INSTITUTO TECNOLÓGICO DE AERONÁUTICA

Arthur José de Sousa Rodrigues

MULTI-AGENT GRAPH EXPLORATION WITHOUT COMMUNICATION

Bachelor's Thesis
2023

Computer Engineering

CDU 681.3

Arthur José de Sousa Rodrigues

MULTI-AGENT GRAPH EXPLORATION WITHOUT COMMUNICATION

Advisor

Prof. Dr. Luiz Gustavo Bizarro Mirisola (ITA)

Co-advisor

Prof. Dr. Vitor Venceslau Curtis (ITA)

COMPUTER ENGINEERING

São José dos Campos
Instituto Tecnológico de Aeronáutica

2023

Cataloging-in Publication Data
Documentation and Information Division

de Sousa Rodrigues, Arthur José
Multi-agent graph exploration without communication / Arthur José de Sousa Rodrigues. São José dos Campos, 2023.
47f.

Bachelor's Thesis – Course of Computer Engineering– Instituto Tecnológico de Aeronáutica, 2023. Advisor: Prof. Dr. Luiz Gustavo Bizarro Mirisola. Co-advisor: Prof. Dr. Vitor Venceslau Curtis.

1. Maze. 2. Graph. 3. Search. 4. Multi-agent. 5. Mixed radix. I. Instituto Tecnológico de Aeronáutica. II. Title.

BIBLIOGRAPHIC REFERENCE

DE SOUSA RODRIGUES, Arthur José. **Multi-agent graph exploration without communication**. 2023. 47f. Bachelor's Thesis – Instituto Tecnológico de Aeronáutica, São José dos Campos.

CESSION OF RIGHTS

Arthur José de Sousa Rodrigues
Rua H8B 226
12.228-461 – Sáo José dos Campos–SP

MULTI-AGENT GRAPH EXPLORATION WITHOUT COMMUNICATION

Bachelor's Thesis approved in its final version by the signatories below:

Arthur José de Sousa Rodrigues

Author

Luiz Gustavo Bizarro Mirisola (ITA)

Advisor

Vitor Venceslau Curtis (ITA)

Co-advisor

Prof. Dr. Marcos Ricardo Omena de Albuquerque Máximo
Course Coordinator of Computer Engineering

São José dos Campos: November 10, 2023.

To my mother Ana Paloma, daughter of Antônio and Vandnei. To my father Jairo José, son of José and Rosinete.

Acknowledgments

I thank my family for investing time and financial resources in my studies. And for giving me unconditionally love and motivation.

I thank my H8 friends for always supporting me with knowledge and positive words.

I thank my advisor Prof. Dr. Luiz Gustavo Bizarro Mirisola and my co-advisor Prof. Dr. Vitor Venceslau Curtis for all patience and attention that has been given to me.

I thank Brazilian Air Force for supporting my studies since I joined the Air Cadet Preparatory School (EPCAr).

"Now to him who is able to do immeasurably more than all we ask or imagine, according to his power that is at work within us, to him be glory in the church and in Christ Jesus throughout all generations, for ever and ever"

— Saint Paul

Resumo

Algoritmos computacionais de busca têm sido estudados por cientistas e companhias de engenharia desde o último século devido às aplicações de tais algoritmos em situações reais, como agendamento de linhas aéreas, planejamento de rota em mapas, algoritmos de busca na Internet, planejamento de rotas em redes de computadores, robótica, etc. Usualmente, esses métodos se baseiam em tipos abstratos de dados, como grafos e árvores, para transferir um problema real para dentro de um contexto delimitado e computacionalmente tratável. No campo da Ciência da Computação, grafos são tipos abstratos de dados que podem servir como suporte ferramental para algoritmos de busca.

Focando em algoritmos de busca em labirintos, este trabalho de graduação propõe um método de exploração de grafos por agentes previamente coordenados, mas incapazes de se comunicarem entre si. Tais agentes são previamente programados para se espalharem pelo grafo de maneira tão dispersa quanto for possível, de forma a evitar a repetição de caminhos e, por fim, minimizar o número de passos até que se encontre o nó-objetivo. Tal abordagem é motivada e inspirada por problemas da realidade, como exploração no fundo do mar, busca em estruturas compostas por muros maciços, busca em locais inóspitos por agentes com restrição energética, etc.

Este trabalho também explora como ferramenta a representação numérica em base mista, a qual, no presente ínterim, é capaz de representar os nós visitados e os caminhos correspondentes desde a raiz, bem como as suas posições relativas em um percurso *in-order* do labirinto visto como árvore. Isto permite não somente continuar um percurso através dos seus vizinhos imediatos, mas também elaborar estratégias para maximizar a dispersão dos agentes pelo labirinto.

Por fim, este relatório apresenta uma comparação de performance entre o algoritmo desenvolvido e o algoritmo de Tarry estendido, proposto por Kivelevitch & Cohen (2010), cujos métodos se ancoram na comunicação entre os agentes, diferentemente da proposta deste trabalho, em que os agentes cooperativamente exploram um grafo sem comunicação entre si.

Ressalta-se que, embora a proposta deste trabalho possa ser entendida para grafos de maneira geral, a presente pesquisa trata apenas de labirintos perfeitos, ou seja, árvores.

Abstract

Computational search algorithms have been studied by many scholars and engineering companies since the last century due to the real-life applications of such algorithms, such as in airline scheduling, planning path on maps, search engine algorithms, social media marketing, Internet routing protocols, robotics, etc. Usually, these methods rely on abstract data types, including graphs and trees, to represent a real world problem as a well defined and computationally tractable algorithm. In the context of Computer Science, graphs are abstract data types that might support computational search algorithms.

Focusing on maze-solving algorithms, this work proposes a method to explore a graph by previously coordinated agents that cannot communicate with each other. The agents are previously programmed to spread through the graph as dispersed as possible in order to avoid traversing the same portion of the graph and then minimizing the number of steps to find the goal node. This approach is related to real problems, such as sea exploration, search in large wall structures, search with low energy-based agents in inhospitable environments, etc. Thus, there are previous motivations to do this research.

This work also investigates a mixed radix numerical representation as a tool, which, in the context of this research, is able to represent the visited nodes and their corresponding paths from the root, as well their related positions in an inorder traverse through the maze structured as a tree. It allows not only to continue on the path through its nearby neighbors but also to elaborate strategies to maximize the dispersion of the agents in the maze.

Finally, this report presents a comparison between our algorithm's performance and the performance of the extended Tarry's algorithm, proposed by Kivelevitch & Cohen (2010), whose methods rely on communication between the agents, differently from this work's purpose, in which the agents cooperatively explore a graph with no communication.

It is worth emphasizing that, although the purpose of this work might be understood from the perspective of graphs in a general way, this research focuses on perfect mazes, i.e., trees.

List of Figures

List of Algorithms

List of Abbreviations and Acronyms

DFS	Depth First Search
BFS	Breath First Search
LCL	Last Common Location

Contents

1 Introduction

This thesis aims to discuss maze exploration by multi-agents programmed to find a goal cell. The main challenge of this research is to distribute the agents through the cells as dispersed as possible with total communication restriction, since an agent's partial knowledge cannot be shared with another agent, and, at the same time, a single agent must avoid repeating a branch of another agent. The only coordination between the agents occurs before they start their traversal: each one receives a different goal, represented as a different real interval between 0 and 1. The proposed maze structure abstraction is a traditional regular grid that might be generalized into graphs.

The current section intends to introduce the general concept of this research and present the motivation (Section 1.1), the related works (Section 1.2), and the main definitions about maze-solving algorithms (Section 1.3).

1.1 Motivation

Graph exploration has been the target of studies since Leonhard Euler proved that Seven Bridges of Königsberg (SHIELDS, 2012) has no solution. It has been researched not only in academia but also in the industry due to several practical applications, like airline scheduling, planning path on maps, search engine algorithms, social media marketing, Internet routing protocols, and robotics.

Specifically in robotics, graph exploration can be used to explore a maze with a single agent through a bunch of traditional algorithms: random mouse, wall follower, Trémaux, etc (SADIK et al., 2010). It can be useful to guide many real-life problems such as search in nuclear plant disasters, burning buildings, and extraterrestrial environments. In these previous examples, multi-agent exploration potentially can speed up the exploration, despite being ineffective if different agents explore the same portions of the graph. Recent studies have explored multi-agent maze-solving algorithms as seen in the Multi-Agent Maze Exploration paper (KIVELEVITCH; COHEN, 2010), where authors proposed a generalization of Tarry's algorithm. It is important to emphasize that maze-solving algorithms consider that the maze structure is unknown.

Traditional multi-agent maze exploration approaches are based on communication between agents, where each agent knows which cells were visited by other agents. This avoids a second exploration of the same path and thus decreases computational costs. But there are some real situations where communication is limited or impossible, such as deep sea exploration, search in large wall structures, or search with energy-limited autonomous agents.

However, the zero-communication approach between agents has not been concretely found in the literature although it may have real applications and may guide search plans in real-world problems. In order to explore it, this work presents some ways to achieve the solution of a perfect maze, i.e., a tree, based on agents without communication, which might be generalized to graph exploration.

1.2 Related works

Multi-agent cooperative system approaches are common in literature, especially in robotics. This chapter presents related works of multi-agent cooperative systems where the agents communicate among themselves, mainly in robotics.

Matarić (1995) established common properties across different scenarios of mobile multi-agent interactions, such as dispersion - "the ability of a group of agents to spread out in order to establish and maintain some minimum interagent distance" -, aggregation - "the ability of a group of agents to gather in order to establish and maintain some maximum interagent distance" -, homing - "the ability of an agent to find a particular region or location" -, etc. The author proposes a synthetic structure in order to abstract different types of interagent "basis behaviors".

Burgard et al. (2005) pointed out that an exploration group of robots takes several advantages over single-agent exploration, although a coordinating group might introduce redundancy. The authors present an algorithm to efficiently explore an environment by mobile and autonomous robots within a centralized communication range. These coordinated robots can completely cover the environment in a significantly reduced time compared to other related approaches cited in the article.

Sadik et al. (2010) presents, as seen in the title of the paper, a comprehensive and comparative study of maze-solving algorithms techniques by implementing graph theory. The research is delimited in the "Micromouse competition" context, which is a famous maze competition that has been performed worldwide since the late 1970s. The authors compared maze-solving methods based on graph theory algorithms, such as DFS (Depth First Search) and BFS (Breath First Search) flood-fill, to common algorithms in the "Micromouse" context, such as Wall Follower. They concluded that, despite the fact that graph

algorithms demand higher computational complexity, they are more proficient compared to related algorithms that don't use graph representation, and only make decisions relying on the neighborhood.

