Article:
Parrott, C., Dodd, T.J., Boxall, J. et al. (2020) Simulation of the behavior of biologically-inspired swarm robots for the autonomous inspection of buried pipes. Tunnelling and Underground Space Technology, 101. 103356. ISSN: 0886-7798

https://doi.org/10.1016/j.tust.2020.103356

Simulation of the Behavior of Biologically-Inspired Swarm Robots for the Autonomous Inspection of Buried Pipes

Christopher Parrott[a,*], Tony J. Dodd[b,**], Joby Boxall[c], Kirill Horoshenkov[a]

[a]*Department of Mechanical Engineering, The University of Sheffield, UK*
[b]*Department of Automatic Control & Systems Engineering, The University of Sheffield, UK*
[c]*Department of Civil & Structural Engineering, The University of Sheffield, UK*

Abstract

The use of robots for the inspection of buried pipelines has gained popularity over the past decade. In this paper we move the vision forward by examining what behavior and attributes would be required for these robots to become autonomous and pervasive within buried water pipe infrastructure. We present the results from novel simulations to evidence the inspection capability of autonomous robots, investigating operation, cooperation and communication attributes. The simulation uses a biologically-inspired behavior that provides complete and consistent coverage of real life example clean water distribution management areas. We show that autonomous robots could operate without a centralized controller and benefit from having some degree of in-pipe communication. We evidence the ability to adapt to changes in communication, speed, and flow conditions. The mathematical model that we derive through the simulation is scalable with the change of network length, topology, robots' speed and number. This work paves the way and sets the specifications for practical development of autonomous pervasive robots for the inspection of complex pipe networks.

Keywords: Swarms, Robotics, Autonomy, Inspection, Stigmergy, Infrastructure

1. Introduction

The unknown condition of buried assets is a big problem for water utilities that manage large networks of pipes. A majority of faults in these systems are only

*Corresponding author
**Current address is: School of Creative Arts and Engineering, Staffordshire University, UK
Email addresses: c.parrott@sheffield.ac.uk (Christopher Parrott), tony.dodd@staffs.ac.uk (Tony J. Dodd), j.b.boxall@sheffield.ac.uk (Joby Boxall), k.horoshenkov@sheffield.ac.uk (Kirill Horoshenkov)

being discovered after they have failed, resulting in leakages, costly repair works, and traffic and supply disruption. Environmental impact is also a concern, as fluids can leak from pipes causing erosion and pollution. Ground water can infiltrate pipes contaminating clean water or adding a significant load on the wastewater treatment plant. Being able to map the position, and assess the condition and performance of these assets will allow for proactive repair and rehabilitation at the early onset of critical change in the pipe in a manner that minimizes disruption and is more cost efficient. Performing such assessment can be achieved through the pervasive use of mapping and inspection technologies. Although a variety of pipe inspection technologies, including robotic, are already deployed by water and other infrastructure companies Liu and Kleiner (2013), there is yet to be a solution that provides the complete and consistent coverage required to see widespread adoption of autonomous robots and achieve the key benefits of human-free operation.

The early days of buried asset inspection were heavily reliant on visual inspection. A human would enter a pipe of a large enough diameter to assess its condition. This method required the pipe to either be emptied or, in the case of sewers, have a low enough flow of water for the person to wade through. This method presented a serious risk to the person assessing the pipes that led to the development and use of Closed Circuit Television (CCTV) devices for remote visual assessment. The majority of commercial inspection solutions for sewer pipes now rely heavily on this particular method Calderon and et al (2014). More novel solutions are to use man-hole zoom cameras for the rapid inspection of sewer networks Laggis (2016). Zoom cameras are optical devices with a high resolution video camera with a telescopic lens and powerful lights attached to the end of a pole that are inserted into man-hole entry points to capture images upstream and downstream from a fixed point. Images are taken at each man-hole in an inspection area and sent to an operator for subsequent analysis. Unfortunately, such images only show a limited range in the pipe meaning that if the pipe network does not have man-holes in close proximity then faults may not be detected. Additionally, human assessment of the images is subjective and prone to misclassification Dirksen et al. (2013). To overcome the range limitation, technologies that employ acoustics, such as SewerBatt, have been developed Horoshenkov et al. (2010). This acoustic technology sends out a pulse of sound from a man-hole location and listens to the reflections to identify anomalies and their locations in the pipe. Unfortunately, some fault types are not detectable by acoustics, leading some to suggest that both acoustic and visual inspection should be used in conjunction Plihal et al. (2016) for this kind of fixed-location method.

Unlike clean water and sewer pipes, gas and oil pipelines do not have regularly spaced entry points. In these pipes remote visual inspection is performed with teleoperated robotic devices known as crawlers Ogai and Bhattacharya (2018).

Crawlers feature a camera and powerful lights attached to a mobile platform that is able to drive down an empty pipe and record video for subsequent analysis. These platforms are typically designed to travel down straight pipes without any branches or obstacles (such as valves), making them unsuitable for large-scale inspection of more complex pipe networks. The need to receive video from a crawler continuously has lead to the majority of crawlers being tethered and externally powered. For these reasons an alternative technique of Pipe Inspection Gauges (PIGs) has seen wide-spread use in gas and oil networks Tiratsoo (1992). These devices are inserted via a launch tube into pipes that carry flow product and are propelled along by the flow until they reach the end where they are collected from a receiver tube. These are self-powered and carry a variety of sensors for assessing the internal condition of pipes. This approach reduces human involvement in the inspection process. Unfortunately, pipes contain features such as valves, intersections, or diameter changes that PIG designs are not equipped to deal with. Additionally, they are expensive and require complex enabling works. This limitation has lead to a number of alternative inspection platforms being proposed Mills et al. (2017). Efforts have been made to introduce PIG-like commercially available devices into water and wastewater pipes, e.g. Pipediver Pure Technologies (2017) and Smartball Pure Technologies (2014). These devices can traverse obstacles such as valves and cross-sectional changes because their size is notably smaller than the diameter of the pipe they are travelling through. However, these devices are passive and designed to follow the flow. This means that they can collect inspection data only along the path that the flow of water would take, leaving gaps in the system's condition knowledge. Additionally, retrieval can often be a substantial, and even prohibitive practical barrier.

Alternatives to in-pipe inspection are also being explored, with ground penetrating radar Liu and Kleiner (2013) being one such technique. Ground penetrating radar (GPR) is a non-invasive inspection technique that uses radar pulses directed into the ground to detect the presence and structure of buried infrastructure. A downside of GPR is that being external to the pipe, any faults that originate from within the asset may not be detected early enough to be proactively responded to. Additionally, it has a low resolution in comparison to the size such faults are likely to form in the soil surrounding the pipe, and it only works in above ground areas that have no buildings or other infrastructure impeding access. Another alternative is that of SmartPipes Sadeghioon et al. (2014), whereby the pipes of a network are themselves capable of assessing their condition and performance. Such technology would augment new and existing pipes with sensors and processing that enables them to determine their own structural and operational conditions, and communicate information externally. Depending on what pipe condition information is required, retrofitting existing pipes can be achieved in a non-invasive (to the pipe) manner Liu and Kleiner (2013); Nuron (2019). Although this tech-

nology has potential for the future, the slow upgrade cycle of water infrastructure assets means that it could take well over a century for the majority of pipes to be upgraded to smart pipes or retrofitted through costly excavation works, massive traffic and service disruptions and pipe renovation costs.

None of the existing solutions for buried water pipeline infrastructure are truly autonomous or pervasive. Our review of pipe inspection technologies suggests that a radical solution is required to address proactively and cost-efficiently the problem of ageing pipe infrastructure. In this paper we propose that pervasive autonomous robots can be this solution. These robots will transform the way utilities collect data on the condition and performance of their buried pipes in real-time, over the entire network using minimal human interaction. This will enable utilities to act timely when the disruption to the service and traffic is minimal and the cost of intervention is relatively low. Currently such autonomous robots do not exist, raising the research question of whether they are actually viable for inspecting buried assets. This paper explores this question from a robot intelligence perspective (as opposed to mechanical, sensing or any other practical aspects of robot design and implementation), and offers a positive answer by examining a biologically-inspired behavior of a swarm of autonomous robots that can provide complete and consistent inspection coverage of a clean water pipe network of a realistic size. The choice of this type of network is not accidental. Data suggests that in England and Wales alone 3.170Bn liters of water are lost every day through pipe leaks Water UK (2019). Detection of leaks with swarms of robots will enable water utilities to locate the position of these leaks and to guide repair equipment with unprecedented accuracy, minimising the associated road disruption and repair costs. In addition, the complete and consistent coverage would enable the monitoring of pipe cross-sectional changes or surface corrosion levels that, if left unchecked, may result in leaks in the future.

Through computer simulations we show that robots with a biologically-inspired behavior are able to operate without a centralized controller and can adapt to changes in communication, speed, and flow conditions. In this paper we assume that these virtual robots are small and intelligent enough to avoid collisions with each other or being stuck in some narrow parts of the pipe network. From the results obtained, a basic mathematical model is derived that is scalable with the change of network length, topology, robots' speed and number. We note that key principles behind the simulated swarm robot behavior under realistic flow conditions and variable communication capabilities are generic to other types of pipe networks such as sewerage, gas and oil.

