CSE 579
Automated Warehouse Scenario Problem
Individual Project

Jahnavi Sri Sai Lavu
Arizona State University
jlavu@asu.edu

***Abstract*—Imagine a big building filled with shelves and boxes, where robots zoom around carrying things. AI is used, which is like a super-smart brain, to figure out the best way to move things around. The AI helps them decide which box goes where, so everything is organized neatly and easy to find. This helps save time and makes sure that when someone needs something, the robots can quickly bring it to them. In this project, a simple ware-house scenario is solved using concepts of Knowledge Representation and Reasoning.**

I. Introduction

Imagine a big building filled with shelves and boxes, where robots zoom around carrying things. For this, AI is used, which is like a super-smart brain, to figure out the best way to move things around. The AI helps them decide which box goes where, so everything is organized neatly and easy to find. This helps save time and makes sure that when someone needs something, the robots can quickly bring it to them. In this project, a simple warehouse scenario is solved using concepts of Knowledge Representation and Reasoning.

In general, a warehouse is like a big storage space, usually a building that companies use to store their products and goods until it is time to send them off to stores or customers. In this specific project, a simplified version of warehouse is defined that is supposed to be automated using Answer Set Programming (ASP) that is learned in this course. The automation should be done such that all the orders are fulfilled in the least amount of time possible, where time is counted in steps and each robot may perform one action at max per timestamp. Few rules are mentioned that act as constraints to the actions of the robots. The task is to write full code using Answer Set Programming (ASP) which guides the actions of the robots in delivering the goods without any collisions in between the robots an run the code using Clingo.

In the warehouse grid system, the movement of robots is constrained to horizontal and vertical pathways between neighboring cells. The primary objective is to efficiently deliver shelves holding specific products to designated picking stations to fulfill orders. These agile robots possess the capability to navigate beneath shelves and lift them when necessary. However, an intriguing challenge arises when a robot carrying a shelf occupies the same space as another robot. This situation results in a collision, potentially disrupting the smooth workflow. Thus, the optimal solution involves orchestrating robot movements in a manner that ensures shelves are strategically relocated to prevent obstructions or clashes. My task is to implement an algorithm or strategy that carefully coordinates the robots' paths and shelf movements, collisions can be effectively avoided, enabling seamless and error-free operations within the warehouse within least amount of time or steps possible.

A. Project Background

The intricate web of the automated warehousing system revolves around the intricate orchestration of robotic navigation to ensure the seamless transportation of products from storage racks to designated picking stations. Within this complex puzzle, the warehouse floor plan intricately divides into numerous cells, each housing distinct zones for picking stations and delineated pathways, termed highways, where shelving units find no placement.

At the heart of this challenge lies the meticulous development of an operational blueprint for every robotic unit, a plan geared toward flawlessly ferrying the required products to their designated pickup stations. However, to surmount this obstacle, the warehouse blueprint demands meticulous configuration, meticulously outlining the positions of robots, shelves, picking stations, and highway pathways.

Traversing the path toward computational resolution, a plethora of constraints and limitations emerge as crucial checkpoints. Ensuring the unique placement of robots and shelves becomes paramount, accommodating scenarios where shelves may house identical products. The understanding that robots can ferry only a single shelf at any given time further complicates the logistical framework. Additionally, the imperative of preventing robots from traversing under shelves while already bearing another shelf amplifies the intricacies of the problem.

Further constraints abound as robots navigate solely in horizontal or vertical trajectories between adjacent cells, restricted to executing a solitary activity at any given instance. The crux of resolving this labyrinthine logistical challenge within the warehouse lies in deftly synchronizing these multifaceted components in perfect harmony, ensuring a flawless orchestration of operations in this bustling automated environment.

II. Explanation of the solution

A. Approach Followed

Initially, my focus was on thoroughly grasping the problem statement, ensuring a comprehensive understanding of the warehouse scenario and the anticipated project outcomes. To devise an effective strategy, I aimed to familiarize myself with the workings of Answer Set Programming (ASP), recognizing its pivotal role in solving the task at hand. This involved delving into the nuances of ASP, comprehending both its hard and weak constraints. The journey toward coding in Clingo began with an in-depth exploration of its concepts and syntax. This involved an extensive study of Clingo's documentation, enabling a thorough grasp of its terminology and programming structures. I solidified my understanding of Answer Set Programming in Clingo by tackling simpler example problems, facilitating a more practical comprehension of its application. In anticipation of potential challenges, I diligently cataloged all conceivable edge cases that might pose issues when integrating constraints. With each identified rule, I meticulously updated the code to ensure its adaptability to all potential edge cases, fortifying its resilience and functionality. This process of addressing edge cases proved instrumental in refining the code to handle diverse scenarios, ultimately enhancing its robustness and versatility.

