has to be invested for carrying information or goods from one point to another. Some well-known examples of man-made transport networks are road, computer and telephone networks. Not all transport networks are man-made however: many (more subtle) networks exist in nature. In fact, the human organism itself relies on several natural networks, like it's vascular and nervous system, for everyday operation. Even more examples can be found in nature: trees transporting water from the roots to the leaves. *Physarum polycephalum*, a true slime mould, also creates networks while foraging. By studying all these natural networks and their design, we can improve our man-made networks. In this paper, we will look at how *P. polycephalum* designs networks and how we can extend these principles to design fault tolerant, robust networks.

Physarum polycephalum, shown in Fig. 1, is a unicellular organism whose body consists of thousands of cell nuclei. It is a slime mould in which nutrients are transported through protoplasmic streaming. This protoplasmic streaming is well studied (Kamiya 1959; Gotoh and Kuroda 1982; Tero et al. 2005; Kobayashi et al. 2006) and is based on tube morphogenesis. Inside the body of *Physarum*, a network of tubes exists and is used during foraging. When presented with food sources (FS), *Physarum* concentrates around these sources to extract nutrients. These nutrients are dissolved in protoplasm and transported through the tubes to the rest of the body. The tubes grow bigger when transporting a lot of protoplasm. By growing bigger, the tubes are better suited for future transport as bigger tubes offer less resistance to the protoplasmic flow. Tubes that do not transport enough protoplasm shrink and eventually disappear due to a lack of flow. This mechanism is

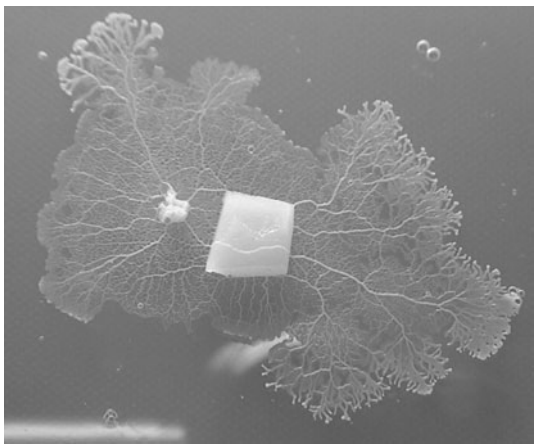

Fig. 1 *Physarum polycephalum* on agar surface. The slime mould is presented with a food source (centre of image) and uses a network of fine tubes during foraging to transport nutrients. The principles of network formation can be used to design fault tolerant networks. Image courtesy of Dr. Tanya Latty, University of Sydney

essentially a feedback loop that reinforces active paths and eliminates unused ones.

All research on *Physarum* and the ease by which it can be cultured has led to various applications. Tsuda et al. (2004) and Jones and Adamatzky (2010) show how *Physarum* can implement logic functions while Adamatzky (2009, 2010) show that *Physarum* can be used as an unconventional computer and Miranda et al. (2011) explores the ability of *Physarum* to generate sound.

A more graph-related application of *Physarum* can be found in Nakagaki et al. (2000, 2001) which show that *Physarum* is capable of finding the shortest path through a maze. Aside from navigating a maze, *Physarum* was also shown to be able to anticipate periodic environmental changes in Saigusa et al. (2008). This behaviour is remarkable, considering the fact that *Physarum* has no central coordinating consciousness, and indicates an advanced underlying foraging strategy. More evidence of the capabilities of the slime mould is presented in Latty and Beekman (2009) and Dussutour et al. (2010) where the organism is shown to optimize its foraging activity to obtain an optimal nutritional diet.

The networking capabilities were further studied in Nakagaki et al. (2004a, b) by letting the slime mould connect multiple FS. A lot of attention has also gone to redesigning existing road networks. Adamatzky and Alonso-Sanz (2011) and Adamatzky (2011) model the cities in the Iberian Peninsula and Mexico respectively as food sources and let the slime mould connect them. The resulting networks are compared to the road networks present in those areas. Road networks however are subject to historical evolution and geographical constraints, both having an influence on the resulting networks which cannot be neglected.

To capture the fundamentals of the slime mould, Tero et al. (2007, 2008) formulate a mathematical model of the inner workings of the slime mould. In Tero et al. (2010), it is used to redesign the rail network in the Tokyo area. The result of *Physarum* was very close to the existing network, proving its applicability to the network design problem. Another model of the fundamentals of the slime mould is presented in Jones (2009) using a multi-agent approach. This model was applied in Becker (2011) showing that it is close to the natural mechanism of *Physarum*, but it fails to find the shortest path through a maze.

In this paper, the mathematical model and algorithm from Tero et al. (2007) are adapted to design fault tolerant networks as needed in for instance telecommunication. Telecommunication networks require high fault tolerance as they have to operate continuously and are prone to link and node failure (power outage, cable break, …). Networks have to be designed to handle these problems (Vasseur et al. 2004; Pickavet et al. 2006) but at the same time an operator wants to avoid over-dimensioning and unnecessary costs. As an important part of the installation cost is trenching and digging (Casier et al. 2008), low total lengths of the networks are desired. To better achieve these goals, 2 extensions are made to the model to allow more automated and unambiguous design. The extended model is less representative of the biological slime mould but better equipped to generate fault tolerant networks.

2 Mathematical model

The basis for the mathematical model further used in this article is developed in Tero et al. (2007). In this model, the FS are represented by nodes which are interconnected through a network of tubes, modelled by edges in a graph. During execution of the algorithm, the properties of the edges constantly change until a state of equilibrium is reached. A typical simulation is given in Fig. 2.

The space in which the network will be built initially only contains nodes representing food sources (see Fig. 2a). To interconnect the FS, a fine mesh of nodes is added to the space along with edges connecting these nodes as shown in Fig. 2b. This models a network of fine tubes which transports the nutrients from one FS to the rest of the organism. All edges are assumed to be bidirectional.

The flow through an edge is a result of the pressure differential between the edge's endpoints. This type of flow is Poiseuille (Kamiya 1959; Batchelor 2000) and can be expressed as

$$Q_{ij} = \frac{\pi a_{ij}^4}{8\kappa} \cdot \frac{p_i - p_j}{L_{ij}} \tag{1}$$

where a_{ij} is the radius of the tube, κ the kinematic viscosity of the fluid, L_{ij} the length of the edge, p_i the induced