The rest of this paper is organised as follows. Section 2 discusses the intelligence challenges faced by autonomous robots performing inspection, and presents the concept of the biologically-inspired robot behavior. Section 3 details the simulation environment created and the code implementation of the proposed robot

behavior. Section 4 details and analyses results obtained from the simulator. Finally, Section 5 concludes the paper and summarises the work.

2. Autonomous Inspection

Internal inspection of a single pipe with a single or multiple robots controlled by an operator is a relatively straightforward process. However, inspection of a realistic network of buried pipes with autonomous robots is a much more complicated process that requires robots to have intelligence to navigate through the network, and to co-operate and communicate to produce efficient paths. Formulating a path to follow, known as planning, is a process that has seen widespread use in society over the past decade, most notably to direct drivers as part of satellite navigation systems for vehicles. These systems use knowledge of road networks to calculate the shortest path for drivers to follow to reach their destinations Laporte (1992); Eksioglu et al. (2009), often considering other factors such as traffic congestion Ahn and Shin (1991). The way these systems achieve this is by representing the road network as a set of connected vertices and edges, known as a graph, and applying algorithms to intelligently search through them. The most well known algorithms for path planning are Dijkstra's Dijkstra (1959) and A-star Hart et al. (1968), which are both able to find the shortest path between any two graph vertices if given sufficient computation time. What makes A-star different from Dijkstra's is that it performs this search more intelligently by including a heuristic to estimate how much further is left to be travelled, often allowing it to arrive at a solution much faster. The inclusion of a heuristic in algorithms like A-star makes them informed searches.

For the autonomous inspection of distribution assets to be performed regularly, an optimal path needs to be computed that allows robots to repeatedly travel along every pipe in the network collecting usable data on the pipe conditions and performance. This is known in graph theory as the Chinese Postman Problem Thimbleby (2003), which seeks to find the shortest route or cycle that a postman can follow to deliver mail to every resident along a set of streets, with the ideal case being an Eulerian cycle. This differs from the well known Travelling Salesman Problem Jünger et al. (1995) as the destinations are not the focus, but rather the streets connecting them. A cycle is considered Eulerian if from a starting point it is possible for a robot to travel along every graph edge (e.g. pipe in our case) exactly once to get back to the starting point Christofides (1973). A less ideal case is that of a semi-Eulerian cycle, which travels along each graph edge once but has different starting and ending points.

Finding an optimal network cycle presents a few challenges for in-pipe inspection with multiple robots. It can be computationally expensive to generate for networks of realisitic sizes, either placing high processing demands on individual

robots or requiring the computation to be offloaded to an external server and the results communicated back. Any change in the network, either as a result of maintenance or changing environmental conditions such as flow rates, requires a new cycle to be computed. Additionally, it is difficult for such techniques for computing cycles to scale to work with multiple robots inspecting a given network at the same time, although some efforts have been made to address this Osterhues and Mariak (2005).

An alternative approach to navigating a network is that of swarm behavior. Instead of computing a cycle for each robot to follow, the robots are treated as biological species exploring the network by reacting to the environment and each other. Many examples of this exist in nature ranging from flocks of birds and schools of fish, to insect colonies Camazine et al. (2001). In all these examples, a complex behavior emerges from local interactions. It enables a swarm to avoid predators, search for food, or to build intricate structures. One particular technique used in nature is that of stigmergy Dorigo et al. (2000), which is also referred to as indirect coordination. In this process individual creatures place chemical signals (pheromones) in the environment that themselves and others can sense and act upon. An example of this is with members of an ant colony that are collecting food. Each ant explores the environment, leaving a chemical trail behind it that acts as a path back to the nest. When an ant finds food, it retraces its path, strengthening the chemical trail in this process, encouraging other ants in proximity to adopt the same path and further strengthen the trail. Any chemical trails that are not regularly travelled along dissipate over time and stop being followed. Eventually, the swarm converges on to a small number of paths from nest to food until that source is exhausted and the process repeats itself once again.

Inspired by ant colonies, a stigmergy-based behavior for in-pipe inspection robots was explored in this paper. Rather than placing chemical signals in the environment for other robots to detect, stigmergy was performed virtually via the use of internal memory on board each robot and inter-robot communication Pinciroli et al. (2016). Figure 1 shows this concept applied to a single robot exploring a cross-shaped network of pipes. Every time the robot entered a pipe, it flagged that a new inspection had started by depositing a strong virtual pheromone in that pipe. Pipes were flagged at the start of an inspection rather than the end to avoid the later situation with multiple robots of one not being aware of the decision of another until after the other had already travelled the pipe length. At the central crossroad the robot decided on the next pipe to go down based on its knowledge of these virtual pheromones. Unlike in the case with the ants, which are drawn to stronger signals to create and to follow a path, in this approach the robot selected the pipe with the weakest signal to form a cycle to follow. When the robot reached a dead-end it turned around, redeposited a strong virtual pheromone, and travelled back along the pipe again. As with ant chemical pheromones, the strength of the

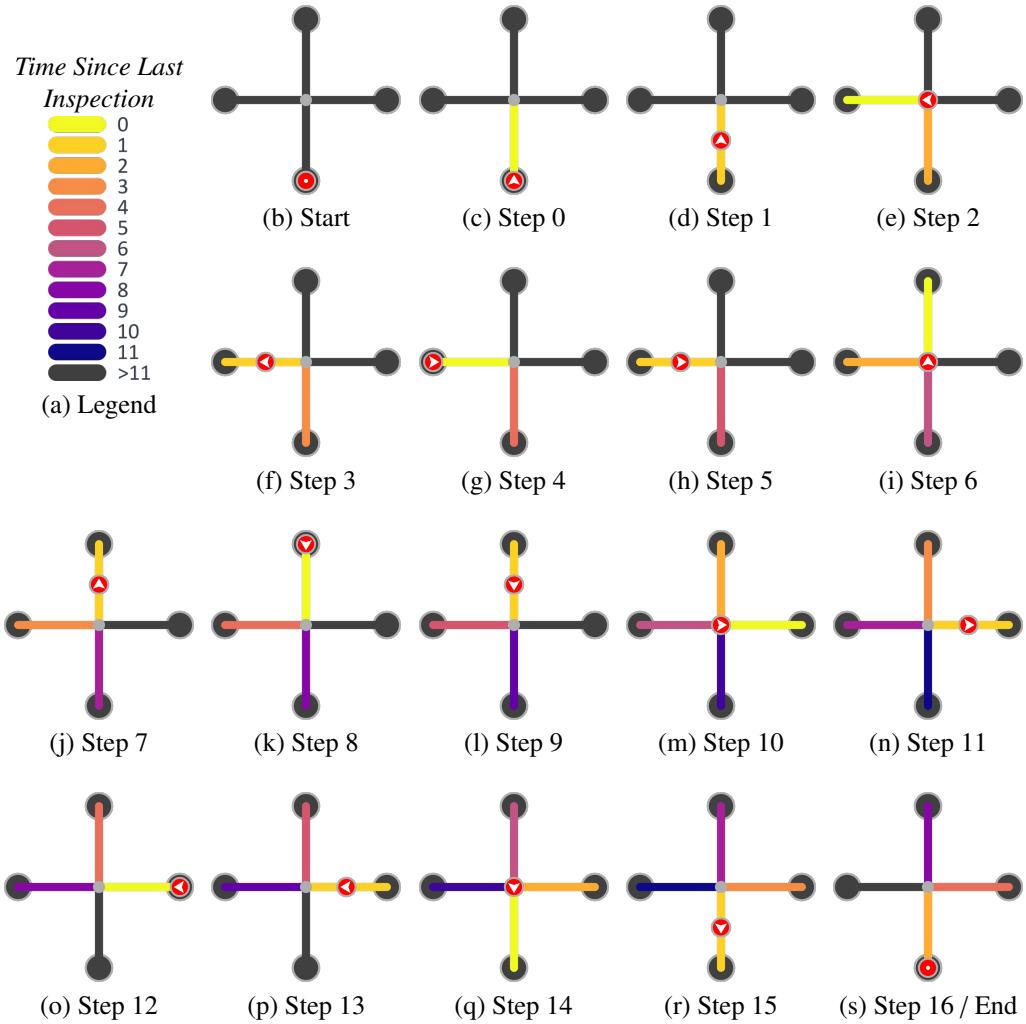
virtual pheromone would reduce over time. In our work it was reduced linearly with each step the robot would take, making the pheromone strength follow an inverse dependence on the *time since last inspection* (TSLI) started of each pipe.

As can be seen in Figure 1, it takes the robot 16 time-steps to inspect all the pipes in the cross-shaped network and get back to its start location. When adding a second robot to this scenario that is delayed by one time-step (as shown in Figure 2), the two robots return to their respective starting locations in 8 time-steps, having equally distributed the network inspection task between the two of them. This can be thought of as having performed the same number of inspections in half the time, or double the number of inspections in the same amount of time.

