**Improving Autonomous Agent Search in Real-World Environments: Strengths, Shortcomings, and Potential AI Techniques**

## Introduction

This report explores AI techniques to improve autonomous agents for dynamic, real-world search tasks. It analyses the strengths and shortcomings of prior approaches and evaluates advanced AI methods for adaptability and efficiency.

## Strengths and Shortcomings of Implemented Approaches

***Strengths***

* **Efficient Pathfinding (Part 1)**: A\* allowed agents to find shortest paths to targets, while taking energy constraints into consideration. This provides a good foundation for optimal pathfinding in static, predictable environments. In real-world problems, such techniques could be leveraged in robotic systems navigating hazardous environments, such as search-and-rescue drones.
* **Route Planning (Part 2)**: The A\* implementation was extended to create more complex routes, accounting for energy. The solutions were concrete, managing to successfully visit all possible oracles in most of the worlds. This would be helpful in environmental monitoring tasks (e.g. surveying multiple ecological hotspots) where energy efficiency impacts operational range and coverage area.
* **Multi-Agent Exploration (Part 3)**: Agents cooperatively explored an unknown maze, uncovering the environment through local observations, illustrating basic multi-agent coordination. Some very simplistic shared information for more efficient exploration was successfully leveraged. Swarm robotic-like tasks in wider area search-and-rescue, or distributed monitoring of ecosystems using multiple agents could benefit from such algorithms.

***Shortcomings***

* **Inefficient Resource Management (Parts 1 and 2)**: Pathfinding to energy stations was slow, and route planning was suboptimal. Agents lacked logic to utilize nearby resources to construct more complex paths, “tunnel visioning” on reaching a single target at a time in the shortest path possible, ignoring additional nearby targets that could be utilised to create a longer, but more rewarding path, resulting in extra steps and energy consumption. In search-and-rescue operations, this could delay locating survivors or waste critical resources.
* **Agent Conflict and Deadlocks (Part 3)**: The lack of sophisticated conflict resolution mechanisms led to agents clashing, resulting in inefficient exploration and pathfinding failures. This caused reduced efficiency, and (albeit shortly lived) bottlenecks at times. In real-world multi-agent scenarios, this can lead to mission-critical delays or resource bottlenecks, compromising the system’s effectiveness.
* **Static Environment Assumptions**: Both algorithms assumed static environments with no dynamic changes. Without the ability to adapt to terrain changes, real-world applications risk becoming ineffective in many disaster (or otherwise) related scenarios.

## Potential AI Techniques for Enhanced Performance

Various AI techniques can be employed to address the above shortcomings through the lens of a dynamic environment. This exploration will focus on the following two:

**1. Reinforcement Learning via the Q-Learning Algorithm:**

Reinforcement Learning allows agents to utilise trial and error to gain the necessary experience to learn optimal policies for a given environment. This can lend itself well to agent adaptability, as opposed to, say, supervised learning where there exists a set with predefined answers. The added ability to make decisions based on probabilistic outcomes is essential in environments where the effects of actions are not deterministic.

The Q-Learning algorithm is an efficient way to employ reinforced learning. Agents learn a Q-value function , representing the expected utility of taking action in state . The estimation of Q-values can be determined either through **Temporal Difference** (comparing the current state and action values with the previous ones), or via **Bellman’s Equation** (recursively calculate the value of a given state, aiming to determine its optimal position). Environmental features such as nearby obstacles, energy levels and target locations should be incorporated in the state representation, allowing agents to perform optimal actions per state. Agents should be rewarded (or punished) depending on actions in such a way as to encourage reaching goals, avoiding obstacles, and conserving energy. This would mean highly positive rewards for reaching a goal, negative for hitting obstacles, and likely mildly negative ones per time step to promote efficiency. Lastly, strategies like ε-greedy could be employed to balance exploration of new actions and exploitation of known good actions.

The above offers great flexibility, allowing for agents to adapt to new situations easily (provided overfitting is accounted for), coupled with scalability, enabling agents to handle large and complex state spaces. However, training time (and computational cost) will be a clear issue, especially given the sheer number of different substantial interactions with the environment agents can/should have during learning. Additionally, collecting training data for real-world applications (e.g. search-and-rescue) may be quite costly, and designing good reward functions can prove quite challenging.

**2. Dynamic Pathfinding via D\*:**

D\* Lite is an incremental heuristic algorithm, building upon A\*. It focuses on updating only the affected parts of the path whenever changes are detected. It is designed for environments where changes occur after the initial path has been planned, and it aims to ensure that the path remains (near) optimal even as the environment changes. Essentially, it reuses previous search efforts to, via consistent heuristics, efficiently replan paths when changes occur, reducing computational overhead.

An implementation of the above could build upon the existing codebase, by computing an initial path from the start to the goal using A\*, while constantly monitoring the environment to observe changes. When a change is detected, the cost of affected nodes can be updated, and the path from the current position can, thus, be adjusted accordingly.

*A black text on a white background

Description automatically generated*

Figure 1: Proposed Pseudocode for D\* Lite Algorithm

This approach provides computational efficiency and minimised processing time. It also lends a good base for real-time agent adaptations in situations where the environment changes frequently. However, the algorithm depends on the agent’s ability to accurately detect changes in the environment. Additionally, frequent changes in highly dynamic environments can still, despite its efficiency, impose computational burdens.

The above techniques would be useful in situations such as search-and-rescue scenarios in dynamic disaster zones. Obstacles (rubble and debris) may shift due to aftershocks or structural collapses, while survivors may be searching for exits, relocating or panicking. Similar semantics apply to looking for fugitives on the run, environmental disaster monitoring, surveillance, and much more.

## Conclusion

Leveraging RL’s adaptability and D\* Lite’s efficient real-time pathfinding, enables agents to overcome foundational limitations, improving performance in real-world resource-limited spatial search problems.