Achieving Efficient Warehouse Management through Automation and Robotics

Saideepthi Korupolu
Master of Science,Computer Science
Tempe
Arizona
skorupo1@asu.edu

ABSTRACT

This paper discusses the advantages of automation in warehouse management and presents an overview of an automated warehousing solution that involves using robots to transport products from racks to fulfill orders. The warehouse is divided into cells, with designated picking stations for order placement and highways that must remain free of shelves. To achieve this, an ASP program must be developed for a warehouse delivery robot to deliver products to the designated locations. The paper also highlights the various constraints involved in the project, such as the positioning of robots and shelves, restrictions on the movement of robots, and limitations on the number of products that can be stored and transported. The ultimate goal of the project is to optimize the process and ensure that products are delivered in the shortest time possible while still satisfying all constraints. This paper concludes with a summary of the progress made on the project and the tools and resources used to develop a workable solution, including programming languages like ASP and Clingo.

CCS Concepts

• Automated warehousing → Robotics → ASP program → Constraints → Optimization → Clingo → Warehouse management.

Keywords

Automated warehousing; Robots; ASP program; Constraints; Optimization; Clingo; Warehouse Management; Operations.

1. INTRODUCTION

The project aims to develop an automated warehousing solution using robots to transport products from racks to fulfill orders. To achieve this, an ASP program must be developed for a warehouse delivery robot (R) to deliver products from picking stations (PS) to Shelves (S) to the designated locations while satisfying various constraints. The constraints include limitations on the number of shelves that robots can carry, the placement of shelves on highways, and the positioning of robots and shelves, among others. The ultimate goal is to optimize the process and ensure that products are delivered in the shortest time possible while still satisfying all constraints. To develop a workable solution, a solid understanding of programming languages such as ASP and Clingo, as well as warehouse management and operations, is crucial. The project progress includes the completion of the necessary environment setup, implementation of constraints, and troubleshooting of errors that arise during each scenario. The successful completion of this project will result in a streamlined, automated warehousing system that can be used in a variety of industries.

2. PROJECT BACKGROUND

The development of a workable solution for the automated warehouse scenario requires a solid understanding of the necessary background information. This includes a focus on programming languages such as ASP and Clingo, which are essential for the successful implementation of the project. To acquire the necessary knowledge, various online learning resources and course lectures and slides were utilized. Additionally, research was conducted on automated warehouse projects that were written in different programming languages to analyze the algorithms in their source code. The lectures were also used to understand how constraints are employed in ASP programming. Furthermore, research was conducted on warehouse management and operations to expand the knowledge further. All these efforts have helped in comprehending the intricacies of automated warehouses, which is reflected in the project's progress. Overall, the project background provides an overview of the steps taken to acquire the necessary knowledge and expertise to successfully implement the automated warehouse solution.

3. SOLUTION

The solution involves steps from identifying the constraints to setting up the environment and getting an optimized solution.

3.1 Identifying Constraints

Before proceeding with the solution first one must identify the constraints to achieve the goal.

The constraints in the project are:

- The positioning of robots and shelves must be distinct.
- Robots are not capable of constructing shelves.
- Highways are not permitted to have any shelves placed on them.
- Robots can pick up and put down a single shelf but are limited to carrying one shelf at a time.
- The same product may be stored on multiple shelves.
- Two robots must not occupy the same cell simultaneously.
- While transporting a shelf, robots are able to move beneath it, but if a robot is already carrying a shelf, it cannot move underneath another one.
- Robots are restricted to moving horizontally or vertically to adjacent cells.
- The number of units of a specific item included in an order cannot exceed the number of units of that item currently in storage.

3.2 Setting up the Environment

After identifying the constraints we have to setup the environment with installing all the required libraries and software needed for the project and also we have to initialize the required files for the project.

- Install clingo ans set up clingo environment in laptop
- Initialized the programming files
- Created a file for input which is used to convert the input file to readable format
- Initialized Robots(R), Shelves(S) and picking locations (PR) and highways (H) in the python file
- A parser ASP file

3.3 Approach

To start the project, the first step is to set up the Clingo environment. This requires the installation of Python 3.7 or higher as a Python file needs to be run. The order of execution is such that the Python script runs at the end, following the five instance-programs provided, an input converter ASP file, and a parser ASP file. The next step involves coding and implementing the code. There are several constraints that need to be addressed for the given problem, including both severe and subtle restrictions. Once the constraints are determined, it is necessary to generate a file that can translate the input schema into a format that is comprehensible and understandable.

Finally, the last two components involve initializing the robot's location and checking the shelves using distinct positions, as well as constructing pickup points and highways. My approach to achieving a consistent mode is to start by identifying all the tasks that a robot could potentially carry out. Robots have the ability to perform diverse tasks and arrive at the same network location within a particular time frame, following the usual practice.

In terms of implementation, an unrestricted approach was initially adopted, executing the movement of robots within the provided framework with a specific goal state. This approach was followed by the implementation of constraints. The focus was on developing the code that would allow the robots to 'pickup' and 'putdown' the shelves. The goal was to ensure that the final state of this process resulted in the shelf being placed on top of the robot. This approach helped in troubleshooting and correcting any errors that arose during each scenario. Debugging the code was a challenging task, and to make the input file readable, an input convert ASP program was created.

Once a working example was created, the code was tested to ensure that the correct number of stable models were generated. The process was repeated to fulfill the various constraints of the task and generate all the coherent models required to fulfill the objective, which is to convey all the details related to the order.

This approach resulted in a streamlined, automated warehousing system.

3.4 Challenges Encountered

When implementing any approach, it is common to face issues, and the same was the case with this project. There were various challenges and obstacles that had to be overcome.