3. Simulation Environment

To evaluate the proposed autonomous robot behavior on more realistic networks of pipes, a simulation environment was developed to observe how various numbers of such robots perform under different conditions. The simulator was built in C++ and used EPANET, the US EPA software that models drinking water distribution systems US EPA (2018), to read in data from .inp files. These are files that contain network position and connectivity information as well as the water demands experienced (from which pipe flow velocities are computed) over a 24-hour period, at a given interval (e.g. 1 hour). Rendering was implemented using built-in Windows Application Programming Interface (API) functions. The simulator stored a network structure as an undirected graph consisting of junctions (vertices) and links (edges). Each junction of the graph had a name, 2D position and a list of the pipes it joined with. The junctions also acted as points of demand of water within the network. Each link in this network had a pipe ID, length, diameter, roughness, calculated flow velocity data, and references to the two junctions that it joined together. Note that there is not necessarily a correlation between a pipe's ID and its location or connectivity in a network. Robots were implemented as points that travel along the lines that links represent.

With the described network representation, simulations of the robot behavior were initialized by randomly inserting robotic agents into the network at junctions marked as communication nodes. Communication nodes were assumed to be those junctions that only had a single link connecting them to the rest of the network, as these are likely candidates for external access and communication with the outside world. Once initialised, agents proceeded to travel at a given velocity down links based on their knowledge of each pipe's TSLI, each time selecting the one that had gone the longest without an inspection. Agents were aware of the current link they were within, based on its pipe ID, and used this to query their internal memory for the TSLI values. It should be noted that the agents did not know their exact coordinates within the network, nor needed high certainty prior



(a) Legend (b) Start (c) Step 0 (d) Step 1 (e) Step 2

(f) Step 3 (g) Step 4 (h) Step 5 (i) Step 6

(j) Step 7 (k) Step 8 (l) Step 9 (m) Step 10 (n) Step 11

(o) Step 12 (p) Step 13 (q) Step 14 (r) Step 15 (s) Step 16 / End

Figure 1: A single robot exploring a cross-shaped network using virtual stigmergy linked to the *time since last inspection* started. The robot is depicted as a red circle, with the white arrow indicating its direction of travel. Large gray circles are end points and the small light-gray circle is the junction point. The robot takes 2 time-steps to travel down a pipe. When multiple pipes have been without an inspection for a similar time, the robot randomly chooses the next one to inspect.

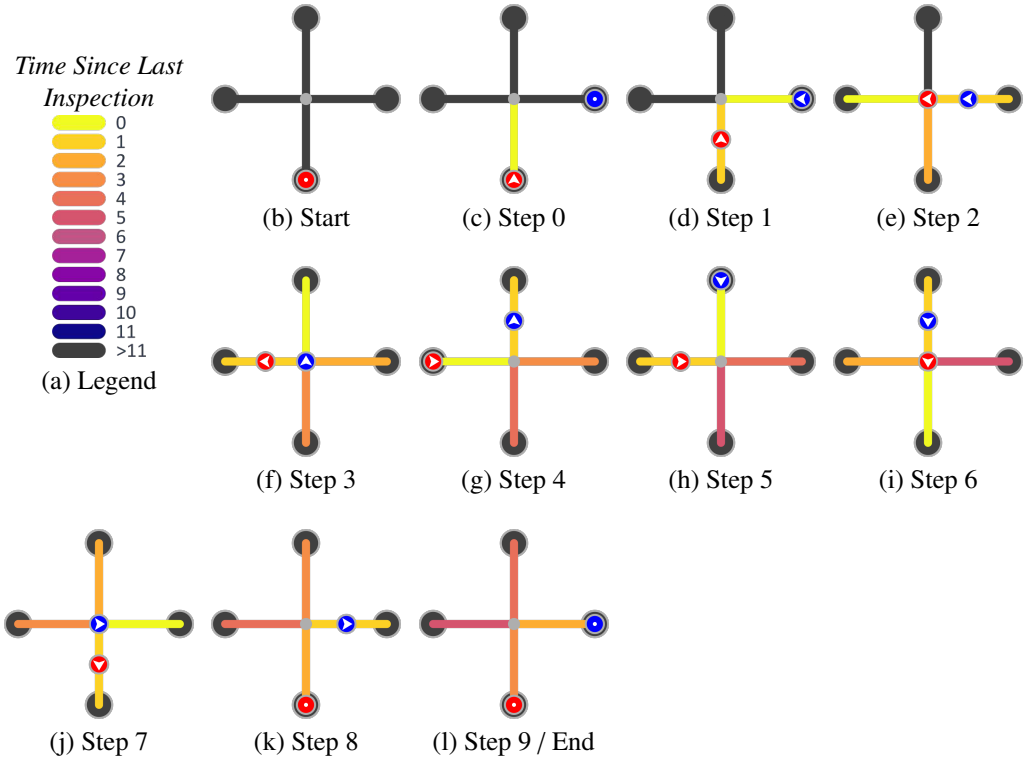


Figure 2: Two robots exploring a cross-shaped network using virtual stigmergy linked to the *time since last inspection* started, with the second robot being inserted 1 time-step after the first. The robots are depicted by red and blue circles, with the white arrows indicating their directions of travel. Large gray circles are end points and the small light-gray circle is a junction point. The robots take 2 time-steps to travel down a pipe. If a robot encounters multiple pipes that have been without an inspection for a similar time, it randomly chooses the next one to inspect.

knowledge of the network layout. Instead, they only knew the current junction they were at or the current link they were within and their distance along it. An illustrative example of the simulator running on a real network with eight agents is shown in Figure 3. The window is divided into two main views:

- *Network View* - Shows the structure of the network with agents overlaid on top, and communication node junctions marked by gray circles (see Figure 3(a)). Links are depicted as lines with their color representing the TSLI up to a user specified duration, and their thickness being a function proportional to the pipe diameter. The Plasma colormap created for Matplotlib Van Der Walt and Smith (2015), which follows a yellow to purple transition, is used for coloring the links. Any links with the TSLI being beyond that specified by the user are shown as Dark Gray. For the simulation run shown, a duration of 84 hours (half a week) was chosen, as this allowed the

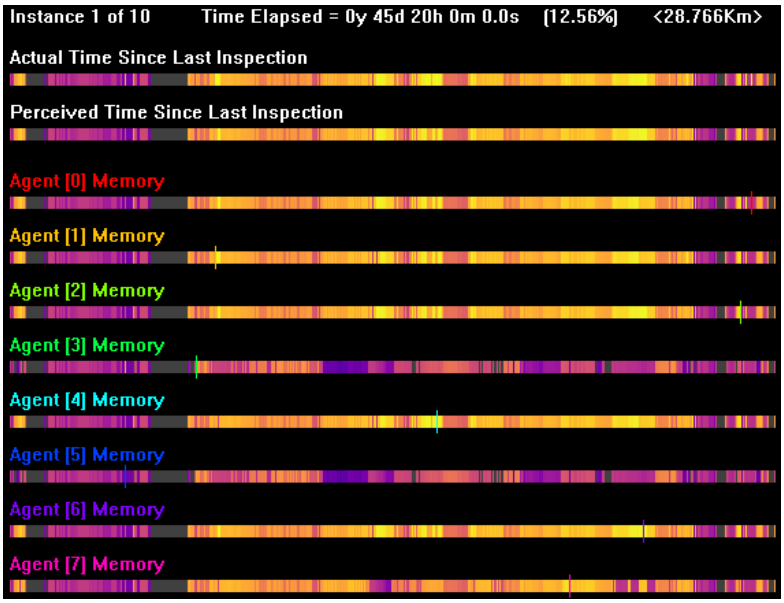
majority of pipes to be assigned a color whilst also being able to observe color differences between neighboring pipes.

- *Memory View* - Shows one-dimensional plots of the TSLI of each link in the network in ascending ID order from left to right (see Figure 3(b)). Both the actual TSLI values recorded by the simulation and the values each agent has stored in its internal memory are shown. In addition, an external perceived plot is shown, representing values that may have been communicated out by the agents to an external system via communication node junctions. Colors follow the same yellow to purple gradient described for the *Network View* window.

The proposed robot behavior was implemented by the simulation environment as follows. Each agent started at a junction and selected the link that it believed had gone the longest without inspection, discounting any for which the flow velocity at that moment was greater than what the agent was capable of travelling against (whenever applicable). If multiple links had gone the same longest time without an inspection, both were equally viable choices to travel down, so one was chosen randomly. This situation was most likely to have occurred at the beginning of a simulation run when no pipes have been inspected, meaning that all of them have gone the longest without an inspection. In order to avoid agents back-tracking unnecessarily, the link that they last travelled down was initially excluded from the link selection process, and only got considered if no other link was available to select. For example, if an agent reached a T-junction, there must have been flow coming from both of the two available links before the agent would consider travelling back down the pipe it arrived by. If no suitable link was identified through which the agent could travel, and back-tracking was not an option, then the agent entered a sleep state for a period of 30 minutes before repeating the selection process again. This duration was chosen as it corresponded to half of the flow data update interval of 1 hour at which the flow data was acquired originally. If a suitable link was found, then the agent set the TSLI of that link in its internal memory to zero, indicating an inspection had started, and would then proceed to travel along the link. At this moment the agent could optionally broadcast (depending on simulation run parameters) the TSLI values stored within its internal memory by a given range for other agents to receive and update their own data. Whether two agents were in broadcast range was determined by the length of the shortest network path between them. When a new TSLI value for a given pipe was received by another agent, it updated its memory with the value only if it was lower than what was already stored, otherwise it was ignored. This process ensured that the agents were always aware of when the latest inspections occurred.

