AI50: ARTIFICIAL INTELLIGENCE 6-MONTHS PROJECT

THEME: PATROLLING PROBLEM

FUNCTIONAL REQUIREMENTS

Walid OUBRAIM

Supervised by Fabrice Lauri, Mahjoub Dridi, and Nicolas GAUD.

AUTUMN 2024

Table des matières

I- CONTEXTUALIZATION

1- Problem description and use case

The multi-agent patrolling problem involves coordinating multiple agents to navigate an environment represented as a graph, ensuring that all strategic zones or points are visited regularly. Patrolling thus consists of visiting strategic locations in a region at regular intervals.

Each zone is represented by a node in the graph, with paths connecting these zones as edges, each with an associated cost that represents the travel time between two nodes. The main objective is to minimize the interval between successive visits to the same zone, referred to as "idleness," to ensure effective and continuous surveillance. This problem is complex due to the need to coordinate the agents to avoid redundant efforts while maximizing the territory coverage.

For the project, the chosen application is the surveillance of a military base. The site is modelled as a connected graph, where each strategic zone (node) must be regularly visited by agents to prevent any intrusions or suspicious activities.

2- Constraints

In the context of patrolling a military base, agents must contend with various constraints that directly impact their ability to conduct effective and continuous patrols. These restrictions, whether related to resource limitations, operational rules, or environmental conditions, add complexity to the multi-agent patrolling problem, presenting additional challenges in planning and coordination.

- **Limited number of agents:** The base has a limited number of patrol agents, requiring optimal allocation to cover all strategic zones without redundancy.
- **Mandatory return to starting node:** Agents must return to a central node (such as a command center) after each patrol to refuel or replenish resources, which imposes return trips and limits the autonomy of each mission.
- **Variable Weather Conditions:** Weather conditions (rain, snow) affect agent movement, potentially increasing travel times or rendering certain areas temporarily inaccessible.

These constraints are not exhaustive but represent the most critical ones we have chosen to focus on for effective patrolling in a military base.

3- Optimization terms definition

The area patrolled by the agents is represented by a weighted, undirected graph $G = (V, E)$ where V is the set of n nodes (representing strategic zones), and E is the set of edges connecting these nodes. Each edge (i, j) has an associated cost C_{ij}, which represents the travel time between nodes i and j.

Our **objective function** is to minimize idleness across the graph. The idleness of a node i at time t, denoted $I_i(t)$, is defined as the time elapsed since the last visit by an agent to that node. Using idleness, we can define two criteria to measure the overall idleness of the graph at any given time:

- **Worst Idleness (WI) :**

$$WI_t(G) = \max_{i=1,\dots,N} I_i(t)$$

- **Average Idleness (AI) :**

$$AI_t(G) = \frac{1}{N} \sum_{i=1}^{N} I_i(t)$$

The objective is to **minimize both the worst idleness and the average idleness** of the graph:

$$\min WI_t(G) \;\&\; \min AI_t(G)$$

II- PRESENTATION OF SOLUTIONS APPROACHES & STRATEGIES

1- Eulerian Circuit

A Eulerian circuit is a path in a graph that traverses each edge exactly once before returning to the starting point. This property can be exploited by dividing the graph into sub-circuits, each assigned to an agent. The algorithm is therefore based on a fixed path, where each agent follows a predetermined route that efficiently covers all areas within the base.

Advantage: The complexity is reduced once the graph is divided, as each agent follows a fixed route.

Disadvantage: Each agent follows a fixed path with no flexibility and without sharing information with other agents.

2- Simulated annealing

Simulated annealing is a stochastic optimization method inspired by the cooling process of metals, and can be adapted to solve the multi-agent patrol problem, where several agents must effectively cover an area while minimizing unmonitored areas and optimizing their trajectories.

Advantage : Simulated annealing offers flexibility and can escape local minima, allowing it to explore a wide range of solutions for effective patrolling coverage. It's also relatively simple to implement and adaptable to various cost functions relevant to multi-agent tasks.

