INSTITUTO TECNOLÓGICO DE AERONÁUTICA

Arthur José de Sousa Rodrigues

MULTI-AGENT GRAPH EXPLORATION WITHOUT COMMUNICATION

Bachelor's Thesis
2023

Computer Engineering

CDU 000.00

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.
30f.

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. I. Instituto Tecnológico de Aeronáutica. II. Title.

BIBLIOGRAPHIC REFERENCE

DE SOUSA RODRIGUES, Arthur José. **Multi-agent graph exploration without communication**. 2023. 30f. Bachelor's Thesis – Instituto Tecnológico de Aeronáutica, São José dos Campos.

CESSION OF RIGHTS

AUTHOR'S NAME: Arthur José de Sousa Rodrigues
PUBLICATION TITLE: Multi-agent graph exploration without communication.
PUBLICATION KIND/YEAR: Bachelor's Thesis / 2023

It is granted to Instituto Tecnológico de Aeronáutica permission to reproduce copies of this work and to only loan or to sell copies for academic and scientific purposes. The author reserves other publication rights and no part of this work can be reproduced without the authorization of the author.

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 non-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: Jun 16, 2023.

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

Acknowledgments

I thank my family for invest time and financial resources in my studies. And to give me unconditionally love and motivation.

I thank my H8 friends for always support me with 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 to Brazilian Air Force for support 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 inteligível. No campo da Ciência da Computação, grafos são tipos abstratos de dados que podem servir como suporte ferramental para algoritmos de busca.

De uma abordagem relacionada a algoritmos de busca em labirintos, este trabalho de graduação propõe um método para explorar grafos em um âmbito multiagente, em que os agentes não são capazes de se comunicarem entre eles, o que faz referência a 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.

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 transfer a real-world problem into a delimited and intelligible environment. In the context of computer science, graphs are abstract data type that might support computational search algorithms.

From the maze-solving algorithms approach, this work aims to propose a method to explore a graph in a multi-agent situation, where the agents cannot communicate with each other, that makes reference to real-life problems, like deep sea exploration, search in large wall structures, search with low energy-based agents in inhospitable environments.

List of Figures

List of Tables

List of Algorithms

List of Abbreviations and Acronyms

DFS	Depth First Search
BFS	Breath First Search

List of Symbols

Contents

1 Introduction

This bachelor thesis aims to discuss the main maze exploration algorithms and then propose a method where multi-agents must find the maze solution without any type of communication, only working with coordinated and deterministic distributions to guide their behaviors. The main challenge of this research is to find a distributed exploration approach 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 proposed maze structure abstraction is a traditional regular grid that can be generalized into graphs.

This report is the initial part of the bachelor's thesis, and this chapter intends to introduce the general concept of the related thesis and present the motivation (Section 1.1), the related work (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). And 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, the multi-agent exploration potentially can speed up the exploration, despite of it will be 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 Tarry's algorithm generalization. It is important to emphasize that maze-

solving algorithms consider that the structure is unknown.

Traditional multi-agent maze exploration approaches is based on internal communication between agents, where each agent knows about visited cells by another agent. It avoids a second exploration in a useless path and thus it decreases computational costs. However, there are some real situations where communication is limited or impossible, such as deep sea exploration, search in large wall structures, or search with low energy-based autonomous agents.

However, the zero-communication approach between agents have not been concretely found in the literature despite 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 maze based on agents without communication, that might be generalized to graph exploration algorithms.

1.2 Related work

Multi-agent cooperative system approaches are common in literature, especially when it comes from robotics researches. In the context of communicable agents, mainly in robotics, this chapter presents some related works.

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, despite 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. However, this bachelor thesis intends to study a different technique based on zero-communication range.

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 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 graph theory algorithms demands higher computational complexity, they are more proficient compared to related algorithms that doesn't use graph representation, and only make decisions relied on the neighborhood.

Based around maze-solving algorithms, Kivelevitch & Cohen (2010) proposed a generalization of Tarry's algorithm, but the new 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 an dynamic map of the maze excluding redundant information, allowing information sharing. The authors presents in the article the performance of the proposed solution, where a group of virtual coordinate agents is required to find the goal without an *a priori* knowledge of the maze, so-called "maze exploration".

Beisel (2014), similarly to 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 approach 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. The author 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 discuss 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). Please check Manber (1989) for more details.

