Automated Robo Warehouse

Pradyumn Mohta
pmohta@asu.edu

Problem Statement

In today's fast-paced e-commerce industry, companies like Amazon operate large automated warehouses where fleets of robots move shelves carrying products to picking stations for human or robotic order fulfillment. The task involves coordinating many robots simultaneously in a dynamic environment while avoiding collisions, ensuring correct deliveries, and minimizing operation time.

This project models a simplified warehouse automation scenario. The warehouse is represented as a rectangular grid, where:

- Robots can move horizontally or vertically between adjacent cells,
- Shelves store products and can be picked up by robots,
- Picking Stations are locations where products must be delivered,
- Highways are special cells where shelves cannot be put down, but robots can pass through.

Each robot in the warehouse is assigned the task of fulfilling a series of orders. Each order specifies a particular quantity of products that must be delivered to a designated picking station. To fulfill an order, a robot must navigate the warehouse, locate a shelf containing the required products, pick up the shelf, transport it to the appropriate picking station, and deliver the necessary quantity of products.

Throughout this process, robots must adhere to a strict set of operational constraints to ensure safety and efficiency:

- Collision avoidance: No two robots may occupy the same grid cell at the same time, nor may they attempt to swap places in a single time step.
- Movement restrictions: Robots can only move horizontally or vertically between adjacent cells; diagonal movements are not permitted.
- Shelf handling: When carrying a shelf, a robot must be careful not to move under another shelf, as the carried shelf adds to the robot's height.
- Highway rules: Robots may pass through highway cells, but they are prohibited from putting down shelves on these cells to maintain clear pathways.

To help visualize the environment, consider a simple 3×3 warehouse grid:

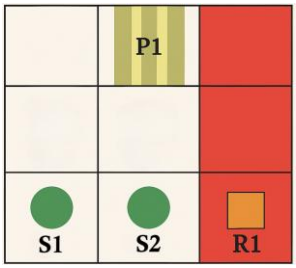

Fig 1. 3x3 grid

Legend: R1 = Robot, S1/S2 = Shelf, P1 = Picking Station, Red Blocks = Highway. In this simple grid, the robot must plan its moves carefully: pick up the shelf, deliver it to the picking station, and avoid entering the highway incorrectly with a carried shelf.

The ultimate objective is to fulfill all outstanding orders correctly and efficiently, completing all deliveries while minimizing the total number of time steps taken. A time step refers to a single unit of time during which a robot can perform one action, such as moving, picking up a shelf, delivering a product, putting down a shelf, or remaining idle. Planning an optimal sequence of actions across all robots, while respecting all operational constraints, is the central challenge of the warehouse automation problem.

Project Background

To model and solve the warehouse planning problem, I used Answer Set Programming (ASP), a form of declarative programming designed for solving complex search and reasoning problems. ASP focuses on describing a problem through logical rules and constraints, allowing a solver to automatically find valid solutions that satisfy all specified conditions. Unlike imperative programming, where control

flow is explicitly defined, ASP specifies what must be true in a solution, and the solver determines how to achieve it.

Before starting the project, I reviewed core ASP concepts, including:

- Representing dynamic worlds where object states change over time,
- Defining actions that cause state transitions (e.g., move, pickup, deliver),
- Enforcing hard constraints to restrict illegal actions (e.g., collisions, movement under shelves, shelf placement on highways),
- Modeling goals that must be achieved over a finite number of time steps.

The learning materials that supported my preparation included the lecture content provided in the course, online tutorials on Clingo (the ASP solver used), and examples of dynamic planning problems such as the Blocks World domain. The Blocks World project was especially useful for understanding how to model time progression, how to track object states dynamically, and how to enforce action effects and preconditions.

For this project, I used Clingo version 5.7.1, running locally to execute the ASP programs. I worked extensively with the simpleInstances provided in the project package, using them to test and debug my encoding before optimizing it to fulfill all orders in the warehouse. Understanding the structure of the initial input facts (e.g., robots' starting locations, shelves' contents, and orders' product requirements) was crucial for accurately designing the dynamic model.