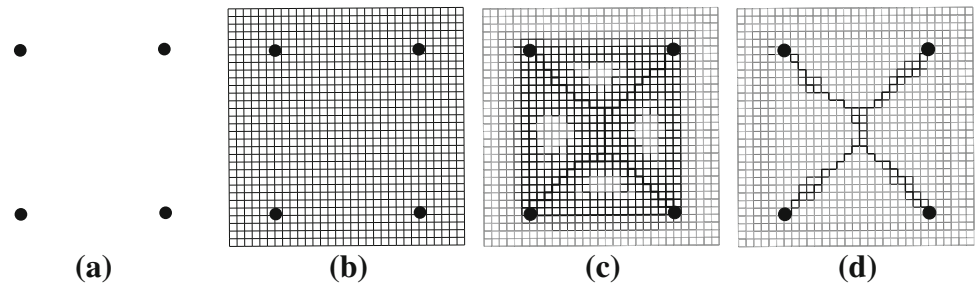

Fig. 2 Typical simulation: **a** the food sources at the start of the algorithm. A fine square mesh of nodes and edges is then added to the food sources in (**b**). Only the edges connecting the nodes are shown as lines, nodes are situated on all line intersections. **c**, **d** The network changing during the simulation. The conductivities of the edges are denoted by the thickness of the edges with thick edges representing edges with a high conductivity

pressure in the node N_i and Q_{ij} the flow through the edge between N_i and N_j. The first term in the right hand side of the equation can be contracted to a single variable, simplifying the equation to

$$Q_{ij} = \frac{D_{ij}}{L_{ij}} \cdot (p_i - p_j) \tag{2}$$

where D_{ij}, the conductivity of an edge, indicates the suitability for transport. A high D_{ij} value indicates that the edge offers low resistance for transport (per unit length) while an edge with a low D_{ij} value offers more resistance and is less suited for transport.

Based on Eq. (2), an iterative algorithm can be developed that modifies the initial network and its edges to create a network connecting the FS. In each iteration, a pair of FS is randomly selected between which a flow is set up. When a flow is imposed between a source and a sink node, it will spread throughout the network according to the following set of equations

$$\sum_j \frac{D_{ij}}{L_{ij}} \cdot (p_i - p_j) = \begin{cases} I & N_i \text{ is source} \\ -I & N_i \text{ is sink} \\ 0 & \text{else} \end{cases} \tag{3}$$

where I denotes the size of the imposed flow. These equations express the conservation of flux inside the network: for each node N_i which is not a source or a sink in the network, the total amount of flow entering the node must equal the total amount of flow exiting the node. Only in the source and sink node can the transported fluid leave the network.

The flow through each edge of the network Q_{ij} can be calculated by combining Eq. (2) with the solution of Eq. (3). Using these flows, the conductivities of all edges can be adjusted, simulating the growth of the tubes in the slime mould. This adaptation can be done by using the following equation

$$\frac{D_{ij}^{n+1} - D_{ij}^{n}}{\Delta t} = f(|Q_{ij}^{n}|) - D_{ij}^{n} \tag{4}$$

where $f(|Q_{ij}^{n}|)$ denotes the growth function of the tube which is chosen to mimic the behaviour of the real slime mould. For low flow values, the edge should decay while for higher flow values, the edge should thicken. There is however a limit on the size of the tube. These constraints translate to a monotonically increasing function saturating for high values. The function used further in this paper is

$$f(|Q|) = \frac{(1+a)|Q^{\mu}|}{1 + a|Q^{\mu}|} \tag{5}$$

with $1 \leq \mu \leq 2$, $a > 0$ resulting in a sigmoid curvature for $f(|Q|)$.

After the adaptation, the algorithm chooses another pair of (source, sink) nodes and repeats the calculations. The (source, sink) pairs are randomly selected with each node having a predetermined probability of being selected. A node can be made more important by increasing the probability of being selected. A higher chance to be selected results in more flow passing through the tubes surrounding the node, possibly resulting in more edges surviving.

Another optimization consists of calculating and averaging several flows in one iteration before the conductivities are adapted and is presented in Watanabe et al. (2011). This can be used to limit fluctuation of Q_{ij} and D_{ij}. Ideally, all (source, sink) combinations should be calculated in one iteration. This would require solving a lot of linear systems as there are $\frac{n \cdot (n-1)}{2}$ such combinations. Only a few combinations are used in the same iteration as a trade-off between combinatorial completeness and execution time. Fluctuation on the values was sufficiently reduced by this (limited) averaging.

3 Extensions

To improve the fault tolerance of the created networks and automate the design process, we propose two extensions to the algorithm of Tero et al. (2007). These extensions consist of the migration mechanism (Sect. 3.1) and the stimulation mechanism (Sect. 3.2). As these extensions are intended to improve fault tolerance, the end results of the extended algorithm will differ from the networks created by the

biological slime mould and the original algorithm. The general outline of the algorithm is given in Algorithm 1. As explained in Sect. 2, the core of the algorithm consist of iteratively selecting a (source, sink) pair, determining the flow through the network and updating the model instance. One iteration consists of lines 3 through 12. The (source, sink) pair is selected randomly with each node having a predetermined probability of being selected. When multiple (source, sink) combinations in the same iteration are wanted to reduce fluctuation of Q_{ij} and D_{ij}, lines 4 through 8 are executed multiple times and the resulting Q_{ij}^k averaged. The algorithm stops on line 13 when the average fluctuation of the D_{ij} falls below a predetermined threshold or after a certain number of iterations.

Algorithm 1: The extended algorithm in pseudocode

```
1  initialization;
2  repeat
3      for k ← 1 to nrPairs do
4          Select new (source, sink) pair
5          Calculate flows Q^k_ij;                    // Eq. (2) & (3)
6          if Q^k_max > τ · I then
7              stimulate alternative paths;           // Sect. 3.2
8          end
9      end
10     Average flows Q^k_ij to Q_ij
11     Update D_ij;                                   // Eq. (4)
12     Migrate nodes;                                 // Sect. 3.1
13 until stopping criterion;
```

3.1 Migrating nodes

The model from Tero et al. (2007) as presented in Sect. 2 assumes all nodes and edges to be at a fixed location. While this is useful to determine the optimal route to be followed through a maze (Nakagaki et al. 2000; Tero et al. 2007), it is less convenient in situations where the location of only some nodes is predetermined. By constraining the position of the nodes and edges, the result of the original algorithm is limited to the paths present in the initial mesh. The type of mesh used then determines the possibilities in the end result. Optimal paths (e.g. straight edges) not present in the initial mesh can be unintentionally excluded. Moreover, the presence of alternative paths can influence the flow distribution at the start of the algorithm by taking their part of the flow. This alters the reinforcement of the edges and ultimately the competition between the different edges.

A possible approach to approximating the optimal paths in the initial mesh could be to use a very fine-grained mesh. This would however increase the number of edges significantly, which in turn would slow down computations severely as the number of equations in the linear system to be solved increases. The optimal configuration could then still not be present in the initial mesh. Another possibility is to use non-uniform meshes with an increased number of edges in the areas of interest but this could favour specific configurations.

To preserve fairness among the edges in the initial mesh and to limit the complexity of the linear system, we propose to let the nodes in the mesh move. By using a simple moving mesh, more mobility is incorporated in the extended algorithm. This effectively reduces the influence of the initial mesh on the end result as the edges can be redistributed. It also simplifies post-processing of the networks, e.g. no more need to manually straighten edges. At the end of each iteration, the coordinates of the nodes are adjusted according to the flows at the nodes themselves. This adjustment is not applied to nodes representing FS, as they have to remain fixed. For each other node N_i, a set of 'target' coordinates T_i is calculated as follows

$$T_i = \frac{\sum_{\forall j} |Q_{ij}| \cdot C_j}{\sum_{\forall j} |Q_{ij}|} \tag{6}$$

These T_i coordinates are weighted sums of the coordinates of the neighbouring nodes, C_i, with the flows on the edges between them acting as weights. The coordinates will be two-dimensional when on a planar map, but they can easily be extended to support higher dimensional spaces. Figure 3 shows the basic mechanism at work. The thickness of the edges between the nodes represents the amount of flow between the nodes. N_2 receives flow from N_1 and sends the bulk part of it to N_3. A smaller amount of flow is sent to N_4. When T_2 is calculated, N_3 has a larger impact than N_4. T_2 is situated closer to the straight edge between N_1 and N_3 than C_2.