I delved into the essence of the challenge by initially visualizing the system's operations outlined in the initialization files on paper. This exercise aimed to comprehend the myriad actions and constraints governing the problem. Concurrently, I scrutinized the initialization files to grasp the input intricacies and simplify them for more manageable implementation. My approach followed an iterative methodology, enabling a comprehensive understanding of the problem's intricacies. Initially, starting without any constraints, I executed robot movements within the framework, targeting a specific node as the goal state. Subsequently, I progressively introduced constraints and actions associated with all elements in the system. This systematic approach proved invaluable for troubleshooting the code and devising optimal solutions for each scenario. Initially, grappling with constraints posed considerable challenges, and debugging the codes emerged as a formidable task. I also encouraged diverse thinking by exploring individual ideas during the project's early phases. This approach fostered an openness to different problem-solving methods, enriching the array of potential solutions and fostering constructive debates to validate their efficacy. Additionally, I maintained a disciplined schedule of self-check-ins and reflective sessions to ensure progress and address any stumbling blocks encountered along the way. I began by immersing myself in the core of the challenge, initially mapping out the system's operations from the detailed initialization files onto paper. This strategic visualization aimed to grasp the intricate web of actions and constraints governing the problem. Simultaneously, I meticulously analyzed the initialization files, unraveling their intricacies to simplify inputs for smoother implementation. My iterative approach laid the groundwork for a comprehensive understanding of the problem's nuances.

Commencing without constraints, I initiated robot movements within the framework, targeting a specific node as the desired endpoint. Gradually, I introduced constraints and actions tied to each element in the system. This systematic progression proved invaluable, allowing for code troubleshooting and the crafting of optimal solutions for various scenarios. However, grappling with constraints initially posed significant challenges, elevating code debugging to a formidable task.

Encouraging diverse thinking was a cornerstone of my strategy, fostering an environment where individual ideas flourished during the project's initial phases. This approach not only encouraged different problem-solving methods but also enriched the pool of potential solutions, sparking constructive debates to validate their effectiveness.

To maintain momentum and ensure steady progress, I adhered to a disciplined routine of self-check-ins and reflective sessions. These deliberate moments allowed for course correction and resolution of any stumbling blocks encountered along the project's intricate path, ensuring a steadfast march towards resolving the challenge at hand.

B. Technical point of view of the solution

The entire project is first divided into tasks and these milestones are solved one by one. First task is to understand the problem statement clearly and making note of all the actions and constraints that are required. Second task to is to make a clear outline about the output of the project. Third task is to understand Answer Set Programming and how to run it using Clingo. Fourth task is to gain some experience with Answer Set Programming and Clingo by solving few simple problems in previous assignments. Fifth task is to clearly noting down all the hard and weak constraints in the problem statement. Sixth task is to note down all the edge constraints and making sure that the code works for all the edge constraints as well. Once all of this is done, then the project can be updated by further adding few constraints that make this scenario more close to that of real world. I have gone thoroughly through the Clingo documentation. I have studied the problems given in the assignments like sudoku, monkey an banana problem, block world problem to have better idea about the approach and steps to be followed. Once I understood the problem statement from a pictorial depiction which I had drawn on paper for clear picture of the scenario, I started writing code without any constraints. Initially, I considered that all the robots can be anywhere in the grid. Later, I added the hard and weak constraints one by one. The constraints that I have added so far are:

- Robots can stay idle.
- A robot can perform only one action at a time.
- In a time unit, the robot can move only one step either vertically or horizontally.
- A robot may perform an action such as taking up or lowering a shelf. A robot can only carry one shelf at a time.

- Robots can only move under shelves if they are not carrying any shelves.
- Two items cannot occupy the same space at the same time.

The edge cases became clearer when I added these conditions one by one, and they were solved accordingly. The optimal solution is obtained by reducing the number of actions.

