FORAGING ALGORITHMS FOR ROBOTIC SWARMS

Sunidhi Azad

A thesis

in

The Department

of

Computer Science

Presented in Partial Fulfillment of the Requirements

For the Degree of Master of Computer Science

Concordia University

Montréal, Québec, Canada

June 2015

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By:	**Sunidhi Azad**
Entitled:	**Foraging Algorithms for Robotic Swarms**

and submitted in partial fulfillment of the requirements for the degree of

Master of Computer Science

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

______________________________ Chair

Dr. G. Butler

______________________________ Examiner

Dr. J. Opatrny

______________________________ Examiner

Dr. T. Fevens

______________________________ Supervisor

Dr. L. Narayanan

Approved ______________________________

Chair of Department or Graduate Program Director

__________ 20 ______ ______________________________

Dr. Amir Asif, PhD, PEng

Dean, Faculty of Engineering and Computer Science

Abstract

Foraging Algorithms for Robotic Swarms

Sunidhi Azad

Foraging is the process of looking for food or other resources to survive in an unknown environment. Various animals exhibit foraging behavior to fend for themselves in the wild. We study group foraging by a swarm of robots, which involves collectively searching for an object of interest and bringing it back to a central location.

We assume n robots start at the center of a $g \times g$ grid, called the *nest*. They are looking for some units of *food* placed at an unknown location in the grid. When the food is found by one of the robots, it must communicate its location to the other robots, which then collectively transport the food back to the nest. We divide the task into three phases: the *exploration phase*, in which the robots search different parts of the grid for the food; the *communication phase*, in which robots communicate the location of the food to each other; and the *transportation phase*, in which the robots transport the food back to the nest. We give five novel algorithms for the exploration phase, and analyze their competitive ratios. For the transportation phase, we give two different approaches, and give a theoretical analysis of a simple case. We implemented all our algorithms and compared their performance to an existing algorithm in the literature.

Acknowledgments

First & foremost, it is a real pleasure to express my deepest gratitude to my supervisor, Dr. Lata Narayanan, for her vast support throughout the tenure of my thesis. I am extremely thankful to her for sharing her expertise with me and her valuable guidance and encouragement extended to me. She has been nothing but kind and really helpful to me and for that I am forever indebted to her.

Next, I wish to express my heartfelt thanks to my beloved family, father Dharampal Azad, mother Usha Rani and brother Rohit Azad for supporting me at all times and making me what I am today. I cannot thank Sahil Singh enough for pushing me when it was needed the most. A very special thanks goes to my dearest friend, Ritika Malhotra, for being there for me no matter what. It could not have been such an amazing journey if not for her.

I would also like to thank my lab mates, Puspal Bhabak, Kangkang Wu, Zhiyuan Li, Meghrig Terzian for all the times we have spent together in the lab supporting each other and solving problems. Finally, I would like to sincerely thank Frances Howell-Slater for her much needed help in the simulation during the final days of completion of my thesis.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction and preliminaries

1.1 Swarm Intelligence

Collective behavior is the term coined by Robert E. Park [10] to describe the aligned conduct of large groups of similar individuals. The main idea behind such a behavior is that there is no central leader yet the individuals work together towards achieving a common goal. The key features of collective behavior include strategies to fairly divide the workload among the group members, the propagation of information across the group, the decision-making process, and the synchronized group movement. These individuals can be mammals, birds, ungulates, fish, small robots, programmable entities or software agents in a simulation. We have been intrigued by the challenge of determining how the limited intelligence and actions of these relatively simple individuals gel together to solve the more complex problems.

There are countless examples that illustrate the existence of such behavior in natural ecosystems. Group operations in biological systems usually include migrating to a new place in herds, protection from predators, finding food in the wild. The term *schooling* is used to describe the behavior when fish swim together really close to each other [11, 12, 13]. They just follow a simple rule to stay close enough to their neighbors so that they don't stray away from the group and far enough to not collide with fellow fish. *Flocking* [14] is the term used for the swarm behavior exhibited by a large number of birds when flying together. Every bird tries to stay close to its neighboring birds but far enough to have its own space to fly, same as the fish do when schooling.

The behavior of nest building ants [15] is the most commonly studied example for both biological swarming behavior as well its application to computer science. Ants are so little in size yet they achieve so much by working in groups. They communicate indirectly by leaving pheromone trails on the ground for the other ants to follow so that they remain together. Their main tasks include searching for food or for material to build or strengthen their nest, and then bring this material back to their nest following the pheromone trail which they laid on the ground. Achieving these tasks is way beyond the capabilities of a solitary ant.

Swarm Intelligence [16] is the term used to describe collective behavior of autonomous agents in the field of Computer Science. It has its roots with the study of behavior of social insects. The application of swarm intelligence to robotics gave rise to the study of *Swarm Robotics*. The main idea is to use several simple robots instead of a single powerful one. The most obvious and important benefit is fault tolerance. Even if some of the robots fail in the middle of an operation, it does not affect the whole system. Among other benefits to this arrangement is the reduction in the cost of robotic hardware. Smaller robots are simpler in terms of hardware complexity and smaller sensors are enough to support them. They need minimal processing power and limited means of communication. There is yet another advantage to using a swarm rather than a single agent and that is the amount of parallelism achieved by dividing the overall task into subtasks which can be performed simultaneously.

However, there are two sides to every coin. Swarm behavior is not just the addition of individual agents' actions, it is how they work together to achieve the global goal. This is the main drawback of the swarm approach, i.e., the agents need to coordinate their actions and communicate their findings to their fellow agents. With the increase in the number of agents in the swarm, management of movement of every individual agent is necessary to ensure that there are no collisions and the agents coordinate their actions. In addition, the agents are believed to have very limited communication hardware and hence, they have to be physically close in order to share the information. Therefore, swarm intelligence can be seen as the emergence of a global behavior among the agents from the distributed execution of the algorithms, sensing of the local environment and the minimum-to-zero communication among the agents.

1.2 Problem Description

The term *foraging* means to wander in search of food or provisions in an unknown environment [17]. The problem discussed in this thesis is *Central Place Foraging* [18, 19] in which a swarm of agents or robots assumed to start at a central location, called the *nest* is required not only to find food located at an unknown location in the environment, but also to bring it back to the nest. Some applications for this problem that have been studied previously include a set of robots that need to find construction blocks and carry them to the construction site [20] or a set of robots that need to locate energy sources to recharge before returning to the base.

In this thesis, we consider a swarm of n agents gathered at the *nest* initially to perform the task of foraging. They leave the nest in a pre-determined order to start searching for an object of interest. We use the term *food* for it, since we take motivation from ants foraging for food. Once the food is located, it has to be brought back to the nest one unit at a time. The amount of food is such that at least several rounds of pick ups and drops are needed. For this reason, we want the food location to be shared with all the agents after it is discovered so that they can help transport it back to the nest as quickly as possible. This requires a protocol for the agents to communicate the food location to agents in such a way that ultimately the entire swarm can be made aware of the food location.

We assume the environment to be a square region and a synchronous system model in which at every step, each agent performs a *Look-Compute-Move* cycle, as is standard in the autonomous mobile robot literature [21, 22, 23]. In the *Look* phase, an agent looks for agents in its proximity to avoid any collision situations in the subsequent time step and may sense for food in its current location. Then, depending upon its perception of the environment, it *computes* its next step. Finally, in the *Move* phase, it moves to its next location. One such cycle happens in one time step. More details about our model are given in the later sections of this chapter.

In this thesis, given n agents initially located at the *nest*, and k units of food placed at an arbitrary unknown location in the environment, the problem of central-place foraging is considered solved when all the food units have been transported back to the *nest*. The time taken is the number of *Look-Compute-Move* cycles needed. We broke down the problem into the following sub-problems; finding the food, communicating its location to all the agents, and transporting the food back to the nest. Accordingly, all our algorithms take a three-phase approach to solving the problem: *Exploration, Communication and Transportation Phase*. Every agent in the swarm goes through all these phases though not necessarily at the same time.

- *Exploration Phase* - This phase involves the agents searching for the food in the given region. We give strategies for the agents to follow so that they cover the whole region with the minimum amount of the grid being visited more than once.

- *Communication Phase* - Each agent enters this phase once it has the food location, either it discovers the food itself or receives the location information from one of its neighbors. This phase involves propagating this food location further to it's neighbors and once this has been done, the agent can move onto the next phase. The agents know their set of neighbors and their locations at every point of time. Since long-distance communication is not possible in our model, they have to reach their neighbors to share the information.

- *Transportation Phase* - The agents in this phase have the food location and they have shared this information with their neighbors. This phase involves picking up one unit from the food location and taking it back to the nest and back and forth again.

1.3 The System Model

1.3.1 Environment Model

The search space is a square region that we divide into g^2 smaller *grid cells*. A cell is uniquely identified by its cartesian coordinates. The size of the grid is represented as $g \times g$. The center of the grid is referred to as the nest in the remainder of this thesis. The distance metric we use is the *Manhattan* distance: the distance between cells with coordinates (p_1, p_2) and (q_1, q_2) is given by $|p_1 - q_1| + |p_2 - q_2|$.

1.3.2 Agent Model

An agent is simply a computational entity which has very limited capabilities. A swarm of n identically programmed agents is initially located at the nest. These agents are capable of taking independent decisions towards the completion of the overall goal. These agents are very minimal in design. In particular, each agent possesses $\theta(\log g)$ memory to perform the needed computations and store some limited information. Every agent possesses the following modules:

- *Sensing Module:* At every step, an agent can sense for the presence of the food in its current cell. It can also sense the boundary of the square region.

- *Visibility Module:* Every agent can *see* another agent in any cell at a distance of at most 2 from its current cell. As can be seen in figure 1, agents located in cells B_1, B_2, B_3, B_4 are the 1-hop neighbors of $Agent_A$ and agents in cells $C_1, C_2, C_3, C_4, C_5, C_6, C_7, C_8$ are the 2-hop neighbors. When observing its environment, $Agent_A$ sees all the agents which are located in these cells.

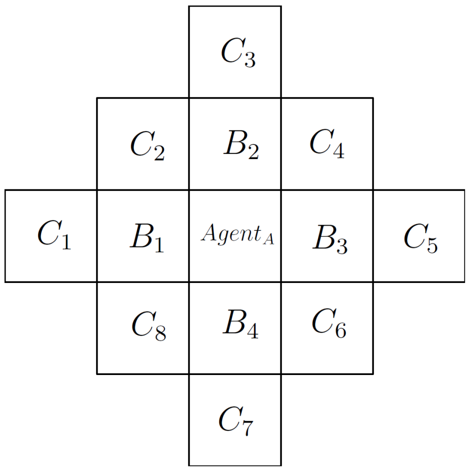


Figure 1: 2-hop neighbors of $Agent_A$

- *Movement Module:* In one Look-Compute-Move cycle of an agent, it can move to its adjacent cell, in any one of the cardinal directions: North, South, West or East.

- *Communication Module:* Every agent can *communicate*, for example, using wireless transmission with agents in any cell at a distance of 1 from its current cell. Since communication is considered the most expensive task in the robotics literature, the agents use this module only to communicate the location of the food. They compute and execute all other actions without any communication with other agents.

- *Transportation Module:* Agents can pick, carry, exchange and drop one unit of food at a time.

- *Memory Module:* Agents possess enough memory to store the program that dictates the execution of the algorithms and location of the food if they know it.

- *Computational Module:* Agents are capable of performing some needed computations such as determining the current location of any other agent on the grid.

Since it is assumed that all the agents are present at the same location initially, it is safe to say that they can determine their total population and number themselves using any naive algorithm. Therefore, each agent has a unique ID. All agents have the same program in their memory, which gives instructions for their next steps depending upon the assigned ID to them and the perception of their local environment. For the same reason, total agreement on the coordinate system is safe to assume i.e. they have a *global sense of direction*. Every agent is aware of its global position i.e. which cell it is in at the time.

Agents, however, are *not aware of the total size of the search space beforehand*. But they do know that they are located at the central location of a square grid initially. So, if they encounter the grid boundary at any point, they can easily determine the size of the search grid by doubling the distance traveled from the nest to reach there.

1.3.3 Collision Situations and Avoidance

Th nest can be seen as collection of cells in which all the agents are present initially. [1]e assume the nest to be a single cell for simplicity. However, no two agents can be present in the same cell at the same time other than the nest. As we can see in Figure 2, agents A and B are aiming to move to the same location which will result in a collision. We forbid such collisions in our model. However, when two agents are in adjacent cells, they can switch places. Referring to Figure 3, agents A and B can make their moves and take each other's locations at the next time-step since a collision is said to have happened only when two agents are in the same location. When realizing the same situation with actual robots, we can say that they reverse their roles, meaning they can resume each other's operations as before.

We want to have a *Collision-Avoidance* mechanism in our algorithms in which

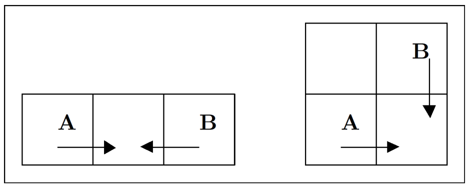


Figure 2: Conflict Situations

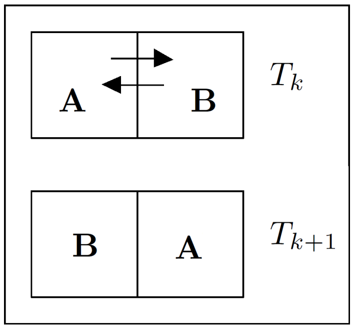


Figure 3: Non-Conflict Situation

an agent needs no communication at all with its fellow agents. We have designed the

[1]W

exploration phase in such a way that every agent has a unique non-overlapping path to follow so that no two agents ever end up in the same cell. In the later phases of the algorithm, we give some basic *collision-avoidance* techniques in which an agent weighs its priority against the other agents in its vicinity before making its move. The low-priority contenders wait at their current positions and weigh their options again in the next time step.

Based on the observation of its environment, if $Agent_A$ at timestep T_k suspects

Algorithm 1 *GetNextMove-CA* $(Agent_A, target, contenders)$

In this algorithm, *target* is the location where $Agent_A$ intends to move to and *contenders* is the list of locations where the presence of other agents can cause a collision with $Agent_A$ in the subsequent time step.

```
1:  $possibleMoves \leftarrow nil$
2:  for each $C$ in 1-hop neighboring cells of $currentPosition$ do
3:      if $distance(C, target) < distance(currentPosition, target)$ then
4:          Add $C$ to $possibleMoves$.
5:      end if
6:  end for
7:  for each cell $C$ in $possibleMoves$ do
8:      if $C.contenders$ is empty then
9:          Move to $C$.
10:     else
11:         $flag \leftarrow true$
12:         for each $B$ in $C.contenders$ do
13:             if $currentPosition.x < B.x \vee (currentPosition.x = B.x \wedge currentPosition.y < B.y)$ then
14:                 $flag \leftarrow false$
15:                 exit loop
16:             end if
17:         end for
18:         if $flag$ then
19:             Move to $C$.
20:         end if
21:     end if
22: end for
```

that making a move might cause a collision, it weighs its priority against the other agents in its vicinity which can also move to its next intended location. If it reckons that it has the highest priority, it makes the move, otherwise it waits at the current location for T_k and calculates its priority again at T_{k+1}. Even if one of the contenders has x-coordinate higher than $Agent_A$ or same x-coordinate with higher y-coordinate, $Agent_A$ does not have the priority to make the move. Other involved agents also use the same technique and only one of them moves to the location in question.

In Figure 1, suppose $Agent_A$ wants to move to B_2 in the next step, it compares its priority against the agents situated at locations C_2, C_3, C_4, we call these agents as *contenders* for $Agent_A$. It then waits or makes the move accordingly. Agents in the other locations in its 2-hop radius does not affect or get affected by $Agent_A$'s movement in the $North$ direction. Similarly, if it wants to move one step to the $West$, the list of *contenders* contain agents at the locations C_1, C_2, C_8. The collision-avoidance procedure is summarized in Algorithm 1.

1.4 Thesis Contribution

We study the problem of central-place foraging by a swarm of k agents in a square region divided into g^2 square cells. We solve the problem in three phases: *exploration phase* in which the agents collectively search for the food in the grid, *communication phase* in which the agents communicate the food location information to their neighbors, and *transportation phase* in which the agents transport the food back to the nest one unit at a time. We give four new strategies for exploration, communication strategies that match with the exploration strategies, and two transportation strategies. We analyze the competitive ratio of our algorithms for exploration and present a theoretical analysis of our transportation algorithms. We also implemented all our algorithms and give an empirical analysis of their performance.

1.5 Thesis Outline

The aim of this thesis is to investigate various algorithms for solving the problem of foraging executed by a swarm of autonomous agents. Chapter 2 outlines some of the previous work done in the field of swarm robotics. In Chapter 3, we explain various exploration algorithms in detail. In Chapter 4, we describe the communication strategies employed by the agents according to the exploration algorithm they use explained in the previous chapter. In Chapter 5, we explain *two* transportation algorithms and compare their running times. Chapter 6 contains the experimental results and the comparison of all the algorithms based on various factors. Chapter 7 contains the conclusion of the thesis and some leads for future work.

Chapter 2

Related Work

The problem of foraging originated from the study of the behavior of ants which have been known to accomplish complex goals in spite of them being such miniature individuals. They do so by exhibiting highly structured behavior and possessing the ability to work cooperatively to achieve a common goal. These goals typically include searching for food, storing it for future consumption, feeding the offspring or looking out for intruders to their nests [24]. In a natural setting, ants go looking out for food in groups in an unknown environment. A solitary ant has no idea where the food is or where it is currently located. To search for the food and to bring it back to the nest with absolutely no information at all, ants use a mechanism called *stigmergy*, [25] which is essentially communicating implicitly by leaving pheromone trails on the ground that will attract other ants to follow that path. Before moving in a particular direction, ants sense for pheromones in their surrounding areas, and move in the direction where its smell is the strongest.

2.1 Foraging using implicit communication

This concept of implicit communication via pheromone trails has always fascinated computer scientists. Multiple researchers have worked on exploiting this approach to design algorithms for finding shortest paths between two points, routing in networks, combinatorial optimization problems [26, 27, 28, 29], and so on.

Nagpal et al [30, 31] tapped the uncanny potential of communicating via the environment to solve the problem of central-place foraging on a continuous plane by a swarm of robots present at a central location initially. They substituted virtual pheromones in place of the actual chemical ones since using such substances in the field is not always feasible because of technical difficulties and increased hardware complexity. Their work focused on the foraging task implemented by a swarm of simple sensing robots with the use of short-range communication.

In their solution, each robot can either be a stationary *beacon* for holding the pheromone values or a wandering *walker* for doing the rambling work. They used two kinds of pheromone value, $foodPheromone$ and $nestPheromone$, which are simply floating-point numbers or integers depending upon which one of the three suggested algorithms they use. All the robots walk out of the nest as $walkers$ initially, some of them will decide to become beacons and stay still at their current positions. The walker robots lay $nestPheromone$ on all the beacons in their range when they are looking for food and $foodPheromone$ when they are carrying food. The beacon robots decrease their pheromone values too with time to simulate the decaying effect of physical chemical pheromones on the ground. They measure the bearing to the maximum pheromone value in their range and start moving in that direction.

Using beacons in the field for holding pheromone values has its own drawbacks. First and foremost, those beacon robots could have been utilized to search for food rather than standing still, which certainly accelerates the process of the food discovery.

Secondly, beacons are continuously transmitting their pheromone values, walkers are laying pheromones on all the beacons in their respective ranges which is a lot of communication happening. In our work, we utilize all the robots for the search operation with minimal amount of communication.

Nilkhamhang et al [32] gave two decentralized foraging algorithms when using a limited number of swarm robots in a continuous plane. Each robot has a small amount of memory for storing map information and is capable of broadcasting its location to other agents. They assumed that the agents have infrared sensors and visual aid to detect obstacles and the target. They also made use of virtual pheromones stored in the form of a map in every agent's memory. These values are updated accordingly whenever a robot receives location information from fellow robots. They always try to move in the strongest pheromone values' direction. If the pheromones are not detected in their immediate surroundings, they just walk randomly. When the food is discovered, they carry it back to the colony using the pheromone trail again. As is obvious, communication is being used extensively for the execution of these algorithms.

Watanabe et al [33] introduced a new way to communicate implicitly by substituting the chemical pheromones with graphics projected on the floor through the foraging task. They proposed $V\text{-}DEAR$(Virtual Dynamic Environment for Autonomous Robots) for experiments on real robots. In this system, CG graphics/colored lights are projected on the floor. The robots are provided with sensors capable of determining the color and brightness of the CG. Whenever a robot locates the food, it turns on its LED and starts moving towards the nest. The V-DEAR system will detect this LED and projects a CG pheromone trail on the ground. The other robots follow this light trail to get to the food and while bringing the food back to the nest, strengthen the trail further by turning on its LED.

2.2 Collective Treasure Hunt Problem

Collective treasure hunt, i.e. search for an object of interest in an unknown environment by multiple agents, is yet another classical problem that has been studied extensively in computer science [34]. Search can be seen as a significant part of the whole foraging process. Feinerman et al [35] introduced the problem of *Ants Nearby Treasure Search* (*ANTS*), a generalization of the classic cow-path problem [36], in which k identical agents, initially present at a central location, have to search for a treasure on an infinite two-dimensional plane collectively. This problem also takes motivation from the ants searching for food sources near their nests.

Agents are seen as identical probabilistic machines that can only communicate between themselves at the origin and are synchronous in the sense that they traverse one edge of the planar graph in one time unit precisely. They employed a simple technique for agents to go to a randomly chosen node from the set of nodes of the planar graph according to some criterion, perform a spiral search around that node for a fixed amount of time and returning back to the central place and repeat the whole process again if the treasure is not located. They showed that the agents can perform well without being aware of their population. However, the running time of the algorithm can be improved by a significant amount by having a constant approximation of their total number.

Emek et al [1] further worked on the *Ants Nearby Treasure Search* (*ANTS*) problem. In contrast to the agent model studied in [35], they let the agents communicate anywhere on the grid in the sense that they can merely *see* the states of other agents present in the same cell. Every agent can decide to either move to one of the four neighboring cells or stay put depending upon its perceived surroundings. They gave a collective search strategy called *RectSearch* in which the swarm is divided into teams of *five* agents. Each team is emitted from the origin, all *five* agents at the same time out of which $four$ agents become *guides*, one for every cardinal direction and the fifth one becomes an $explorer$. The guides can be seen as marking the axes of the grid. The explorer does the search work by going from one guide to the next following a zig-zag path as shown in Figure 4. Whenever a guide is visited by an explorer, it moves to the next cell in its cardinal direction. In this way, the grid is traversed by the agents covering the cells which are at a distance d from the origin in one round.

Emek et al [2] analyzed the minimum number of agents required to locate the treasure within a given time frame. They showed that a minimum of $four$ agents are required to locate the treasure in an asynchronous system out of which *three* are guides - $NorthGuide(N)$, $WestGuide(W)$ & $EastGuide(E)$ and one is an $explorer(X)$ which is responsible for exploring the grid in triangular fashion going from one guide to the next as shown in Figure 5. Further, they showed that if it is a synchronous system, only *three* agents are enough to search the grid out of which one is an *explorer* to do the search work, the second one is a *guide* to depict the axes and the third is an *origin-guide* to mark the origin.

The closest model to our work is used in Lopez-Ortiz et al [3], which also studied the problem of collective treasure hunt in which k agents were used to search for an object of interest on an infinite lattice. They gave a search strategy for $k = 2^i$ agents in which every agent moves in spiral paths around the origin but their paths don't overlap as shown in Figure 4. They analyzed the total time taken to locate the object as well as the total effort put in by the agents which is the total distance traversed by them compared to the shortest path from the origin to the object. In Chapter 6, we compare the performance of our algorithms with their algorithm.

Another related problem that is receiving interest recently is the so-called *evacuation problem*[37, 38]. In this problem, a team of robots is looking for an *exit* to a

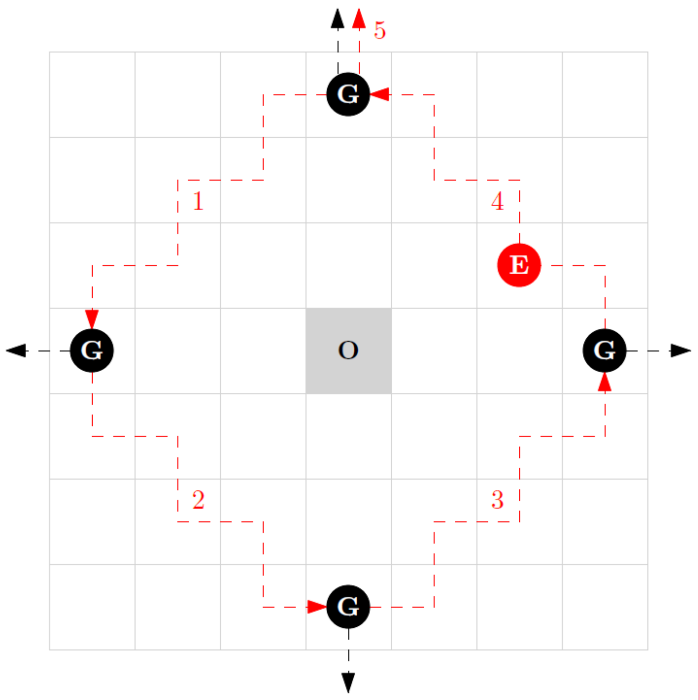


Figure 4: Four guides(G) and one Explorer(E) searching the grid using *RectSearch* in [1]

region. When a robot finds the exit, it must communicate the location of the exit to the remaining robots, which then move towards the exit. The performance metric is the time taken for the *last* robot to reach the exit. Two different communication models have been proposed: the *face-to-face* model, in which two robots can only communicate when they are at the same location, and the *wireless* model, in which two robots can communicate instantly regardless of their positions in the region. Our work assumes the face-to-face model, and is similar to the evacuation problem in the sense that we ask that all robots reach the food/exit. However, we ask not just that all robots reach the food/exit, but that they transport food back to the nest.

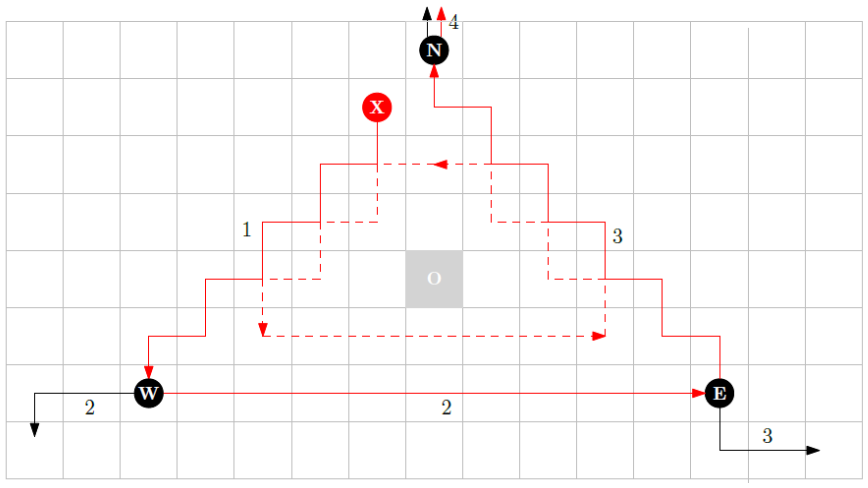


Figure 5: Four agents searching the grid using *TriangleSearch* in [2]

2.3 Miscellaneous multi-agent problems

The domain of multi-agent systems is not only limited to the foraging and search operations but also include problems such as pattern formation[39, 4], uniform scattering on a grid[40], map building[41], blocks construction[42] among various others.

In pattern formation problems, the main goal is usually to form a visually coherent pattern by multiple agents. The main task towards the solution of this problem is to assign final positions to individual agents in the swarm which would amount to the whole pattern. After the positions has been assigned, it is up to the agents to reach there from their initial positions avoiding collisions on the way. The positioning of the agents should be done in such a way that every agent moves only as little as required from its current position to form the bigger pattern at hand. This can be useful in variety of situations such as deployment of sensors in an unknown field, selection of a leader based on the geographical positions of the agents among many others.

Gordon et al [43] studied the problem of pattern formation by a swarm of agents on a grid with limited communication capabilities. They took inspiration from the biological system as well, the dancing bees to be precise. They exploited the fact that bees are known to communicate their findings by performing dance-like motions.

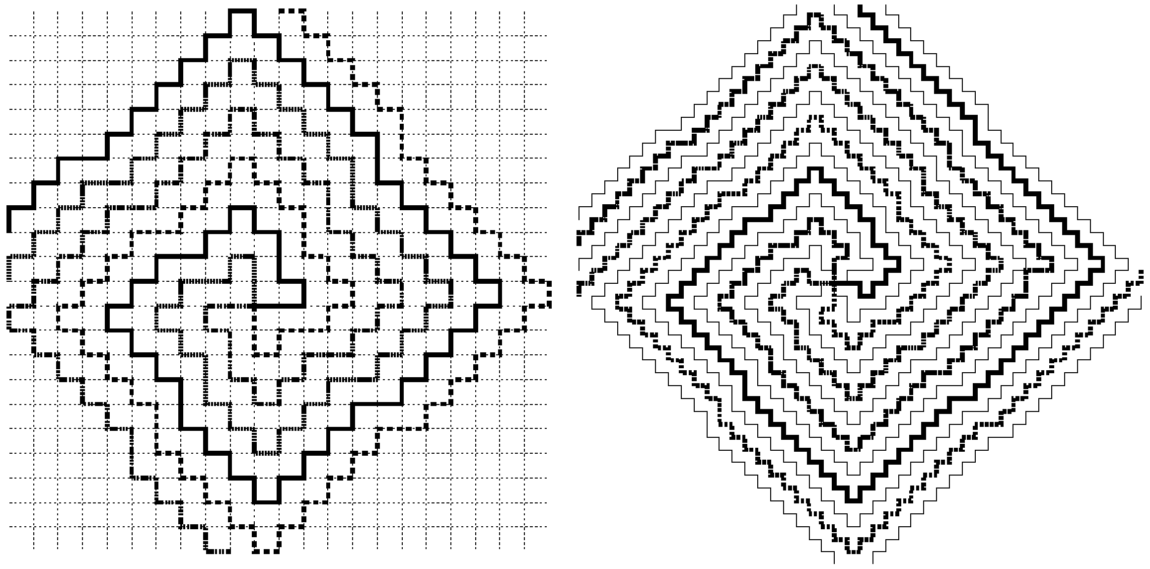


Figure 6: Grid covered by 4 and 8 agents using the search pattern introduced in [3]

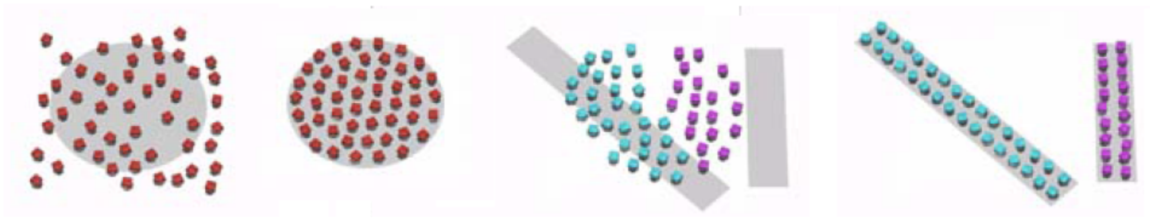


Figure 7: Before & After snapshot of 50 agents perform the *Pattern-Formation* [4]

In their algorithm, every agent calculates their $CenterofMass(COM)$ based on the positions of all the other agents and they try to move towards COM. Eventually, they will all be located in the COM cell and will use that cell as their origin. Once they are all set up at one place, they agree on x-axis, y-axis and total ordering by performing a series of little *dances*. Then, the formation of pattern is left a trivial task. Every agent i goes to a unique position q_i on the pattern.

Construction is yet another problem which can benefit from automation by robots immensely, especially in dangerous terrains. Nagpal et al [20] have been vigorously studying the construction problem using a swarm of autonomous robots and different kinds of blocks ranging from just inert to highly intelligent blocks. In their algorithm, agents do not communicate between themselves but rely on the information gained from the partially built structure and the final user-specified structure.

2.4 Robotic hardware

There has been a lot of research happening in the field of robotics hardware. Researchers are particularly looking to make the robots smaller in size and cheaper in cost and provide them with more sophisticated sensors and actuators at the same time. This piqued the interests of computer scientists to work on the distributed algorithms for swarms of robots. We conduct our experiments on a simulator. However, the algorithms we give can be easily implemented by the actual robots. The actions we require them to perform include walking on the grid, basic odometry skills, sense for food, short-range one-to-one communication, picking and dropping of one food unit at a time.

Mondada et al[6] developed a miniature robot, *s-bot* which was approximately 116 mm in diameter. The main focus of this project was inter-robot communication and the level of team work that could be productively exploited. The s-bots were provided with a gripper which was used to attach themselves to other s-bots in the team. They called this artifact as a *Swarm-bot* in which multiple s-bots were physically interconnected in order to adapt to their environment. For example, to overcome a cavity that is not doable by an individual s-bot, to lift a heavy substance, or when they are required to follow a common path as shown in Figure 8.

