UNIVERSITY OF TRENTO

Department of Information Engineering and Computer Science

Master's Degree in
Artificial Intelligence Systems

Final Dissertation

Exploring the Use of LLMs for Agent Planning: Strengths and Weaknesses

Supervisor
Paolo Giorgini

Student
Davide Modolo
229297

Academic year 2023/2024

Contents

Abstract

1 Introduction

2 Background

In this thesis, we will analyze in detail the behavior of an LLM as an agent within a controlled environment.

Before presenting all the work carried out in detail, this chapter aims to provide a comprehensive explanation of all the theoretical foundations necessary to understand the steps presented in the following chapters. Starting from a brief introduction of Artificial Intelligence just to define the boundaries in which we are working, we will move to the core concepts. In particular, we want to highlight what an LLM is and how it works, with a special focus on the Attention mechanism and how the uncertainty of an LLM can be calculated. This will serve as a basis for correctly interpreting the results analyzed in Section 6.

There will also be a broader discussion on agents in a strict sense and "LLM agents" to better show the difference between our implementation and what is currently being discussed over the media.

To better define the context of this thesis, we will examine the main alternative approaches to solving a logistical problem currently studied in the literature.

2.1 Artificial Intelligence

Right now in the media, AI is being used as a synonym for Large Language Models. However, AI is a broader concept that includes many techniques and methodologies.

Summary of different kind of AI ending with Generative Models end with NLP history

2.2 Large Language Models - LLMs

LLMs are generative models released with the paper "Attention is All You Need"

2.2.1 LLMs' Uncertainty

Understanding the uncertainty of an LLM is crucial to correctly interpret the text it generates. If we ask for a yes/no question, it would make a different impact on us reading "Yes" or "Yes - Uncertainty 49%". Moreover, this would let us get some kind of explainability behind these complex and opaque systems.

cite Hallucinations

2.3 Agents

As widely explained in the book "An Introduction to Multiagent Systems" [2], we can summarize the definition of an agent as an autonomous entity that perceives its environment through sensors and acts upon it through effectors, making decisions based on its perceptions and objectives in order to achieve specific goals.

This definition highlights several key aspects of agents:

- Autonomy: Agents operate without direct human intervention, controlling their own actions.
- Perception and Action: They interact with the environment via sensors (perception) and effectors (action execution).
- Decision-making: Agents select actions based on their internal model, goals, and the state of the environment.
- Non-determinism and Adaptability: Since environments are generally non-deterministic, agents must be prepared for uncertainty and potential failures in action execution.

- Preconditions and Constraints: Actions are subject to certain conditions that must be met for successful execution.

Thus, an agent's fundamental challenge is deciding which actions to perform in order to best satisfy its objectives, given the constraints and uncertainties of its environment.

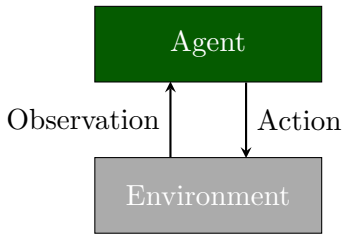


Figure 2.1: Agent Design Scheme
Source: redesign of a scheme in [2]

As shown in Figure 2.1, an agent is some entity that perceives the environment and reacts to it. The environment can be anything from a simple thermostat to a complex system like a self-driving car. The idea is that the agent is able to react to a change in the environment and take actions to achieve its goals.

We will analyze in detail the prompts and the choices in the Chapter 5 Section 5.2, but to give an some kind of help to align with the definition above, we can map some of its concept to what this thesis will analyze:

- Perception and Action: what the server sends about the current state of the environment can be seen as the perception of the agent, while the action it can take will be given in the prompt in a specific way.
- Decision-making: the decision-making process will be the generation of the text by the LLM, weighted by the uncertainty.
- Non-determinism and Adaptability: to emulate the non-determinism of the environment, the state received by the server will be used "raw" in the prompt, without any hard processing or parsing.
- Preconditions and Constraints: living in a "limited" map with a fixed number of cells, is itself a constraint the agent must consider.