Two conditions were supported by the simulator for agents travelling along links. The first was to move at their driving velocity regardless of the current

(a)



(b)

Figure 3: An example screenshot from the simulator performing a run on a real network with eight agents driving at 0.1 ms^{-1} under flow conditions. The image is divided in to two main views, a) *Network View* and b) *Memory View*.

flow velocity, and the second was to move at their driving velocity minus the current flow velocity. If the resulting velocity was greater than the driving velocity then the driving velocity was used, and if it was negative then an agent was stopped. Although the former approach was less realistic in a real-world scenario, it was considered less computationally expensive because checks for the current flow did not need to be performed at every simulation time-step. When an agent reached the end of its selected link, and it was at a new junction considered to be a communication node, an optional synchronisation routine could occur (depending on simulation run parameters), whereby the agent updated its memory through the comparison with the TSLI values stored in its memory and externally. At this point a new link selection process could begin. This information was the *Perceived Time Since Last Inspection* as illustrated in Figure 3(b). The specific implementation of this behavior is explained in Algorithm 1. Videos of the simulator in operation under a variety of conditions can be found in the supplementary material that accompanies this paper (Parrott et al. (2019)).

4. Behavior Analysis

To analyse the proposed robot behavior, simulations were conducted on four networks: two artificial without flow (see Figures 4(a-b)) and two real district metered areas (DMAs) with 24-hour flow data that exist in the UK (see Figures 4(c-d)). The first artificial network was a branching case where every junction except the ends presents at least two choices to the robot agents, and the second was a branching case created by randomly adding new junctions linked to the previous ones. All links in the artificial networks were 10 meters in length. For the real networks, the first was a branching network, and the second contained both branches and loops. Key parameters for all the networks are summarised in Table 1.

In this paper the effect of communication method and robot driving speed under the presence and absence of flow was studied. The mean and standard deviation of the *time between inspections* (TBI) that each link experienced was determined from these simulations. Unlike TSLI, which is a measure of how long a link has gone without an inspection relative to the current time in a simulation, TBI is the amount of time separating any two sequential inspection events on a link. If a link did not experience two sequential events during a simulation run, then the TBI for that link was taken as being simulation duration. All simulations were performed for the number of robot agents being between $1 \leq N \leq 32$. These agents were initially deployed at random communication node junctions within each network type for a simulation duration of 28 days (4 weeks) with a 0.1 s time-step. This duration was chosen as it allowed agents to perform multiple inspection passes at slow driving speeds to produce meaningful results. The

Algorithm 1 Pseudocode of the behavior executed by each agent within the simulation. Agents begin at junctions and travel at a specified driving speed. For networks without flow data, line 22 and 28 are omitted.

Require: $CurrentJunction \neq null$, $DrivingSpeed > 0$
1: $LastLink \leftarrow null$
2: **loop**
3: $NextLink \leftarrow$ SelectLink($CurrentJunction$, $DrivingSpeed$, $GetTime()$, $LastLink$)
4: **if** $NextLink \neq null$ **then**
5: $TimeSinceLastInspection$ of $NextLink \leftarrow 0$
6: (Optional) Broadcast all known $TimeSinceLastInspection$ values to nearby agents
7: Travel along $NextLink$ at $DrivingSpeed$
8: $CurrentJunction \leftarrow$ opposite junction of $NextLink$
9: $LastLink \leftarrow NextLink$
10: **if** Only 1 link is connected to $CurrentJunction$ **then**
11: (Optional) Synchronise $TimeSinceLastInspection$ values with Node
12: **end if**
13: **else**
14: Sleep for 30 minutes
15: **end if**
16: **end loop**
17:
18: **function** SelectLink($Junction$, $DrivingSpeed$, $CurrentTime$, $LastLink$)
19: $CandidateList \leftarrow empty$
20: $BackTrackingCandidate \leftarrow null$
21: **for all** $Links$ connected to $Junction$ **do**
22: **if** $DrivingSpeed > FlowVelocity$ of $Link$ at $CurrentTime$ **then**
23: **if** $Link \neq LastLink$ **then**
24: Add $Link$ to $CandidateList$
25: **else**
26: $BackTrackingCandidate \leftarrow Link$
27: **end if**
28: **end if**
29: **end for**
30: **if** $CandidateList \neq empty$ **then**
31: $SelectionList \leftarrow empty$
32: $MaxTimeSinceLastInspection \leftarrow 0$
33: **for all** $Links$ in $CandidateList$ **do**
34: **if** $TimeSinceLastInspection$ of $Link > MaxTimeSinceLastInspection$ **then**
35: $MaxTimeSinceLastInspection \leftarrow TimeSinceLastInspection$ of $Link$
36: $SelectionList \leftarrow empty$
37: **end if**
38: Add $Link$ to $SelectionList$
39: **end for**

```
        return A random link from SelectionList
    else if BackTrackingCandidate ≠ null then
        return BackTrackingCandidate
    end if
    return null
end function
```

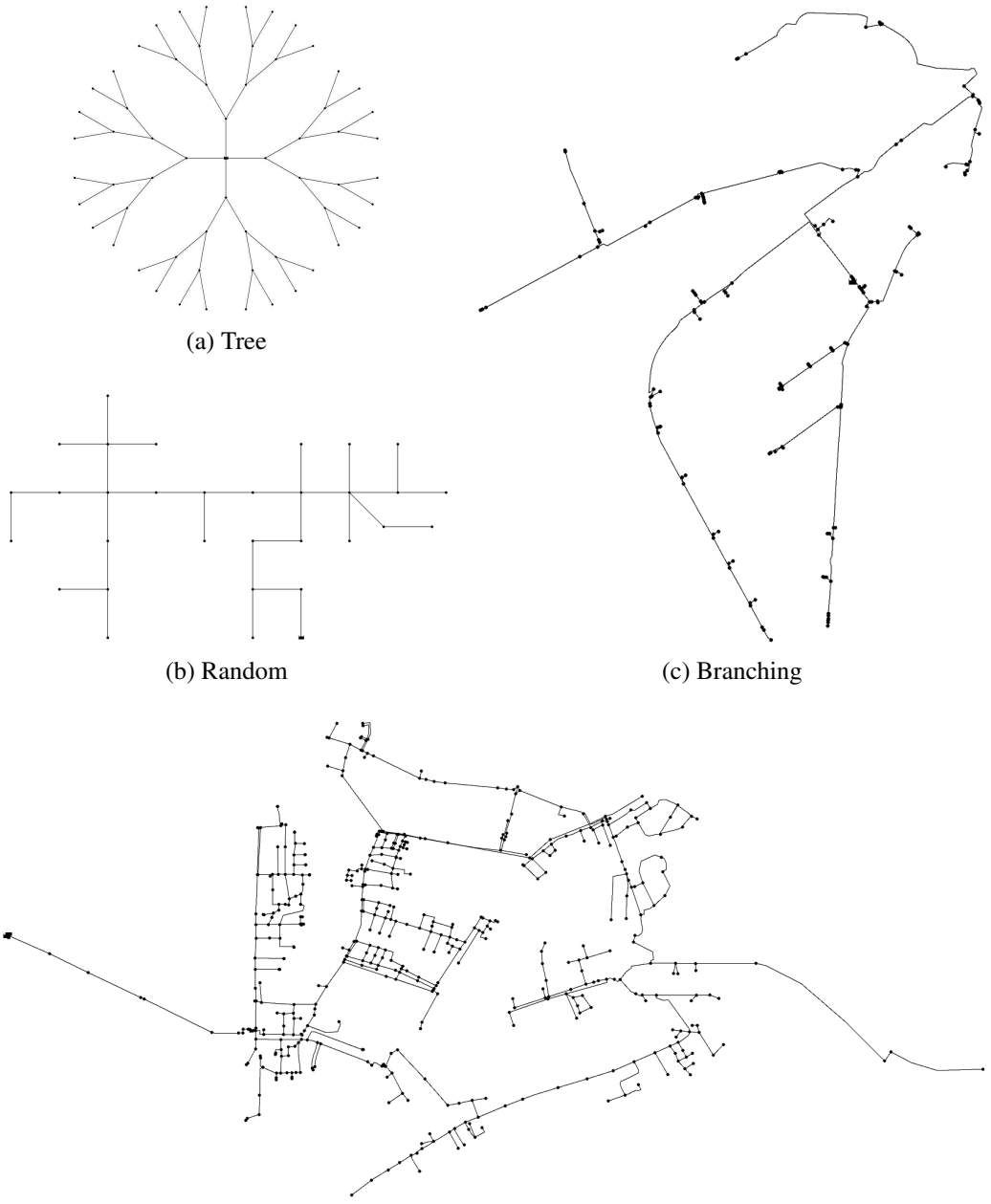
Table 1: Attributes of the four example networks used to study the robots' behavior. Networks with an A suffix are artificially created and networks with an R suffix are from real pipe network data.

