AI for Robotics II
Automated Warehouse Modeling

Salterini Filippo, Amato Francesca, Polese Carolina
Robotic Engineering - University of Genoa

***Abstract*—This report presents the modeling of an automated warehouse scenario as part of the AI for Robotics II course. The warehouse involves two mobile mover robots and at least one fixed loader robot, coordinating to transport crates of varying weights from their initial locations to a loading bay, where they are placed on a conveyor belt. The core objective was to model this system using efficient AI planning techniques while respecting constraints such as crate weight, movement times, and loading rules.**

Additionally, the model incorporates several optional extensions to increase its realism and complexity. These include the handling of grouped crates that must be delivered together, the use of two loaders with differentiated capabilities, the inclusion of battery limits and recharging requirements for mover robots, and the special treatment of fragile crates requiring slower, more careful handling. The resulting model demonstrates how advanced planning strategies can be used to manage real-world warehouse logistics scenarios with a high degree of efficiency and flexibility.

***Index Terms*—PDDL, AI, OPTIC**

I. Introduction

Warehouse automation is a key application of artificial intelligence and robotics, aimed at improving efficiency, reducing costs, and enhancing logistics reliability. In the *AI for Robotics II course*, we were tasked with designing an automated warehouse system where robots collaborate to transport and load crates for delivery. The goal was to create an efficient and realistic model, suitable for AI planning.

The scenario involves physical constraints, robot coordination, and timing requirements. While two mobile robots move crates and a fixed loader places them on the conveyor belt, optional extensions introduce additional challenges, such as managing crate groups, handling fragile items, adding a second loader, and modeling robot energy consumption and recharging.

To address the problem, we used the OPTIC planner, a temporal planner that handles durative actions and numerical fluents, enabling us to manage timing constraints and optimize the plan duration. The aim was to capture a real-world system's limitations while ensuring solvability through modern AI planning tools. This report details the modeling process and key design choices to balance expressiveness and computational efficiency.

II. Material and methods

The automated warehouse scenario was modeled using PDDL, separating domain knowledge from specific problem instances. The domain.pddl file defines the general structure of the environment, including types, predicates, and actions common to all tasks. Each individual scenario is described in a separate problem.pddl file, which specifies the initial state and goals. A unique problem.pddl is created for each instance to be executed with the planner, ensuring flexibility and modularity in testing different configurations.

A. PDDL - Domain

In the PDDL model for the automated warehouse system, the domain.pddl file plays a critical role in defining the types, predicates, functions, and actions required to model the environment and the operations that can be executed. This file essentially sets the foundation for the problem-specific logic, allowing the planner to generate plans based on the defined system dynamics.

The domain file specifies the types of objects (such as robots, crates, and bases), the possible predicates (such as the locations of robots and crates, or the availability of loaders), and the functions (such as battery levels and distances). It also defines various actions, including durative actions (which span over time), to simulate the robots' movement, crate handling, and interactions with loaders.

1) Functions Used: In the domain.pddl file, the following functions are defined:

- Distance: Represents the distance between a crate and a base. This is critical for calculating movement durations.
- Weight: Indicates the weight of a crate, which influences whether a robot can carry it or not.
- Velocity: A constant function representing the velocity of the robots, affecting how quickly they can move.
- Battery: Tracks the battery level of a robot, which influences its ability to perform tasks.

These functions enable the planner to consider numeric constraints, such as movement times based on distance and battery consumption during actions.

2) Actions Defined: The domain.pddl file defines several actions related to the operations of the warehouse. Key actions include:

- The *charging* and *fully_charged* actions manage the battery charging cycle of the robots. The *charging* action allows a robot at a base to occupy a charging station and incrementally recover battery, while *fully_charged* frees the station once the battery reaches 20%, making it available for other robots.
- The group formed by *move_to_crate*, *pick_up*, *move_back*, and *drop_one_robot* defines the sequence for a single robot to handle crates of light or moderate weight. A robot moves toward a crate, picks it up, returns to the base, and drops it there—each step affecting the robot's battery depending on distance and weight.
- For heavier crates, the domain uses a dedicated sequence: *move_two_robot*, *pick_two_robot*, *move_back_two_robot*, and *drop_two_robot*. In this case, two robots coordinate to approach, pick up, transport, and unload a crate too heavy to be handled alone. Battery usage and timing are calculated jointly.
- The action *move_back_two_robot_light_crate* is a variant of the previous group but is specifically used when two robots carry a light crate together. This leads to a reduced duration and energy cost compared to the heavy-crate counterpart.
- The actions *grasp* and *loading* involve the transfer of crates from base to conveyor belt using the main loader. The crate is picked up (*grasp*) and then loaded onto the belt (*loading*) over a fixed duration. These actions are used when the principal loader is available and active.
- A separate flow using *grasp_cheap_loader* and *loading_cheap_loader* handles loading through a cheap loader when the main loader is already working. This secondary path ensures that light crates can still be processed in parallel by delegating work to an auxiliary system.
- When a crate is fragile, the robot uses the *move_to_fragile_crate* action to approach it with greater care. This path ensures that the crate is handled delicately from the very start.
- Finally, *pick_fragile_crate* and *loading_fragile_crate* follow the fragile handling process. Once the robot reaches the fragile crate, it carefully picks it up and loads it onto the belt, using specific handlers designed to avoid damage—highlighting their exclusive use for fragile cargo.

B. PDDL - Problems

In PDDL, the domain file defines the general structure of the planning environment, including all the types, predicates, and actions

Problem	Crate Type	Weight (kg)	Distance	Group
0.5	Regular	70	10	-
	Regular	20	20	A
1	Regular	70	10	-
	Fragile	20	20	A
	Regular	20	20	A
2	Regular	70	10	A
	Fragile	80	20	A
	Regular	20	20	B
	Regular	30	10	B
3	Regular	70	20	A
	Fragile	80	20	A
	Regular	60	30	A
	Regular	30	10	-
4	Regular	30	20	A
	Fragile	20	20	A
	Fragile	30	10	B
	Fragile	20	20	B
	Fragile	30	30	B
	Regular	20	10	-

TABLE I
SUMMARY OF CRATE SPECIFICATIONS FOR EACH PROBLEM INSTANCE.

that can be used by the planner. In contrast, the problem file describes a specific instance of the planning scenario: it specifies the objects involved, their initial state, and the goal to be achieved. Each problem file therefore encodes a concrete challenge that the planner must solve using the actions defined in the domain.

In this project, five different problem instances have been designed, each increasing in complexity and intended to test different aspects of the planning domain, such as the handling of heavy crates, fragile items, and group-based constraints. All problems assume that the movers begin at the loading bay, and that the distances are straight-line measurements from the loading bay to each crate. There is no risk of collision or interference along the paths.

The table I provides a brief description of each problem instance.

C. Planning Engine - OPTIC

D. Model evaluation

III. Results and Conclusion