2.3.1 BDI Architecture

The Belief-Desire-Intention (BDI) architecture is a widely adopted framework in artificial intelligence (AI) for modeling rational agents. It was formally developed by Rao and Georgeff in 1995 [1] and has been implemented in several architectures, including PRS (1987), dMARS (1998), JAM (1999), Jack (2001), and JADEX (2005). BDI provides a structured approach to practical reasoning, allowing agents to function effectively in dynamic and unpredictable environments.

2.3.1.1 Core Components of BDI

BDI agents operate based on three key components:

- Belief: Represents the agent's knowledge about the world, including past events and observations. Given the agent's local perception and limited computational resources, beliefs act as cached, imperfect information rather than absolute knowledge
- Desire (Goals): Defines the agent's objectives or preferred end states, such as "desiring to graduate." Desires help in justifying why an agent takes specific actions and enable reasoning about goal interactions, particularly in failure recovery scenarios
- Intention: Represents the commitments of an agent toward achieving specific goals through selected plans. Intentions provide structure by ensuring persistence and internal consistency,

allowing for incremental planning and adaptation in real-time. They also serve as a means of coordinating multiple agents in distributed systems.

BDI has been extensively used in fields like robotics, automated planning, and multi-agent systems. For instance, JADEX is a popular Java-based BDI framework used in simulations and decision-making applications.

2.4 State of the Art

A logistic problem is a fundamental challenge in the field of Artificial Intelligence (AI), encompassing tasks such as route optimization, supply chain management, and delivery scheduling. These problems arise in various domains, including transportation, e-commerce, and manufacturing, where efficient resource allocation and decision-making are critical. Given the complexity of modern logistics, AI has emerged as a powerful tool for finding optimal or near-optimal solutions.

Several AI-based approaches can be used to address logistic problems. Traditional operations research techniques, such as linear programming and heuristics, have been widely employed. However, with the increasing availability of data and computational power, machine learning (ML) and deep learning methods have become more prevalent. These methods can predict demand, optimize routes dynamically, and enhance decision-making under uncertainty. Additionally, reinforcement learning (RL) has gained attention for its ability to learn optimal strategies through trial and error, particularly in dynamic and unpredictable environments.

In the recent years with the explosion of Large Language Models (LLMs), many researchers started to apply them to different fields, including planning and logistics.

2.4.1 PDDL Based Solutions

Planning Domain Definition Language (PDDL) is a human-readable format for problems in automated planning that gives a description of the possible states of the world, a description of the set of possible actions, a specific initial state of the world, and a specific set of desired goals.
Source: Wikipedia[1]

The fundamental distinction between a PDDL-based solution and any Machine Learning/Deep Learning approach lies in the very nature of how problems are defined and solved. In a PDDL-based system, the problem must be explicitly encoded using a formal, structured language that describes the initial state, goal state, and available actions. The planner then takes on the computationally intensive task of exploring a vast search space, systematically generating and evaluating possible action sequences to determine an optimal path from the initial state to the goal state.

While effective in structured environments, this method is inherently time-consuming and computationally demanding. Since the planner must traverse a potentially enormous state space—guided by heuristics but still constrained by the rigid formalism of PDDL, it may struggle with real-time decision-making, making it unsuitable for dynamic, fast-paced applications.

Listing 2.1: Domain file example for a bit toggle problem

```
1  (define (domain bit-toggle)
2    (:requirements :strips :negative-preconditions)
3    (:predicates
4      (bit ?b)                        ; predicate meaning
5                                      ; bit ?b is set (true)
6    )
7
8    (:action setbit
9      :parameters (?b)
10     :precondition (not (bit ?b))    ; can only set a bit if
11                                     ; it is not already set
```

[1] https://en.wikipedia.org/wiki/Planning_Domain_Definition_Language

```
12      :effect (bit ?b)                ; setting the bit to true
13    )
14
15    (:action unsetbit
16      :parameters (?b)
17      :precondition (bit ?b)          ; can only unset a bit if
18                                      ; it is currently set
19      :effect (not (bit ?b))          ; setting the bit to false
20    )
21  )
```

Listing 2.2: Problem file example for a bit toggle problem

```
1   (define (problem bit-toggle-full-problem)
2     (:domain bit-toggle-full)
3     (:objects
4        b1 b2 b3
5     )
6     (:init)                           ; Initially all bits are unset (false)
7
8     (:goal                            ; It can be any combination of T/F
9        (and (bit b1) (bit b2) (not(bit b3)))
10    )
11  )
```

With the increasing number of variables (actions or predicates), the number of arcs grows exponentially. A little example that makes this problem easy to visualize is the Domain where we can have N possible bits, that can be turned to `true` or `false` (Domain file in Listing 2.1) and the Problem where everything start at `false` and we want a specific final combination (Problem file in Listing 2.2).

