Biologic Transport Network Algorithms

Loes van Uffelen, Dennis Vinke, Joep Schyns

June 18, 2019
University of Twente

Keywords: Transport Networks, Multi-Agent Systems, Agent Based Modelling, Biologic, Artificial Ants, Slime Mold, Framework

1. Introduction

This report is a result of an assignment for the course Multi-Agent Systems at the University of Twente. A multi-agent system (MAS) is a system consisting of communicating intelligent agents that can make (autonomously) decisions in order to reach their objective. Agents behave strategically, which means that communication between agents includes cooperation, coordination and negotiation [1, p. 13]. Multi-agent systems can be simulated with an agent-based model (ABM).

We are interested in how biological networks could be applied to transportation problems. Biological networks are networks that are computed by algorithms inspired by nature. Often inspiration is drawn from animal species that perform swarming behaviour, such as locusts, schools of fish, ant colonies, bacterial growth or starlings. Emergent structures are a common strategy found in many of those animal groups. Other organisms grow in the form of an interconnected network as individual, such as fungi and molds.

Transport networks are an essential part of this interconnected world: everyday, huge numbers of people, resources, energy and data are moved. The ubiquity of transport networks makes it interesting to delve deeper into how those networks perform and how they could be optimised. Furthermore, networks can be designed to solve specific transportation problems. Natural behaviour for computing such networks can be used for that.

1.1. Challenge

In this report we address the best transport network problem by utilising ant colony optimisation algorithms. We chose for ant behaviour because they

have clear rules for communication. Therefore, we might be able to simulate their behaviour in favour of solving transportation problems.

In order to scope the project, a research question has been created as follows:
How can we create a framework to simulate biological transport network algorithms?

To answer the research question, ant colony optimisation algorithms will be used, and therefore we describe the following sub research question:
How can we simulate ant colony optimisation algorithms?

We define a transport network as a field with several points of interests (nodes) that are in some way connected to each other. By tuning the variables, multiple networks on a specific map with multiple nodes can be be created. Those networks each perform differently. Various transportation problems have specific needs in order to perform best. Performance factors include shortest path, pheromone communication cost, time and fault tolerance.

Included in the report is further elaboration on the problem domain and approaches in which we zoom in on related work. Thereby is investigated how studies use ant optimisation algorithms and from there we defined our own. Also slime molds are shortly discussed, as they in some ways behave similar to ants. Then the design of the multi-agent system is explained. Ant behaviour is simulated in Unity. Lastly, we conclude our results and discuss the method.

1.2. Ant behaviour

A nest of ants can be considered as a multi-agent system. Ant networks are a well-defined example of a self-organized system in which ants act as decision-making agents in order to discover and distribute food sources. Ant colony optimisation algorithms (ACO) are based on ant's foraging behaviour. Indirect communication by means of pheromone trails enables them to find the shortest path between food sources and the ant nest. They prefer to follow paths with a high amount of pheromone on it, thereby reinforcing the 'strength' of the path with their own pheromone deposits. On the other hand, less efficient routes or dead-end routes will be less desirable over time as pheromones evaporate. Pheromone quantities determine the probability

of whether a path will be followed by an ant, though alternative routes are discovered by individual ants by continuous randomised selection.

2. Problem Domain

Ants foraging behaviour is explored in various fields of study. Researchers use diverse levels of abstraction of ant network optimisation and apply them to problems in society [2]. Ant colony optimisation techniques are also a widely used approach to understand transportation problems [3]. There exists a set of distinct transportation problems rather than routing from origin A to destination B. To the best of our knowledge, related work addressed specific transportation problems utilising ant colony optimisation, hence we want to provide a framework that determines the 'best path' by tuning the importance of variables (the performance factors).

2.1. Traditional approaches to transportation problems

Studies researched the vehicle routing problem in which multiple vehicles serve customers from a central depot. Thereby multiple factors could play a role, such as vehicle capacity, customer satisfaction and fuel consumption [4, 5, 6]. Others considered cross-docking networks [7] or the travelling salesman problem [6, 8, 9, 2]. Most studies thereby define the performance of (simulated) networks by either looking at the shortest or fastest routes [10] and therefore also often referred to as efficiency.