Based on maze-solving algorithms, Kivelevitch & Cohen (2010) proposed a generalization of Tarry's algorithm, but one important characteristic of this approach is that all visited cells of the maze are known by each agent, since each agent shares its knowledge with all the others. In that sense, each one holds a dynamic map of the maze, allowing information sharing. The authors present in the article the performance of the proposed solution, where a group of virtual coordinating agents is required to find the goal without an *a priori* knowledge of the maze, so-called "maze exploration".

Beisel (2014), similarly to the aforementioned, worked in simulation and mathematical analysis for strategies related to cooperative autonomous robots, that can share messages with each other to exit a maze. The author concluded that a cooperative approach might result in significant performance improvements compared to uncooperative and uncoordinated robots.

1.3 Definitions

Mainly considering these previous references, this chapter intends to define common terms about maze-solving algorithms, graph theory, and mathematical tools that were useful to abstract the proposed solution. Furthermore, it intends to clarify the domain of this work.

1.3.1 Graph

Graph theory has been used in various real-life applications, such as in biology, social sciences, engineering, computer science, etc. Manber (1989) establishes that a graph $G = (V, E)$ consists of a set V of vertices (also called nodes), and a set E of edges. Each edge corresponds to a pair of vertices and represents relationships among the vertices. A graph can be directed or undirected. The edges in a directed graph are ordered pairs, i.e., the order of the edge connection between two vertices is important. On the other hand, the edges in an undirected graph are unordered pairs, i.e., the order of the edge connection between two vertices is not important. Manber (1989) gives an example: a graph may represent a set of people, and the edges may connect any two persons who know each other. Moreover, he discusses several computational problems in terms of graphs, where he describes famous graph exploration algorithms, such as DFS (Depth First Search) and BFS (Breath First Search).

Manber (1989) works with some representations of graphs in his book. One of the representations is the adjacency list of a graph, that is described by an array of lists. In the adjacency list representation, each vertex is associated with a linked list consisting of all the edges adjacent to this vertex. Supposing that $|V| = n$, the adjacency list will have n linked lists. In the case of this work, graphs are always undirected, therefore, if the linked list i of a vertex v_i has a vertex v_j, the linked list j surely has the vertex v_i.

It is worth mentioning that trees are a subset of graphs. Manber (1989) indicates that, in a tree, it is possible to hierarchize all the edges from the root (the head node in terms of tree hierarchy), and hence trees are sometimes called rooted trees.

Section 2.2 exposes the perspective of a maze from tree and graph topologies.

1.3.2 Maze

This report defines a maze from the same perspective presented in Kivelevitch & Cohen (2010). A maze is a n-dimensional gridded space of any size, usually rectangular. The gridded space is composed of a set of cells, while a cell is the elementary item of a maze, defined as a delimited n-dimensional space. Cells might be connected or not connected to another adjacent cell, separated by a "wall" in the latter case. Without losing generality, as presented in Section 2.1, this work considered a maze, for simulation purposes, as a two-dimensional gridded space composed of two-dimensional bounded cells.

Thus, from the above definition, a 4-neighbor 2D grid graph, where a wall is represented by eliminating the edge between 2 neighboring cells, is a possible computational representation of a maze, which is a tree if the maze has no loops, or equivalently, if there is only one path between any two cells, including the start and/or goal positions.

1.3.3 Agent

As defined in Kivelevitch & Cohen (2010), an agent is an autonomous entity that can traverse the maze obeying the connection of the cells.

In a maze context, a multi-agent approach describes the coordinating behavior of several autonomous agents.

1.3.4 Maze-solving algorithms

A maze has a goal to be achieved, which is usually a single marked cell or a path to an agent in order to exit the maze. Since the last century, many scholars have studied algorithms to solve the maze computationally, and the proposed solutions are so-called

"maze-solving algorithms".

1.3.5 Mixed Radix

Mixed radix is a numerical representation that generalizes standard positional numerical systems to allow a different base for each digit. In a mixed radix representation, a number is represented by a sequence of numerals in different bases, where each numeral is a multiple of the previous numerical sequence, however, relied on a different base.

For example, Arndt (2011) establishes a arithmetical method to manipulate a specific subset of mixed radix numbers. The mixed radix representation $A = [a_0, a_1, a_2, ..., a_{n-1}]$ of a number x with respect to a radix vector $M = [m_0, m_1, m_2, ..., m_{n-1}]$ is given by:

$$x = \sum_{k=0}^{n-1} a_k \prod_{j=0}^{k-1} m_j \tag{1.1}$$

where $0 \leq a_j < m_j$ (and $0 \leq x < \prod_{j=0}^{n-1} m_j$, so that n digits suffice). In a traditional positional numerical system, the vector M is like $M = [r, r, r, ..., r]$, and then the relation is simply given by:

$$x = \sum_{k=0}^{n-1} a_k r^k \tag{1.2}$$

For instance, if x is represented by $1_2 3_9 5_6$, and considering that a_i is on the right of a_{i+1}, x might be transformed to the decimal base by the following steps:

$$\begin{aligned} x &= 1_2 3_9 5_6 \\ A &= [5, 3, 1] \\ M &= [6, 9, 2] \\ n &= 3 \\ x &= \sum_{k=0}^{2} a_k \prod_{j=0}^{k-1} m_j = 77 \end{aligned} \tag{1.3}$$

Despite the aforementioned perspective, this work presents a slightly different approach, since it uses a mixed radix representation of numbers contained in the real range $[0, 1]$. In this context, the mixed radix representation $A = [a_0, a_1, a_2, ..., a_{n-1}]$ of a number x with respect to a radix vector $M = [m_0, m_1, m_2, ..., m_{n-1}]$, where $x \in [0, 1]$, is given by:

$$x = \sum_{k=0}^{n-1} a_k \prod_{j=0}^{k} \frac{1}{m_j} \tag{1.4}$$

where $0 \leq a_j < m_j$.

For instance, if x is represented by $0.5_8 1_2 8_{10} 0_3 3_9$, and considering that a_i is on the left of a_{i+1}, x might be transformed to the decimal base by the following steps:

$$\begin{aligned} x &= 0.5_8 1_2 8_{10} 0_3 3_9 \\ A &= [5, 1, 8, 0, 3] \\ M &= [8, 2, 10, 3, 9] \\ n &= 5 \\ x &= \sum_{k=0}^{4} a_k \prod_{j=0}^{k} \frac{1}{m_j} \approx 0.7381944 \end{aligned} \tag{1.5}$$

This thesis also presents a pseudocode in Section 2.4 for calculating the value of a mixed radix number inside the real range $[0, 1]$.

2 Models

2.1 Maze and multi-agent exploration

For simulation purposes, this work used an open-source tool that models a maze under a 4-neighbor 2D grid graph perspective, where each cell is a square with its edges composed by a wall or not. If there is a wall, an agent cannot traverse the maze through the corresponding edge, in any direction. On the other hand, if there is not a wall, an agent has a free way to traverse the maze through the related edge in both directions.

The maze has a goal that is a single marked cell, and an agent inside the maze aims to find the marked cell, traversing the maze cell by cell. This agent is an autonomous entity that follows a specific algorithm and previously programmed rules. To avoid revisiting cells, it stores the visited cells. Thus, when there are various possible branches to extend the agent's path, it ignores cells already visited by itself. If there is no candidate to be a possible branch, the agent backtracks through already visited cells until it finds an unvisited branch. Furthermore, if more than one agent traverse the maze in order to find the goal, differently from some approaches presented in Beisel (2014), Burgard et al. (2005), and Kivelevitch & Cohen (2010), there is no intercommunication between agents in the approach of this research, i.e., an agent knows nothing about another agent's search.

Naeem (2021) developed an open-source maze generator. It is a python module that creates random mazes and enables the user to simulate its own maze-solving algorithm. In this context, this work presents a multi-agent maze-solving algorithm that has been simulated over Naeem (2021) open-source code, with modifications. Figure 2.1 presents, for example, 2 agents traversing a 6×6 maze toward the goal.

Naeem (2021) software creates by default a "perfect maze", which means that there is one and only one path to the goal from any cell. However, it is possible to set the code to generate a imperfect maze.

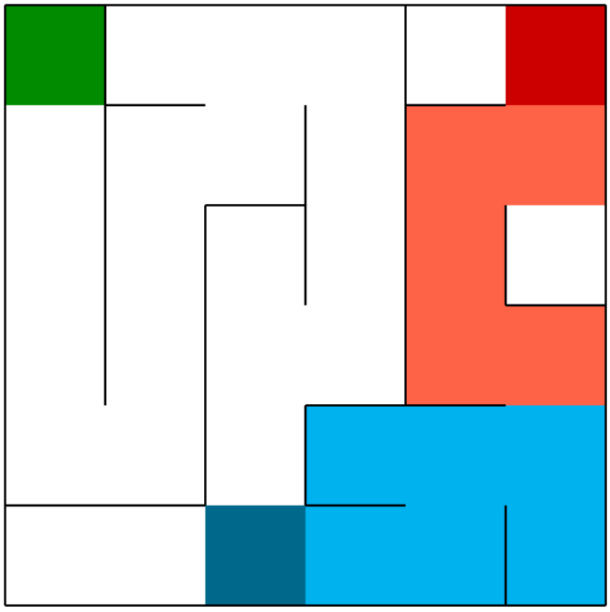


FIGURE 2.1 – Blue and red agents traversing a 6×6 maze. The goal is represented by the green entity. This maze is a perfect maze (NAEEM, 2021).

2.2 Maze from a graph topology perspective

As pointed out in Section 2.1, given that an agent ignores visited cells, there are important statements related to this work:

- if there is only one path to the goal from any cell, it is valid to consider a maze as a tree, i.e., a perfect maze (NAEEM, 2021);
- if there is more than one path to the goal from any cell, the maze cannot be considered as a tree, but it might be considered as a general 2D grid;
- regarding the last statement, even though not every maze can be considered as a tree, an agent path might be individually interpreted as a tree if such agent does not go through already visited cells, except in backtracking situations.

Figure 2.4 gives an example of the graph representation of the maze presented in Figure 2.3, where a red agent is traversing the maze toward the green goal. As established by Naeem (2021), the maze cells are programmatically addressed as indices of a matrix, such as represented in Figure 2.2.