Once the T_i's are calculated, the nodes will migrate towards their target location. If in the following iterations in the extended algorithm a similar flow distribution is encountered, the path of the dominant flow will be shorter, requiring less energy for transport. To prevent extensive migration, node movement is restricted to the area around the initial position. To this end, the movement vector M_i is calculated by $\overrightarrow{M_i} = T_i - C_i^0$ with C_i^0 denoting the coordinates of N_i at the start of the algorithm.

The movement vectors are then used to calculate (and limit) the distance of T_i to C_i^0. To limit the migration to a disc of radius ϵ around the initial coordinates, it suffices to

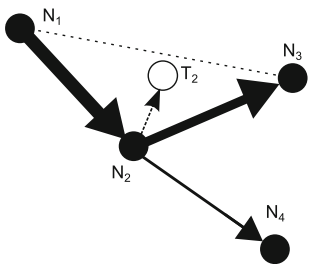

Fig. 3 Migration mechanism applied on a single node. N_2 wants to migrate towards T_2 which is predominantly determined by N_1 and N_3 because of the size of the flows towards and from N_2, respectively

calculate the norm of $\overrightarrow{M_i}$ and clip those movement vectors that have a norm $> \epsilon$. To find the final target coordinates (FC_i), the clipped movement vectors can be added to the original coordinates C_i^0. More elaborate schemes can be used to prevent the node from going into forbidden areas e.g. walls in a maze.

Using the current coordinates C_i^n, the new coordinates of the nodes are then calculated as

$$C_i^{n+1} = \frac{C_i^n + \psi \cdot FC_i}{1 + \psi} \tag{7}$$

where ψ is used to smooth migration over several iterations in the algorithm. As mentioned in Sect. 2, only a few (source, sink) combinations are used each iteration. As a result, the T_i can vary across iterations. The smoothing prevents excessive fluctuation of C_i. In the later iterations, most redundant edges have disappeared, resulting in fewer contributions to and fluctuation on T_i. The new coordinates can then be used to calculate the new lengths of the edges in the next iteration.

3.2 Stimulation of alternative paths

A second extension to the model from Tero et al. (2007) focuses on improving the fault tolerance of the resulting networks. The biological slime mould forms its networks by thickening the tubes that carry a lot of flow. The cultivated *Physarum* networks consist of many tubes with varying radii. To extract a useful network from these experiments, edges have to be selected. In Adamatzky and Alonso-Sanz (2011), the selection is done based on weights associated with the edges. The weights are calculated as the ratio of the experiments where the edge occurred to the total number of experiments performed. The authors use a threshold to identify important edges. The threshold greatly influences fault tolerance, as using a low threshold and retaining a lot of edges will result in a higher fault tolerance than when only the thickest tubes are retained.

Furthermore, once a thick tube is formed between 2 parts of the network, it tends to carry most flow between the 2 parts (Fig. 4a). The dominance leaves only little flow for the alternative paths which then often disappear due to the lack of reinforcement. This lack of alternative paths is reflected in a low fault tolerance.

To prevent paths from becoming too dominant, the flows in the extended algorithm are redistributed when a node has too much flow passing through it before they can affect the edge's conductivity. First, the node K with the maximal amount of flow going through it, Q_{max}, is determined by

$$Q_{max} = \max_n \frac{1}{2} \sum_i |Q_{in}| \tag{8}$$

$$K = \arg\max_n \frac{1}{2} \sum_i |Q_{in}| \tag{9}$$

If Q_{max} is larger than some (predetermined) percentage of the total flow $(\tau \cdot I)$, the total flow should be redistributed. The total flow can be divided in 2 parts: a part Q'_{ij} that represents an amount of flow passing through K and the remainder that follows alternative routes. An approximation of Q'_{ij} can be found by solving

$$\sum_j \frac{D_{ij}}{L_{ij}} \cdot (p'_i - p'_j) = \begin{cases} -2Q_{max} & N_i = K \\ Q_{max} & N_i = N_r \text{ or } N_k \\ 0 & \text{else} \end{cases} \tag{10}$$

$$Q'_{ij} = \frac{D_{ij}}{L_{ij}} \cdot (p'_i - p'_j) \tag{11}$$

where N_r and N_k refer to the original source and sink used in the current iteration of the iterative algorithm, respectively. The set of equations are similar to Eq. (3), only now the original source and sink act as sources, both sending a flow of Q_{max} to K. The resulting Q'_{ij} values represent an amount of flow Q_{max} going from the original source to the original sink, passing through K (Fig. 4b). The remainder is then given by $Q_{ij} - Q'_{ij}$. To increase the flow through alternative paths (Fig. 4c), it suffices to set the new Q_{ij} as follows

$$Q_{ij}^{new} = \alpha \cdot \tau \cdot I \frac{Q'_{ij}}{Q_{max}} + \beta \cdot (1 - \tau) \cdot I \frac{Q_{ij} - Q'_{ij}}{(I - Q_{max})} \tag{12}$$

where α and β can be set to increase the relative importance of the terms. The Q_{ij}^{new} values can then be used in Eq. (4) to adapt the conductivities.

The first term in Eq. (12) contains the Q'_{ij} values representing a flow of size Q_{max} from the source to the sink passing through K. This flow is rescaled to a flow of size $(\tau \cdot I)$. As the redistribution is only done when too much flow passes through a node, this rescaling lowers the amount of flow passing through the forming dominant path. The second term represents flow from source to sink using other paths. This flow increases in size, resulting in an increased reinforcing of the alternative paths to improve the fault tolerance of the network.

Because the stimulation mechanism is integrated in the extended algorithm, the fault tolerance is integrated in the design process. This results in conductivities that are coupled with the importance of the edges for fault tolerance. The selection of edges based on conductivities/radii now more accurately captures the fault tolerance objectives.

