Universiteit Leiden

Master Computer Science

Studying the impact of obstructed communication using a novel multi-agent foraging simulation.

Name:	Jasper Ruigrok
Student ID:	S1969838
Date:	13/5/2025
Specialisation:	Artificial Intelligence
1st supervisor:	Mike Preuss
2nd supervisor:	Tessa Verhoef

Master's Thesis in Computer Science

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Contents

Abstract

Multi-agent systems are prevalent in nature, within human society, and more recently are being employed in fields such as swarm robotics. However, these systems inherently depend on the emergent cooperation between individual agents, which can make them vulnerable to disruptions in their coordination. The goal of this work is to study how obstructions in the communication between agents can affect the behaviour of a group of cooperating agents. Here a foraging simulation is created that emulates a group of agents that work together to find and deliver food in a discrete map. These agents are guided by a pre-defined behaviour tree and are able to communicate by sharing information about discovered food sources. As a baseline, the behaviour of the group was first examined without any communication obstructions. Communication was then hindered by giving agents a set chance to miss-communicate, which distorts the information received by the receiving agents. In certain circumstances this caused the group to become fixated on non-existent food sources. It was found that this effect only happens when information can be passed around frequently enough, allowing false information to be preserved by spreading through the group. Limiting how often agents can communicate, reducing the maximum communication range and decreasing the amount of agents was found to reduce the formation of this phenomenon. Making agents more skeptical by allowing them to ignore some of the false information was also found to make the group more resistant to this effect.

1 Introduction

Termite colonies, teams of rescue drones and sports teams all share a common trait: they rely on a group of autonomous agents that cooperate in order to fulfil a common goal. These systems, where individuals interact and coordinate their actions, are present across a wide range of disciplines. In computer science, multi-agent systems (MAS) are often used as a means of solving problems that are hard to solve using a monolithic setup. Here the field of natural computing does this by taking inspiration from such systems found in nature. An interesting example of this is the ant colony optimization algorithm, which is inspired by the foraging behaviour of real ant colonies [2]. Another example is the artificial bee colony algorithm, which is based on how consensus is formed in honeybee swarms [8]. The field of swarm robotics takes this idea one step further, using swarms of physical robots to perform tasks. For example, in Shader et al.[11], up to 64 robots were used in a brick-layering scenario.

In other areas, agent-based models (ABM) are instead used to increase our understanding of the behaviour of these complex systems. Here the goal is not to create an application, but to be able to make predictions about multi-agent systems. This is used in fields such as sociology, where the behaviour of humans is modelled using computational models. One example of this is van der Kam et al. [14], where the charging behaviour of electric vehicle drivers is modelled.

These examples show that multi-agent systems are used effectively in many different situations. In nature, the social organization of ant colonies allow them to be one of the most successful animal groups on the planet, with ants having colonised almost every landmass on earth. Human society also implicitly makes use of this distributed approach, for example in companies where tasks are divided among cooperating individuals. In technology, using this approach has several benefits over using a monolithic system, in which decisions are made by a single governing entity. Spreading a task across multiple agents can offer more flexibility as agents can make autonomous decisions. This division of labour can also limit overhead costs, thereby making the system more efficient[5]. However, these systems also come with a challenge. The emergent nature of multi-agent systems can make it hard to predict how a system will behave under different circumstances, sometimes leading to unexpected and unwanted outcomes. One extreme example of this is in colonies of ants. Ants make use of pheromones, a chemical substance used for communication. When an ant finds food, it leaves a trail of pheromones which allows other ants to also find the discovered food. However, if a group of ants gets separated from the colony these ants sometimes end up following their own trail, forming a loop. These ants then follow a circular path, reinforcing the trail, until the ants die of exhaustion. This phenomenon is known as an ant mill or death spiral, named after the circular rotating movement of the group.

One of the most important aspects of these distributed systems is communication between individual agents. When unexpected disruptions affect communication between agents, for example as seen in ant mills, the consequences can be severe. The goal of this work is to study how disturbances in the communication of distributed systems can cause these systems to lose performance or even fail entirely.This can give an indication of how different circumstances affect the impact of these communication problems. When building a multi-agent system, for example, in the field of swarm robotics, this prior knowledge may help in preventing unintended behaviour. This is also relevant to fields such as sociology, where preventing miscommunication between human actors is important.

1.1 Overview

In this work, our aim is to study how obstructions in communication impact the performance of multi-agent systems. Our goal is to identify what circumstances make a system most vulnerable to problems with communication between agents. Here we study three different types of factors that impact these systems:

- The external environment of the system. These are factors that are not directly controlled by the group of agents, like the layout of the physical space around the group.

- The internal workings of the group. This includes things like the number of agents in the group and the behaviour of the individual agents.
- Constraints placed on the communication between agents. This includes constraints like limiting the amount, range and accuracy of communication.

In order to study the effect of these different conditions, a simulation is created that models a group of foraging agents that aim to collect food placed on a discrete map. Here the behaviour of the agents is controlled through a behaviour tree, which is a rule-based data structure that can be easily modified and designed. These agents are homogeneous, they share the same behaviour and perform the same tasks, and work together by sharing information about discovered food sources. The objective of the group is to maximise the amount of food gathered in set amount of time. This allows us to evaluate the performance of the group quantitively, allowing us to assess the impact of different factors. The impact of communication obstructions is then studied in four steps:

1. A foraging simulation is created where a group of agents cooperate in order to gather food in a limited amount of time. Agents can cooperate by sharing information about discovered food sources.
2. In the base setup, the impact of different parameters, such as the population size and the layout of the map, is explored without intentionally obstructing communication. This gives an idea of how the group behaves under ordinary circumstances.
3. Communication constraints are now introduced. These constraints serve to limit or distort communication between individual agents.
4. The impact of several parameters is again tested. The behaviour and performance of the group can now be compared to the results found in the base setup. This gives some indication of which parameters are relevant when communication is hindered.

2 Simulation

2.1 Foraging simulations

As discussed in the introduction, solving problems with a group of agents has some advantages over using a centralized system. The downside of these distributed systems is that their emergent nature makes it harder to predict what will happen in a given situation. This can be a large disadvantage when it is important that a setup works the way it is intended to work. Knowing where such a system is most vulnerable can mitigate this risk. The goal of this work is to create a test environment of cooperating agents and see what kind of disruptions are most hindering to the performance of the group. There are many types

of simulations that have already been made in order to study the behaviour of groups of agents. One example of this is in Deadman et al., 1999[4], where an agent simulation was used to study the tragedy of the commons. Studying group behaviour is also done in fields such as sociology, for example in Wang et al, 2022[17], where a simulation was made in order to model the evacuation of pedestrians from disasters.

Before we present the simulation created in this work we highlight some properties that the simulation is expected to have:

- The mechanics of the simulation should be relatively simple.
- Emergent behaviour should be possible.
- The problem introduced in the simulation should be non-trivial.
- The computational resources required to run the simulation should be minimal.
- The problem solved by the agents in the simulation should be comparable to problems found in the real world.

The mechanics of the simulation should be simple for two reasons: first, this makes it easier to analyse the results and second, this makes the results more generalisable to other scenarios. The group of agents in the simulation should also display emergent behaviour. Without emergent behaviour the results are not likely to present new insights. The same is the case for the third point, if the problem is trivial to solve then the simulation is not a good test environment. Lastly, the computing power needed to run the simulation should be minimized. A high computation cost will either limit the scope of what can be explored or harm the accuracy of the results. For example, a simulation that involves a physics system would be less preferable as this often involves a higher performance requirement.

Primarily for these reasons a decision was made to create a simple foraging-based simulation. Multi-Agent Foraging (MAF) problems involve groups of agents that cooperate to either gather, find or move some objects. One main advantage of this setup is that it mimics many real world problems such as finding unexploded mines (Acar et al., 2003 [1]), searching for fire (Marjovi et al., 2009 [9]) or transporting resources (Vaughan et al., 2000 [15]). Thus understanding the behaviour of a group of agents in a similar setup has the potential to help with these problems as well. MAF problems can also satisfy our other requirements: The mechanics can be relatively simple which makes them easy to study and they do not require extensive computational power. Solving these problems is also non-trivial and requires the agents to work together in an emergent manner.

2.2 Simulation details

Our simulation is a type of foraging simulation, created using the Python programming language. The objective of the agents in our simulation is to gather as much food as possible in a set amount of time. They do this by moving to a piece of food, picking it up, and bringing it back to the delivery point at the centre of the map. Here the agents search for food by traversing a map of discrete square tiles. In order to simplify the problem we allow multiple agents to stand on the same tile, preventing the agents from bumping into each other. Food sources are placed on some of the tiles on the map, which allow agents to pick up a piece of food when standing on the same tile. This food can then be delivered at the centre of the map which increases the score of the group. Not all food is created equal, some food gives more points than other food when brought back. Tiles can also contain multiple pieces of food, which means multiple trips can be made before the food runs out. Additionally, some tiles contain walls which prevent agents from moving over them. As time is limited, the challenge for the group is to maximize the time spent gathering the most valuable type of food. The group can benefit from cooperation by sharing information on the location and value of discovered food sources, but it is also possible for the agents to gather food independently, though this may be less efficient.

The simulation is run using a turn-based system. Each turn, all agents choose one action to perform. They choose between three possible actions:

- The agent moves up, down, left or right to an adjacent tile that does not contain a wall. Agents cannot move diagonally.
- The agent does not move. This is sometimes useful when communicating with other agents.
- The agent picks up the food that is on their current tile. Agents can only hold one piece of food at a time. If the agent then moves onto the tile at the centre of the map their food is removed and the food value is added to the total score.

Additionally agents can also save information to their internal memory, for example the position of food that they found. This is explained in more detail in the next section.

The simulation can be run in two modes: hidden or visualized. In hidden mode multiple simulations can be run in parallel. Afterwards the results, such as the total food collected, are saved. Visualized mode shows a visualization of the simulation in real time. Figure 1 shows a moment in time of the simulation for an example scenario.

This shows a group of six agents gathering from three different patches of food. Here the agents not currently holding food are shown as white circles. When an agent picks up a piece of food they become slightly tinted in the

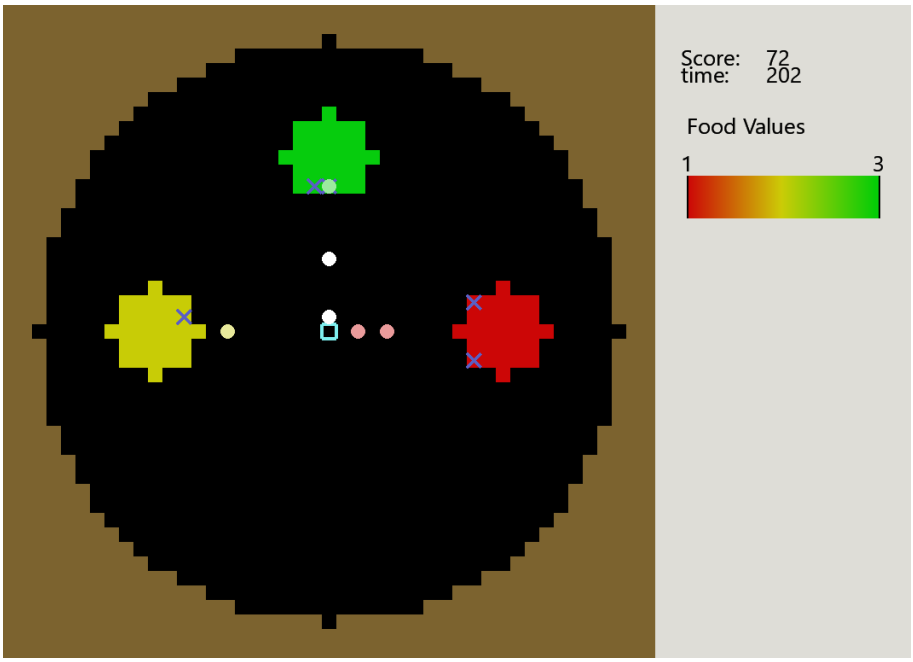


Figure 1: Example of the simulation being run in real-time in a test scenario. Agents, depicted as white circles, start at the blue drop-off point at the centre. The agents are able to gather food from one of the three food deposits by first picking up a piece of food, shown as agents matching the colour of the food, and then delivering the food at the centre. The sidebar shows how much food has been collected so far(Score) and how much has passed(Time) in time steps since the start of the simulation. The colour gradient shows the score received from bring back each type of food. This scenario is only used to illustrate the mechanics of the simulation and is not used further.

colour of the food they picked up. Food is shown as coloured squares, with the colour indicating the value of the food, as indicated by the gradient on the right. In this case the red food has a value of 1, the yellow food a value of 2 and the green food a value of 3. The brown squares surrounding the agents are walls that are impassable. The blue basket in the middle indicates where the agents can drop off food. The blue crosses indicate coordinates that agents have remembered.

2.3 Behaviour trees

The goal of the simulation is to model a group of cooperating agents and to see what disruptions harm the performance of the group. Therefore, a framework for modelling the behaviour of the agents is needed. Though it is possible to explicitly program the behaviour of the agents, in this case it was decided to use

a behaviour tree(BT). Behaviour trees are most commonly used in video games to define the behaviour of non-player characters. However recently they have also received attention in fields such as robotics, see e.g. Iovino et al. [7]. An interesting application is the use of BTs in robot swarms, for example in hogg et al. [6], where behaviour trees were generated using an artificial evolution algorithm. Using behaviour trees in our context has several benefits:

- Behaviour trees are easy to visualise and understand without programming knowledge.
- Behaviour trees can be easily modified, for example by adding an additional condition somewhere within the tree.
- An internal memory system that allows agents to store information is straightforward to implement with behaviour trees.
- Though this is not done in this work, it is possible to use self-learning algorithms, e.g. a genetic algorithm, to automatically generate or adjust a behaviour tree by maximizing an objective value.

Initially Finite state Machines(FSMs) were used for designing the behaviour of the agents. However implementing internal memory proved to be more straightforward using behaviour trees.

2.3.1 Implementation

Behaviour trees consist of a tree of nodes. Each simulation tick the agents traverse this tree, starting from the root, and end at a leaf node that represents the task that the agent will perform. This way the agents decide, based on the conditions within the tree, what action they will perform. The tasks that can be done by the agent consist of pre-defined behaviours that accomplish a sub-task, like picking up food. The full list of tasks that can be performed is shown here:

- **Random walk**: Agents take a random step in a direction not blocked by a wall. This is mainly used for exploration.
- **Return home**: Agents take a step into the direction of the delivery point of the colony at the centre of the map.
- **Gather food**: Agents select the closest food within range and take a step toward it. If the agent is already standing on a tile with food then a piece of food is picked up instead of moving.
- **Go to waypoint**: Agents can remember the coordinates of positions they have previously been on. This task makes the agents return to a memorized position.
- **Stay still**: The agent remains at the same tile.

The function of the behaviour tree is to decide which of these tasks to perform at any given moment. Each node of the tree has zero or more child nodes, which are ordered below the parent node from left to right. A node is executed when it is reached during the traversal of the tree. If the node has child nodes then these nodes are instead executed first, from left to right. After execution, the child node returns a boolean to the parent which indicates whether it was successful or not. The parent then decides to either continue executing the remaining children or pass control back to its own parent node. In our implementation Behaviour trees consist of six distinct types of nodes:

- **Selector**: A composite node with one or more children. When executed it loops through it's child nodes from left to right until a node returns True, in which case the selector returns True. If each child returned False when executed then the selector returns False. This way the selector 'selects' one of the child nodes. It is similar to an OR operator as it returns True when at least one of its child nodes is True, and otherwise returns False.

- **Sequence**: A composite node with one or more children, which can be seen as the opposite of the selector. It also loops through its child nodes from left to right but now stops when a child node returns False, in which case the sequence returns False. The sequence returns True when all of its children have returned true. Therefore the the child nodes form a 'sequence' that most fully executed in order to succeed. It is similar to an AND operator as it only returns True when all its child nodes are true and otherwise returns False.

- **Conditions**: Conditions are leaf nodes that return True or False based on some measurable fact by the agent. An example is the *IsHoldingFood* Condition which returns True when the agent is holding food and False otherwise. Conditions can also take the form of inequalities, for example $NearbyFood > 3$ checks if there is more than three pieces of food nearby. Conditions are usually placed in a sequence before a task so that the task only executes if the condition is met.

- **Tasks**: Task are leaf nodes that make the agent do a pre-defined behaviour such as gathering food. Before executing an action the node checks if the task can be performed, for example the *Gather food* task requires that there is food in range. If the task is possible then an action is performed and the node returns True. Otherwise the node returns False. Trees are constructed so that only one task node can ever return True per tick.

- **Decorators**: Decorators are nodes with exactly one child. Some examples are the 'Succeeder' and the 'Failer' which respectively always return True and False, regardless of the outcome of the child node. Another subset of decorators are loops, which execute child nodes multiple times. Behaviour trees in this work can make use of a 'communication loop' which iterates over all nearby agents within the detection range.

- **Memory node**: Memory nodes change the internal memory of the agent in some way. Some examples could be saving the current coordinate of the agent or incrementing a counter by one. Memory nodes always return True.

In our implementation, behaviour trees can be constructed using pre-defined classes representing subtypes of the above node types. The behaviour tree can then automatically generate Graphviz code that is used to visualize the tree. To illustrate, Figure 2 shows a simple behaviour tree visualized in this manner.