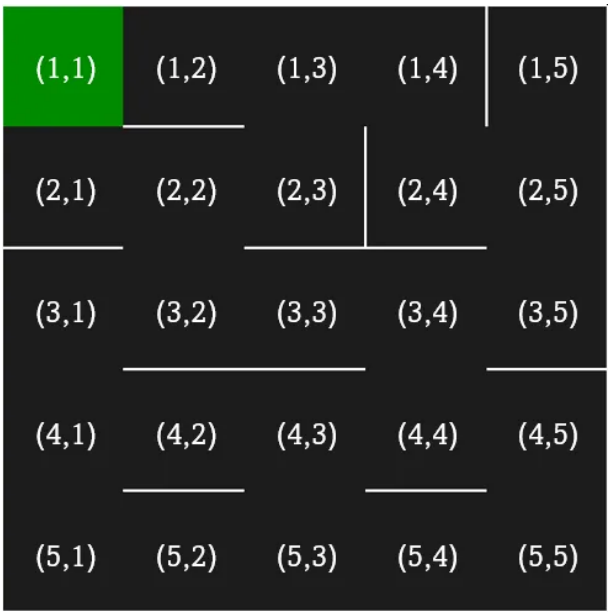
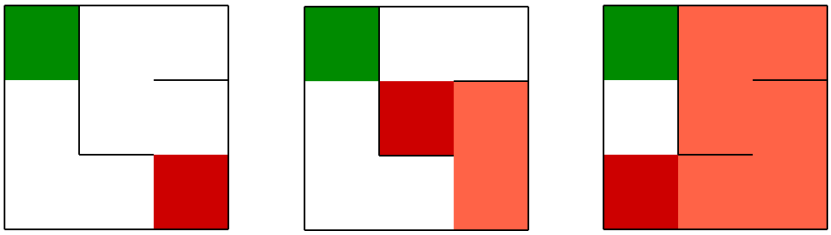


FIGURE 2.2 – Maze cell indices. Source: Naeem (2021).



(a) Agent at starting cell (b) Agent's 2nd step (c) Agent's 10th step

FIGURE 2.3 – Red agent traversing a maze toward the green goal. This maze is a perfect maze (NAEEM, 2021).

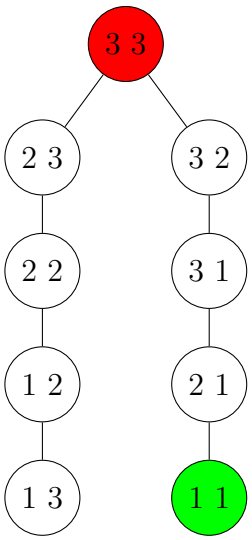


FIGURE 2.4 – Graph representation of the maze presented in Figure 2.3. The cell indices are pointed out. The agent starts at index (3,3), and achieves the goal at index (1,1).

2.3 Multi-agent exploration without communication

2.3.1 Our algorithm's key concepts

The goal of this work is to present a maze-solving algorithm in a multi-agent environment, where an agent cannot communicate with the other agents. Thus, each agent must be previously programmed to avoid exploring the same portion of the maze as other agents. Indeed, they should be as dispersed as possible.

First of all, this research considers a maze as a graph, where each node is a cell representing the maze. Given an initial node from which the agent starts its path, the agent checks if the node has children. Since the graph is undirected, i.e., there are no completely walled cells, this work establishes some statements:

- if an agent finds the goal cell, the agent finishes its path;
- if an agent is not in the goal cell and the cell has only one unwalled edge, the agent necessarily goes through such edge;
- if an agent is not in the cell where the goal is and the cell has more than one unwalled edge, the agent needs to decide which edge it will go through.

To establish a decision algorithm to the last statement, this work proposes firstly an interval division related to the agents and the nodes around them. Supposing that there are k agents $a_1, a_2, ..., a_k$ to explore the maze, each agent will have a corresponding and proportional range of action related to the graph, as presented below:

$$\begin{aligned} d &= 1/k \\ a_1\text{: } &[0, d[\\ a_2\text{: } &[d, 2d[\\ a_3\text{: } &[3d, 4d[\\ &... \\ a_k\text{: } &[(k-1)d, 1] \end{aligned} \tag{2.1}$$

where d is the size of the partition corresponding to each agent. Furthermore, every node will also have an interval in order to converge to itself the agents whose intervals intersect the node's convergence interval.

To disperse the agents through the maze, this work establishes some steps in order to guide the agent, that knows nothing about another agent's search. As a starting point for graph exploration, our algorithm works specifically in the context of perfect

mazes (NAEEM, 2021). The following steps base the reasoning expressed in the proposed pseudocode in Section 2.3.2:

- every agent a_i has a previously programmed interval;
- every agent starts its search at the same cell, i.e., the root of an agent's tree is the same for any agent. The root's converge interval is $[0, 1]$;
- every agent explores the maze calculating dynamically the convergence intervals of the children of the current node where the agent is. These convergence intervals are calculated based on the following steps:
  - if there is only one child, the child's convergence interval is the same as the current node's convergence interval;
  - if there is more than one child, the current node's convergence interval is uniformly split between its children;
- every agent must go to the first child whose convergence interval intersects the agent's interval. The order of children must be well defined so that every agent gets the same calculation result of some node's convergence interval. For perfect mazes, this ensures that every cell (node) has the same convergence interval regardless of the agent. This research established a clockwise order for the cells - North, East, South, West - from left to right regarding the insertion of children, which helps visualize the dispersion as each agent tends to follow a different direction. But there is no need for the children to have any specific ordering - the ordering just needs to be consistent among all agents;
- an agent doesn't visit an already visited node, except in backtracking situations where there are no children to visit, i.e., the node in fact has no children, the children have already been visited, or there are no remaining unvisited children that match their convergence intervals to the agent's interval;
- every agent must fill its interval going through the corresponding nodes in in-order until finds the goal. If it doesn't find the goal, but it fills its interval, surely the convergence interval of the goal node is outside of the agent's interval. In this context, the agent stops the search or changes its decision algorithm. In the case that it changes its behavior, this work proposes two optional secondary conducts:
  - the agent modifies its target interval and the exploration order from left-right to right-left. Section 3 discusses broader about this point;
  - the agent explores the maze using some DFS algorithm, but ignores already visited nodes.

The following figures expose key concepts of our algorithms:

- Figure 2.5 is related to the dispersion concept, in which every agent explores the maze as dispersed as possible from another agent;
- Figure 2.6 presents the convergence intervals of the cells (nodes) of a fictional maze when a given agent explores it based on the aforementioned steps;
- Figure 2.7 exposes an agent with an interval $[0, \frac{1}{2}[$ prioritizing to fill completely its interval, despite there being a bigger subtree on the right of the root. In fact, our algorithm is an exploration algorithm, so an agent has no information about the structure of the maze. It is worth to emphasizing that when an agent finishes its interval, it needs to change its behavior or simply stop the search;
- Figure 2.8 exposes the visit order of an agent with interval $[\frac{1}{3}, \frac{2}{3}[$. Due to the fact that the agent finishes its interval in the 7th visited node, and it doesn't go through already visited nodes, except in backtracking situations, it never finds the green goal, regardless of the behavior that it assumes after finishing its interval. Therefore, an agent might be unsuccessful in the search because it doesn't repeat a crossing - from a node to another node in the same direction - more than one time.

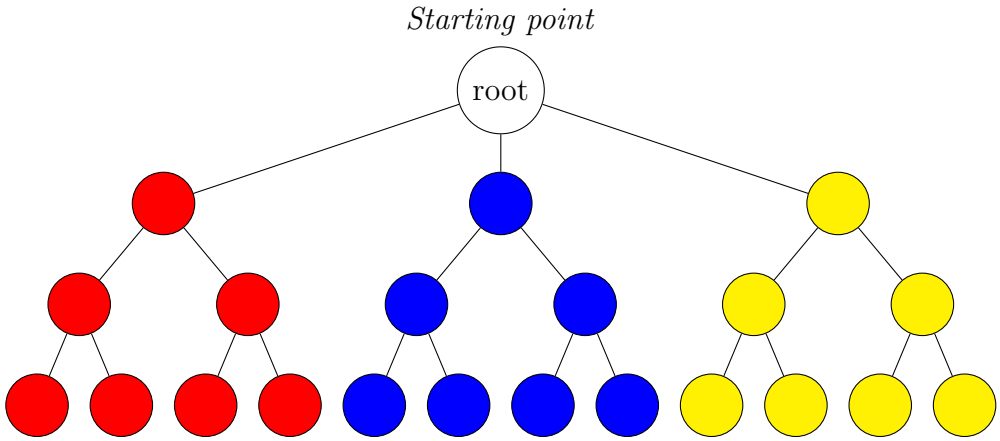


FIGURE 2.5 – Three agents (Red, Blue, and Yellow) disperse from each other in a fictional and perfect maze.

2.3.2 Pseudocode for our algorithm

As pointed out in Section 1.3.1, a graph may be represented as an adjacency list. Given that an agent path through an unknown maze may be represented as a tree in the situation approached in Section 2.2, where an agent does not go through already visited cells, except in backtracking situations, and considering the tree as an adjacency list A,

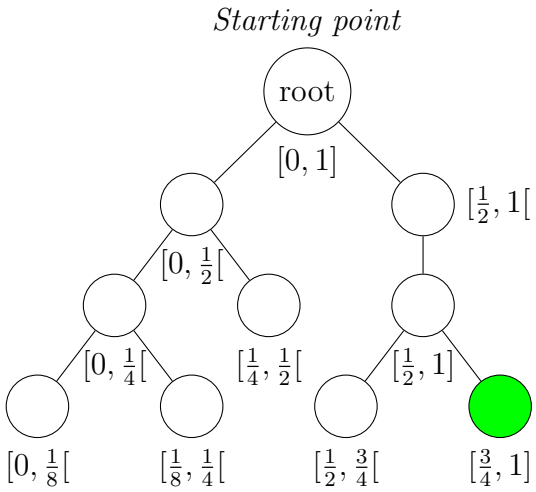


FIGURE 2.6 – A fictional maze represented as a tree where each node has an established interval according to an agent's traverse guided by Pseudocode 1. The green node represents the goal.