4 Simulations

This section presents the results of the extended algorithm. First, the general methods used in the simulations are briefly explained in Sect. 4.1. Then, two more technical aspects of

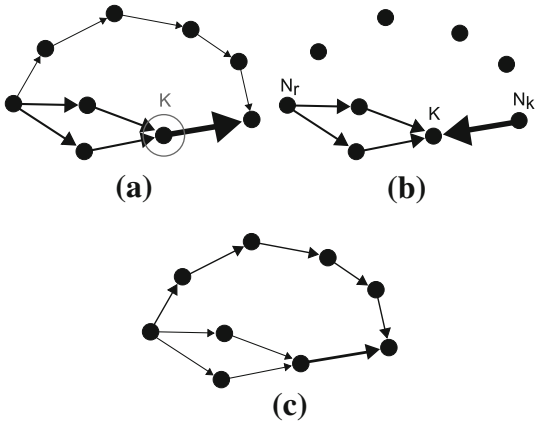

Fig. 4 Steps in stimulation process: **a** too much flow passing through node *K*. The flows converging in K are calculated by Eq. (10) and shown in (**b**). The flow is redistributed by Eq. (12) as shown in (**c**)

the algorithm are discussed. The extended algorithm is shown to find the shortest path through a maze in Sect. 4.2 and the applied thresholding is justified in Sect. 4.3. Section 4.4 shows the benefits of the extensions while Sect. 4.5 relates to the work in Adamatzky and Alonso-Sanz (2011). An application to telecommunication networks is given in Sect. 4.6.

4.1 Methods and parameters

The starting networks used in the following simulations were generated by considering the points to be connected as FS, characterized by their Euclidean coordinates. To these FS nodes, a square 100 × 100 mesh was added with each FS connected to its closest neighbours. As the mesh will be changed by the migration mechanism, the simple square configuration suffices to get different interconnection structures. The linear systems of equations were solved by calculating the minimum norm residual solutions. To minimize fluctuations on D_{ij}, $nrPairs = 2$ different (source, sink) pairs were calculated and averaged. Unless specified otherwise, the different pairs were randomly generated with each of the FS having equal probability of being selected.

The parameters used in the calculation of flows were $(a, \Delta t) = (1, 0.01)$. The ψ-parameter used in the migration was set to 0.3. The clipping of movement vectors was done using an increasing disc size $\epsilon = (0.15 \cdot l) \cdot 1.0025^i$ with l the average initial link length and i the current iteration. The parameters were chosen to limit the migration in the first 1000 iterations. The stimulation parameters were set to $(\alpha, \beta, \tau, I) = (1, 2, 0.5, 1)$. The simulations were stopped after 10000 iterations. This stopping criterion could be improved as the network evolution stabilized sooner and final network structure could be extracted after fewer iterations.

To evaluate the simulation results, several metrics are calculated for comparison. The total length is the sum of the lengths of all edges present in the end result with a conductivity higher than 5 % of the maximal conductivity present (see Sect. 4.3). All lengths and distances are calculated using the Euclidian distance measure. The single (resp. double) fault tolerance indicates which percentage of single (resp. double) link faults can be handled by the rest of the network. A fault is successfully handled when all nodes are still connected after the link has failed. The probability of a link failing is taken to be proportional to its length. This definition is closely related to the availability of a (telecommunication) network, which indicates the percentage of time a network is operational. The calculation of availability would however require estimating realistic recovery times and link failure rates. The diameter of a network is the longest path among all shortest paths in the graph. The fault diameter is the maximum diameter of the graph after a fault has occurred (Krishnamoorthy and Krishnamurthy 1987). The average internodal distance is calculated by averaging the shortest paths between all pairs of nodes.

To compare the simulation results to more theoretical graphs, the Gabriel Graph (GG), Relative Neighbourhood Graph (RNG) and Minimum Spanning Tree (MST) are used. These graphs are part of the Toussaint hierarchy of proximity graphs (Jaromczyk and Toussaint 1992): MST $\subseteq$ RNG $\subseteq$ GG and can be created based on the coordinates of all nodes. The GG is created by adding an edge between node a and node b if no other node is in the closed disc with line segment $|ab|$ as diameter. The RNG is constructed by adding a link between two nodes if no other node is closer to both nodes than they are to each other (Toussaint 1980). The MST is the graph with the lowest weight that still directly interconnects all nodes. The length of the MST is a good indication of the minimal length of any network connecting all nodes (without Steiner points). The Euclidean Minimal Spanning Tree (=MST with inclusion of Steiner points) was not used as the difference in length was very small ($\sim$3.5 %) and both have no redundancy.

4.2 Validation

As the original model is extended with two extensions, the path-finding abilities of the model might not be present in the extended model. Figure 5 shows the result of the extended algorithm on a maze. The walls were considered forbidden territory during migration. The extended algorithm can (still) find the shortest path through the maze. The extended algorithm cannot attain maximal single fault tolerance as there are no completely disjoint paths from source to sink. The result does contain two alternatives which increase fault tolerance. These alternatives can be

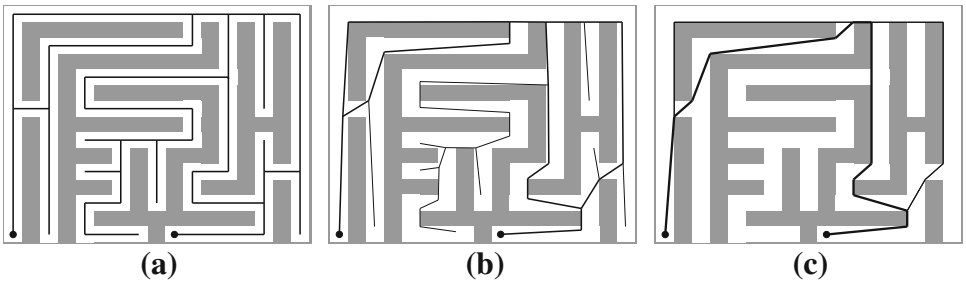

Fig. 5 Simulation of maze-solving capabilities. **a** Start of the simulation with all edges having the same conductivity. **b** The dead end cutting after a few iterations and **c** the final state of the network. Edges with a high conductivity are drawn thicker than edges with a low conductivity

eliminated by increasing the μ-parameter, resulting in a trade-off.

During the simulations, the two phases from Tero et al. (2007), dead end cutting and selection of the solution path from the competitive paths, are also observed. The dead ends in the maze do not receive flow due to the lack of FS and die out quickly. The competition between the different paths is more complex than the dead end cutting and requires more iterations.

4.3 Effects of extensions on thresholding

During analysis of the end results of the simulations, a threshold is applied on the conductivities of the edges. This thresholding discards all edges with a low conductivity as they offer too much resistance to the flow. The percentage of edges surviving in a simulation using the extended algorithm is shown in Fig. 6. Using low thresholds results in more edges from the initial graph 'surviving' the computation. A threshold of 0.001 % (of the maximum conductivity present) results in 4.5 % of the initial edges being present in the end result. As the initial graph contained a lot of redundant edges, added along with the fine mesh, it is natural that only a small percentage is left in the end. Increasing the threshold results in more edges being considered unsuitable and cut from the end result. This lowers the total length but could eliminate alternative paths, lowering the fault tolerance of the networks. However, as Fig. 6 shows, there is a clear separation in the conductivity values. In the experiment of Fig. 6, no edges were present in the end result with a conductivity value in the range of 0.2–7 %. Varying the threshold in this range would not influence the end result (no extra edges would be dropped). As this separation was present in all simulations, a static threshold could be applied independent of all other parameters. Based on a limited number of simulations, the threshold was set at 5 % and used in all further simulations.