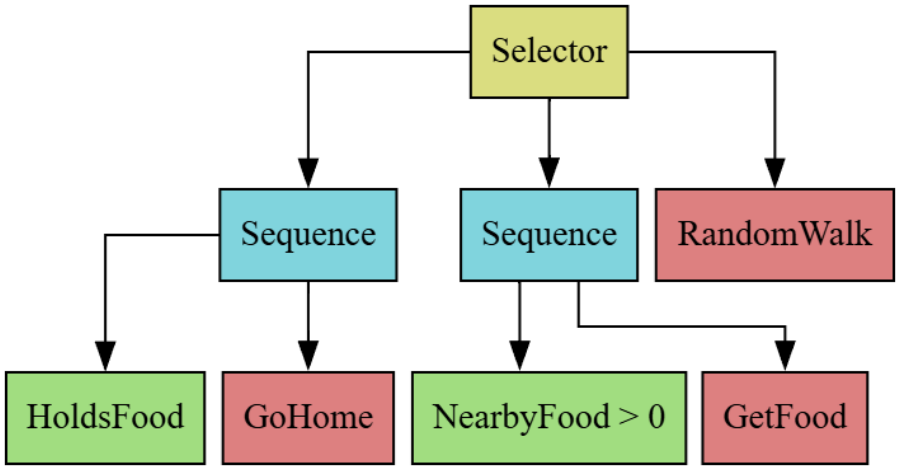


Figure 2: A simple behaviour tree that makes agents independently gather food. Here Selectors are shown in yellow, sequences are shown in blue, tasks are shown in red and conditions are shown in green.

This behaviour tree makes agents walk around randomly until food is detected. The agent then picks up the food and brings it back to the delivery point. The tree starts with a selector as its root, which means it chooses one of it's child nodes to execute. The first child node that is checked is a sequence of a condition and a task. This sequence first checks if the agent is holding food and if so makes the agent walk to drop off the food. If this sequence was successful, i.e. the agent was holding food and was able to make progress walking home, then the behaviour tree stops because the selector found a child node to execute successfully. Otherwise the next child node is checked, another sequence with a condition and a task. This counts the amount of food that the agent can see and then checks if this amount is higher than zero. If so the agent spends this step working towards gathering the closest piece of food. Notice that once the agent has picked up the food, the condition in the previous child node will now return True, making the agent walk back home. If the agent is neither holding a piece of food or sees any food nearby, then the selector will check its last child node which makes the agent walk in a random direction. This serves to make

the agent explore until it detects nearby food.

This behaviour tree is enough to make the agents gather food, but is less efficient than more complex trees. One problem the tree has is that once the agent has brought back a piece of food, it does not remember where it was discovered, and again starts moving randomly. One way to improve the tree is to make use of memory nodes. Agents can at any point save some known value to their internal memory. There are four data types that can be saved to internal memory: Booleans, integer values, continues values and coordinates. Agents can always store their current coordinates to their internal memory, which we will from now on refer to as waypoints. Agents can then return to these waypoints using the *Go Waypoint* task. Figure 3 shows an improved behaviour tree that makes use of memory nodes.

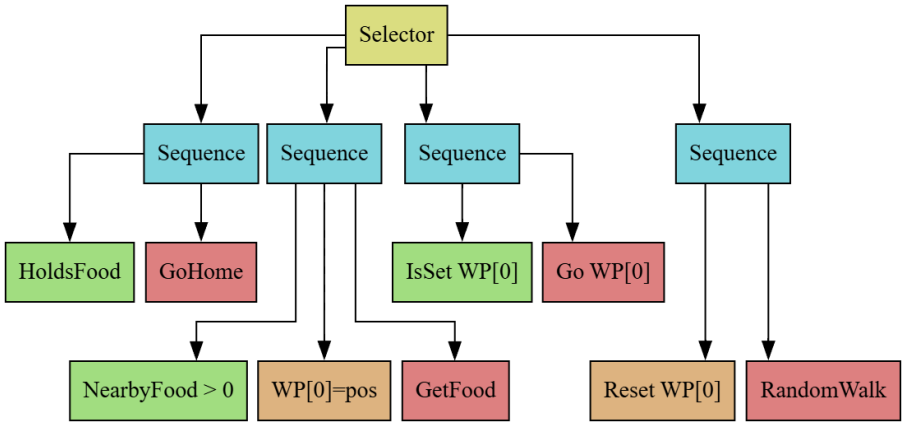


Figure 3: A behaviour tree that is adjusted from the behaviour tree in Figure 2. The tree now contains memory nodes shown in orange that let agents remember the location of food.

Now when an agent detects a piece of food it stores its current coordinate, which is indicated as $WP[0] = pos$. Here the 0 is a memory address and *pos* refers to the current coordinate of the agent. This allows the agent to store multiple coordinates at a time, for example setting $WP[1] = pos$ would allow the agent to store a second waypoint without overwriting the first waypoint. The third sequence now checks if the waypoint at address 0 is set with $IsSetWP[0]$. If so, the agent moves back to the coordinate stored in memory. This makes it so that agents move back and forth between food and drop-off point, without having to rediscover the food every time. In the case that the agent moves back to the waypoint but no food is found, probably because all the food has been gathered, then the agent will execute the fourth sequence. This resets the waypoint with $ResetWP[0]$ and again lets the agent explore for new food with the *RandomWalk* task.

2.3.2 Communication

The previously shown behaviour trees do not include any interaction between the agents, which means the agents operate completely independently of each other. However, it is possible for agents to cooperate by sharing information about the position and quality of the food they have discovered. An example behaviour tree that makes use of communication between agents is shown in Figure 4.

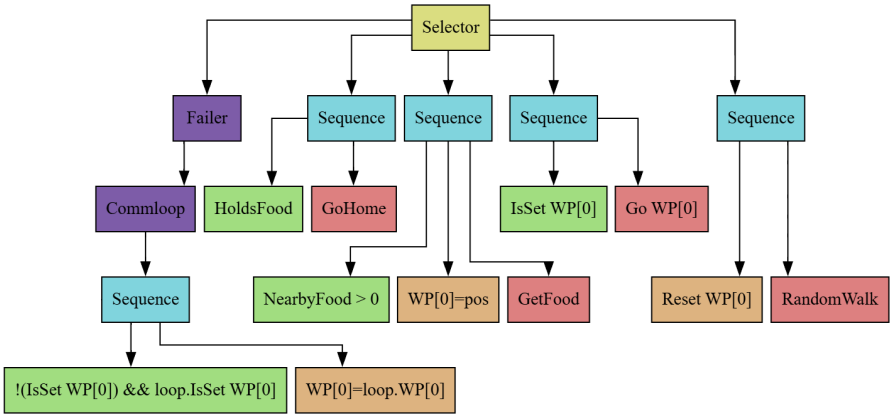


Figure 4: Behaviour tree that allows agents to share the position of food. A *Failer* is included above the communication loop which always returns *False*. This is done so that the selector executes the remaining sub-trees.

Now, when an agent without a waypoint encounters an agent that does have a waypoint, it will copy that waypoint. Here, communication is implemented using a communication loop node, shown on the left in purple. This node executes its child node once for each nearby agent. Descendant nodes of the communication loop are then able to access all information known by the agent that is currently being iterated over, including its internal memory. This is shown in the behaviour tree through the *loop* prefix. For example *loop.WP[0]* retrieves the coordinates saved at memory address 0 for the agent that is currently in the loop. If a variable does not contain a prefix, then it retrieves information from the agent running the behaviour tree. This setup makes it so that communication is performed by the agents that want to retrieve information, while the agents sending the information are not actively involved. Future setups could consider implementing a form of mutual communication where both the sender and the retriever are actively involved, but this is not done in this work.

2.4 Object detection and pathfinding

In the simulation, agents can only detect or interact with objects that are close enough to them. Two parameters determine the detection range of agents:

Smell range(R_{smell}) and *Communication range*(R_{comm}). Food detection uses the *Smell Range* parameter. When the distance between an agent and a piece of food is greater than R_{smell} then the agent will be unable to detect the food. The *Communication range* determines whether two agents are able to communicate. If the distance between two agents is greater than R_{comm}, then the agents will not detect each other when running a communication loop. In both cases, the distance is defined as the minimum amount of steps an agent needs to take to traverse from one coordinate to another. This also takes into account the presence of walls. If the path between two coordinates is blocked by walls then the distance between the coordinates will be longer. This is illustrated in Figure 5. The figure shows the detection boundary for the *Smell range* and *Communication range* of the agent at the centre.

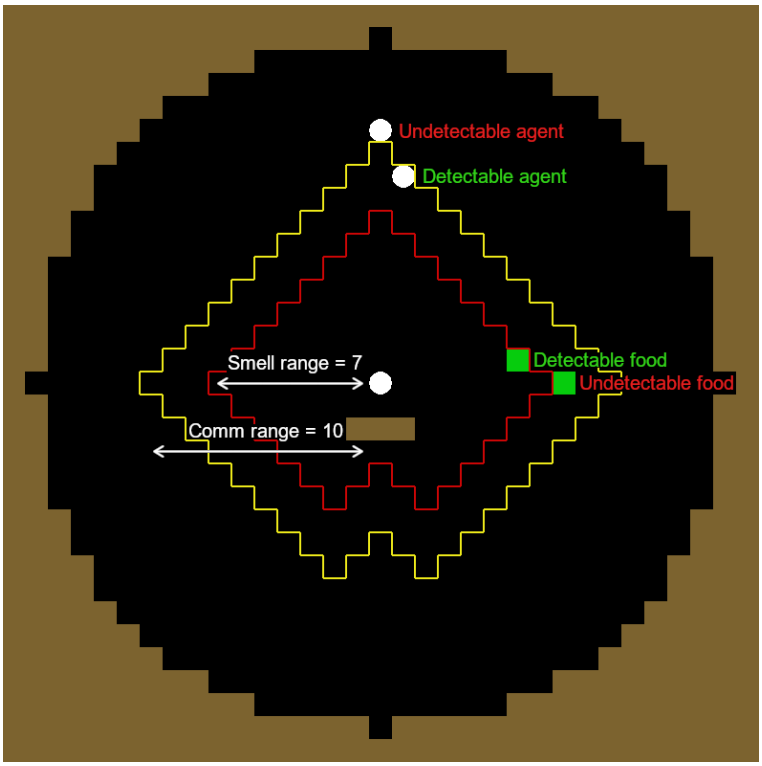


Figure 5: Illustration of the *Smell range*, shown in red, and the *Communication range*, shown in yellow, from the perspective of the agent in the centre. Two extra agents are shown in white, the one closest to the centre is detectable, while the farthest agent is not detectable. Two pieces of food are also shown, again with one being detectable and one being undetectable. The wall at the bottom of the agent illustrates that walls can also limit the detection range.

The distances between each pair of coordinates are calculated using a simple pathfinding algorithm. Before running a simulation, all distances are calculated and stored in array. The distances between coordinates can then be accessed without having to do additional calculations. Because our maps are currently static, i.e. the walls do not change over time, we only need to calculate the distances once per map. This is very beneficial because most of the results involve running the same simulation multiple times. The pathfinding algorithm also stores the shortest path between every pair of coordinates. This is used in tasks that make agents walk to a specified coordinate, for example, when traversing to a detected piece of food.

3 Base setup

In the previous section we illustrated the mechanics of our foraging simulation. In the simulation, a group of agents work together to find and gather food. Here, agents can communicate by sharing their internal memory. However, before a simulation can be run, three parameters need to be set:

- The map that is used. This determines the placement of walls and food and the value of each food source when gathered.
- The behaviour tree of the agents. All agents use the same behaviour tree that determines their actions at any moment in time.
- Other free parameters. These are the *smell range* R_{smell}, the *communication range* R_{comm} and the population size n_{pop}.

In this section we introduce the setup that is used throughout this work. We first discuss the map and the behaviour tree for the agents, which were both designed manually. We then illustrate some of the group's behaviour and explore how the behaviour depends on the remaining free parameters, like the population size. This serves as a baseline for the next section, where we explore how disruptions in the communication of the agents affect the behaviour of the group.

3.1 Map

The goal in designing a map is to create a situation where agents benefit from cooperation through communication. This can be encouraged in two ways: First, food should be non-trivial to find. This makes it so that agents can benefit from communication by sharing the position of discovered food sources. Secondly, we want to include multiple food sources of differing quality. This way, agents still need to consider new information even when a food source has already been found. Figure 6 shows the map named *Four Rooms*, which was designed to have these properties.

Here, agents are initially placed at the centre of an empty circular room. The agents can find food in each of the four smaller connected rooms. The room to

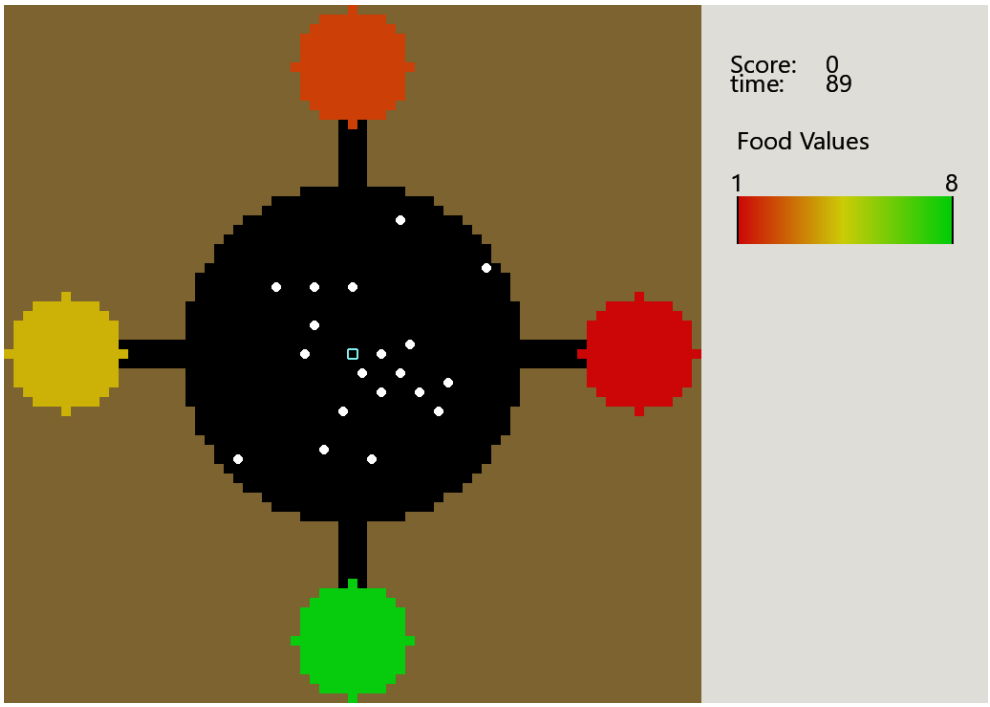


Figure 6: The map *Four Rooms*, which is used in all experiments.

the right is filled with food with a quality of 1. When this food is brought back to the centre, the score of the group is increased by 1. The other rooms contain food with increasingly higher values: The room on the top contains food with a value of 2, the room on the left has food with a value of 4 and the room on the bottom has food with a value of 8. The different food sources make it so that agents benefit from exploration. Even if a food source has already been found, it is possible that a better food source exists.

The exact shape of the map, and the values of the food, can be changed using a set of free parameters, shown in Table 1. This table shows the default values of these parameters used throughout the experiments. One notable parameter is ρ_{food}, which determines the number of food pieces per tile of food. The higher the value of this parameter, the longer it will take the agents to gather all the food placed on the map.

3.2 Behaviour tree

The behaviour of the agents is guided by a manually created behaviour tree, which is shown in Figure 7. This behaviour tree is an updated version of the example behaviour tree that was shown in Figure 4. Here agents again move

	R_1	R_2	d	w	f_1	f_2	f_3	f_4	ρ_{food}
Value	6-22	6	6	3	1	2	4	8	$10n_{pop}$

Table 1: The default parameters that determine the shape of the *Four Rooms* map. Here R_1 is the radius in the central room, R_2 is the radius of the outer rooms, d is the length of the tunnels between the rooms and w is the width of tunnels. f_1, f_2, f_3, f_4 determine the value of the food in the right, top, left and bottom room respectively. ρ_{food} determines the amount of food pieces per tile, which is set to 10 times the amount of agents.

randomly in order to find food. When food is found, the agents remember the location of the found food. When the food is delivered, agents move back towards the previously gathered food. There are two differences between this and the previous behaviour tree. First, agents try to more actively communicate the position of discovered food. They do this by standing still at the drop-off point for a set amount of time, making it easier for other agents to copy the coordinates of the discovered food. The second change is to the communication loop. When communicating with other agents, agents now compare the quality of their food source with the food source of other agents. Agents only copy information when the other food source is more efficient.

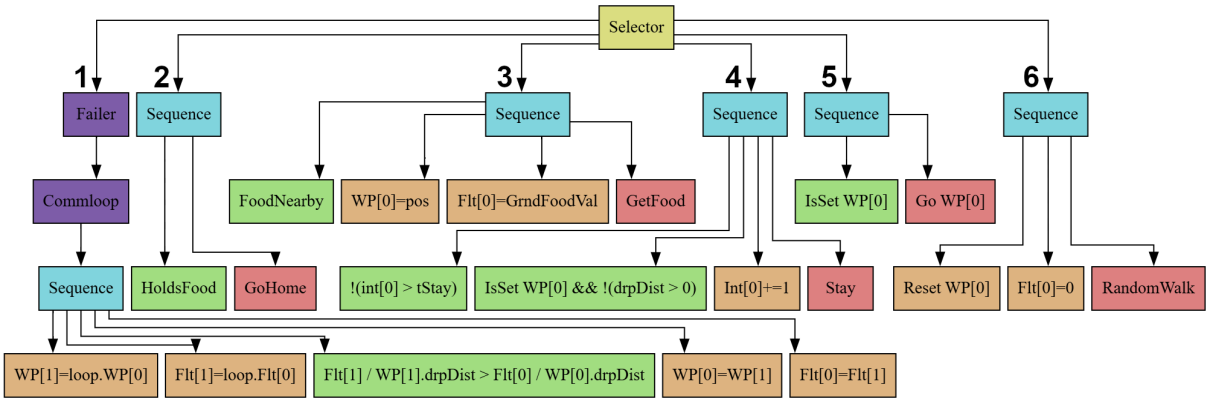


Figure 7: Behaviour tree that is used for the agents in the base setup. This behaviour allows agents to search for and gather food and share the location of detected food sources. Here, selectors are shown in yellow, sequences are shown in blue, conditions are shown in green, tasks are shown in red, memory nodes are shown in orange and decorators are shown in purple. Each sub-tree is labelled with a number and explained in more detail below.