Disadvantage : It can be computationally expensive and may take a long time to converge, especially for complex multi-agent systems with large search spaces. Additionally, finding the right cooling schedule and parameters requires tuning, which can be challenging and time-consuming.

3- Reactive agents (without planning)

Each reactive agent moves towards the neighbouring zone that has been neglected the longest (i.e. the one with the highest idleness). Agents make local decisions based only on information from their immediate surroundings.

Steps:

- **Next Zone Selection**: At each time step, each agent observes the neighbouring zones (neighbouring nodes in the graph) and moves to the one that hasn't been visited for the longest time.
- **Reactive Movement**: Movement is based solely on reducing local idleness. Agents do not communicate directly with one another and do not coordinate their actions.
- **Update**: After each move, the idleness of zones is updated, and the process continues.

Advantage: This algorithm is simple to implement and does not require coordination between agents.

Disadvantage: Lack of communication between agents.

4- Cognitive agents (based on Dijkstra or A*)

Cognitive agents can plan their movement by considering the entire territory. Each agent plans its route to the node with the highest idleness (even if that node is not a direct neighbour) by using a shortest-path algorithm, such as Dijkstra or A*. Agents communicate to avoid choosing the same destination as another agent.

Advantage: Enables a more strategic patrol by targeting high-idleness nodes across the entire graph and coordinating agent movements to improve coverage.

Disadvantage: Higher computational complexity due to path planning and communication overhead.

5- Ant colony optimization

This approach is inspired by the behavior of ants, where agents (or "ants") deposit pheromones on paths they travel and prioritize paths that have not been visited recently. Over time, pheromones evaporate, encouraging agents to explore new areas instead of following the same routes.

At each time step, the next node that an ant will visit is determined by the concentration of pheromones on each neighboring node. Each ant visits unvisited nodes

(tracked using a "taboo list") until it completes a full circuit of the graph. This cycle is repeated for all ants, starting from their initial node.

The ACO approach involves deploying different ant colonies on the graph at each iteration. After each iteration, the colony that minimizes the worst idleness of the graph is selected. The ants in this optimal colony then deposit additional pheromones on the nodes they traveled through, encouraging subsequent ants to favor this optimized path. Each new iteration builds on the previous one, with higher pheromone concentrations guiding ants toward the best-performing solution.

Advantage: The ACO approach dynamically balances exploration and exploitation of paths, adapting over time to minimize idleness across the graph.

Disadvantage: Requires repeated iterations to converge to an optimal solution and may be computationally intensive, especially on larger graphs.

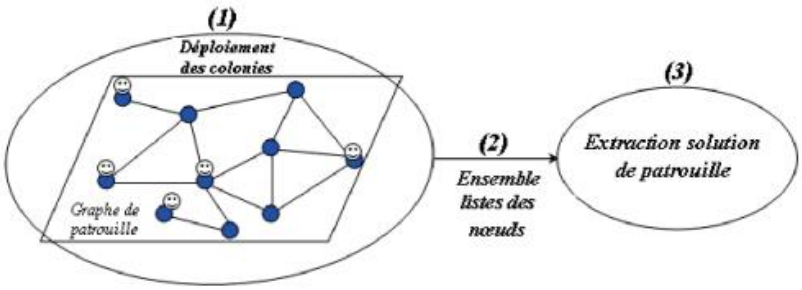

Figure 1: Emergence of optimal solution

Within a colony, ants move across different nodes in the graph, with a preference for adjacent nodes that have been visited the least. Each ant k in colony l maintains a taboo list specific to its colony, recording the nodes it has already visited. When selecting the next node j to visit, the ant uses a probability $p_{i,j}^{k,l}$. This probability determines the likelihood of choosing node j among all adjacent nodes, with the ant prioritizing the node that has the highest probability.

The probability $p_{i,j}^{k,l}$. that an ant k from colony l will move from node i to node j is calculated using a specific formula (as covered in IA52) :

$$p_{ij}^{k,l}(t) = \begin{cases} \frac{[\tau_{ij}(t)]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{u \in autorisé_l}[\tau_{iu}(t)]^{\alpha}[\eta_{iu}]^{\beta}} & \text{si } j \in autorisé_l \\ 0 & \text{sinon} \end{cases}$$

In this context, **authorized** refers to the set of nodes that the ants in a colony have not yet visited.