4.4 Comparison to original model

To show the effects of the extensions, the original and the extended algorithms are used on a model of Belgium. A set of 16 Belgian cities was made based on the number of

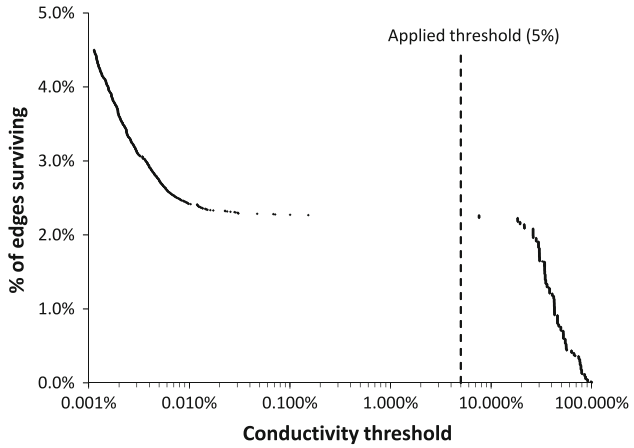

Fig. 6 Relationship between edge survival and the applied conductivity threshold. The graph shows the percentage of edges present in the end result (compared to the total number of edges in the initial mesh) when the conductivity threshold is varied. Using a low threshold results in a lot of edges remaining while higher thresholds result in more edges being cut. No edges were present in the end result with a conductivity value in the range of 0.2–7 %, resulting in a wide gap around 1 %. The threshold of 5 % used in all experiments is denoted by the *dashed line*. Threshold values are displayed using a logarithmic scale

inhabitants and/or regional importance: Aalst, Antwerp, Arlon, Bruges, Brussels, Charleroi, Ghent, Hasselt, Kortrijk, La Louvière, Leuven, Liège, Mechelen, Mons, Namur and Wavre. Figure 7 shows some simulation results of the extended algorithm together with the MST, RNG and GG. The importance of Antwerp and Brussels was increased to incorporate their relative importance. Figure 7a shows how the extended algorithm connects all cities. By increasing μ in Eq. (5), fewer edges survive as shown in Fig. 7b, c, similar to results in Tero et al. (2007). The RNG and MST do not have high fault tolerance as several cities can be disconnected by a single fault.

Figure 8 compares the results of the original algorithm to those of the extended. The μ-parameter from Eq. (5) was varied between 1.3 and 1.9 to obtain the simulation results, all other parameters were as described in Sect. 4.1. The analysis results varied nicely with μ. High μ-values resulted in few edges surviving, corresponding to the results in the left part of the graph. When μ was lowered, the resulting graphs had more edges resulting in an

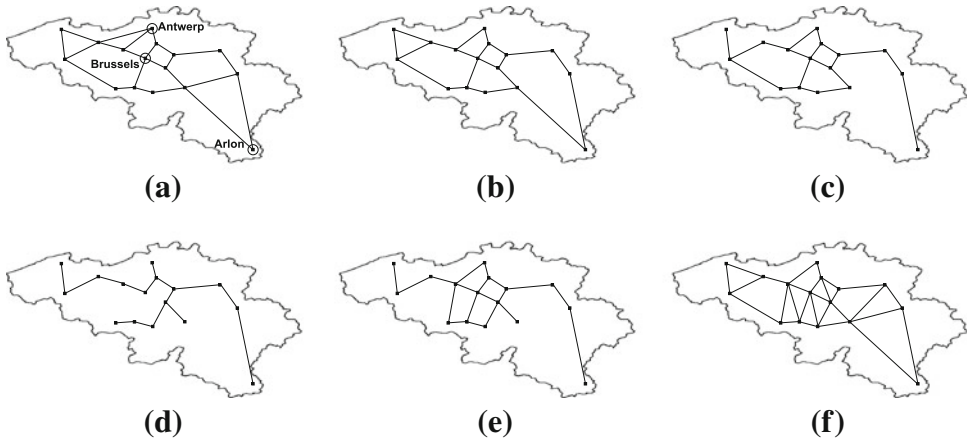

Fig. 7 Results for Belgian network: **a**–**c** Some results for the extended algorithm with constant stimulation parameters and μ respectively 1.3, 1.4 and 1.6. **d**, **e** and **f** The MST, RNG and GG connecting the cities

increased total length and fault tolerance. The range of μ-values was chosen to show the trade-off between cost and fault tolerance.

The effect of the stimulation mechanism can be best seen in the top portion of the fault tolerance graphs. The extended algorithm can attain maximal single fault tolerance while the original algorithm cannot. Arlon, the most southern city on the map, is relatively far away from the rest of the graph making it very costly to connect to the rest of the graph with more than one path. While an extra path greatly increases the fault tolerance, it also increases the total network length significantly. This results in very few analysis results (for the extended algorithm) around 1.6 MST_{length}. Both situations (extra path and no extra path) can also be seen in Fig. 7b, c. The influence of the increase in μ is too big for the stimulation mechanism to handle without increasing the stimulation parameters in Fig. 7c.

The effect of the migration mechanism is best seen in the left portions of the fault tolerance graphs in Fig. 8. On top of the (limited) fault tolerance increase of the stimulation mechanism, the length of the networks is shorter due to the paths being made straight by the migration. This shows the importance of the initial mesh for the end result.

4.5 Comparison to living slime mould

To compare the extended algorithm to the biological slime mould, the extended version of the algorithm is applied to the same input dataset of Adamatzky and Alonso-Sanz (2011) and the results are compared. The same set of cities on the Iberian Peninsula is used as input for the network generation. The biological network and the road network are recreated by using straight interconnections to obtain the same logical structures. The MST, RNG and GG are created based solely on the model of the Iberian Peninsula. Figure 9 shows some results of the extended algorithm. More important cities (Madrid, Lisbon) were given a larger weight in the selection process to incorporate their relative importance. Table 1 shows an analysis of the different networks.

The length of the existing road network and the length of the result of the living slime mould are much higher than the lengths of the simulation results. In the man-made network, this is caused by the multiple (redundant) roads starting in a.o. Madrid (centre of Spain). The length of the biological network depends on the cut value (Adamatzky and Alonso-Sanz 2011). By raising the cut value, lower lengths can be obtained but this would disconnect Madrid from the rest of the graph. Varying the cut value would only provide limited gains. The simulation results do not have this problem as the conductivities are tied to the fault tolerance. The cut value could be kept constant for all simulations as the conductivities of the edges sharply fell around the applied threshold (Sect. 4.3). The best (fault) diameter is found in the road network together with the highest total length.