1. The first sub-tree handles the communication between agents. The tree starts with a *Failer*, which ensures that the selector continues after executing the sub-tree. The communication loop then loops through all

agents within communication range. This loop retrieves the coordinate (*loop.WP[0]*) and food value (*loop.Flt[0]*) of the last food that the looped agent has found. The tree then checks if the food source of the other agent is more efficient than its current food source. Here the efficiency is the value of the food(*Flt[0]* and *Flt[1]*) divided by the distance to the drop-off point(*drpDist*). If it is more efficient, then the agent forgets its current food source and copies the information about the food source of the other agent.

2. If the agent is already holding food, then this sub-tree makes the agent deliver the food to the centre.

3. This sub-tree serves to make agents pick up nearby food. First it checks if the agent can detect any food using the condition *FoodNearby*. If this is the case then the agent stores it's current location and the value of the food in its internal memory. The agent then moves to pick up the closest food piece.

4. This sub-tree helps agents communicate information once food is found. If the agent is at the drop-off point(drpDist is 0) and has a waypoint set(*IsSet WP[0]*) then the agent will stand still. This allows other agents which have not yet found a food source to communicate with the agent and move to the food already found. The agent makes use of a counter *Int[0]* which increments every tick the agent stands still. When the agent has stood still for t_{Stay} seconds, the agent continues as normal.

5. If the agent has already found a food source, or has heard of a food source from another agent, this sub-tree makes the agent move towards the saved coordinates of the food.

6. This tree is executed only if all other sub-trees failed to fully execute. This makes the agent move in a random direction, effectively making the agent explore for new food. This sub-tree also resets the internal memory. This makes it so that if the agent moves towards a waypoint but does not find any food, the agent then does not return to that waypoint.

3.3 Setup parameter overview

For future reference, Table 2 gives an overview of all setup parameters that are experimented with. This also includes parameters not yet used in the base setup that become relevant when communication disruptions are introduced.

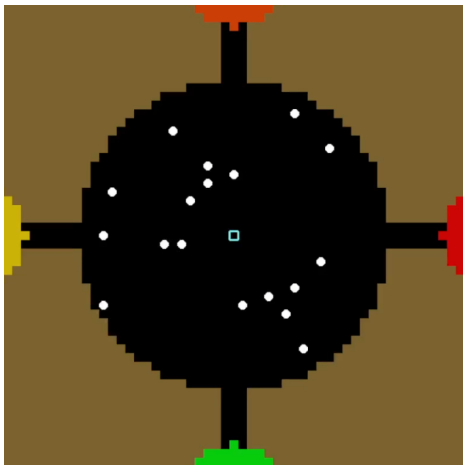
Symbol	Name	Description	Value
n_{pop}	Population size	Amount of agents placed in the simulation	1-100
R_{comm}	Communication range	Maximum range at which agents are able to share information	0-12
R_{smell}	Smell range	Maximum range at which agents can detect food	5
λ_{comm}	Average communication budget	Average amount of agents that can be communicated with each simulation step	0-∞
p_{distort}	Distortion chance	Probability that information is distorted when agents share information	0-1
λ_{distort}	Average distortion distance	Average distance a coordinate is displaced when a distortion occurs	1-8
R_1	Central room size	Radius of the room that agents are initially placed in	6-22
f_1, f_2, f_3, f_4	Food values	Score received for gathering food from each of the four food deposits	1-27

Table 2: Overview of the most relevant setup parameters that are set before running a simulation. Here Value indicates the range of values that is experimented with. Here λ_{comm}, p_{distort} and λ_{distort} determine the severity of communication disruptions and are not yet relevant for the base setup.

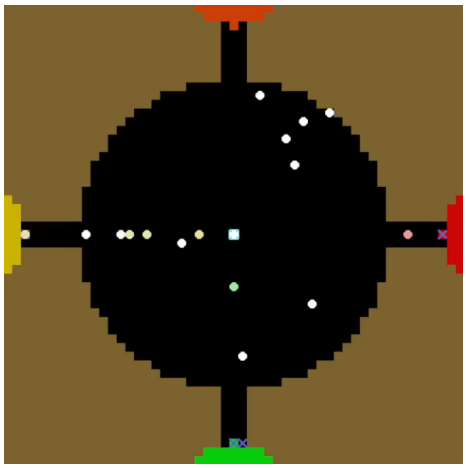
3.4 Group behaviour

In the previous sections, the structure of the map was defined, along with the behaviour of the individual agents. Now we explore the behaviour of the group as a whole in this setup. In this setup, the group generally goes through three distinct phases: The **Exploration Phase**, the **Communication Phase** and the **Exploitation Phase**. Initially, the group is focused on exploring the map and finding food. When food is found, agents proceed to communicate and decide on the best food source. Eventually the agents converge on a single food source. These phases are illustrated in more detail in Figure 8. Here the simulation is shown for three moments in time, one for each of the three different phases.

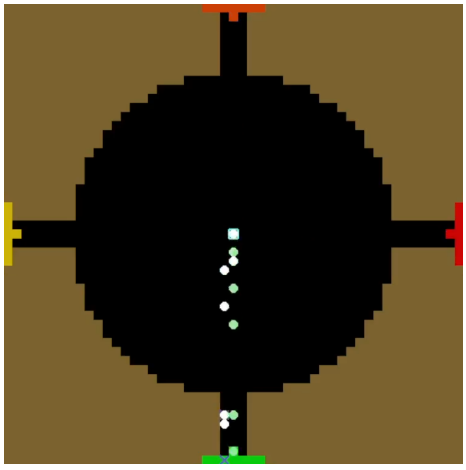
The different phases are further illustrated in Figure 9. This shows the score over time for 50 different runs using a population size of 20 and a communication range of 2. The different phases can be identified based on how much food is collected per tick. In the exploration phase, no food is being collected, as



(a) **Exploration Phase**: The agents are initially placed at the centre of the map indicated by the light blue basket. At this point none of the agents know about the location of any of the food, therefore they start taking steps in random directions. This causes the agents to spread out over the map. By chance, one or more agents will eventually walk close enough to a food source. The agent will then bring back the food and attempt to communicate the position to the other agents, which marks the end of the exploration phase.



(b) **Communication Phase**: The communication phase happens when one or more agents find a food source. These agents then stand still at the centre of the map for a set amount of time, allowing other agents to learn the position of the food from them. Agents can only communicate with other agents in their communication range, which has the effect that agents on the edge of the map keep exploring for more food. When communicating, agents prefer better food sources, which means that agents switch to more appealing food if able. Eventually, all the agents converge on a single food source.



(c) **Exploitation Phase**: After communicating, the agents eventually agree on gathering from a single food source. In this phase the agents walk towards the food source, pick up a piece of food and deposit it at the basket in the centre. This continues until the food source is depleted. Then, when agents encounter the now empty food source, they clear their internal memory. The cycle then repeats with the agents again entering the exploration phase.

Figure 8: An overview of the three broad phases that the group of agents can be in.

no food sources have yet been found. In the communication phase, agents gradually collect more food over time as more agents are told about discovered food sources. In the exploitation phase all agents gather from the same food source, causing a linear increase in score over time.

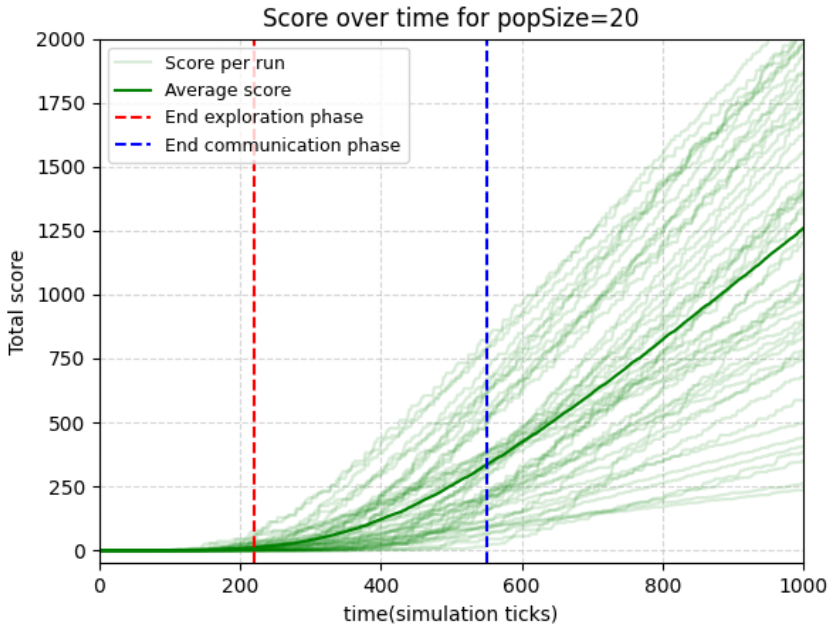


Figure 9: Score over time for a population of 20 agents with a communication range of 2. Here the red line indicates when most groups have entered the communication phase and the blue line indicates when most groups have entered the exploitation phase.

3.5 Evaluating group performance

In our simulation, the performance of a group of agents can be determined by measuring the score of the group after a set amount of time. However, this score is not very meaningful in a vacuum. This can be improved by normalizing the score, which is done by dividing the score by the maximum possible score that could be achieved. The normalized score then gives some indication of the efficiency of the group compared to the maximum theoretical efficiency. The maximum possible score at any point in time can be calculated using a simple algorithm, which is shown in algorithm 1. The idea behind this algorithm is straightforward: If the group is working at maximum efficiency then the group must always be gathering from the most efficient food source, as measured by the food value divided by walking distance. Thus the most efficient strategy is

to first gather all the food from the most efficient food source until it is depleted, and then move on to the second best food source. This continues until either the time is up or all food on the map is depleted. In a real simulation run, gathering from the best food source is not trivial, as agents have to first find and communicate the position of food sources, which means that the theoretical maximum will almost never be reached in practice.

Algorithm 1 Calculating maximum attainable score

Require: $t_{max} > 0$ ▷ Runtime of the simulation
Require: $n_{pop} > 0$ ▷ Amount of agents
Require: size($\vec{d}$)=n ▷ Distance of each food source to centre
Require: size($\vec{v}$)=n ▷ Value of each food source
Require: size($\vec{a}$)=n ▷ Amount of food pieces per food source
  for i is 0 to n **do** ▷ Calculate gathering speed for each food source
    $g_i \leftarrow v_i/(2d_i - 1)$
  end for

  $\vec{d} \leftarrow$ **sortby**$(\vec{d}, -\vec{g})$ ▷ Sort food by gathering speed, highest first
  $\vec{E} \leftarrow$ **sortby**$(\vec{v}, -\vec{g})$
  $\vec{a} \leftarrow$ **sortby**$(\vec{a}, -\vec{g})$
  $\vec{g} \leftarrow$ **sortby**$(\vec{g}, -\vec{g})$

  $t \leftarrow 0$
  $t_{new} \leftarrow 0$
  $s \leftarrow 0$
  for i is 0 to n **do** ▷ Loop through each food source
    $dt \leftarrow a_i \cdot v_i/(n_{pop} \cdot g_i)$ ▷ Time needed to deplete food source
    $t_{new} \leftarrow t + dt$
    if $t_{new} < t_{max}$ **then**
      $s \leftarrow s + a_i \cdot v_i$ ▷ Add gathered food to score
      $t \leftarrow t_{new}$
    else ▷ Stop when time is over, add partially gathered food
      $s \leftarrow s + a_i \cdot v_i \cdot (t_{max} - t)/(t_{new} - t)$
      break
    end if
  end for
  return *score*

One of the main advantages of normalizing by the maximum attainable score is that it allows us to compare the performance of the group in different scenarios. For example, if the food is placed further away from the centre of the map then the agents will naturally have to walk a larger distance to gather food, decreasing performance. This decrease is then not a result of the poor performance of the group but rather due to the less suitable environment.

Similarly, large groups of agents naturally gather more food than small groups, and maps with high value food yield better performance than maps with low value food. By dividing the score by the maximum attainable score, these factors are automatically taken into consideration.

3.6 Impact of population size

The effectiveness of communication can be understood by looking at the performance of the group for different population sizes. Figure 10 shows the impact of the population size on the normalized score of a group, for groups of communicating and non-communicating agents.

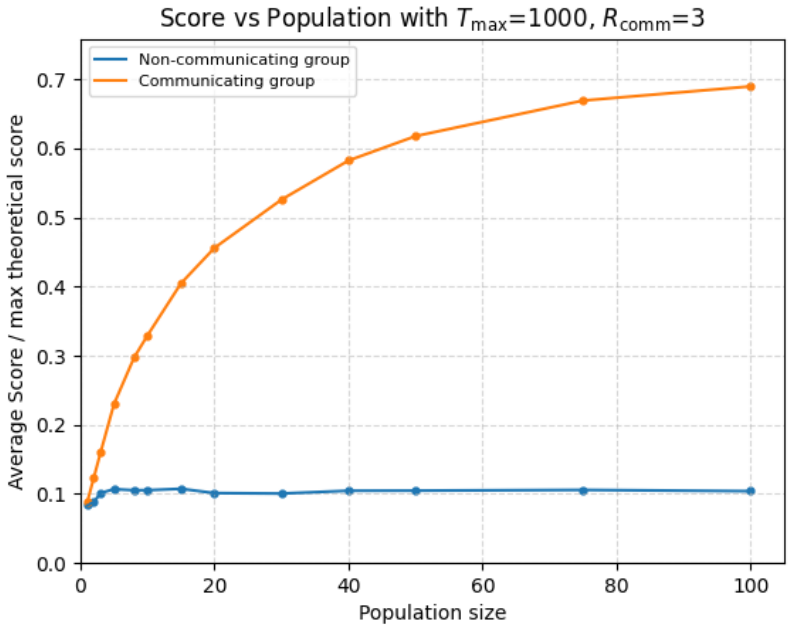


Figure 10: Normalized performance as a function of the population size. Here each data point represents the average of 200 runs. Each run uses a communication range $R_{\text{comm}} = 3$, a central room radius $R_1 = 18$ and a time limit of $T_{\text{max}} = 1000$.

Here the communicating groups use the behaviour tree shown in Figure 7, which allows agents to share information, while the non-communicating groups use the tree in Figure 3, which only allows agents to individually remember coordinates. Here we see that the normalized performance of the communicating group increases monotonically with the population size, going from about a 10% effectiveness at a population size of 1 to about a 70% effectiveness at a population size of 100. This shows that communication becomes more valuable as the

group increases in size. The reason for this is that when agent communicate, only one agent needs to find a food source for the group to discover its location. This makes it so that each additional agent decreases the average time spent looking for food, increasing efficiency. In comparison, the non-communicating group does not get any additional benefits from increasing in size. This is because each agent acts independently, so the agents do not benefit from their shared existence.

3.7 Impact of communication range

As with the population size, we now examine the impact of the communication range on the performance of the group. Here the communication range R_{comm} determines the maximum distance at which agents can still communicate. At short communication ranges, agents are only able to communicate with other close-by agents. If the communication range is set to the minimum of 0 then two agents can only communicate if they share the same position, which is possible since agents can occupy the same tile. For large communication ranges, agents will be able to communicate with agents at a large distance. For sufficiently large communication ranges, each pair of agents will always be able to communicate, even if they are positioned on opposite sides of the map. The influence of the communication range on the performance of the group is shown in Figure 11.

Here the green line shows the normalized score after 1000 time steps for groups using different communication ranges. Interestingly, we see that in order to maximize performance, the communication range should be neither too high nor too low. For this scenario, the optimal communication range seems to be somewhere between 3 and 6. This result of needing a balanced communication range can be explained by the trade-off between exploration and exploitation. If the communication range is short, the group spends a relatively long time in the communication phase. This is because agents that are exploring are less likely to hear about a food source from other agents, because agents need to get close in order to tell them. This results in agents spending more time exploring before the group converges on a food source. The food source that is eventually converged on is then more likely to have food of a high value. This advantage is indicated by the black dotted line in Figure 11. This shows the probability that the group finds the most valuable food source in the bottom room. We see that as the communication range increases, this probability goes towards 0.25, which indicates that the group instantly converges once the first food source found. However, time that is used to explore is time not spent gathering food, meaning fast convergence also has an upside. This is shown through the red dotted line in Figure 11, which shows the total food collected after 1000 steps, ignoring the value of the food. Here we see that groups with a high communication range gather more food than low communication range groups, as they spend less time exploring and thus more time gathering food. In conclusion, the communication range serves as a way to balance exploration and exploitation, making a balanced communication range preferable.