In this report we do not consider the vehicle routing problem nor the travelling salesman problem. Firstly, in vehicle routing a certain capacity plays a dominant role we do not take into account. Secondly, a travelling salesman must visit all nodes exactly once, and return to the starting point using the path with the minimum cost. Rather we are looking for the minimal spanning tree, which is the shortest possible network connecting all the nodes. From there, efficiency and robustness can be enhanced by increasing the 'meshedness' of the network.

3. Multi-agent approaches to the problem

We develop a transport network algorithm based on state of the art ant optimisation algorithms. Although other research optimised ant optimisation algorithms for their specific goals, we use their studies to create our own algorithm.

3.1. Towards modelling ant behaviour

Not all ants behave equally: different species and even separate colonies of ants implement distinct forms of ant colony optimisation that suit best for their environment [11]. For this reason, an ant colony optimisation cannot be copied one-to-one and subsequently be exercised in various environments. In order to approximate natural ant behaviour, the most commonly occurring factors in ant colony optimisation are taken and applied in the transport network algorithm. This includes the use of pheromones to indicate paths to food sources [8] and a universal algorithm for exploration [12].

Ample research simplify their research problem into a discrete form [3, 13, 14, 15]. However, in the real world such discrete form often does not exist. Naturally, ants are not restricted to a limited set of possibilities each step[16], therefore the transport network algorithm is implemented in a continuous domain (although we are dependent on a set resolution).

Exploration and use of pheromone in this continuous world, can be seen as follows: in figure **??** an exploring ant can be seen. The green field represents the possible directions of the ant. The round dots represent pheromone traces deposited by other ants. First of all, the ant has the option to explore in a pheromone direction or in a non-pheromone direction. The intensity of the pheromone determines how likely the ant is to chose the pheromone direction. Ants deposit pheromones on the path they take. A direction with high pheromone intensity is likely to contain a food source. However, an ant always has the possibility to explore a new direction, which is beneficial as it exposes new food sources or reveals a shorter route to a known food source. When an ant has found food, it returns to the nest using existing paths, formed by those pheromones. By adding pheromones to this path, the ant accentuates the path towards the location with food. Thanks to the pheromones, the ant agents have a non-requested positive relationship with each other. The ant agents do not interact with each other directly. The result of an ant agent achieving its goal is helping the other agents achieving their goal, which seems a consequence relationship [17]. The information sharing between the agents is solely done with the pheromones. Even the information sharing with the help of the pheromones is done on a local scale as the agents only request the pheromone count of a node in a limited range.

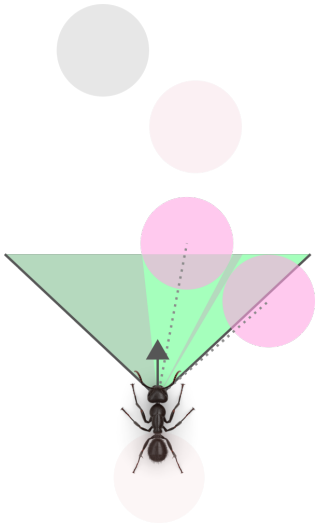


Figure 1: Ant exploration behaviour

3.2. Practical validation

Due to the season in which this research was carried out, naturally occurring ants could not be studied. In autumn, activity of ants drops due to the decreasing temperatures [18], even though food sources are present near the nest [18]. We tried studying ant behaviour on a warm day by captivating their movements on a white cloth using time lapse [19]; see Figure **??**. During the observation, a camera hung above a white sheet in front of multiple ant nest entrances. On the sheet, sugar was placed to imitate food sources. However, after 3 hours of observation only a few ants can be seen on the border of the sheet. No ants set foot on the sheet. Therefore, we conclude that either the sheet 'wiped out' pheromone trails or the temperature had dropped too much to study ants ourselves.

Figure 2: Screen capture of ant observation time lapse

3.3. Performance metrics

In order to evaluate networks made be simulated ants, factors should be introduced that form the performance metrics. Tero et al. defined performance of a slime mold network by transport efficiency, cost and robustness respectively by calculating the minimum distance, total length and fault tolerance [20]. Buhl et al. used comparable performance factors, also defining efficiency, cost and robustness, in ant networks [21]. We use similar performance factors, as defined below.