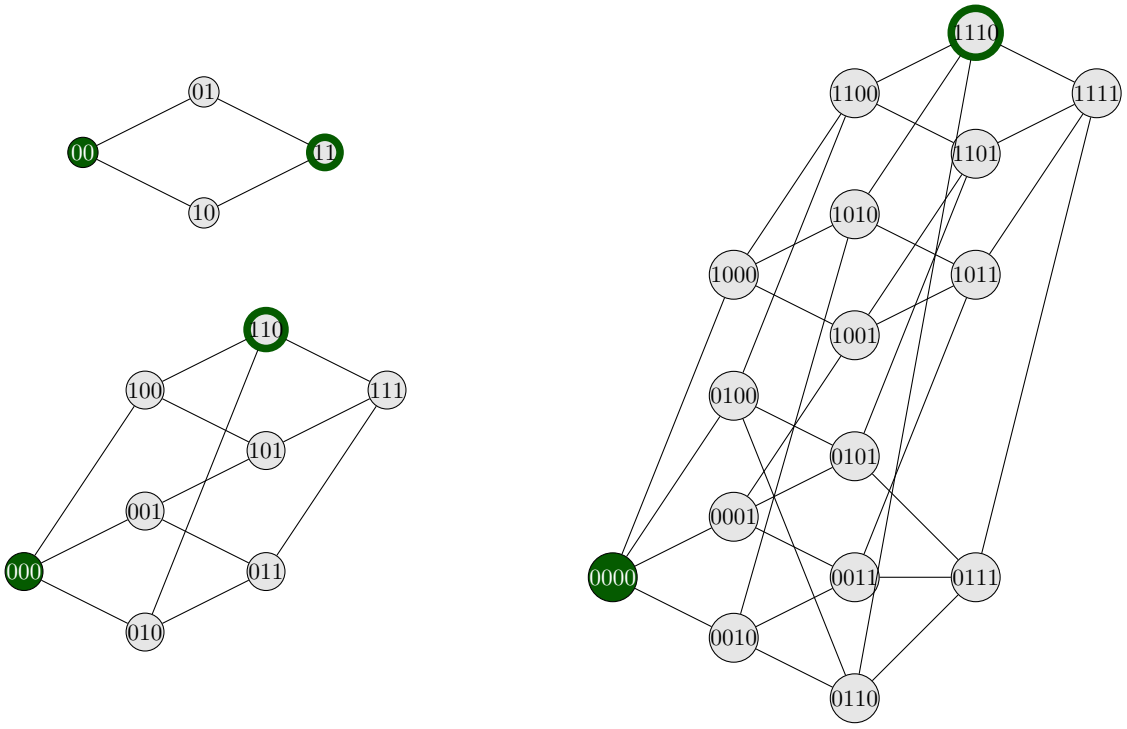
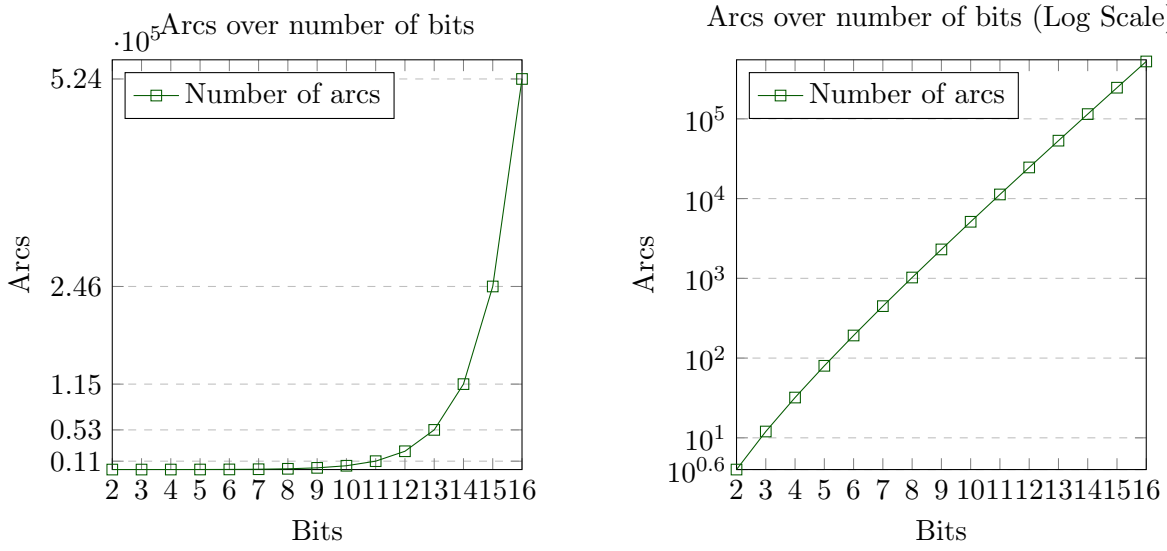