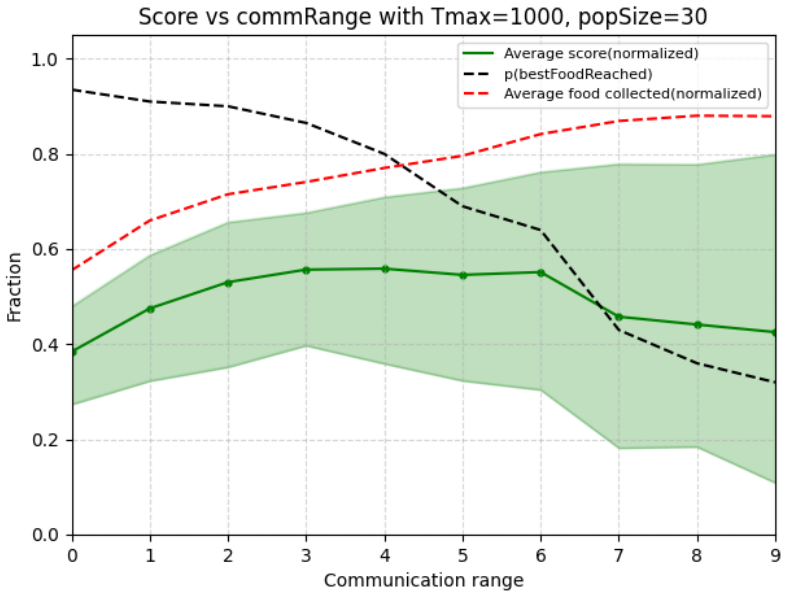


Figure 11: Normalized performance(green) as a function of communication range, with the shaded area showing one standard deviation. Each data point represents the average of 200 runs. The black dotted line shows the fraction of runs where the best food in the southern room was found. The red dotted line shows the amount of food collected ignoring the value of food. All runs use a population size of 30, a central room radius of $R_1 = 14$ and a time limit of $T_{\max} = 1000$.

3.8 Impact of map size

Previously it was discovered that the communication range can serve as a means to control the level of exploration and exploitation. This makes it so that choosing the right communication range is not trivial. We now examine how the optimal communication range, the range that maximizes the average performance of the group, changes in different circumstances. One factor that is likely to be important is the shape of the map. The map *Four Rooms* uses several parameters that define its shape, with the parameter R_1 determining the radius of the central room. Setting R_1 to a small value means food is relatively easy to find, as agents only have to explore a small area. At higher values of R_1, food becomes harder to find, as the food is placed further away from the centre. The impact of the central radius size on the communication range optimum is illustrated in Figure 12.

This shows the normalized score after 1000 steps for setups with different communication ranges and central radii. As expected, performance generally

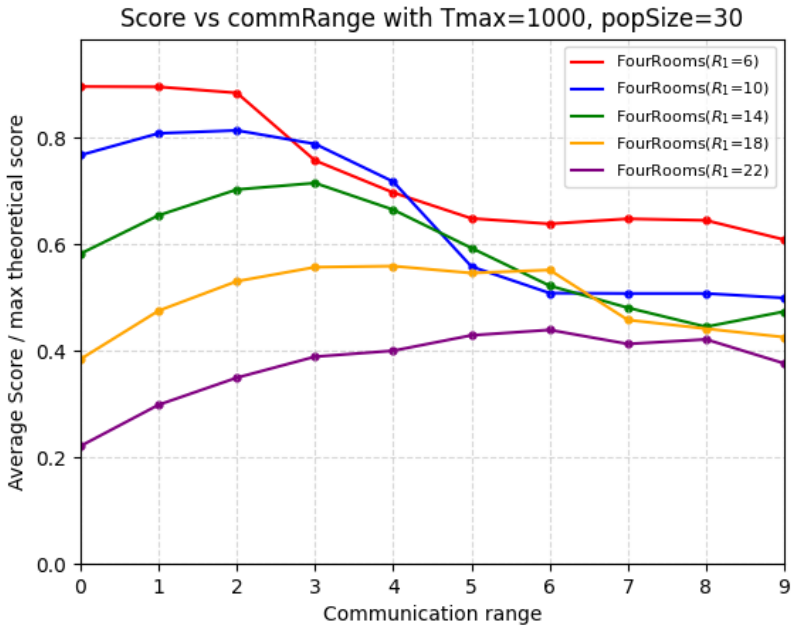


Figure 12: The normalized score after 1000 time steps for different communication ranges and map sizes, using a population size of 30 agents.

decreases as the central room radius increases. This happens because the group has to spend more time on average to find the food sources. We also see that the optimal communication range increases as the central room radius increases. For a small central radius of 6, the optimal communication range is somewhere between 1 and 3, while for a large room size of 22, the optimal communication range is somewhere between 5 and 7. In larger maps, agents are generally more spread out. This makes a larger communication range beneficial, as it allows the agents to still communicate efficiently.

3.9 Impact of food value distribution

Another factor that could affect the optimal communication range is the distribution of food sources. Here the idea is that if the simulation contains a variety of different food sources, agents need to explore longer in order to find the best food source. In case there is only a single food source, or all food sources are the same, then the group does not need much exploration as they can simply converge on the first food source found. In the simulation this distribution of food can be varied by setting the value of the four available food sources. Previously an exponentially increasing distribution was used, where each food source had a value two times higher than the previous food source. This relationship

is generalized in equation 1.

$$f_i = \alpha^{i-1}, \quad i = 1, 2, 3, 4, \quad \alpha \geq 1 \tag{1}$$

Here f_1 determines the food value in the right most room, f_2 for the top room, f_3 for the left room and f_4 for the bottom room. α determines the variance in food source values. If α is set to 1 then all food sources have the same value. As α gets bigger, the difference between best and worst food source becomes larger, making exploration more important. As previously α was set to 2 for all experiments, we now test the simulation again for other values of α in order to examine the influence of the distribution of food. Figure 13 shows the normalized performance as a function of communication range for different values of α.

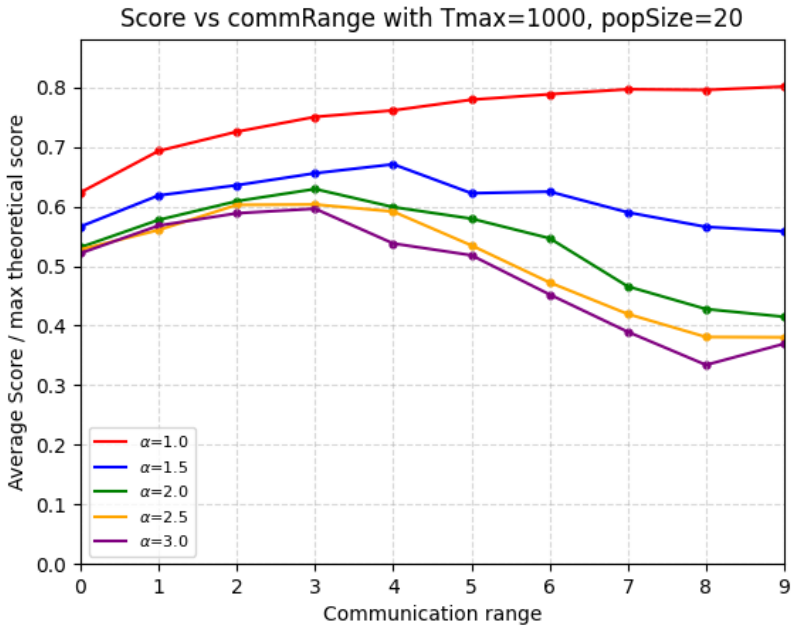


Figure 13: Normalized performance as a function of communication range, for several food distributions α. Here each data point represents the average of 200 runs. All runs use a population size of 20, a room size of $R_1 = 14$ and a time limit of $T_{\max} = 1000$.

As expected, we see that the optimal communication range generally decreases as α increases. Here a lower communication range leads to more exploration, which is more important as food becomes unevenly distributed. When α is set to 1, the performance increases monotonically with the communication range. Since all food has the same value in this case, it is optimal to instantly

converge on the first food found, which is only possible with a high enough communication range. When α is set to a higher value, the lowest value food is almost worthless compared to the highest value food. Here it becomes very important that the group converges on the best food source, which makes a relatively small communication range optimal.

3.10 Discussion

In the previous experiments we examined the behaviour of the group in an environment without intentional obstructions. Before moving on, we summarize some of the results and compare them to results found in other works. Firstly, we saw that the performance per agent grows as the population size increases. Because agents are able to communicate, each individual benefits from the presence of other agents, as they share information on the position of food. This shows that the group indeed benefits from cooperation under ordinary circumstances.

Secondly, we saw that choosing the optimal maximum communication range is not trivial. When the communication range is set too high, agents converge too fast on discovered food sources. When the communication range is too low, information about discovered food sources travels slowly, making it so agents spend an unnecessary amount of time exploring. A similar dilemma can be found when using the particle swarm optimization(PSO)[16] algorithm. PSO is an optimization algorithm inspired by the movement of groups of animals such as flocks of birds and schools of fish. PSO works by having a group of agents semi-independently search the search space. These agents are affected by two components: an individual component and a social component. Here the social component pulls agents towards the best found solution of nearby agents, which is very similar to how agents communicate in our simulation about memorized food sources. In PSO, a major challenge is adjusting the relative strength of the individual and the social component. If the strength of the social component is too high then the group converges too quickly on one solution. On the other hand, if the individual component is too strong then agents lose the benefit of cooperation. Here the relative strength of the social component can be compared to the communication range in our simulation, as both need to be optimized to balance individuality and conformity.

This problem of choosing the right communication range can also be compared to the multi-armed bandit problem[13] in probability theory. In this problem, a decision maker iteratively chooses between multiple actions, with each action having an unknown reward distribution. Here the trade-off is again between exploring for potentially better rewards and sticking to a known good solution. The optimal balance between exploration and exploitation depends on the reward distribution of the different actions. This is also the case in our simulation, where the optimal communication range, and thus the amount of exploration, is determined by the distribution of food. If all food is equal in

value then the optimal strategy is to immediately converge on the first food found. However, if some food is vastly more valuable than other food, then exploration becomes more important as finding the best food source becomes essential.

4 Setup with impeded communication

Previously, we explored the performance of the agents in an ideal environment. Here the group of agents was able to effectively find and gather food. In this section we now explore what happens when communication between agents is impeded. We first introduce the way communication is impeded. Secondly, we study under what circumstances the loss in communication is most detrimental to the performance of the group. Lastly, we look at some measures that can be taken to prevent the negative effects of communication disruptions, and study in what circumstances these measures are effective.

4.1 Communication constraints

In our new setup, communication can be obstructed in two ways: Limiting the amount of communication and limiting the accuracy of communication. In our base setup, agents are able to communicate with all agents within their communication range each tick. In our new setup, this amount is limited, making it so that agents can only communicate with a subset of the nearby agents. Each tick each agent is given a communication budget n_{comm}. When running a communication loop, agents are then only able to communicate with n_{comm} nearby agents, instead of being able to communicate with all nearby agents. For example, an agent might be surrounded by 6 other agents and have a communication budget n_{comm} of 3. The agent can then only communicate with 3 out of the 6 agents. The size of the communication budget is randomly determined each tick depending on the parameter λ_{comm}. Here n_{comm} is take from a Poisson distribution, as in equation 2.

$$n_{comm} \sim \text{Poisson}(\lambda_{comm}) \tag{2}$$

In case λ_{comm} is set to 0, we obtain a situation where no communication is possible, as the communication budget is always 0 for all agents. In the case of $\lambda_{comm} \to \infty$, we obtain the same situation as in our base setup, giving agents an unlimited amount of communication. Thus the parameter λ_{comm} allows us to scale the amount of communication between no communication and unlimited communication.

The second communication obstruction is the possibility of miscommunication. In the base setup agents can read the internal memory of other agents with perfect accuracy. In the new setup, each time an agent communicates with another agent it has a $p_{distort}$ chance of miscommunication. This causes the agent to 'mishear' the other agent, which perturbs the information to some extend.

This perturbation affects how the stored coordinates are received. When a coordinate is miscommunicated, the coordinate is displaced by the distortion size $n_{distort}$. The new coordinate, heard by the receiving agent, is then obtained by taking $n_{distort}$ random steps. The size of the distortion is again sampled from a Poisson distribution, as in Figure 3. Here $\lambda_{distort}$ is a parameter that determines the average distortion size.

$$n_{perturb} \sim \text{Poisson}(\lambda_{distort}) \tag{3}$$

Figure 14 gives an example to illustrate the two communication constraints.

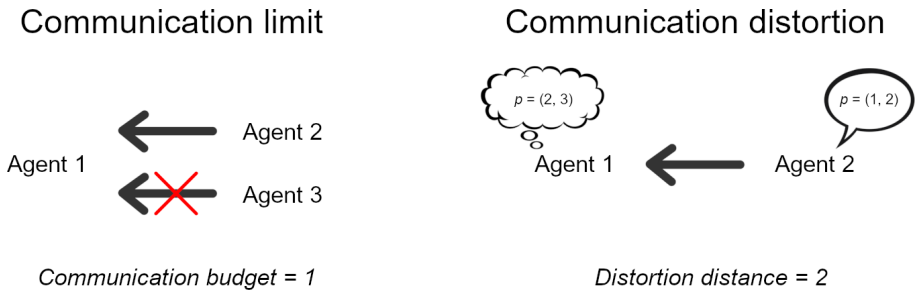


Figure 14: Two ways in which communication is obstructed between agents. The figure on the left shows an example where communication is limited by a communication budget. In this case, the communication budget of 1, sampled from a Poisson distribution, prevents the agent from communicating with more than one other agent per tick. The figure on the right depicts an example of communication distortion. Here agent 2 attempts to communicate the position (1, 2) to agent 1. However, this information is miscommunicated, displacing the coordinate by the distortion distance of two steps. As a result, agent 1 believes that agent 2 has found food at position (2, 3) instead of position (1, 2).

4.2 Impact of communication limit

In order to study the effect of communication disruption we first introduce the the communication limit to the simulation without including communication distortions. Here the parameter λ_{comm} determines the average communication budget given to the agents each tick. This parameter thus limits how much agents can communicate, with agents being completely unable to communicate when λ_{comm} is set to 0. We now test the influence of λ_{comm} on the performance of the group. When experimenting with this, it was discovered that the effect of λ_{comm} is similar to the effect of the communication range, which is shown in Figure 15.

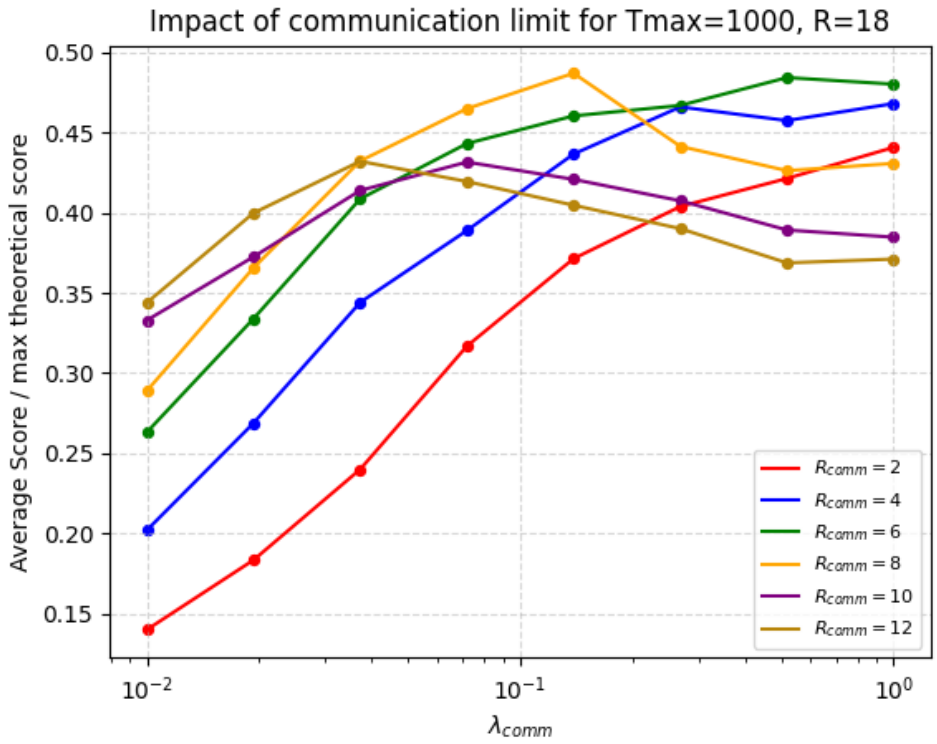


Figure 15: Normalized performance as a function of the average communication budget λ_{comm} for several communication ranges. Here the Average communication budget ranges logarithmically from 0.01 to 1. Each data point represents the average of 400 runs. All runs use a population size of 20, a room size of $R_1 = 18$ and a time limit of $T_{\max} = 1000$.

This shows the normalized performance as a function of the average communication budget λ_{comm} for several different communication ranges. As with the communication range, we see that a balanced communication budget is optimal. For example, when the communication range is set to 8, the performance is optimal for a λ_{comm} slightly above 0.1, where lower or higher values result in lower performance. This can again be explained as a balancing between exploitation and exploration. When the communication budget is too low, agents take an unnecessary long time to communicate the position of discovered food. However, if the communication budget is too high, the group converges too fast on discovered food even if better food is available. Since the communication range and the average communication budget serve a similar purpose, they can be used to offset one another. For example, if the communication budget is too low, this can be compensated by increasing the communication range and vice versa. This is reflected in Figure 15. When the communication budget is set

to 0.01, the best performing communication range is 12. In contrast, when the communication budget is set to 1, the best performing communication range is 6. In conclusion, limiting the communication between agents is not always detrimental and can sometimes be beneficial for the performance of the group.