In other words, performance indicator *cost* is a factor indicating the total building cost of a network, which could be computed by looking at the number of pheromones in a network. Performance factor *efficiency* is the average distance between two food sources. Lastly, *fault tolerance* is a performance factor that indicates the robustness or redundancy of the network. Here, we can look into what happens with the network when one node is deleted: are there parts of the network becoming isolated?

By combining these factors in a three dimensional performance vector, we get the performance of a singular transport network (see figure 4). However, ants dynamically create networks, therefore for each attempt the end result can turn out slightly different. To balance out these inconsistencies, the sample mean performance can be calculated, see figure 5. Furthermore, we use normalised performance factors (see figure 3) to be able to compare between multiple network problems. This means that transport networks

with different placements of nodes could be compared.

Besides network performance, the required computational power to solve the problem in question is also of interest. The required computational power indicates the degree of optimisation of the script. Although, for in advance calculated problems, this factor is not important. Solely when a real time solution is required, this measurement becomes of importance.

$$\hat{cost} = \frac{cost}{|cost|}$$

Figure 3: Example of normalised performance factor

$$\hat{\mathbf{p}} = [\hat{cost}_i, \hat{faulttolerance}_i, \hat{efficiency}_i]$$

Figure 4: Transport network performance

$$\overline{\mathbf{p}} = \frac{\sum_{i=1}^{n} [\hat{cost}_i, \hat{faulttolerance}_i, \hat{efficiency}_i]}{n}$$

Figure 5: Ant transport network performance

3.4. Slime Molds

Besides ants, slime molds are widely studied for their abilities to create transport networks[22, 23, 24, 20, 25, 26, 26, 27]. Physarum polycephalum, a widely studied slime mold, shows similar characteristics to eusocial insects like ants. For instance, it can solve the shortest path problem. One advantage of this slime mold is its sensitivity to light: it avoids light, which can be applied as simulation method for mountains or lakes in transport networks. Besides redrawing rail and roadmaps, it has applications in robot control [28], bio-computing devices [29] and solving mazes [30].

Based on slime mold behaviour described in literature, we created an algorithm that can be applied in our tool set to simulate ant transport networks. By simulating the two biologic systems side by side, the two systems can be compared on their (1) cost effectiveness (2) robustness and (3) efficiency.

3.4.1. Method

Simply put, slime molds grow a plain of cells around an initial cell. The cells within the plain grow or shrink based the flux off food travelling trough

the cells. Cells that connect to food sources grow, and cells that not contribute to food distribution shrink and/or die off completely, see example in figure **??**.

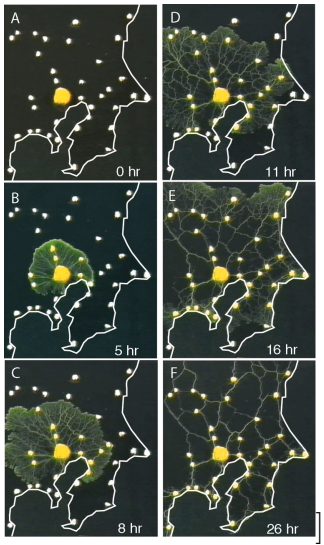


Figure 6: Imitation of the Tokio transport network by Slime molds[20]

We designed an equation to calculate the flux through each cell, see equation **??**. In equation **??** foodNodes are food sources the cell is connected to, directly or indirectly. ConnectedCells are the cells directly connected to a foodNode and distributedFlux is the flux originated by food emitted by a food source.

$$flux_c(t+1) = \frac{\sum_{i=1}^{foodNodes_c} \frac{flux_c(t)}{\sum_{q=1}^{connectedCells_i} flux_q(t)} * distributedFlux_i}{lengthPath_c}$$

Figure 7: Equation for calculating cell flux in a Slime mold network

Discussion. Slime molds and ants both create transport networks, each with their own unique methods. Although ant and molds are simple organisms, their behaviours can be hard to imitate in a non organic system. Equation **??** is a first step into understanding slime molds. The equation shows only a

part of the behaviour of slime molds, while it already includes many variables. These variables make or break the systems performance and likeliness of the imitated organic system. These variables need to be tested and tuned before any conclusion can be drawn on system performance. Due to the limited time span of this research project, we therefore chose to only focus on ant transport networks, as the ant algorithm also requires tuning and testing of multiple variables.