Network	Junctions	Links	Total Length (m)	Longest Direct Path (m)	# of Junctions with Degree					
					1	2	3	4	5	6
Tree (A)	61	60	600.00	80.00	32	0	28	1	0	0
Random (A)	32	31	310.00	120.00	15	9	4	3	1	0
Branching (R)	168	175	2858.50	1187.53	38	87	39	1	1	2
Looping (R)	510	560	28765.51	4489.83	124	188	172	26	0	0

time-step was selected to ensure that agents could not travel along the entirety of a network's shortest pipe in a single simulation update. For example, the looping network's shortest pipe was 0.1 m in length, meaning the agents would need to be travelling at $1\,\mathrm{ms}^{-1}$ for this situation to occur. In order to reduce the effects of starting conditions, 20 iterations (Monte Carlo simulations) of each case were performed with a random seed and the results were combined to determine the mean and standard deviation in the *time between inspections*. These are our two chosen measures of performance. The mean gives the arithmetic average of the inspection regularity experienced by a network over the simulation duration, and the standard deviation gives a measure of any irregularities experienced, both between pipes and between different time periods. The choice of 20 iterations was made to balance the accuracy versus the computational costs of the simulation runs. Considering that each simulation is 24192000 individual update loops, a single run would take hours if the number of iterations were chosen greater than 20. The TBI values were normalized by the robots' speed and pipe network length:

$$T_n = \frac{T_{bi} \times S}{L}, \tag{1}$$

where T_{bi} is the TBI predicted from the simulation, T_n is the dimensionless, normalized TBI, S is the robots' nominal travel speed, and L is the total length of all the pipes in a given network. The proposed normalization procedure allowed for comparison of the results predicted for networks of different topologies and lengths.



(a) Tree

(b) Random

(c) Branching

(d) Looping

Figure 4: Two artificial networks (a-b) and two real district metered area (DMA) networks (c-d) used to study the virtual stigmergy behavior of a robot swarm. Links (pipes) are represented as lines, junctions are solid circles, and the main flow inlet junction is depicted as a square.

4.1. Communication Method

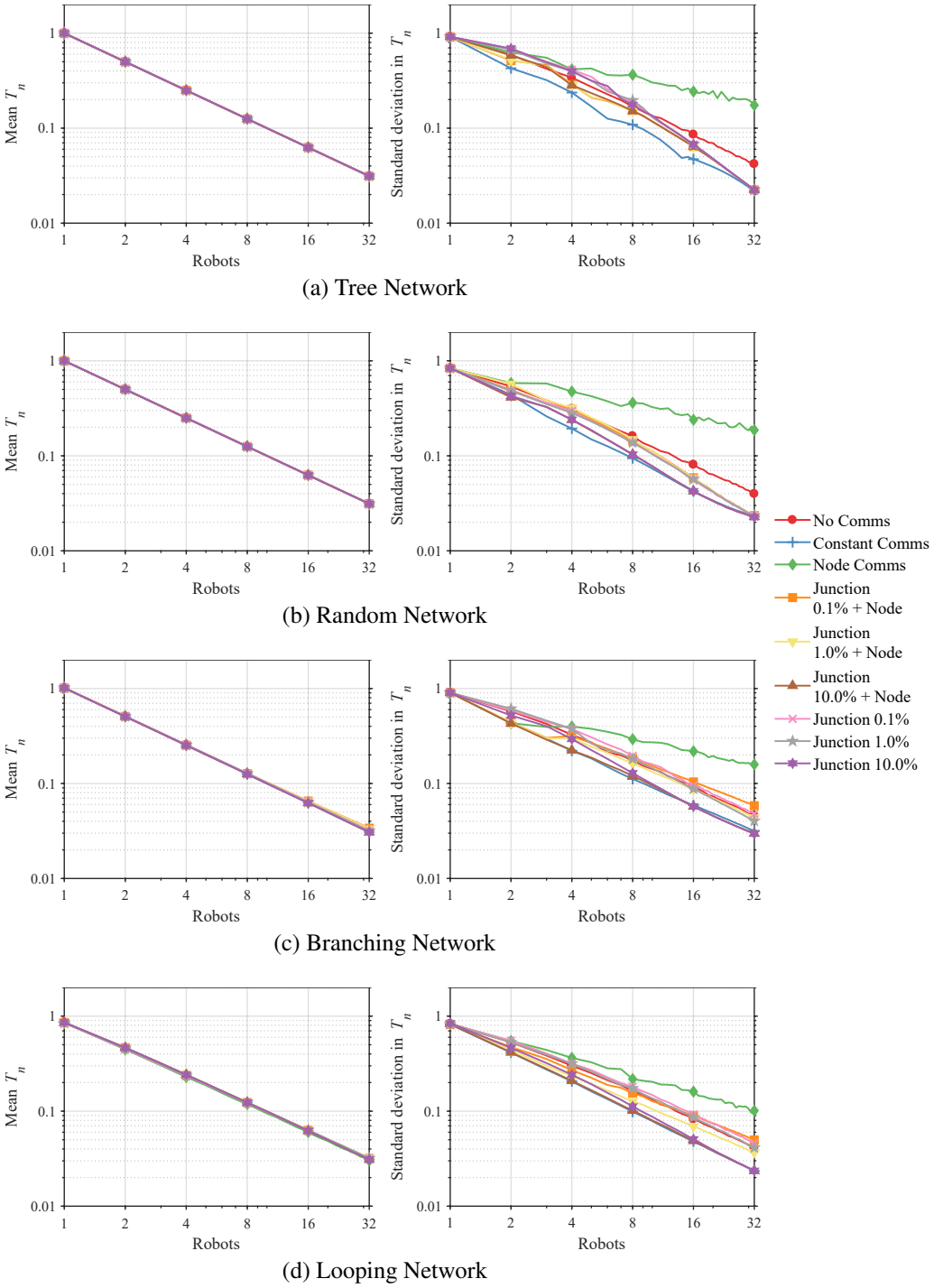
Five different methods of communication were simulated:

- *No Communication* - Agents are only aware of the latest inspections they themselves have performed.
- *Constant Communication* - Agents are aware of all the latest inspections that have been performed.
- *Node Synchronisation* - Agents synchronise their knowledge of when inspections last occurred with an external system that is connected to all communication node junctions.
- *Junction Broadcast* - Agents broadcast their knowledge of when inspections last occurred a given distance from the junction they are at, for other agents in proximity to receive.
- *Junction & Node Communication* - Agents both synchronise at communication nodes and broadcast at junctions (including those that are communication nodes).

The broadcast range of the latter two methods was set to 0.1, 1.0 and 10.0% of the *longest direct path* of each network (see Table 1), giving a total of nine communication conditions to simulate.

The mean and standard deviation of T_n as a function of the number of agents in the four networks is shown in Figure 5. These results were obtained for the no-flow condition in the pipes and for agents travelling at 0.1 ms^{-1}. As can be seen, the mean T_n for all the networks follows the same linear trend in log-log space irrespective of the communication condition chosen. The reason for this is currently unclear and deserves a separate mathematical interpretation. The introduction of additional robots to the system exponentially reduces the T_n required to inspect the network. T_n is in an inverse relationship with the number of robots deployed in the pipe network. The dependence of the mean T_n on the type of network is marginal.

The graphs with the standard deviation in T_n show the effect of the communication condition as a function of the number of agents. This effect is strong. For any of the four networks having the robot agents in constant communication with each other leads to a steeper decline in the standard deviation of the T_n when compared to the no communication case. The higher values of the standard deviation in the *time between inspections* are when there is no communication at all or when the robots use node communication. In the particular case of the tree network (see Figure 4(a)), the standard deviation in the T_n can be up to an order of magnitude higher when communicating at a node (e.g. see 'node comms' vs 'junction 10%' for 32 robots in Figure 5(a)). Broadcasting at junctions reduces the standard



(a) Tree Network

(b) Random Network

(c) Branching Network

(d) Looping Network

Figure 5: Log-log plots of the mean (left) and standard deviation (right) normalized *time between inspections* (T_n) as a function of the number of agents working under nine communication conditions and travelling at 0.1 ms^{-1} in networks without flow.

deviation in T_n considerably in comparison with no-communication case. Communicating over a longer broadcast range produces results that converge towards the constant communication case as the number of robots increases.

Node synchronisation presents an interesting case, as for all networks the standard deviation in T_n showed much less dependence on the number of agents present in the network than in any other case considered here. Although synchronisation at communication nodes causes all robot agents within the network to share knowledge about the *time since last inspection* (TSLI) started of each pipe in the networks, this sharing makes them unaware of the actions of other agents in their immediate proximity until the next node is reached. This leads to the same decisions being made by multiple agents resulting in the agents clustering together and attempting to inspect the network as a crowd rather than spreading out to perform more uniform inspection across the whole network. When node synchronisation is combined with junction broadcasting the clustering effect disappears. Additionally, at broadcast ranges greater than 1.0% of the *longest direct path* this combination acts as a 'range amplifier', reducing the standard deviation in T_n considerably (e.g. see the result 'junction 10% + node' communication condition in Figure 5(a))).