III. Results description

For the purpose of validating the program's functionality across the range of scenarios outlined in the project, instances were kindly provided. Our code implementation consistently and successfully produces the desired outcomes in accordance with the described approach through rigorous testing and execution. This thorough testing procedure confirmed our code's effectiveness and confirmed that it can provide the intended results in a variety of situations and scenarios that fall under the project's parameters.

The algorithm generated the following results when it is run with instance 1.

```
==========================================
action(object (robot, 1) , move (-1,0) ,0) action(object (robot, 1), move (-1,0), 1) action(object (robot, 2), move (0, -1), 1) action (object(robot, 2) , move (1,0) ,2) action(object (robot, 1) , move (-1,0) ,3) action(object (robot, 2) , move (0,1) ,5) action(object (robot, 1) , move (0, -1 ) ,6) action (object (robot, 2) , move (0, -1) ,7) action(object (robot, 1) , move (0,1) ,8) action(object (robot,2) ,pickup,0) action(object(robot, 1) ,pickup, 2) action(object (robot, 2) ,pickup,6) action(object (robot, 1) ,pickup, 7) action(object (robot, 2) ,putdown,4) action(object (robot, 1) , putdown, 5) action(object (robot, 2), deliver (2,2,1),3) action(object (robot, 1), deliver (1,1,1) ,4) action(object(robot,2), deliver (3,4,1) ,8) action(object (robot, 1), deliver (1,3,4), 9) numActions (19)
Optimization: 64
OPTIMUM FOUND
Models : 1
Optimum : yes
Optimization : 64
Calls : 1
Time :0.454s (Solving: 0.31s 1st Model: 0.05s Unsat: 0.27s)
CPU Time : 0.422s
==========================================
```

The algorithm generated the following results when it is run with instance 2.

```
==========================================
Optimization: 73
Answer: 3
action(object (robot, 1), move (-1, 0) ,0) action (object (robot, 1), move (-1,0) ,1) action(object (robot, 2) , move (1,0) ,1) action(object(robot, 2) , move (0, -1) ,2) action(object (robot, 1) , move (-1,0) ,3) action(object (robot, 1), move (1,0) ,5) action(object (robot, 2) , move (0, 1),5) action (object (robot, 2) , move (-1,0) ,7) action(object (robot, 2) , move (-1, 0) ,8) action(object (robot, 2), move (0,1), 9) action(object (robot, 2) ,pickup,0) action(object (robot, 1) ,pickup,2) action(object (robot, 2) ,pickup,6) action(object (robot, 2) , putdown,4) action(object (robot, 2), deliver (2,2,1),3) action(object(robot, 1), deliver (1,1, 1),4) action(object (robot, 2), deliver (1,3,2), 10) numActions (17)
Optimization: 72
OPTIMUM FOUND
Models : 3
Optimum : yes
Optimization : 72
Calls : 1
Time : 0.719s (Solving: 0.61s 1st Model: 0.05s Unsat: 0.41s)
CPU Time : 0.719s
==========================================
```

The algorithm generated the following results when it is run with instance 3.

```
==========================================
Optimization: 32
Answer: 21
action (object (robot, 1), move (0, -1), 0) action (object (robot, 1), move (-1, 0), 1) action(object (robot, 1), move (0, -1) ,3) action(object(robot, 2), move (0, -1), 3) action (object(robot, 1), move (0,1) ,5) action(object (robot, 2), move (1,0),5) action(object (robot, 2) ,pickup, 1) action(object (robot,1), pickup,2) action(object(robot, 1), deliver (2,4, 1) ,4) action(object (robot 2), deliver (1,2,1) , 6) numActions(10)
Optimization: 31
OPTIMUM FOUND
Models : 21
Optimum : yes
Optimization :31
Calls : 1
Time : 0.344s (Solving: 0.28s 1st Model: 0.02s Unsat: 0.11s)
CPU Time :0.250s
==========================================
```

The algorithm generated the following results when it is run with instance 4.

```
==========================================
Answer: 6
action (object (robot, 1), move (-1,0) ,0) action (object (robot, 1), move (-1,0), 1) action(object (robot, 2) , move (1,0) ,1) action (object (ro bot, 2), move (0, -1) ,2) action(object (robot, 1), move (-1,0) ,3) action(object (robot, 2) ,pickup, 0) action(object (robot, 1) ,pickup, 2) action (object (robot, 2), deliver (2,2,1),3) action(object (robot, 1), deliver (1,1,1),4)
```

action(object (robot, 2), deliver (3, 2, 2), 4) numActions (10)
Optimization: 20
OPTIMUM FOUND
Models : 6
Optimum : yes
Optimization : 20
Calls : 1
Time : 0.406s (Solving: 0.30s 1st Model: 0.02s Unsat: 0.14s)
CPU Time : 0.375s
==========

The algorithm generated the following results when it is run with instance 5.

