Automated Warehouse
CSE 579: Project Milestone 4

Sarthak Swetang Shah
School of Computing and Augmented Intelligence, Arizona State University, Tempe
ssshah45@asu.edu

Problem Statement

In an automated storage facility, the main task of robots is to transport items to specific stations for order fulfillment. The facility is organized as a rectangular matrix, where robots navigate by moving to cells either vertically or horizontally adjacent to them. These robots are assigned to carry shelves that contain required products to the designated stations. The robots are designed with a flat structure, allowing them to slide underneath the shelves to lift and move them. However, if a robot transporting a shelf encounters another shelf blocking its path, the blocking shelf must be moved first. The overarching objective is to fulfill all orders in the least amount of time, where time is counted in steps. Each robot is capable of performing one action in each step.

Project Background

Warehousing plays a crucial role in contemporary supply chain management, offering an infrastructure for businesses to extract, clean, and store extensive data. This data is crucial for powering automation through knowledge-driven algorithms. The primary aim of a data warehouse is to equip knowledge workers with factual information, enabling informed decision-making. Despite the significant amount of information stored on computers, a substantial portion of a company's intellectual assets resides in the expertise of its employees. E-commerce giants like Amazon, Target, Best Buy, and Walmart have established large warehousing infrastructures. These warehouses, which are either fully or partially automated, are designed for cost-effective and timely order processing, as well as enhancing service quality. The NetSuite's article on Warehouse Automation [7] gave me some useful insights which helped me in my project. It includes the process and benefits of automating inventory movement, types of warehouse automation (basic to advanced), and the role of technologies like robotics, sensors, and warehouse management systems. My solution proposes the use of robots to manage order fulfillment in warehouses, with specific attention to avoiding collisions.

For this project, I chose Answer Set Programming (ASP) because it offers a solid, knowledge-based framework for addressing this problem. ASP allows for the creation of constraints that minimize collision risks, something that is more challenging in traditional programming languages. The use of ASP is particularly advantageous over conventional methods as it decreases the likelihood of logical conflicts, reduces the number of required algorithms, and mitigates time-space-compute challenges common in traditional approaches. I utilized CLINGO as the coding platform for ASP.

Approach to Solve the Problem

In initiating this project, I first installed and configured CLINGO on my local machine. Subsequently, I delved into the basics of Answer Set Programming (ASP) and learned how to convert ASP facts into CLINGO's syntax. My strategy involved dissecting the problem into smaller, manageable segments for easier code debugging and solution verification. The solution encompasses five key areas:

- Goal State: This defines the completion of the scenario, where all items in the order must be delivered, indicated by a zero-item count.
- Object Declarations: These rules establish the environment or domain, specifying elements like robot locations, picking stations, and order definitions.
- Fluents: These are supplementary rules that assist the object domain.
- Law of Inertia: These constraints prevent scenarios like two robots occupying the same space simultaneously or switching positions in a single time step.
- Robot Actions: These rules outline various actions a robot can perform, including picking up items, moving in different directions, or remaining idle.

The first significant step in the project involved creating a warehouse layout as a grid to serve as the operational environment. The grid features different elements: red cells as highway cells, yellow cells as picking stations, green cubes representing robots, blue cylinders as shelves, and blue cells as neutral spaces.

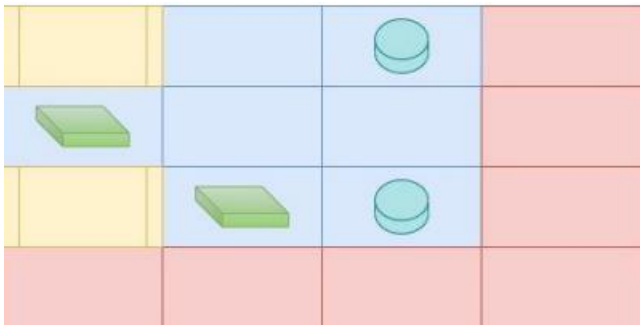

Fig. 1: Sample Warehouse Environment

After establishing the grid, I encoded the properties of the shelves, picking stations, and robots. This included positioning picking stations and designating highway cells exclusive to robot movement. Initial positions for robots and shelves were hardcoded, along with product placements on shelves. Sample orders were then created for testing the algorithm's effectiveness in product retrieval and delivery.