All networks, aside from the MST, RNG and GG, have maximal single fault tolerance. The MST, RNG and GG cannot handle all single link faults as a single link failure can isolate Barcelona from the rest of the network. The fault diameter of these networks should be ∞ (infinite distance from any node to Barcelona). The values given in Table 1 for these networks however are calculated by ignoring faults that cause the graph to become disconnected. Fault diameters calculated by ignoring some failures are denoted with an asterisk (*). The RNG and GG were expanded by adding the link (Barcelona, San Sebastian) and the link (Malaga-Marbella, Cadiz) (already in GG). The numerical results of these networks are denoted by RNG+ and GG+. The results of the RNG+ are good, considering their construction time, and similar to the results of the extended algorithm. The extended algorithm does have notably smaller fault diameters. The average internodal distance is slightly lower for the simulation results and the double fault tolerance values of the simulation results are higher than the values for the

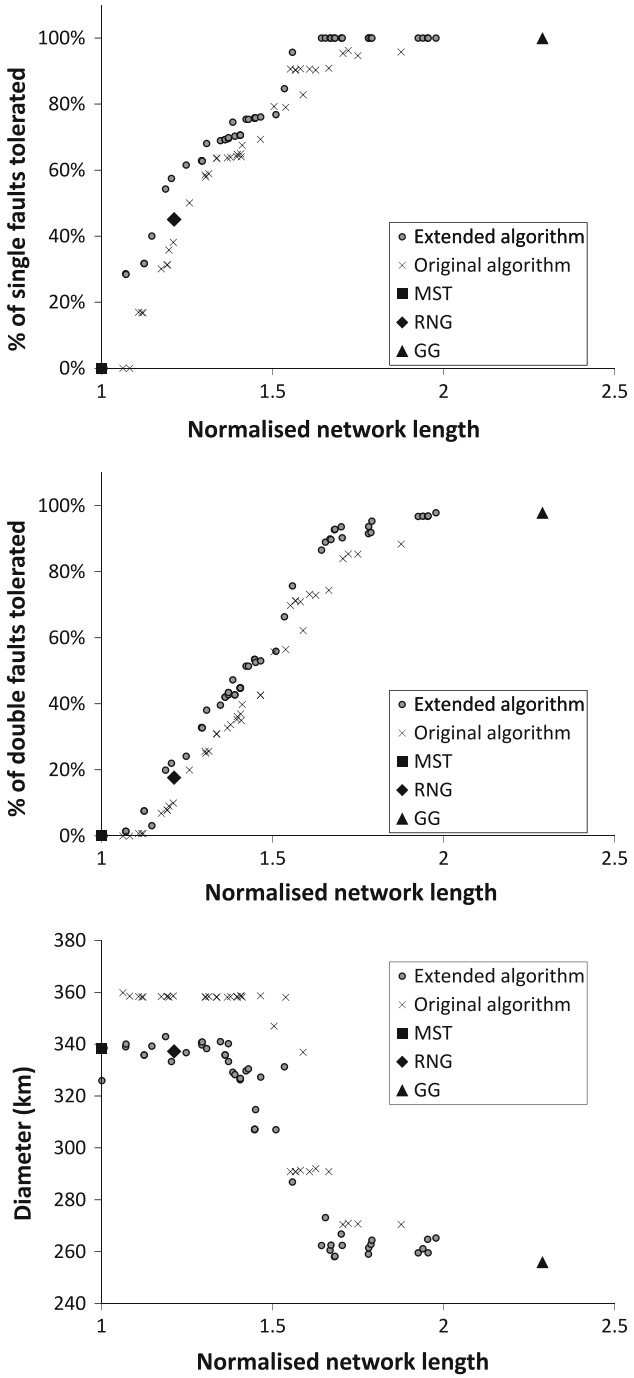

Fig. 8 Analysis of simulation results. The fault tolerance and diameter of the results are compared to the total length. The indicated lengths are calculated by dividing the total length of the networks by the total length of the MST. The simulation results using the model without using any extension are represented by *crosses*. The simulation results using both extensions are denoted by *circles*. The MST has unit length. The RNG and GG of the network are represented the *diamond* and the *triangle*

reference networks. The (human) adjustment to the RNG greatly increases its resilience but this dependency is not desirable for automated design of more complex networks. The GG+ also has good results but has a high network cost.

In summary, the results of the extended algorithm on the Iberian Peninsula show the benefits of the extended algorithm. Thanks to the coupling of the conductivity values to the fault tolerance, the cut value for retaining edges in the end result could be kept constant while it had to be determined manually for the living slime mould. Manual corrections, needed for the proximity graphs, were not needed in the results of the extended algorithm. The algorithm further allows to design networks with various levels of fault tolerance by varying μ.

4.6 Application to telecommunication networks

To show the potential of the extended algorithm to design telecommunication networks, it is compared to a realistic telecommunication network. The reference material for this network is available in Orlowski et al. (2010). Figure 10 shows some simulation results, the reference telecommunication network and the proximity graphs. The numerical results are shown in Table 2.

With the exception of the MST and RNG, all networks had maximal single fault tolerance. The RNG could handle 45.55 % of all link failures, the MST (by definition) none. The fault diameter was calculated as in Sect. 4.5. The results of the extended algorithm greatly resemble the reference telecommunication network. Figure 10h shows all edges present in both the reference network of Fig. 10g and the simulation result in Fig. 10b. Most differences are found in the centre of the graph. Only one city (Munich) has no edges that appear in both the reference network and the simulation result. Table 2 shows that the total length is lower, the average internodal distance, diameter and fault diameter are smaller and the double fault tolerance is higher. As in Sect. 4.5, the network designer can vary μ to design networks with a desired network resilience. Figure 10c attains a 13 % reduction in total length compared to the reference telecommunication network in exchange for a lower network resilience.

The reference RNG has a low single/double fault tolerance. The RNG construction process has difficulties connecting the cities due to the specific network topology. For several cities like Athens, Glasgow, Madrid, Stockholm and Warsaw an extra edge would be desirable to increase fault tolerance but there's always another city that causes the city to have no other edge (e.g. an edge between Athens and Rome would be desirable but Belgrade is closer to both Athens and Rome than Athens is to Rome, preventing the RNG construction process to add an edge). This effect was also visible in the RNG on the Iberian Peninsula and in the Belgian experiment. Manually adjusting the result is more ambiguous now as a lot of adjustments are needed to attain