4. Design of a Multi-Agent Sytem

Transport networks connect points of interest based on requirements of a transportation problem. To enable development of biologic transport network algorithms, we design a framework that can handle multiple biological algorithms. To further narrow down the scope, the framework is build for continuous spaces. Continuous spaces means that agents are not restricted to a set of path possibilities, rather, agents are able to optimise with infinitely small changes. In the developed framework, network algorithms in continuous domains can be compared and tuned. Thereby, the advantages and disadvantages of various algorithms can be validated. Due to time constraints, we implemented only one biologic algorithm. Using the implemented algorithm we established a proof of concept for the framework.

The framework, see figure **??**, exists of three parts: (1) a simulation engine, (2) a agent behaviour component and (3) a component that calculates network performance when simulation is finished. To support different types of biologic network algorithms, a fundamental abstract class is created for (2). This class acts as a basis whereon different types of agent behaviour can be created. The logic required to expand basic behaviour, creates certain Agent types. Similarly, a abstract network node class is developed so that the agent behaviour can interact with the environment.

For biologic ant networks, pheromone nodes and food resources are deduced from the network node class. These nodes are then used by the agent class to create structure within the continuous working space. The framework code and intrinsic ant transport network algorithm code can be found at `https://github.com/DennisVinke/MAS-Ant-framework`

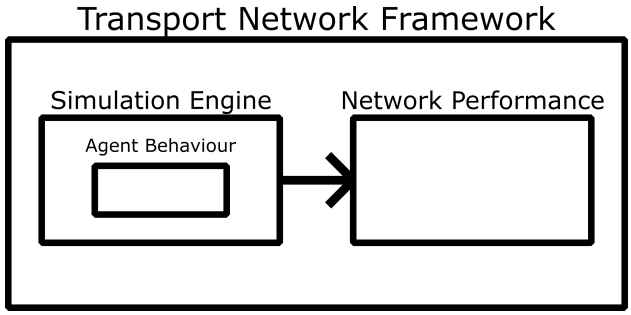


Figure 8: Transport Network Framework

4.1. Simulation; Unity as an engine

To make a framework for a continuous environment, known MAS toolboxes like NetLogo [13], Anylogic[31] and Repast [32] were not viable. Current toolboxes do not allow simulations outside of a discrete domain. Therefore, the gaming engine Unity [33] was chosen to build the framework upon. Unity allows a simple graphical interface and can cope with complex problems, such as continuous domains. However, using Unity implicated that the framework had to be implemented from scratch. Nonetheless the approach did offer lots of flexibility to design without any constraints.

4.2. Network node

Network nodes are nodes within the simulation environment that mark a location. These nodes can be created and connected by agents. When connected the network nodes form a graph containing the transport network. At the start of simulation the graph contains starting point(s) and goal point(s) for all agents. The starting nodes are unconnected. Agents can create a new node in the graph every time they find an undiscovered point in the world. The graph checks if this node already exists and based on that check, it will return the created node, or the node that already exists on that place in the world. The ant agents connect every node they visit with the node they previously visited. Thereby capturing their journey.

4.2.1. Pheromone node

Network nodes in ant network algorithms act as pheromone nodes. The pheromone nodes added by the ants play an important role in the coordination of this MAS. Every pheromone node acts as a simple data container. The existence of a pheromone marks the cost of the certain path in the ant network.

4.3. Agent behaviour

Agent behaviour components spawn based on a predefined starting position within the environment, after which they execute their transport network algorithm. The agents have one goal: find goal nodes in the environment. In nature these goal nodes would be something like a food sources. Abstract agent behaviour components have no means to execute environment exploration. Agent locomotion is algorithm specific.

4.3.1. Ants

The direction in which the Ant agent locomotes is based on several factors. First, the ant will decide if it wants to explore, or follow a known path. The explore probability is based on the formula found in figure **??**. If the ant decides to explore, it will choose a heading based on the previous heading. The new heading is taken from a heading range. For example, this could be 15 degrees of deviation of the previous heading.