Manber (1989) works with some representations of graphs in his book. One of 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 will have the vertex v_i.

It is worth to mention that there is a subdomain of graph data type called tree. A tree is also an abstract data type, which can be represented topologically as a hierarchical structure of nodes. 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.

From Section 2.2, a perspective of a maze from a tree and a graph topologies will be explored.

1.3.2 Maze

This report define a maze at 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 by 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 will be presented in Section 2.1, this work considered a maze, for simulation purposes, as a two-dimensional gridded space composed by 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 good computational representation of a maze, which will be a tree if the maze has no loops, or equivalently, 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 for an agent 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 which generalizes standard positional numerical systems to allow a different base for each digit. In a traditional numerical system, there is not a base variation through the numbers, nevertheless, 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:

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

Despite the aforementioned representation, this work presents a slightly different approach in Section 2.4.

2 Models

2.1 Maze and multi-agent exploration

For simulation purposes, this work modeled 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 across the related edge. On the other hand, if there is not a wall, an agent has a free way to traverse the maze through the related edge. It is worth to mention that, if an edge has a wall, the edge of the adjacent corresponding cell also has necessarily a wall.

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 according the current explored path and its programmed initial rules. So that it doesn't go through the same path more than one time, it stores the visited cells. Thus, when there are various posible branches to extend the agent's path, it ignores cells already visited by itself, and, if there is not a candidate to be a possible branch, the agent go back to the previous visited cell. Furthermore, specifically to this research, differently from some approaches presented in Beisel (2014), Burgard et al. (2005), and Kivelevitch & Cohen (2010), there is no intercommunication between agents in a multi-agent situation to solve the maze.

Naeem (2021) developed an open-source maze generator. It is a python module that creates randomly 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.

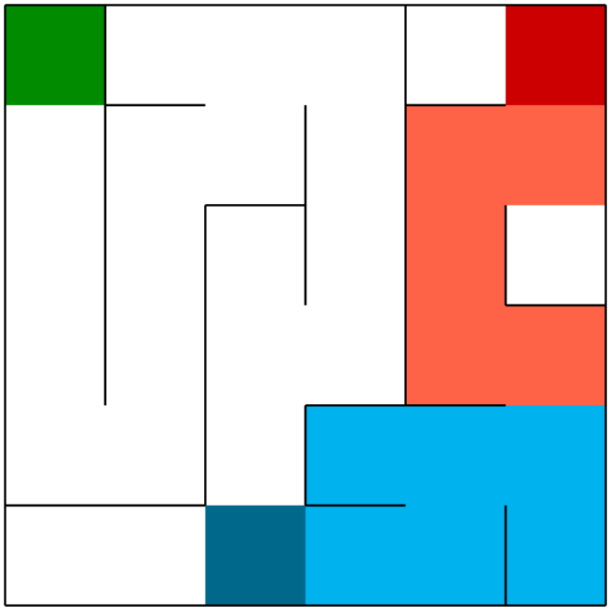

FIGURE 2.1 – Blue and red agents traversing a 6×6 maze. The goal is represented by the green entity. This maze is a perfect maze (NAEEM, 2021).

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;
- despite the last statement, and also considering that an agent ignores visited cells, an agent path always might be individually interpreted as a tree.

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.

2.3 Multi-agent exploration without communication

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

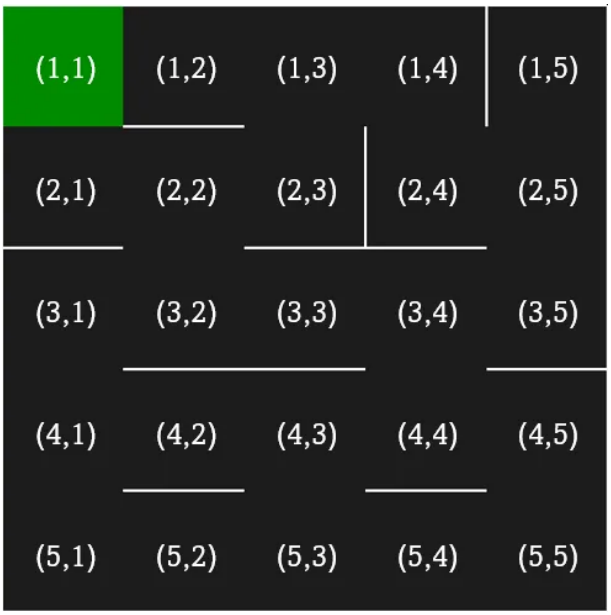

FIGURE 2.2 – Maze cell indices. Source: Naeem (2021).