Figure 2.2: Graphs for bit-toggle problem with 2, 3, and 4 bits

As we can see in the plot Figure 2.3, we can visualize the number of arcs (example of graphs for 2, 3 and 4 bits in Figure 2.2) grows exponentially with the number of bits, as well as the number of states



(a) label 1 (b) label 1

Figure 2.3: Arcs per Bit

obviously. This shows how even a simple problem with a simple solution can become time-intensive and not suitable for real-time applications.

Listing 2.3: Plan for the bit toggle problem, solved by LAMA-first planner

```
1          ; Found Plan (output)
2 (setbit b3)
3 (setbit b2)
4 (setbit b1)
```

However, a PDDL approach is more explainable, since all the information is provided by the user and the output result is a sequence of actions (example at Listing 2.3). This makes it easier to understand and debug the solution, as each step is explicitly defined. Of course, there might be different paths to reach the goal, and the planner might choose one based on heuristics or optimization criteria. This transparency in the decision-making process is one of the key advantages of using PDDL for planning problems.

An example of a problem related to the one presented in this thesis, solved using PDDL, can be found in the paper "An AI Planning Approach to Emergency Material Scheduling Using Numerical PDDL" by Yang et al. [3]. In their work, they utilize PDDL 2.1 that allows to model the scheduling problem, incorporating factors such energy consumption constraints. Their approach employs the Metric-FF planner to generate optimized scheduling plans that minimize total scheduling time and transportation energy usage. However, while this demonstrates the applicability of AI planning to emergency logistics, their model simplifies the real-world scenario by assuming predefined transport routes, limited vehicle types, and abstract representations of congestion effects. This highlights a broader limitation of PDDL in capturing the full complexity of dynamic and uncertain environments often encountered in emergency response situations.

2.4.2 Reinforcement Learning Solutions

Reinforcement Learning (RL) is a branch of machine learning focused on making decisions to maximize cumulative rewards in a given situation. Unlike supervised learning, which relies on a training dataset with predefined answers, RL involves learning through experience. In RL, an agent learns to achieve a goal in an uncertain, potentially complex environment by performing actions and receiving feedback through rewards or penalties.

Source: GeegksforGeeks [2]

Reinforcement Learning is a learning setting, where the learner is an Agent that can perform a set of actions depending on its state in a set of states and the environment. In performing action `a` in state `s`, the learner receive an immediate reward `r(s,a)`. In some states, some actions could be not possible or valid.

The task is to learn a policy (a full specification of what action to take at each state) allowing the agent to choose for each state the action maximizing the overall reward, including future moves.