When the ant does not decides to explore, it will go to the node that has the strongest pheromone value in sight. Once a goal node is found, the agent will track back to the initial starting place based on the path it took. The agent will now mark all the nodes with pheromones it encounters on its way from the goal node to the start node. The amount of pheromones a node gets is evenly distributed and based on the amount of pheromones an agent contains divided by the amount of nodes visited to create the path.

$$P_{explore} = \frac{1}{PheromoneNodesInView} * WillingnessToExplore$$

Figure 9: Equation for calculating probability to explore

Every ant is limited to how many steps they can take before they need to return to a start node. An ant is only allowed to take steps that are 0 or 1 unit from its current position in both the X and Y axis. Diagonal movement is possible if the agent decides it should move 1 unit in the X axis and 1 unit in the Y axis. Without this restriction an ant could explore indefinitely. This also means that every goal node should be in reach of an agent's step limit or else it will never be reached. A goal node can also become a start node if the network asks for it. This way it is possible to create even more elaborate networks with different starting points.

5. Results

To test how well the networks created by the Multi-Agent biological transport network algoritm perform, we test the network performance on three metrics, see section 3.3.

5.1. Test setup

We test several simulations ranging in complexity to see how the ant agents perform in solving the network problems we gave to them. The parameters for the agents can be found in Table 1. The parameters for the agents stayed the same for all two different test cases. For all the cases, the goal is to connect all the starting vertices so that all nodes are visited at least once. All the measurements done are in a Unity unit. So a distance of 1 is 1 Unity unit. This can be translated to everything from 1 centimetre to 10000 feat as long as the conversion stays consistant.

Parameter	**Value**
Amount of ants	200
Starting position	Base (Middle of the network)
Field of view	90 degrees
Steps to explore	3
Willingness to explore	1
Step length	0 or 1 for both axis
Max pheromones ant	10
Starting resources	1000
Pheromone node limit	10
Evaporation rate pheromones	1 per step

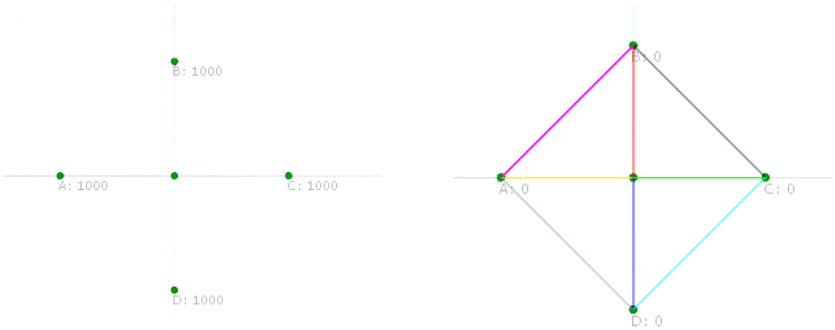
Table 1: Parameters used for the test simulations

5.1.1. Test setup 1

Test setup 1 is based on a simple star network, which can be seen in figure 10. This test is to prove the MAS can build a simple network between all the required vertices.

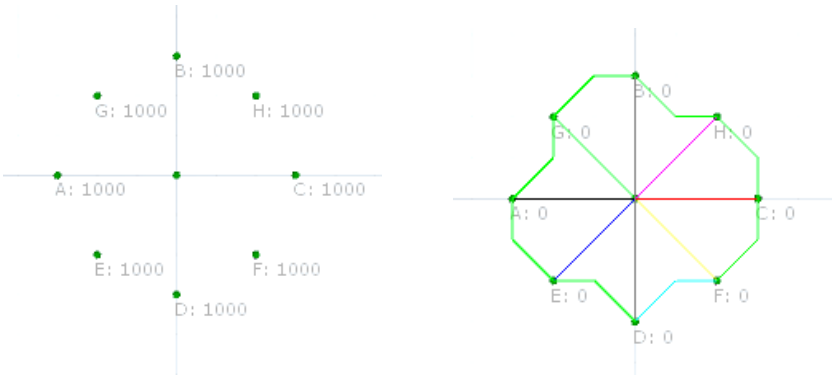
5.1.2. Test setup 2

This setup test how well the MAS performs on a more complex network that contains more nodes. This also tests how well the agents can adept



(a) Start setup test 1 (b) Result test 1

Figure 10: Test 1



(a) Start setup test 1 (b) Result test 1

Figure 11: Test 2

to nodes that are not placed ideally for the step size. Because of the node placement the agents are not able to walk across all the nodes in a straight line.

5.2. Fault tolerance

To test the fault tolerance of the network, we look at all the paths created by the system between two nodes. We take the shortest path between the two points and look if the two points in the network are still connected by one of the other paths that are made by the agents. We start cutting the path of nodes from the start after which we take the second node of the shortest

path and so on. We do this to see if it matters if nodes further from a vertex being removed matters.