The subsequent phase involved coding the environmental constraints, a complex task akin to solving an unsatisfiability loop. I applied Behavior Driven Development (BDD) methodologies, conceptualizing robot behavior into simple rules, translating these into ASP, and troubleshooting in CLINGO. Key constraints included preventing robots from occupying the same cell, moving outside the grid, swapping positions, or carrying more than one shelf at a time, and ensuring shelves are delivered correctly.

Ultimately, the goal was to ensure robots could efficiently deliver the right products to the appropriate picking stations. This required generating step-by-step instructions for the robots, optimizing each step until the most efficient solution was achieved. This approach enabled the development of a robot capable of moving between adjacent cells, picking up shelves, delivering them to the correct stations, and avoiding collisions.

Results and Analysis

For testing my code, I used five different test cases: inst1.asp, inst2.asp, inst3.asp, inst4.asp, and inst5.asp. My testing revealed an interesting point: if the code doesn't go through a sufficient number of steps, the outcome is unsatisfiable. This means there's a minimum number of steps necessary to get a good result, which is why I ended up writing a Python script to automate the process. Finding the smallest number of steps that still gives a satisfactory result is the key here. Fortunately, this approach worked well, and I was able to get the desired results for all five test instances, small portions of which are as follows:

```
Answer: 1
occurs(object(robot,1),move(-1,0),0) occurs(objec
t(robot,1),move(-1,0),1) occurs(object(robot,2),m
ove(0,-1),1) occurs(object(robot,2),move(1,0),2)
occurs(object(robot,1),move(-1,0),3) occurs(objec
t(robot,2),move(0,1),5) occurs(object(robot,1),mo
ve(0,-1),6) occurs(object(robot,2),move(0,-1),7)
occurs(object(robot,1),move(0,1),8) occurs(object
(robot,2),pickup,0) occurs(object(robot,1),pickup
,2) occurs(object(robot,2),pickup,6) occurs(objec
t(robot,1),pickup,7) occurs(object(robot,2),putdo
wn,4) occurs(object(robot,1),putdown,5) occurs(ob
ject(robot,2),deliver(2,2,1),3) occurs(object(rob
ot,1),deliver(1,1,1),4) occurs(object(robot,2),de
liver(3,4,1),8) occurs(object(robot,1),deliver(1,
3,4),9) timeTaken(9) numActions(19)
Optimization: 64
OPTIMUM FOUND

Models       : 1
  Optimum    : yes
Optimization : 64
Calls        : 1
Time         : 0.363s (Solving: 0.24s 1st Model:
0.03s Unsat: 0.20s)
CPU Time     : 0.281s
```

Fig. 2: Optimal Solution of Instance 1

```
Answer: 3
occurs(object(robot,1),move(-1,0),0) occurs(objec
t(robot,1),move(-1,0),1) occurs(object(robot,2),m
ove(1,0),1) occurs(object(robot,2),move(0,-1),2)
occurs(object(robot,1),move(-1,0),3) occurs(objec
t(robot,1),move(1,0),5) occurs(object(robot,2),mo
ve(0,1),5) occurs(object(robot,2),move(-1,0),7) o
ccurs(object(robot,2),move(-1,0),8) occurs(object
(robot,2),move(0,1),9) occurs(object(robot,2),pic
kup,0) occurs(object(robot,1),pickup,2) occurs(ob
ject(robot,2),pickup,6) occurs(object(robot,2),pu
tdown,4) occurs(object(robot,2),deliver(2,2,1),3)
 occurs(object(robot,1),deliver(1,1,1),4) occurs(
object(robot,2),deliver(1,3,2),10) timeTaken(10)
numActions(17)
Optimization: 72
OPTIMUM FOUND

Models       : 3
  Optimum    : yes
Optimization : 72
Calls        : 1
Time         : 0.598s (Solving: 0.50s 1st Model:
0.03s Unsat: 0.33s)
CPU Time     : 0.672s
```