Approach to Solving the Problem

The core idea behind solving the warehouse automation problem was to model the dynamic behavior of the robots, shelves, and products over discrete time steps using Answer Set Programming (ASP). I structured the encoding around the following key elements:

Modeling the Initial State:

```
%
% Initial Setup of the Warehouse
%

% Robots and shelves initial positions at time 0
at(B,U,V,0) :- init(object(robot,B),value(at,pair(U,V))), step(0).
shelf_at(K,U,V,0) :- init(object(shelf,K),value(at,pair(U,V))), step(0).

% Products initially stored on shelves
stock(M,K,Q) :- init(object(product,M),value(on,pair(K,Q))).

% Products required by orders
pending(N,M,Q) :- init(object(order,N),value(line,pair(M,Q))).
```

Fig 2: Initial Setup of the Warehouse

I began by translating the initial warehouse configuration into logical facts:

- Robots, shelves, products, picking stations, and highways were represented as distinct types.
- Their initial positions and properties were extracted from the input facts provided in the instances.

This established the starting point for the dynamic simulation of robot movements and shelf handling.

Defining Robot Actions:

Each robot could perform exactly one action at each time step:

- Movement to an adjacent cell (up, down, left, or right),
- Pickup of a shelf if located at the same position,
- Putdown of a carried shelf,
- Delivery of products at a picking station,
- Or waiting (idle action).

ASP choice rules were used to allow these possibilities while ensuring that only one action per robot was selected per step.

Modeling Action Effects:

The effects of each action were carefully modeled:

- Movement updated the robot's position.
- If a robot was carrying a shelf, the shelf's position was updated alongside the robot.
- Pickup actions changed the robot's carrying status to reflect that it was now moving with the shelf.
- Putdown actions released the shelf onto the grid at the robot's current location.
- Delivery actions reduced both the number of products on the shelf and the number of products required by the order.

All these updates were modeled over time, ensuring that the world evolved dynamically based on robot actions.

Enforcing Constraints:

Hard constraints were crucial to ensure realistic behavior:

- Robots were prevented from moving outside the warehouse grid.
- Two robots could not occupy the same cell at the same time.
- Robots were prohibited from swapping places in a single move.
- Robots carrying shelves could not move under other shelves.
- Shelves could not be put down on highway cells.
- Deliveries were allowed only at picking station locations.
- Robots were restricted from delivering more product units than available on the shelf or required by the order.

These constraints eliminated invalid action sequences and guaranteed warehouse safety rules were respected.

Tracking Product Quantities:

I dynamically tracked the quantities of products on shelves and the undelivered quantities for each order across time steps as you can see in figure 3 below:

- Deliveries reduced product quantities both on the shelves and in the pending orders.

- This ensured that orders were only marked fulfilled when the required units were completely delivered.

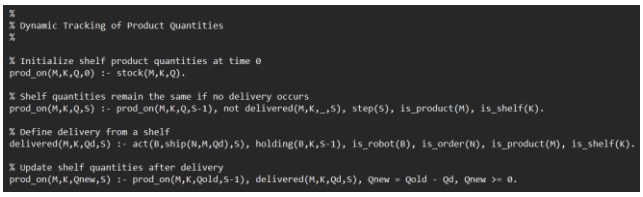

Fig 3: Dynamic Tracking of Product Quantities

Defining the Goal:

A final constraint was added to guarantee that, at the maximum allowed time step, no undelivered products remained for any order. This forced the ASP solver to produce only plans that fully completed all required deliveries.

Optimization:

To improve the efficiency of the generated plans, I used a #minimize directive to minimize the total number of action steps across all robots. This directed the solver to prefer shorter and more efficient plans. I used Clingo's optimization mode (--opt-mode=opt) during execution to obtain the best solution in terms of makespan.

Main Results and Analysis

After completing the ASP encoding, I tested the program using the provided simpleInstances, beginning with inst1.asp. I ran the program using Clingo version 5.7.1 and executed the solver in optimization mode with the command:

clingo warehouse.lp inst1.asp --opt-mode=opt

This directed Clingo to find not just any valid plan, but an optimized plan that minimized the number of action steps required to fulfill all orders.