==========
Answer: 17
action (object (robot, 1), move (-1,0) ,0) action(object (robot, 1), move(-1, 0), 1) action(object (robot, 1), move(-1,0),3) action(object (robot, 2), move (-1, 0), 3) action (object (robot, 1), move (1,0),5) action (object (robot, 2), move (0,1),5) action (object (robot, 1), pickup, 2) action(object (robot,2),pickup,4) action(object(robot, 1), deliver (1,1,1),4) action(object (robot, 2), deliver (1,3,4), 6) numActions(10)
Optimization: 31
OPTIMUM FOUND
Models : 17
Optimum : yes
Optimization : 31
Calls : 1
Time : 0.522s (Solving: 0.43s 1st Model: 0.02s Unsat: 0.03s)
CPU Time : 0.422s
==========

IV. Conclusion

A. Individual Project

I have done this project all by myself and this is an individual project in the Knowledge Representation and Reasoning course (CSE 579).

B. Skills and Self-Assessment

This project provided me with numerous insights, particularly in the realm of Answer Set Programming (ASP). Taking on the responsibility of overseeing every facet of the project allowed me to work meticulously, ensuring scrutiny at each stage. Immersed in the realm of propositional logic for this endeavor, I gained a profound understanding of ASP's nuances, encompassing both fundamental concepts and intricate ASP ideas. The practical application of answer set languages in real-world scenarios involving autonomous robots underscored their effectiveness and relevance. Navigating the project's complexity entailed constant adjustments to multiple constraints. Occasionally, updates to rules would unearth unexpected situations that required thorough debugging. Ensuring the code's reliability involved comprehensive testing across all scenarios outlined in the project files, systematically addressing issues that emerged with each constraint addition. Initially, starting with a minimal number of robots and gradually increasing their count revealed an intriguing relationship between robot quantity and parcel delivery time. However, scaling up without proper restrictions led to collision incidents, presenting a new challenge. Balancing delivery speed while averting collisions required meticulous testing of varying robot counts across all project scenarios, culminating in the identification of the most optimized configuration. Upon completion, my script yielded superlative outcomes, achieving optimal results with the fewest steps across all provided instances. The robots adeptly traversed the workspace, executing tasks flawlessly by sidestepping collisions and swiftly executing instructions, culminating the project's development phase on a high note. The project's applications transcend various industries involved in order management and product warehousing. The efficiency and reliability of robotic technologies outshine human labor in order handling, making them the preferred choice for streamlined operations in these sectors. The seamless integration of automated processes elevates efficiency, reliability, and speed, making robotic technologies indispensable in modern warehouse management paradigms. Robot programming substantially increases the adaptability of these automated processes—which include storage, retrieval, relocation, and delivery—and enables the execution of intricate tasks and quick order fulfillment in certain situations. Strong environments like this one have many benefits and represent a forward-thinking method of solving problems, which is why they support the broad deployment of robots not just in warehouses but in many other industries as well.

References

[1] P. Obermeier, "Scalable Robotic Intra-Logistics with Answer Set Programming," in OASICS, 2018.

[2] G. Martin, K. Roland, K. Benjamin, O. Max, S. Torsten, and T. Sven, "A User's Guide to gringo, clasp, clingo, and iclingo," University of Potsdam, 2010.

[3] C. Drescher, M. Gebser, T. Grote, B. Kaufmann, A. König, M. Ostrowski, and T. Schaub, "Conflict-driven disjunctive answer set solving," in Proceedings of the Eleventh International Conference on Principles of Knowledge Representation and Reasoning (KR'08), G. Brewka and J. Lang, Eds. AAAI Press, 2008, pp. 422–432.

[4] C. Drescher, M. Gebser, T. Grote, B. Kaufmann, A. König, M. Ostrowski, and T. Schaub, "Conflict-driven disjunctive answer set solving," in Proceedings of the Eleventh International Conference on Principles of Knowledge Representation and Reasoning (KR'08), G. Brewka and J. Lang, Eds. AAAI Press, 2008, pp. 422–432.