Overall, these simulations suggest that for networks either without flow or with robots able to overcome all flow velocities, in-pipe communication at a certain range is necessary to receive a benefit from the proposed stigmergy behavior beyond that achieved by merely inserting additional robots into the pipe network. Additionally, node synchronisation alone is not a viable method of sharing stigmergy information between robot agents, but can aid other methods.

4.2. Flow and Driving Speed

To understand how a 24-hour flow cycle affects the inspection performance of robot agents within the two real-world networks, simulations were run for all nine communication conditions for driving speeds ranging from $0.025\,\mathrm{ms}^{-1}$ to $0.5\,\mathrm{ms}^{-1}$, in $0.025\,\mathrm{ms}^{-1}$ increments. The upper speed was chosen as it was $\geq 99.5\%$ of the flow velocities in both networks, and the lower speed and increment were chosen to give sufficient resolution across the range. We considered two scenarios, one in which the flow in the branching and looping networks only affected the decision making process, and one where the flow in these two networks affected both the decision making process and robots' speed. In this process the speed of robots traversing each pipe is updated as the flow conditions change. Figure 6 shows the results for robots travelling at the speed of $0.1\,\mathrm{ms}^{-1}$ in the presence of flow (in the interest of paper length raw results for other speeds are not shown). It is of interest to compare these results against the no-flow results at the same speed shown in Figure 5(c-d). Clearly, the addition of flow to the simulation had a strong effect on both the mean and standard deviation of T_n predicted at this

robots' speed. In the presence of flow the *time between inspection's* dependence on the number of agents is less than that in the case of no-flow. There is a much greater effect of communication condition on the mean and lesser effect on the standard deviation in T_n. For the branching network node synchronisation visibly remains the poorest choice because of the robots' clustering effect.

The results presented in Figure 6 suggest that the dependence of the mean T_n and its standard deviation on the number of agents is approximately linear when plotted on the logarithmic scale, i.e.:

$$\log(T_n) = b\log(N) + \log(a), \tag{2}$$

where a and b are the coefficients in the regression. These coefficients relate to the pipe network topology, flow and communication conditions and they have a clear physical meaning. The coefficient a controls the absolute value of the mean T_n or its standard deviation in a pipe network being inspected with one robot only. The value of the coefficient b relates to the rate at which the mean T_n or its standard deviation reduces with the increased robot agent number (N). Basically, this pair of coefficients provides a mathematical estimate of how the T_n and its statistics would change with a change in the conditions in a realistic pipe network. An alternative form of eq. (2) is:

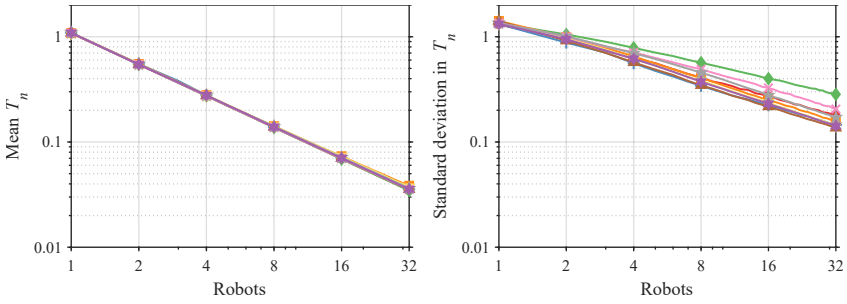
$$T_n = aN^b. \tag{3}$$

The application of the above equation to (1) yields its non-normalized form:

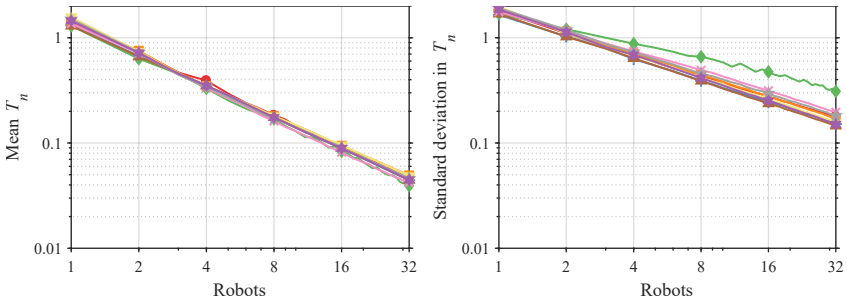
$$T_{bi} = aN^b\frac{L}{S}, \tag{4}$$

where T_{bi} is the mean, non-normalized TBI or its standard deviation expected from the deployment of N agents in a given pipe network. The unit for T_{bi} can be flexible depending upon the choice of the units for L and S. This approach is likely to make the model scalable and valid for a range of network topologies and lengths.

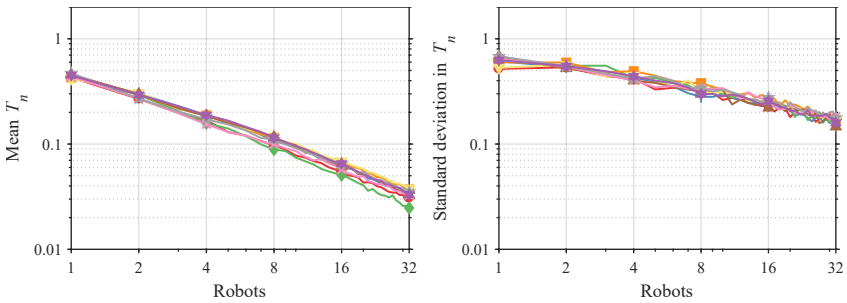
We applied the above regression analysis to the mean T_n and its standard deviation data obtained for the case when the flow was affecting the agent's decision making process only and that when it was affecting both the agents' decision making process and robots' speed. This enabled us to determine the values of coefficients a and b in eq. (2) as a function of the communication and flow conditions and decision process. The results are shown in Figures 7 and 8 where the two coefficients are plotted against the normalized robots' speed for a range of communication conditions. In these figures the robots' speed (S) was normalized by the maximum flow velocity (V_f) in the network: $V_f = 0.7744\,\mathrm{ms}^{-1}$ for the branching network; and $V_f = 2.0775\,\mathrm{ms}^{-1}$ for the looping network. The results shown in



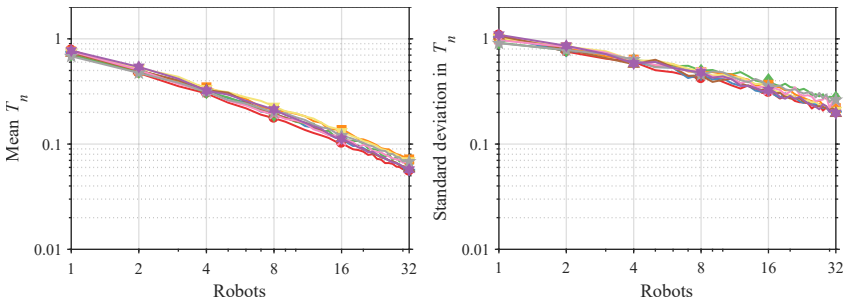
(a) Branching Network - Flow affecting decision only



(b) Branching Network - Flow affecting decision and driving speed



(c) Looping Network - Flow affecting decision only



(d) Looping Network - Flow affecting decision and driving speed

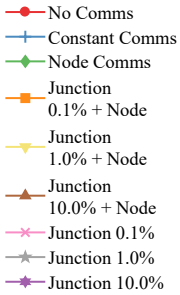


Figure 6: Log-log plots of the mean (left) and standard deviation (right) in the normalized TBI as a function of the number of agents and communication conditions taken over a 24-hour flow cycle. The robots' speed is 0.1 ms^{-1}.

these figures correspond to the regression model fit with the coefficient of determination $R^2 \geq 95\%$. A few of the robot driving speeds produced results for which the fit was poorer than that so were excluded from these figures. The full data set and plots of T_n versus N for each S, and the Matlab code used to perform this analysis, can be found in the supplementary material that accompanies this paper (Parrott et al. (2019)).