The solver successfully generated a valid and optimized plan, as shown in the output screenshot (Figure 4). The solver confirmed that:

- All product deliveries were completed correctly,
- No robot collisions or illegal actions occurred,
- The plan achieved an optimization value of 10, meaning that all deliveries were accomplished within 10 action steps,
- The "OPTIMUM FOUND" message indicated that the best possible solution was found,
- Clingo explored 1223 models during its search and completed the optimization process in under one second.

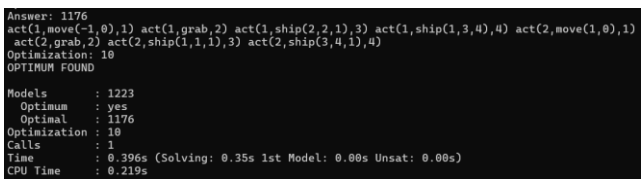

Fig 4: Output of inst1.asp with optimization.

Output Analysis:

The action sequence produced (excerpted below) clearly shows robots moving, picking up shelves, and delivering products efficiently:

act(1,move(1,0),1)
act(1,grab,2)
act(1,ship(2,2,1),3)
act(1,ship(1,3,4),4)
act(2,move(1,0),1)
act(2,grab,2)
act(2,ship(1,1,1),3)
act(2,ship(3,4,1),4)

- Robot 1 and Robot 2 both started by moving toward shelves,
- Both robots picked up shelves,
- Deliveries were executed correctly across subsequent steps,
- The plan effectively leveraged multiple robots working in parallel to reduce overall time.

The results demonstrated that the ASP encoding correctly modeled the dynamic aspects of the warehouse environment, properly enforced all hard constraints, and achieved efficient coordination among robots to fulfill orders.

Conclusion

This project provided an excellent opportunity to apply knowledge representation techniques to a real-world inspired planning problem. By using Answer Set Programming (ASP), I was able to model the dynamic and complex environment of an automated warehouse, where multiple robots coordinate to fulfill orders while adhering to a variety of movement and interaction constraints.

Through careful modeling of robot actions, action effects, state transitions, and hard constraints, the final ASP encoding successfully generated valid and optimized plans across the provided instances. The use of Clingo's optimization features allowed me to produce efficient action sequences that minimized the overall number of steps needed to complete all deliveries.

One of the most challenging aspects of the project was ensuring that all dynamic interactions — particularly the handling of shelves during pickup, carrying, and putdown actions — were accurately reflected over time. Managing multiple robots without collisions or deadlocks also required precise constraint design. However, the structured and modular approach I took allowed me to isolate and solve these challenges step-by-step.

Overall, I am satisfied with the outcome of the project. The final solution was able to handle the complexity of the domain while maintaining clarity and extensibility in the

encoding. This project deepened my understanding of dynamic domain modeling in ASP and strengthened my skills in logical thinking, constraint formulation, and automated planning.

Opportunities for Future Work

While the current solution successfully models and solves the basic warehouse automation scenario, there are several directions for future improvements and extensions:

- Scaling to Larger and More Complex Warehouses: The current model handles relatively small grids with a limited number of robots and shelves. Future work could focus on scaling the solution to handle larger warehouses with hundreds of shelves, multiple picking stations, and higher robot densities. This would require additional optimization strategies to maintain solver performance.
- Incorporating Dynamic Order Arrivals: In real-world warehouses, orders often arrive continuously rather than all being known in advance. Extending the model to handle dynamic, on-the-fly order arrivals would add realism and complexity to the planning problem.
- Minimizing Total Travel Distance or Energy Usage: In addition to minimizing makespan (time steps), it could be valuable to optimize for other metrics such as the total distance traveled by robots or overall energy consumption. This would involve adjusting the optimization criteria and potentially introducing weighted cost functions.
- Handling Robot Failures or Shelf Blockages: Realistic warehouse environments must be robust to unexpected events like robot breakdowns or shelves becoming inaccessible. Adding fault-tolerant planning capabilities would make the model more practical for real-world deployment.
- Developing a Visualization Tool: Building a simple visualization of robot movements and deliveries based on the output action sequence would greatly enhance the ability to debug and understand generated plans, especially for larger instances.

These extensions would deepen the complexity of the problem and provide richer opportunities for applying advanced knowledge representation and reasoning techniques.