4.3 Impact of communication distortion

We now examine the impact of distorting the communication between agents. Here communication distortion is governed through two parameters: the distortion chance $p_{distort}$ and the mean distortion size $\lambda_{distort}$. This allows us to scale both the frequency of distortions and the size of distortions independently. When adding communication distortion to the base setup, it quickly becomes apparent that the distortions can drastically reduce the performance of the group. An example of this is shown in Figure 16, which shows the score over time for a setup with a moderate amount of distortion. Here we see that in most runs the agents eventually stop gathering food altogether. This effect sometimes ends, where agents start to gather food again, only for it to return at a later time.

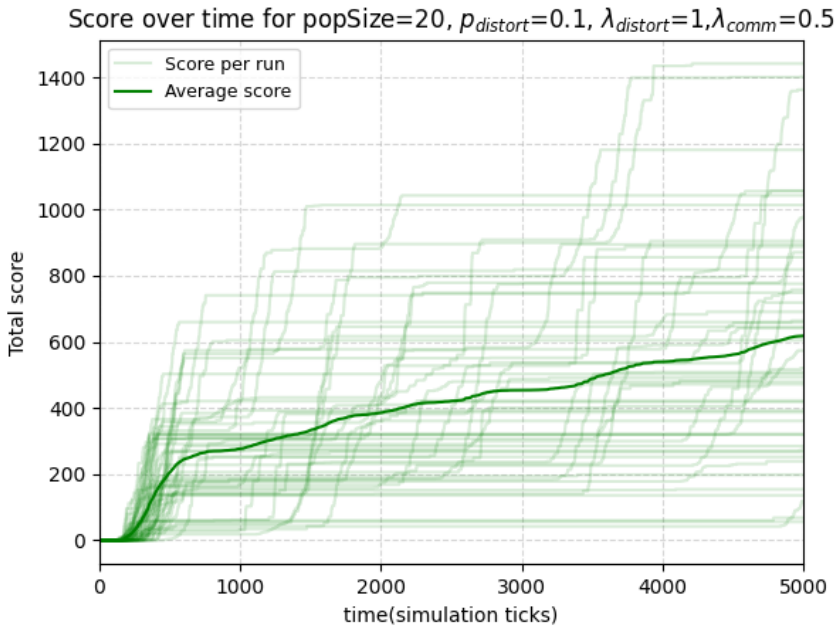


Figure 16: 50 runs of agents placed in the *Four Rooms* map, now with communication distortion enabled. Sometimes the score of individual runs abruptly stops increasing, which indicates no food is being collected. This is not the case in figure 9, where communication is not being distorted.

This phenomenon is better observed by looking at the simulation in real time. When agents are gathering food and communication distortion is enabled, agents sometimes start swarming around a coordinate close to the entrance of a food source. Here agents form a circular blob that pulls in other nearby agents. This swarm then gradually moves towards the centre of the map, where it stays.

The reason for this phenomenon can be explained by the way that agents communicate. When an agent is told about another food source, it compares this new food source to the food source it has currently memorized. Agents prefer food sources that have a high value and are close to the centre, and will only switch food sources if the new source is better in this regard. Due to the communication distortion, agents sometimes mishear that the found food source is closer to the centre than it actually is. Because this non-existent food source would be better than the real food source, the new coordinate is rapidly spread through the group. Eventually an agent again mishears another agent, and comes to believe there is an even better food source closer to the centre. The new coordinate is then again spread through the population, with this process repeating until the centre is reached. From now on we call this phenomenon a **phantom swarm**, as agents are swarming around a non-existent food source. The formation of a phantom swarm is shown visually in Figure 17, which shows three moments in time during phantom swarm formation.

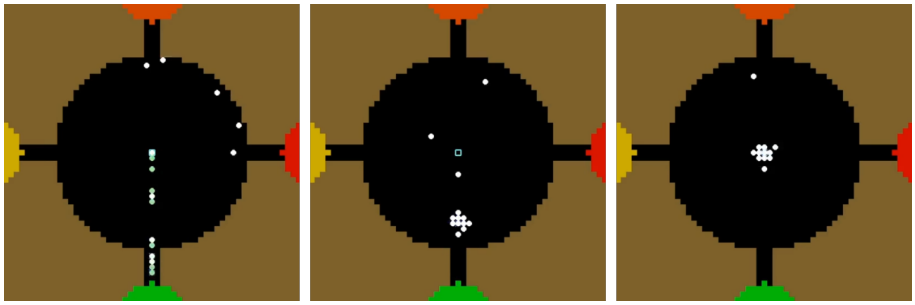
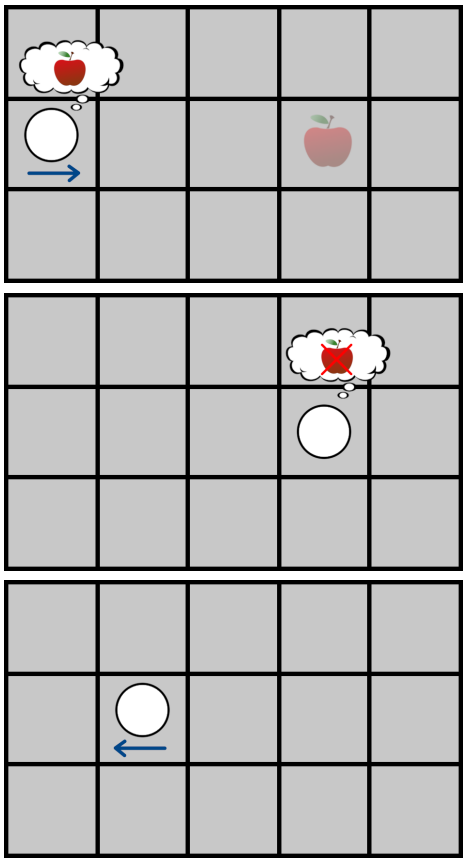


Figure 17: The formation of a phantom swarm over time. Initially agents are effectively gathering food from a food source, as shown in the left image. An agent then mishears another agent, leading it to believe there is food in front of the entrance to the food, shown in the middle image. This drift in coordinates continues until agents swarm around the centre of the map, as shown in the right image.

Agents do have a defence mechanism against the formation of phantom swarms: When an agent finds no food at a stored coordinate, it empties its internal memory. This was originally made so that when a food source is depleted, agents continue searching for new food sources. In the case of a phantom swarm, agents also delete their internal memory when they discover that their memorized coordinate does not contain any food. This leaves the question of

why phantom swarms can exist for a prolonged amount of time. The reason is that even though agents are constantly clearing the fake signal, the signal is also kept alive because it is transmitted to other agents. This is shown in more detail in Figure 18, where an overview of how the false signal is maintained is shown.



(a) In the initial state, the agent is walking towards the non-existent food source. This could be because it misheard another agent talking about a real food source, or because it is told about a false food source by an agent in the phantom swarm. In this state, agents also communicate the false information to other agents.

(b) When the agent reaches its destination, it discovers that the specified food does not exist. It then clears its internal memory, removing the fake coordinate.

(c) Because the agent has cleared it's memory, it now again starts exploring for new food, generally moving away from the fake food. However, if the agent communicates with another agent in state a, it will again start moving back to the false coordinate.

Figure 18: While in a phantom swarm, agents generally exist in one of three states: moving towards the fake food source, standing on the fake food source and moving away from the fake food source. The reason that the phantom swarm does not dissipate is that agents continuously move through these states. Agents in state 1 can then transmit the false information to agents in state 3, which continues the loop.

After some experimentation, it was found that stopping the formation of phantom swarms is not entirely trivial. Furthermore the formation of phantom swarms also seems to depend highly on the simulation parameters that are being used. One example of this is that phantom swarms seem to be more common for groups with a higher amount of agents. The remainder of this work focuses on

better understanding this phenomenon. We first examine how phantom swarms relate to previously discovered phenomena. Secondly, we examine under what circumstances phantom swarms are most likely to occur. Lastly, we propose some measures that can be taken to prevent phantom swarms, and test the effectiveness of these measures.

4.4 Comparison to other phenomena

In our simulation, phantom swarms can significantly affect the performance of the group. Here, the formation of phantom swarms is an emergent effect that can happen due to the way that communication is set up. This effect, however, is not restricted to our environment, and can happen in other scenarios as well, as long as the following conditions are met:

- There is a group of communicating entities that have the ability to share information.
- The entities prefer to accept some pieces of information over other information.
- Untrue information is generated in some way.

Phantom swarm like phenomenon are still possible even if the exact mechanisms behind these conditions are different. For example, if untrue information is generated in a different way, perhaps by being introduced by adversarial agents, then phantom swarms can still occur. The agents that form the group also do not have to be simulated entities; the group could also consist of humans, animals or robots.

One example of a real-world phenomenon comparable to phantom swarms is the spread of misinformation on social media. On social media, individuals can have a bias on what kind of content they accept and share. For example, in Weeks, 2015[18], it was found that individuals are more likely to believe dubious information that aligns with their political beliefs than information that runs counter to their beliefs. Another example is the sensationalizing of news, where more outlandish claims might be more likely to be shared than relatively simple claims. Because of this bias, misinformation is able to spread through the population of users. This spread of misinformation can be likened to the spread of a virus, which is done for example in Rubin et al., 2019[10], which proposed a conceptual model for modelling "fake news" as an epidemic. In Shin et al., 2018[12], the spread of online rumours was studied, and found that false political rumours tend to resurface multiple times, often turning more extreme over time. Here misinformation on social media and the formation of phantom swarms is similar, as in both cases false information is spread through a population due to the way agents accept and communicate information.

Another phenomenon that is similar to phantom swarms is the formation of ant mills, which were mentioned earlier in the introduction. When a group of army ants gets separated from the colony, it can happen that they start following each other's pheromone track, creating a continuous circle. Similarly to in our simulation, the ants are following information that does not reflect reality. When ants follow the circular track, they expect the track to lead them to some endpoint, for example food. However, because ants reinforce the pheromone trail as they walk, they do not manage to filter out the false information. As with our simulation, the communication system of the group becomes a hindrance, as the agents are not able to deal with incorrect data. Interestingly, ant mills have also been observed to happen in swarm-based algorithms[3]. This further shows that communication systems have to be carefully considered when implemented in multi-agent systems.

4.5 Detecting phantom swarms

As phantom swarms can halt the performance of the group, we want to explore under what circumstances phantom swarms occur. To do this, a method is needed to automatically detect when a phantom swarm is occurring. This allows us to run many simulations without having to manually check for phantom swarms. For the current setup, phantom swarms can be detected using a simple heuristic. This is based on two observations:

- When a phantom swarm is happening, almost no food is being collected.
- When a phantom swarm is happening, agents believe there is food close to the centre of the map when there is not.

When both conditions are met, the group is considered to be in a phantom swarm. For the first condition, the amount of food that was collected in the last t_{sum} steps is determined, where t_{sum} is a parameter that can be tuned. If this value is below a predefined minimum then the condition is met, which is shown in equation 4. Here $n_{\text{food}}(t)$ is the cumulative amount of food collected up to time t, R_{walk} is the minimum distance that agents need to walk to gather food and n_{pop} is the number of agents in the population. Here we include the walking distance and population size in order to normalize the amount of collected food. This is normalized so that if $c_1(t) = 1$ all agents are gathering at maximum efficiency.

$$c_1(t) = R_{\text{walk}} \frac{n_{\text{food}}(t) - n_{\text{food}}(t - t_{\text{sum}})}{t_{\text{sum}} \cdot n_{\text{pop}}} \tag{4}$$

The walking distance R_{walk} is constant for the *Four Rooms* map, as all food sources are at the same distance. Here the walking distance is calculated as:

$$R_{\text{walk}} = 2 \cdot (R_1 + d) + 1 \tag{5}$$

For the second condition the average distance of memorized coordinates to the centre is determined. If this distance is smaller than a predefined value, then the condition is met. The exact calculation is shown in equation 6.

$$c_2(t) = \frac{1}{R_{\text{walk}}} \frac{\sum_{i=0}^{n_{\text{pop}}} m_i(t) R_{mem,i}(t)}{\sum_{i=0}^{n_{\text{pop}}} m_i(t)} \tag{6}$$

Here $R_{mem,i}(t)$ is the distance to the centre for the memorized coordinate of agent i. $m_i(t)$ indicates if agent i is storing a coordinate, where agents without a memorized coordinate are ignored. The walking distance is again included to normalize the distance. The total condition for phantom swarm detection is shown in equation 7. Here $c_{1,\min}$ and $c_{2,\min}$ are parameters that define the sensitivity of the condition. The rightmost condition also checks that at least one agent has stored a coordinate.

$$c_1(t) < c_{1,\min}, \quad c_2(t) < c_{2,\min}, \quad \sum_{i=0}^{n_{\text{pop}}} m_i > 0 \tag{7}$$

In order to test this heuristic, the simulator now shows in real-time whether this condition is met. In this way, it can be checked if the condition accurately predicts the presence of phantom swarms. This is used to choose the values for $c_{1,\min}$ and $c_{2,\min}$. By doing this, it was discovered that the second condition, which is based on the position of perceived food, was a sufficient indicator on its own. The first condition requires measuring the amount of food collected over some time period, making it unresponsive in practice. We therefore set $c_{1,\min} = 1$ and $c_{2,\min} = 0.75$, meaning the first condition is always met. The heuristic is illustrated in Figure 19, where we show the detection of phantom swarms based on this condition for a single run.

4.6 Measuring swarm frequency

The previously defined heuristic allows us to measure when a phantom swarm is occurring. Our goal now is to study how different parameters affect the prevalence of phantom swarms. The prevalence of phantom swarms can be described by two measurements:

- The likelihood of the group forming a phantom swarm.
- The average duration of a phantom swarm.

In order to measure these factors, we measure the duration of periods with and without phantom swarms. For example, in Figure 19, a swarm starts at around step 700 and ends at around step 2800, giving a duration of roughly 2100 ticks. Here the duration of each gap between the start and end of a phantom swarm is gathered, as well as the duration between start and end of each period without a phantom swarm. Gaps with a length smaller than 10 are ignored, which prevents short random disturbances from splitting up the periods. A simple way to calculate the average phantom swarm duration is to take the

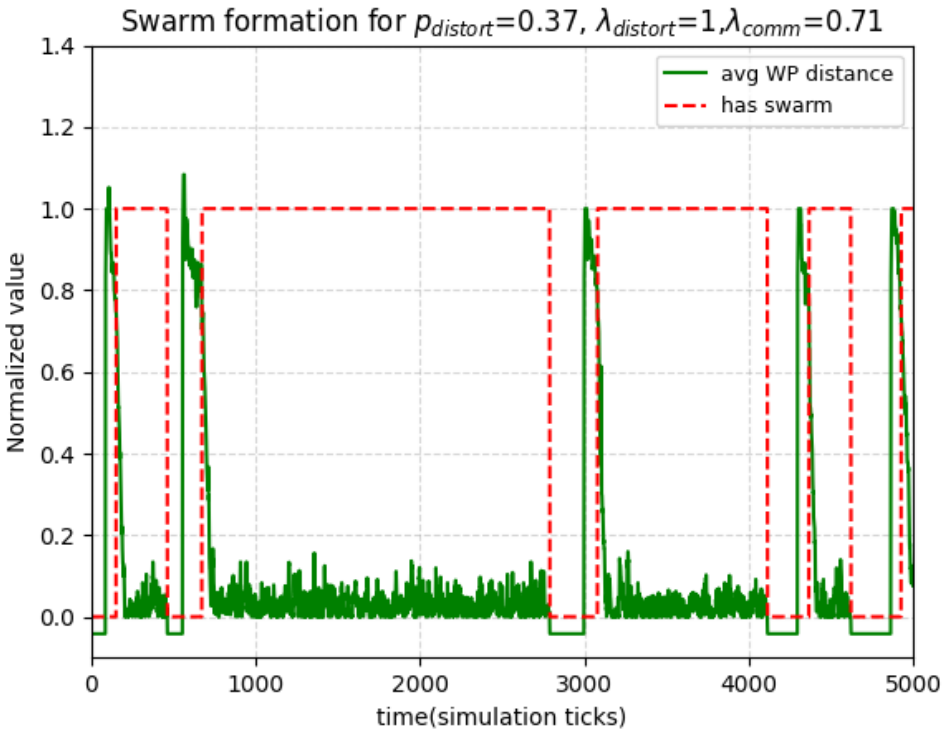


Figure 19: The average distance between waypoints and the centre of the map for a single run(green) and the swarm detection heuristic(red). Here the red dotted line is 1 when a phantom swarm is detected and 0 when a swarm is not detected. A negative average waypoint distance indicates no waypoints are set.

average of the duration of all recorded swarms. However, this has one flaw, which is that when the simulation ends after 5000 ticks, it is unknown how much longer a possible phantom swarm at that time would last. For example, in Figure 19, a swarm occurs just before the end of the simulation, which lasts an undetermined amount of time. By using a statistical model, finished periods, periods that end during the simulation, can be separated from these unfinished periods. If it is assumed that the formation of phantom swarms is memoryless, i.e. that the probability of the group forming a swarm is the same each step, then a geometric probability distribution can be used. Under this assumption, the system is defined by two probabilities. First, in the case that the group is not currently in a swarm, the swarm formation probability p_{swarm} determines the chance that a swarm is formed each tick. In the case that the group is in a swarm, the dissipation probability $p_{\text{dissipate}}$ determines the chance that the group forgets the false coordinate and returns to normal. The values for p_{swarm} and $p_{\text{dissipate}}$

can be found by maximizing the posterior probability. The posterior probability can be constructed using the probability mass function(pmf) and cumulative distribution function(cdf) of a geometric distribution, shown in equation 8. Here the pmf can be interpreted as the probability of phantom swarm forming/ending after t simulation ticks. The cdf can be interpreted as the probability that a swarm forms/ends before t simulation step.

$$\text{pmf}(t, p') = (1-p')^t p', \quad \text{cdf}(t, p') = 1 - (1-p')^{t+1} \tag{8}$$

The likelihood function describes the likelihood of a set of a set of phantom swarm/non-phantom swarm durations $\vec{t}$, as shown in Figure 9. Here f_i records whether phantom swarm i is finished or unfinished, with $f_i = 0$ meaning the swarm is finished and $f_i = 1$ meaning the swarm is unfinished, i.e. ends after the simulation ends. This determines whether to use the cdf, in the case of $f_i = 1$, or the pmf, in the case of $f_i = 0$.

$$p(\vec{t}, p') = \prod_{i=0}^{n} f_i(1 - \text{cdf}(t_i, p')) + (1 - f_i)\text{pmf}(t_i, p') \tag{9}$$

We can then estimate the probability $\hat{p}$ given the data by maximizing the logarithm of the likelihood function, which is equivalent to maximizing the posterior probability, as shown in Figure 10.

$$\hat{p}(\vec{(t)}) = \arg\max_{p'} \sum_{i=0}^{n} f_i(1 - \text{cdf}(t_i, p')) + (1 - f_i)\text{pmf}(t_i, p') \tag{10}$$

This allows us to estimate the probabilities p_{swarm} and $p_{\text{dissipate}}$. Here p_{swarm}, the probability of a swarm forming, depends on the duration of periods without a swarm, with shorter periods indicating a higher probability of formation. Thus in this case $\vec{t}$ is set to the durations of all swarm-less period. The same is true for $p_{\text{dissipate}}$, the probability of swarms dissolving, but now the set of durations with a swarm is used for $\vec{t}$.