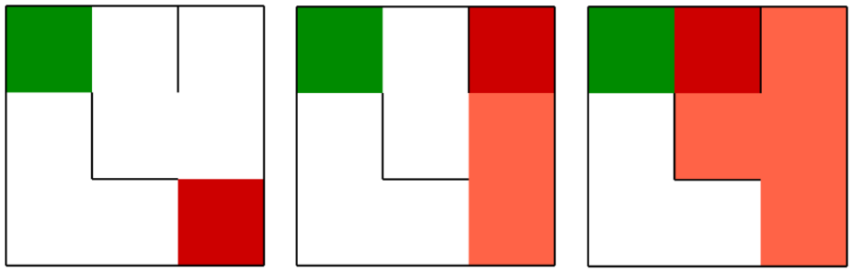

FIGURE 2.3 – Red agent traversing a maze toward the green goal. This maze is not a perfect maze.

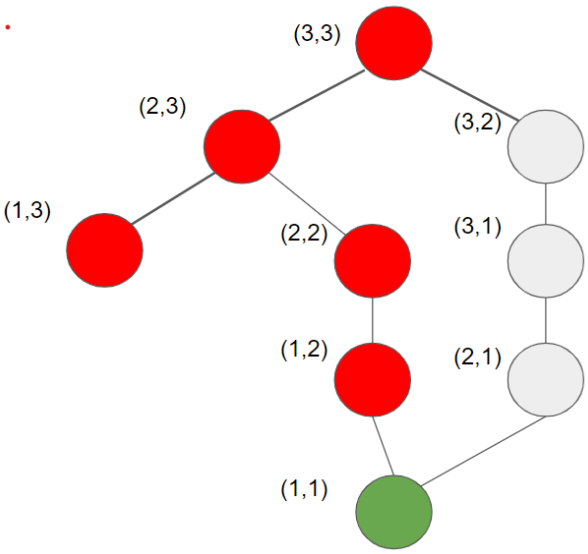

FIGURE 2.4 – Graph representation of the maze presented in Figure 2.3. The cell indices are pointed out, and the cells visited by the red agent are also pointed out.

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 representation of the maze. Given a initial node that the agent starts its path through it, the agent checks if the node has children. Supposing that a node of the graph has at least one child, i.e., there are not completely walled cells, important statements are established:

- if an agent finds the node where is the goal, the agent finishes its path;
- if an agent is not in the node where is the goal and the node has only one child, the agent necessarily goes through this node child;
- if an agent is not in the node where is the goal and the node has more than one child, the agent needs to decide to which child it will move itself.

To establish an decision algorithm to the last statement, this work proposes firstly a interval division related to the agents and the children. 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. On the other hand, this work has handled convergence intervals to each node, where the agents tends to match its interval with the node convergence interval. In this way, an agent intends to traverse the maze as dispersed from the others as possible.

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, and considering the agent path as an adjacency list A, 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. As mentioned previously, there is no communication between agents, therefore each agent needs to explore the maze individually, and calculates 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.

Figure 2.5 presents the converge intervals of the nodes if at least two agents explored the maze presented in Figure 2.3 using Pseudocode 1.

Thus, the goal of this work aims to establish an efficient algorithm that interrelates the agent intervals to the maze node intervals. Furthermore, the authors observed that a mixed radix representation to the agent path is a powerful mathematical tool to relate the path to the interval of the node that the agent is visiting, and it will be present in the next report.