To deal with this delayed reward problem, the agent has to trade-off exploitation and exploration:

- **Exploitation**: the agent chooses the action that it knows will give some reward
- **Exploration**: the agent tries alternative actions that could end in bigger rewards

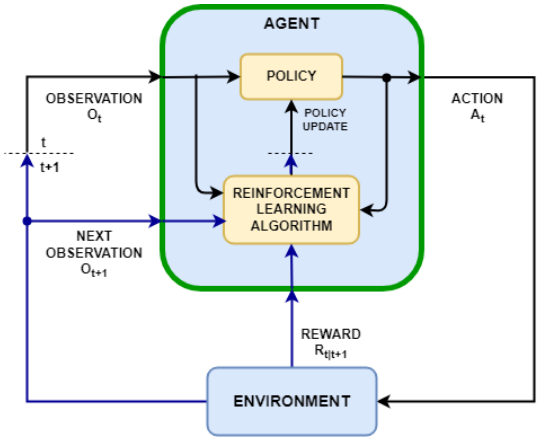


Figure 2.4: RL Agent Scheme
Source: Mathworks[3]

When considering a logistics problem, reinforcement learning naturally comes to mind. This is because defining a reward function is relatively straightforward—it could be measured in terms of packages delivered per minute, per step, or a similar metric. Additionally, the entire process can be simulated in a virtual environment, allowing multiple parallel simulations to accelerate the agent's learning process. As illustrated in Figure 2.4, the structure of the Reinforcement Learning framework closely resembles the agent-based model depicted in Figure 2.1. In both cases, the agent interacts with its environment, receives feedback in the form of rewards, and continuously refines its policy to optimize future performance.

Reinforcement Learning works by defining:

- **Environment**: the world in which the agent operates
- **Agent**: the decision-maker that interacts with the environment
- **Actions**: the possible moves the agent can make
- **Rewards**: the feedback the agent receives for its actions
- **Policy**: the strategy the agent uses to select Actions

However, RL has its own set of challenges. The most common one is the convergence to a local minimum in the reward function. This means that the agent might learn a suboptimal strategy that is not the best one. Moreover, RL is not explainable, meaning that we can't understand why the agent

[2] `https://www.geeksforgeeks.org/what-is-reinforcement-learning/`

[3] `https://it.mathworks.com/help/reinforcement-learning/ug/create-agents-for-reinforcement-learning.html`

took a specific action in a specific situation. This is a big problem in real-world applications, where we need to understand the decision-making process to ensure safety and reliability.

Another issue with RL is the cost of training. Since the agent learns through trial and error, it needs to perform a large number of actions to explore the environment and learn the best strategy. This can be computationally expensive and time-consuming, especially for complex problems with many variables and states. Moreover, once the agent is trained, its adaptability to new environments or situations is limited, as it is optimized for a specific reward function and environment configuration.

> An example of a problem similar to the one presented in this thesis, solved using Reinforcement Learning, can be found in the paper "DeliverAI: a distributed path-sharing network based solution for the last mile food delivery problem" by Ashman et al.

2.4.3 Planning in LLM

2.4.3.1 LLM Agents

Agenti LLM (es. o1 di OpenAI): introduzione a temi emergenti come il Chain of Thought, sottolineando come viene eseguito il reasoning e cosa differisce dal mio approccio

3 Experiment Setting

3.1 Problem Definition

3.2 Environment - Deliveroo.js

3.3 GPT Models

4 Agent Development

4.1 First Approach

4.2 Second Approach

4.3 Final Agent

4.4 Closest Cell to the Goal

5 Data Collection

5.1 Visualize the Attention

5.2 Prompts

5.3 Prompt Creation Choices

5.4 Heatmap Generation

6 Results Discussion

6.1 Stateless

6.2 Stateful

6.3 Stateless and Stateful Combined results

6.4 Closest Cell to the Goal Problems

6.5 Models Comparison

7 Future Works

8 Conclusions

Bibliography

[1] Anand S. Rao and Michael P. Georgeff. BDI agents: From theory to practice. In Victor R. Lesser and Les Gasser, editors, *1st International Conference on Multi Agent Systems (ICMAS 1995)*, pages 312–319, San Francisco, CA, USA, 12-14 June 1995. The MIT Press.

[2] Michael Wooldridge. *An Introduction to Multiagent Systems*. John Wiley & Sons, 1 edition, 2002. Chapter 2.

[3] Liping Yang and Ruishi Liang. An ai planning approach to emergency material scheduling using numerical pddl. In *Proceedings of the 2022 International Conference on Artificial Intelligence, Internet and Digital Economy (ICAID 2022)*, pages 47–54. Atlantis Press, 2022.

Appendix A Attachment

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.