Path	**Reachable**	**Extra cost**	**Average paths lost**
Origin to point A	Yes	0.83	3
Origin to point B	Yes	0.83	3
Origin to point C	Yes	0.83	3
Origin to point D	Yes	0.83	2.5
Point A to point B	No	-	1
Point B to point C	No	-	1
Point C to point D	No	-	1
Point D to point A	No	-	1

Table 2: Network 1: Fault tolerance with cuttof of 1 node in shortest path

Path	**Reachable**	**Extra cost**	**Average paths lost**
Origin to point A	Yes	0.83	3
Origin to point B	Yes	0.83	3
Origin to point C	Yes	0.83	3
Origin to point D	Yes	0.83	3
Origin to point E	Yes	0	1
Origin to point F	Yes	0	2.5
Origin to point G	Yes	0.83	2.5
Origin to point H	Yes	0	1
Point A to point G	Yes	0	1
Point G to point B	No	-	1
Point B to point H	No	-	1
Point H to point C	Yes	1.83	1
Point C to point F	Yes	0	1
Point F to point D	No	-	1
Point D to point E	No	-	1
Point E to point A	Yes	0	1

Table 3: Network 2: Fault tolerance with cuttof of 1 node in shortest path

5.3. Efficiency

To test the efficiency of the network we compare the shortest distance between two Vertices of the network with the shortest edge created by our

MAS. We also look at the average length per edge produced by the MAS.

Path	**Shortest distance**	**Shortest distance MAS**	**Average distance**
Origin to point A	3	3	3.71
Origin to point B	3	3	3.71
Origin to point C	3	3	3.71
Origin to point D	3	3	3.66
Point A to point B	3	3	3.00
Point B to point C	3	3	3.00
Point C to point D	3	3	3.00
Point D to point A	3	3	3.00

Table 4: Efficiency of the network 1

Path	**Shortest distance**	**Shortest distance MAS**	**Average distance**
Origin to point A	3	3	3.69
Origin to point B	3	3	3.71
Origin to point C	3	3	3.71
Origin to point D	3	3	3.71
Origin to point E	2.83	2.83	3.22
Origin to point F	2.83	2.83	3.22
Origin to point G	2.83	2.83	3.27
Origin to point H	2.83	2.83	3.22
Point A to point G	2.23	2.41	3.21
Point G to point B	2.23	2.41	3.26
Point B to point H	2.23	2.41	3.19
Point H to point C	2.23	2.41	3.22
Point C to point F	2.23	2.41	2.89
Point F to point D	2.23	2.41	3.07
Point D to point E	2.23	2.41	3.12
Point E to point A	2.23	2.41	3.36

Table 5: Efficiency of the network 2

5.4. Minimum spanning tree

The last metric we look at for the networks created by the multi-agent system is the minimum spanning tree (MST). The costs we look at here is

the length of the edges to cross the whole network. We want to compare the actual distance of visiting all the Vertices of the network starting from the starting point of the agents. We compare the MST created by the MAS with the MST of the most optimal network that can be created with the Vertices. We also look at the average MST each test produced.

Network	**Optimal MST**	**MAS MST**	**Average MST (MAS)**
Network 1	15	15	15.66
Network 2	20.67	22.11	28.54

Table 6: Minimum spanning tree of the networks created by the simulations

6. Conclusion

The main research question: *"How can we create a framework to simulate biological transport network algorithms?"* can be answered as follows. This research has been an explorative study to understand how a framework can be created to simulate and validate transport network algorithms. The framework is, in its current state, a first iteration towards a tool that could be used to simulate and compare transport network algorithms. In the current state only one transport network algorithm is implemented. Therefore, we cannot conclude that the framework is able to support various kinds of transport network algorithms. Although, the algorithm that is implemented can be compared against it self. The algorithm consist of parameters that could be tuned for specific transport network problems. Using the performance metrics these parameters can be compared. Furthermore, the framework is the first one of its kind. Therefore, it cannot be compared to pre-existing works. We cannot confirm that the performance metrics give a effective all-purpose statistics for all kinds of transport network problems.