With this setup, it is now possible to quantify the presence of phantom swarms. Here a high value of p_{swarm} indicates that swarms are likely to form. A low value of $p_{\text{dissipate}}$ on the other hand, indicates that swarms are likely to last a long time once formed. Under the assumption that swarm formation/dissipation follows a geometric distribution, we can calculate the expected duration of periods with or without a swarm, as shown in equation 11.

$$t_{\text{swarm}} = \frac{1 - p_{\text{dissipate}}}{p_{\text{dissipate}}}, \quad t_{\text{no swarm}} = \frac{1 - p_{\text{swarm}}}{p_{\text{swarm}}} \tag{11}$$

Here t_{swarm} is the average duration of a swarm and $t_{\text{no swarm}}$ is the average duration of periods between swarms. Because the presence of phantom swarms is unwanted, it is preferable to have both a low value for t_{swarm} and a high value for $t_{\text{no swarm}}$. Both metrics can be combined to obtain a single estimator that takes both values into account, as shown in equation 12.

$$\phi_{\text{swarm}} = \frac{t_{\text{swarm}}}{t_{\text{swarm}} + t_{\text{no swarm}}} \tag{12}$$

Here ϕ_{swarm} is the swarm rate, the fraction of time where a phantom swarm is present. A high swarm rate then indicates that at any point in time there is a high chance of the group being in a phantom swarm. By substituting equation 11, this equation can be rewritten in terms of the swarm formation/dissipation probabilities, as shown in equation 13.

$$\phi_{\text{swarm}} = \frac{p_{\text{swarm}}(1 - p_{\text{dissipate}})}{p_{\text{swarm}}(1 - p_{\text{dissipate}}) + p_{\text{dissipate}}(1 - p_{\text{swarm}})} \tag{13}$$

This circumvents division by zero when either p_{swarm} or $p_{\text{dissipate}}$ is zero.

4.7 Results

4.7.1 Population size dependence

The estimators p_{swarm}, $p_{\text{dissipate}}$ and ϕ_{swarm} can be used to measure the formation rate, dissipation rate and overall presence of phantom swarms. This is now used to examine how the different parameters of our setup influence the manifestation of this phenomenon. We first analyse how the population size, the average communication budget λ_{comm} and the distortion chance p_{distort} affect swarm formation. Here 50 simulations are run for each combination of 4 different population sizes, 8 values for λ_{comm} and 8 values for p_{distort}, resulting in a total of 12800 simulation runs. The results are shown in Figure 20. This shows the swarm formation probability p_{swarm}, the swarm dissipation probability $p_{\text{dissipate}}$ and the swarm rate ϕ_{swarm} as a function of population size, λ_{comm} and p_{distort}. Here we see a clear boundary between systems suffering or not suffering from phantom swarms, most clearly seen by looking at the swarm rate. The position of this boundary seems to depend on all three of the considered parameters. Firstly, we see that limiting communication by lowering the average communication budget can prevent swarms from persisting. One explanation is that when communication is limited, agents can disprove false signals faster than the signal can spread, causing the swarms to dissipate. We also see that phantom swarms are more likely when information has a higher chance of being distorted. Here we see that even with a relatively small distortion chance the formation probability is high enough for swarms to form. However, we also see that with a small distortion chance swarms quickly dissipate. Finally, we see that the population size also plays a significant role. At a population size of 10, the swarm rate does not come close to 1 with any parameters, even with guaranteed information distortion. At a population size of 25, phantom swarms occur for most circumstances. The reason for this is likely the same as with the communication budget. When more agents are present, more information is exchanged on average each step. This makes it so that swarms do not easily dissipate, as false information is preserved by transferring between agents.

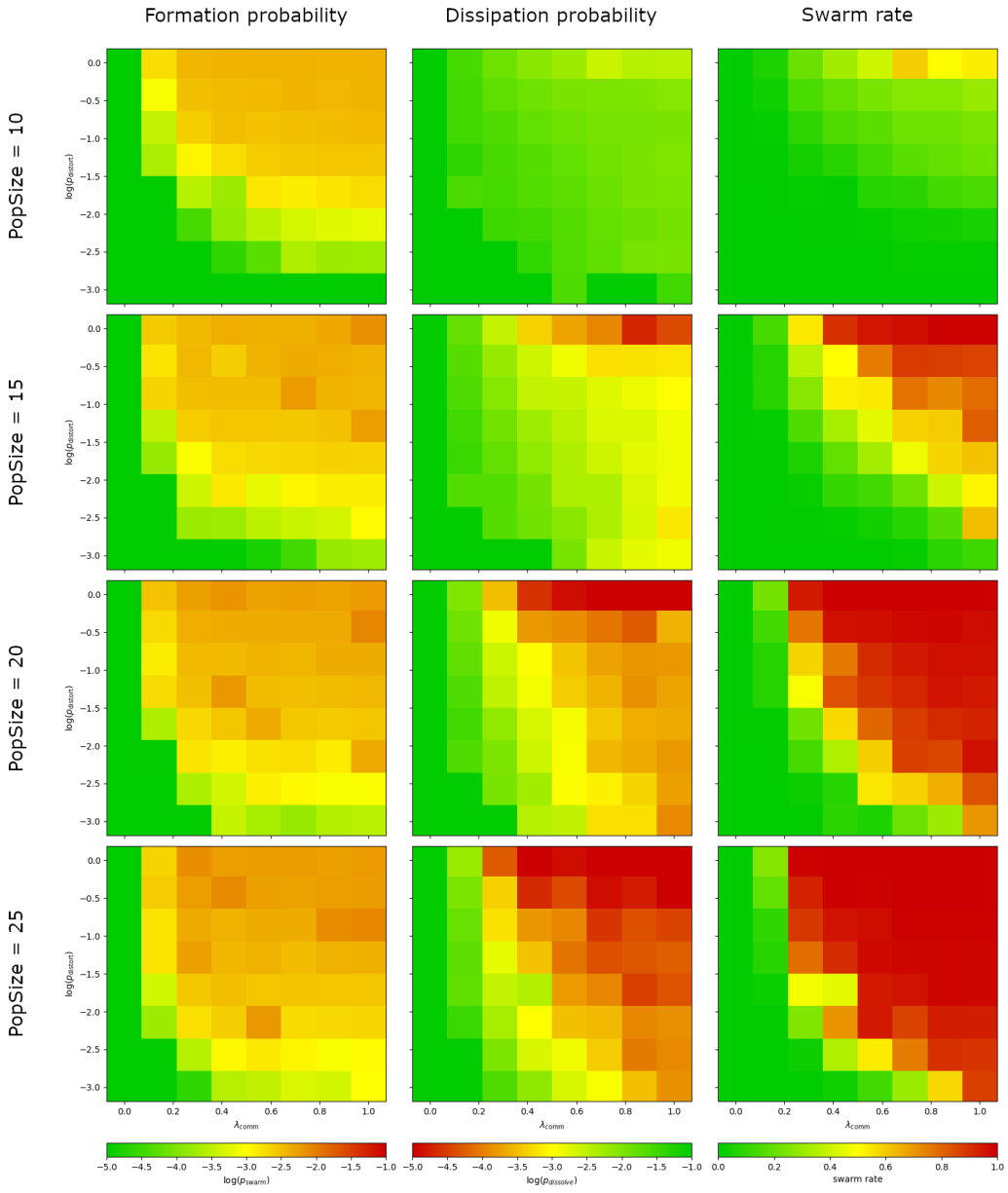


Figure 20: Heat-maps for the swarm formation probability, swarm dissipation probability and swarm rate for different simulation parameters. Each data point represents the average over 50 simulation runs. Here the x-axis represents the average communication budget ranging linearly from 0 to 1, which determines how much agents can communicate each tick. The y-axis shows the distortion chance ranging logarithmically from 0.1% to 100%. All simulations use an average distortion size of $\lambda_{\text{distort}} = 1$, a communication range of $R_{\text{comm}} = 6$ and a room radius of $R_1 = 18$.

4.7.2 Distortion size dependence

We now examine the effect of the average distortion size λ_{distort} on the formation of phantom swarms. Here the average distortion size determines how much information is distorted when a miscommunication happens. The higher the distortion size, the higher the distance between the initially communicated coordinate and the received distorted coordinate. Figure 21 again shows the results of 12800 simulations, now for 4 different values of λ_{distort}. Here we see that the distortion size is much less influential than the population size. The dissipation probability is not noticeably affected by the distortion size. The formation probability is only affected when the distortion chance is low, in which case the formation probability increases with the distortion size. One explanation is that if the distortion chance is low, a larger distortion size helps the swarm get going because it helps to create a false signal away from the food. When the swarm has formed, the false signal already exists, and the distortion size does not affect how long the swarm remains.

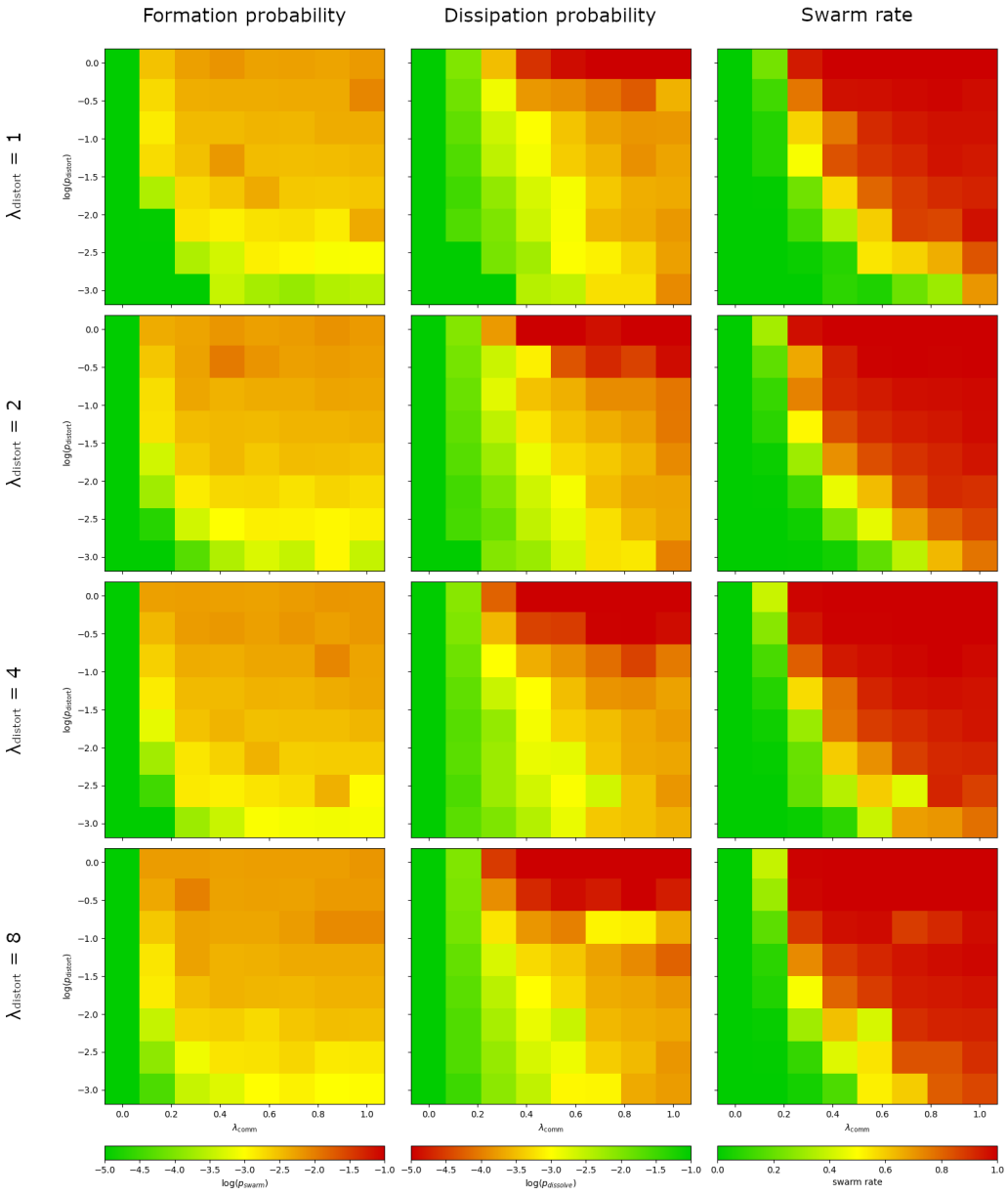


Figure 21: Heat-maps for the swarm formation probability, swarm dissipation probability and swarm rate for different simulation parameters. Each data point represents the average over 50 simulation runs. Here the x-axis represents the average communication budget ranging linearly from 0 to 1, which determines how much agents can communicate each tick. The y-axis shows the distortion chance ranging logarithmically from 0.1% to 100%. All simulations use a population size of 20, a communication range of $R_{\text{comm}} = 6$ and a room radius of $R_1 = 18$.

4.7.3 Communication range dependence

Lastly, we study the influence of the communication range on phantom swarm formation. Here the communication range determines the maximum distance at which two agents can communicate. Figure 22 shows the results of 12800 simulations, for 4 different communication ranges. This shows that the communication range has a large influence on the presence of phantom swarms. When the communication range is set to 1, swarms quickly dissipate for all tried parameters. Swarms can still form when the distortion chance and communication budget are high enough but swarms do not last for a prolonged period of time. In this case, false signals cannot spread fast enough due to the limited communication range, allowing agents to reset their internal memory. Even though this prevents agents from being stuck in place for long, this can have the downside that agents forget the position of the original food source. The swarm rate noticeably increases when the communication range is increased to 3 or 6. Here the increased communication range makes it possible for the false signal to propagate. However, when the communication range is increased to 9, the swarm rate does not noticeably increase. At this point the communication range is sufficient for signals to propagate efficiently, making any additional range irrelevant.