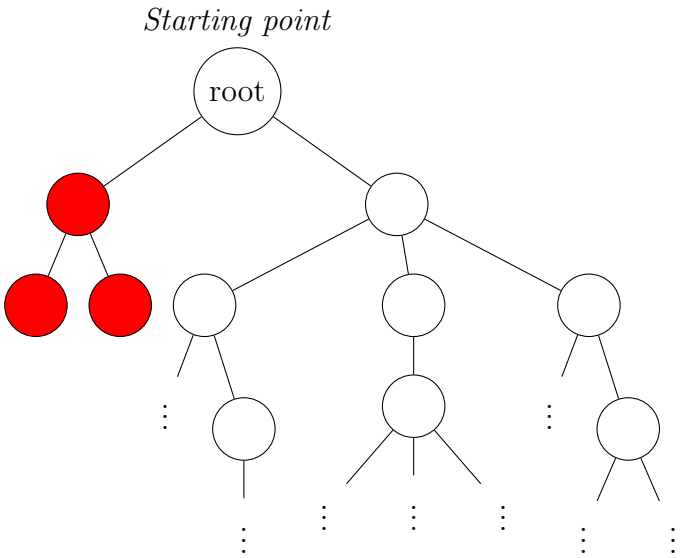


FIGURE 2.7 – A red agent with an interval $[0, \frac{1}{2}[$ prioritizing to fill completely its interval in a fictional maze.

where the first explored node (root) v_1 has a convergence interval equal to $[0, 1]$, while the other nodes v_i of the maze are unknown, i.e., the agent doesn't know previously their convergence intervals, Pseudocode 1 establishes the convergence intervals of each node specifically in the context of perfect mazes (NAEEM, 2021).

As mentioned previously, there is no communication between agents. Therefore, Pseudocode 1 considers that each agent needs to explore the maze individually and calculate by itself the convergence intervals of each node. Furthermore, it is important to emphasize that each agent considers individually the maze as tree, where the root is the agent's starting node.

Algorithm 1 Traverse of the agent through the maze (interpreted as a tree by the agent).

```
/* Previously defined interval of the agent */
agent_interval ← [a_min, a_max[

/* Agent starts the exploration at node v_1 */
current_node ← v_1

/* Array of visited nodes by the agent is initially empty */
visited_nodes ← [ ]

/* v_1 is the root, so its interval comprises the entire range of visits */
intervals[v_1] ← [0, 1]

/* Initialize the empty adjacency list of the tree A */
A ← ⊘

/* In the tree A, create a node for v_1, and fill its children list with the neighboring cells
of v_1. The children insertion order must follow some immutable rule. In the case of
this work, the insertion is clockwise (North, East, South, West) */
complete(A, v_1)

/* Create a flag to know if the agent filled its interval passing by all the corresponding
nodes in the tree */
finished_interval ← FALSE


while current_node is not the goal cell do
    if current_node has no child or all of current_node's children were visited then
        if current_node == v_1 then                          ▷ Unsuccessful search
            break
        else
            current_node ← get_parent(A, current_node)
            continue
        end if
    end if
    if current_node is not in visited_nodes then
        append(visited_nodes, current_node)
        max ← get_max_value(intervals[current_node])
        min ← get_min_value(intervals[current_node])        ▷ Node's mixed radix value
        partition ← (max - min) /amount_of_children
        count ← 0
        for all child v_j of current_node in A do
            intervals[v_j] ← [min + count · partition, min + (count+1) · partition[
            complete(A, v_j)
            count ← count + 1
        end for
    end if
```

Algorithm 1 Traverse of the agent through the maze (interpreted as a tree by the agent).

```
    for all child v_j of current_node in A do
        max_node ← get_max_value(interval[v_j])
        min_node ← get_min_value(interval[v_j])
        max_agent ← get_max_value(agent_interval)
        min_agent ← get_min_value(agent_interval)
        if v_j is in visited_nodes then
            continue
        else if max_agent < min_node then
            /* If the node's interval is on the right of the agent's interval, surely the
            agent has finished its interval. Since the order of the nodes are well defined,
            and in increasing order of intervals, the following condition improve the
            algorithm's performance. It is optional */
            finished_interval ← TRUE
            break
        else if min_agent < max_node and max_agent > min_node then
            /* This condition verifies if there is intersection between the agent's interval
            and the node's interval. The order of children must be well defined */
            current_node ← v_j
            break
        end if
    end for

    /* If current_node remains the same as it was at the beginning of the iteration,
    and it is the root, the agent has finished its interval */
    if current_node == v_1 and finished_interval == FALSE then
        finished_interval ← TRUE
    end if

    /* If the agent completely filled its interval, it needs to change its behavior, since its
    behavior is domain dependent. It might continue the DFS under some criteria.
    Otherwise, it must come back to parent in order to verify if there are more nodes to
    visit */
    if finished_interval == TRUE then
        current_node ← continue_DFS_under_some_criteria(A, current_node)
    else
        current_node ← get_parent(A, current_node)
    end if
end while
```

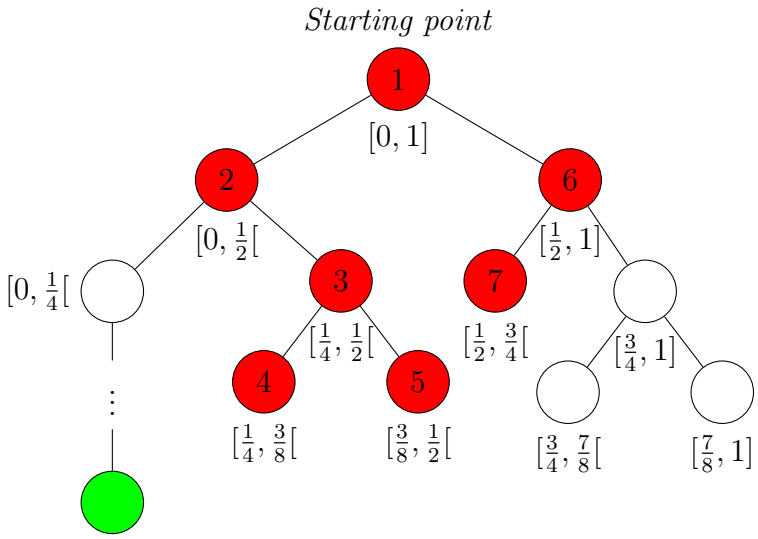


FIGURE 2.8 – An agent with interval $[\frac{1}{3}, \frac{2}{3}[$ finishes its interval in the visit order indicated. Regardless of the behavior that it assumes after finishing its interval, it never finds the green goal since it doesn't go through already visited nodes, except in backtracking situations. It is a fictional maze.

Thus, the goal of this work aims to establish an efficient algorithm that interrelates agent intervals to node convergence intervals. Furthermore, the authors observed that a mixed radix representation to the agents' path is a powerful mathematical tool for relating such path to the node's convergence interval that the agent is visiting. Section 2.4 discusses the mixed radix representation.

2.4 Mixed radix representation to the agent's path

In Section 2.3, this work presented a method for an agent to traverse the maze by defining ranges of action. Pseudocode 1 exposes this method and establishes a way to explore the maze while an agent records the visited nodes and the explored tree. However, the agent doesn't need in fact to save all the structure of the tree, because it only needs to know the current node's convergence interval and the current node's edges to make a decision.

This work establishes a method to calculate the node's convergence intervals without the agent having to save necessarily all the structure of the tree. In order to traverse the maze while the agent autonomously makes a decision node by node, it saves its path in a mixed radix numerical representation, where the starting representation is "0.", and it is limited to 1, i.e., $[0, 1]$. Pseudocode 2 proposes a computational method to get the value of mixed radix numerical representation. Equation 2.2 shows examples of numbers under a mixed radix representation.

$$
\begin{aligned}
&0.0_2 1_2 = 0.25 \\
&0.0_2 2_3 = 0.333... \\
&0.0_2 2_3 1_2 = 0.41666... \\
&0.1_5 1_3 0_2 1_2 = 0.28333... \\
&0.8_9 2_7 2_3 2_3 5_8 \approx 0.93584656
\end{aligned}
\tag{2.2}
$$

According to the order of analysis of the children and the mathematical tools to calculate node convergence intervals that are presented in Pseudocode 1, the items below present how the agent saves its path based on a mixed radix representation. These points guide the reasoning in Pseudocode 2.

- If the current node has only one child and the agent goes through this child, it appends I_1 to the mixed radix numerical representation of the path;
- If the current node has n children, where $n > 1$, and the agent decides to go through the ith child, it appends $(i-1)_n$ at the end of the mixed radix numerical representation of the path;
- If the agent must go back to the parent, it removes the last appended value from the mixed radix numerical representation of the path.

It is important to note that the value of the mixed radix numerical representation is the same as the minimum value of the convergence interval of the node that the agent is visiting. Moreover, the mixed radix numerical representation carries information about the maximum value of the node's convergence interval, since it is the sum of the minimum value with the lowest possible value of the current mixed radix representation. For instance, if the agent's current path is $0.1_3 0_2 2_4$, the minimum value of the current node's convergence interval is $0.1_3 0_2 2_4$, and the maximum is $0.1_3 0_2 3_4$ ($0.1_3 0_2 2_4 + 0.0_3 0_2 1_4$). Figure 2.9 presents an example of the approach of a mixed radix numerical representation, where the agent's path carries information about the current node's convergence interval.

2.5 Extended Tarry's algorithm

Rao et al. (1999) explain that Tarry's algorithm is a graph search algorithm where a single agent passes through each edge of a graph in order to visit all vertices, going through each edge once and only once in each direction. As Rao et al. (1999) state, it is similar to the common depth-first search algorithm used in graphs. Thus, if a maze is structured as a graph, Tarry's algorithm might be a tool to solve such maze.