Sub research question: *How can we simulate ant colony optimisation algorithms?* can also be answered. To simulate ants a lot of concessions had to be made to make the ants viable for the first iteration of the framework. In real life ants have a very sophisticated way to find the required needs to survive. In this iteration of the framework only a small part of that behaviour is simulated.

The simulations run with this limited implementation of an ants behaviour is more than promising. The agents were more than capable to

solve different kind of network problems thrown at the system. The minimum spanning tree the MAS achieves is really close to the most optimal spanning tree which means that the overall results of the created networks is more than acceptable for a biological shortest distance MAS.

If we look at the efficiency of the network the difference between the most optimal MST and the MST the system achieved can be explained (table 4 and 5). If all the points are reachable in a straight line, the shortest path between two points is always found. The points of the network the agents cannot reach in a straight line are the bottleneck. That is a problem the simulation itself can already solve by adjusting a variable (length of step size).

The networks created are also very robust. The extra distance that needs to be covered is minimal for the paths that get visited often. There seems to be a problem with vertices that are visited last by the agents. Also vertices that are not on the same X-axis or Y-axis position as the agents start from seem to perform less on the fault tolerance metric. These networks seem to contain fewer unique paths between those vertices. This results in a less robust path between those points. Extra vertices between two points help, as can be seen in the results of table 3, but a better solution for this problem is still needed.

All by all, the system created for a first iteration of a potential biological transport network toolkit seems more than capable to handle network problems. The features included for the ACOA already seems to be efficient and accurate enough to be featured in its own toolkit. Future research will tell if this implementation is also good when it is compared with real life transport network problems. The metrics the system contains at the moment are still lacking and should be expanded upon.

7. Discussion

7.1. Ant transport network algorithm

As mentioned before, the ant colony optimisation algorithm used in the toolkit is limited and lack many features real ants do have. The most important feature missing right now is a real decentralised solution for the framework. The graph which is being used for finding duplicate nodes is the only centralised part of the current MAS. The MAS could be totally decentralised, by letting the ants communicate to the vertices in the network which notes are there. The reason for not implementing this in the current

version of the framework is that it would add to much uncertainty and time to the project, which was undesirable in the given time frame of the project. A lot of time could have been lost by testing if the decentralising is not missing nodes, or generates undesired behaviours. Implementing this behaviour would also make it so that the system can work on any problem as now the performance will drop significantly when a really big network is created.

Next to that, real life ants also drop pheromones every step they take, instead of as soon as they find a resource. This was done to speed up the simulation, but it also influences the paths ants take. To create a simulation that is more conform the behaviour of real life ants, it would be preferred to change the current pheromone logic. To make that option viable performance wise, the system needs to be decentralised.

7.2. Framework

For a second iteration of the framework, other biological networks could be implemented in order to validate whether the framework is able to compare network performance between algorithms. For example, slime molds would be a research subject that can be elaborated on. Subsequently, the framework could be compared with real world situations to validate its purpose. Ideally, imported real world transport network problems could be mapped according to the performance metrics. In this way, transport network algorithms could be tuned to fit these problems.

8. References

[1] M. Wooldridge, An introduction to multiagent systems, 2nd edition, 2009.

[2] M. Dorigo, L. M. Gambardella, Ant colonies for the travelling salesman problem, biosystems 43 (1997) 73–81.

[3] A. Chandrasekhar, D. M. Gordon, S. Navlakha, A distributed algorithm to maintain and repair the trail networks of arboreal ants, Scientific reports 8 (2018) 9297.

[4] G. Huang, Y. Cai, H. Cai, Multi-agent ant colony optimization for vehicle routing problem with soft time windows and road condition, in: MATEC Web of Conferences, volume 173, EDP Sciences, p. 02020.

[5] J. E. Bell, P. R. McMullen, Ant colony optimization techniques for the vehicle routing problem, Advanced engineering informatics 18 (2004) 41–48.

[6] A. Kazharov, V. Kureichik, Ant colony optimization algorithms for solving transportation problems, Journal of Computer and Systems Sciences International 49 (2010) 30–43.

[7] R. Musa, J.-P. Arnaout, H. Jung, Ant colony optimization algorithm to solve for the transportation problem of cross-docking network, Computers & Industrial Engineering 59 (2010) 85–92.

[8] C. Blum, Ant colony optimization: Introduction and recent trends, Physics of Life reviews 2 (2005) 353–373.