There is yet another project building on the concept of *swarm-bot* called *swarmanoid* [7] in which they emphasized that the insistence on the homogeneity of the agents in a swarm is holding back the researchers to further exploit swarm intelligence. *Swarmanoid* was introduced to curb this problem and consists of three distinct robot types - *foot-bots*, *hand-bots* and *eye-bots*. As their names would suggest, *foot-bots* are locomotory robots specializing in the transportation of goods, traversing even and

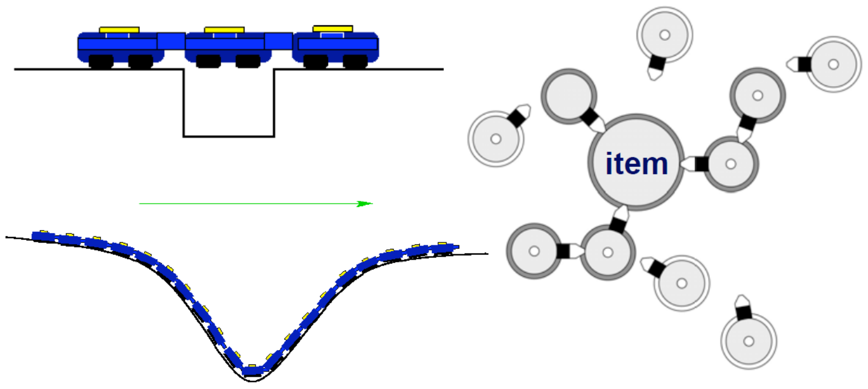


Figure 8: Tasks performed by a *swarm-bot* [5]

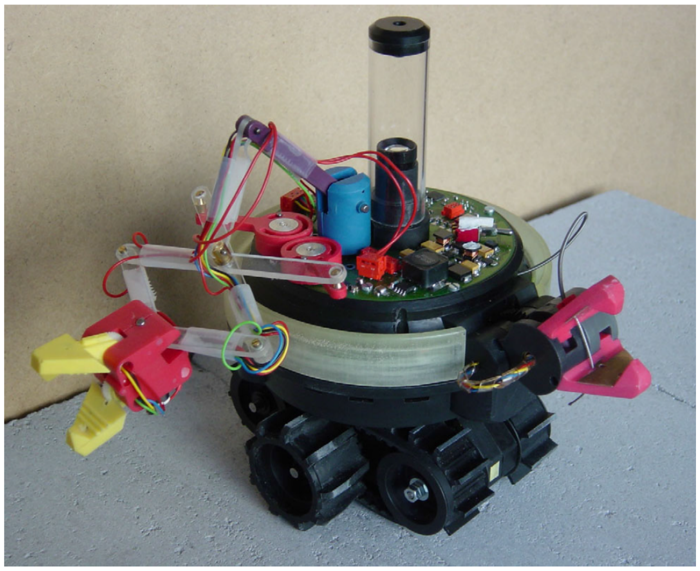


Figure 9: The *s-bot* robot[6]

uneven terrains, *hand-bots* are the robots used for vertical climbing, working on small artifacts and picking up and dropping off of items, *eye-bots* are the robots used to visually analyze the environment. It is quite clear that having specialized robots in the swarm could really change the way the problems are being solved in the swarm robotics field. This principle is still very new and is expected to be utilized in solving various swarm-related problems in the near future.

The e-puck robot, shown in figure 11, was developed by Martinoli et al [8] in 2009 as an open tool for educational purposes. It is capable of performing all the actions

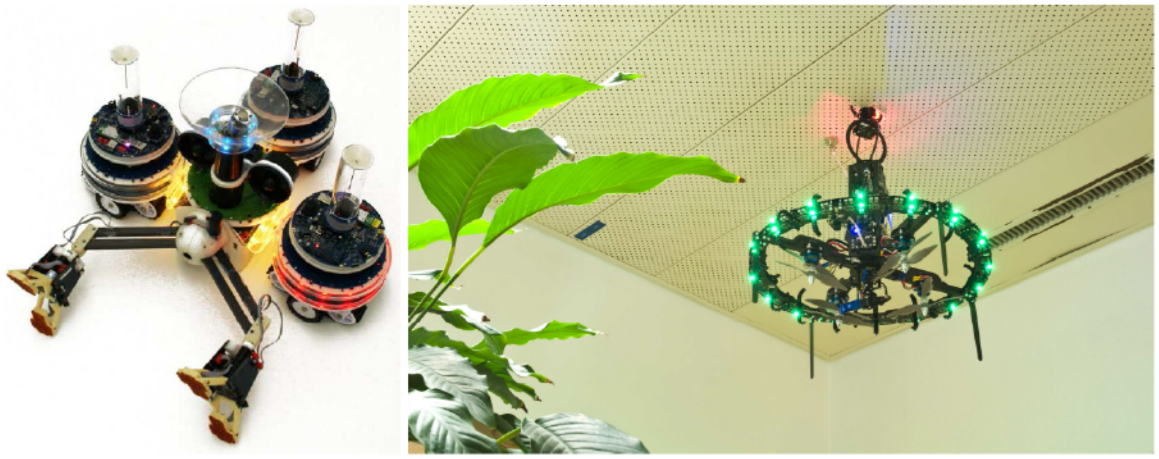


Figure 10: The *swarmanoid* robots: *one* hand-bot with *three* foot-bots & *one* eye-bot attached to the ceiling to monitor the environment [7]

required for the foraging task and is very easily available. It is sold for €550 by most companies. It has a micro-controller with 8 kB of RAM and 144 kB of flash memory which is sufficient for the required computations and the amount of information we need them to store. It has the infrared proximity sensors embedded which helps to identify its neighbors in the adjacent locations. Various mechanic extensions are also available in case we require to incorporate complex functionalities later.

Kilo-bot is yet another miniature robot developed by Nagpal et al [9], shown

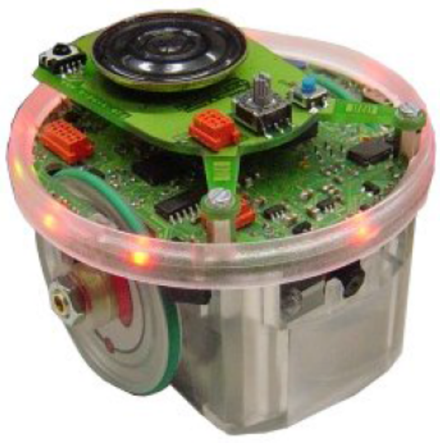


Figure 11: The *e-puck* robot[8]

in Figure 12, which is capable of carrying out complex tasks in a large collective.

This robot was specifically designed for educational purposes with its hardware parts costing only $14 and assembly time of just 5 minutes. It is capable of performing the basic operations expected of a swarm robot such as moving, rotating, communicating with nearby robots, measuring bearing to the agents in proximity and possessing enough memory to perform basic computations.

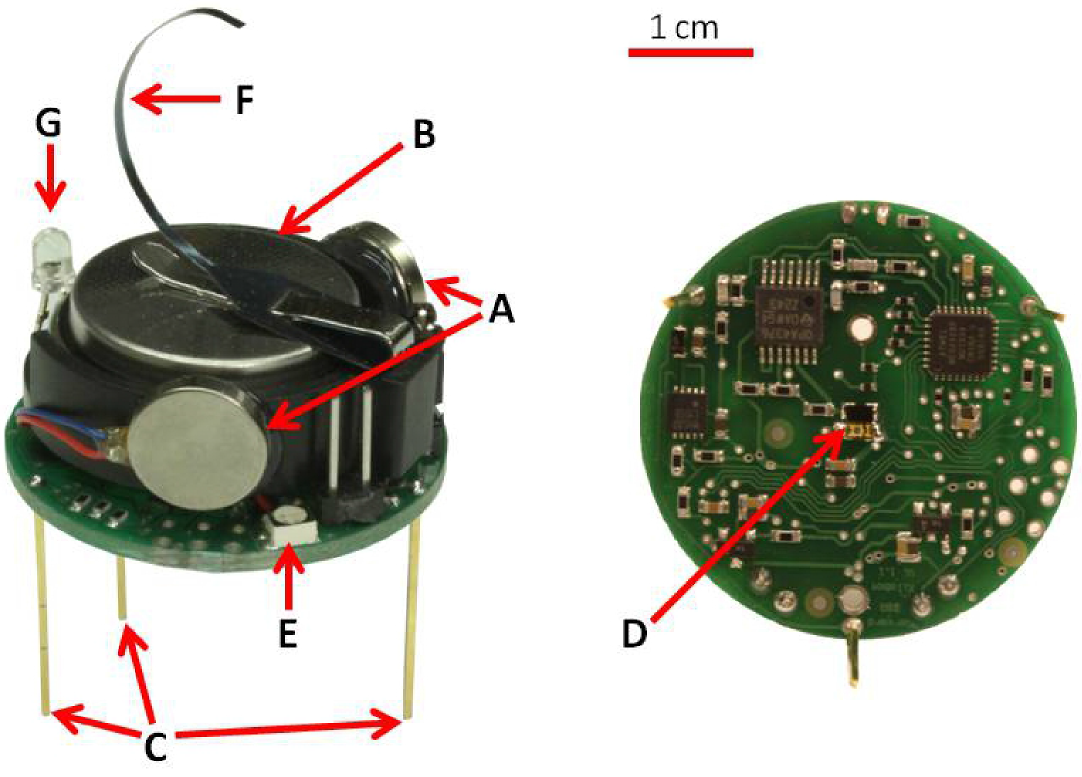


Figure 12: Kilo-bot in [9]

2.5 Challenges in our work

The main challenge of swarm robotics in general is the inter-robot coordination. A solitary robot in the field might not produce the desired results. Particularly for the task of foraging, it is strikingly clear that having multiple agents on the field speeds up the process. To achieve that, the agents should be able to share their findings with their fellow agents. Parallelism can not be achieved if only a single agent is aware of the location of the food. There should be a way for this information to get propagated throughout the swarm since the agents are not capable of broadcasting any information. They can only communicate with agents in their immediate proximity. Designing algorithms for exploration by multiple agents along with facilitation of communication strategy is the most challenging part of our work. Since communication is considered as an expensive task in the robotics community, we try to keep it as minimum as possible.

Chapter 3

Exploration Phase

Exploration Phase is the most crucial of all phases in the foraging problem. As its name would suggest, it involves searching the grid for the food source by a swarm of agents which is centrally located initially. They have no preliminary knowledge of the size of the grid. The algorithms that we give in this chapter basically give a particular pattern for the agents to follow to cover the grid. There are two parts to the decision making process of following a pattern:

- The first one incorporates the strategy of fairly partitioning the square grid among the agents of the swarm.
- The second part is about choosing the right movement pattern to cover the ground.

Evidently, the agents are required to know the grid size in order to partition the grid equally among themselves. They determine it by moving to any of the four borders following a straight path and doubling the number of steps to get there since they are all aware that they are located at the nest initially which is known to be in the middle of the grid. They move out of the nest in a pre-determined order. Since we assume that they are numbered, there is no ambiguity in the ordering of the agents.

Once the size of the grid is known, they divide the grid into equal sized regions. Every agent knows exactly which region it is supposed to cover and the exact location from where to start the exploration in that region. We call this location as the *Start-Cell* of the region in the remainder of this chapter. The phase starting from the time when the agents start moving out of the nest till they reach their *Start-Cell* is known as *Initial-Deployment Phase*. After an agent has taken its position at *Start-Cell*, it starts exploring the assigned region following a particular pattern. This phase is known as the *Exploration Phase*. We give *five* novel algorithms to explore the grid in the following sections. Please note that in the remainder of this chapter, we assume that in a grid of size $g \times g$, the x-coordinate and the y-coordinate of the cells lie in the interval $[0, g-1]$ where the cell $(0,0)$ is located at the top-left corner of the grid and $(g-1, g-1)$ at the bottom-right corner.

3.1 Strips

As is clear from its name, the square grid is partitioned into equal-sized rectangular bands in *Strips*. The pre-requisite for this algorithm to work is that the grid size should be a multiple of the number of agents. Every agent is assigned a single strip to cover. After determining the grid size by making the trip to the border, every agent divides it by their total population to ascertain the strip size.

As shown in Figure 13, the agents move out of the nest one at a time in a single direction and depending upon the chosen direction, the grid is divided into either horizontal or vertical strips. We assume, for simplicity, that the agents move out from the nest in *West* direction. [1] In this case, they partition the grid into horizontal strips and start the search from the west end and go towards the east end of the grid. Depending upon the assigned number to the agent, it knows which strip

[1]It can be easily adjusted for other cardinal directions.

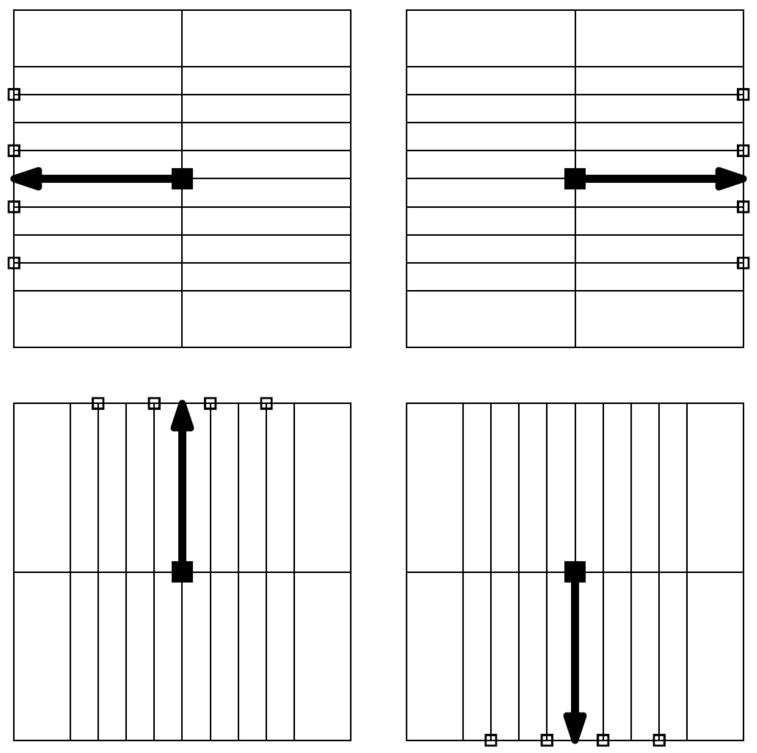


Figure 13: Division of the grid depending upon the direction in which the agents move out from the nest : Horizontal Strips when EAST/WEST & Vertical Strips when NORTH/SOUTH

it has to cover. Our goal was to design a pattern for the agents such that they meet their neighbors regularly while exploring the environment so as to facilitate quick propagation of food location information throughout the swarm. All the agents in the swarm need to start the exploration at the same time to achieve this.[2]

Every strip has its *Start-Cell* on the intersection of the grid-border and the strip-border from where the assigned agent starts searching the region. The first strip has its *Start-Cell* at the bottom corner and the second one has it at the top corner making them adjacent cells across the strip-border. Similarly, the rest of the strips also have their *Start-Cell*s marked in this fashion. If the number of agents is even, every agent has a partner in the adjacent strip as shown in figure 14 but if it is odd, there is one

[2]Two agents meet on their shared strip border only if they start the exploration together, i.e., follow the mirror image of each other's movement pattern in its own strip.

agent who has its *Start-Cell* at the bottom corner of the grid with no partner as shown in figure 15. However, this does not affect the execution of the algorithm.

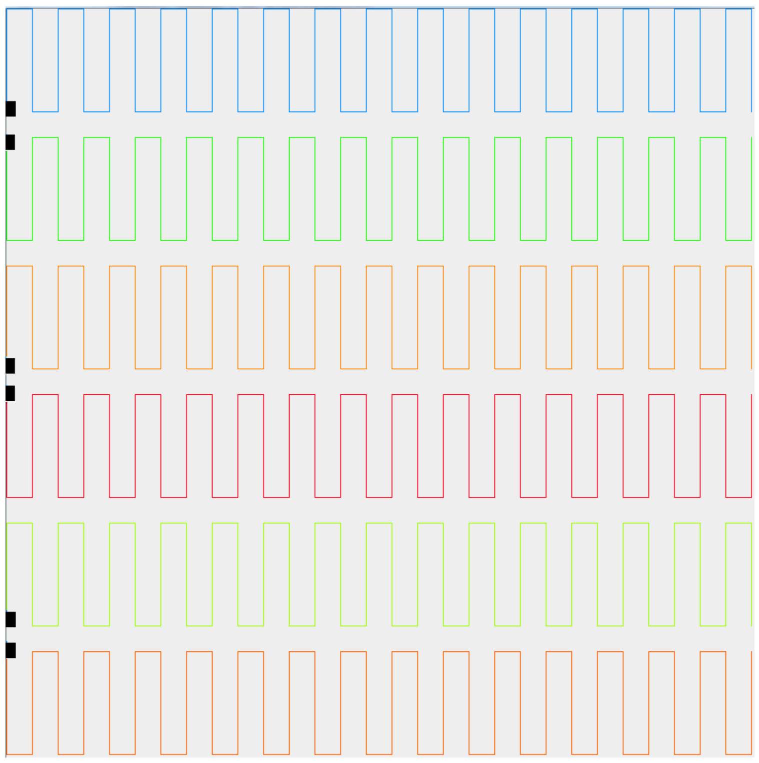


Figure 14: Positions of *Start-Cells* and the path followed by agents in Strips when n is even

When an agent touches the border and determines the strip size, it also calculates the list of *Start-Cells* and sort them in descending order of their distance from the location where it touched border. The agent which comes out first from the nest gets the strip with the farthest *Start-Cell*, the second agent gets the next in the list and so on. Now that the agents know their *Start-Cell*, they take the shortest path to reach there i.e. the path along the grid border. Since the first agent to come out of the nest gets the farthest strip to cover, it is the last agent to complete its *Initial-Deployment* phase. It is at this time step that all the agents enter into the *Exploration Phase* together.

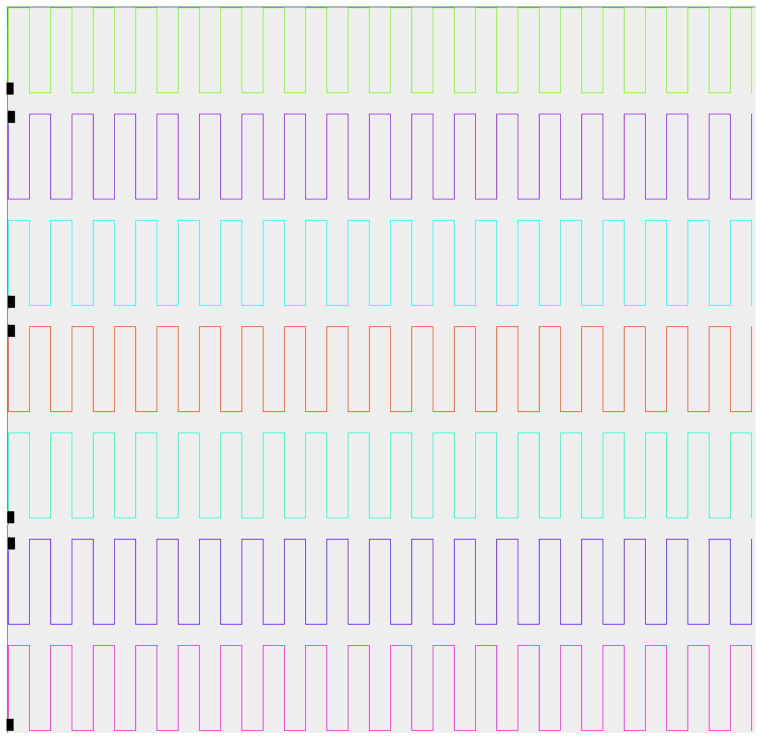


Figure 15: Positions of *Start-Cells* and the path followed by agents in Strips when n is odd

Let g denote the grid size, n the total number of agents and T_{dep} the time step at which the last agent completes its *Initial Deployment* phase, and hence, the time step at which all the agents enter into *Exploration* phase. Then, referring to Figures 16 and 17, T_{dep} is given by:

$$T_{dep} = \begin{cases} g-1 & \text{when } n \text{ is odd} \\ g(1-1/n) & \text{when } n \text{ is even} \end{cases}$$

Depending on the corner on which an agent's *Start-Cell* is located, it either moves in the *North* or *South* direction. Suppose an $Agent_A$ has its *Start-Cell* on the bottom corner of the strip. First, it moves in the *North* direction for *strip* number of steps where $strip = \frac{g}{n}$. When it touches the border of the strip, it moves one step

Algorithm 2 *StripsInitialDeployment* ($Agent_A$)

```
1: Start moving in $WEST$ direction at time step $Agent_A.id$ till the border is encountered.
2: $g \leftarrow 2$ * Distance traveled from the nest to border
3: $strip \leftarrow g/n$
4: $Start\text{-}Cell \leftarrow$ GETSTARTPOSITION–STRIPS($strip, n, Agent_A$)
5: Go to $Start\text{-}Cell$.
6: if $n$ is odd then
7:     Wait till $T_{g-1}$.
8: else
9:     Wait till $T_{g(1-1/n)}$.
10: end if
11: STRIPSEXPLORATIONPHASE($Start\text{-}Cell, strip$).
```

Algorithm 3 *GetStartPosition-Strips* ($strip, n, Agent_A$)

```
1: $List_S \leftarrow nil$
2: while $List_S.size \neq n$ do
3:     $i \leftarrow strip - 1$
4:     ADD $cell(0, i)$ to $List_S$
5:     if $List_S.size = n$ then exit loop.
6:     ADD $cell(0, i + 1)$ to $List_S$
7:     $i \leftarrow i + 2 * strip$
8: end while
9: Sort cells in $List_S$ in descending order o f their distance from the current position.
10: return $Start\text{-}Cell$ from $List_S$ corresponding to $Agent_A.id$.
```

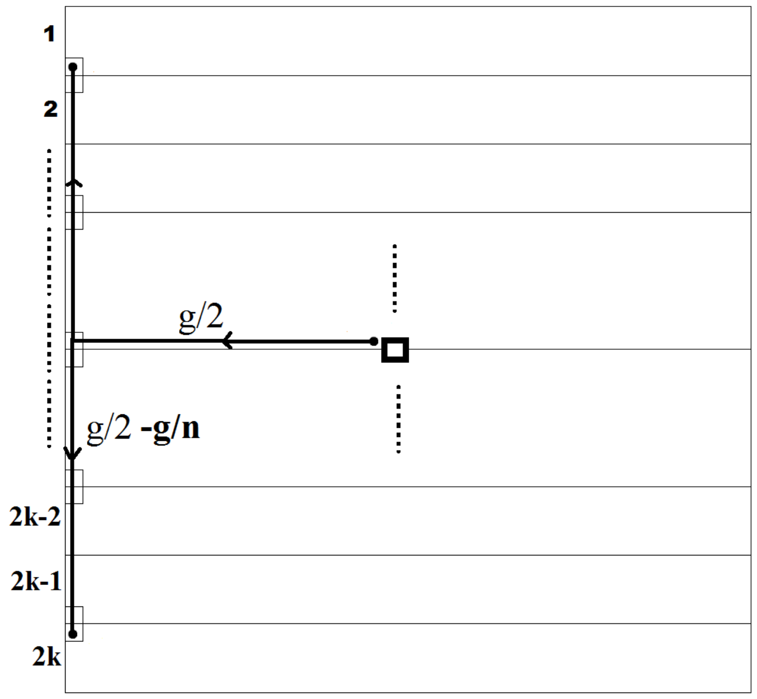


Figure 16: Farthest $StartCell$ when n is even

in the $East$ direction and then $strip$ number of steps in the $South$ direction. It keeps repeating this process either until it is aware of the food location or its has finished searching its region. If $Start\text{-}Cell$ was at the top corner instead, the agent would have moved the mirror image of this movement pattern. Because of this reason, the partner agents meets regularly on the border after covering one $North(strip)$-$East(1)$-$South(strip)$ trip.

Note that Dir_k in Algorithm 4 means moving in Dir direction for k number of time-steps.

Theorem 3.1.1. *The competitive ratio of* Strips *is* $\Theta(g)$.

Proof. It is not difficult to see that the worst case is obtained when the food is in the same row as the origin at the opposite end from where the robots start after their initial deployment. Thus, the Nest-food distance is $\frac{g}{2}$, while the time taken by the algorithm is $g - \frac{g}{n} + \frac{g^2}{n}$ when n is even and $g - 1 + \frac{g^2}{n}$ when n is odd. Thus, the

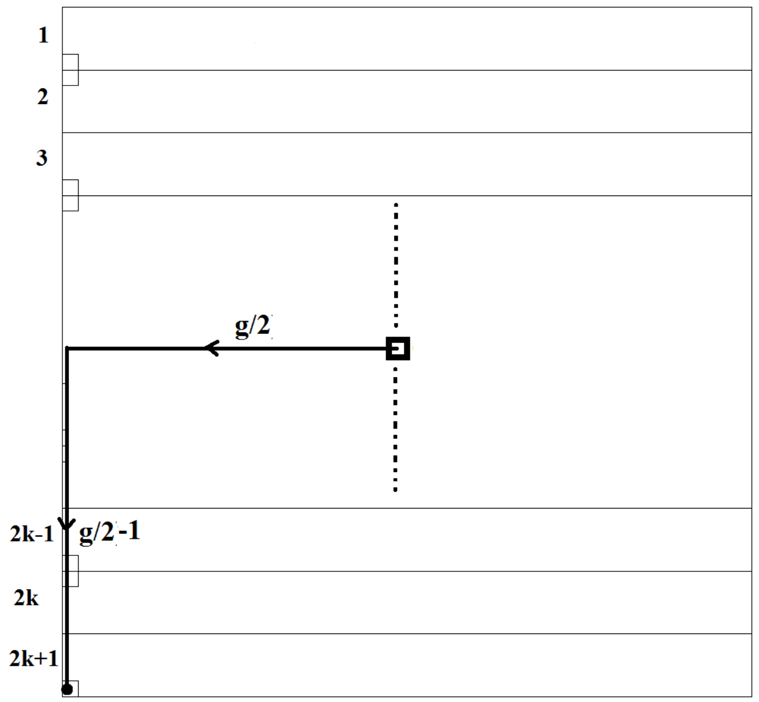


Figure 17: Farthest $StartCell$ when n is odd

Algorithm 4 $StripsExplorationPhase$ ($Start\text{-}Cell, strip$)

```
1: repeat
2:     if $Start\text{-}Cell$ is in bottom corner of the strip then
3:         Move $North(strip)$;$East(1)$;$South(strip)$;$East(1)$.
4:     end if
5:     if $Start\text{-}Cell$ is in top corner of the strip then
6:         Move $South(strip)$;$East(1)$;$North(strip)$;$East(1)$.
7:     end if
8: until $East$ border is encountered
```

competitive ratio is $\Theta(g)$ as claimed. □

3.2 ZigZag

We designed *ZigZag* keeping in mind that an ideal pattern should cover the region which is closer to the starting location before moving on to the farther regions as shown in Figure 19. But since diagonal movement in the grid is not allowed as per

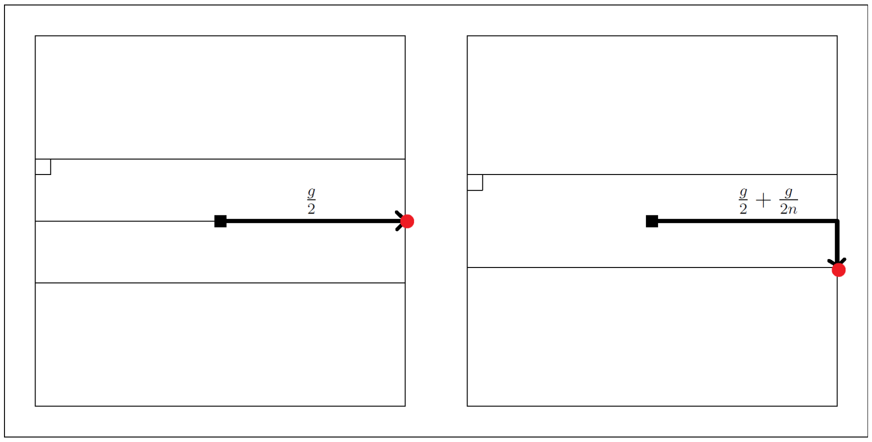


Figure 18: The cell with the highest competitive ratio when n is even and odd in *Strips*

our system model, we adjusted the zigzag motion for the agents.

Figure 20 shows the grid covered by a single agent using zigzag pattern which is

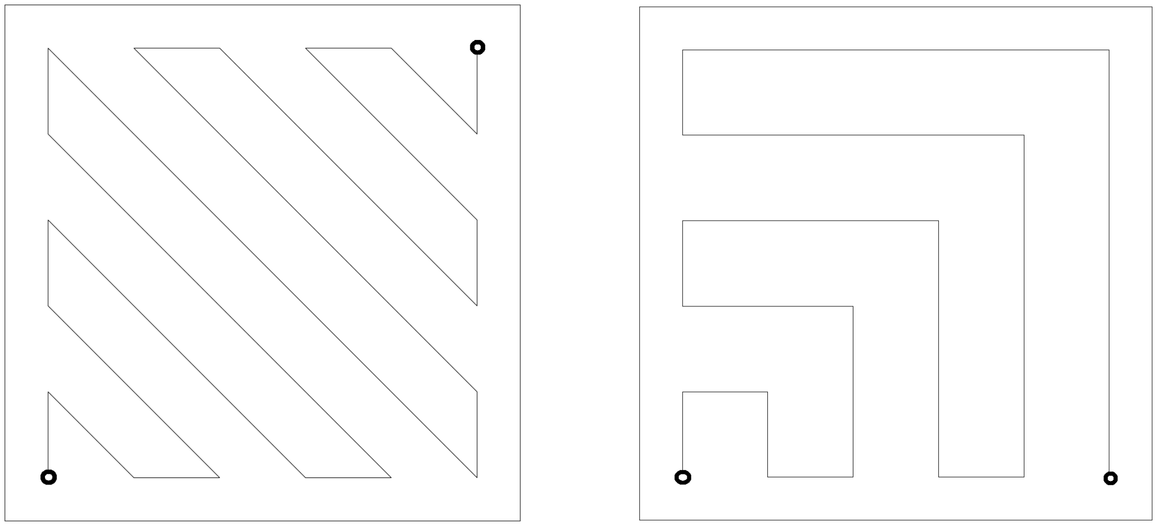


Figure 19: Original & adjusted *Zigzag* pattern

essentially spiraling around the nest. Following the same pattern by four agents on the grid would need them to start from the center of their respective quadrants as shown in Figure 21.

However with this strategy, the agents meet only towards the end of the search process. To facilitate quick communication between the agents, we introduced the

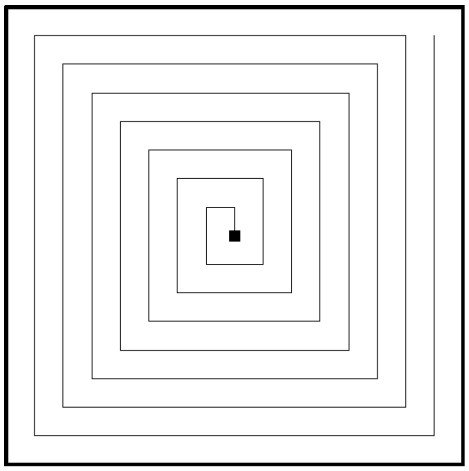


Figure 20: Grid covered by a single agent using *Zigzag* pattern

search pattern shown in Figure 22. The agents start the exploration from the nest itself and they meet two of their fellow agents regularly along the axes as shown in Figure 22. Hence, *Initial-Deployment Phase* is not needed at all. This pattern only works when the number of agents is a power of 4. We only divide a square region into four further square sub-regions. Following this, the next possible configuration is having 16 agents in the field. Each quadrant is divided into four squares as shown in Figure 23 and the *Start-Cells* are present on the point of intersection of those 4 squares. In order to determine and reach these *Start-Cells*, the agents are required to know the size of the grid. They use the same technique as in *Strips* to determine the size of the grid i.e. moving to the border of the grid and setting the grid size as twice the distance traveled to get there.

As shown in the figure 24, the time step at which the last agent completes its *Initial Deployment* phase is given by:

$$T_{dep} = 3 \times \frac{g}{2} - 2 \times \frac{g}{\sqrt{n}}$$

The movement of the agents depend on whether they move out horizontally or vertically from their *Start-Cells*. Figure 25 shows different codes agents follow to have

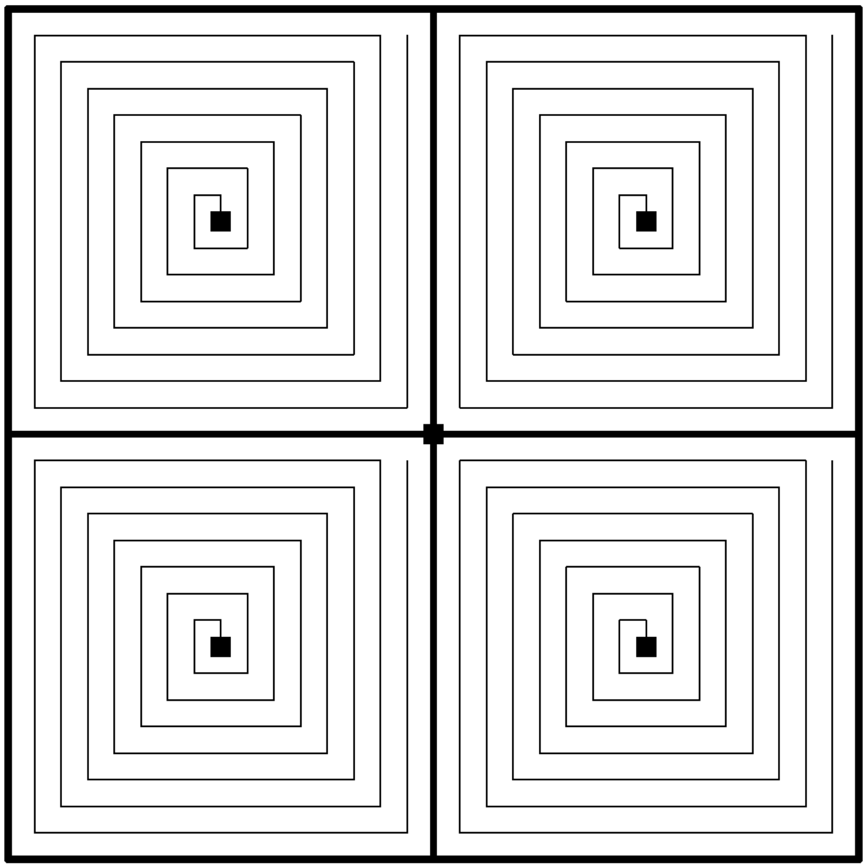


Figure 21: Grid covered by four agents using *Zigzag* pattern

symmetric movement in their respective regions.