Algorithm 2 Methods related to mixed radix numerical representation.

```
/* Structure of mixed radix element */
struct mr_element
{
  type numeral
  type radix
}

/* Initialize the agent's path. The agent is at the tree's root. Numerically it means zero
in the mixed radix representation */
path ← [ ]

procedure APPEND(path, numeral, radix):
    struct mr_element element
    element.numeral ← numeral
    element.radix ← radix
    add(path, element)              ▷ Append the element to the end of the array path
end procedure

procedure REMOVE(path):
    subtract(path)                  ▷ Remove the last element of the array path
end procedure

procedure GET_MIXED_RADIX_VALUE(path):
    value ← 0
    interval_size ← 1               ▷ In order to calculate the interval size step by step
    for all element in path do
        if element.numeral == I then        ▷ "I" represents a node with only one child
            continue
        end if
        value ← value + interval_size · (element.numeral / element.radix)
        interval_size ← interval_size / element.radix
    end for
    return value
end procedure
```

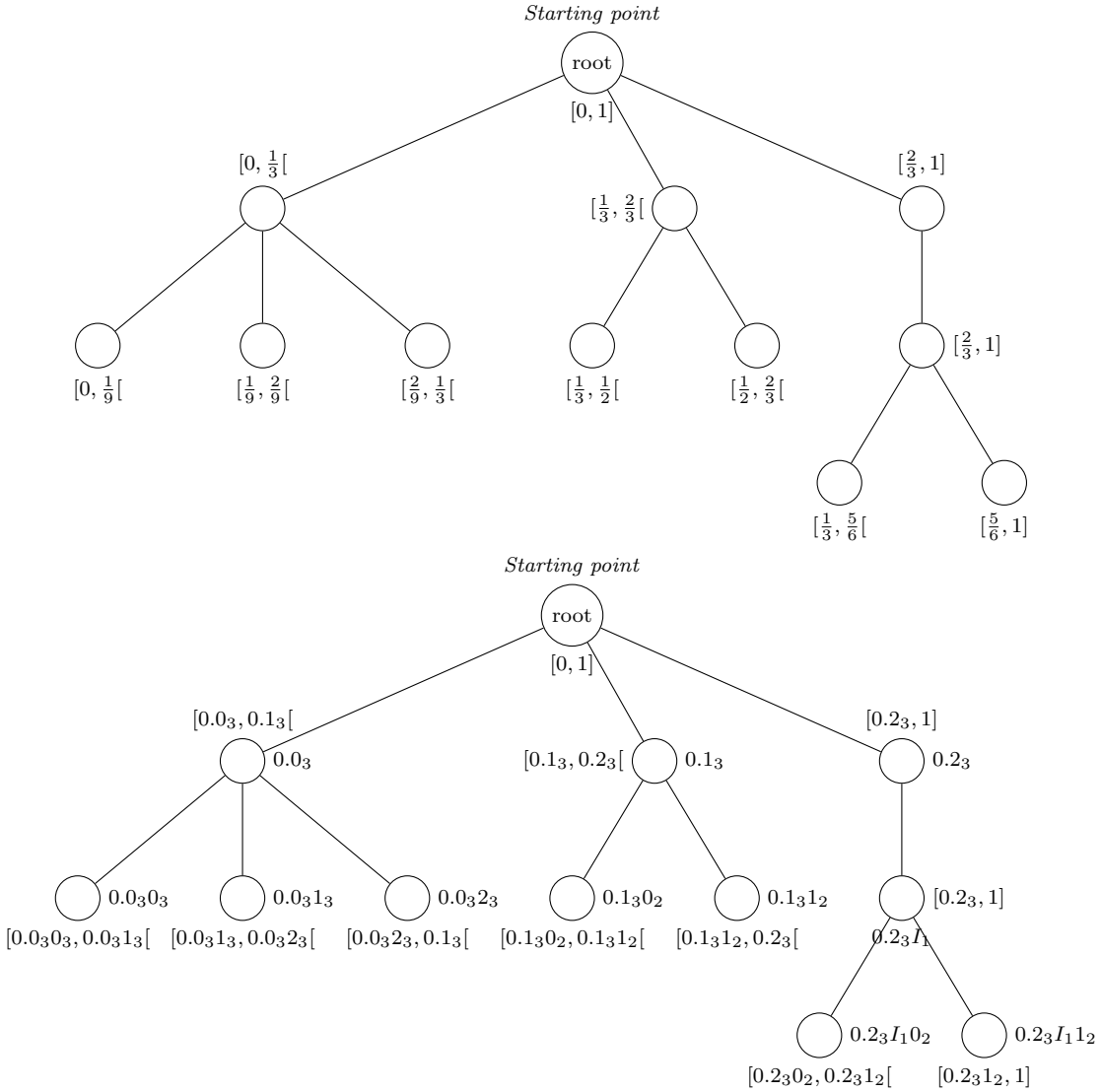


FIGURE 2.9 – Both trees are the same, but the top one has its node convergence intervals presented in a fractional representation, and the bottom one has its node convergence intervals presented in a mixed radix representation. As the agent's path carries information about the nodes' convergence intervals when the path is structured in a mixed radix representation, it is possible to save, with only one numerical representation, information about the tree explored by the agent and the convergence intervals of the tree's nodes. It is a fictional maze.

Kivelevitch & Cohen (2010) propose an extended version of Tarry's algorithm to solve a maze where a set of agents tries to find the goal cooperatively. Such agents have no *a priori* knowledge about the maze's structure, but they can share information dynamically about the cells explored by each agent. Therefore, the authors present a multi-agent maze-solving algorithm with communication, whose performance was compared to our algorithm's performance, even though the agents cannot share information in the scope of this work.

Basically, to solve a maze in a multi-agent context, Kivelevitch & Cohen (2010) pro-

gram each agent to follow a generalization of Tarry's algorithm. However, all the visited nodes are shared with each agent since the agents communicate with each other. Thus, the following steps guide the reasoning of the proposed algorithm (KIVELEVITCH; COHEN, 2010):

- The agent should move to cells that have not been traveled by any agent.
- If there are several such cells, the agent should choose one arbitrarily.
- If there is no cell that has not been traveled by an agent, the agent should prefer to move to a cell that has not been traveled by itself.
- If all the possible directions have already been traveled by the agent, or if the agent has reached a dead-end, the agent should retreat until a cell that meets one of the previous conditions.
- All the steps should be logged by the agent in its history.
- When retreating, mark the cells retreated from as "dead end".

Furthermore, the algorithm has a second phase: given the path of the pioneer agent that first finds the goal, the other agents must backtrack to the last location that matches a location on the pioneer's path in order to achieve the goal as the pioneer did. So, when the pioneer finds the goal, no more cells are explored, since the other agents come back to the "Last Common Location (LCL)" relating to the pioneer and finally follow the pioneer's path until the goal.

This work used this maze-solving method for comparison to our algorithm, as exposed in Section 3.3. To know more about the aforementioned generalization of Tarry's algorithm, please see Kivelevitch & Cohen (2010).

3 Results and Discussion

The authors developed a code in Python based on the methods exposed in Section 2 for assessing our algorithm's performance. Pseudocode 1 and Pseudocode 2 guided the reasoning behind the code, which is available on github.com/ArthurJose2000/mazeexploration.

This work used perfect mazes for simulating multi-agent scenarios where agents go through the mazes in order to find the goal. Section 3.1 presents the agents's performance when they use our algorithm. Section 3.2 proposes an agent policy modification to improve the agent's performance when it finishes its interval. Finally, Section 3.3 compares our algorithm's performance to the performance of the extended Tarry's algorithm (KIVELEVITCH; COHEN, 2010), despite the latter having communication between agents.

3.1 Our algorithm's perfomance

To assess our algorithm's performance, this work generated, using the code provided by Naeem (2021), 250 random and perfect mazes for each size - 10×10, 20×20, 30×30, and 40×40. As they are perfect mazes, i.e., there is only one path to the goal from any cell, any node of the maze's tree representation has the same convergence interval for every agent. Thus, the developed code demonstrates the dispersion concept, where the agents try to go through the maze as dispersed as possible without communication.

Over each generated maze, simulations were run varying the number of agents from 1 to 40. Thus, in total, $250 \times 40 \times 4$ simulations were executed. In order to analyze the results, the authors computed the following classes of analysis:

- Average of steps: the average number of steps of one agent in each maze. In other words, the average distance for one agent.
- Pioneer's average of steps: the average of steps of the first agent that reaches the goal, i.e., the pioneer.
- STDEV - Average of steps: the standard deviation of the average of steps of one agent considering all the 250 different mazes.

- Fraction of maze explored: the percentage of visited cells until every agent stops.
- Fraction of maze explored when pioneer reaches target: the percentage of visited cells at the moment when the pioneer reaches the goal.

As pointed out in Section 2.3.1, where the key concepts of this research are exposed, it is worth emphasizing that some agents don't find the goal, so it is possible that some agents stop the search even before the pioneer finds the goal. It is clearer for big mazes, as seen in Figure 3.1d, in which, in some situations, the average of steps is smaller than the pioneer's average of steps, which might seem incoherent at first look. If the approach of Pseudocode 1 is modified so that each agent must find the goal anyway, obviously the average of steps must be bigger than the pioneer's average of steps.

Figure 3.1 presents, for each maze size - 10×10, 20×20, 30×30, and 40×40 -, the results of the average of steps, the pioneer's average of steps, and the standard deviation of the average of steps considering all the 250 different mazes for each size. Similarly, Figure 3.2 presents the results of the fraction of explored maze and the fraction of maze explored when the pioneer reaches the target.

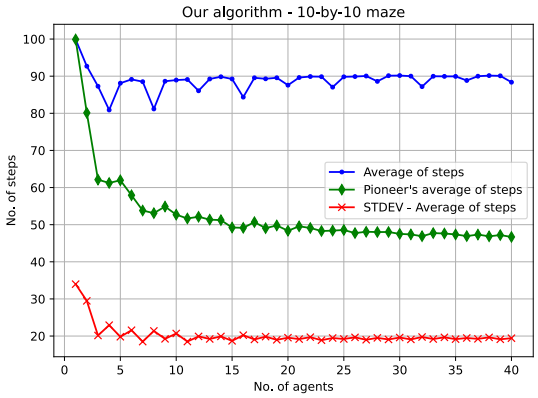
3.2 Incremental policy for the agents

After establishing an algorithm where the agents focus on filling its interval, this work implemented a slight modification, where the agent changes its interval and its order of action after finishing its starting interval.

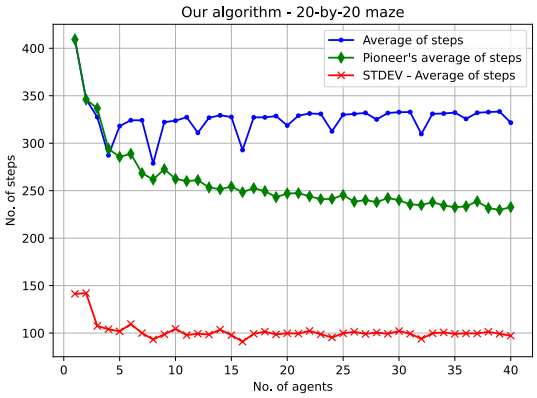
As pointed out in Pseudocode 1, the order of action of each agent is clockwise in the context of mazes - North, East, South, and West. Supposing that there are k agents $a_1, a_2, ..., a_k$ to explore the maze, there will be k different acting intervals, as seen in Equation 2.1. With our incremental policy, when the agent a_i finishes its interval, its interval gets the value of the starting interval of a_{i+1} if $i < k$, and, if $i = k$, it gets the value of the starting interval of the agent a_1. Furthermore, the agent changes its order of action from clockwise to counter-clockwise - West, South, East, and North -, i.e., from right to left relating to the order of children.