[9] G. Lu, Z. Liu, Multicast routing based on ant-algorithm with delay and delay variation constraints, in: Circuits and Systems, 2000. IEEE APCCAS 2000. The 2000 IEEE Asia-Pacific Conference on, IEEE, pp. 243–246.

[10] Y.-T. Hsiao, C.-L. Chuang, C.-C. Chien, Ant colony optimization for best path planning, in: Communications and Information Technology, 2004. ISCIT 2004. IEEE International Symposium on, volume 1, IEEE, pp. 109–113.

[11] M. Dorigo, T. Stützle, Ant colony optimization: overview and recent advances, Techreport, IRIDIA, Universite Libre de Bruxelles 8 (2009).

[12] H. V. D. Parunak, " go to the ant": Engineering principles from natural multi-agent systems, Annals of Operations Research 75 (1997) 69–101.

[13] S. Tisue, U. Wilensky, Netlogo: Design and implementation of a multi-agent modeling environment, in: Proceedings of agent, volume 2004, pp. 7–9.

[14] W. Xiang, H. P. Lee, Ant colony intelligence in multi-agent dynamic manufacturing scheduling, Engineering Applications of Artificial Intelligence 21 (2008) 73–85.

[15] A. Colorni, M. Dorigo, V. Maniezzo, et al., Distributed optimization by ant colonies, in: Proceedings of the first European conference on artificial life, volume 142, Paris, France, pp. 134–142.

[16] K. Socha, M. Dorigo, Ant colony optimization for continuous domains, European journal of operational research 185 (2008) 1155–1173.

[17] F. Von Martial, Interactions among autonomous planning agents, Decentralized AI (1990) 105–119.

[18] X. Cerdá, J. Retana, A. Manzaneda, The role of competition by dominants and temperature in the foraging of subordinate species in mediterranean ant communities, Oecologia 117 (1998) 404–412.

[19] J. Schyns, ant behaviour observations, ????

[20] A. Tero, S. Takagi, T. Saigusa, K. Ito, D. P. Bebber, M. D. Fricker, K. Yumiki, R. Kobayashi, T. Nakagaki, Rules for biologically inspired adaptive network design, Science 327 (2010) 439–442.

[21] J. Buhl, J. Gautrais, R. V. Solé, P. Kuntz, S. Valverde, J.-L. Deneubourg, G. Theraulaz, Efficiency and robustness in ant networks of galleries, The European Physical Journal B-Condensed Matter and Complex Systems 42 (2004) 123–129.

[22] A. Adamatzky, R. Alonso-Sanz, Rebuilding iberian motorways with slime mould, Biosystems 105 (2011) 89–100.

[23] T. Nakagaki, R. Kobayashi, Y. Nishiura, T. Ueda, Obtaining multiple separate food sources: behavioural intelligence in the physarum plasmodium, Proceedings of the Royal Society of London B: Biological Sciences 271 (2004) 2305–2310.

[24] T. Nakagaki, H. Yamada, M. Hara, Smart network solutions in an amoeboid organism, Biophysical chemistry 107 (2004) 1–5.

[25] A. Tero, R. Kobayashi, T. Nakagaki, Physarum solver: A biologically inspired method of road-network navigation, Physica A: Statistical Mechanics and its Applications 363 (2006) 115–119.

[26] D. P. Bebber, J. Hynes, P. R. Darrah, L. Boddy, M. D. Fricker, Biological solutions to transport network design, Proceedings of the Royal Society of London B: Biological Sciences 274 (2007) 2307–2315.

[27] W. Gawlitta, K. V. Wolf, H.-U. Hoffmann, W. Stockem, Studies on microplasmodia of physarum polycephalum, Cell and tissue research 209 (1980) 71–86.

[28] S. Tsuda, K.-P. Zauner, Y.-P. Gunji, Robot control with biological cells, Biosystems 87 (2007) 215–223.

[29] A. Adamatzky, Steering plasmodium with light: Dynamical programming of physarum machine, arXiv preprint arXiv:0908.0850 (2009).

[30] T. Nakagaki, H. Yamada, Á. Tóth, Intelligence: Maze-solving by an amoeboid organism, Nature 407 (2000) 470.

[31] Anylogic: Simulation modeling software tools solutions for business, ????

[32] The repast suite, ????

[33] Unity, ????