3.3 Hilbert

Hilbert Curve [44] was first introduced in 1891 as a continuous fractal space-filling curve. It maps 2D space into a 1D line in such a way that the line never goes in the same direction for more than three steps, never revisits a site and still manages to cover every point in the space. A fractal [17] is defined as a geometric pattern that is repeated at every scale. The basic building block of this curve is an open square with three connected lines [45] which will cover *four* cells if drawn on a grid as shown in Figure 26 and the pattern is built using these multiple open squares in a certain

Algorithm 5 *ZigZag* ($Agent_A$)

```
1: Start moving in $WEST$ direction at timestep $Agent_A.id$ till the border is encountered.
2: $g \leftarrow 2$ * Distance traveled from $Nest$ to border
3: $size \leftarrow \frac{g}{\sqrt{n}}$
4: $Start\text{-}Cell \leftarrow$ GetStartPosition−Hierarchical$(g, n, Agent_A)$
5: Go to $Start\text{-}Cell$.
6: Wait till time $T_{3\times\frac{g}{2}-2\times size}$.
7: $i \leftarrow 1$
8: while $i \neq size$ do
9:     Follow Direction(i) for $Agent_A.code$ from Table 1
10:    $i \leftarrow i + 2$
11: end while
```

Algorithm 6 *GetStartPosition-Hierarchical* $(g, n, Agent_A)$

```
1: $List_S \leftarrow nil$
2: $size \leftarrow \frac{g}{\sqrt{n}}$
3: while $List_S.size \neq n$ do
4:     for $j = size - 1$ to $g$ STEP $2 \times size$ do
5:         for $i = size - 1$ to $g$ STEP $2 \times size$ do
6:             ADD $cell(i, j)$ to $List_S$
7:             ADD $cell(i + 1, j)$ to $List_S$
8:             ADD $cell(i, j + 1)$ to $List_S$
9:             ADD $cell(i + 1, j + 1)$ to $List_S$
10:        end for
11:    end for
12: end while
13: Sort cells in $List_S$ in descending order of their distance from the current position.
14: return $Start\text{-}Cell$ from $List_S$ corresponding to $Agent_A.id$.
```

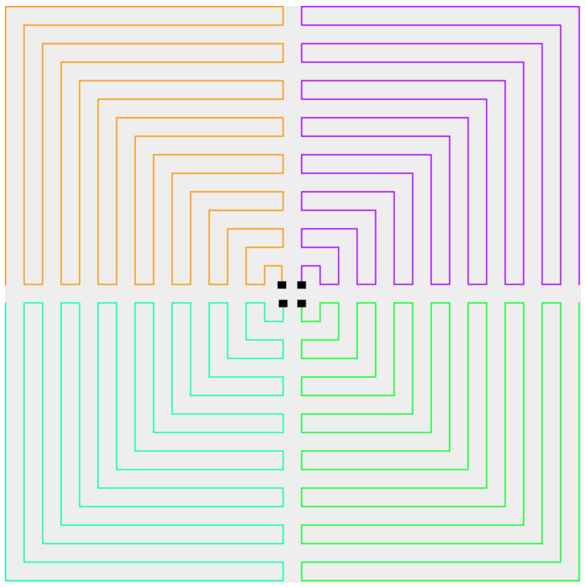


Figure 22: Grid covered by four agents using *Zigzag*

Code	Direction(i)
1	$E - N_i - W_i - N - E_{i+1} - S_{i+1}$
2	$W - N_i - E_i - N - W_{i+1} - S_{i+1}$
3	$S - E_i - N_i - E - S_{i+1} - W_{i+1}$
4	$S - W_i - N_i - W - S_{i+1} - E_{i+1}$
5	$N - E_i - S_i - E - N_{i+1} - W_{i+1}$
6	$N - W_i - S_i - W - N_{i+1} - E_{i+1}$
7	$E - S_i - W_i - S - E_{i+1} - N_{i+1}$
8	$W - S_i - E_i - S - W_{i+1} - N_{i+1}$

Table 1: Code-Direction Table for Zigzag

manner repetitively. Figure 27 shows expanded version of the open squares in Figure 26.

Hilbert algorithm follows the same strategy as that of *Zigzag* for the division of grid and the assignment of respective *Start-Cells*. Hence, the initial deployment phase is the same for both the patterns. They differ in the way the agents move to cover the ground after they have reached their respective *Start-Cells*. The prerequisites for following the *Hilbert* algorithm is that the grid size should be a power of 2 and the number of agents should be a power of 4. We gave a recursive solution

Algorithm 7 Hilbert Algorithm ($Agent_A$)

```
1: Start moving in $WEST$ direction at timestep $Agent_A.id$ till the border is encountered.
2: $g \leftarrow 2$ * Distance traveled from $Nest$ to border
3: $Start\text{-}Cell \leftarrow$ GetStartPosition−Hierarchical($g, n, Agent_A$)
4: Go to $Start\text{-}Cell$.
5: $size \leftarrow \frac{g}{\sqrt{n}}$.
6: Wait till time $T_{3\times\frac{g}{2}-2\times size}$.
7: HilbertExplorationPhase($g, n, Agent_A$)
```

Algorithm 8 *HilbertExplorationPhase*($g, n, Agent_A$)

```
1: $size \leftarrow \frac{g}{\sqrt{n}}$
2: $order \leftarrow 2 \times \log_4(size)$
3: $codelist \leftarrow nil, route \leftarrow nil$
4: $codelist.add(A.code)$
5: $route.add(START)$
6: while $order \neq 0$ do
7:     $route \leftarrow$ GetRoute($codelist, route$)
8:     $codelist \leftarrow$ GetSubCodes($codelist$)
9:     $order \leftarrow order - 1$
10: end while
```

Algorithm 9 *GetRoute* ($codelist, route$)

```
1: $updatedRoute \leftarrow nil$
2: for $i = 1$ to $codelist.size$ do
3:     Add $Directions(route[i])$ for $codelist[i]$ to $updatedRoute$.
4: end for
5: return $updatedRoute$
```

Algorithm 10 *GetSubCodes* ($codelist$)

For every *code* in the *codelist*, this algorithm returns 4 sub-codes.

```
1: $sublist \leftarrow nil$
2: for $i = 0$ to $codelist.size$ do
3:     $code \leftarrow codelist[i]$
4:     if $code \leq 4$ then
5:         Add $code + 4, code, code, 9 - (code + 4)$ to $sublist$.
6:     else
7:         Add $code - 4, code, code, 9 - (code - 4)$ to $sublist$.
8:     end if
9: end for
10: return $sublist$
```

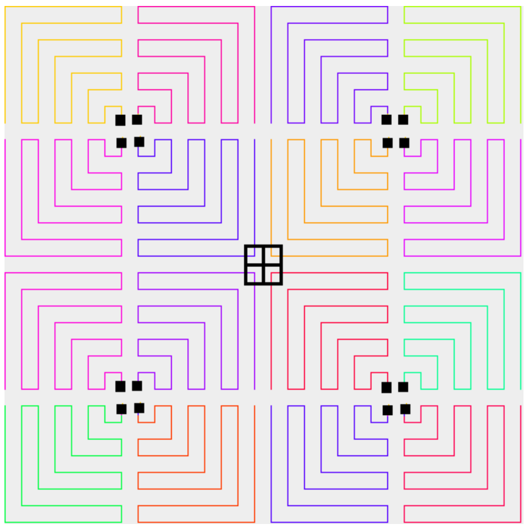
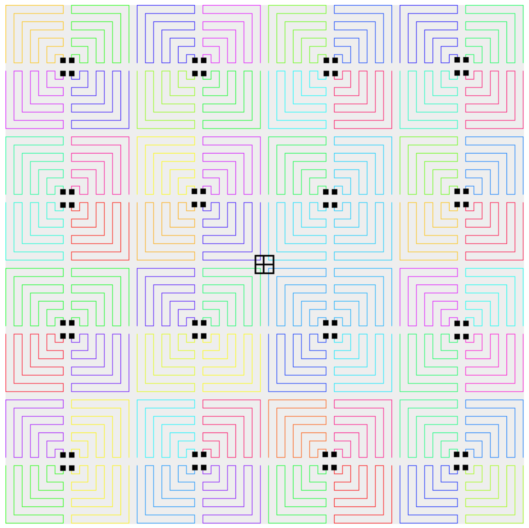



Figure 23: Grid covered by 16 and 64 agents using *ZigZag Algorithm*

Code	Directions
1	$X - E - N - W$
2	$X - W - N - E$
3	$X - S - E - N$
4	$X - S - W - N$
5	$X - N - E - S$
6	$X - N - W - S$
7	$X - E - S - W$
8	$X - W - S - E$

Table 2: Directions(X,code) Table for Hilbert Algorithm

for the formation of the pattern given by *Hilbert* by using the same codes we used for *zigzag* shown in Figure 25 and the corresponding direction instructions given in Table 2 for every code. Suppose $Agent_A$ wants to follow $code = 3$ for covering a grid of size $8 * 8$. This code is repeatedly broken down into sub codes as shown in Figure 30. At the root node, sits the agent's code and the *Start-Cell* of that agent. This code is further broken into $four$ sub-codes using Algorithm 10 and the directions from the parent node are passed on to the child nodes. The leaf nodes are converted into a route for $Agent_A$ to follow by going from the leftmost node towards the rightmost node, and taking the directions for codes from Table 2. For example, when we give

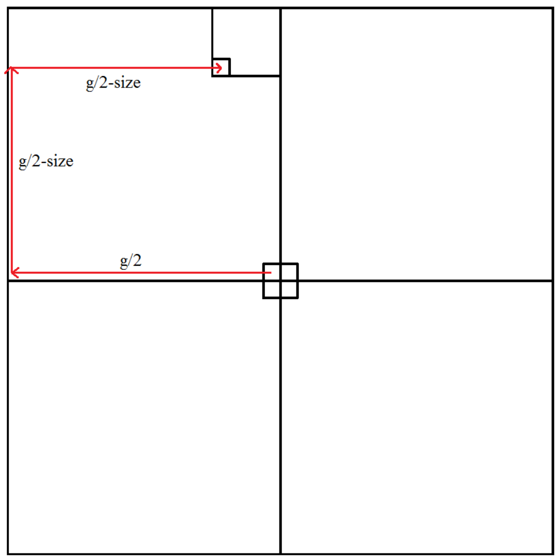


Figure 24: Path to the farthest $StartCell$ from the border

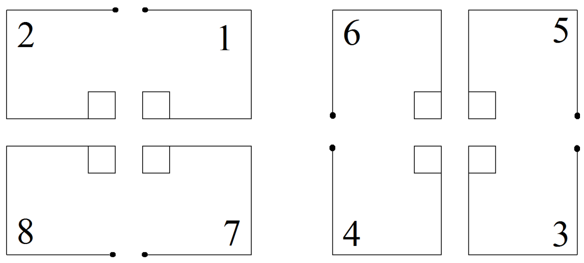


Figure 25: Codes for different orientations: Squares depict the starting positions of the agents

$3(St)$ as an input to Table 2, it gives us $St - S - E - N$ as the output, $7(E)$ gives $E - E - S - W$ and so on.

Theorem 3.3.1. *The competitive ratio of* Zigzag *and* Hilbert *is* $\Theta(g)$ *when the number of agents is* 4 *and* $\Theta(g^2)$ *when there are more than* 4 *agents.*

Proof. The worst case occurs when the food is placed in the same row as the origin at the end of the grid when there are only 4 agents as shown in Figure 31. Thus, the nest-food distance in this case is $\frac{g}{2}$, while the time taken by our algorithms is $\frac{g^2}{4}$ which proves the competitive ratio is $\Theta(g)$ as claimed.

When there are more agents, the worst case happens when the food is placed

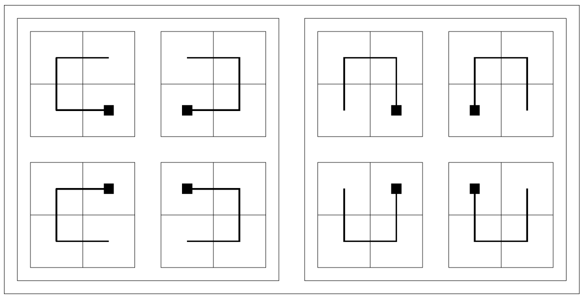


Figure 26: Hilbert's basic building blocks covering $four$ cells in a grid ($order = 1$)

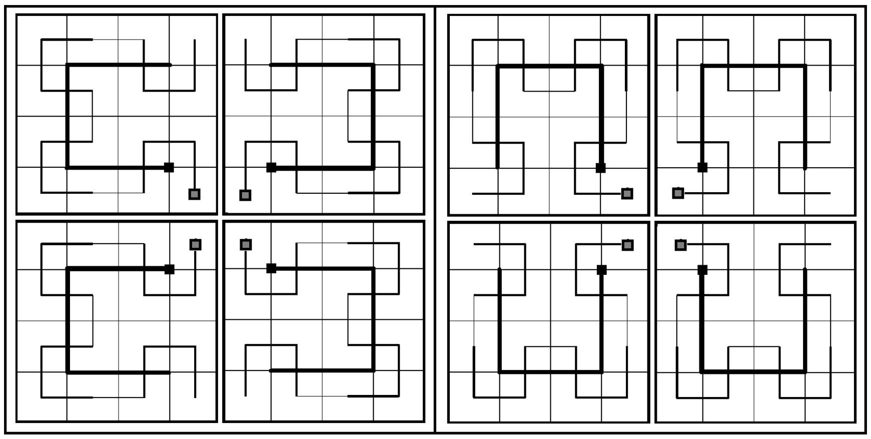


Figure 27: Hilbert curves of different orientations covering 16 cells in a grid ($order = 2$)

just one step away from the nest as shown in Figure 31, while the time taken by our algorithms is $\frac{3g}{2} - \frac{2g}{\sqrt{n}} + \frac{g^2}{n}$. Therefore, the competitive ratio in the worst case is $\Theta(g^2)$. $\square$

3.4 Sectors

The exploration strategies we have seen so far require the agents to take certain positions in the grid first, wait for a fixed amount of time and then start exploring the environment together. They are also required to be aware of the grid size beforehand to determine these positions for which they move to the border first. The motivation behind *Sectors* was to think of a way to get rid of this *Initial Deployment* phase

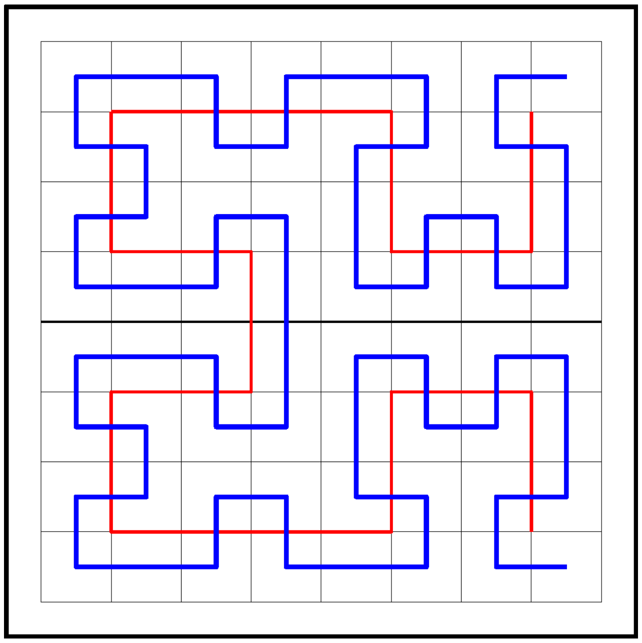


Figure 28: Hilbert curves of different orientations covering 64 cells in a grid ($order = 3$)

and start the exploration from the nest itself. It can also be seen as a different generalization of the *Zigzag* pattern.

3.4.1 Division of the grid into exploration areas for agents

The strategy suggested for the division of grid among agents is to cut it into sectors. We want these sectors to be equal in area so that there is fair division of the cells to be traversed among the agents. Every agent is responsible for covering its own sector by exploring the region around the nest first and then moving away to the farther regions. When there are only four agents in the field, division into sectors is equivalent to dividing the grid into 4 quadrants and the movement pattern to cover the ground is identical to that of *Zigzag* strategy for 4 agents. We use this pattern when the total number of agents is a multiple of 8 for symmetry purposes. However, it is easily adjustable for other numbers too. Every $Agent_A$ has a slope range $[\alpha, \beta)$ associated with it. $Agent_A$ only covers the cells in the grid whose slope w.r.t. the nest falls

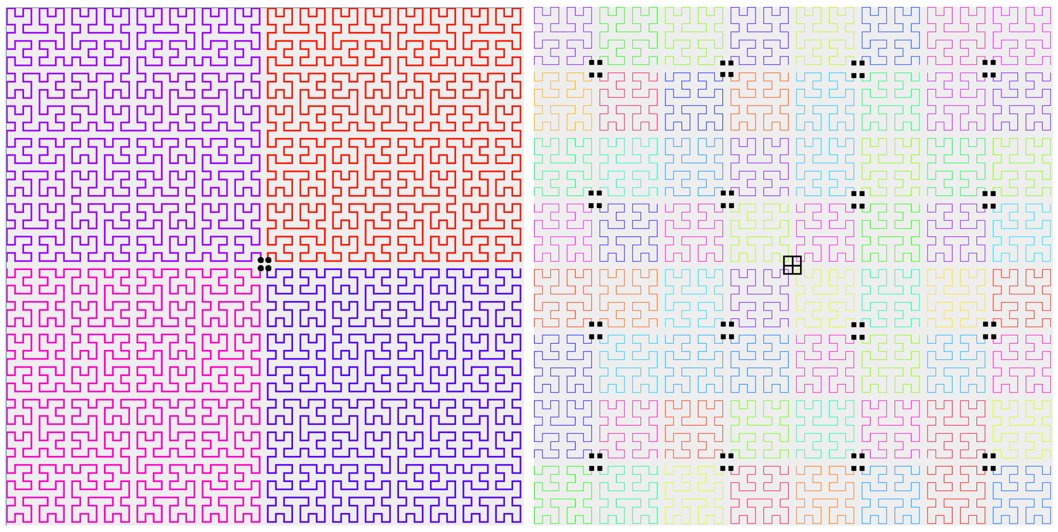


Figure 29: Grid covered by 4 and 64 agents following Hilbert Algorithm

$$\begin{tikzcd}
 & & & & & & & 3(St) \arrow[dllllll] \arrow[dll] \arrow[drr] \arrow[drrrrrr] & & & & & & & & \\
 & 7(St) \arrow[dl] \arrow[d] \arrow[dr] \arrow[drr] & & & & 3(S) \arrow[dl] \arrow[d] \arrow[dr] \arrow[drr] & & & & 3(E) \arrow[dl] \arrow[d] \arrow[dr] \arrow[drr] & & & & 2(N) \arrow[dl] \arrow[d] \arrow[dr] \arrow[drr] & & \\
3(St) & 7(E) & 7(S) & 6(W) & 7(S) & 3(S) & 3(E) & 2(N) & 7(E) & 3(S) & 3(E) & 2(N) & 6(N) & 2(W) & 2(N) & 3(E)
\end{tikzcd}$$

Figure 30: Steps involved in forming Hilbert Pattern when $Agent.code = 3$ for covering a grid of size $2^3 \times 2^3$

within this range. Suppose the total number of agents on the grid is $n = 8k$ implying there are $2k$ agents in each quadrant. Our aim is to partition every quadrant into $2k$ equal-sized triangles/sectors. We divide the quadrant into 2 triangles by connecting the origin to the opposite vertex and each of these triangles is further divided into k triangles having a base of length $\frac{1}{k}$ as shown in Figure 32. Clearly, since all the triangles have the same height and base, they are equal in area. We divide the *East* and *North* border of the first quadrant into k equal-sized segments each and connect the nest to the partitioning points to get $2k$ sectors of equal areas as shown in Figure 32. There are $2k-1$ lines dividing the sectors and their slopes will be given by $(0, \frac{1}{k}, \frac{2}{k}, ..., \frac{k-2}{k}, \frac{k-1}{k}, 1, \frac{k}{k-1}, ..., \frac{k}{2}, k)$. $Agent_0$ is responsible for covering the cells

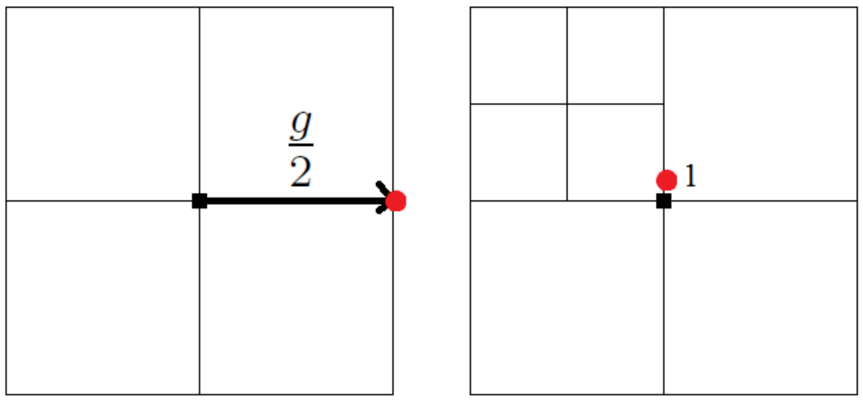


Figure 31: The cell with the highest competitive ratio when the number of agents is 4 and when there are more than 4 agents in *Zigzag* & *Hilbert*

whose slope falls in the range of $[\tan^{-1} 0, \tan^{-1} \frac{1}{k})$, $Agent_1$ gets the cells whose slope falls in the range of $[\tan^{-1} \frac{1}{k}, \tan^{-1} \frac{2}{k})$ and so on. Agents in the other quadrants also follow the same technique to determine their slope range.

3.4.2 Start-Cell determination

Every agent determines the nearest cell from the nest that falls into its range, i.e. its *Start-Cell* and also for the other agents using Algorithm 11.

Algorithm 11 Start Cell Assignment - Sectors ($Agent_A$)

```
1: $i \leftarrow 1$
2: while Start-Cell not assigned to every agent do
3:     $nlist \leftarrow$ i-hop cells
4:     for $j = 1$ to $nlist.size$ do
5:         Calculate slope of $nlist[j]$ w.r.t. origin.
6:         Find the agent in whose range this slope falls.
7:         Assign it as Start-Cell to that agent; if not already assigned.
8:     end for
9:     $i \leftarrow i + 1$
10: end while
```

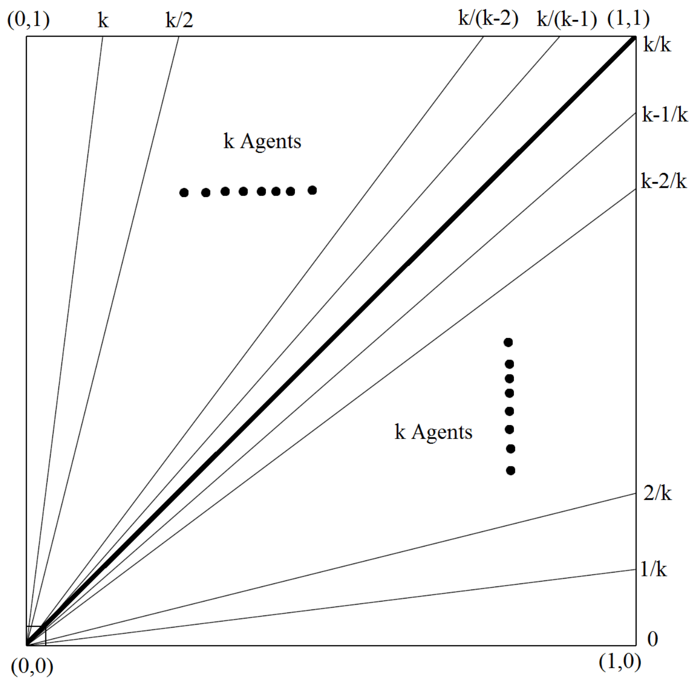


Figure 32: Division of a square into equal-sized sectors

3.4.3 Emission order of agents

The ordering scheme that was followed by the previously mentioned patterns does not work here since the distance of an agent's *Start-Cell* from the nest does not depend on the ID assigned to it. The agents whose *Start-Cells* are farther should come out of the nest earlier than the ones whose *Start-Cells* are nearer to the nest. Suppose $A_0, A_1, ..., A_7$ are the *Start-Cells* for 8 agents in a quadrant. This quadrant is further sub-divided into 2 sub-quadrants. The agents A_0, A_1, A_2, A_3 have their *Start-Cells* in the lower sub-quadrant lq and A_4, A_5, A_6, A_7 in the upper sub-quadrant uq. The agents sort the cells in lq in decreasing order of their horizontal distance from the nest and cells in uq in decreasing order of their vertical distance from the nest. After sorting, they get $lq : A_3, A_1, A_2, A_0$ and $uq : A_7, A_5, A_6, A_4$. Note that when horizontal/vertical distance is the same for two cells, they consider the other distance. Then, they intermingle these two sorted sub-lists of *Start Cells*. One agent comes

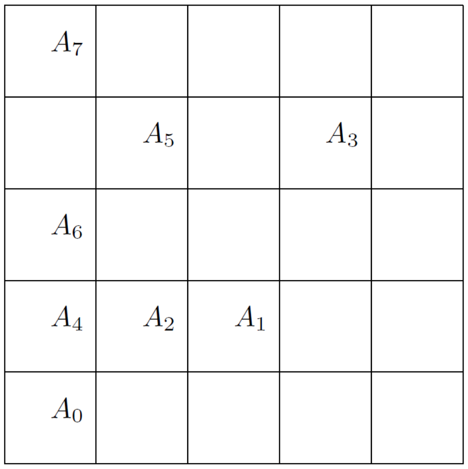


Figure 33: Start Cells of 12 sectors

out from the nest in *lq* followed by the next agent from *uq* and so on. Final emission order in this case is $A_7 \rightarrow A_3 \rightarrow A_5 \rightarrow A_1 \rightarrow A_6 \rightarrow A_2 \rightarrow A_4 \rightarrow A_0$. We prove later in this section that following this scheme for the emission order, there are no collisions between the agents.

3.4.4 Initial Deployment

After the agents know their order and the *Start-Cell*, they only move towards it using two directions depending upon the quadrant they are in. For example, agents in Figure 33 are present in the first quadrant, they go to their *Start-Cells* initially in *East* direction and turn towards *North* only when they reach the same column as their *Start-Cell*.

3.4.5 Exploration Strategy

After the grid has been divided into sectors, *Start-Cells* have been determined and the ordering of agents has been decided, the next step is to find a movement pattern to cover the ground. In the patterns we have seen so far, the grid was divided in rectangular regions and the movement patterns suggested covered the ground completely without any cells being visited more than once. There is no such definite movement pattern to cover all the cells when the grid is partitioned into sectors. If the agents follow some basic rules that we give, they cover the ground with only a few of the cells being visited twice. The first rule that the agents follow is to always move

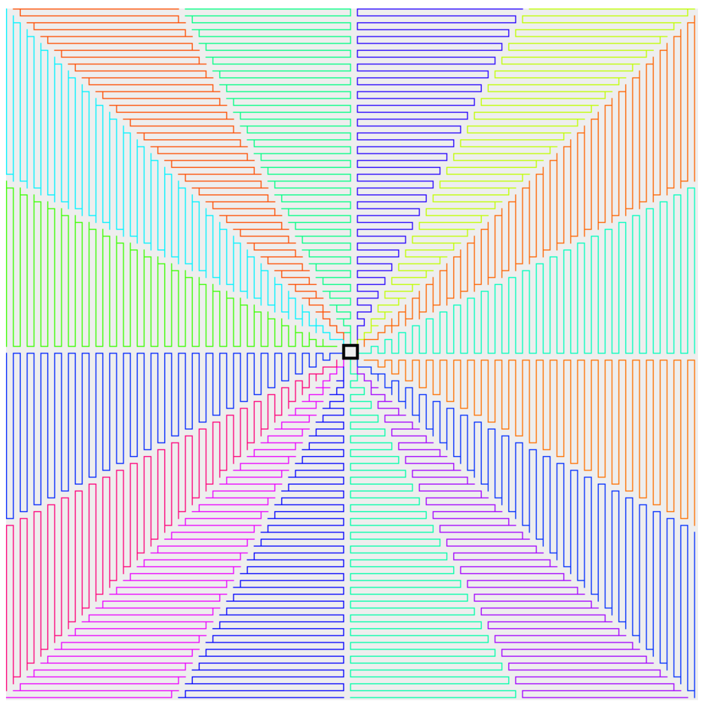


Figure 34: Division of a grid into 16 sectors

Sub-Quadrants	transition	forbidden	tlevel
11 & 42	*East*	*West*	*Vertical*
12 & 21	*North*	*South*	*Horizontal*
22 & 31	*West*	*East*	*Vertical*
32 & 41	*South*	*North*	*Horizontal*

Table 3: Parameter values for covering sectors

away from the nest. We use *tlevel*, short for *TransitionLevel*, to specify the level that should be covered first before moving away from the nest. The division of each

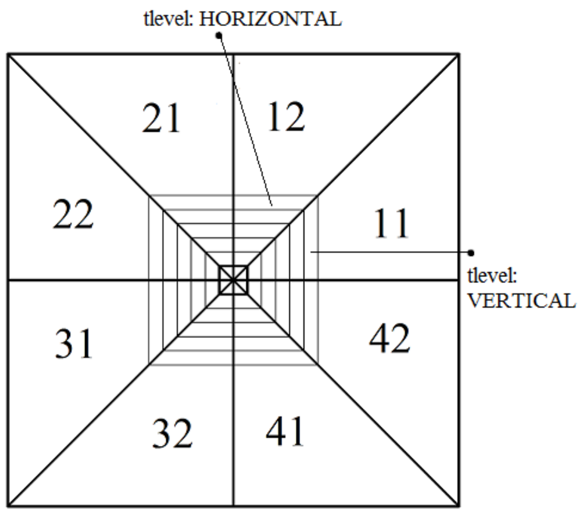


Figure 35: $tlevels$ and $Sub-Quadrants$ for the division of grid into sectors

quadrant into sub-quadrants and the corresponding *tlevels* are shown in figure 35. In sub-quadrants 11 and 42, *tlevel* is $Vertical$ meaning all the cells which have the same x-coordinate value as the cell in which the agent is located currently and fall within range should be visited before making the *transition* to the next level in *East* direction. The $forbidden$ direction for these sub-quadrants is $West$ since making a move in this direction will take the agent towards the nest. Similarly, *tlevel* for sub-quadrants 12 and 21 is $Horizontal$ meaning the agents in this region should visit the cells with the same y-coordinate values as the cell in which the agent is located currently before transitioning to the next *tlevel*.

Algorithm 12 Sectors Exploration Phase ($Agent_A$)

```
1: while Border is not encountered do
2:     Go to Start-Cell.
3:     Visit all the cells within range in the current tlevel.
4:     Determine the cells falling within range in the next tlevel.
5:     Make the move in transition direction to the next tlevel staying within the
       sector.
6: end while
```

3.4.6 Collisions

We designed *Initial-Deployment* phase of the *Sectors* algorithm in a way such that no two agents end up in the same place. Also, in the *Exploration* phase, agents don't step out of their sectors. However, at a given point of time, some agents could be in their exploration phase while some are still in their initial deployment phase. We claim that no collisions occur even then.

Refer to Figure 36 for possible collision scenarios in lower sub-quadrant of the first quadrant. In this case, the agents are only allowed to move in *East* while still in the same row as the nest and $North$ otherwise when in initial deployment and *East*, $North$ and *South* directions in exploration phase.

In Figure 33(a), $Agent_2$ can only be in exploration phase since it is moving in *South* direction. $Agent_1$ has to be in initial deployment phase if it is moving to the cell assigned to $Agent_2$ since agents don't step out of their own region in exploration. However, it is not possible that $Agent_2$ is already in its exploration phase while $Agent_1$ whose *Start-cell* is clearly farther than that of $Agent_2$, and hence should have moved out of the nest before $Agent_2$, has not even reached there yet. Therefore, this collision situation will never happen.

The situation shown in Figures 33(b),(c) and (f) will never happen because agents are not allowed to move in $West$ direction no matter the phase they are in.