Fig. 3: Optimal Solution of Instance 2

```
Answer: 21
occurs(object(robot,1),move(0,-1),0) occurs(objec
t(robot,1),move(-1,0),1) occurs(object(robot,1),m
ove(0,-1),3) occurs(object(robot,2),move(0,-1),3)
 occurs(object(robot,1),move(0,1),5) occurs(objec
t(robot,2),move(1,0),5) occurs(object(robot,2),pi
ckup,1) occurs(object(robot,1),pickup,2) occurs(o
bject(robot,1),deliver(2,4,1),4) occurs(object(ro
bot,2),deliver(1,2,1),6) timeTaken(6) numActions(
10)
Optimization: 31
OPTIMUM FOUND

Models       : 21
  Optimum    : yes
Optimization : 31
Calls        : 1
Time         : 0.189s (Solving: 0.14s 1st Model:
0.02s Unsat: 0.09s)
CPU Time     : 0.141s
```

Fig. 4: Optimal Solution of Instance 3

```
Answer: 6
occurs(object(robot,1),move(-1,0),0) occurs(objec
t(robot,1),move(-1,0),1) occurs(object(robot,2),m
ove(1,0),1) occurs(object(robot,2),move(0,-1),2)
occurs(object(robot,1),move(-1,0),3) occurs(objec
t(robot,2),pickup,0) occurs(object(robot,1),picku
p,2) occurs(object(robot,2),deliver(2,2,1),3) occ
urs(object(robot,1),deliver(1,1,1),4) occurs(obje
ct(robot,2),deliver(3,2,2),4) timeTaken(4) numAct
ions(10)
Optimization: 20
OPTIMUM FOUND

Models       : 6
  Optimum    : yes
Optimization : 20
Calls        : 1
Time         : 0.268s (Solving: 0.17s 1st Model:
0.02s Unsat: 0.09s)
CPU Time     : 0.234s
```

Fig. 5: Optimal Solution of Instance 4

After running the CLINGO program, various potential actions that the robot can undertake to fulfill an order and deliver products are revealed. Figure 6 below displays a segment of the output for inst5.asp, as the entire result is too extensive to be fully presented in this format. The solution can be divided into the individual activities of each robot across distinct time intervals.

```
Answer: 17
occurs(object(robot,1),move(-1,0),0) occurs(objec
t(robot,1),move(-1,0),1) occurs(object(robot,1),m
ove(-1,0),3) occurs(object(robot,2),move(-1,0),3)
 occurs(object(robot,1),move(1,0),5) occurs(objec
t(robot,2),move(0,1),5) occurs(object(robot,1),pi
ckup,2) occurs(object(robot,2),pickup,4) occurs(o
bject(robot,1),deliver(1,1,1),4) occurs(object(ro
bot,2),deliver(1,3,4),6) timeTaken(6) numActions(
10)
Optimization: 31
OPTIMUM FOUND

Models       : 17
  Optimum    : yes
Optimization : 31
Calls        : 1
Time         : 0.299s (Solving: 0.22s 1st Model:
0.01s Unsat: 0.01s)
CPU Time     : 0.297s
```

Fig. 6: Optimal Solution of Instance 5

occurs(object(robot,1),move(-1,0),0)
occurs(object(robot,1),move(-1,0),1)
occurs(object(robot,1),pickup,2)
occurs(object(robot,1),move(-1,0),3)
occurs(object(robot,1),deliver(1,1,1),4)
occurs(object(robot,1),move(1,0),5)

Fig. 7: Robot 1 Actions

occurs(object(robot,2),move(-1,0),3)
occurs(object(robot,2),pickup,4)
occurs(object(robot,2),move(0,1),5)
occurs(object(robot,2),deliver(1,3,4),6)

Fig. 8: Robot 2 Actions

Figure 7 illustrates the sequential actions of Robot 1 over various time intervals. Initially, at time step 0, it ascends by one cell. Subsequently, at time step 1, it continues its upward movement, arriving at the designated shelf for collection. During time step 2, it successfully retrieves the shelf located within its cell. In the following action at time step 3, the robot advances further up by one cell, reaching the collection station. At time step 4, it accomplishes the task of handing over the collected shelf to the collection station. Finally, at time step 5, Robot 1 descends a cell, vacating the collection station and thus facilitating collision-free deliveries by other robots.