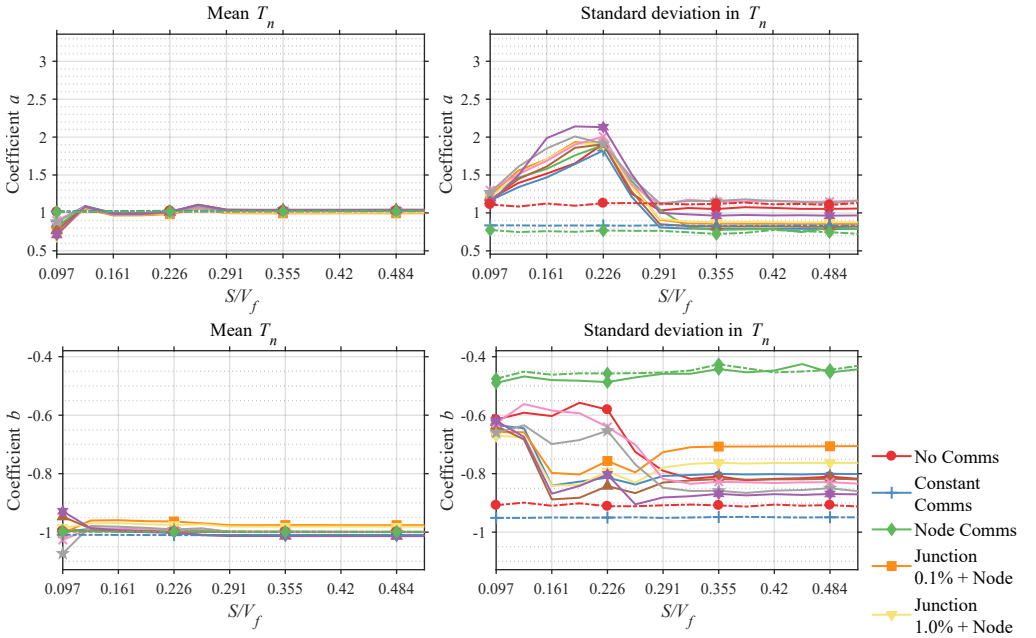
Figures 7 and 8 show that the flow condition affects more strongly the values of the two coefficients at a relatively low robots' speed. This observation is true for the regression coefficients obtained both for the mean T_n and its standard deviation. At a higher robots' speed this effect is reduced, particularly on the values of the two coefficients obtained from the regression analysis of mean T_n data. The point at which this transition occurs is around 32% and 14% of the maximum flow velocity for the branching and looping network, respectively. It should be noted that the transition points of the two networks were the same for both flow scenarios tested, suggesting that simulations where only the decision making process of the robots is affected by flow are sufficient for determining this value. However, when looking at results lower than the transition point, features that are observed when flow affects both decision making and driving speed are not reproduced by the scenario where driving speed is not affected, meaning those results are of limited value. An example of this is the peak of the standard deviation coefficient a at $S/V_f = 0.226$ in Figure 7(b), which is not present in Figure 7(a).

Overall, the results from the simulations suggest that although in-pipe inspection robots need to be able to drive against the flow in order to successfully perform inspection with the proposed behavior, they do not need to be capable of driving against all flow velocities that can occur in the pipe network.

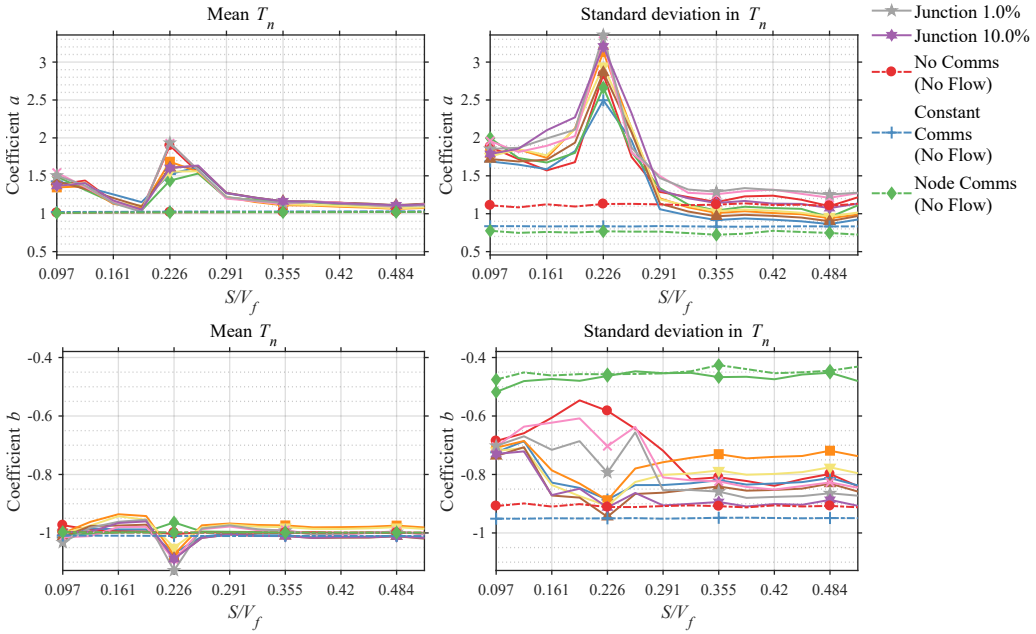
5. Conclusions

In this paper we have demonstrated that from an intelligence perspective autonomous robots are a viable method of future inspection of realistic buried pipe networks. This can be achieved with a relatively compact swarm of robots ($1 \leq N \leq 32$) replicating a biologically-inspired behavior that allows for complete and consistent coverage of a realistic pipe network without the need for a centralized planner, control system, or high certainty prior knowledge of the network layout. In the future, these swarms of robots can be developed and equipped with the right sensors to detect leaks in clean water pipes, blockages and structural damage in wastewater pipes, and gas escape in gas pipes. The buried pipes can also be retrofitted with communication nodes to convey this information from robots to the control center.

The principle of this behavior has been detailed and simulations have been performed for two artificial and two real clean water networks of pipes. These

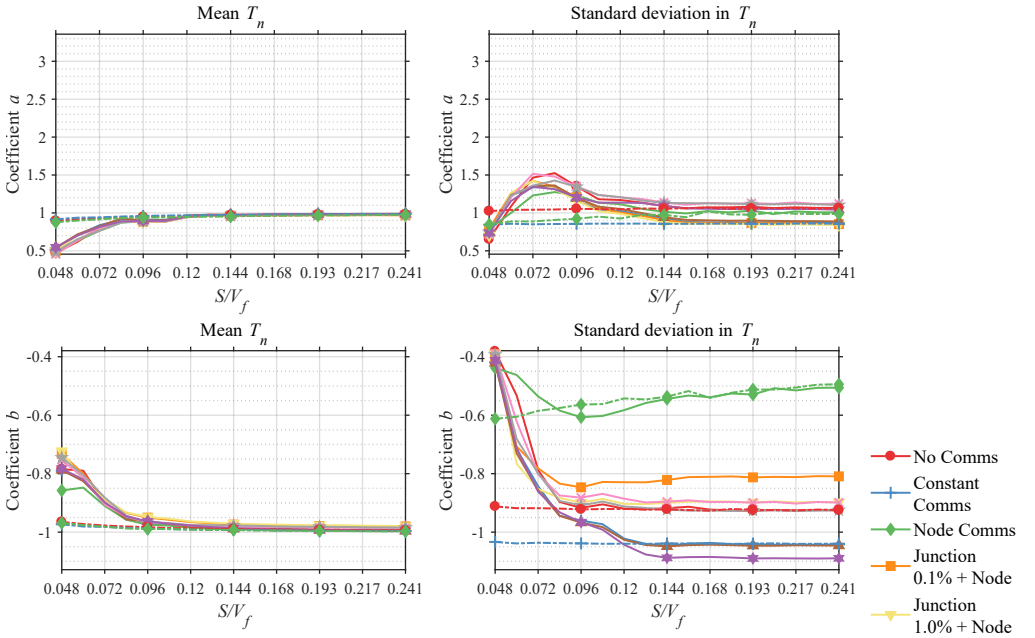


(a) Flow affecting decision only

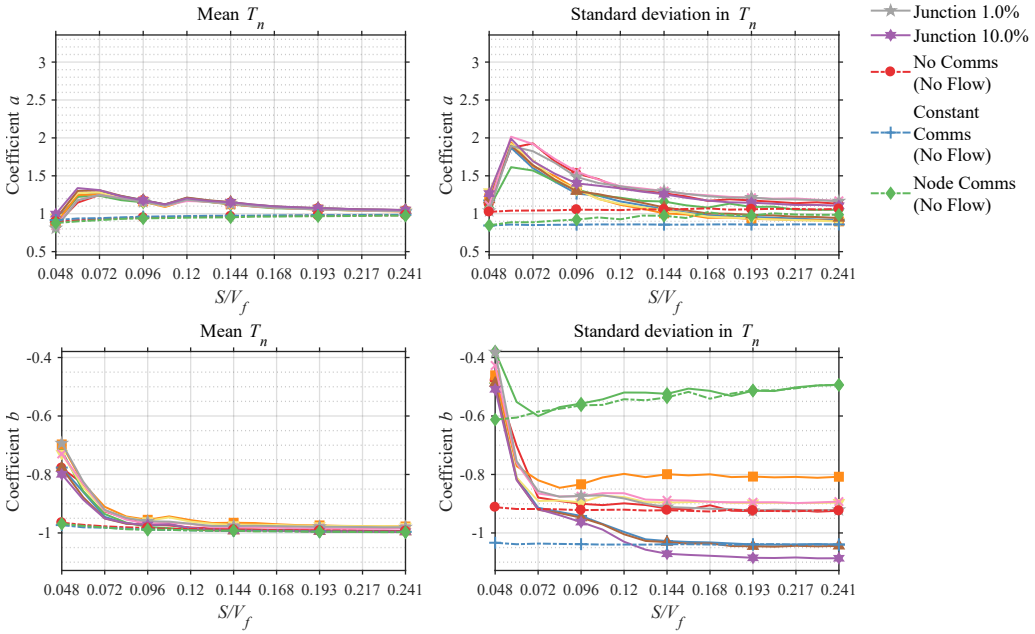


(b) Flow affecting decision and driving speed

Figure 7: The values of the two regression coefficients a and b from equation $\log(T_n) = b\log(N) + \log(a)$ as a function of the normalized robots' speed and communication conditions derived for the branching pipe network: fitted to the mean normalized TBI data (left); fitted to the standard deviation in the normalized TBI data (right).