Figure 3.3 gives an example of such an incremental policy, where an agent with interval $[0, \frac{1}{3}[$ finishes its interval in the 5th node, and then it changes its interval to $[\frac{1}{3}, \frac{2}{3}[$, and also changes the order of action from left-to-right to right-to-left.

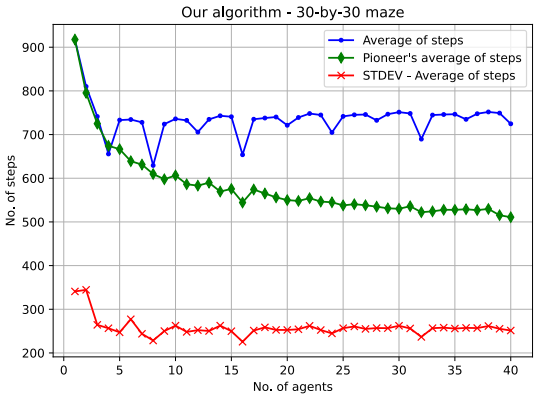
Figures 3.4 and 3.5 present the comparison between our "1 interval" algorithm - evaluated in Section 3.1 - and our "2 intervals" algorithm, where the incremental policy was implemented.



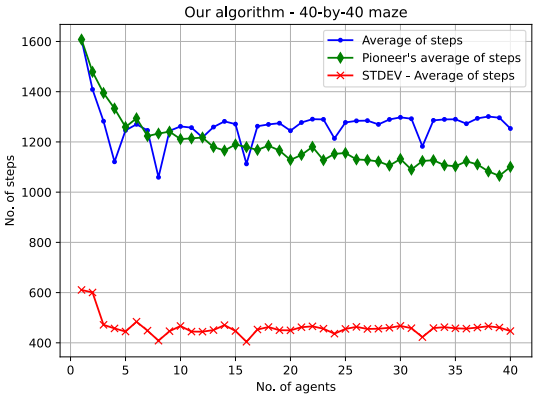
(a) 10×10 maze



(b) 20×20 maze

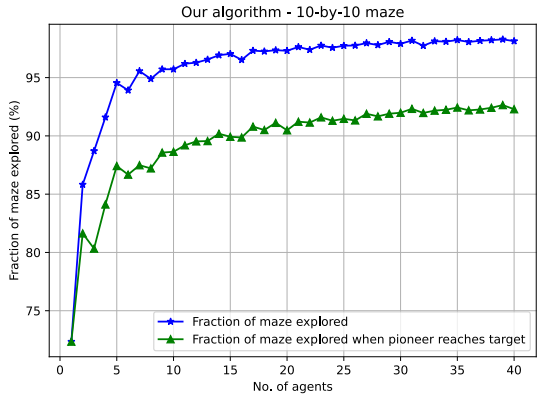


(c) 30×30 maze

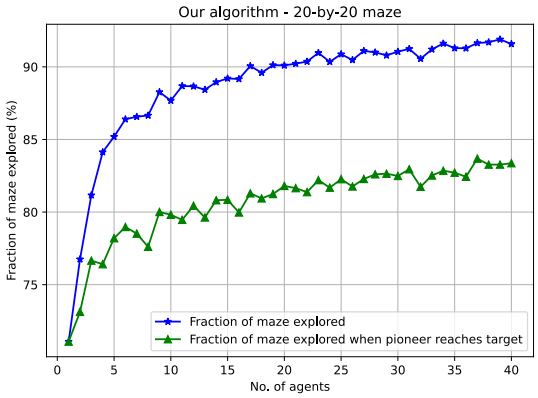


(d) 40×40 maze

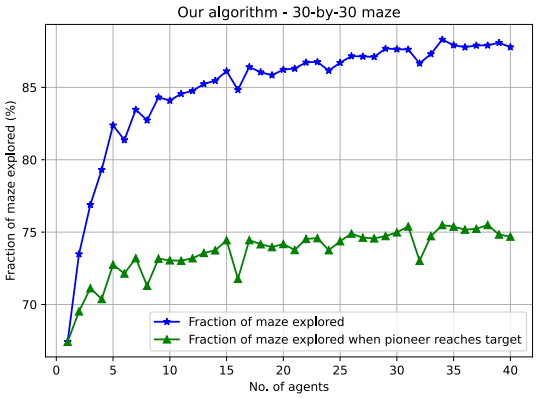
FIGURE 3.1 – Results of the average of steps, the pioneer's average of steps, and the standard deviation of the average of steps. 1 single agent until 40 cooperative agents were run in 250 different mazes for each size.



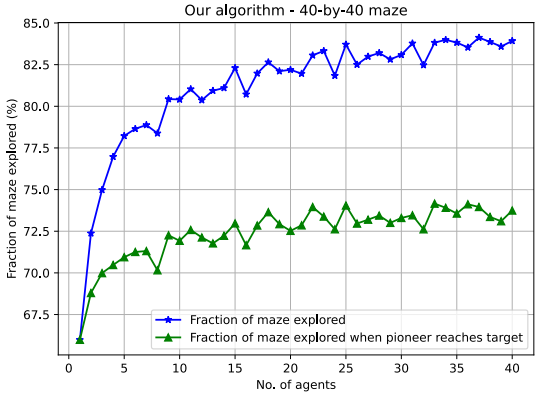
(a) 10×10 maze



(b) 20×20 maze



(c) 30×30 maze



(d) 40×40 maze

FIGURE 3.2 – Results of the fraction of maze explored and the fraction of maze explored when the pioneer reaches the target. 1 single agent until 40 cooperative agents were run in 250 different mazes for each size.

The first conclusion that might be taken from the incremental policy is that it improves the search in the sense of the number of steps that the pioneer must take to reach the target. For instance, as seen in Figure 3.4e, in the context of 40 cooperative agents in 40×40 mazes, with the incremental policy, the pioneer reaches the target with about 18% fewer steps compared to our "1 interval" approach. Moreover, as such a policy improves the dispersion, more cells are visited, as seen in Figure 3.5.

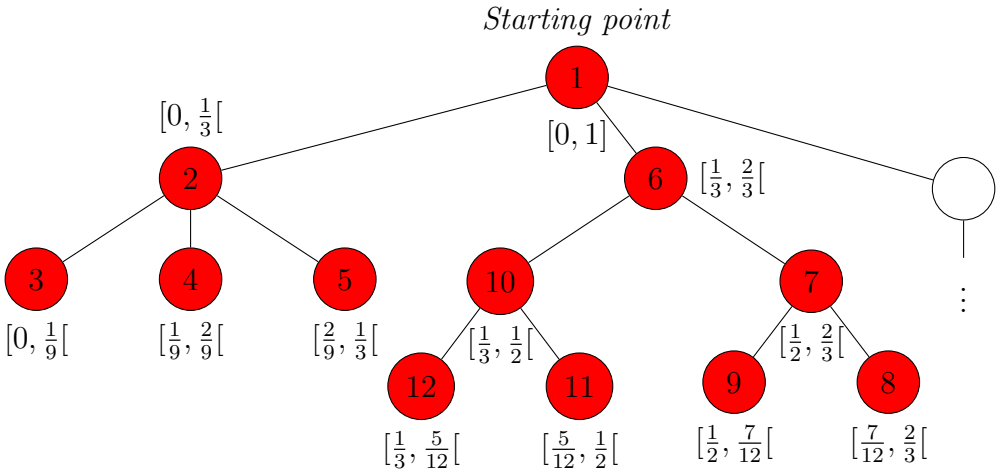


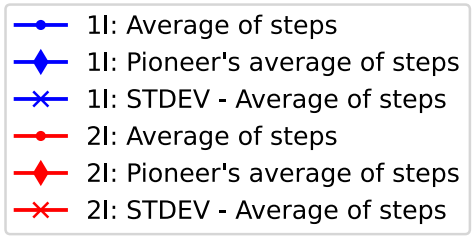
FIGURE 3.3 – Example of an agent with interval $[0, \frac{1}{3}[$ under the incremental policy, where it changes its interval to $[\frac{1}{3}, \frac{2}{3}[$ when it finishes its starting interval, which occurs in the 5th visited cell, and it changes the order of action to right-to-left. It is a fictional maze.

3.3 Our algorithm vs. Tarry's algorithm

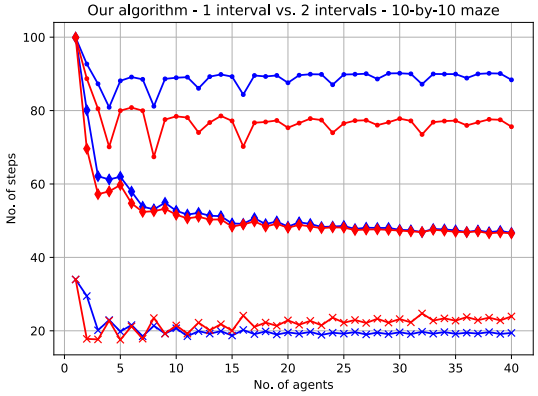
Finally, our algorithm was compared to the extended Tarry's algorithm (KIVELEVITCH; COHEN, 2010), which considers all agents with the capability to communicate with each other. As pointed out in Section 2.5, the extended Tarry's algorithm also aims to find the goal node of some undirected graph. Thus, the authors implemented this algorithm for testing in the same mazes previously generated.

It is important to emphasize that, differently from the extended Tarry's algorithm, our algorithm considers that the agents cannot communicate with each other. Furthermore, Kivelevitch & Cohen (2010) use the "Last Common Location" concept, where the agents are able to come back to the last location that matches the pioneer's path, and then they go through the maze following the pioneer's path to also achieve the goal.