Figure 8 illustrates the sequence of activities undertaken by Robot 2 across various time intervals. Initially, at time

steps 1 and 2, the robot remains stationary, undertaking no actions. In the subsequent time step 3, the robot ascends one cell, arriving at the designated shelf for retrieval. During time step 4, it executes the task of collecting the shelf. Following this, at time step 5, the robot navigates one cell towards the right, arriving at the designated pick-up station. Finally, at time step 6, it accomplishes the delivery of the shelf to the pick-up station and stays put, as all the designated orders have been fulfilled.

This analysis indicates that the CPU time required can significantly fluctuate based on the number of iterations in a particular scenario. The solution is designed to determine the most efficient number of steps for each case. Consequently, it's apparent that the solution effectively fulfills the project's specifications.

Conclusion

The introduction of automated robots has revolutionized warehouse management, moving away from traditional manual methods. Technologies like Knowledge Representation, Answer Set Programming (ASP), and CLINGO have significantly simplified the automation process. ASP stands out for its brevity and efficiency, enabling programmers to concentrate on problem-solving rather than extensive algorithm development and coding.

Initially, mastering ASP and CLINGO presented a learning curve, but they proved to be straightforward and engaging once I grasped their concepts. My primary focus was on decomposing the project into manageable components, ensuring each met the constraints without any rule violations. Setting up the initial warehouse environment was time-consuming. I began by employing a trial-and-error approach to resolve logic and coding issues. However, I soon realized that a more thoughtful strategy, focusing on the robot's behavior and applying constraints incrementally, led to smarter coding solutions.

A significant challenge was managing the collision constraints for the robots, preventing them from occupying the same space. Extensive research and additional constraints integrated into my existing framework helped me overcome this hurdle. A crucial lesson from this project was the importance of commenting on code. In ASP, even minor mistakes can cause 'soft failures,' where the code runs without errors but fails to yield correct results. To avoid lengthy debugging sessions, I learned to test the code methodically after each modification.

This project not only enhanced my understanding of Knowledge Representation but also provided valuable insights into its application across various domains.

Opportunities for Future Work

The current methods of automating warehouses can be enhanced by enabling robots to handle several orders at once, boosting time efficiency and effectiveness. Warehouses serve as a prime example to illustrate the significant role of ASP (Answer Set Programming) and KRR (Knowledge Representation and Reasoning) in addressing intricate real-world challenges. As these problems grow in complexity, traditional computer science methodologies become increasingly inadequate. It is in these scenarios that logic-based functional programming emerges as a key solution. The narrative focuses on the constraints and how KRR adeptly manages these complex real-world constraints.

There is a broad scope for employing ASP in the realm of AI, Robotics, Computational Biology and Bioinformatics, Workforce management, Intelligent Call Routing for contact centers, and Decision Support Centers. ASP's pivotal role in tackling multifaceted issues across various domains is due to its capacity for recursive definitions, aggregates, weight constraints, optimization statements, default negations, and external atoms. Such built-in expressiveness enables ASP to be utilized in a declarative manner, addressing both combinatorial search challenges (like planning and diagnosis) and knowledge-intensive tasks (such as query answering and explanation generation).

References

[1] CSE 579: Lecture Videos and Slides by Dr. Joohyung Lee

[2] Automated Warehouse Scenario description pdf

[3] *Answer Set Programming*, Vladimir Lifschitz (2019)

[4] *Answer Set Solving in Practice: Synthesis Lectures on Artificial Intelligence and Machine Learning*, Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub (2012)

[5] Hamberg, R. & Verriet, J. eds., 2012. Automation in Warehouse Development. Available at: http://dx.doi.org/10.1007/978-0-85729-968-0.

[6] Guo, W.L. & Huang, Z.J., 2014. Analysis and Research of the Automated Warehouse Management and Control System. *Applied Mechanics and Materials*, 511-512, pp.1095–1098. Available at: http://dx.doi.org/10.4028/www.scientific.net/amm.511-512.1095.

[7] https://www.netsuite.com/portal/resource/articles/inventory-management/warehouse-automation.shtml

Acknowledgments

I would like to thank Dr. Samira Ghayekhloo for providing this opportunity of working on this ASP Challenge 2019 project and Dr. Joohyung Lee for making these awesome video lectures and slides.