(a) Flow affecting decision only



(b) Flow affecting decision and driving speed

Figure 8: The values of the two regression coefficients a and b from equation $\log(T_n) = b\log(N) + \log(a)$ as a function of the normalized robots' speed and communication conditions derived for the looping pipe network: fitted to the mean normalized TBI data (left); fitted to the standard deviation in the normalized TBI data (right).

simulations have enabled us to study the effects of communication and flow velocity on the robots' behavior given its relatively simple decision rules. We have applied a regression analysis to the data obtained from our simulations. This analysis resulted in two regression coefficients that relate to the *time between inspections* and to the rate at which this time reduces with the increase in the number of robot agents inspecting the pipes. These coefficients provide a good mathematical estimate of how the *time between inspections* and its statistics would change with the change in the conditions in a realistic pipe network.

The presented results for the robots' behavior scale with the number of robots introduced, network topology and length. The work shows the importance of communication to reduce the standard deviation in the *time between inspections*. It has been shown that the ability of robots to communicate between each other even within a limited range helps them perform better and to cover the pipe network more uniformly. It has been discovered that having robots communicate exclusively via communication node locations is not a viable method of sharing behavior data and should be avoided unless paired with in-pipe communication. It has been shown that the mean normalized time between inspections, T_n, for all the networks follows the same linear trend in log-log space irrespective of the communication condition chosen. The reason for this is currently unclear and deserves a separate mathematical interpretation. The introduction of additional robots to the system exponentially reduces the T_n required to inspect the network. T_n is in an inverse relationship with the number of robots deployed in the pipe network. The dependence of the mean T_n on the type of network is marginal. It has also been shown that the robots' behavior can adapt to a 24-hour flow cycle in which the flow velocity can change significantly and prevent robots from travelling through the pipes. There are clear benefits from having a percentage of robots being able to travel against the flow in a realistic pipe network. Future work will involve introducing more physical attributes to the simulation such as size and weight restrictions, analysis of the relative efficiency of various propulsion methods, power management, and methods of controlling the swarm to allow for priority pipes to be inspected more frequently.

Acknowledgment

This work was supported by the Engineering and Physical Sciences Research Council [Grant No. EP/N010124/1 and EP/S016813/1]. Thanks are given to Yorkshire Water for providing the real network data used in this study. The authors are grateful to Prof. Simon Tait, Dr. Richard Collins and Gavin Sailor for their useful comments on practical aspects of the application of robots in real pipe networks.

References

Ahn, B.H., Shin, J.Y., 1991. Vehicle-routeing with time windows and time-varying congestion. Journal of the Operational Research Society 42, 393–400. doi:10.1057/jors.1991.81.

Calderon, D., et al, 2014. Pipe condition assessment using CCTV. https://www.nassco.org/sites/default/files/SPECIFICATION GUIDELINE - CCTV 15Dec2014.pdf. (last accessed: 18/02/2020).

Camazine, S., Franks, N.R., Sneyd, J., Bonabeau, E., Deneubourg, J.L., Theraula, G., 2001. Self-Organization in Biological Systems. Princeton University Press, Princeton, NJ, USA. ISBN: 0691012113.

Christofides, N., 1973. The optimum traversal of a graph. Omega 1, 719 – 732. doi:10.1016/0305-0483(73)90089-3. ISSN: 0305-0483.

Dijkstra, E.W., 1959. A note on two problems in connexion with graphs. Numerische Mathematik 1, 269–271. doi:10.1007/BF01386390. ISSN: 0945-3245.

Dirksen, J., Clemens, F., Korving, H., Cherqui, F., Le Gauffre, P., Ertl, T., Plihal, H., Müller, K., Snaterse, C., 2013. The consistency of visual sewer inspection data. Structure and Infrastructure Engineering 9, 214–228. doi:10.1080/15732479.2010.541265.

Dorigo, M., Bonabeau, E., Theraulaz, G., 2000. Ant algorithms and stigmergy. Future Generation Computer Systems 16, 851 – 871. doi:10.1016/S0167-739X(00)00042-X. ISSN: 0167-739X.

Eksioglu, B., Vural, A.V., Reisman, A., 2009. The vehicle routing problem: A taxonomic review. Computers & Industrial Engineering 57, 1472–1483. doi:10.1016/j.cie.2009.05.009. ISSN: 0360-8352.

Hart, P.E., Nilsson, N.J., Raphael, B., 1968. A formal basis for the heuristic determination of minimum cost paths. IEEE Transactions on Systems Science and Cybernetics 4, 100–107. doi:10.1109/TSSC.1968.300136. ISSN: 0536-1567.

Horoshenkov, K., Long, R., Tait, S., 2010. Improvements in and relating to apparatus for the airborne acoustic inspection of pipes. Patent WO2010020817.

Jünger, M., Reinelt, G., Rinaldi, G., 1995. Chapter 4 the traveling salesman problem, in: Network Models. Elsevier. volume 7 of *Handbooks in Operations Research and Management Science*, pp. 225 – 330. doi:10.1016/S0927-0507(05)80121-5. ISSN: 0927-0507.

Laggis, L., 2016. Get a better look at your sewer pipes. https://www.mswmag.com/editorial/2016/11/get_a_better_look_at_your_sewer_pipes. (last accessed: 16/04/2019).

Laporte, G., 1992. The vehicle routing problem: An overview of exact and approximate algorithms. European Journal of Operational Research 59, 345–358. doi:10.1016/0377-2217(92)90192-C. ISSN: 0377-2217.

Liu, Z., Kleiner, Y., 2013. State of the art review of inspection technologies for condition assessment of water pipes. Measurement 46, 1–15. doi:10.1016/j.measurement.2012.05.032. ISSN: 0263-2241.

Mills, G.H., Jackson, A.E., Richardson, R.C., 2017. Advances in the inspection of unpiggable pipelines. Robotics 6. doi:10.3390/robotics6040036. ISSN: 2218-6581.

Nuron, 2019. Make fibre make sense. http://www.nuron.tech. (last accessed: 16/04/2019).

Ogai, H., Bhattacharya, B., 2018. Pipe Inspection Robots for Structural Health and Condition Monitoring. Springer. doi:10.1007/978-81-322-3751-8. ISBN: 978-81-322-3749-5.

Osterhues, A., Mariak, F., 2005. On variants of the k-Chinese postman problem. Operations Research und Wirtschaftinformatik, Universität Dortmund (last accessed: 18/02/2019).

Parrott, C., Dodd, T., Boxall, J., Horoshenkov, K., 2019. Supplementary Material for: Simulation of the Behavior of Biologically-Inspired Swarm Robots for the Autonomous Inspection of Buried Pipes. doi:10.17632/WH3H75BX6Y.1.

Pinciroli, C., Lee-Brown, A., Beltrame, G., 2016. A tuple space for data sharing in robot swarms, in: Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (Formerly BIONETICS), ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), ICST, Brussels, Belgium, Belgium. pp. 287–294. doi:10.4108/eai.3-12-2015.2262503. ISBN: 978-1-63190-100-3.

Plihal, H., Kretschmer, F., Ali, M.T.B., See, C.H., Romanova, A., Horoshenkov, K.V., Ertl, T., 2016. A novel method for rapid inspection of sewer networks: combining acoustic and optical means. Urban Water Journal 13, 3–14. doi:10.1080/1573062X.2015.1076857.

Pure Technologies, 2014. SmartBall - Leak and Gas Pocket Detection. http://puretechltd.com/technology/smartball-leak-detection/. (last accessed: 16/04/2019).

Pure Technologies, 2017. PipeDiver - Innovative, free-swimming condition assessment platform for water and wastewater pipelines. https://puretechltd.com/technology/pipediver-condition-assessment. (last accessed: 16/04/2019).

Sadeghioon, A.M., Metje, N., Chapman, D.N., Anthony, C.J., 2014. SmartPipes: Smart wireless sensor networks for leak detection in water pipelines. Journal of Sensor and Actuator Networks 3, 64–78. doi:10.3390/jsan3010064. ISSN: 2224-2708.

Thimbleby, H., 2003. The directed chinese postman problem. Software: Practice and Experience 33, 1081–1096. doi:10.1002/spe.540.

Tiratsoo, J.N., 1992. Pipeline pigging technology. Gulf Professional Publishing. ISBN: 0872014266.

US EPA, 2018. EPANET - Model for Water Distribution Piping Systems. https://www.epa.gov/water-research/epanet/. (last accessed: 16/04/2019).

Van Der Walt, S., Smith, N., 2015. matplotlib colormaps. https://bids.github.io/colormap/. (last accessed: 16/04/2019).

Water UK, 2019. Leaking pipes. https://www.discoverwater.co.uk/leaking-pipes. (last accessed: 3/10/2019).