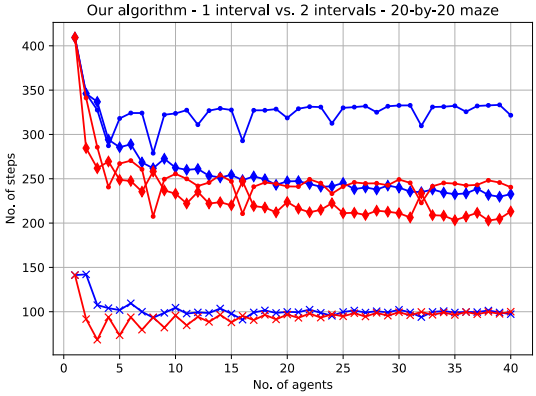
Figure 3.6 compares the pioneer's average of steps of the three algorithms, i.e., our "1 interval" algorithm, whose results were exposed in Section 3.1, our "2 intervals" algorithm, whose results were exposed in Section 3.2, and the extended Tarry's algorithm. Similarly, Figure 3.7 compares the fraction of maze explored when the pioneer reaches the target. As



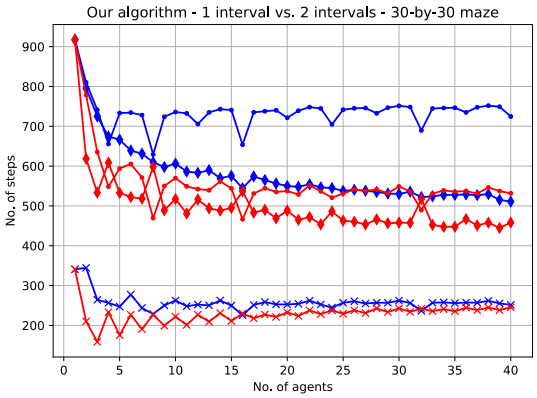
(a) Legend



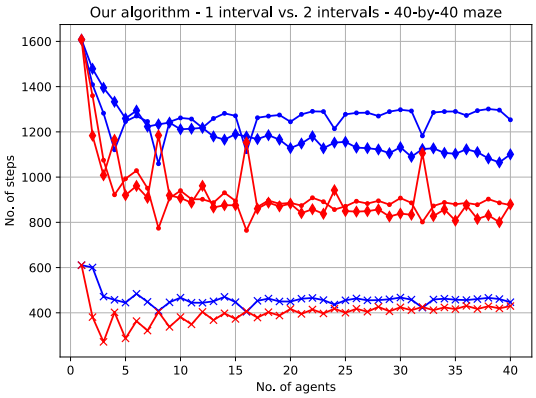
(b) 10×10 maze



(c) 20×20 maze

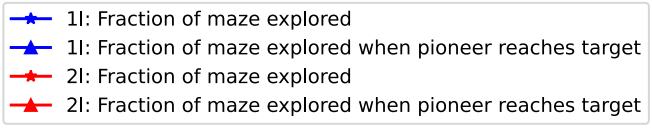


(d) 30×30 maze

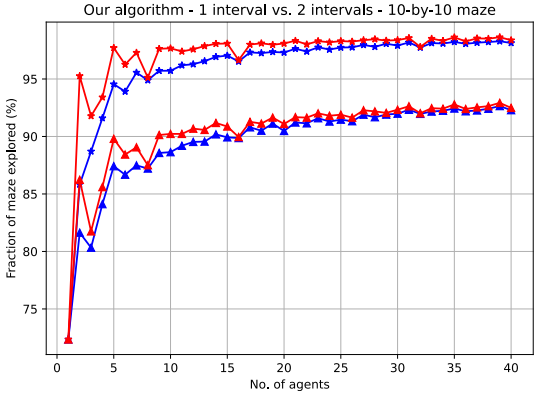


(e) 40×40 maze

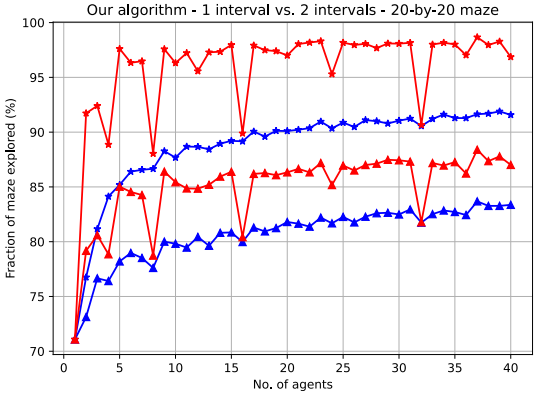
FIGURE 3.4 – Comparison between the results of the average of steps, the pioneer's average of steps, and the standard deviation of the average of steps relating to our "1 interval" algorithm and our "2 intervals" algorithm. 1 single agent until 40 cooperative agents were run in 250 different mazes for each size, and for each algorithm.



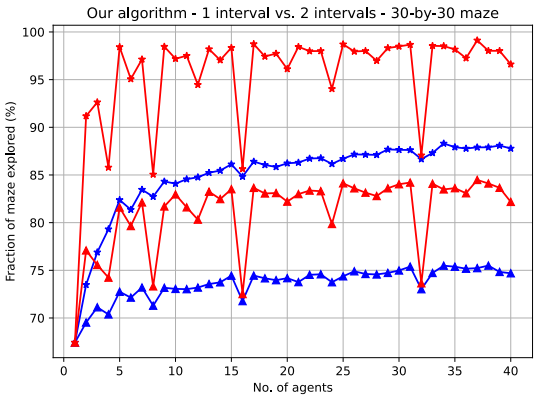
(a) Legend



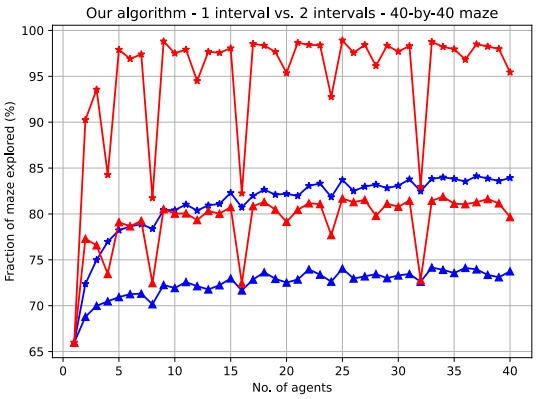
(b) 10×10 maze



(c) 20×20 maze



(d) 30×30 maze

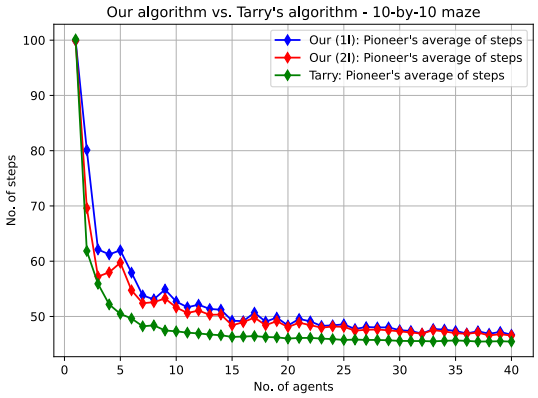


(e) 40×40 maze

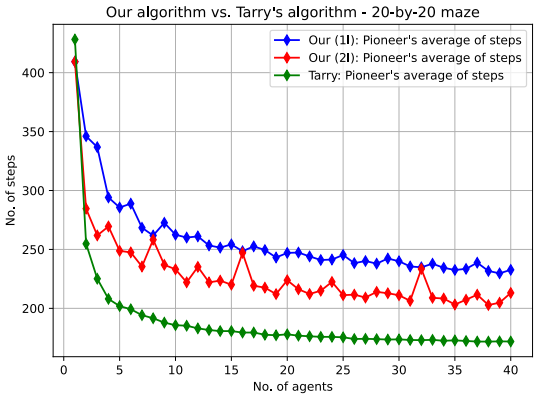
FIGURE 3.5 – Comparison between the results of the fraction of maze explored and the fraction of maze explored when pioneer reaches the target relating to our "1 interval" algorithm and our "2 intervals" algorithm. 1 single agent until 40 cooperative agents were run in 250 different mazes for each size, and for each algorithm.

seen, this work doesn't compare the average of steps of the three algorithms, since all the agents in the extended Tarry's algorithm reach the goal, and it also doesn't compare the fraction of maze explored, since the agents stop the exploration when the pioneer finds the goal, once they come back to the "Last Common Location" to follow the pioneer's path.

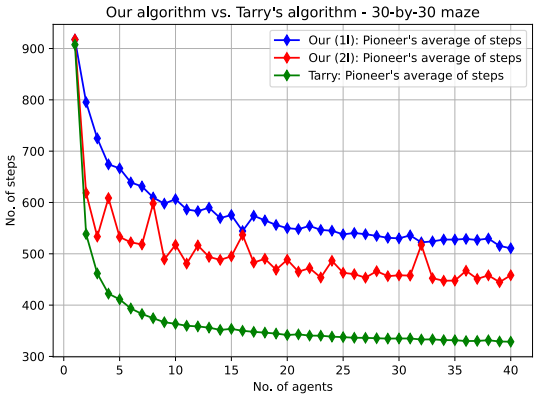
For reaching the goal, the performance of the extended Tarry's algorithm is better than our algorithm, since the first improves the agent dispersion mainly because of the network of visited cells shared with each agent, in which they are able to make the decision of avoiding paths that have already been covered. Fortunately, our algorithm is open to improvements, since it can accept new policies to guide the agent after it finishes its interval. As seen in Section 3.2, our incremental policy improves our algorithm performance, since the pioneer finds the goal faster than our algorithm without incremental policy. Moreover, as seen in Figure 3.7, the agents explore more maze cells in the context of the extended Tarry's algorithm, at least until the pioneer finds the goal, since it boosts the dispersion.