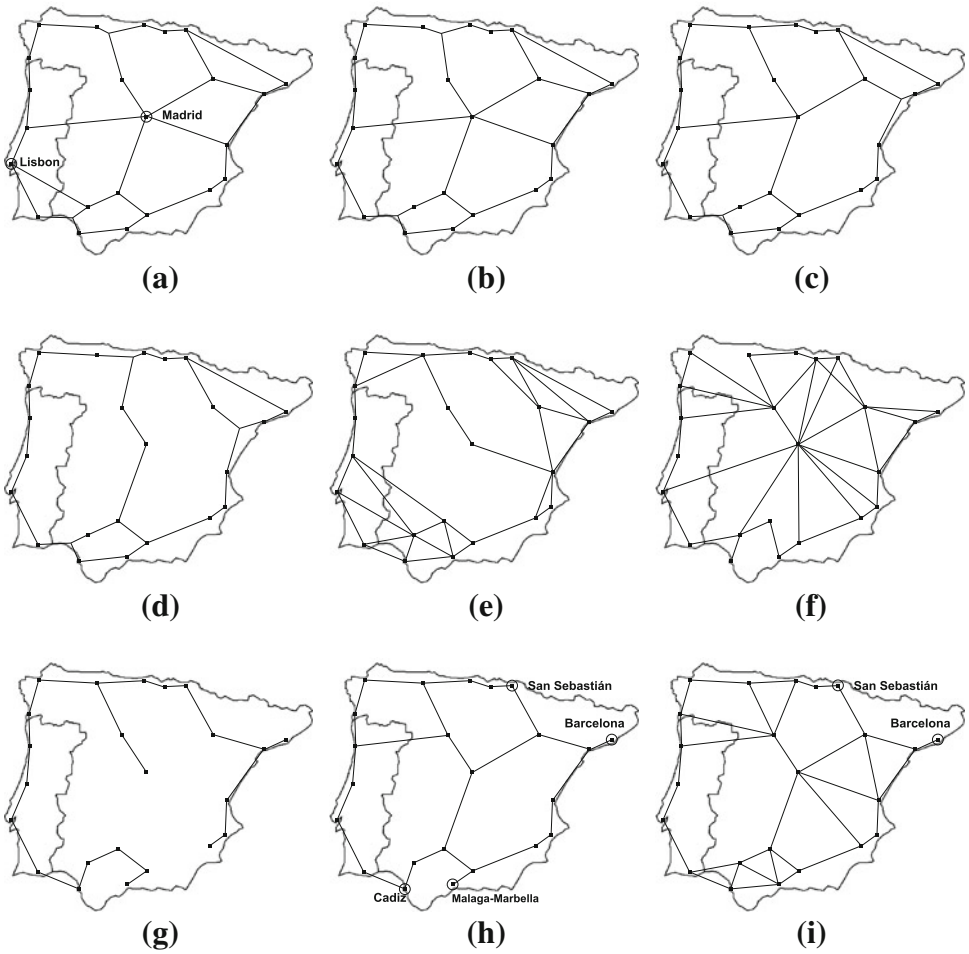

Fig. 9 Results for the Iberian Peninsula: **a–d** the results of the extended algorithm with varying μ values respectively 1.2, 1.3, 1.4 and 1.5. **e** The results of the living slime mould and **f** the existing road network, both taken from Adamatzky and Alonso-Sanz (2011). **g–i** The MST, RNG and GG connecting the cities

Table 1 Results for the Iberian Peninsula

Network	Total length (km)	Double fault tolerance (%)	Average internodal distance (km)	Diameter (km)	Fault diameter (km)
Simulation					
Fig. 9a	5946.18	98.37	618.95	1175.82	1563.10
Fig. 9b	5199.35	96.88	623.85	1175.64	1563.36
Fig. 9c	4957.57	96.15	643.38	1317.38	1565.75
Fig. 9d	4226.58	87.77	670.23	1372.04	2209.85
Reference					
Biological	7059.65	98.78	636.60	1284.12	2025.91
Roads	8373.09	96.99	600.31	1154.39	1294.85
GG	6335.71	96.33	595.81	1310.18	1524.65*
GG+	6738.59	99.32	594.36	1310.18	1524.65
RNG	4254.27	82.50	664.34	1337.40	1970.23*
RNG+	4825.37	95.99	649.64	1329.82	1714.45
MST	3121.00	0.00	1026.00	3121.00	3121.00*

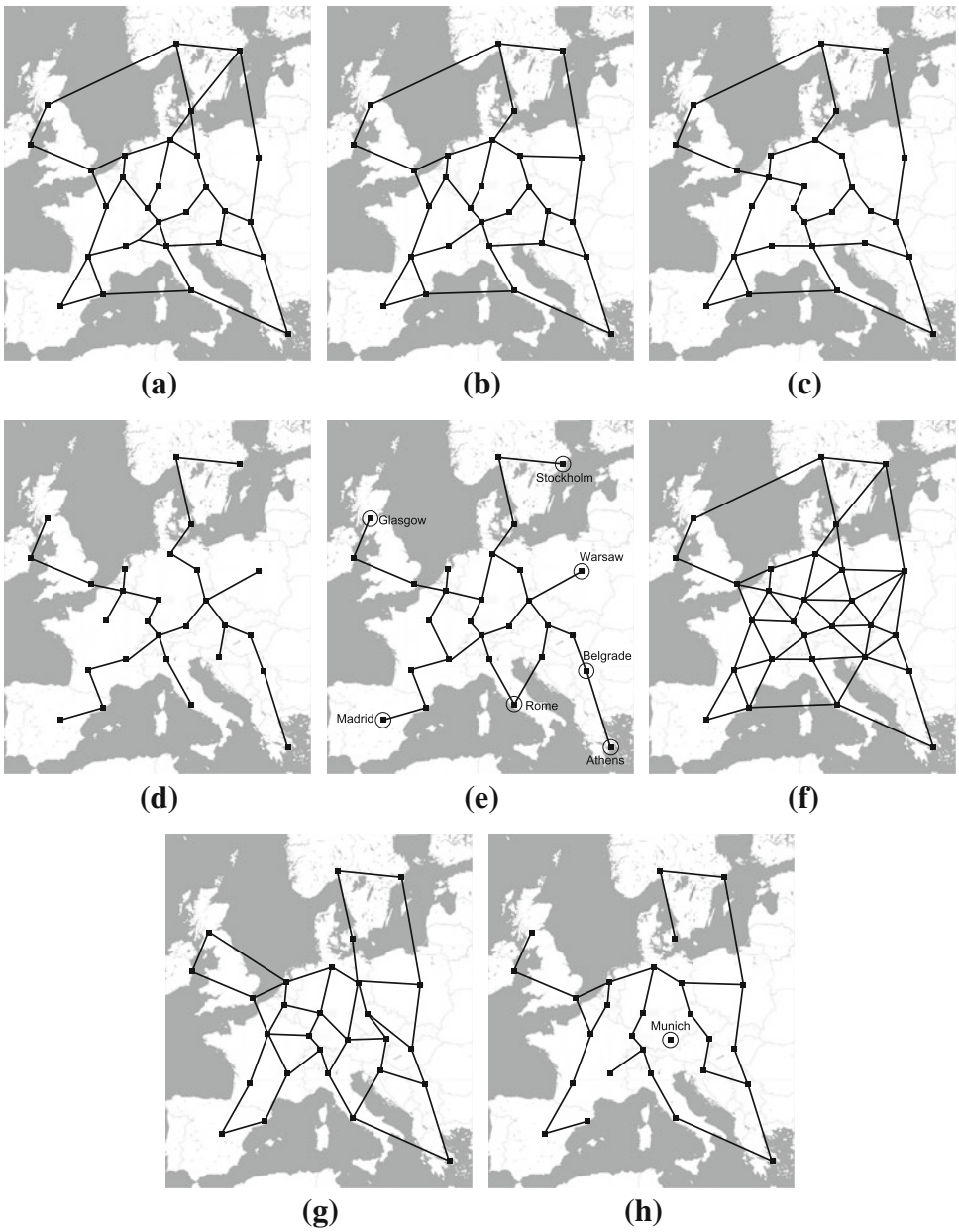

Fig. 10 Telecommunication networks: **a**–**c** some simulation results of the extended algorithm obtained with μ resp. 1.25, 1.35 and 1.55. **d**–**f** The MST, RNG and GG connecting the cities. **g** The reference network from Orlowski et al. (2010) while **h** is the graph containing all common edges between (**b**) and (**g**)