In Figure 33(d) and (e), $Agent_2$ has to be in exploration phase because of the direction it is in while $Agent_1$ can only be in initial deployment phase if it is moving to a cell assigned to $Agent_2$. However, if that is the case, *Start-Cell* of $Agent_1$ is clearly farther from the nest than $Agent_2$ and it should have been gotten out of the nest before $Agent_2$ in which case these situations could never have arisen.

In all the other exploration strategies, it is quite clear that the agents can never collide. It is not that obvious in *Sectors*. However, the above discussion led us to the

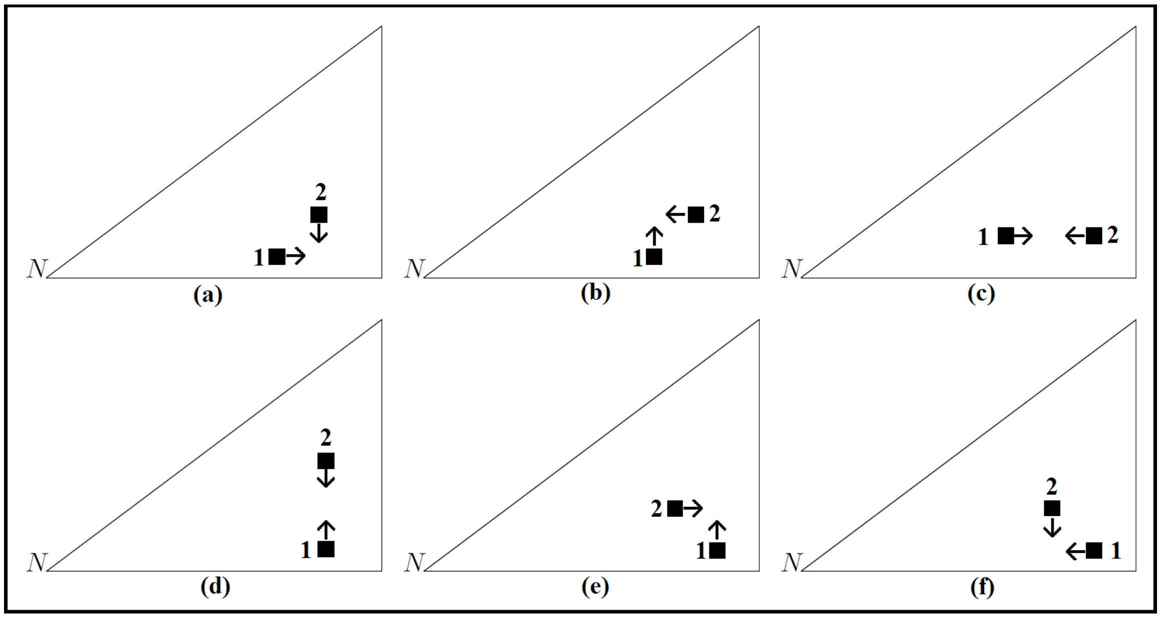


Figure 36: Possible collision scenarios: N is the nest

following proposition.

Proposition 1. *There are no collisions in the* Sectors *algorithm.*

Theorem 3.4.1. *The competitive ratio of* Sectors *is* $\Theta(g)$*.*

Proof. We observe that in the worst case, the food is placed at the location shown in Figure 37 which is $\frac{g}{2}$ steps away from the nest while the time taken by *Sectors* is approximately $\frac{g^2}{n}$. Therefore, the competitive ratio of *Sectors* is $\Theta(g)$ as claimed. $\square$

3.5 Spiral Search

Lopez-Ortiz et al [3] studied the problem of multiple agents with limited visibility searching for an object of interest on a lattice. These agents start searching from the same point initially i.e. the center of the grid. They gave this spiral search strategy for exploring the region with $n = 2^k$ agents. It involved symmetric movements of agents on the lattice in such a way that it gave an impression of agents spiraling

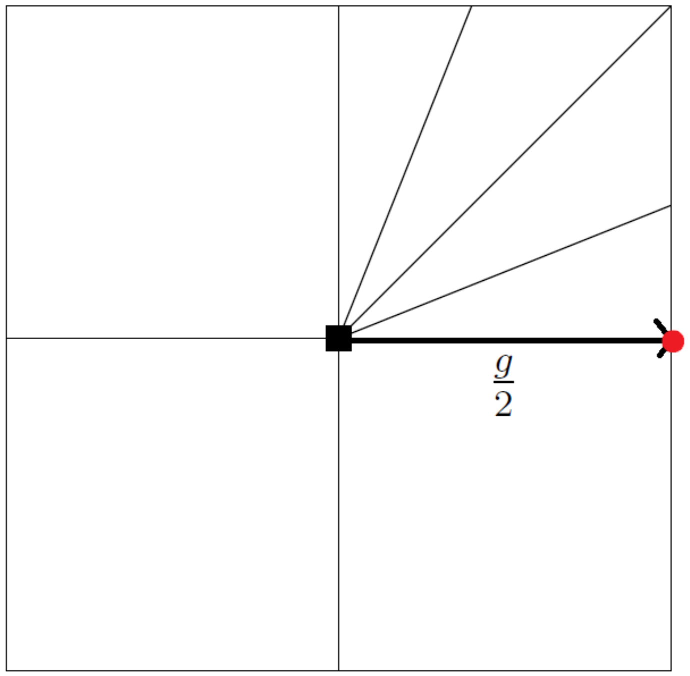


Figure 37: The cell with the highest competitive ratio in *Sectors*

around the origin.

Figures 39 and 40 show the pattern followed by *four* and *eight* agents respectively. Note that the original pattern introduced by Lopez-Ortiz involved searching on an infinite lattice but since we have a finite grid to search, the movement pattern needed to be slightly adjusted. Referring to Figure 39, whenever an agent touches the grid boundary, instead of going the full circle as it is supposed to go in the original pattern, it moves along the boundary to reach the point from where it can continue following the original path. If an agent is not aware of the food location at the end of the search operation, it will just halt at its last location and wait for some fellow agent to come to it with this information.

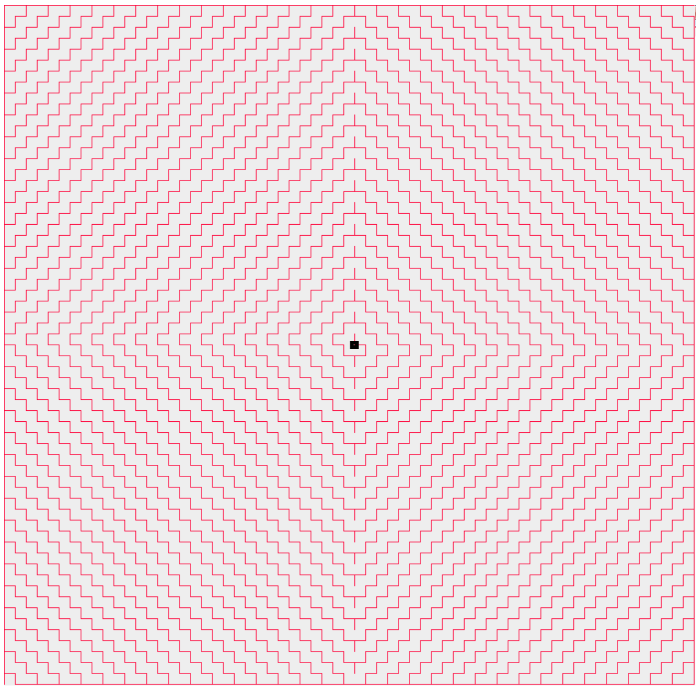


Figure 38: Spiral Search by a single agent on a finite grid

3.6 Modified Zigzag

As we have already established, the primary goal of the agents is to locate the food as quickly as possible. All the agents start the search together in the patterns we have seen so far, either from the very end of the grid or from the nest itself. As is expected,their performance depends on how far the food is placed from the nest. We wanted to design such an algorithm in which the area closer to the nest and the far out areas of the grid get explored for food simultaneously to accelerate the search process. We call this algorithm $Modified\ Zigzag$ because the pattern agents to follow to cover their share of the ground is the same as $Zigzag$ algorithm, the difference lies in the division of grid among agents.

This algorithm is only given for a swarm consisting of 8 agents, two in every quadrant. At time T_1, $four$ agents, one in every quadrant, move out from the nest towards the border, the rest of the agents come out in the next time step following

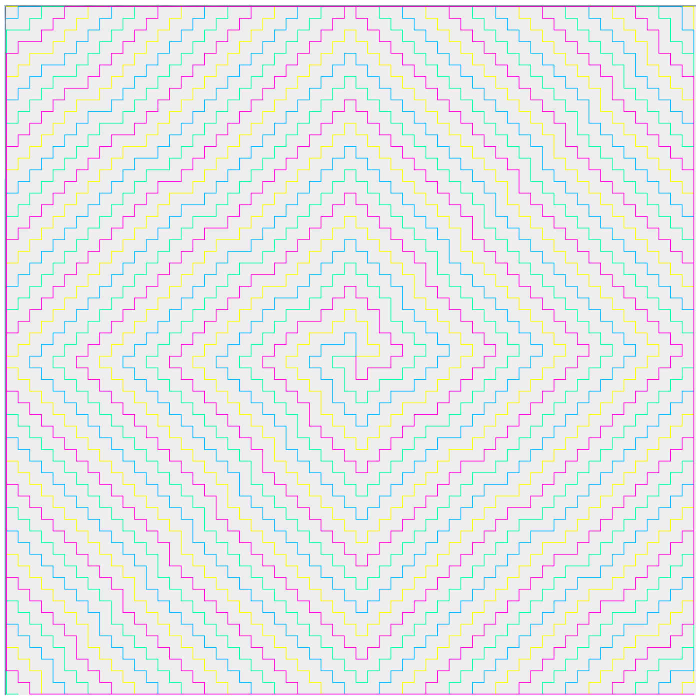


Figure 39: Spiral Search by *four* agents

the previous agents to the border. The first set to come out of the nest touches border first and establish the grid size, all four at the same time in their respective quadrants. These agents constitute the *Outer Agent Set*. At the very next time step, the remaining four agents, constituting the *Inner Agent Set* touch border, determine the grid size and move back towards the nest. When they reach the nest, they start covering the grid following the *Zigzag* strategy. *Communication Phase* is

Code	Direction(i)
1	$N_i - E_i - S - W_{i-1} - S_{i-1} - E$
2	$N - W_i - S_i - W - N_{i+1} - E_{i+1}$
3	$N_i - W_i - S - E_{i-1} - S_{i-1} - W$
4	$N - E_i - S_i - E - N_{i+1} - W_{i+1}$
5	$S_i - W_i - N - E_{i-1} - N_{i-1} - W$
6	$S - E_i - N_i - E - S_{i+1} - W_{i+1}$
7	$S_i - E_i - N - W_{i-1} - N_{i-1} - E$
8	$S - W_i - N_i - W - S_{i+1} - E_{i+1}$

Table 4: Code-Direction Table for Modified Zigzag

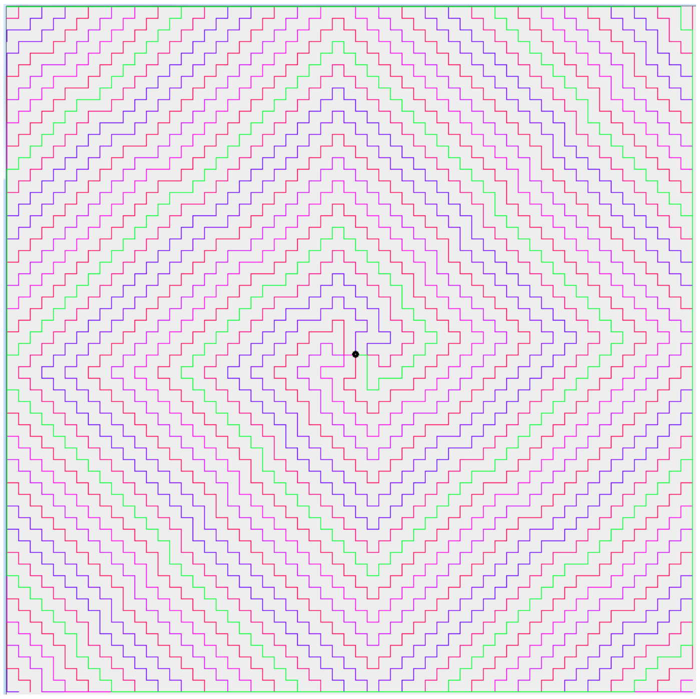


Figure 40: Spiral Search by *eight* agents

dependent on the fact that the agents know the path their neighbors are going to follow and they can get to them when they want to share the food location information. For that reason, each agent is required to know the grid size even though the inner agents will only cover the area nearby the nest and does not really need the grid size information for the exploration phase. If any one of the inner agents locates the food, it shares that information with its outer counterpart and it can only do that if it knows how to reach the agent.

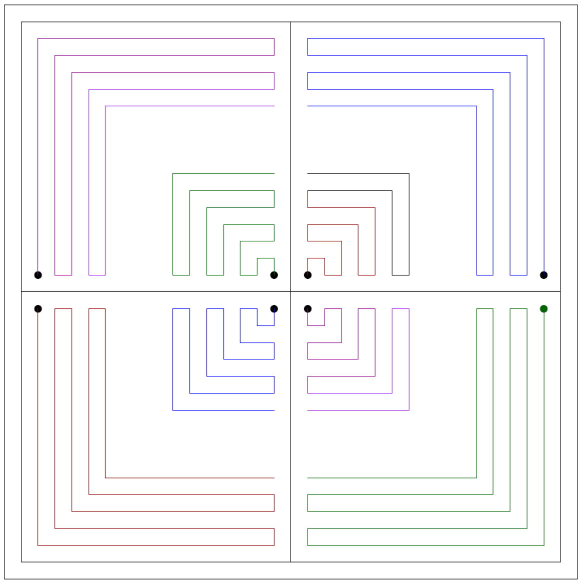


Figure 41: Modified Zigzag Pattern by 8 agents

Algorithm 13 *Modified Zigzag* ($Agent_A$)

```
1: Start moving away from the $Nest$ horizontally till the border is encountered.
2: $g \leftarrow 2 *$ Distance traveled from $Nest$ to border
3: if $outerCircleAgent$ then
4:     $i \leftarrow \frac{g}{2}$
5:     while $\neg haveFood \lor$ has not met inner counterpart do
6:         Follow Direction(i) for $Agent_A.code$ from Table 4.
7:         $i \leftarrow i - 2$;
8:     end while
9: else
10:     Go back to $Nest$.
11:     $i \leftarrow 1$
12:     while $\neg haveFood \lor$ has not met outer counterpart do
13:         Follow Direction(i) for $Agent_A.code$ from Table 4.
14:         $i \leftarrow i + 2$;
15:     end while
16: end if
```

Chapter 4

Communication Phase

In this chapter, we describe our approach to the communication phase. The goal of this phase, which commences as soon as one of the robots finds the food, is to enable all agents to discover the location of the food via inter-agent communication. Recall that in our model, two agents communicate only when they are in adjacent cells. The communication phase ends when all agents know the location of the food. The algorithm for the communication phase for an agent A that has come to know the location of the food must specify:

- To which agents should A communicate the location of the food?
- In what order should it try to meet these agents, and where should it meet them?

Observe that for n agents foraging in a $g \times g$ grid, the exploration phase takes $\Omega(\frac{g^2}{n})$ time in the worst case, but the communication phase for any agent takes $\mathcal{O}(g)$ time. Therefore, the total time for the communication phase is $\mathcal{O}(gn)$. Note that this is a very loose upper bound and the actual time taken for the communication phase is much lower than this.

For all exploration strategies, we follow the same strategy for communication.

Every agent is responsible to communicate the location of the food to its neighbors, that is, the agents who are exploring an adjacent area of the grid. We assume that an agent can determine the location of any agent on the grid and hence, can figure out where it can meet the other agent. Then, it takes the shortest possible path to reach its neighbor. However, the communication phase for our four different exploration strategies differ in the sense that they have a different set of neighbors to share the information with depending upon the strategy they used to divide up the search grid.

4.1 Strips Communication

Strips is specifically designed to accelerate the propagation of information across the swarm. The agents start exploring their respective regions from some designated spots all at the same time so that they can meet at the border region after regular intervals. When an agent locates the food, it is responsible for taking that information to two of its fellow agents if it is in the middle of the grid, and only one agent when it is either covering the first or the last strip. It does so simply by following its regular path because it is anyway taking the shortest path to get to its strip's border where its neighboring agent will also be present in the adjacent location. An agent who is not a locater but received the location from a neighbor will be responsible for communicating it to its other neighboring agent and it will achieve that simply by following its regular path. If an agent does not find the food in its entire search area and does not get any information from its neighbors, it will just remain stationery at the last location of its search region and wait for its neighbors to bring the information to it.

Suppose the food was located by the agent assigned to the j^{th} strip in the i^{th} row of the grid. If the agent was moving in the $North$ direction when it discovered food as shown in Figure 42, total time taken to propagate the food location information

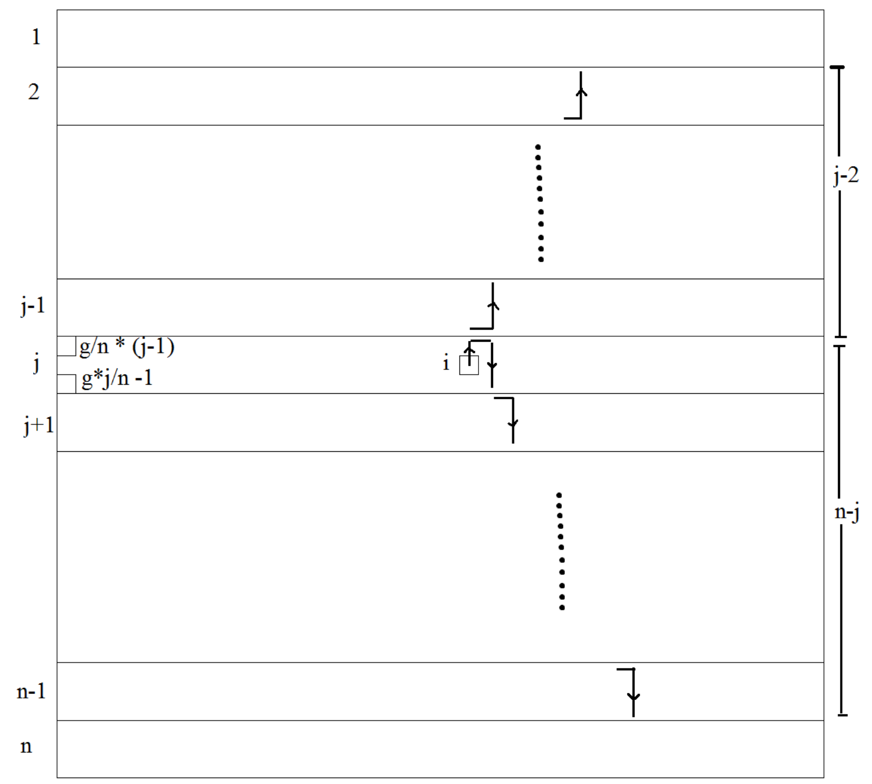


Figure 42: Agent in j^{th} strip locates food in the i^{th} row while going up

throughout the swarm would include the time taken to reach the border shared with $j-1^{st}$ strip plus the time taken by the information to travel through either $n-j$ or $j-2$ strips, whichever is greater because the information flow on both the sides is happening simultaneously and the greater number of strips on one side would be the maximum time taken. Time steps to cover one strip is $\frac{g}{n}$. Total time taken can be given by $i-[\frac{g}{n}\times(j-1)]+\frac{g}{n}\times\max(n-j,j-2)$.

If the agent was moving in the *South* direction at the time of discovery as shown in Figure 43, total time for the information propagation includes time taken to reach to the border shared with $j+1^{st}$ strip plus the time taken by the information to travel through either $j-1$ or $n-j-1$ strips, whichever is greater. This time is given by $i-[\frac{g}{n}\times(j-i-1)]+\frac{g}{n}\times\max(n-j-1,j-1)$.

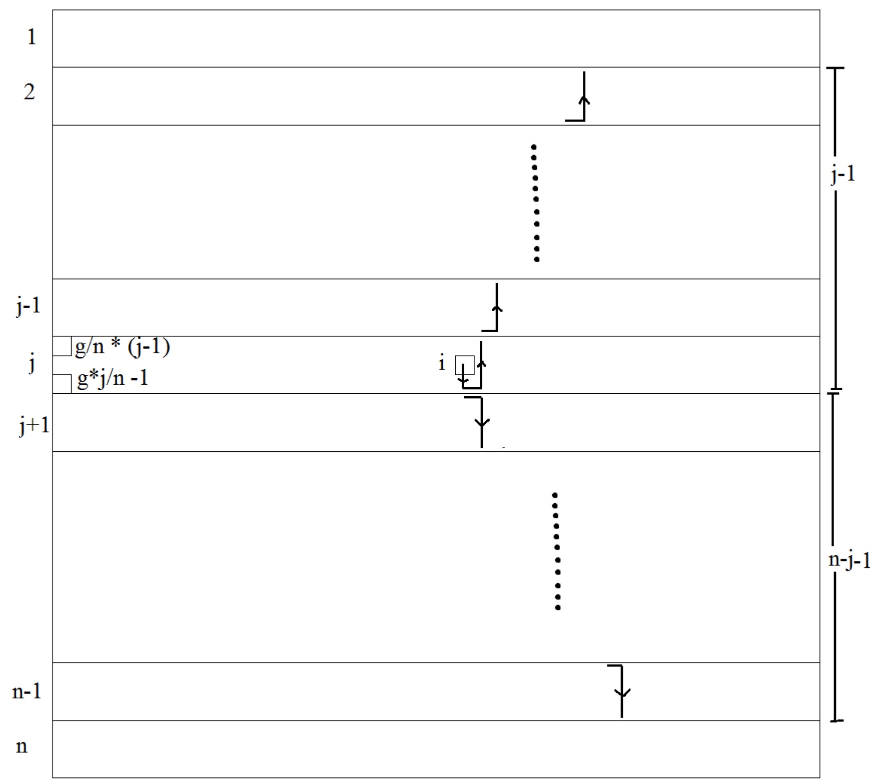


Figure 43: Agent in j^{th} strip locates food in the i^{th} row while going down

4.2 Communication strategy for ZigZag & Hilbert

ZigZag and *Hilbert* patterns use the same strategy to divide the grid into squares repeatedly as shown in Figure 44. Hence, they have the same procedure for deciding their neighboring agents. Agents can be present in one of the three kinds of regions-corner(A), border(C) or central(B) as shown in Figure 45. Whenever an agent has been assigned a region, it has a set of neighbors which consists of the agents sharing the region boundaries with it. Referring to Figure 45, A has 2 neighbors, B has 4 and C has 3. If the said agent is the food locater, it is required to get that information to all its neighbors. Otherwise, an agent not being the locater implies that one of the neighboring agents has communicated the food location information to it. In that case, it is required to meet all its other neighbors except that particular agent.

Once an agent has figured out its neighbors, the next step is to determine the

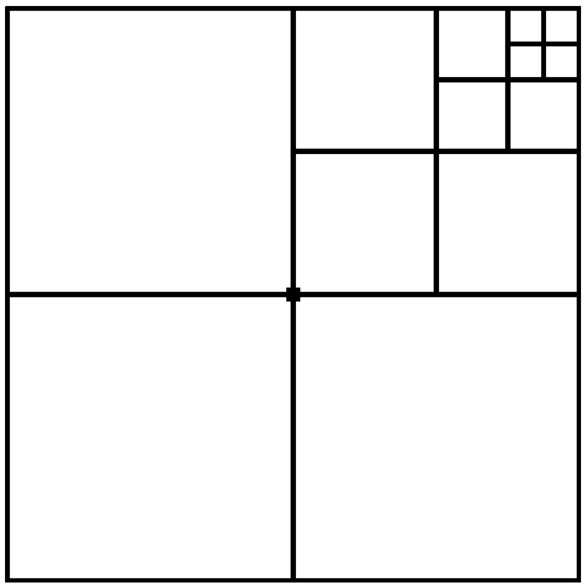


Figure 44: Division of the square grid into further squares

order in which it meets them. It chooses one particular order on the basis of its completion time step. For example, $Agent_A$ has three neighbors,A_1, A_2, A_3 and it is the food locater. In this case, it has to communicate to all three of them. First, it determines the time it will take to meet all of them in all possible orders of meetings i.e. (A_1, A_2, A_3); (A_1, A_3, A_2); (A_2, A_1, A_3); (A_2, A_3, A_1); (A_3, A_1, A_2); (A_3, A_2, A_1). $Agent_A$ then selects the order which gives the least completion time step. Once it has communicated to all its neighbors, it enters into the next phase i.e. *Transportation-Phase*. When the grid is covered by only $four$ agents following *ZigZag* algorithm, the movement of all the agents is symmetrical in their respective quadrants as is shown in figure 46. Every agent meets two of the other agents on a regular basis. Suppose $Agent_A$ locates the food at the highlighted location in Figure 46, it just keeps following its exploration path and after some time, it meets $Agent_B$ and shares the food location information with it. At this time, only $Agent_A$ and $Agent_B$ are aware of the food location. Both of them again continue on their exploration paths. After a few more time steps, $Agent_B$ meets $Agent_C$ and $Agent_A$ meets $Agent_D$ at the

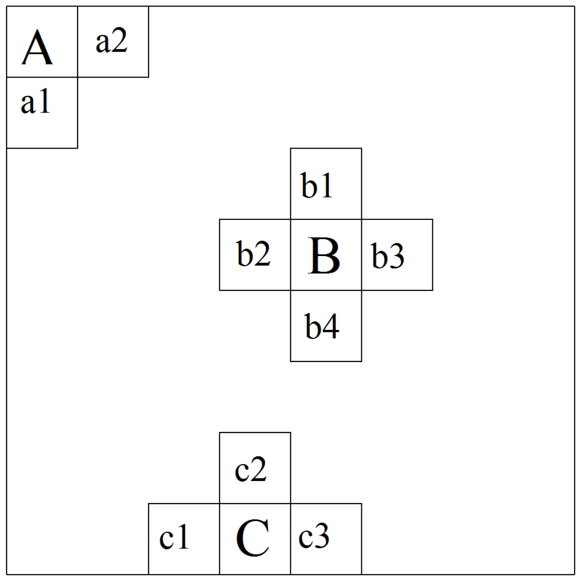


Figure 45: Neighbors according to the positioning of agents

same time step since their movements are identical and symmetrical. At this time, *Communication Phase* is over i.e. all the agents are aware of the food location and they all enter into the *Transportation Phase* together.

4.3 Modified Zigzag Communication

Modified Zigzag is only designed to work for *eight* agents. As already discussed, the swarm is divided into two sets, *Inner Agent Set* and *Outer Agent Set* consisting of *four* members each in this strategy. Referring to Figure 47, suppose $Agent_2$ from the *Inner Agent Set* locates the food at the highlighted position while it is moving towards $Agent_4$, it shares this information with $Agent_4$ when they meet at the border, say at T_k. Now, both of them are aware of the food location and they take this information to the rest of the members of the *Inner Agent Set*. They reach their neighbors at the same time, $Agent_2$ shares the information with $Agent_8$ and $Agent_4$ with $Agent_6$ at the same time step, say $T_{k+\delta}$. At this point, all the members of the *Inner Agent Set* are aware of the food location.

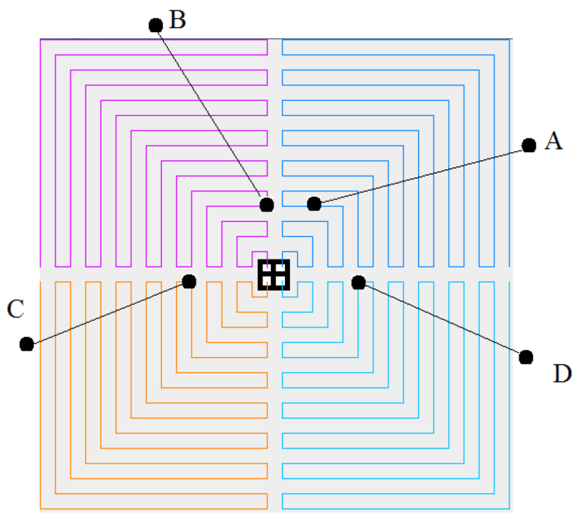


Figure 46: Grid covered by 4 agents using *ZigZagPattern*

Since they also determined the size of the grid in the *Initial Deployment Phase*, every member of the *Inner Agent Set* can reach its counterpart from the *Outer Agent Set*. Note that the outer agents initiated their search operation together at the same time and they follow the same pattern in their respective quadrants. When the inner agents at the axis at $T_{k+\delta}$ determine where to reach their counterparts, every agent gets the same location to reach them and they take the shortest path to reach them. After some time, $Agent_2$ communicates to $Agent_1$, $Agent_4$ to $Agent_3$, $Agent_6$ to $Agent_5$, $Agent_8$ to $Agent_7$ at the same time.

If the food location discoverer belongs to the *Outer Agent Set*, it can communicate the information to the fellow members from its set first and then every outer agent takes the information to its inner counterpart in the same way as they did in the above example. Note that every agent in the swarm is entering into the next phase at the same time. It is not the case with the other exploration strategies in which there is a period when some agents are still in *Communication Phase* when some have already entered *Transportation Phase*

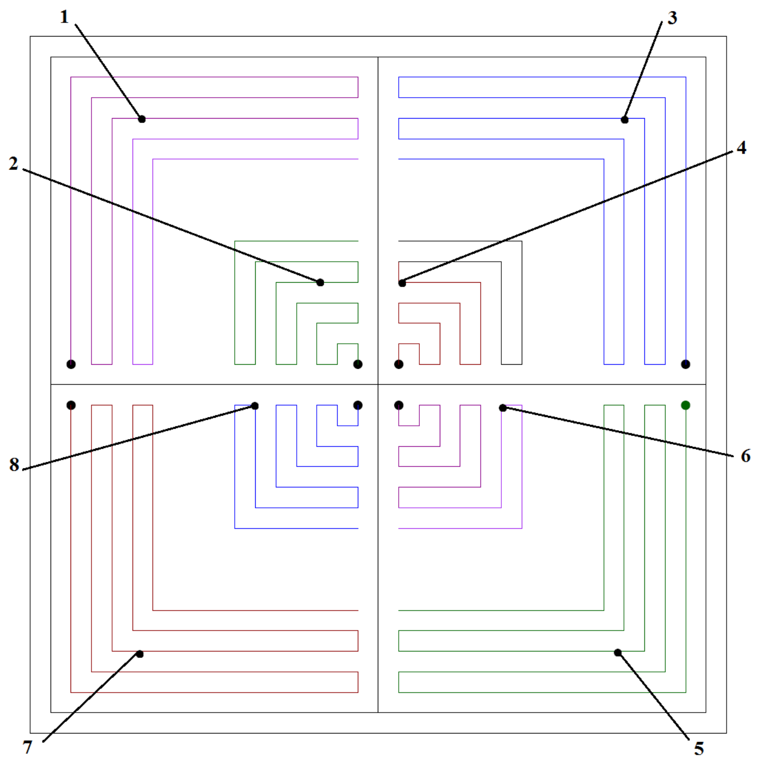


Figure 47: Communication Phase in *Modified Zigzag* algorithm

4.4 Communication for Spiral Search and Sectors

For communication in Spiral Search and Sectors, we again exploit the fact that the agents can determine the paths followed by their neighbors and hence, can figure out where to reach them to share the location of the food. Referring to Figure 48, every agent's path in spiral search is enclosed by two fellow agents' paths and these are the ones which constitutes the neighbors of that agent. The food locater will take the information to both the agents and the rest will take it to only one neighbor. Similarly, in Sectors, every agent has two neighbors which are exploring the neighboring sectors.

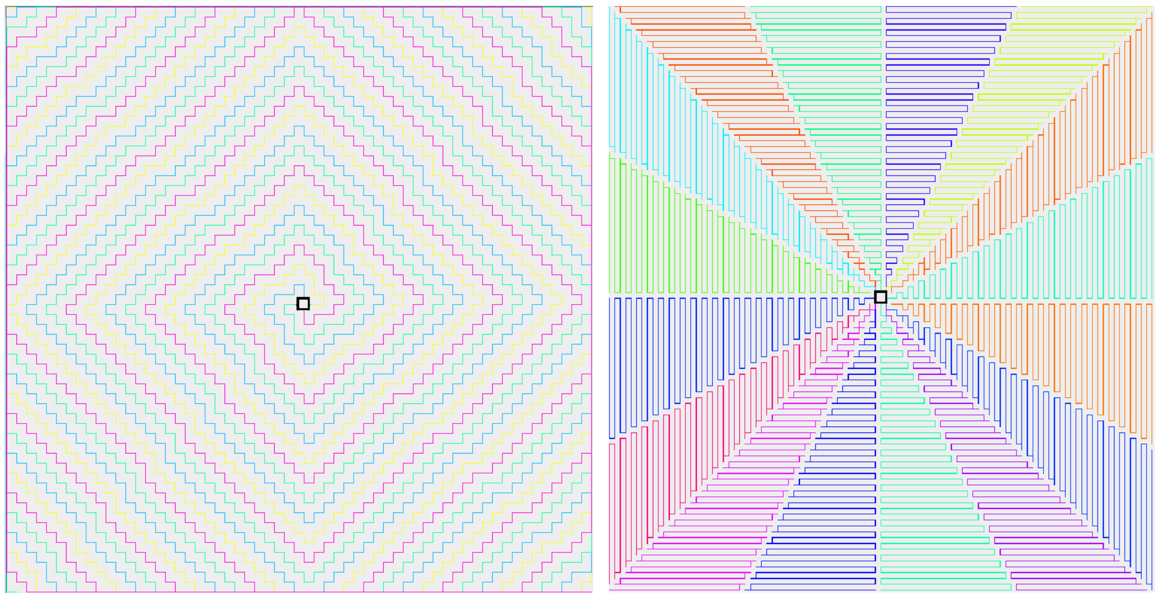


Figure 48: Agents following *Lopez-Ortiz* and *Sectors* patterns

Chapter 5

Transportation Phase

When an agent has completed the *Exploration* and *Communication* phases, it enters into *Transportation-Phase.* Making multiple round trips from *Food* to *Nest* might be required of an agent depending upon the amount of food present at the source since it can only hold one food unit at a time. We have designed the *Exploration-Phase* in such a way that every agent follows a well-defined path without any possibility of collisions. However, in the *Communication-Phase*, we already saw that we opt to have every agent follow the shortest path to the agents with which it needs to communicate, which necessitates the use of collision avoidance mechanism since such situations can be arisen in which multiple agents are contending for the same location on the way. Similarly, agents in the *Transportation-Phase* determine their movements on the fly which can result in collisions. Hence, this phase also calls for following Algorithm 1 for collision avoidance. We investigate two approaches for the execution of this phase:

- *End*-to-*End Transportation*: As the name suggests, every agent picks up one unit of food, takes it to the *Nest*, drops it off, goes back to the *Food* location and repeat the process.