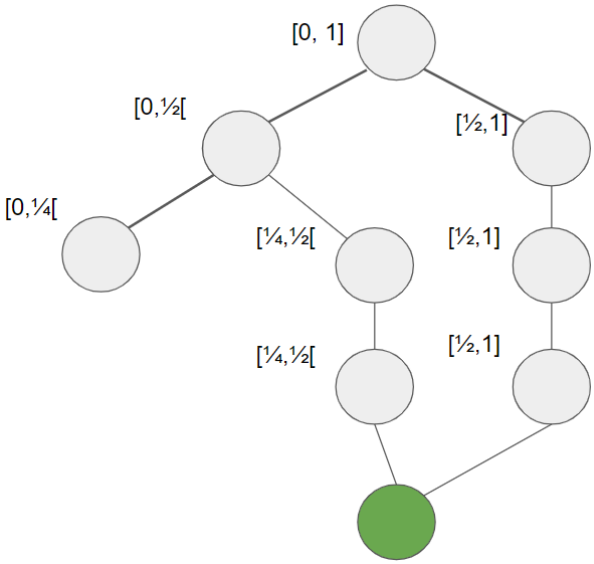

FIGURE 2.5 – Graph representation of the maze presented in Figure 2.3, with intervals related to the agent's range of action.

2.4 Mixed radix representation to the agent path

In Section 2.3, this work presented a method for an agent to traverse the maze by defining ranges of action, despite it needs some improvements. Pseudocode 1 explains 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 the tree, because it only needs to know the current node interval to make a decision.

This work establishes a method to calculate the node intervals without the agent having necessarily to save 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., $[0, 1]$. According to the order of analysis of the children and the mathematical tools to calculate node intervals

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

```
agent_interval ← [a_min, a_max[      // Previously defined interval of the agent
current_node ← v_1                   // Agent starts the exploration at node v_1
visited_nodes ← [ ]                  // Array of visited nodes by the agent is initially empty
intervals[v_1] ← [0, 1]              // v_1 is the root
A ← ⊘                                // Initialize the empty adjacency list of the tree A
complete(A, v_1)                     // In the tree A, create a list for v_1 and complete the list of
                                     // v_1 with the children of v_1. The order of insertion of child
                                     // must follow some rule. In the case of this work, the inser-
                                     // tion is clockwise (North, East, South, West)

while current_node is not the goal cell do
  if current_node is not in visited_nodes then
    append(visited_nodes, current_node)
    max ← get_maximum_value(intervals[current_node])
    min ← get_minimum_value(intervals[current_node])
    partition ← (max − min)/amount_of_children
    count ← 0
    foreach 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 foreach
  end if
  if current_node has no child or all of current_node's children were visited then
    current_node ← get_parent(A, current_node)
    continue
  end if
  foreach child v_j of current_node in A do
    if v_j is in visited_nodes then
      continue
    else
      max_node ← get_maximum_value(interval[v_j])
      min_node ← get_minimum_value(interval[v_j])
      max_agent ← get_maximum_value(agent_interval)
      min_agent ← get_minimum_value(agent_interval)
      if min_agent < max_node then
        // This condition only works because the order of children is well defined
        current_node ← v_j
        break
      end if
    end if
  end foreach
  // If current_node remains the same as the beginning of the iteration, it means
  // that the agent finished its interval, and then it needs to change its behavior.
  // Then its behavior is domain dependent. It will continue the DFS under some
  // criteria
  if current_node remains the same as the beginning of the while iteration then
    current_node ← continue_DFS_under_some_criteria(A, current_node)
  end if
end while
```

presented in Pseudocode 1, the items below present the agent behavior while it doesn't find the goal cell. It is important to emphasize that the value of the mixed radix numerical representation is the same as the minimum value of the node interval.

- if the current node has only one child, append I_1 to the mixed radix numerical representation of the path, and go to the child;
- if the current node has n children, and $n > 1$, then calculate the value of the mixed radix numerical representation of the path and, from this value, the related value to each child, and then make the decision according to the agent interval, as approached in Pseudocode 1. Pseudocode 2 presents a way to calculate computationally the value of the mixed radix numerical representation of the path;
- if the current node has no child, go back to the parent, and remove the last appended value from the mixed radix numerical representation of the path.

Figure 2.6 presents an example for the mixed radix numerical representation approach. A blue agent with an interval $[1/3, 2/3[$ is traversing a maze, and its current path is $0.0_2 2_3$. The minimum value for each interval is also presented.