STRATEGIES ON STARTING NODE:

- **Single Initial Node**: All agents begin their patrol from the same initial node.
- **Dispersed Deployment**: In this approach, agents are spread out across the graph to occupy the nodes furthest from one another. This distribution phase is determined by an evolutionary algorithm, serving as a heuristic to form pre-cycles. Consequently, agents are not required to initiate and conclude their patrol from their initial deployment node. The starting and ending node of each patrol cycle is instead the node assigned during the distribution phase.
- **Deployment found through genetic algorithm:** The initial deployment is found through a genetic algorithm to find the optimal strategy.

III- PROJECT STACK AND EVALUATION

1- Tools and Technologies

To effectively implement and visualize our multi-agent patrolling solution, we have selected a set of tools tailored to our project's requirements in terms of graph manipulation, agent coordination, and environment visualization:

- **Python**: As the primary programming language for this project, Python offers extensive libraries for graph-based algorithms, pathfinding, and visualization, making it ideal for developing and testing our patrolling algorithms.
- **NetworkX**: Network is a Python library dedicated to the creation, manipulation, and analysis of complex networks. In this project, NetworkX will be used to model the military base environment as a weighted, undirected graph. This includes defining nodes (representing strategic zones), edges (paths between zones), and travel costs, which are essential for implementing and evaluating our patrolling strategies.
- **Pygame**: Pygame is a Python library designed for developing games and graphic applications. In our project, it will serve as a visualization tool to simulate and display agent movements across the graph in real time. Pygame will allow us to create an interactive graphical interface where users can observe agent patrol patterns, node coverage, and idleness over time, enhancing our ability to assess and refine our algorithms.

This stack of tools provides a flexible and robust foundation for implementing and testing multi-agent patrolling strategies, ensuring that the solution is both functional and visually interpretable.

2- Testing and results evaluation

To ensure that our multi-agent patrolling algorithms are effective and adaptable across different environments, we will test them on a diverse set of graph structures. This diversity in testing will allow us to evaluate algorithm performance on various configurations, node counts, and connection types, simulating different operational conditions within a military base.

- **Graph Diversity**: We will create a range of test graphs with varying numbers of nodes and edge configurations (tiny graphs, medium graphs, big graphs, low-density graphs, high-density graphs, mixed-type graphs, …)
- **Benchmark Resources**: To validate our results and ensure comparability, we will utilize benchmark data provided by Mr. Lauri on Teams. Additionally, we will reference data and methodologies from the following sources (IJCA **Online Journal, Volume 88, Number 12 and CEC 2014 Proceedings**)

This combination of diverse graph structures and standardized benchmarks will enable a comprehensive evaluation of our algorithms' effectiveness in minimizing idleness and achieving optimal patrolling patterns across different scenarios.

CONCLUSION & PERSPECTIVES

This document outlines the functional scope of our multi-agent patrolling project, establishing a foundational understanding of the problem and the constraints relevant to patrolling a military base. We have defined the primary objectives, including minimizing idleness across strategic zones and optimizing the coordination of agents to maximize coverage. Several algorithmic approaches were introduced, each with unique strengths and trade-offs in terms of complexity, flexibility, and efficiency.

In the next phases of the project, our focus will shift to the practical implementation and testing of these algorithms. Future developments could include:

- **Enhanced Constraints**: Incorporating additional real-world factors such as energy limitations, dynamic threats, or detection of suspicious activities to further refine agent behavior.
- **Algorithmic Improvements**: Exploring advanced optimization techniques or hybrid algorithms that combine the strengths of different approaches.
- **Scalability Testing**: Extending our framework to larger and more complex environments to test robustness and adaptability.
- **Real-Time Adaptation**: Developing capabilities for agents to dynamically adapt routes based on changing conditions, such as new obstacles or unexpected events.

These perspectives aim to build a fully functional and adaptable solution that can meet the demands of real-world military surveillance scenarios, ensuring continuous and efficient patrols in even the most challenging environment