Table 2 Results for the European telecommunication network

Network	Total length (km)	Double fault tolerance (%)	Average internodal distance (km)	Diameter (km)	Fault diameter (km)
Simulation					
Fig. 10a	17587.99	98.25	1298.72	3443.64	4658.34
Fig. 10b	16751.63	97.77	1307.62	3443.02	4657.71
Fig. 10c	15039.37	95.37	1409.65	3391.08	5023.13
Reference					
SNDlib	17344.00	96.71	1360.82	3615.40	4763.34
GG	23946.12	99.29	1169.33	3213.70	4316.25
RNG	10612.77	16.65	1472.88	3888.82	4895.38*
MST	9361.08	0.00	1639.53	4381.95	4381.95*

maximal single fault tolerance. This shows that, despite its simplicity and the good results the RNG(+) produced for the Iberian Peninsula, its fault tolerance is dependent on the topology of the graph and can be quite low. For designing fault tolerant networks, this dependency is undesirable. The GG achieves maximal fault tolerance but at a 30 % increase in total length compared to Fig. 10b. The MST again has the shortest length but no fault tolerance.

5 Conclusion

In this paper, the potential of the true slime mould *P. polycephalum* to design fault tolerant networks is analyzed. The mathematical model from Tero et al. (2007) is implemented and adapted. While the resulting networks of the original algorithm can be steered towards fault tolerance, the original model has no special provisions focused on it. The fault tolerance in the networks is very sensitive to the threshold applied during post-processing edge selection on the generated networks. To reduce this dependency, we extend the original model with a stimulation mechanism that redistributes the flow through the network. This enables alternative paths to survive and results in higher fault tolerance and smaller fault diameters in the generated networks. The threshold used for edge selection could be kept constant as the conductivities were more tightly coupled to the fault tolerance and sharply fell around the threshold. Another dependency of the original model lies in the initial configuration. The initial positions of the nodes and their interconnections determine the possibilities in the resulting networks. By extending the model with a migration mechanism, the possible paths can change during execution, offering more freedom to the algorithm without severely increasing computation times. The extensions were tested on models of Belgium, Europe and the Iberian Peninsula. Our extended algorithm can achieve higher fault tolerances than the original algorithm and can be tuned according to the desired level of fault tolerance. Comparisons to reference road and telecommunication networks show that the extended algorithm can be used when fault tolerant networks are desired and total length is to be minimized. It can also overcome specific weaknesses in geometric graph designs such as the Minimum Spanning Tree and the Relative Neighbourhood Graph and find the shortest path through a maze. By varying the μ-parameter, a trade-off between the fault tolerance of the resulting network and its total length can be made.

References

Adamatzky A (2009) From reaction-diffusion to *Physarum* computing. Nat Comput 8:431–447

Adamatzky A (2010) *Physarum* machines. World Scientific Publishing Company, Singapore

Adamatzky A (2011) Approximating mexican highways with slime mould. Nat Comput 10:1195–1214

Adamatzky A, Alonso-Sanz R (2011) Rebuilding Iberian motorways with slime mould. BioSystems 105:89–100

Batchelor G (2000) An introduction to fluid dynamics. Cambridge University Press, Cambridge

Becker M (2011) Design of fault tolerant networks with agent-based simulation of *Physarum polycephalum*. In: IEEE congress evolutionary computation, pp 285–291

Casier K, Verbrugge S, Meersman R, Colle D, Pickavet M, Demeester P (2008) A clear and balanced view on FTTH deployment costs. J Inst Telecommun Prof 2:27–30

Dussutour A, Latty T, Beekman M, Simpson SJ (2010) Amoeboid organism solves complex nutritional challenges. Proc Natl Acad Sci USA 107(10):4607–4611

Gotoh K, Kuroda K (1982) Motive force of cytoplasmic streaming during plasmodial mitosis of *Physarum polycephalum*. Cell Motil 2:173–181

Jaromczyk JW, Toussaint GT (1992) Relative neighborhood graphs and their relatives. In: Proceedings of IEEE, pp 1502–1517

Jones J (2009) Approximating the behaviours of *Physarum polycephalum* for the construction and minimisation of synthetic transport networks. In: Unconventional computing, proceedings. Lecture notes in computer science, vol 5715. Springer, pp 191–208

Jones J, Adamatzky A (2010) Towards *Physarum* binary adders. BioSystems 101:51–58

Kamiya N (1959) Protoplasmic streaming. Springer-Verlag, Vienna

Kobayashi R, Tero A, Nakagaki T (2006) Mathematical model for rhythmic protoplasmic movement in the true slime mold. J Math Biol 53:273–286

Krishnamoorthy M, Krishnamurthy B (1987) Fault diameter of interconnection networks. Comput Math Appl 13:577–582

Latty T, Beekman M (2009) Food quality affects search strategy in the acellular slime mould, *Physarum polycephalum*. Behav Ecol 20(6):1160–1167

Miranda ER, Adamatzky A, Jones J (2011) Sounds synthesis with slime mould of *Physarum polycephalum*. J Bionic Eng 8(2): 107–113

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

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

Nakagaki T, Kobayashi R, Nishiura Y, Ueda T (2004a) Obtaining multiple separate food sources: behavioural intelligence in the *Physarum plasmodium*. Proc R Soc Lond B 271:2305–2310

Nakagaki T, Yamada H, Hara M (2004b) Smart network solutions in an amoeboid organism. Biophys Chem 107:1–5

Orlowski S, Wessäly R, Pióro M, Tomaszewski A (2010) SNDlib 1.0 survivable network design library. Networks 55(3):276–286

Pickavet M, Demeester P, Colle D, Staessens D, Puype B, Depré L, Lievens I (2006) Recovery in multilayer optical networks. J Lightwave Technol 24(1):122–134

Saigusa T, Tero A, Nakagaki T, Kuramoto Y (2008) Amoebae anticipate periodic events. Phys Rev Lett 100(1):018101

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

Tero A, Kobayashi R, Nakagaki T (2007) A mathematical model for adaptive transport network in path finding by true slime mold. J Theor Biol 244:553–564

Tero A, Yumiki K, Kobayashi R, Saigusa T, Nakagaki T (2008) Flow-network adaptation in *Physarum* amoebae. Theory Biosci 127(2):89–94

Tero A, Takagi S, Saigusa T, Ito K, Bebber DP, Fricker MD, Yumiki K, Kobayashi R, Nakagaki T (2010) Rules for biologically inspired adaptive network design. Science 327:439–442

Toussaint GT (1980) The relative neighbourhood graph of a finite planar set. Pattern Recognit 12:261–268

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

Vasseur J, Pickavet M, Demeester P (2004) Network recovery: protection and restoration of optical, SONET-SDH, IP, and MPLS. Morgan Kaufmann Publishers, Burlington

Watanabe S, Tero A, Takamatsu A, Nakagaki T (2011) Traffic optimization in railroad networks using an algorithm mimicking an amoeba-like organism, *Physarum plasmodium*. BioSystems 105(3):225–232