- *Relay Transportation*: In this technique, there will be a single direct path from $Food$ to $Nest$ on which the agents will form a relay network and the food units will be handed over from one agent to the next when they are in adjacent locations.

5.1 End-to-End Transportation

When an agent enters *Transportation-Phase*, it is already aware of the location of the food. It will try to reach there taking the shortest path possible. When it reaches at the food location, it will pick up one unit of food and head straight for the $Nest$. Upon reaching there, it will drop the food unit it was carrying and repeat the whole process again till there is no food left at the source location. In this technique, every agent is working on its own making the rounds from $Food$ to $Nest$ and vice versa.

Algorithm 14 *End*-to-*End Transportation* $(Agent_A)$

```
1: $target \leftarrow Food$
2: while $haveFood \lor foodUnits > 0$ do
3:     if $currentLocation = Food$ then
4:         $foodUnits$ - -;
5:         $haveFood \leftarrow true$
6:         $target \leftarrow Nest$
7:     end if
8:     if $haveFood \land currentLocation = Nest$ then
9:         $haveFood \leftarrow false$
10:        $target \leftarrow Food$
11:    end if
12:    GetNextMove-CA($Agent_A, target, contenders$)
13: end while
```

5.2 Relay Transportation

As its name would suggest, *Relay Transportation* involves forming a relay network of agents on a single path between $Nest$ and $food$ as shown in figure 49. Every agent uses the same algorithm to determine the path once it knows the food location and joins the relay network upon finishing its *Communication-Phase*. They can do so by either moving to the closest location on the path or by entering the path from the $Food$ end. Once on the path, every agent will follow algorithm 15 till the food has been transported back to the $Nest$ completely. Agents are, basically, oscillating to-and-fro on the path and the food units are being picked up by agent nearest to the $Food$ location, handed over from agent-to-agent in the middle, and dropped off at the $Nest$ by the agent closest to the $Nest$.

We have four possibilities when $Agent_A$ and $Agent_B$ find themselves in adjacent

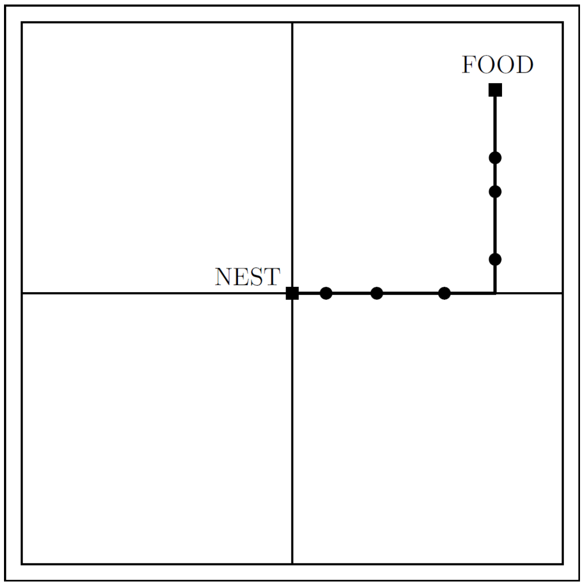


Figure 49: Agents performing *Relay Transportation* on Nest-Food path

locations as shown in figure 50. They all will be handled differently as follows:

- $A.haveFood \land \neg B.haveFood$

  Since $Agent_A$ is closer to $Nest$ than $Agent_B$ in this case, food will not be

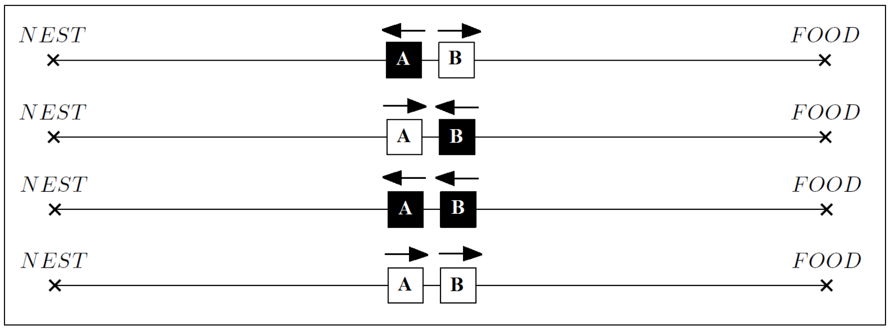


Figure 50: *Two* agents in adjacent locations on *Nest-Food* path:Dark agents carry *food*

exchanged. Rather, $Agent_A$ will continue moving towards $Nest$ and $Agent_B$ towards $Food$.

- $\neg A.haveFood \wedge B.haveFood$

  Food will get exchanged in this case since $Agent_A$ is closer to $Nest$ than $Agent_B$. After the exchange, they will both turn in opposite directions.

- $A.haveFood \wedge B.haveFood$

  Since both the agents are carrying food, no exchange can take place. They will keep moving back-to-back towards $Nest$.

- $\neg A.haveFood \wedge \neg B.haveFood$

  When none of the agents is carrying food, they will keep moving towards $Food$ back-to-back.

5.3 Equidistant agents on a Nest-Food Path

In this section, we analyze the behavior of the two transportation algorithms we presented in the previous section. For comparison purposes, we assume that all n agents have entered $Transportation$ phase at the same time and they are all placed

Algorithm 15 *Relay-Path-Transportation* ($Agent_A$)

```
while $haveFood \vee foodUnits > 0$ do
    if $haveFood$ then
        $nextStep \leftarrow$ adjacent location on the path closer to $Nest$
        if $currentLocation = Nest$ then
            $haveFood \leftarrow false$
            RELAY-PATH-TRANSPORTATION($Agent_A$)
        else $Agent_B$ at $nextStep$ & $\neg Agent_B.haveFood$
            $haveFood \leftarrow false$
            RELAY-PATH-TRANSPORTATION($Agent_A$)
        end if
    end if
    if $\neg haveFood$ then
        $nextStep \leftarrow$ adjacent location on the path closer to $Food$
        if $currentLocation = Food$ then
            $foodUnits$ - -;
            $haveFood \leftarrow true$
            RELAY-PATH-TRANSPORTATION($Agent_A$)
        else $Agent_B$ at $nextStep$ & $Agent_B.haveFood$
            $haveFood \leftarrow true$
            RELAY-PATH-TRANSPORTATION($Agent_A$)
        end if
    end if
    GETNEXTMOVE-CA($Agent_A, nextStep, contenders$)
end while
```

equidistant on the *Nest-Food* path initially. [1] Further, we assume that n units of food are placed at the source location and the agents have to transport them back to the *Nest*. We compare the total time taken by the agents to complete this task using both the algorithms.

Let L_k denote the k^{th} location on the path and T_j be the j^{th} timestep in the simulation. We assume that they are all moving towards the *Food* initially. We consider the cases when the separation between adjacent agents is odd and even separately.

Assume there are n agents placed equidistant on the *Nest-Food* path and the distance between every pair of adjacent agents is $2i+1$ units. Then, the total length of the path is $(n+1)(2i+1)$. When *Nest* and *Food* are placed in the same row/column, *End*-to-*End Transportation* will not allow the agents to take the shortest path between them. Since multiple agents will be making these trips from *Nest* to *Food* and back, they will have to move out of the way to avoid colliding with fellow agents and hence, not taking the shortest path. So, for simplicity, we assume that *Nest* and *Food* are not in the same row/column when determining the running time of *End-to-End Transportation* algorithm.

Theorem 5.3.1. *Given n agents placed x units of distance apart on a path from Nest to Food, transportation of n units of food takes $x(2n+1)$ time steps using End-to-End Transportation.*

Proof. Suppose the agents are located equidistant on the *Nest-Food* path at T_0 as shown in Figure 51. They start moving towards *Food* in the next time step. Every agent is responsible for picking up one unit food from the *Food* location and dropping it off at the *Nest*.

[1] It is not necessary that all the agents enter this phase at the same time. Some of them might still be communicating the food location information to their neighbors while others have already started transporting the food.

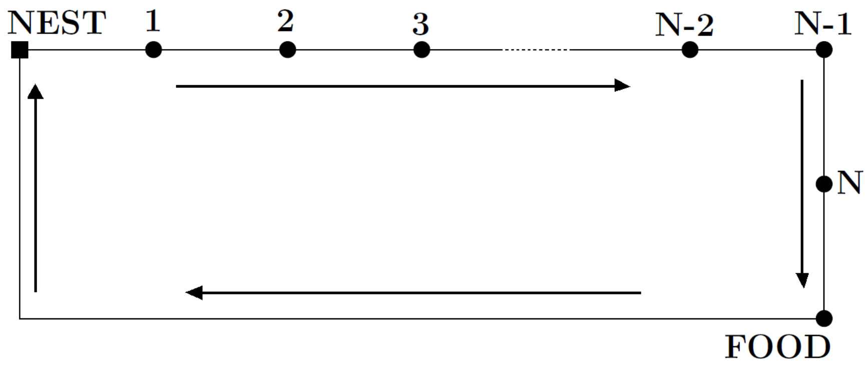


Figure 51: n agents placed equidistant on a bent *Nest-Food* path at T_0

$Agent_N$ will be the first one to reach $Food$ at T_x and it will pick up one food unit and move towards the $Nest$ using the alternate path. After every x time steps, the subsequent agents in the line will repeat the process. $Agent_1$ will reach $Food$ at T_{nx} and since this is the last agent in the line, the time step at which this agent reaches $Nest$ will be the time step at which the transportation is concluded. $Agent_1$ will take another $x(n+1)$ time-units after reaching $Food$ to reach $Nest$ which gives us $T_{x(2n+1)}$ as the completion time step. This proves that theorem 5.3.1 holds true. □

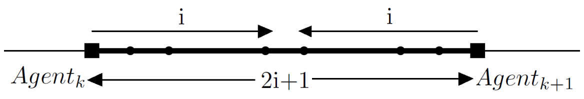


Figure 52: When two agents located $2i+1$ units of distance apart want to meet, they will be in adjacent cells after moving i steps towards each other

By substituting $x = 2i + 1$ in Theorem 5.3.1, we get the following corollary.

Corollary 5.3.1.1. *Given n agents placed $(2i + 1)$ units of distance apart on a path from $Nest$ to $Food$, transportation of n units of food takes $(2i+1)(2n+1)$ time steps using End-to-End $Transportation$.*

Next we analyze the behavior of *Relay-Transportation* algorithm. It is clear that in this algorithm, all the agents can stay on the *Nest-Food* path at all times i.e. it is irrelevant whether the $Food$ and $Nest$ are in the same row/column. For simplicity, we assume that $Nest$ and $Food$ are situated in the same row/column as shown in figure 54 in which case there is only a single shortest path between them. $Nest$ is located at L_0 and the food is placed at $L_{(n+1)(2i+1)}$. Every $Agent_k$ for $k \in [1, n]$ is placed at location $L_{k(2i+1)}$ at time T_0.

Figure 53 shows $four$ agents placed $2i + 1$ units of distance apart using *Relay-*

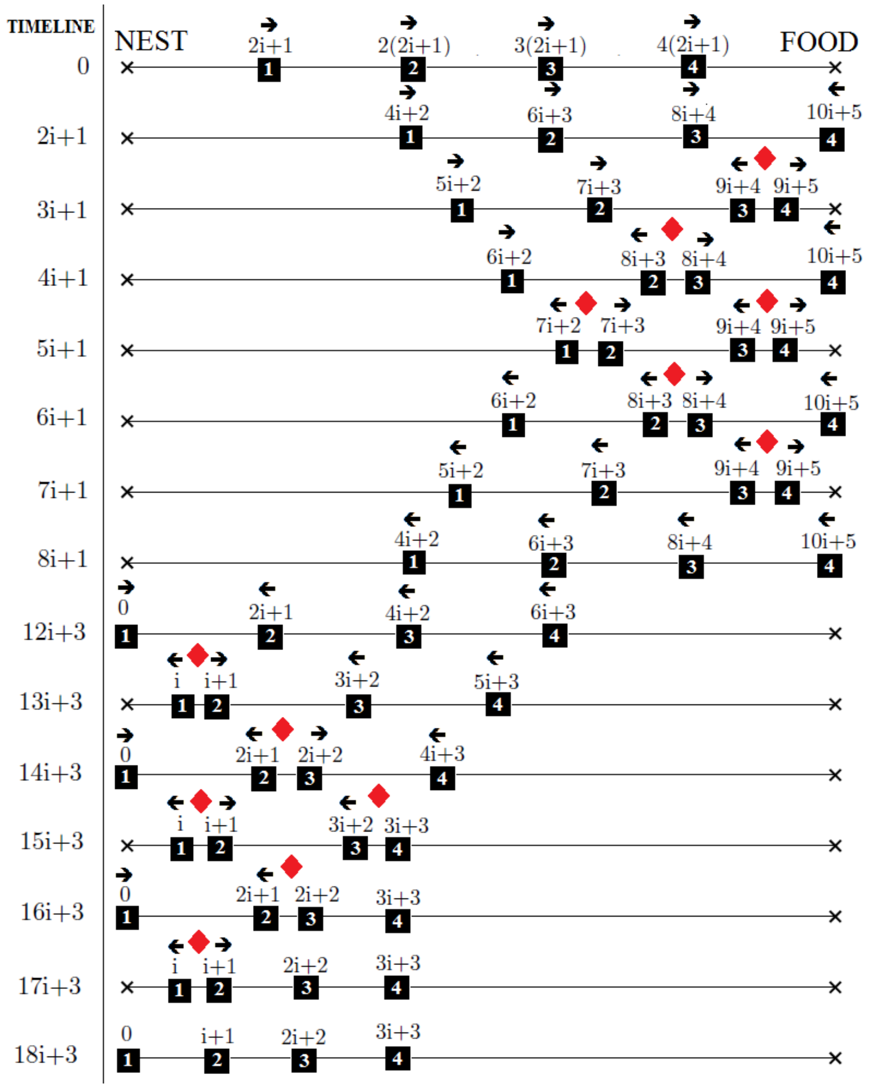


Figure 53: Relay transportation of 4 food units on a path of length $10i + 5$ with 4 agents placed $2i + 1$ units of distance apart initially

$Transportation$ algorithm to transport n units of food in $18i + 3$ time steps. We observe that since the food units are being transferred from agent-to-agent on the

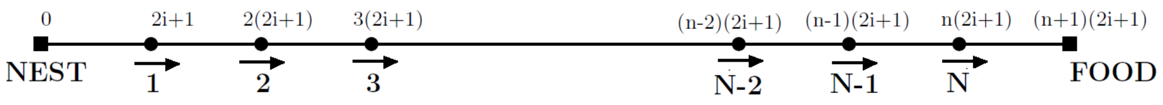


Figure 54: Agents placed equidistant on a straight *Nest-Food* path at T_0: Every $Agent_k$ is at location $L_{k(2i+1)}$ for $\forall k \in [1, N]$

path, it can be said that the meetings between the agents play a crucial role in the execution. If we want to determine the running time of this algorithm for n agents, we need to analyze where and when the meetings are happening. For the remainder of this section, when we say $Agent_a$ meets $Agent_b$ at location L_k, it means that $Agent_a$ is located at L_k and $Agent_b$ at L_{k-1} if $b < a$ and at L_{k+1} otherwise. Now, we analyze the meetings between the adjacent agents on the path.

Lemma 5.3.2. *$Agent_k$ meets $Agent_{k+1}$ for the first time to receive the first food unit at location $L_{i(n+k+2)+k+1}$ and time $T_{i(n-k+2)+1}$ for $\forall k \in (1, N-1)$.*

Proof. We give a proof by induction on k. For the base case with $k = N-1$, we claim that $Agent_{N-1}$ has its first meeting with $Agent_N$ to receive the first food unit at location $L_{i(2n+1)+n}$ and time T_{3i+1}. As we already know that $Agent_N$ is located at $L_{n(2i+1)}$ at T_0, and is separated by $2i+1$ units of distance from the *Food* location, it will reach there at T_{2i+1}. It will pick up the first food unit and turn back on the path in the next time step. $Agent_{N-1}$ is located at $L_{n(2i+1)}$ at this time, which is $2i+1$ steps away from $Agent_N$. After moving i steps towards each other, $Agent_{N-1}$ will meet $Agent_N$ at location $L_{i(2n+1)+n}$ and time T_{3i+1} to receive the first food unit that $Agent_N$ was carrying which proves that the basis holds true.

Assume that Lemma 5.3.2 holds true for $k = m$, i.e. $Agent_m$ has its first meeting with $Agent_{m+1}$ to receive the first food unit at location $L_{i(n+m+2)+m+1}$ and time $T_{i(n-m+2)+1}$ after which both of them will turn in opposite directions. At this time, $Agent_{m-1}$ is located $2i+1$ units of distance left to the location of $Agent_m$ i.e. at $L_{i(n+m)+m}$. It has not yet met any of its fellow agents. After moving i steps towards each other, $Agent_{m-1}$ will meet $Agent_m$ for the first time at location $L_{i(n+m+1)+m}$ and time $T_{i(n-m+3)+1}$. Since $Agent_m$ was carrying the first food unit, $Agent_{m-1}$ will receive the first food unit at this time. This proves that the statement also holds for $k = m - 1$ which completes our proof by induction. □

We now use the above lemma to determine the time steps and the locations of the subsequent meetings between agents in the following lemma.

Lemma 5.3.3. *$Agent_k$ has its j^{th} meeting with $Agent_{k+1}$ to receive j^{th} food unit at location $L_{i(n+k+2)+k+1}$ and time $T_{i(n-k+2j)+1}$ for $1 \leq j \leq k$ and $\forall k \in (1, N-1)$.*

Proof. We prove this by induction on j. For the base case with $j = 1$, the lemma states that $Agent_k$ has its first meeting with $Agent_{k+1}$ to receive the first food unit at location $L_{i(n+k+2)+k+1}$ and time $T_{i(n-k+2)+1}$. We know this to be true from Lemma 5.3.2.

For the induction step, we assume that Lemma 5.3.3 holds true for $j = m$ i.e. $Agent_k$ has its m^{th} meeting with $Agent_{k+1}$ to receive m^{th} food unit at location $L_{i(n+k+2)+k+1}$ and time $T_{i(n-k+2m)+1}$. We claim that $Agent_k$ has its $(m+1)^{st}$ meeting with $Agent_{k+1}$ to receive $(m+1)^{st}$ food unit at the same location at time $T_{i(n-k+2m+2)+1}$.

When $Agent_k$ meets $Agent_{k+1}$ for the m^{th} time and receives the m^{th} food unit, both of them will move in opposite directions in the next time step. After i time steps at $T_{i(n-k+2m+1)+1}$, $Agent_{k-1}$ meets $Agent_k$ at $L_{i(n+k+1)+k}$ to receive the m^{th} food unit and $Agent_{k+1}$ meets $Agent_{k+2}$ at $L_{i(n+k+3)+k+2}$ to receive the $(m+1)^{st}$ food unit. At this point, $Agent_k$ carrying no food and $Agent_{k+1}$ with the $(m+1)^{st}$ food unit, separated by $2i + 1$ units of distance, start moving towards each other again. Evidently, $Agent_k$ will meet $Agent_{k+1}$ for the $(m+1)^{st}$ time to receive the $(m+1)^{st}$ food unit after moving i steps at $L_{i(n+k+2)+k+1}$ and $T_{i(n-k+2m+2)+1}$ which shows that the lemma also holds for $j = m + 1$. This completes our proof by induction. □

As is clear from the lemma, subsequent meetings between two given agents happens after every $2i$ time steps at the same location for k times. Now, we analyze what happens after they have met k times.

Lemma 5.3.4. *After meeting $Agent_{k+1}$ for the k^{th} time to receive the k^{th} food unit at time $T_{(n+k)+1}$, $Agent_k$ continues to move left without meeting $Agent_{k-1}$ for the next $i(n+3)+2$ time steps for $\forall k \in [1, N]$.*

Proof. We prove this by induction on k. For the base case, we claim that the lemma holds true for $k = 1$. We know from Lemma that $Agent_1$ receives the first unit at location $L_{i(n+3)+2}$ and time $T_{i(n+1)+1}$. At this time, when $Agent_1$ turns back on the path, there are no agents to its left. Hence, it will keep moving unhindered towards the $Nest$ for another $i(n+3)+2$ time steps and will reach there at $T_{i(2n+4)+3}$ and drop the food unit it was carrying[2].This proves our base case to be true.

Now we assume that the lemma holds true for $k = m$ and claim that it is also true for $k = m+1$. We know from Lemma 5.3.3 that $Agent_m$ receives the m^{th} food unit at location $L_{i(n+m+2)+m+1}$ and time $T_{i(n+m)+1}$. We also know from the same lemma that $Agent_{m+1}$ receives the $m+1^{st}$ food unit at location $L_{i(n+m+3)+m+2}$ and time $T_{i(n+m+1)+1}$ after which it turns towards the $Nest$. At this time, $Agent_m$ is located $2i+1$ units of distance to its left but is moving towards the $Nest$. Hence, they will not meet at the regular location and time. As per the inductive assumption, $Agent_m$ continues to move left for another $i(n+2)+2$ time steps and meets $Agent_{m-1}$ at location $L_{i(m-1)+m-1}$ and time $T_{i(2n+m+3)+3}$ and then turns towards the $Food$. $Agent_{m+1}$ will be located at $L_{i(m+1)+m}$ at this time. Intuitively, after another i steps, $Agent_{m+1}$ will meet $Agent_m$ at location L_{im+m} and time $T_{i(2n+m+4)+3}$ which is exactly $i(n+3)+2$ time steps after it received the $(m+1)^{st}$ food unit. The sequence of these

[2] $Nest$ can be seen as an agent too and $Agent_1$ reaching the $Nest$ can be considered as a meeting with the agent to its left.

events is shown in Figure 55. This proves that the lemma also holds for $k = m + 1$ which completes the inductive proof. $\square$

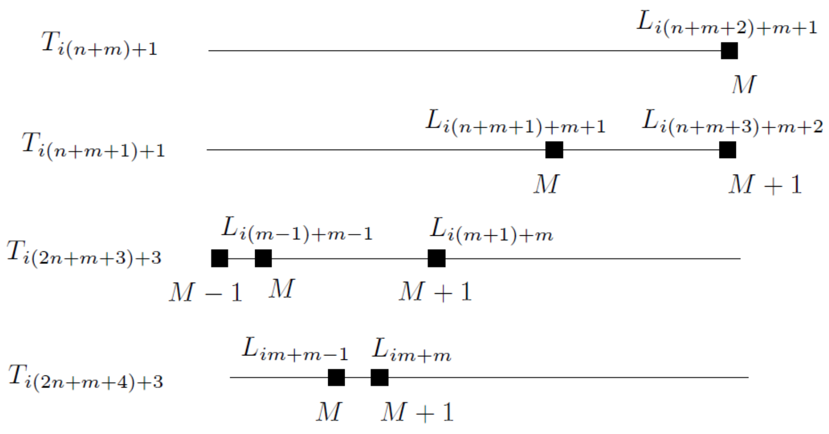


Figure 55: Meetings between $Agent_{m-1}$, $Agent_m$ & $Agent_{m+1}$

Now that we know the time and location of the $(k+1)^{st}$ meeting of $Agent_k$ and $Agent_{k+1}$, let us proceed to find out where and when will the subsequent meetings happen in the following lemma.

Lemma 5.3.5. *$Agent_k$ has its j^{th} meeting with $Agent_{k+1}$ to receive j^{th} food unit at location L_{ki+k-1} and time $T_{i(2n-k+2j+2)+3}$ for $k+1 \leq j \leq n$ and $\forall k \in (1, N-1)$.*

Proof. We give a proof for this lemma by induction on j. For the base case with $j = k + 1$, it states that $Agent_k$ has its $(k+1)^{st}$ meeting with $Agent_{k+1}$ to receive $(k+1)^{st}$ food unit at location L_{ki+k-1} and time $T_{i(2n+k+4)+3}$. We know from Lemma 5.3.4 that after meeting $Agent_{k+2}$ for the $(k+1)^{st}$ time to receive the $(k+1)^{st}$ food unit at time $T_{(n+k+1)+1}$, $Agent_{k+1}$ continues to move left for the next $i(n+3)+2$ time steps and meets $Agent_k$ at $T_{i(2n+k+4)+3}$ which proves that our basis holds true.

For the induction step, we assume that the lemma holds for $j = m$ and claim that it also holds for $j = m + 1$ where $k \leq m \leq n - 1$. From our assumption, we know that $Agent_k$ has its m^{th} meeting with $Agent_{k+1}$ to receive m^{th} food unit at location

L_{ki+k-1} and time $T_{i(2n-k+2m+2)+3}$. After this meeting happens, both of them will move in opposite directions in the next time step. After i time steps at $T_{i(2n-k+2m+3)+3}$, $Agent_k$ meets $Agent_{k-1}$ at $L_{(k-1)i+k-1}$, hands over the m^{th} food unit and $Agent_{k+1}$ meets $Agent_{k+2}$ at $L_{(k+1)i+k}$, receives the $(m+1)^{st}$ food unit.

At this point, $Agent_k$ carrying no food at $L_{(k-1)i+k-1}$ and $Agent_{k+1}$ with the $(m+1)^{st}$ food unit at $L_{(k+1)i+k}$, separated by $2i+1$ units of distance, start moving towards each other again. Evidently, $Agent_k$ will meet $Agent_{k+1}$ for the $(m+1)^{st}$ time to receive the $(m+1)^{st}$ food unit after moving i steps at L_{ki+k-1} and $T_{i(2n-k+2(m+1)+2)+3}$ which shows that the lemma also holds for $j = m+1$. This completes our proof by induction. □

We can now determine the time at which $Agent_1$ receives the j^{th} food unit using the above lemmas. Let us compute the time step at which it is dropped at the $Nest$.

Lemma 5.3.6. *$Agent_1$ drops the j^{th} food unit at the $Nest$ at $T_{2i(n+j+1)+3}$ for $1 \leq j \leq N$.*

Proof. We know from Lemma 5.3.3 that $Agent_1$ has its first meeting with $Agent_2$ to receive the first food unit at location $L_{i(n+3)+2}$ and time $T_{i(n+1)+1}$. After this meeting when $Agent_1$ turns around on the path towards the $Nest$, there are no agents to its left. Hence, it will not turn back again till it has reached the $Nest$. Since distance from the meeting location to the $Nest$ is $i(n+3)+2$ units, $Agent_1$ will reach there at $T_{i(2n+4)+3}$ and it will drop the first food unit at this time which proves that Lemma 5.3.6 holds for $j = 1$.

Substituting $k = 1$ in Lemma 5.3.5, we get that $Agent_1$ has its j^{th} meeting with $Agent_2$ to receive j^{th} food unit at location L_i and time $T_{i(2n+2j+1)+3}$ for $2 \leq j \leq n$. After every meeting, $Agent_1$ will take i steps from L_i to reach $Nest$ i.e. at $T_{i(2n+2j+2)+3}$ which proves that Lemma 5.3.6 also holds for $2 \leq j \leq N$. □

Substituting $j = N$ in Lemma 5.3.6 gives us the time step at which the n^{th} food unit is dropped at the $Nest$ by $Agent_1$ which is $T_{i(4n+2)+3}$ which leads to the following theorem:

Theorem 5.3.7. *Given n agents placed $2i + 1$ units of distance apart on a path from $Nest$ to $Food$, transportation of n units of food takes $i(4n + 2) + 3$ time steps using Relay-$Transportation$.*

We deduce from Corollary 5.3.1.1 and Theorem 5.3.7 that n agents take $2(n - 1)$ fewer time steps using *Relay-Transportation* to transport n units of food than the *End*-to-$End\ Transportation$.

Chapter 6

Empirical Analysis

In this chapter, we present the results of our experiments comparing the performance of all the algorithms described so far. First we compare the time taken by each of the algorithms in the exploration phase. Next we compare the time taken by each of the algorithms in the exploration as well as the communication phase. Finally, we compare the time taken for all three phases of foraging to be completed. We study the performance of the algorithms for different ranges of food locations.

6.1 Simulation

We developed our own simulator in Java for the comparison of these algorithms and it is available online[1]. We ran simulations keeping the grid size constant at $128 * 128$ with other varying parameters such as nest-food separation, the number of agents and the transportation techniques. We vary the *Nest-Food* separation in the following ranges: $0\% - 20\%$, $20\% - 40\%$, $40\% - 60\%$, $60\% - 80\%$, $80\% - 100\%$ meaning we choose the location of food randomly in such a way that its distance from the nest falls within the range. Every result shown in the following graphs has been averaged

[1]Source Code can be found at `https://code.google.com/p/foraging-sunidhi/source/browse/`

over 40 results taken from 40 randomly chosen food locations within the respective range of *Nest-Food* separation.

6.2 Exploration

First, we compare the time taken by all algorithms to locate the food in the grid. This time also includes the initial deployment phase time. As we have already established, some algorithms require the agents to move to some designated spots first so that they can coordinate their movements to meet each other regularly. While this may help them in the later communication phase, we analyze whether this overhead leads to a considerable increase in the exploration time. Figure 56 shows the performance

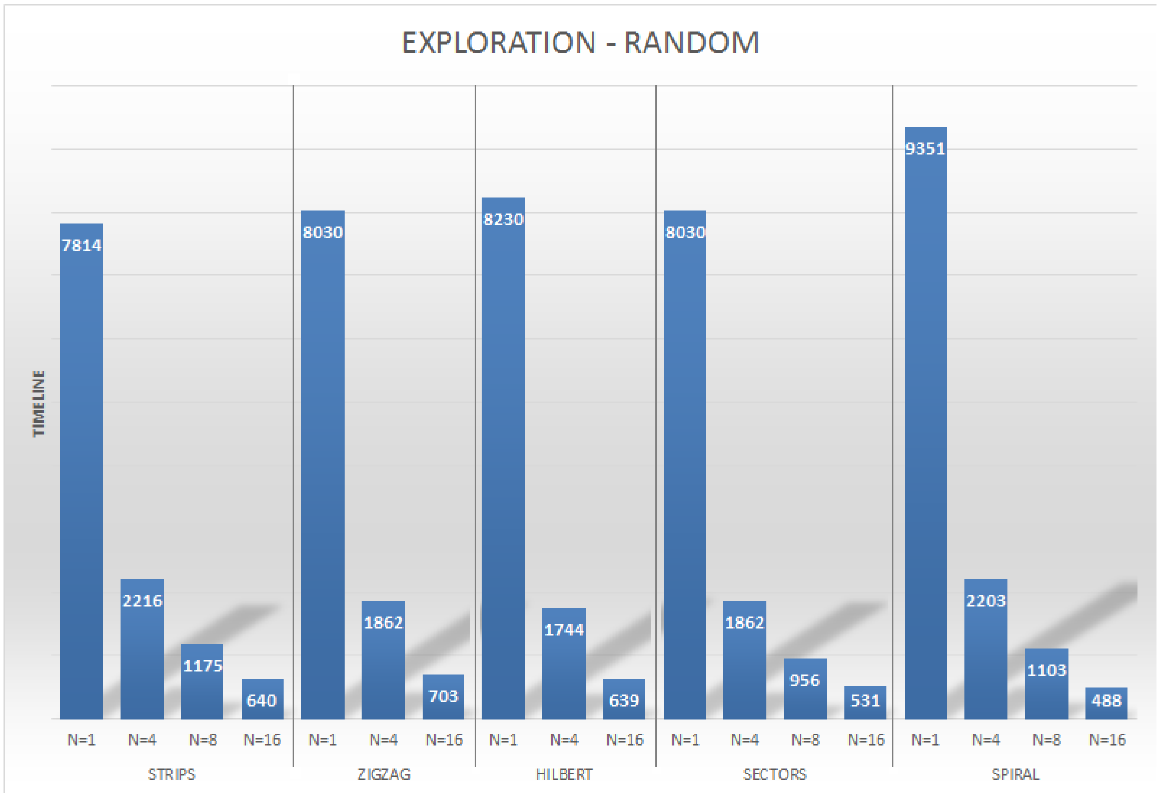


Figure 56: Exploration time when food is randomly placed

of all algorithms when food is placed at random locations in the grid. As expected, we observe that increasing the number of agents in the foraging process decreases the time taken to locate the food in almost every case. Figures 57, 58, 59, 60 and 61 present the exploration time for all algorithms with 1, 4, 8 and 16 agents when the Nest-Food separation is $0\% - 20\%$, $20\% - 40\%$, $40\% - 60\%$, $60\% - 80\%$ and $80\% - 100\%$ respectively.