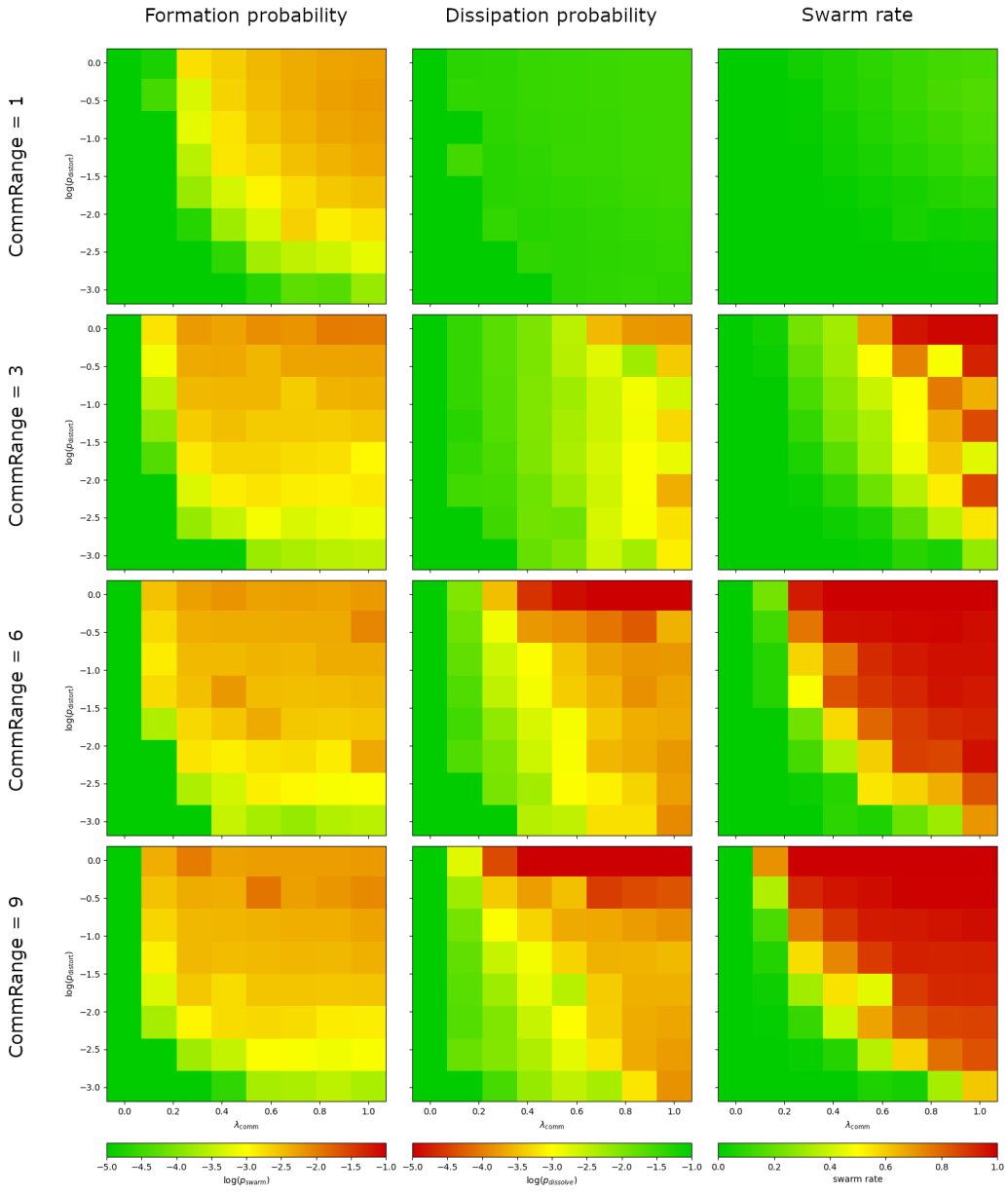


Figure 22: Heat-maps for the swarm formation probability, swarm dissipation probability and swarm rate for different simulation parameters. Each data point represents the average over 50 simulation runs. Here the x-axis represents the average communication budget ranging linearly from 0 to 1, which determines how much agents can communicate each tick. The y-axis shows the distortion chance ranging logarithmically from 0.1% to 100%. All simulations use a population size of 20, an average distortion size of $\lambda_{\text{distort}} = 1$ and a room radius of $R_1 = 18$.

4.8 Counter measures

In the previous results, it was shown that phantom swarms can significantly impact the behaviour of the group. However, in this case the agents do not have any defence mechanisms designed to stop phantom swarms. We now look at some measures that can be taken by the agents in order to stop phantom swarms, and examine under which circumstances these measures are effective.

With the behaviour tree that is currently used, agents compare received information with information stored in their internal memory. When the received information, i.e. the position and value of a food source, sounds more preferable than the currently memorized information then the new information is copied. Here the agents only compare how much they prefer each piece of information, and do not consider the accuracy of the information. This is not unreasonable in principle, as none of the agents are intending to be deceptive. However, due to the communication distortions, some skepticism might help the agents discern true and false information. One simple way of making agents more skeptical is by letting agents ignore information that they can readily disprove. Agents are sometimes told about food sources that would be in their detection range. In this case, if the agent does not actually sense any food nearby, the agent can readily ignore this supposed food source, as it can not exist. This allows agents to ignore some of the false signals that are propagated through the population, preventing the formation of phantom swarms. This change is implemented by modifying the behaviour three that was previously used. Here an additional condition is added that stops agents from accepting food sources that are within their smell range when no food is detected.

4.8.1 Population size dependence

We now perform the same experiments that were performed previously, now with the adapted behaviour tree that can partially ignore false information. We start by again looking at the impact of the population size on phantom swarm formation, which is shown in Figure 23. When comparing this graph to Figure 20, which showed swarm presence for the initial behaviour tree, we see that both the swarm formation probability and dissipation probability have changed notably. For the formation probability, the boundary between swarms forming and not forming has shifted so that a higher communication budget and distortion chance is needed for swarms to form. The dissipation probability has noticeably increased, with swarms only failing to dissipate when communication distortion is nearly guaranteed. We also see that the population size again influences swarm presence. At a population size of 10, the swarm rate is less than 30% for all tried parameters. For a population size of 25, long term swarms can occur when the communication budget and distortion chance are high enough. However, the required communication budget and distortion chance for swarms to occur are much higher than was the case with the original behaviour tree.

The difference between these results and the results for the previous behaviour tree can be understood by looking at the behaviour of the group in real time. Previously, even though communication distortions were necessary for phantom swarms to form, these distortions were not necessary for swarms to persist. Once enough distortions were introduced, a false signal could be transmitted without having to be distorted further. With our new setup, transmissions of obviously false signals are ignored. In this case if only a single distortion is introduced, this false signal is eventually disproven by the group. Phantom swarms then only persist if new false signals are continuously generated. These false signals are further away, making them harder to disprove. This has the effect that instead of converging on the centre, the swarm is now continuously changing position. This idea explains the increased importance of the distortion chance, as seen in Figure 23. Here we see that a higher distortion chance is necessary for swarms to persist compared to with the previous behaviour tree.

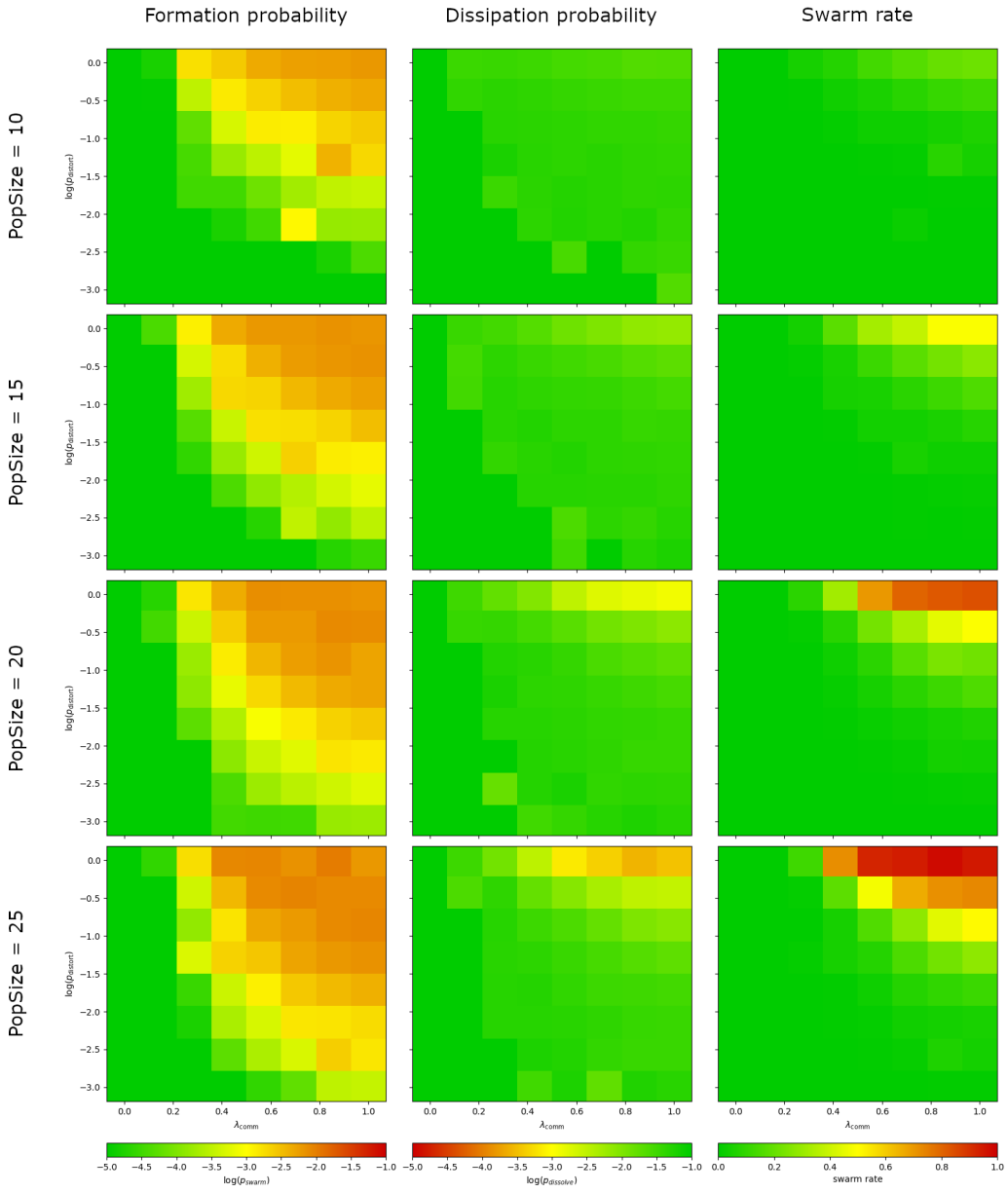


Figure 23: Heat-maps for the swarm formation probability, swarm dissipation probability and swarm rate for different simulation parameters, now using an adapted behaviour tree that helps agents ignore provably false information. Each data point represents the average over 50 simulation runs. Here the x-axis represents the average communication budget ranging linearly from 0 to 1, which determines how much agents can communicate each tick. The y-axis shows the distortion chance ranging logarithmically from 0.1% to 100%. All simulations use an average distortion size of $\lambda_{\text{distort}} = 1$, a communication range of $R_{\text{comm}} = 6$ and a room radius of $R_1 = 18$

4.8.2 Distortion size dependence

We now examine the effect of the distortion size on the presence of phantom swarms when using the adapted behaviour tree. Figure 23 again shows the result of 12800 simulation runs in total for 4 different values of λ_{distort}. As opposed to with the original behaviour tree, we now see that the distortion size does affect the swarm rate. When the average distortion size is set to 1, the swarm rate only comes close to 100% when the distortion chance is set to 100%. However, when the average distortion size is set to 8, the swarm rate can come close to 100% even for distortion sizes below 10%. The reason that the distortion size is relevant in this case likely has to do with how distortions help preserve the phantom swarm. Due to the added condition in the behaviour tree, agents do not accept new coordinates if they can prove that these coordinates contain no food. Large distortions are more likely to generate coordinates outside of the detection range of the receiving agent, meaning that these agents can not disprove the presence of food and thus accept the false information.

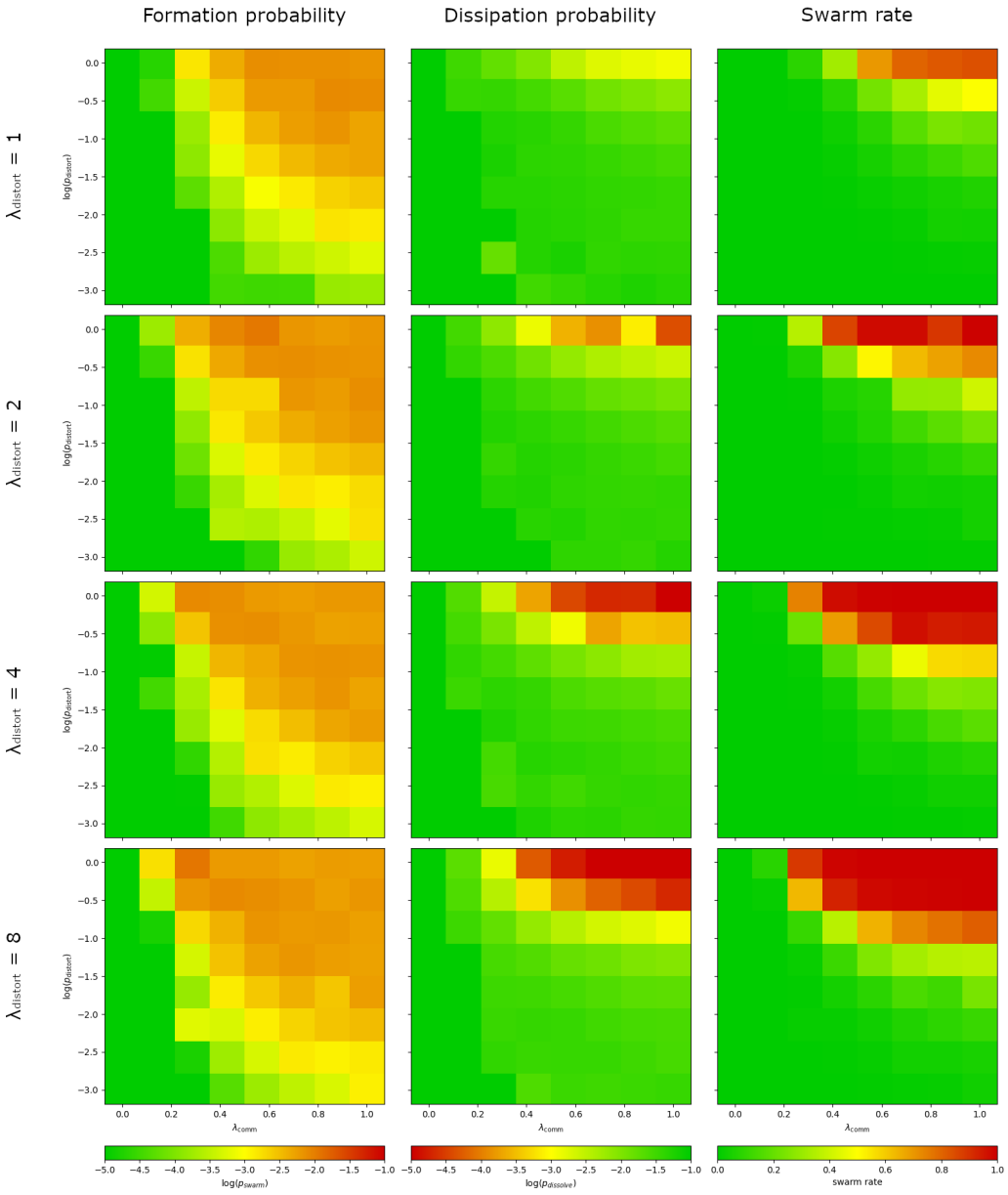


Figure 24: Heat-maps for the swarm formation probability, swarm dissipation probability and swarm rate for different simulation parameters, now using an adapted behaviour tree that helps agents ignore provably false information. Each data point represents the average over 50 simulation runs. Here the x-axis represents the average communication budget ranging linearly from 0 to 1, which determines how much agents can communicate each tick. The y-axis shows the distortion chance ranging logarithmically from 0.1% to 100%. All simulations use a population size of 20, a communication range of $R_{\text{comm}} = 6$ and a room radius of $R_1 = 18$.

4.8.3 Communication range dependence

Lastly we again examine the influence of the communication range on phantom swarm formation, now using the updated behaviour tree. Figure 25 shows the relationship between phantom swarm presence and communication range when using the updated behaviour tree. Compared to the results with the previous behaviour tree, continued phantom swarms are now absent for communication ranges 3 and below. However, at communication ranges 6 and 9, persistent swarms are possible with a high enough distortion chance. The figure also shows a noticeable increase in the swarm rate when increasing the communication range from 6 to 9, which was not present when using the original behaviour tree. Since the phantom swarm now requires constant distortions in order to be preserved, agents in the swarm are more spread out. This additional increase in communication range then prevents agents from splitting off from the swarm.

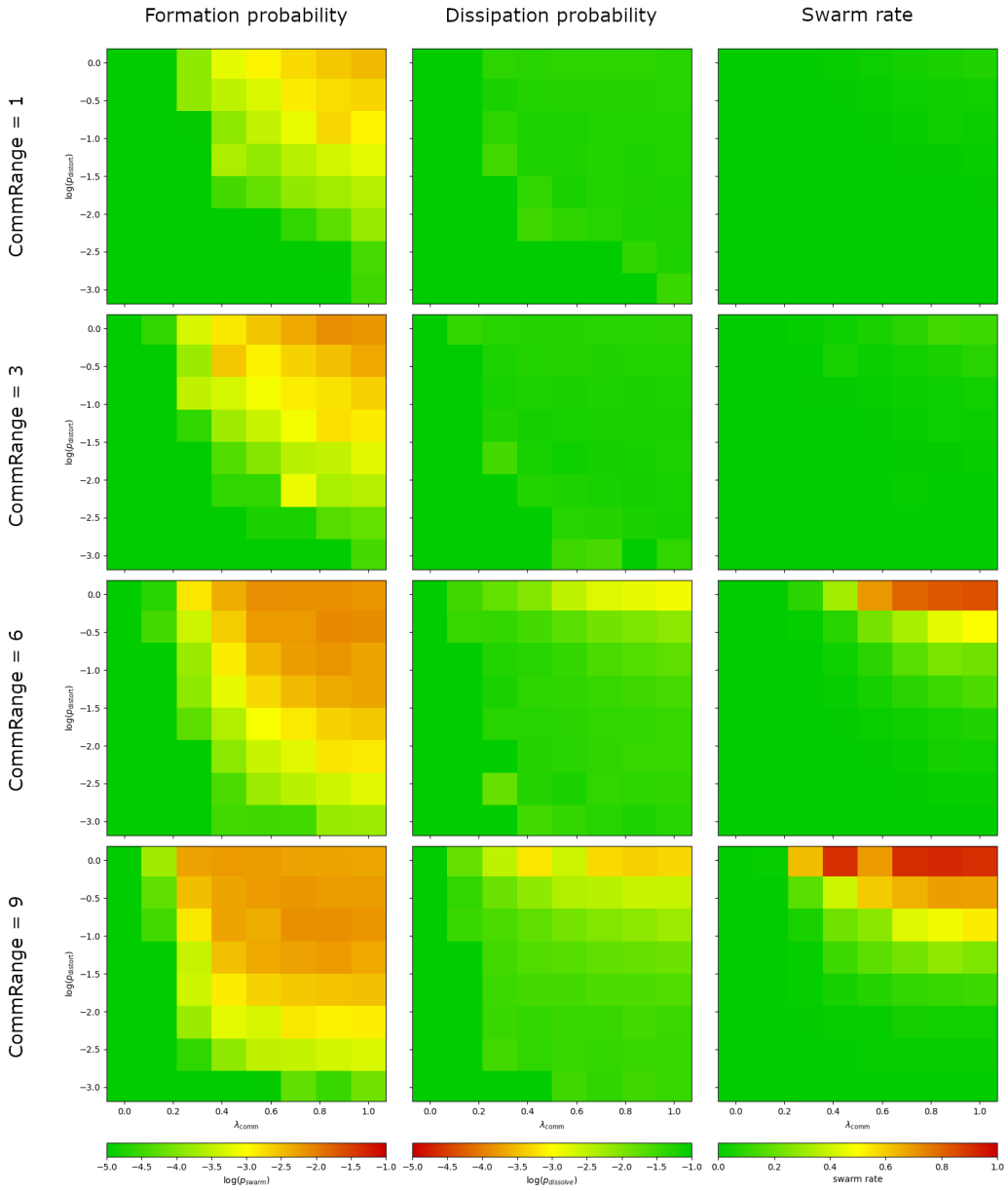


Figure 25: Heat-maps for the swarm formation probability, swarm dissipation probability and swarm rate for different simulation parameters, now using an adapted behaviour tree that helps agents ignore provably false information. Each data point represents the average over 50 simulation runs. Here the x-axis represents the average communication budget ranging linearly from 0 to 1, which determines how much agents can communicate each tick. The y-axis shows the distortion chance ranging logarithmically from 0.1% to 100%. All simulations use a population size of 20, an average distortion size of $\lambda_{\text{distort}} = 1$ and a room radius of $R_1 = 18$.