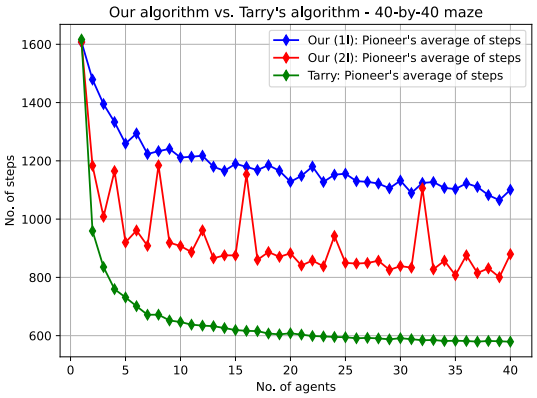
(a) 10×10 maze



(b) 20×20 maze

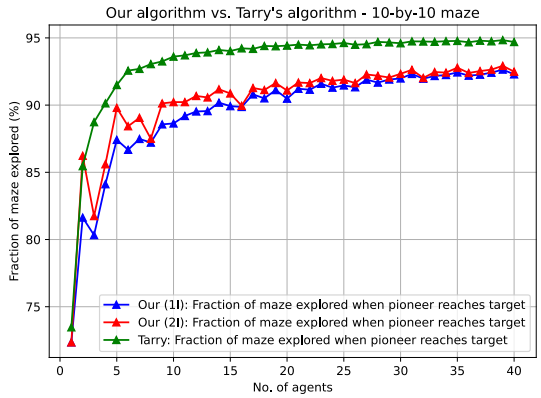


(c) 30×30 maze

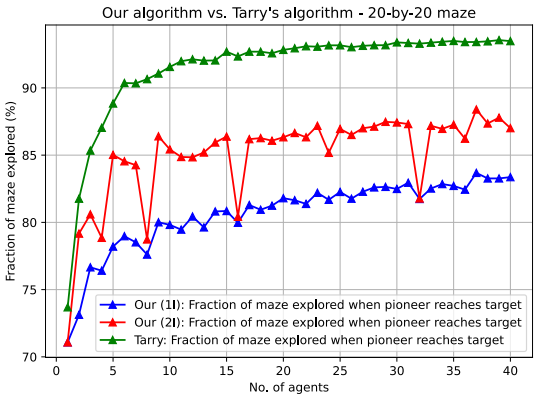


(d) 40×40 maze

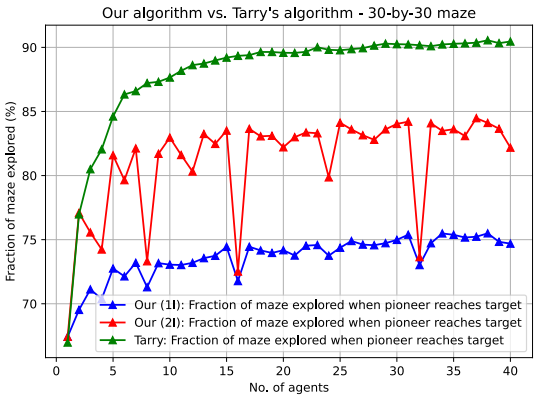
FIGURE 3.6 – Comparison between the results of the pioneer's average of steps relating to our algorithms and the extended Tarry's algorithm. 1 single agent until 40 cooperative agents were run in 250 different mazes for each size, and for each algorithm.



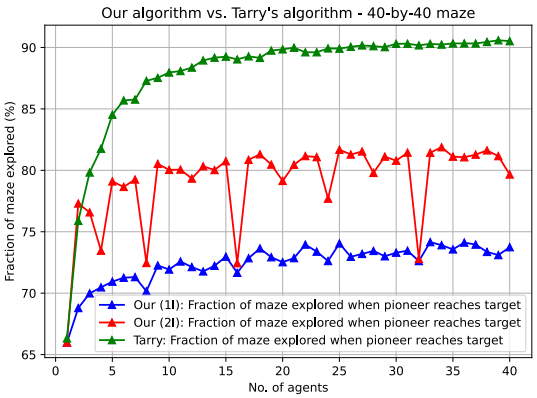
(a) 10×10 maze



(b) 20×20 maze



(c) 30×30 maze



(d) 40×40 maze

FIGURE 3.7 – Comparison between the results of the fraction of maze explored when pioneer reaches the target relating to our algorithms and the extended Tarry's algorithm. 1 single agent until 40 cooperative agents were run in 250 different mazes for each size, and for each algorithm.

4 Conclusions and Future Works

This work presented a solution for searching a goal node in a tree that is simultaneously visited by agents with no communication between them, where the agents are previously programmed to go through the tree as dispersed as possible. Despite the scope of this research being search in undirected acyclic graphs, i.e., trees, it may be extended to more general graphs, and consequently to search the solution of imperfect mazes, in which there is more than one path to find the goal from any cell.

The approach of using a mixed radix numerical representation to the agent's path proved to be a powerful mathematical tool to represent, at the same numerical value, the agent's path and the convergence intervals of the nodes scanned by the agent. Moreover, it saves part of the tree's structure, which might be useful in other situations of graph exploration.

This work also could examine the performance of the extended Tarry's algorithm (KIVELEVITCH; COHEN, 2010), which proved to be an effective algorithm to find maze solutions in a multi-agent context, even though it needs communication between the agents to run, differently from our algorithm.

It is also worth emphasizing that our algorithm is flexible, since it is open to improvements mainly when the agent finishes its starting interval. As seen in Section 3.2, a simple policy improvement produces a faster pioneer, which finds the goal with fewer steps. Therefore, more policies might be integrated in order to make the agents smarter.

Thus, the authors proposed a multi-agent algorithm to search a goal node in trees, where the agents cannot communicate with each other. In this way, they need to be previously programmed to traverse the tree as dispersed as possible. Foremost, this research is open to improvements and comparisons with future works in the context of multi-agent graph exploration without communication.

Bibliography

ARNDT, J. **Mixed radix numbers.** In: Matters Computational. Springer, Berlin, Heidelberg, 2011. https://doi.org/10.1007/978-3-642-14764-7_9

BEISEL, B. W. **Distributed Maze Solving By Cooperative Robotic Platforms.** University of Maryland, Department of Electrical and Computer Engineering, MSc, 2014.

BURGARD, W.; MOORS, M.; STACHNISS, C.; SCHNEIDER, F. E. **Coordinated multi-robot exploration.** IEEE Transactions on Robotics, 2005. 21(3), 376-386. doi:10.1109/tro.2004.839232

CORMEN, Thomas H.; LEISERSON, Charles E.; RIVEST, Ronald L.; Stein, Clifford. **Introduction to Algorithms.** 4th ed. The MIT Press, 2022.

KIVELEVITCH, Elad; COHEN, Kelly. **Multi-Agent Maze Exploration.** Journal of Aerospace Computing, Information, and Communication, 2010. 7. 391-405. 10.2514/1.46304.

MATARIĆ, M. J. **Designing and Understanding Adaptive Group Behavior.** Adaptive Behavior, Vol. 4, No. 1, 51-80. 1995. 4(1), 51-80. doi:10.1177/105971239500400104

MANBER, U. **Introduction to Algorithms**: A Creative Approach. 1st ed. Addison-Wesley, 1989.

NAEEM, M. A. **pyamaze**. GitHub, 2021. Available at: <https://www.github.com/MAN1986/pyamaze>. Accessed on: June 6, 2023.

RAO, Nageswara; KARETI, Srikumar; SHI, Weimin; IYENGAR, Sundararaj. **Robot Navigation in Unknown Terrains**: Introductory Survey of Non-Heuristic Algorithms. 1999. 10.2172/10180101.

SADIK, A. M. J.; DHALI, M. A.; FARID, H. M. A. B.; RASHID, T. U.; SYEED, A. **A Comprehensive and Comparative Study of Maze-Solving Techniques by Implementing Graph Theory.** International Conference on Artificial Intelligence and Computational Intelligence, 2010. doi:10.1109/aici.2010.18.

SHIELDS, R. **Cultural Topology: The Seven Bridges of Konigsburg, 1736.** Theory, Culture & Society, 2012. 29(4-5), 43-57. doi:10.1177/0263276412451161.

FOLHA DE REGISTRO DO DOCUMENTO

1. CLASSIFICAÇÃO/TIPO	2. DATA	3. DOCUMENTO Nº	4. Nº DE PÁGINAS
TC	31 de outubro de 2023	DCTA/ITA/TC-061/2023	47

5. TÍTULO E SUBTÍTULO:

Multi-agent graph exploration without communication

6. AUTOR(ES):

Arthur José de Sousa Rodrigues

7. INSTITUIÇÃO(ÕES)/ÓRGÃO(S) INTERNO(S)/DIVISÃO(ÕES):

Instituto Tecnológico de Aeronáutica – ITA

8. PALAVRAS-CHAVE SUGERIDAS PELO AUTOR:

Maze; Graph; Search; Multi-agent; Mixed radix

9. PALAVRAS-CHAVE RESULTANTES DE INDEXAÇÃO:

Algoritmos; Sistemas multiagentes; Árvores (matemática); Grafos (computação); Trajetórias; Computação.

10. APRESENTAÇÃO: (**X**) **Nacional** ( ) **Internacional**

ITA, São José dos Campos. Curso de Engenharia de Computação. Orientador: Prof. Dr. Luiz Gustavo Bizarro Mirisola. Coorientador: Prof. Dr. Vitor Venceslau Curtis. Publicado em 2023.

11. RESUMO:

Computational search algorithms have been studied by many scholars and engineering companies since the last century due to the real-life applications of such algorithms, such as in airline scheduling, planning path on maps, search engine algorithms, social media marketing, Internet routing protocols, robotics, etc. Usually, these methods rely on abstract data types, including graphs and trees, to represent a real world problem as a well defined and computationally tractable algorithm. In the context of Computer Science, graphs are abstract data types that might support computational search algorithms.

Focusing on maze-solving algorithms, this work proposes a method to explore a graph by previously coordinated agents that cannot communicate with each other. The agents are previously programmed to spread through the graph as dispersed as possible in order to avoid traversing the same portion of the graph and then minimizing the number of steps to find the goal node. This approach is related to real problems, such as sea exploration, search in large wall structures, search with low energy-based agents in inhospitable environments, etc. Thus, there are previous motivations to do this research.

This work also investigates a mixed radix numerical representation as a tool, which, in the context of this research, is able to represent the visited nodes and their corresponding paths from the root, as well their related positions in an inorder traverse through the maze structured as a tree. It allows not only to continue on the path through its nearby neighbors but also to elaborate strategies to maximize the dispersion of the agents in the maze.

Finally, this report presents a comparison between our algorithm's performance and the performance of the extended Tarry's algorithm, proposed by Kivelevitch & Cohen (2010), whose methods rely on communication between the agents, differently from this work's purpose, in which the agents cooperatively explore a graph with no communication.

It is worth emphasizing that, although the purpose of this work might be understood from the perspective of graphs in a general way, this research focuses on perfect mazes, i.e., trees.

12. GRAU DE SIGILO:

(**X**) **OSTENSIVO** ( ) **RESERVADO** ( ) **SECRETO**