As shown in Figure 57, breaking the usual trend, the time taken to locate the food by $Zigzag$ and $Hilbert$ algorithms increases with an increase in their number from 4 to 16 when the food is placed very near to the nest i.e. within 0%-20% of distance. This is because when there are only 1 or 4 agents on the grid, they start the search from the $Nest$ itself eliminating the need for *Initial Deployment* phase. However, when there are 16 agents on the grid, every agent goes to the center of its respective quadrant as shown in Figure 23 to start the search due to which the area closer to the nest is explored for food much later. So, increasing the number of agents has a detrimental effect in this case. Note that when the food is placed farther from the nest, the initial deployment phase does not have a detrimental effect for these two algorithms.

When the Nest-Food separation is between 0%-60%, *Spiral* [3] search gives the best results with $Sectors$ as a close second no matter the number of agents. *Strips* take significantly higher time when the food is located this near to the nest because the agents move to the end of the grid in their *Initial-Deployment* phases due to which they explore the region closer to the nest after covering almost half of the grid. However, *Strips* algorithm starts performing substantially better than the other algorithms when the separation between the nest and the food is greater than 60%. The performance of *Strips* also depends on which end of the grid the food is placed. Suppose the agents start the search from the west-end of the grid, then the food will be located almost immediately if the food was placed at this end and if it was on the

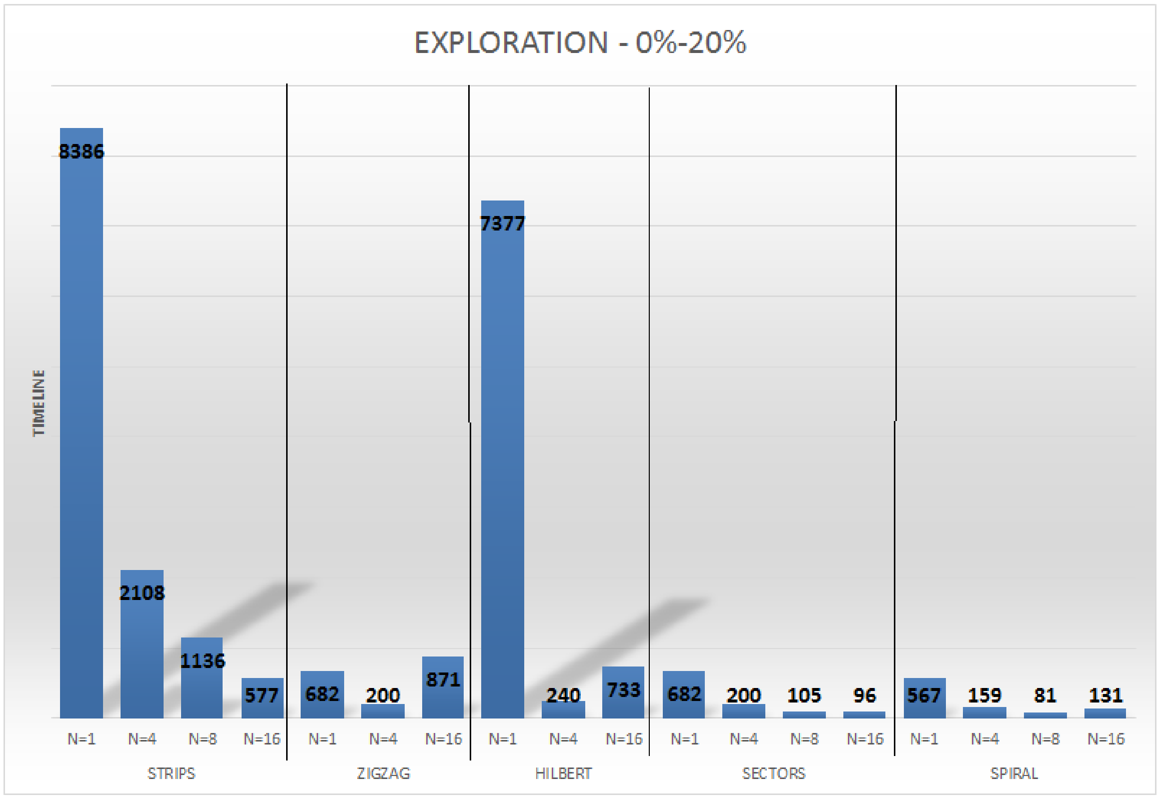


Figure 57: Exploration time when Nest-Food separation is 0%-20%

east-end, it will be located after the agents have covered almost all the grid. As we increase the *Nest-Food* separation further, the performance of *Spiral* [3] declines and *Sectors* algorithm starts performing better. In *Spiral* search, the agents have to modify their paths towards the end to limit their movement to the finite grid and many cells are visited several times because of that as shown in Figures 38, 39 and 40. The agents spiraling across the grid cease to give the best results when the food is placed far away from the nest. *Spiral* search was designed to cover an infinite grid while *Sectors*, on the other hand, was designed for a finite one, which is more realistic and does not have a strong dependency on the food location. When the food is placed in the farthest regions of the grid, *Strips* comes out as a clear winner because of the way it is designed irrespective of the number of agents as shown in Figure 61. The performance of *Spiral* deteriorates when the food is placed far away

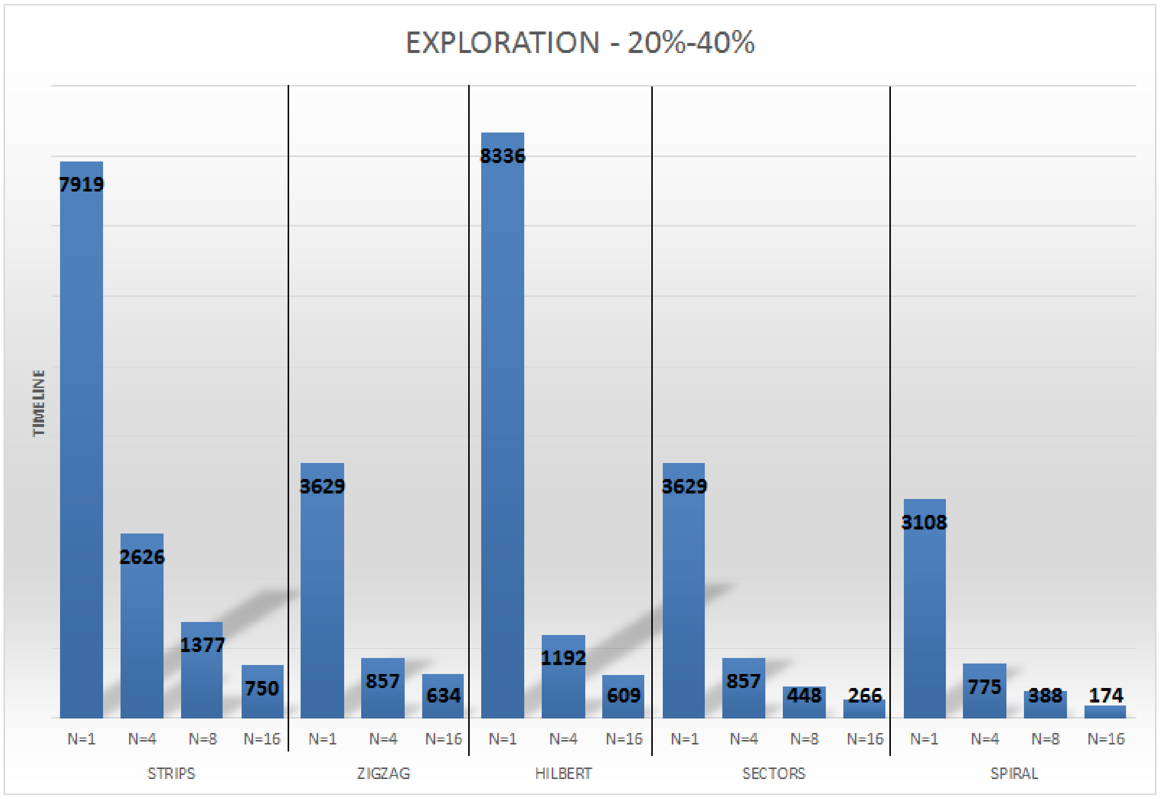


Figure 58: Exploration time when Nest-Food separation is 20%-40%

from the nest relative to the other algorithms. *Hilbert* performs better than *Zigzag* and *Sectors* because it is designed in such a way that it covers the closer regions to its starting point first, then goes out to the farthest regions and then come back for the region that falls in the middle range. Hence, the food gets located earlier.

6.3 Search and Communication

Now that we have seen the performance of the algorithms in locating the food, the next step is to compare them on how quick they are able to communicate their findings to the fellow agents. The communication time shown in graphs in this section is calculated from the moment the first agent locates the food till the time the last agent completes its *Communication Phase* and enters into the transportation phase.

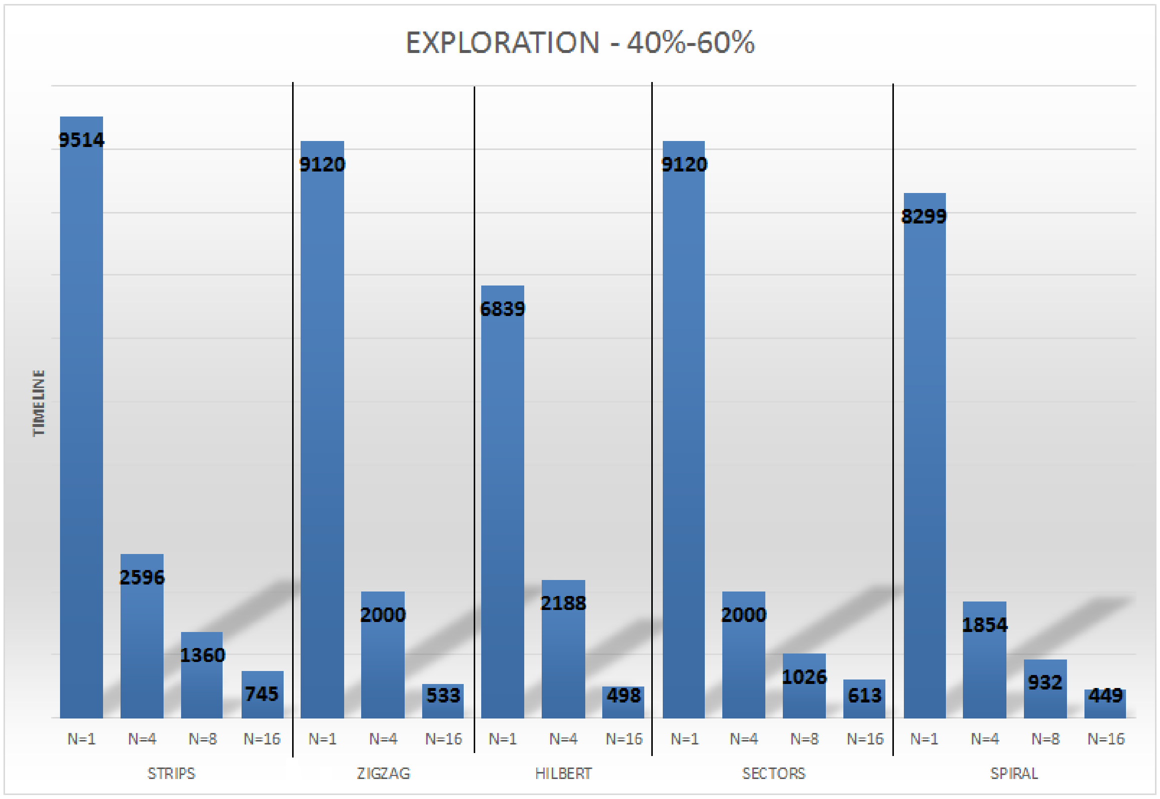


Figure 59: Exploration time when Nest-Food separation is 40%-60%

Note that for this period of time, some agents will still be communicating while others may already have been entered into the *Transportation Phase*.

Figure 62 shows the time taken to find the food plus the time taken for the last agent to complete the communication phase when the food is randomly placed over the grid. We observe that the communication time increases slightly with an increase in the number of agents for every algorithm. Figure 63 shows the ratio of the time taken in *Communication Phase* to the *Exploration Phase* time as a function of the number of agents. As we already know, the agents spend a lot of time in *Strips* algorithm in *Initial Deployment* phase to coordinate their movements to meet regularly. Due to this arrangement, we observe in our experiments that communication time for *Strips* is almost negligible as compared to the exploration time no matter the number of agents or Nest-Food Separation. For the other algorithms, communication time

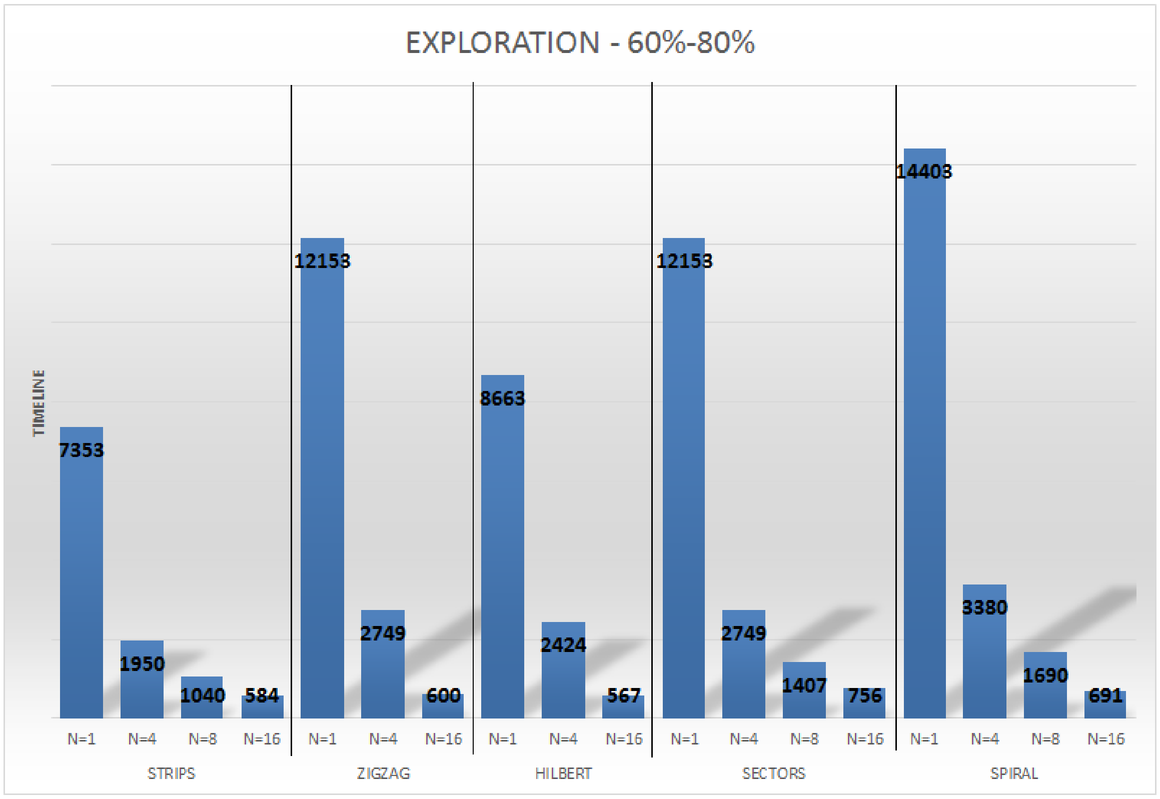


Figure 60: Exploration time when Nest-Food separation is 60%-80%

increases with an increase in the number of agents marginally except for *Zigzag* and *Hilbert* in which the communication time increases drastically because of the way agents start their exploration.

Figures 64, 65, 66, 67 and 68 present the exploration and the communication time for all algorithms with 4, 8 and 16 agents when the Nest-Food separation is $0\% - 20\%$, $20\% - 40\%$, $40\% - 60\%$, $60\% - 80\%$ and $80\% - 100\%$ respectively. Communication time also increases with an increase in the Nest-Food separation for all the algorithms except *Zigzag* and *Hilbert*. For these two algorithms, the communication time increases for a nest-food separation up to 40%, decreases for 40%-60% separation range and then increases again. This is because agents start their search from the region falling in 40%-60% separation range when there are 16 agents on the field and they are nearby each other in that period.

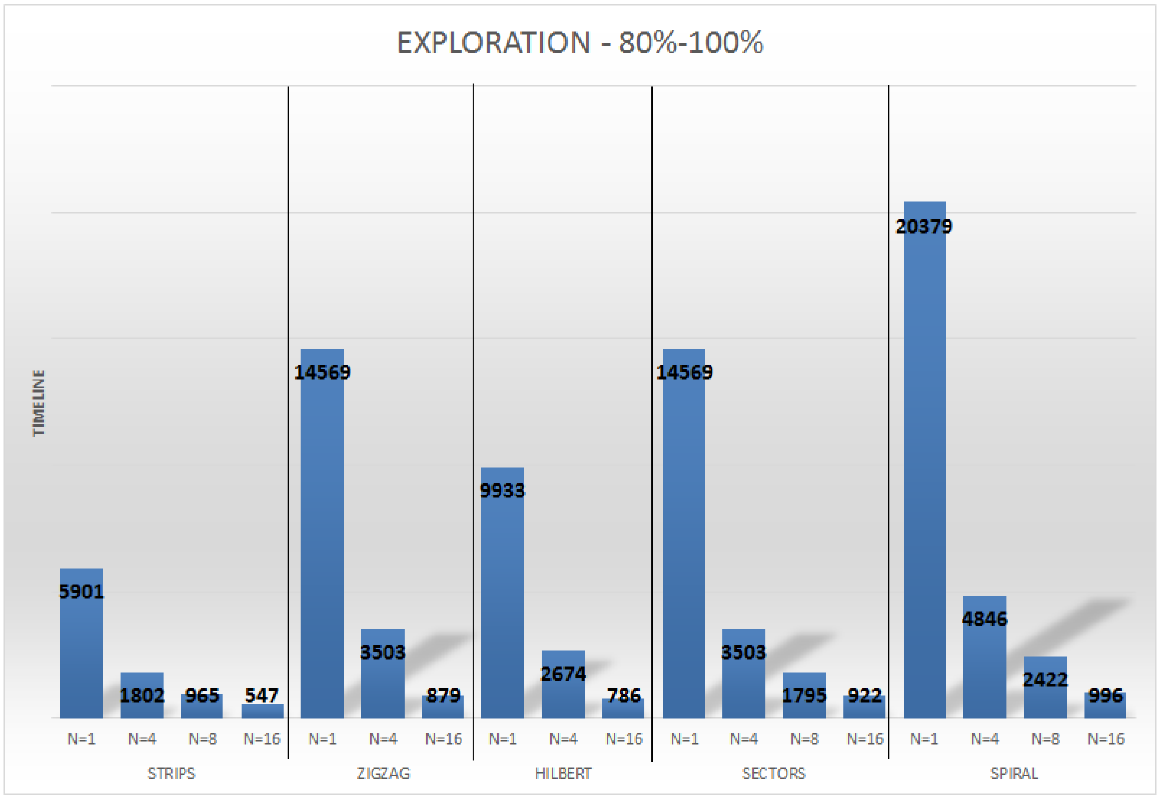


Figure 61: Exploration time when Nest-Food separation is 80%-100%

6.4 Exploration, Communication and Transportation

In this section, we compare the transportation costs incurred by all algorithms over different ranges of separation. In Figure 69, we compare the performance of the three *Transportation-Phase* algorithms i.e. *Relay-Food*, *Relay-Path* and *End-to-End*. We executed *Relay-Transportation* introduced in Chapter 5 in *two* variants: *Relay-Food* and *Relay-Path*. Since an agent is aware of the food location when it enters into *Transportation-Phase*, it can easily determine the relay path. In *Relay-Food* technique, the agent enters the relay path from the food location itself after picking one food unit from there whereas in *Relay-Path* technique, the agent enters the relay path by moving to the closest location on the path from its current location and then

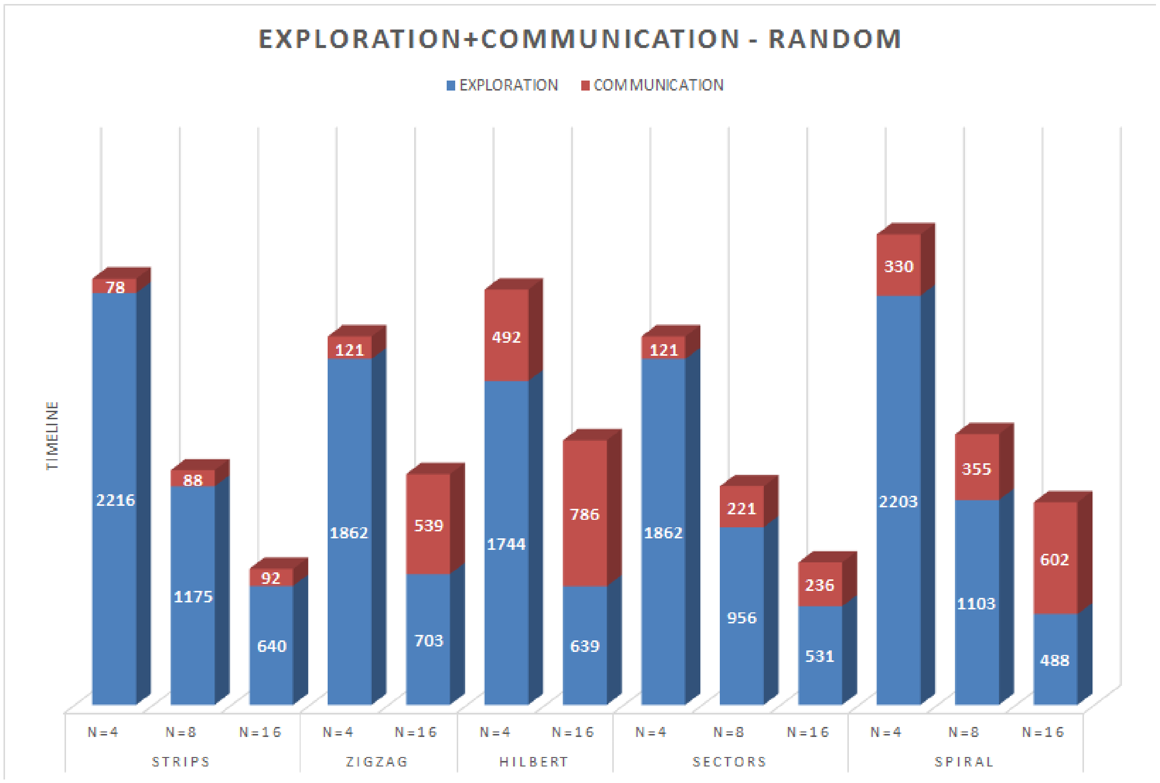


Figure 62: Exploration and Communication time when the food is randomly placed

starts moving towards the food. After the agent has entered the relay path following either one of these techniques, the rest of the algorithm to transport the food remains the same. We observe that *Relay-Food* transportation gives better results than the other two in every case. Therefore, in the rest of this section, we only use *Relay-Food* as the transportation algorithm.

There is a period of time in the execution of the algorithms in which some of the agents are communicating while some of them have already entered their transportation phases. In the previous section, we presented this time period as a part of the communication phase. However, in this section, we show this time period separately. Also, data used for the graphs shown in this section is given at the end in Figures 78 and 79.

Figure 70 presents the total execution time of the algorithms when the food is

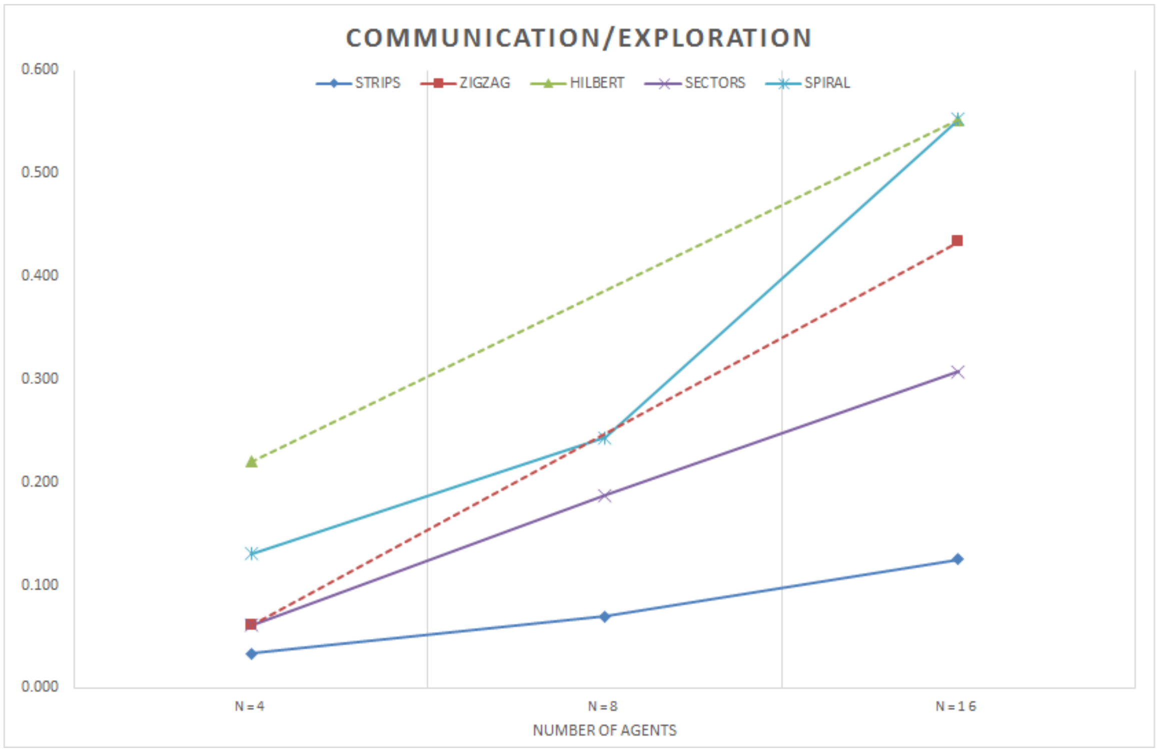


Figure 63: Ratio of the time taken in *Communication Phase* to the *Exploration Phase* time

placed randomly on the grid. As expected, the transportation time increases with increasing Nest-Food separation and decreases with an increase in the number of agents irrespective of the algorithm followed for exploration. We observe that *Sectors* algorithm gives the best performance no matter the number of agents with *Strips* as a close second.

Figures 71, 72, 73, 74 and 75 present the execution time for all algorithms with 4, 8 and 16 agents when the Nest-Food separation is $0\% - 20\%$, $20\% - 40\%$, $40\% - 60\%$, $60\% - 80\%$ and $80\% - 100\%$ respectively. As expected, we observe that the transportation time increases with the increasing separation between nest and the food.

We observe that for all the nest-food separations, the time period where some of the agents are communicating while some are transporting keeps increasing with an increase in the number of agents irrespective of the algorithm they follow for the

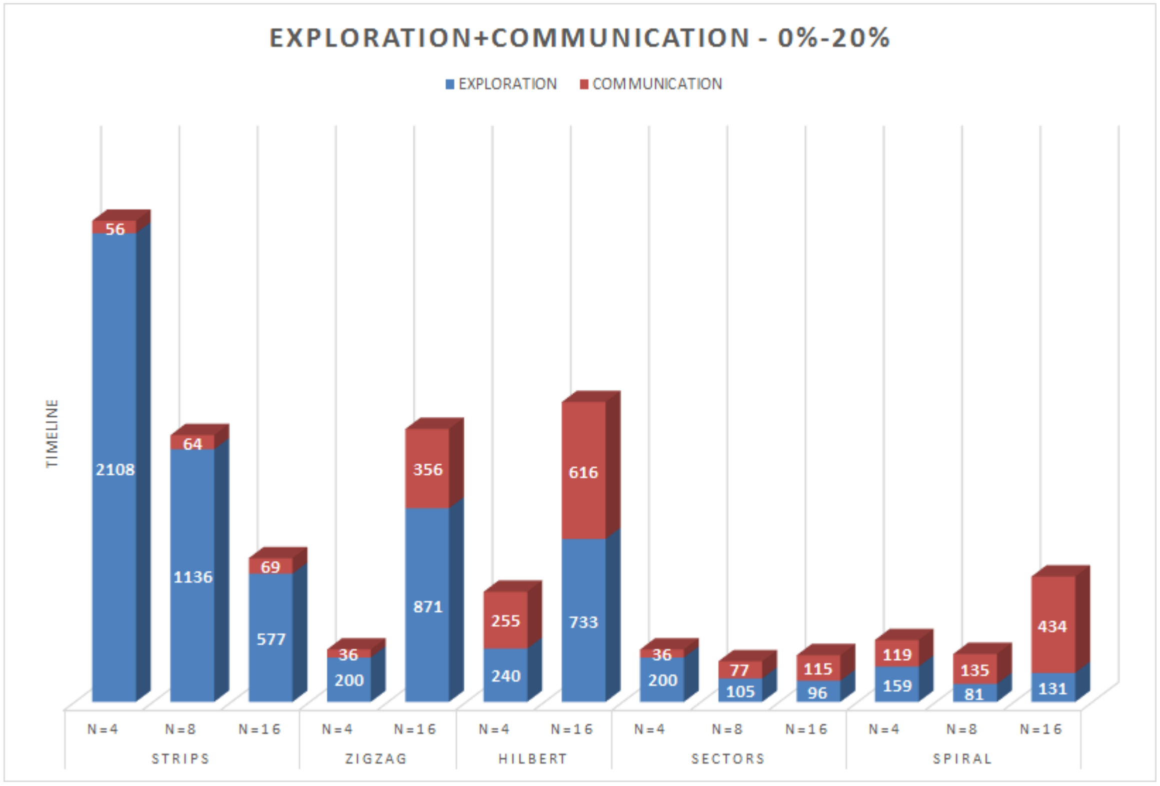


Figure 64: Search and Communication time when the Nest-Food separation is 0%-20%

exploration phase. Also, this time period is the longest for *Spiral Search* [3] followed by *Hilbert* and the shortest for *Strips* irrespective of the number of agents.

Although the time spent in the *Transportation Phase* is almost the same for all the algorithms for a given number of agents, we observe that it is the lowest for *Strips* among all. This is because whenever an agent locates the food in *Strips*, it communicates this information to its neighbors quickly as compared to the other algorithms. Therefore, the agents are not that far from the food location when they get this information and it does not take them long to reach there. *Four* agents following *zigzag* strategy for the exploration also reach the food location quickly because of the same reason. *Spiral Search* [3] has the longest communication plus transportation period which makes its transportation phase the costliest compared to the other algorithms with the same number of agents and same nest-food separation.

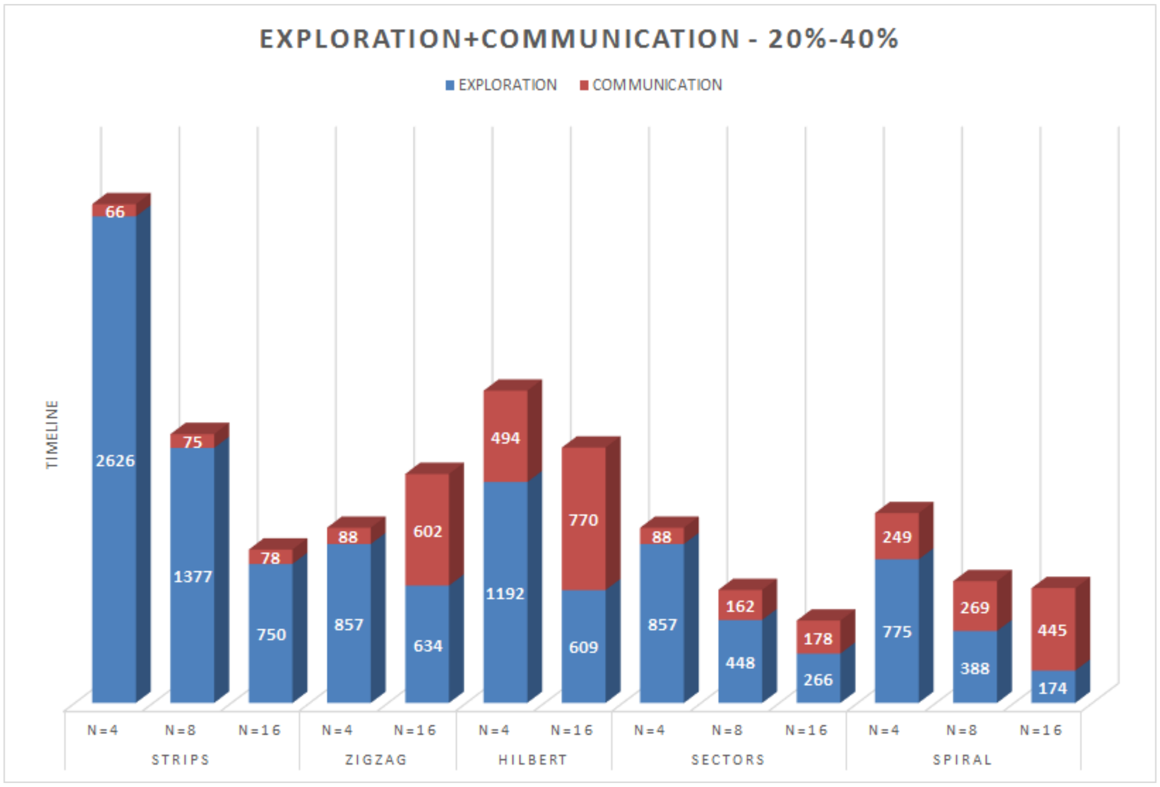


Figure 65: Search and Communication time when the Nest-Food separation is 20%-40%

Figure 76 shows the percentage of total time of execution spent in different phases. It is quite clear that *Strips* spends the minimum portion of the execution time in the communication phase. With $four$ agents on the grid, *Zigzag* and *Sectors* spends very less time in the communication phase and there is no time period when some agents are communicating while some are transporting meaning that all $four$ agents enter the *Transportation Phase* at the same time. When there are 16 agents on the grid following *Zigzag* and *Hilbert*, they spend a considerable amount of their execution time communicating. It is also quite clear that the time spent in *Communication phase* increases consistently with an increase in the number of agents for *Spiral search* [3] and *Sectors*.