4.9 Disregarding indirect sources

In the previous section, an additional condition was added to the behaviour tree that helped prevent phantom swarms by making agents ignore provably false information. This measure helped reduce the frequency and duration of phantom swarms but still allowed swarms to form in circumstances with a large enough communication distortion. We now examine another way of limiting the effect of communication distortions. In scholarship, secondary and tertiary sources are generally considered less reliable than primary sources. This principle can be translated to our simulation, by making agents ignore information that has been passed around too many times. In order to do this the agents now keep track of the age of the information they have currently memorized. When an agent picks up a piece of food and remembers the location, the age is set to 0. If this information is then transmitted, the receiving agent copies the age of the information and increments it by one. In this way the other agents will be able to tell that this agent is a secondary source, and thus reconsider accepting their message. Here we set a limit τ_{max} for the maximum acceptable information age τ, so that any message with an age higher than τ_{max} is ignored. The downside of this approach is that it requires us to track the source of each piece of information, which in reality might not always be possible. Another downside is that this can hinder messenger agents, which are used in some multi-agent systems to transport information.

We now examine the effect of indirect source prevention in the same way as the previous experiments. Here τ_{max} is set to 2, meaning agents only accept primary, secondary and tertiary sources. We also again use the previously defined adaptation that helps agents ignore provably false information. Instead of running the simulation for a variety of setup parameters, we now only run the simulation for the setup that was most challenging. Here the population size is set to 25, the average distortion size λ_{distort} is set to 8, and the communication range is set to 9. The result is shown in Figure 26, which again shows the swarm formation probability, dissipation probability and swarm rate. Here we see that even for this challenging scenario, phantom swarms are almost completely prevented. When running this setup in real time, we see that any distorted signals generated through miscommunication are not able to be preserved for long. These false signals are only able to be passed back and forth a fixed amount of times, making them dissipate before they can seriously affect the group.

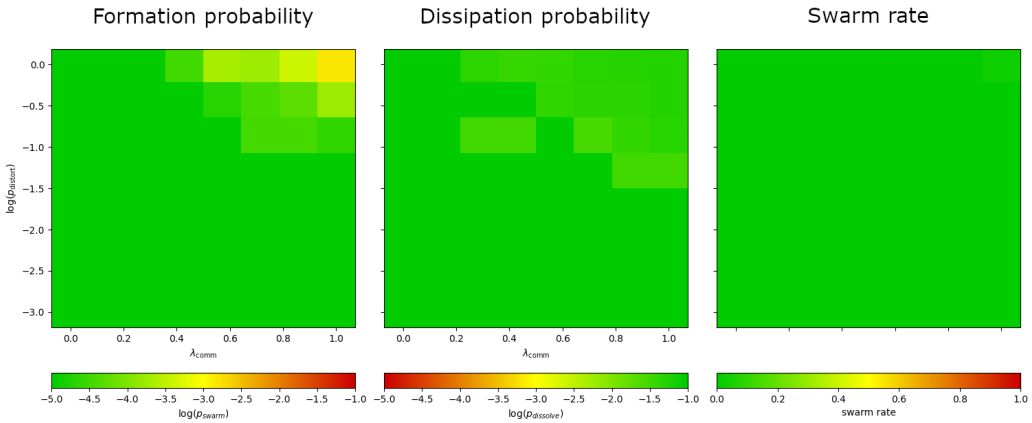


Figure 26: Heat-maps for the swarm formation probability, swarm dissipation probability and swarm rate. Here the behaviour tree is adapted so that agents ignore messages that are provably false and messages that have been passed around too many times. Each data point represents the average over 50 simulation runs. Here the x-axis represents the average communication budget ranging linearly from 0 to 1, which determines how much agents can communicate each tick. The y-axis shows the distortion chance ranging logarithmically from 0.1% to 100%. All simulations use a population size of 25, an average distortion size of $\lambda_{\text{distort}} = 8$, a communication range of $R_{\text{comm}} = 9$ and a room radius of $R_1 = 18$.

5 Discussion and conclusion

This work introduced a multi-agent foraging simulation with the goal of studying the effect of communication obstructions on the performance of groups of cooperating agents. Here the simulation involved a colony of autonomous agents that gather food and work together by sharing information about detected food sources. The behaviour of these agents was rule-based, governed by pre-made behaviour trees. In order to test the influence of communication obstructions, we first examined the behaviour of the group under ordinary circumstances without any obstructions. Communication was then hindered in two ways: limiting the frequency of communication and introducing communication distortions which modify the transmitted information. We then studied under which circumstances these obstructions are most harmful to the performance of the group. Here we now summarize and discuss our findings, as well as address some limitations and consider some possible future work.

5.1 Key findings

Initially the simulation was run without including a communication limit or information distortion. Here the performance of the group as a function of several setup parameters was examined. The most notable result is the influence of the communication range:

- **The communication range influences the rate of exploration and exploitation.**

Here we saw that balancing the communication range, the maximum range at which agents can communicate, is not trivial. When the communication range is set too high, the group converges too quickly on a discovered food source, disregarding possible better food sources. When the communication range is set too low, agents are more independent, and spend too much time exploring. This result is partly due to the limitations of the relatively simple behaviour of the agents, with smarter agents potentially being able to overcome the downside of large communication ranges. However, this effect still needs to be taken into account when dealing with a multi-agent system. When designing a multi-agent system, for example in the case of swarm robotics, an unlimited communication range is not necessarily optimal. The same principle should also be considered when organizing human groups. One example of communication between humans having a negative effect is found in Lorentz et al, 2011[19], which studied the wisdom of the crowd effect. Here the wisdom of the crowd effect is a phenomenon where taking the average judgement of a group of people can result in surprisingly accurate predictions. However, when individuals are allowed to communicate, it was found that this effect is undermined.

Comparable to the communication range, we also studied the effect limiting the frequency of communication had on the performance of the group. Here agents are assigned a communication budget each step, which has the effect that agents can communicate less often. Similar to the communication range, it was found that:

- **Limiting the frequency of communication can be beneficial in some circumstances.**

Here limiting the frequency of communication can prevent the group from converging too fast on a food source. As with the communication range, choosing the right communication frequency comes down to a balance between exploration and exploitation. Here the risk of excessive communication is that agents lose their individuality, making it so that the group loses the advantage of parallelism.

Another way of obstructing communication is by distorting transmitted information. Here agents have a chance p_{distort} of mishearing other agents, resulting in agents receiving another coordinate than was sent. Due to the way

agents accept new information, this can drastically reduce the performance of the group.

- **Distorting the information sent between agents can cause the system to fail when agents accept incorrect information too readily.**

With our initial setup, when agents receive information that sounds more preferable than their original memory, agents always accept the more appealing information. This then leads to an information cascade, where false but desirable information quickly spreads through the population. In our simulation, this leads agents to believe food sources exist closer to drop-off points than in actuality. What is unexpected in this case is that this false information persists even when individual agents have the ability to disprove it over time. When agents reach a target coordinate and fail to find a food source, their false information is discarded. However, because this false information can spread between agents faster than it can be disproven, it endures unchecked. Here this effect was named a **Phantom swarm**, as the group swarms around a non-existing food source. When studying this phenomenon, it was found that the formation of phantom swarms depends highly on the setup parameters that are used. Most significantly, we find that:

- **Limiting communication helps prevent the negative effects of communication distortions.**

The spread of false distorted information can be compared to the propagation of a virus, here with the information or virus being spread from agent to agent through communication. When the amount of communication is limited by the setup parameters, we see that phantom swarms quickly dissipate. Here three parameters can affect the amount of communication. Firstly, increasing the communication range was found to increase the presence of phantom swarms. Here a larger communication range makes it easier for false information to be spread and maintained. In a similar way, increasing the frequency of communication also increases phantom swarm presence. Lastly, a high population size also indirectly increases phantom swarm presence. Here having more agents increases communication because there are more agents that communicate. In all these cases, a relatively small change in parameters can sometimes drastically change the disruptiveness of phantom swarms. This is similar to a phase transition, where phantom swarms are either able to preserve themselves for long periods of time or fall apart quickly, depending on the setup parameters. This can have some counter-intuitive results, where for example, increasing the total number of agents can actually decrease performance.

Another way of preventing phantom swarms that was examined is by making agents more skeptical when accepting new information. Here we found that:

- **Making agents more skeptical when accepting new information helps prevent the negative effects of communication distortions.**

One of the main reasons that phantom swarms occur is that agents do not take into consideration the likelihood of incoming information being true, and only compare the preferability of information. If agents can ignore a portion of incoming false information then phantom swarms can be greatly diminished. One preventative measure that was tested is making agents ignore provably false information. This made it so that phantom swarms only persist in setups with a high enough distortion chance. In practice we see that without this change, swarms can persist even when distorted information is only introduced once. When agents are made more skeptical, swarms require false information to constantly mutate in order for the swarm to persist. We see that when false information is constantly changing, disproving it becomes harder. This is also reflected in the influence of the average distortion size, which determines how much information is perturbed when a miscommunication happens. In this setup, a larger distortion size leads to phantom swarms becoming more persistent. Here frequent large distortions make it hard to disprove false information, as new fake food sources crop up faster than can be disproven.

Another measure that was tested is making agents track the age of information. When agents receive a piece of information that is too far removed from the primary source, they elect to ignore it. This was shown to completely prevent phantom swarms from persisting, even for the least favourable setup parameters. However, the downside of this is that it requires every agent in the group to track the source of every piece of information, which might not be possible or desirable for some systems.

5.2 Limitations

The main goal of this work was to examine the influence of communication obstructions on groups of cooperating agents. Here we discuss some limitations of this work in regard to the discovered findings. One major limitation is that the experiments were performed for one specific type of map with the agents having a pre-defined behaviour. Here our map contains four static food sources of equal distance to the centre. The behaviour of the agents was designed to be effective on this map, and will fail in a more complicated environment. More complex behaviour and maps were designed but tests on this new environment were not included. Here one significant challenge in testing these more complex environments is that these usually include more setup parameters, making them exponentially more time consuming to explore fully. Our results should thus not be taken as generalisable to most multi-agent systems, but rather as considerations when engaging with a group of communicating actors.

Another limitation is the statistical model that was used to approximate phantom swarm formation/dissipation. Here it was assumed that the probability of a phantom swarm forming/dissipating was time independent, with each tick having an equal chance of swarm formation/dissipation. We see that in practice this assumption does not fully hold, for example because the group

cannot form a swarm right after it has dissipated because the agents have to relocate a food source. Therefore the results should not be considered as exact quantitative predictions.

5.3 Future research

The main result discovered in this work was that communication disruptions can cause groups of agents to collapse when individuals prefer some information over other information. We now discuss some ways this idea can be studied further.

As mentioned previously, the scope of our simulation was limited, with agents having relatively simple behaviour and with the map being mostly static. In a more complex environment with more intelligent agents the effect of communication distortions could be significantly different. One way to make the simulation more complex is by making food sources dynamic, appearing and disappearing over time. Agents could adapt to this by storing multiple coordinates, allowing agents to switch between discovered food sources. One thing to look out for is a possibly subtler version of a phantom swarm, where instead of having all agents swarming around a coordinate agents periodically return to a false coordinate. This would be much harder to detect than a regular phantom swarm and could conceivably harm a more intelligent group.

One design goal when creating the simulation was to make it easily modifiable, which was accomplished by using customizable maps and behaviour trees. This allows it to be adapted for other scenarios. One interesting continuation would be to look at non-heterogeneous populations, where not all agents have the same behaviour tree, and examine the effect of communication distortions on these groups. Here, the population could for example be split into multiple sub-groups with different roles, such as looking for food or relaying information. Another option would be to make the scenario non-cooperative, with some agents seeking to hinder another group. Here some agents could be designed to deliberately spread disinformation, aiming to waste the time of the main group. An interesting experiment would then be to see what measures work well to prevent these agents from hindering the group.

Another interesting experiment would be to see if phantom swarm like phenomenon can be found in other scenarios. One possible example which was mentioned previously is the spread of misinformation on social media. Here an idea is to simulate a group of social media users and then test the effect of introducing false but appealing information. It is conceivable that a phantom swarm like phenomenon can happen in this context, with false information persisting over time by being transmitted between agents. By simulating this, it is again possible to test in which circumstances this effect is most damaging.

References

[1] Ercan U Acar et al. “Path planning for robotic demining: Robust sensor-based coverage of unstructured environments and probabilistic methods”. In: *The International journal of robotics research* 22.7-8 (2003), pp. 441–466.

[2] Christian Blum. “Ant colony optimization: Introduction and recent trends”. In: *Physics of Life reviews* 2.4 (2005), pp. 353–373.

[3] Ahmad Reza Cheraghi, Jochen Peters, and Kalman Graffi. “Prevention of ant mills in pheromone-based search algorithm for robot swarms”. In: *2020 3rd International Conference on Intelligent Robotic and Control Engineering (IRCE)*. IEEE. 2020, pp. 23–30.

[4] Peter J Deadman. “Modelling individual behaviour and group performance in an intelligent agent-based simulation of the tragedy of the commons”. In: *Journal of Environmental Management* 56.3 (1999), pp. 159–172.

[5] Ali Dorri, Salil S Kanhere, and Raja Jurdak. “Multi-agent systems: A survey”. In: *Ieee Access* 6 (2018), pp. 28573–28593.

[6] Elliott Hogg et al. “Evolving behaviour trees for supervisory control of robot swarms”. In: *Artificial Life and Robotics* 25 (2020), pp. 569–577.

[7] Matteo Iovino et al. “A survey of behavior trees in robotics and ai”. In: *Robotics and Autonomous Systems* 154 (2022), p. 104096.

[8] Dervis Karaboga and Bahriye Akay. “A comparative study of artificial bee colony algorithm”. In: *Applied mathematics and computation* 214.1 (2009), pp. 108–132.

[9] Ali Marjovi et al. “Multi-robot exploration and fire searching”. In: *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2009, pp. 1929–1934.

[10] Victoria L Rubin. “Disinformation and misinformation triangle: A conceptual model for “fake news” epidemic, causal factors and interventions”. In: *Journal of documentation* 75.5 (2019), pp. 1013–1034.

[11] Michael Schader and Sean Luke. “Exploring planner-guided swarms running on real robots”. In: *International Conference on Practical Applications of Agents and Multi-Agent Systems*. Springer. 2023, pp. 307–319.

[12] Jieun Shin et al. “The diffusion of misinformation on social media: Temporal pattern, message, and source”. In: *Computers in Human Behavior* 83 (2018), pp. 278–287.

[13] Aleksandrs Slivkins et al. “Introduction to multi-armed bandits”. In: *Foundations and Trends® in Machine Learning* 12.1-2 (2019), pp. 1–286.

[14] Mart Van Der Kam et al. “Agent-based modelling of charging behaviour of electric vehicle drivers”. In: *Journal of Artificial Societies and Social Simulation* 22.4 (2019), p. 7.

[15] Richard T Vaughan et al. “Blazing a trail: insect-inspired resource transportation by a robot team”. In: *Distributed autonomous robotic systems 4* (2000), pp. 111–120.

[16] Dongshu Wang, Dapei Tan, and Lei Liu. “Particle swarm optimization algorithm: an overview”. In: *Soft computing* 22.2 (2018), pp. 387–408.

[17] Yiyu Wang, Jiaqi Ge, and Alexis Comber. “An agent-based simulation model of pedestrian evacuation based on bayesian nash equilibrium”. In: *arXiv preprint arXiv:2211.14260* (2022).

[18] Brian E Weeks. “Emotions, partisanship, and misperceptions: How anger and anxiety moderate the effect of partisan bias on susceptibility to political misinformation”. In: *Journal of communication* 65.4 (2015), pp. 699–719.

[19] Nic M Weststrate, Susan Bluck, and Judith Glück. “Wisdom of the Crowd”. In: *The Cambridge handbook of wisdom* (2019), pp. 97–121.