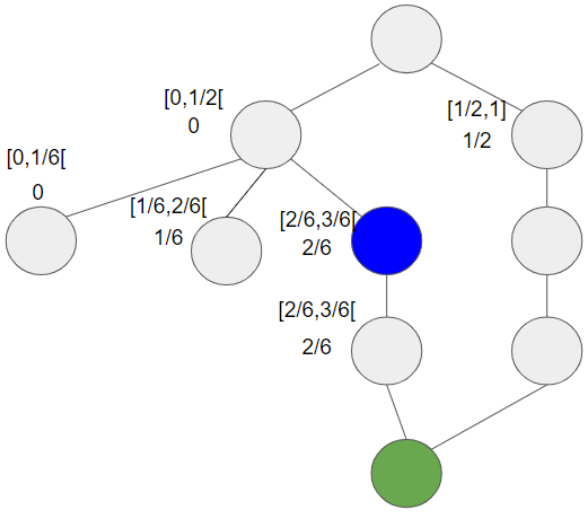

FIGURE 2.6 – Blue agent with an interval $[1/3, 2/6[$ traversing the maze presented in Figure 2.3. Its current path is $0.0_2 2_3$, that is equal to $2/6$.

Algorithm 2 Methods related to mixed radix numerical representation approach.

```
struct mr_element      // Structure of mixed radix element
{
  type numeral
  type radix
}
path ← [ ]             // Initialize the agent path. The agent is at the root of the tree.
                       // Numerically it means zero (or "0.")

function append(path, numeral, radix):
  struct mr_element element
  element.numeral ← numeral
  element.radix ← radix
  add(path, mr_element)     // Append the element to the end of the path array

function remove(path):
  subtract(path)     // Remove the last element of the path array

function get_mixed_radix_value(path):
  value ← 0
  factor ← 1     // It is necessary to take into account if the numeral of the element
                 // is 0
  foreach element in path do
    if element.numeral == I then
      continue
    end if
    value ← value + (1 − value) · factor · element.numeral/element.radix
    if element.numeral == 0 then
      factor ← factor · element.radix
    else
      factor ← 1
    end if
  end foreach
  return value
```

3 Results

A code has been developed to compare the proposed algorithm with other maze-solving algorithms. The code was delivered to the advisor and co-advisor by email.

4 Next steps

Next steps has been discussed with the advisor and the co-advisor, and the main points are described below.

- Until 18th August: to find out a tree simulation application;
- Until 1st September: to establish a mixed radix representation to the agent path;
- Until 6th October: to compare the proposed algorithm with well-known algorithms in the context of maze-solving challenges;
- Until 10th November: to generalize the proposed algorithm to search algorithms in graphs.

Bibliography

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

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

Burgard, W., Moors, M., Stachniss, C., Schneider, F. E. (2005). **Coordinated multi-robot exploration.** IEEE Transactions on Robotics, 21(3), 376-386. doi:10.1109/tro.2004.839232

Cormen, Thomas H., Leiserson, Charles E., Rivest, Ronald L. and Stein, Clifford. **Introduction to Algorithms.** 4th : The MIT Press, 2022.

Kivelevitch, Elad & Cohen, Kelly. (2010). **Multi-Agent Maze Exploration.** Journal of Aerospace Computing Information and Communication. 7. 391-405. 10.2514/1.46304.

Matarić, M. J. (1995). **Designing and Understanding Adaptive Group Behavior.** Adaptive Behavior, 4(1), 51-80. doi:10.1177/105971239500400104

Manber, U. **Introduction to Algorithms: A Creative Approach** 1th : Addison-Wesley, 1989.

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

Sadik, A. M. J., Dhali, M. A., Farid, H. M. A. B., Rashid, T. U., Syeed, A. (2010). **A Comprehensive and Comparative Study of Maze-Solving Techniques by Implementing Graph Theory.** 2010 International Conference on Artificial Intelligence and Computational Intelligence. doi:10.1109/aici.2010.18.

Shields, R. (2012). **Cultural Topology: The Seven Bridges of Konigsburg, 1736.** Theory, Culture & Society, 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	06 de junho de 2023	DCTA/ITA/DM-000/2023	30

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

9. PALAVRAS-CHAVE RESULTANTES DE INDEXAÇÃO:

Maze; Graph; Search; Multi-agent

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. Defesa em 00/00/2023. Publicada em 00/00/2023.

11. 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 inteligível. No campo da Ciência da Computação, grafos são tipos abstratos de dados que podem servir como suporte ferramental para algoritmos de busca.

De uma abordagem relacionada a algoritmos de busca em labirintos, este trabalho de graduação propõe um método para explorar grafos em um âmbito multiagente, em que os agentes não são capazes de se comunicarem entre eles, o que faz referência a 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.

12. GRAU DE SIGILO:

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