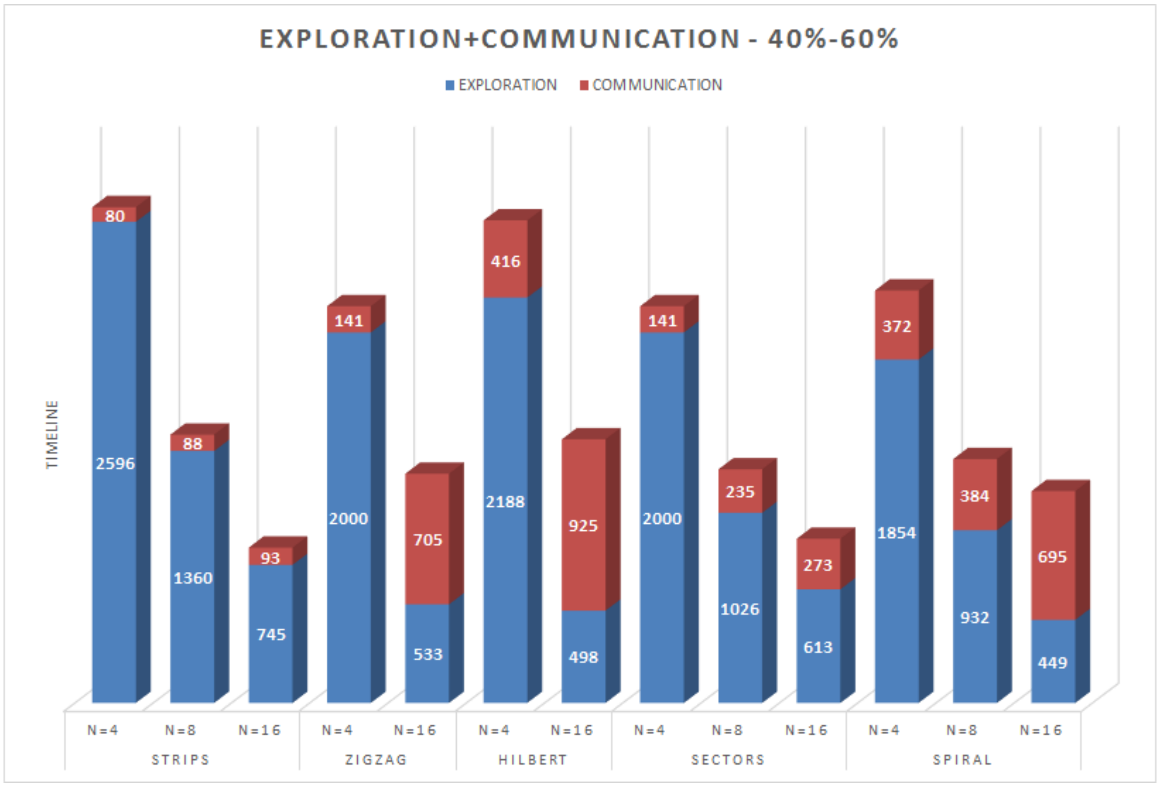


Figure 66: Search and Communication time when the Nest-Food separation is 40%-60%

6.5 Modified Zigzag

We developed a variant of the zigzag algorithm for 8 agents wherein half of them start the search from the end of the grid and half of them start from the nest itself. Figure 77 presents the comparison of this variant with the other algorithms. We observe that it gives the best results when the food is placed in the farthest regions of the grid, even better than the *Strips* algorithm which gave very good results when the food was placed far in the grid.

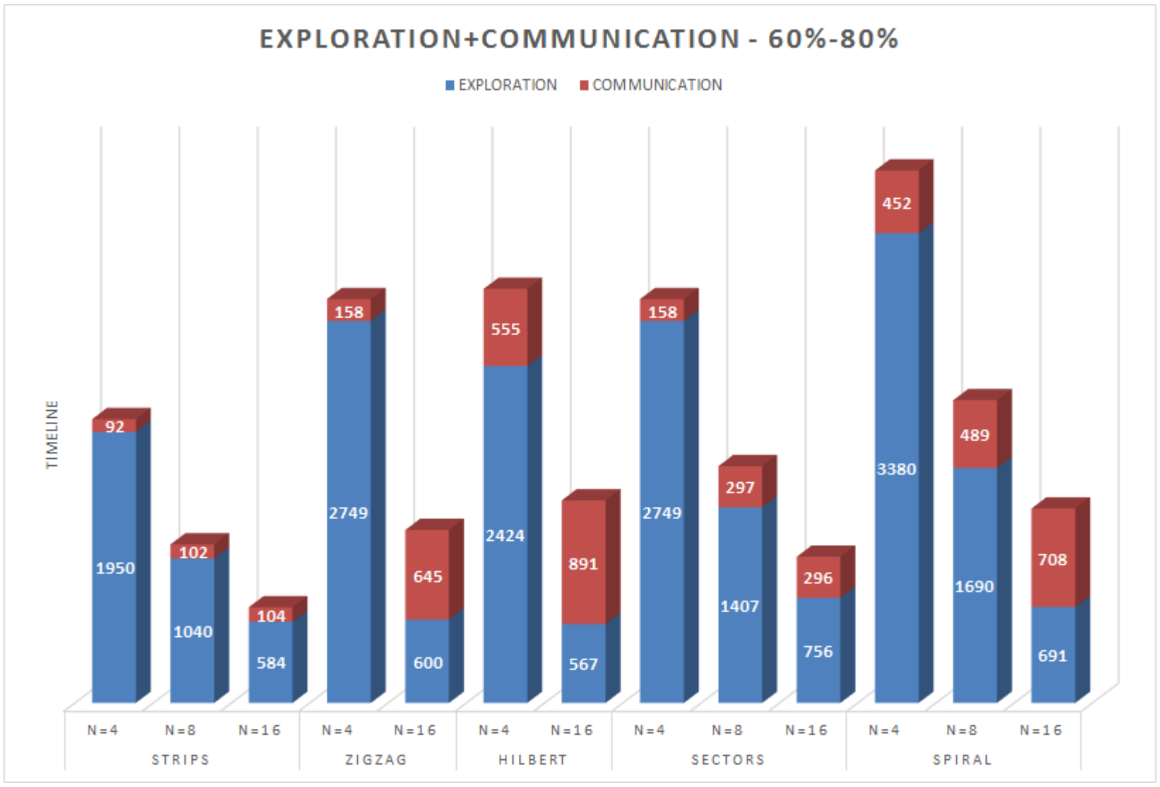


Figure 67: Search and Communication time when the Nest-Food separation is 60%-80%

6.6 Discussion

All our algorithms divide the grid so that the agents can search in parallel. However, the exact manner of dividing the grid can affect many factors influencing the performance of the foraging algorithms:

- How much time is spent in the *Initial Deployment phase*?
- How often do the agents meet?
- How many cells are traversed repeatedly?

We found from our simulations that although the agents following *Strips* algorithm spend extra time in the *Initial Deployment Phase* for dividing up the grid into strips, they meet regularly and none of the cells are traversed twice in the *Exploration Phase*.

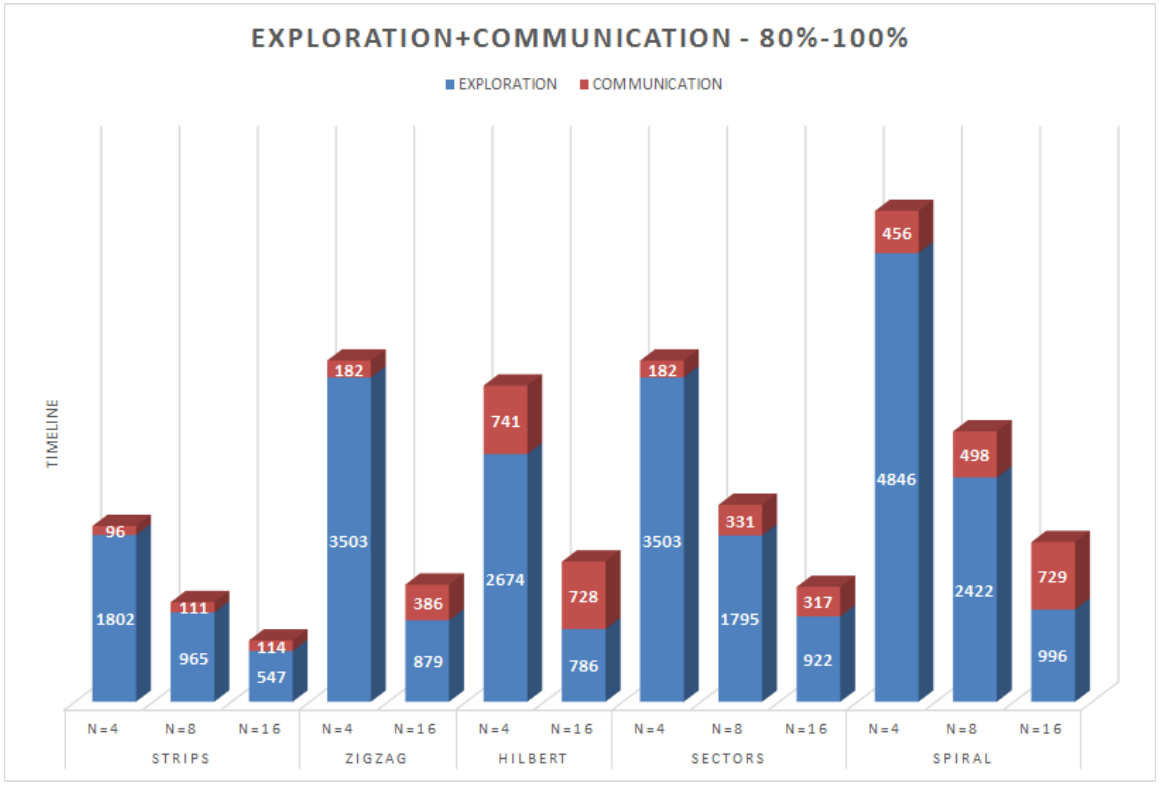


Figure 68: Search and Communication time when the Nest-Food separation is 80%-100%

Also, because of the quick communication, they perform slightly better in *Transportation Phase* too. For *Zigzag* and *Hilbert* with only 4 agents, they divide the grid into *four* quadrants and there is no *Initial Deployment Phase* and they perform the best in locating the food. Their performance declines with an increase in the number of agents as compared to the other algorithms. *Spiral Search* [3] gave the best results at locating the food when the nest-food separation is up to 60% of distance. However, its performance deteriorates with an increase in this separation as this strategy was designed for an infinite grid and a lot of cells are traversed repeatedly when we adjusted it for our finite grid model which affected its performance. Also, its communication cost is also high as compared to the other algorithms. *Sectors* came out to be the best performer on an average as expected. It is quick at locating the food and the communication cost is also low as compared to the other algorithms since the agents are always close to their neighbors in their respective sectors.

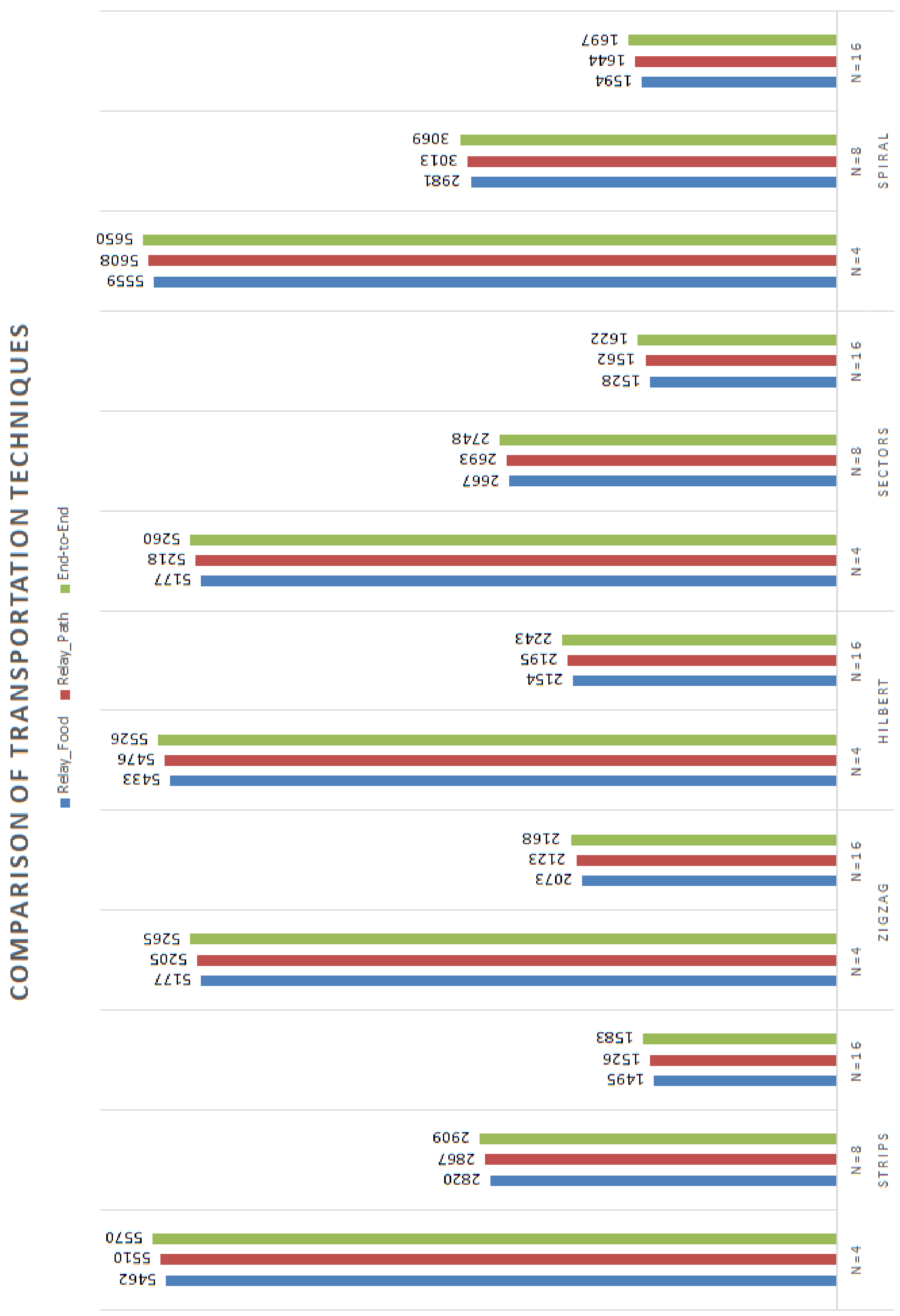


Figure 69: Comparison of different Transportation algorithms

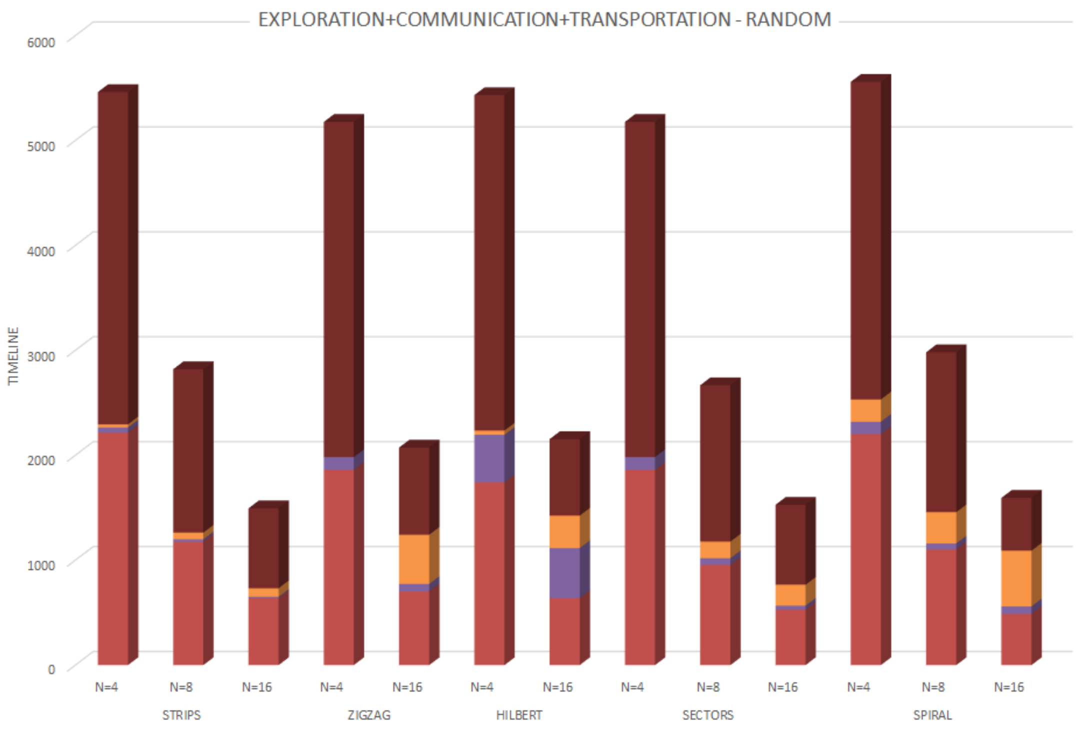


Figure 70: Total execution time when food is randomly placed over the grid

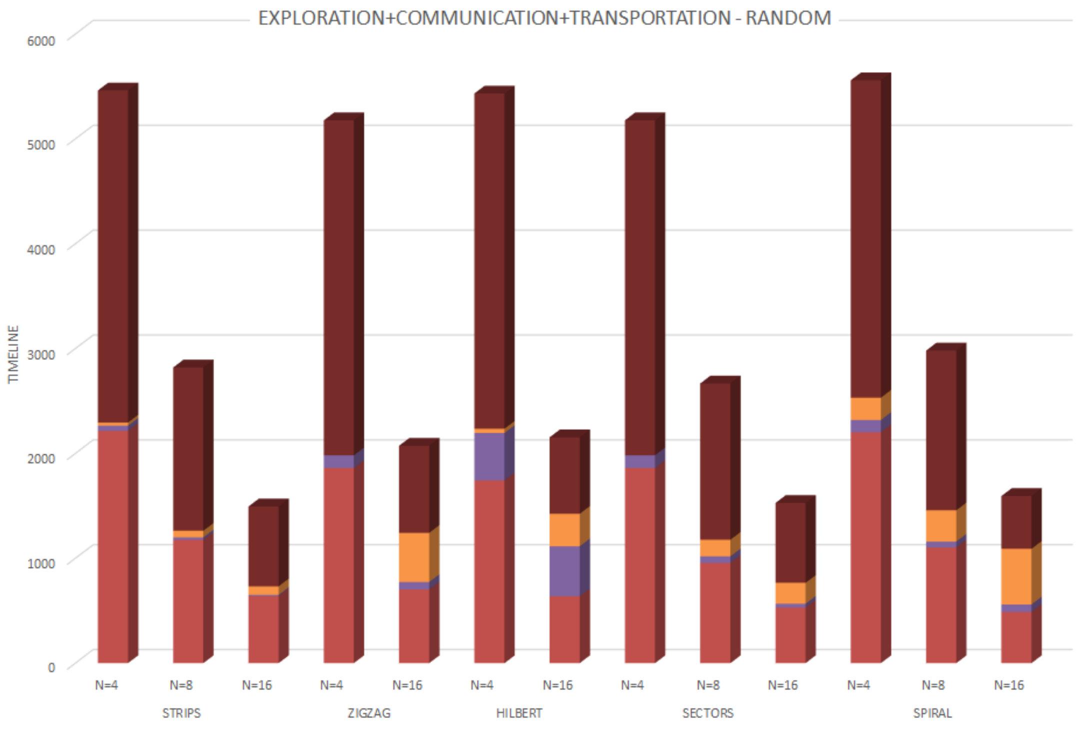


Figure 71: Total execution time when Nest-Food separation is 0%-20%

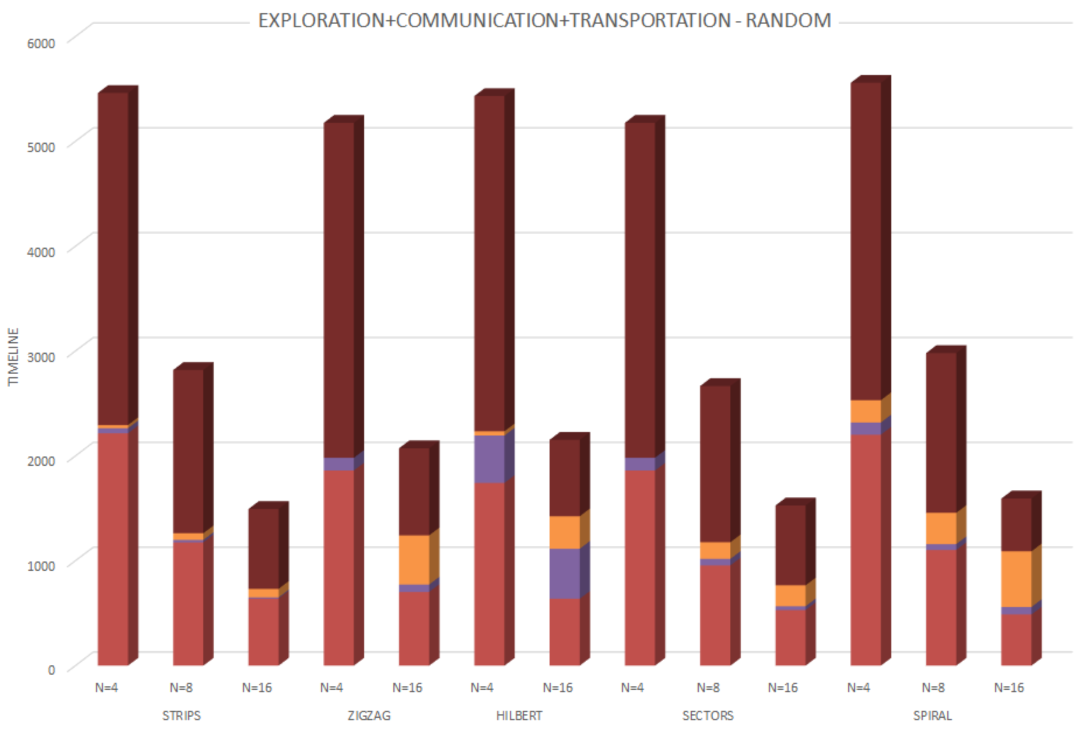


Figure 72: Total execution time when Nest-Food separation is 20%-40%

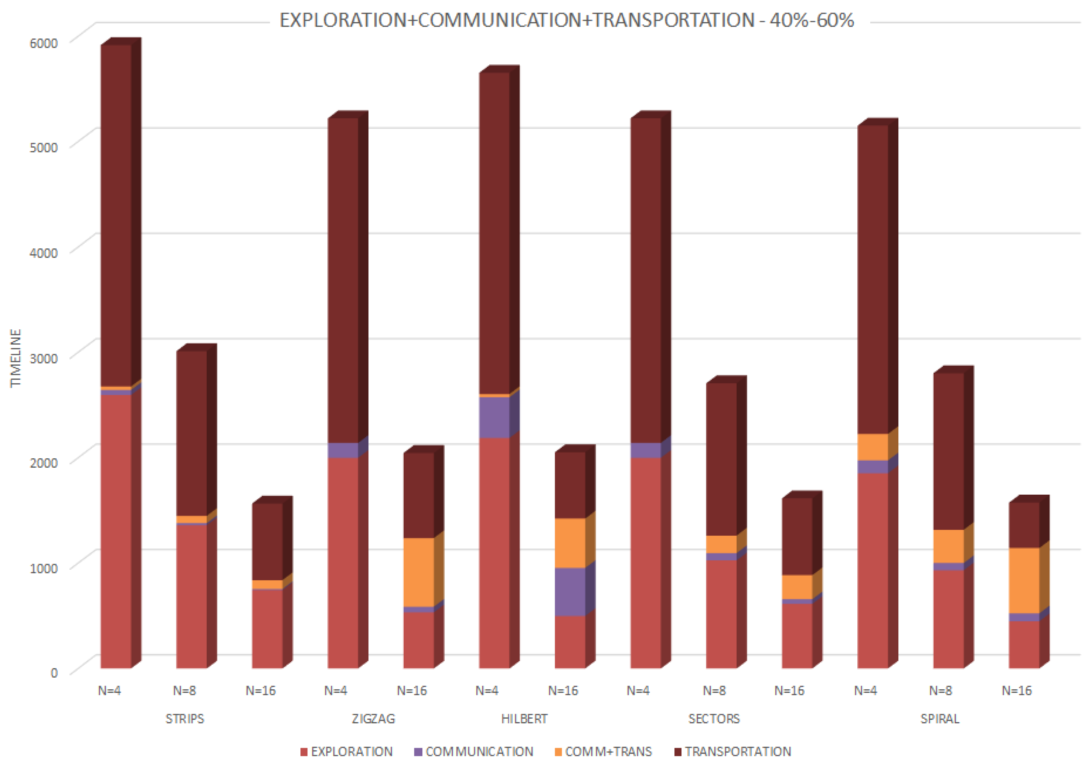


Figure 73: Total execution time when Nest-Food separation is 40%-60%

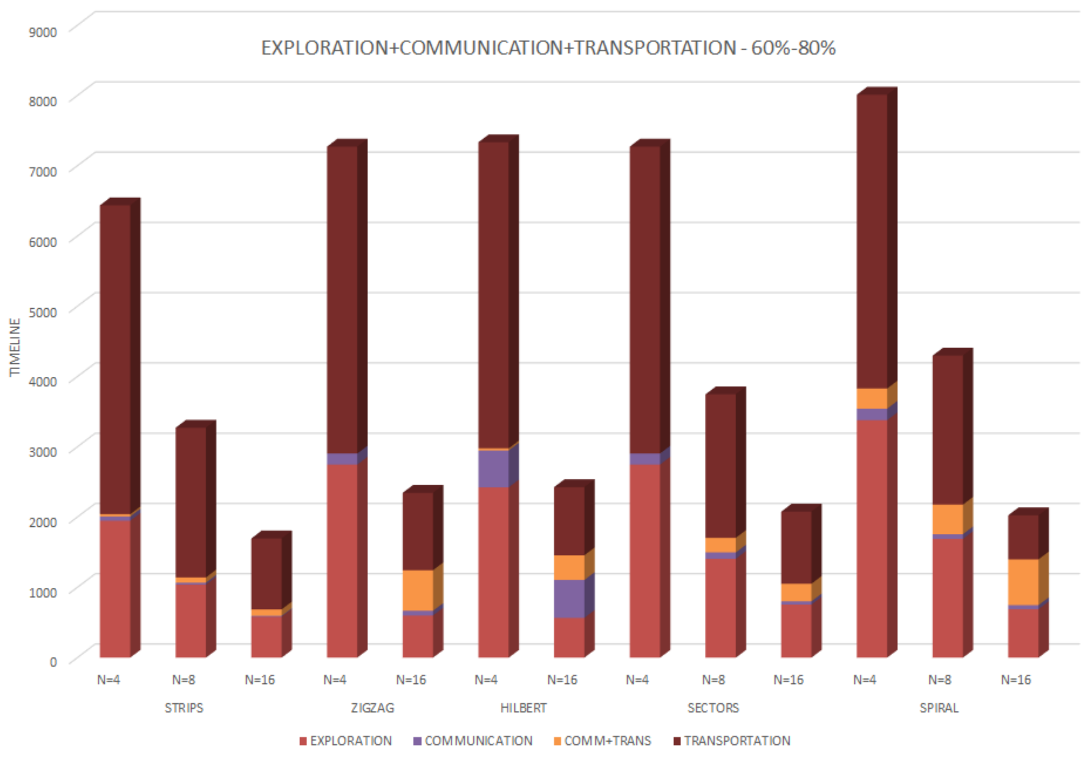


Figure 74: Total execution time when Nest-Food separation is 60%-80%

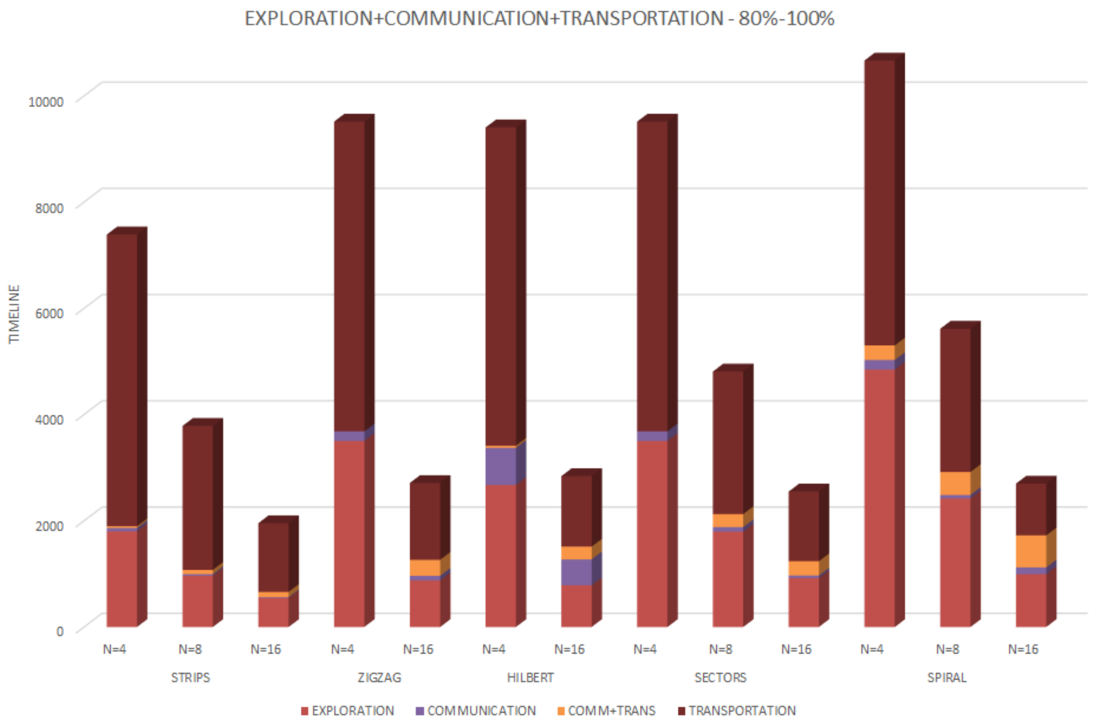
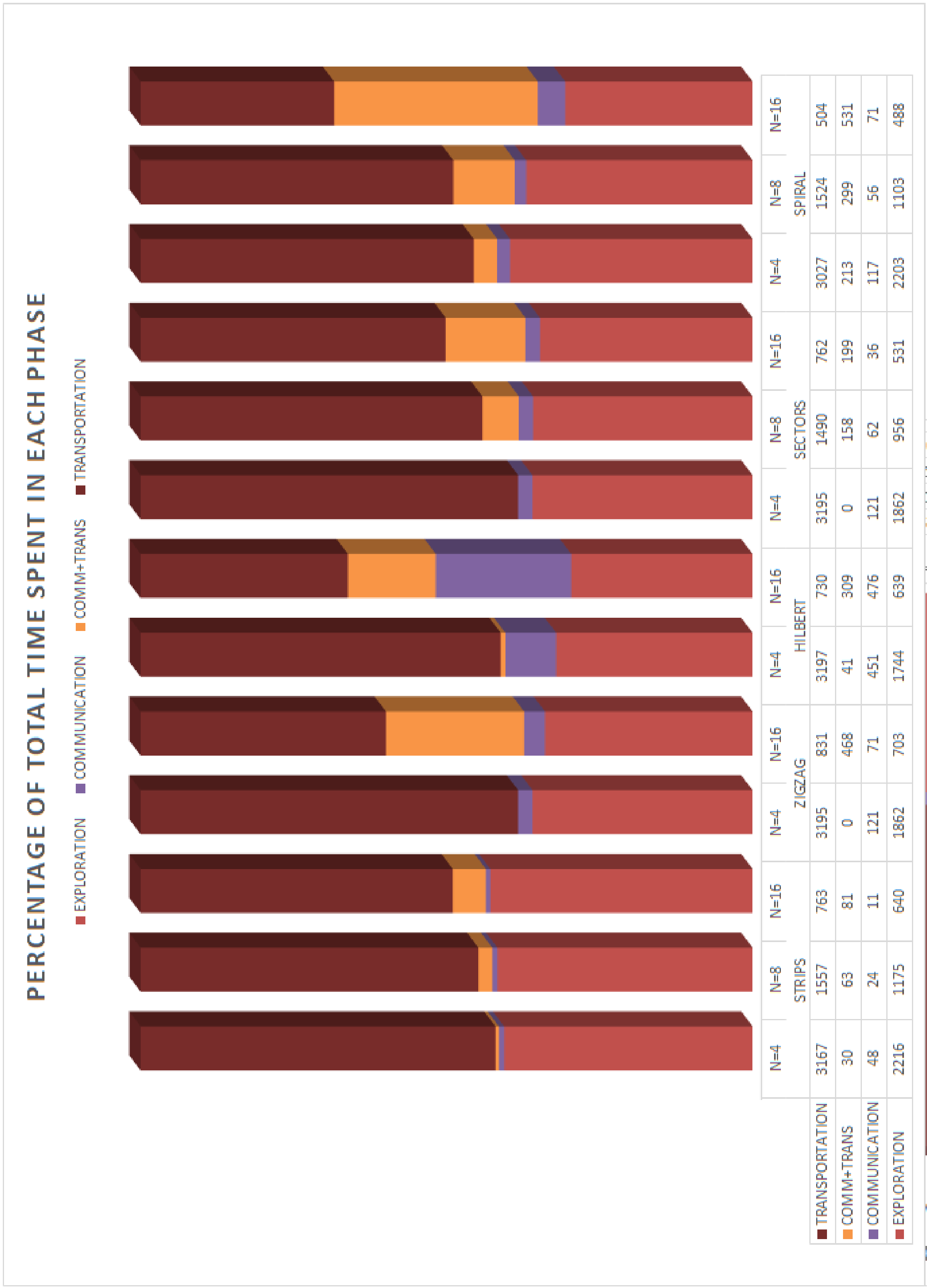
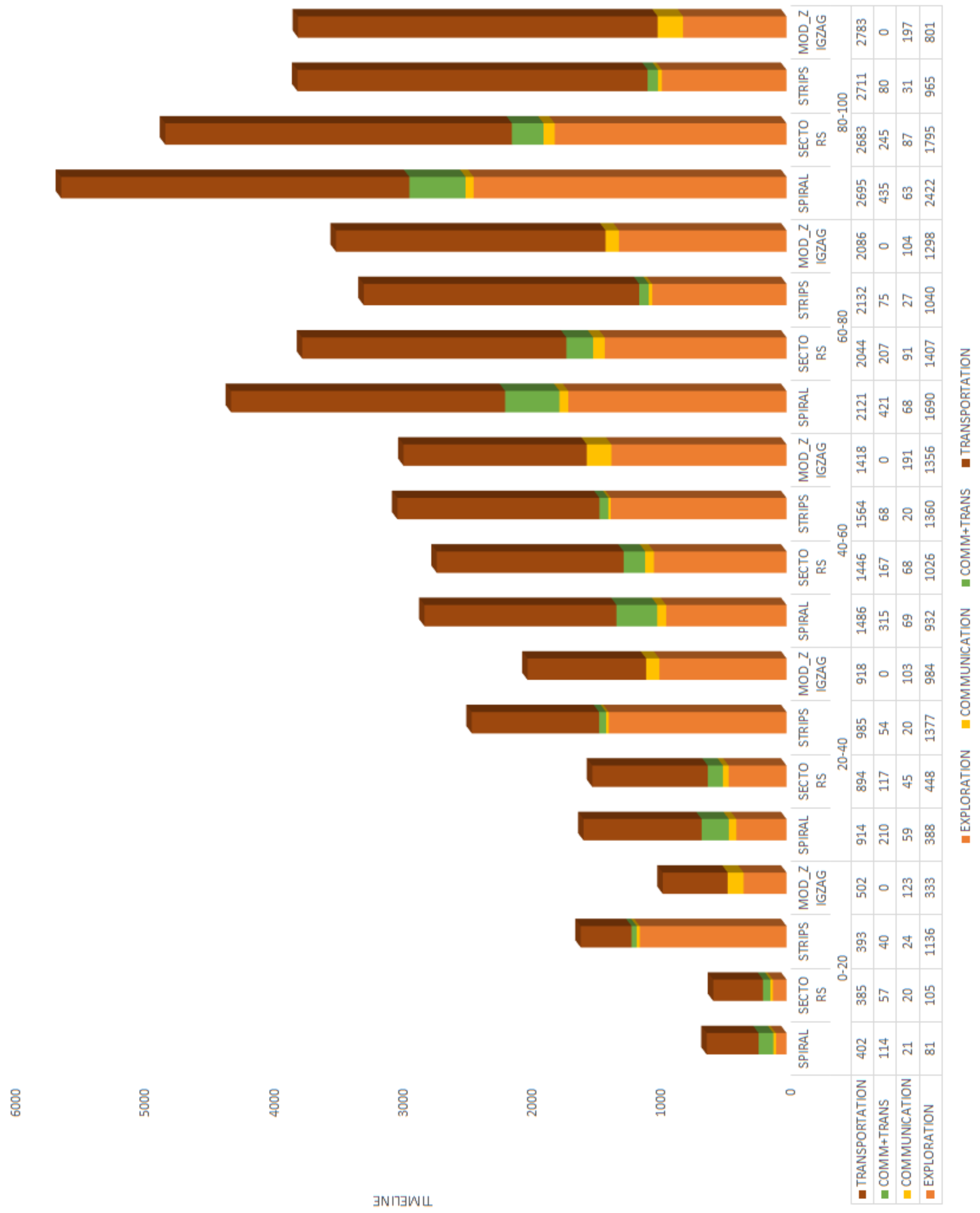


Figure 75: Total execution time when Nest-Food separation is 80%-100%



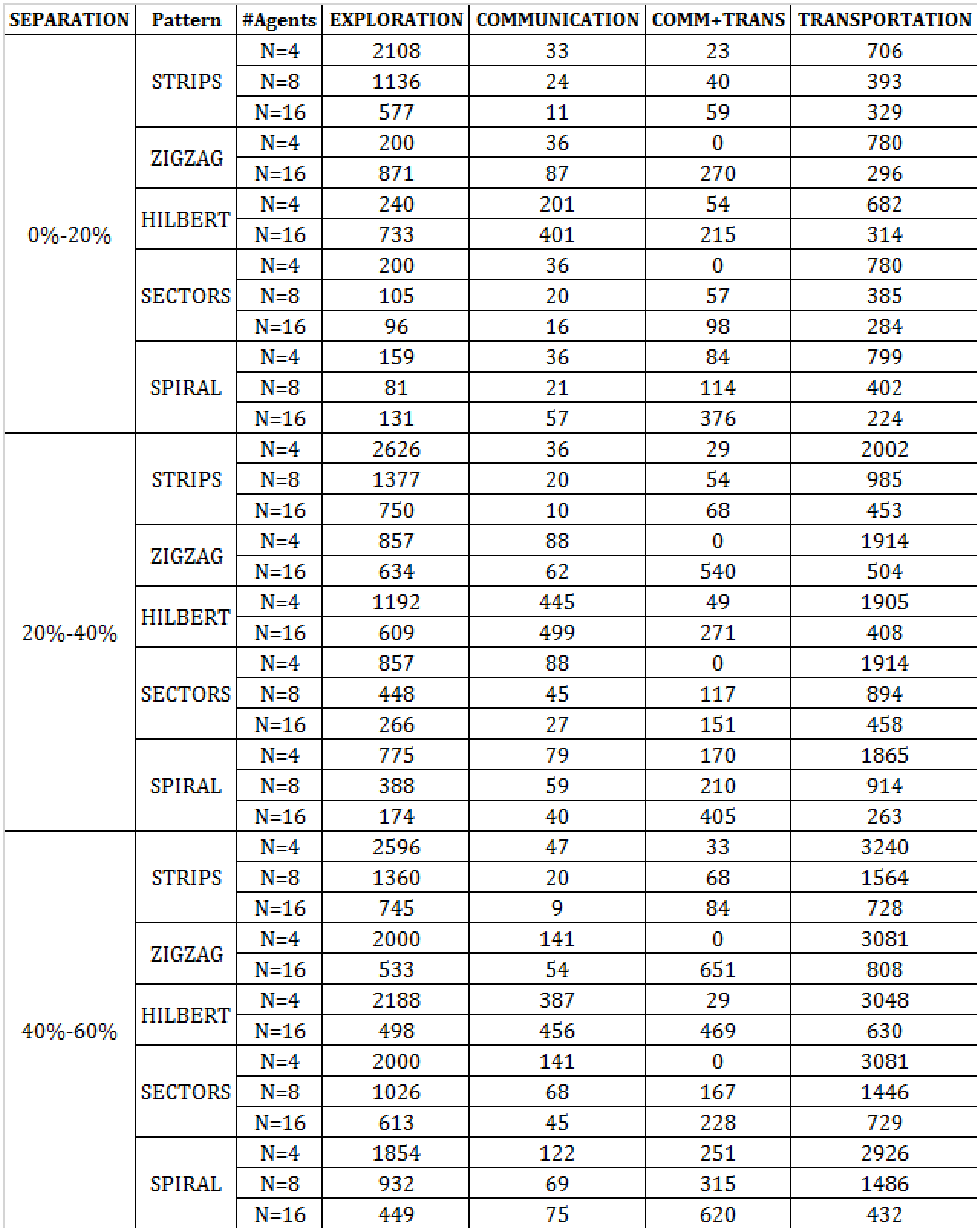
	STRIPS			ZIGZAG		HILBERT		SECTORS			SPIRAL		
	N=4	N=8	N=16	N=4	N=16	N=4	N=16	N=4	N=8	N=16	N=4	N=8	N=16
🟥 TRANSPORTATION	3167	1557	763	3195	831	3197	730	3195	1490	762	3027	1524	504
🟧 COMM+TRANS	30	63	81	0	468	41	309	0	158	199	213	299	531
🟪 COMMUNICATION	48	24	11	121	71	451	476	121	62	36	117	56	71
🟥 EXPLORATION	2216	1175	640	1862	703	1744	639	1862	956	531	2203	1103	488

Figure 76: Percentage of the total execution time spent in different phases



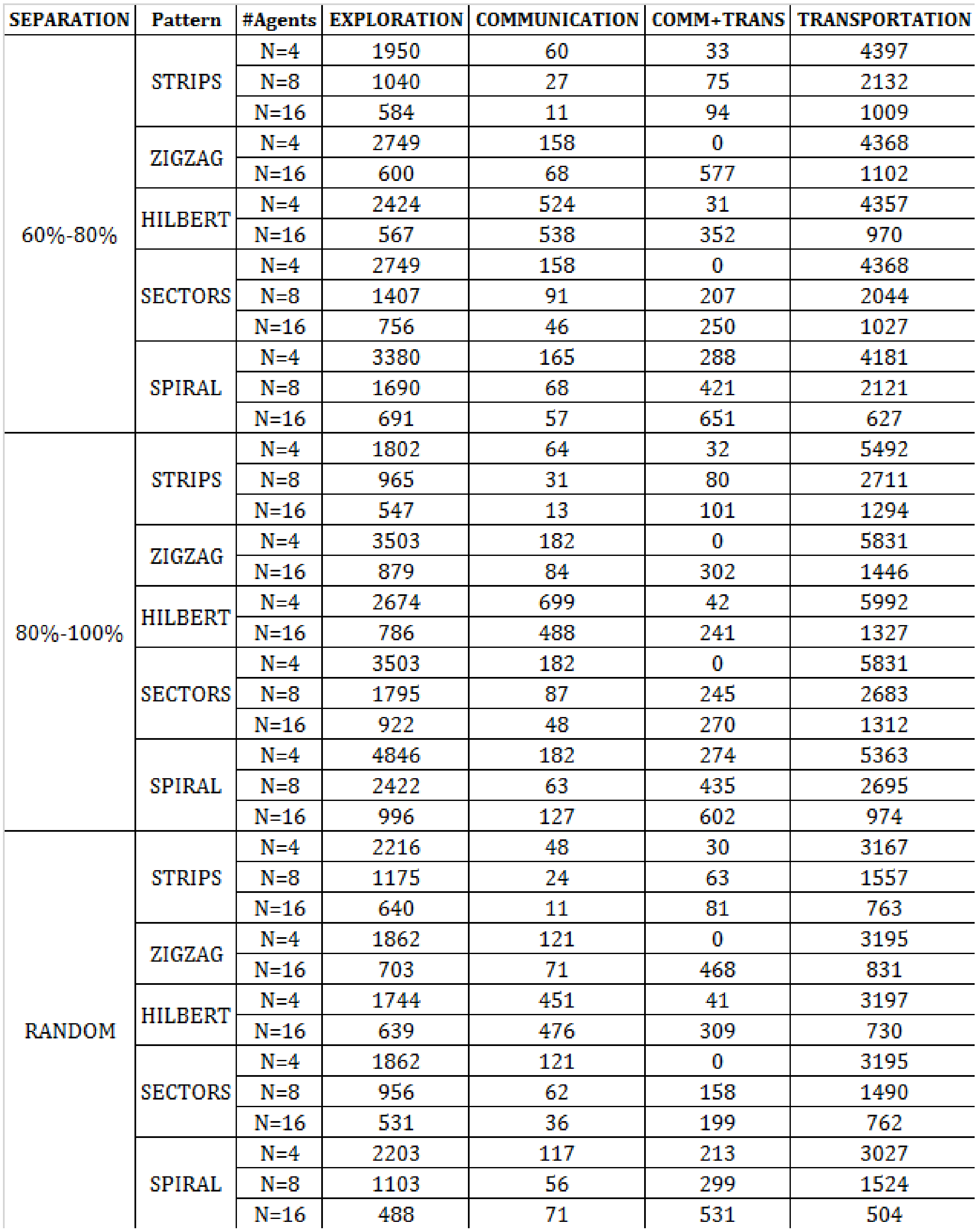
	0-20				20-40				40-60				60-80				80-100			
	SPIRAL	SECTORS	STRIPS	MOD_ZIGZAG	SPIRAL	SECTORS	STRIPS	MOD_ZIGZAG	SPIRAL	SECTORS	STRIPS	MOD_ZIGZAG	SPIRAL	SECTORS	STRIPS	MOD_ZIGZAG	SPIRAL	SECTORS	STRIPS	MOD_ZIGZAG
🟫 TRANSPORTATION	402	385	393	502	914	894	985	918	1486	1446	1564	1418	2121	2044	2132	2086	2695	2683	2711	2783
🟩 COMM+TRANS	114	57	40	0	210	117	54	0	315	167	68	0	421	207	75	0	435	245	80	0
🟨 COMMUNICATION	21	20	24	123	59	45	20	103	69	68	20	191	68	91	27	104	63	87	31	197
🟧 EXPLORATION	81	105	1136	333	388	448	1377	984	932	1026	1360	1356	1690	1407	1040	1298	2422	1795	965	801

Figure 77: Comparison of Modified Zigzag algorithm with other algorithms when the number of agents is 8



SEPARATION	Pattern	#Agents	EXPLORATION	COMMUNICATION	COMM+TRANS	TRANSPORTATION
0%-20%	STRIPS	N=4	2108	33	23	706
		N=8	1136	24	40	393
		N=16	577	11	59	329
	ZIGZAG	N=4	200	36	0	780
		N=16	871	87	270	296
	HILBERT	N=4	240	201	54	682
		N=16	733	401	215	314
	SECTORS	N=4	200	36	0	780
		N=8	105	20	57	385
		N=16	96	16	98	284
	SPIRAL	N=4	159	36	84	799
		N=8	81	21	114	402
		N=16	131	57	376	224
20%-40%	STRIPS	N=4	2626	36	29	2002
		N=8	1377	20	54	985
		N=16	750	10	68	453
	ZIGZAG	N=4	857	88	0	1914
		N=16	634	62	540	504
	HILBERT	N=4	1192	445	49	1905
		N=16	609	499	271	408
	SECTORS	N=4	857	88	0	1914
		N=8	448	45	117	894
		N=16	266	27	151	458
	SPIRAL	N=4	775	79	170	1865
		N=8	388	59	210	914
		N=16	174	40	405	263
40%-60%	STRIPS	N=4	2596	47	33	3240
		N=8	1360	20	68	1564
		N=16	745	9	84	728
	ZIGZAG	N=4	2000	141	0	3081
		N=16	533	54	651	808
	HILBERT	N=4	2188	387	29	3048
		N=16	498	456	469	630
	SECTORS	N=4	2000	141	0	3081
		N=8	1026	68	167	1446
		N=16	613	45	228	729
	SPIRAL	N=4	1854	122	251	2926
		N=8	932	69	315	1486
		N=16	449	75	620	432

Figure 78: Simulation results-1



SEPARATION	Pattern	#Agents	EXPLORATION	COMMUNICATION	COMM+TRANS	TRANSPORTATION
60%-80%	STRIPS	N=4	1950	60	33	4397
		N=8	1040	27	75	2132
		N=16	584	11	94	1009
	ZIGZAG	N=4	2749	158	0	4368
		N=16	600	68	577	1102
	HILBERT	N=4	2424	524	31	4357
		N=16	567	538	352	970
	SECTORS	N=4	2749	158	0	4368
		N=8	1407	91	207	2044
		N=16	756	46	250	1027
	SPIRAL	N=4	3380	165	288	4181
		N=8	1690	68	421	2121
		N=16	691	57	651	627
80%-100%	STRIPS	N=4	1802	64	32	5492
		N=8	965	31	80	2711
		N=16	547	13	101	1294
	ZIGZAG	N=4	3503	182	0	5831
		N=16	879	84	302	1446
	HILBERT	N=4	2674	699	42	5992
		N=16	786	488	241	1327
	SECTORS	N=4	3503	182	0	5831
		N=8	1795	87	245	2683
		N=16	922	48	270	1312
	SPIRAL	N=4	4846	182	274	5363
		N=8	2422	63	435	2695
		N=16	996	127	602	974
RANDOM	STRIPS	N=4	2216	48	30	3167
		N=8	1175	24	63	1557
		N=16	640	11	81	763
	ZIGZAG	N=4	1862	121	0	3195
		N=16	703	71	468	831
	HILBERT	N=4	1744	451	41	3197
		N=16	639	476	309	730
	SECTORS	N=4	1862	121	0	3195
		N=8	956	62	158	1490
		N=16	531	36	199	762
	SPIRAL	N=4	2203	117	213	3027
		N=8	1103	56	299	1524
		N=16	488	71	531	504

Figure 79: Simulation results-2

Chapter 7

Conclusion and Future Work

This chapter summarizes the thesis and contains some recommendations for future work concerned with the distributed algorithms for a swarm of agents to forage for food on a finite grid and related topics.

7.1 Conclusion

We studied the central place foraging problem in this thesis: Given n robots initially present at the nest at the center of a square region, and multiple food units at a single unknown location in the region, find an efficient algorithm for the robots to collectively discover the food and bring it all back to the nest. We divided the problem into three phases: exploration, communication and transportation. The main aim of the *Exploration* phase was to find efficient ways to search the space so that the food gets discovered as soon as possible no matter where it is placed. We gave five different algorithms for the execution of this phase: Strips, Zigzag, Hilbert, Sectors and Modified Zigzag. We analyzed the competitive ratios of all our algorithms, and ran extensive simulations to analyze their performance. We also implemented the Spiral Search pattern introduced by Lopez-Ortiz [3] and compared its performance with our algorithms. Our experiments show that Strips is the fastest at locating the food when the food is relatively far from the nest, while Spiral Search is the best at locating the food when the food is relatively close to the nest. However, Sectors algorithm is the best on an average when the food is placed at any random location. The Zigzag and Hilbert algorithms have the best performance for 4 agents at locating the food but do not do so well with other number of agents.

For the *Communication* phase, the agents just need to know the set of neighbors to whom they are supposed to communicate as they are capable of determining the locations of their neighbors and reaching them. The time taken to complete this communication phase depends on how the grid is divided initially and the movement pattern the agents follow to cover the ground. We found that Strips algorithm has the cheapest communication phase. Zigzag and Hilbert have fast communication when there are only 4 agents. However, this cost increases drastically when there are more agents because of the *Initial-Deployment* phase. The Sectors algorithm has consistently average communication phase over different number of agents and various nest-food separations. We also found out that the communication phase is costlier for the Spiral Search than the Sectors algorithm and it gets costlier with the increasing number of agents and the nest-food separation.

We investigated two approaches for the transportation phase of the problem: *Relay* and *End*-to-*End*. We found out that relay transportation performs better than the latter. We also investigated their performance theoretically by considering the case when agents are placed equidistant on the path between the nest and the food and analyzing the time taken by them to transport the food back to the nest using both the approaches.

7.2 Future Work

There is a lot of scope for further research in this field. One direction is to try to find ways to make the exploration algorithms fault-tolerant, i.e., handle the situations in which some of the agents are not able to finish their share of the execution. There can be a lot of reasons for agent failure in the real world. For example, they can die of energy deprivation. Another direction for investigation is to find ways for the agents to re-adjust their paths if they come across obstacles on the grid, which is a more realistic situation than having a square grid with no obstacles. We consider the case in this thesis where all the food units are placed at a single location in the grid. Another possible direction is to investigate the case where the food is scattered all over the grid, and all the food units have to be brought back to the nest.

In this thesis, we do not take into account the costs associated with different kinds of operations performed by the agents such as carrying food, picking up and dropping of food, exchange of food between agents, transmitting and receiving data, among many others. In reality, these operations have different costs. One direction for new research is to assign realistic costs to each of these operations and analyze the *total cost* of all the algorithms.

Dividing the region to be explored into discrete grid cells facilitated the exploration of the region in a systematic fashion. However, as soon as an agent is aware of the food location, in the communication and transportation phases, the agent can simply take a straight-line path to its next location. Indeed, in these phases, the agent does not even really need the *Look* phase of the Look-Compute-Move cycles. It would be interesting to remove the restriction on directions of movement for agents; assume a fixed velocity for agents, and compute the time taken by different algorithms. Considering other shapes of search spaces, such as other polygons, would also be an interesting avenue for further study. For example, a hexagonal space could be divided into hexagonal cells, and might be amenable to interesting exploration strategies.

We gave some algorithms in which the paths followed by the agents are designed so that they meet after regular intervals. However, the *Sectors* algorithm was not designed in such a way. In the future, it can be investigated whether it is possible to adjust this algorithm, that is, make the agents wait at specific locations so that they do meet their neighbors regularly. Also, after the grid has been divided into sectors in this algorithm, new movement patterns to cover the grid can be investigated. For example, the cells at a distance of i from the nest get visited by the respective agent before the cells at a distance of $i + 1$.

Another possible area of research is in finding the running time of relay transportation when the agents are placed arbitrarily on the Nest-Food path initially. Finally, there is yet another interesting investigation that can be done using these algorithms. We are capturing the time at which the agents locate the food, the time taken by them to communicate this to their neighbors and the time taken to transport the food. However, the time taken for the last agent to reach the food location can also be calculated after it has completed its communication phase, which can be seen as the time taken to evacuate the grid. This time has been called the *evacuation time* and has received some interest recently.

Bibliography

[1] Y. Emek, T. Langner, J. Uitto, and R. Wattenhofer, "Ants: Mobile finite state machines," *CoRR*, vol. abs/1311.3062, 2013.

[2] Y. Emek, T. Langner, D. Stolz, J. Uitto, and R. Wattenhofer, "How Many Ants Does It Take To Find the Food?," in *21th International Colloquium on Structural Information and Communication Complexity (SIROCCO), Hida Takayama, Japan*, July 2014.

[3] A. Lpez-Ortiz and G. Sweet, "Parallel searching on a lattice," in *Proceedings of the thirteenth Canadian Conference on Computational Geometry*, 2001.

[4] J. Alonso-Mora, A. Breitenmoser, M. Rufli, R. Siegwart, and P. Beardsley, "Multi-robot system for artistic pattern formation," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 4512–4517, May 2011.

[5] M. Dorigo, "Swarm-bots - http://www.swarm-bots.org/," 2001.

[6] F. Mondada, A. Guignard, M. Bonani, D. Bar, M. Lauria, and D. Floreano, "Swarm-bot: From concept to implementation," in *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, vol. 2, pp. 1626–1631, IEEE, 2003.

[7] M. Dorigo, D. Floreano, L. Gambardella, F. Mondada, S. Nolfi, T. Baaboura, M. Birattari, M. Bonani, M. Brambilla, A. Brutschy, D. Burnier, A. Campo,

A. Christensen, A. Decugniere, G. Di Caro, F. Ducatelle, E. Ferrante, A. Forster, J. Martinez Gonzales, J. Guzzi, V. Longchamp, S. Magnenat, N. Mathews, M. Montes de Oca, R. O'Grady, C. Pinciroli, G. Pini, P. Retornaz, J. Roberts, V. Sperati, T. Stirling, A. Stranieri, T. Stutzle, V. Trianni, E. Tuci, A. Turgut, and F. Vaussard, "Swarmanoid: A novel concept for the study of heterogeneous robotic swarms," *Robotics Automation Magazine, IEEE*, vol. 20, pp. 60–71, Dec 2013.

[8] F. Mondada, M. Bonani, X. Raemy, J. Pugh, C. Cianci, A. Klaptocz, S. Magnenat, J. christophe Zufferey, D. Floreano, and A. Martinoli, "The e-puck, a robot designed for education in engineering," in *In Proceedings of the 9th Conference on Autonomous Robot Systems and Competitions*, pp. 59–65, 2009.

[9] M. Rubenstein, C. Ahler, and R. Nagpal, "Kilobot: A low cost scalable robot system for collective behaviors," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3293–3298, May 2012.

[10] R. E. Park and E. W. Burgess, *Introduction to the Science of Sociology.* Chicago, Illinois: The University of Chicago Press, 1921.

[11] M. Abrahams and P. Colgan, "Risk of predation, hydrodynamic efficiency and their influence on school structure," *Environmental Biology of Fishes*, vol. 13, no. 3, pp. 195–202, 1985.

[12] T. Pitcher, *Functions of Shoaling Behaviour in Teleosts.* Springer US, 1986.

[13] T. Pitcher and J. Parrish, "Functions of shoaling behaviour in teleosts," 1993.

[14] C. Feare, *The Starling.* Oxford: Oxford University Press, 1984.

[15] A. Colorni, M. Dorigo, V. Maniezzo, *et al.*, "Distributed optimization by ant colonies," in *Proceedings of the first European conference on artificial life*, vol. 142, pp. 134–142, Paris, France, 1991.

[16] G. Beni and J. Wang, "Swarm intelligence in cellular robotic systems," in *Robots and Biological Systems: Towards a New Bionics?*, pp. 703–712, Springer, 1993.

[17] *Oxforddictionaries.com.* Oxford Dictionary Press, 2015.

[18] D. Stephens, "Decision ecology: Foraging and the ecology of animal decision making," *Cognitive, Affective, and Behavioral Neuroscience*, vol. 8, no. 4, pp. 475–484, 2008.

[19] D. L. Kramer and W. Nowell, "Central place foraging in the eastern chipmunk, tamias striatus," *Animal Behaviour*, vol. 28, no. 3, pp. 772 – 778, 1980.

[20] J. Werfel, Y. Bar-Yam, D. Rus, and R. Nagpal, "Distributed construction by mobile robots with enhanced building blocks," in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pp. 2787–2794, May 2006.

[21] G. DAngelo, G. Stefano, and A. Navarra, "Gathering asynchronous and oblivious robots on basic graph topologies under the look-compute-move model," in *Search Theory* (S. Alpern, R. Fokkink, L. Gsieniec, R. Lindelauf, and V. Subrahmanian, eds.), pp. 197–222, Springer New York, 2013.

[22] E. Kranakis, D. Krizanc, and E. Markou, *The Mobile Agent Rendezvous Problem in the Ring.* Synthesis lectures on distributed computing theory, Morgan & Claypool Publishers, 2010.

[23] P. Flocchini, G. Prencipe, and N. Santoro, *Distributed Computing by Oblivious Mobile Robots.* Synthesis Lectures on Distributed Computing Theory, Morgan & Claypool Publishers, 2012.

[24] E. Talmage and M. Parter, "Disc 2014 review," *ACM SIGACT News*, vol. 45, no. 4, pp. 94–99, 2014.

[25] M. Dorigo, E. Bonabeau, and G. Theraulaz, "Ant algorithms and stigmergy," *Future Generation Computer Systems*, vol. 16, no. 8, pp. 851 – 871, 2000.

[26] M. Dorigo, G. Di Caro, and L. M. Gambardella, "Ant algorithms for discrete optimization," *Artif. Life*, vol. 5, pp. 137–172, Apr. 1999.

[27] R. Mayet, J. Roberz, T. Schmickl, and K. Crailsheim, "Antbots: A feasible visual emulation of pheromone trails for swarm robots," in *Swarm Intelligence* (M. Dorigo, M. Birattari, G. Di Caro, R. Doursat, A. Engelbrecht, D. Floreano, L. Gambardella, R. Gro, E. ahin, H. Sayama, and T. Sttzle, eds.), vol. 6234 of *Lecture Notes in Computer Science*, pp. 84–94, Springer Berlin Heidelberg, 2010.

[28] G. Rudolph, T. Jansen, S. Lucas, C. Poloni, and N. Beume, *Parallel Problem Solving from Nature - PPSN X: 10th International Conference Dortmund, Germany, September 13-17, 2008 Proceedings.* Lecture Notes in Computer Science / Theoretical Computer Science and General Issues, Springer Berlin Heidelberg, 2008.

[29] J. L. Almeter, "Experiments in Path Optimization via Pheromone Trails by Simulated Robots, Indiana University Computer Science Dept.," Sept. 1996.

[30] N. R. Hoff, *Multi-Robot Foraging for Swarms of Simple Robots.* PhD thesis, Harvard University,Cambridge, Massachusetts, May 2011.

[31] N. Hoff, A. Sagoff, R. J. Wood, and R. Nagpal, "Two foraging algorithms for robot swarms using only local communication," in *ROBIO*, pp. 123–130, IEEE, 2010.

[32] S. Chattunyakit, T. Kondo, I. Nilkhamhang, T. Phatrapornnant, and I. Kumazawa, "Two foraging algorithms for a limited number of swarm robots," in *Proceedings of SICE Annual Conference (SICE), 2013*, pp. 1056–1061, Sept 2013.

[33] K. Sugawara, T. Kazama, and T. Watanabe, "Foraging behavior of interacting robots with virtual pheromone," in *Proceedings. 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2004. (IROS 2004).*, vol. 3, pp. 3074–3079 vol.3, Sept 2004.

[34] Albers, Kursawe, and Schuierer, "Exploring unknown environments with obstacles," *Algorithmica*, vol. 32, no. 1, pp. 123–143, 2002.

[35] O. Feinerman, A. Korman, Z. Lotker, and J.-S. Sereni, "Collaborative search on the plane without communication," *CoRR*, vol. abs/1205.2170, 2012.

[36] R. A. Baeza-Yates, J. C. Culberson, and G. J. E. Rawlins, "Searching in the plane," *Inf. Comput.*, vol. 106, pp. 234–252, Oct. 1993.

[37] J. Czyzowicz, L. Gsieniec, T. Gorry, E. Kranakis, R. Martin, and D. Pajak, "Evacuating robots via unknown exit in a disk," in *Distributed Computing* (F. Kuhn, ed.), vol. 8784 of *Lecture Notes in Computer Science*, pp. 122–136, Springer Berlin Heidelberg, 2014.

[38] S. Fekete, C. Gray, and A. Krller, "Evacuation of rectilinear polygons," in *Combinatorial Optimization and Applications* (W. Wu and O. Daescu, eds.), vol. 6508 of *Lecture Notes in Computer Science*, pp. 21–30, Springer Berlin Heidelberg, 2010.

[39] E. Sahin, T. H. Labella, V. Trianni, J.-L. Deneubourg, P. Rasse, D. Floreano, L. M. Gambardella, F. Mondada, S. Nolfi, and M. Dorigo, “Swarm-bot: Pattern formation in a swarm of self-assembling mobile robots,” in *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, vol. 4, pp. 145–150, IEEE Press, 2002.

[40] L. Barriere, P. Flocchini, E. Mesa-Barrameda, and N. Santoro, “Uniform scattering of autonomous mobile robots in a grid,” in *IEEE International Symposium on Parallel Distributed Processing, 2009. IPDPS 2009*, pp. 1–8, May 2009.

[41] A. Marjovi, J. Nunes, L. Marques, and A. de Almeida, “Multi-robot fire searching in unknown environment,” in *Field and Service Robotics* (A. Howard, K. Iagnemma, and A. Kelly, eds.), vol. 62 of *Springer Tracts in Advanced Robotics*, pp. 341–351, Springer Berlin Heidelberg, 2010.

[42] J. Werfel, “Collective construction with robot swarms,” in *Morphogenetic Engineering* (R. Doursat, H. Sayama, and O. Michel, eds.), Understanding Complex Systems, pp. 115–140, Springer Berlin Heidelberg, 2012.

[43] N. Gordon, I. Wagner, and A. Bruckstein, “Discrete bee dance algorithm for pattern formation on a grid,” in *Intelligent Agent Technology, 2003. IAT 2003. IEEE/WIC International Conference on*, pp. 545–549, Oct 2003.

[44] D. Hilbert, “Ueber die stetige abbildung einer line auf ein flchenstck,” *Mathematische Annalen*, vol. 38, no. 3, pp. 459–460, 1891.

[45] M. Kesson, “Hilbert curve-concepts and implementation-http://www.fundza.com/algorithmic/space_filling/hilbert/basics/index.html,” 2002.