- One of the constraints was that two robots should not be placed at the same place at the same time. This was a critical constraint that had to be addressed since it could have led to collisions or other unintended consequences
- Another constraint that was encountered was that the sum of units of a product should not exceed the units stored on the shelves. This was a more subtle constraint that required careful consideration and analysis. The issue arose when a robot tried to pick up a shelf that had more units of a particular product than what the robot could carry.
- Edge cases refer to exceptional or unusual scenarios that are not considered during the regular testing process. While working on the project, identifying edge cases was crucial since it allowed to test the limits of the code and ensure that it was robust enough to handle all possible scenarios.
- Finally, defining algorithms to get an optimized solution was a crucial part of the project. An optimized solution meant that the robots completed the task in the shortest time possible while adhering to all constraints

3.5 Overcome the Challenges

I overcome the above challenges as follows:

- For the first challenge, I closely examined the code and identified the specific areas where the robots were initialized. I then introduced a check in the code to ensure that the robots were never initialized at the same location at the same time. This helped in avoiding any potential conflicts between the robots, which could have led to incorrect results or failure to meet the desired objective.
- To address the second challenge, I analyzed the code to check if all the constraints were properly implemented and if there were any logical errors. Upon identifying the issues, I made necessary modifications to the code and tested it for different scenarios to ensure that the constraint was met. This helped me in ensuring that the program was reliable and provided accurate results. that met the capacity criteria.
- Third challenge I tackled by observing my code for all the test cases and keeping track of the scenarios where the code could fail. By doing so, I was able to identify the edge cases and develop a strategy to handle such cases. I also tested the code for these edge cases to ensure that the program was robust and could handle such cases effectively.
- I explored different algorithms and their complexity to determine which algorithm would be the most efficient for the problem. I analyzed the problem and identified the factors that could affect the performance of the algorithm. Based on this analysis, I chose the most optimal algorithm that would provide the best performance for the problem. This helped in achieving an optimal solution that was efficient and reliable.

4. RESULTS

The project was completed successfully, and all the constraints were considered while designing the program. The movement of robots was restricted based on time and the number of requests that needed to be sent to the assigned picking station.The program was able to solve all the tasks with the minimum number of steps required. The code developed was able to identify the optimal solution for each task provided. The robot was able to navigate the

area without colliding with any obstacles, and it was able to complete all orders efficiently within the shortest possible time.

Table 1: Running time and optimal steps of the instance programs

Program	Optimization	CPU time
inst1.asp	13	19.82s
inst2.asp	11	3.07s
inst3.asp	7	0.123s
inst4.asp	10	6.124s
inst5.asp	6	0.063s

The table presented above displays the duration of execution and effectiveness of the 5 program instances developed for the warehouse problem. It is evident that the programs' execution time is minimal, which indicates that we have obtained an optimized solution, as the output is generated as quickly as possible.

5. CONCLUSION

Based on the approach used, challenges faced, and solutions implemented, it can be concluded that the warehouse problem statement was successfully solved with an efficient algorithm. The challenges encountered, such as initializing the constraints, troubleshooting with the sum of units of a product, identifying edge cases, and defining the algorithms to get an optimized solution, were overcome by debugging the code, observing all test cases, and testing various algorithms to achieve the most optimal solution. The implemented constraints were crucial in solving the problem efficiently, and the identified edge cases were taken care of, resulting in a well-rounded solution. The robots were able to navigate the area with ease, complete all orders in the shortest time possible, and not collide with any obstacles. The running time of the code was minimal, indicating that the optimal solution was achieved with efficient programming. Overall, the project was a success, and the solutions provided can be utilized in real-world warehouse automation scenarios to improve efficiency and reduce costs.

6. CONTRIBUTION

For this project, I made significant contributions in various stages of the project. First and foremost, I set up the required environment, including installing the necessary software and tools needed to implement the project. Next, I created the Python files, ensuring that the code was well-organized and easily readable for other team members.

When implementing the code, I took into consideration the various constraints that needed to be followed, such as ensuring that robots did not collide with each other and that the sum of units of a product did not exceed the units stored on the shelves. During this process, I encountered several challenges such as initializing the constraints and troubleshooting issues related to the constraints. However, with persistence and patience, I was able to overcome these challenges through debugging and observing the code for all test cases.

```
(base) saideepthikorupolu@Saideepthi simpleInstances % clingo warehousescenario.lp inst1.asp -c t=13 -t 8

clingo version 5.6.2
Reading from warehousescenario.lp ...
Solving...
Progression : [1;inf]
Progression : [2;inf]
Answer: 1
occurs(object(robot,2),move(-1,0),1) occurs(object(robot,1),move(-1,0),1) occurs(object(robot,2),move(0,1
),3) occurs(object(robot,1),move(-1,0),3) occurs(object(robot,2),move(0,-1),5) occurs(object(robot,1),mov
e(-1,0),5) occurs(object(robot,2),move(1,0),7) occurs(object(robot,1),move(0,-1),8) occurs(object(robot,2
),move(1,0),8) occurs(object(robot,1),move(1,0),9) occurs(object(robot,2),move(0,-1),10) occurs(object(ro
bot,1),move(1,0),11) occurs(object(robot,1),move(0,-1),12) occurs(object(robot,2),move(1,0),12) occurs(ob
ject(robot,2),pickup,2) occurs(object(robot,1),pickup,4) occurs(object(robot,2),putdown,6) occurs(object(
robot,1),putdown,7) occurs(object(robot,2),pickup,9) occurs(object(robot,1),pickup,10) occurs(object(robo
t,2),deliver(1,3,4),4) occurs(object(robot,1),deliver(1,1,1),6) occurs(object(robot,2),deliver(3,4,1),11)
 occurs(object(robot,1),deliver(2,2,1),13)
Optimization: 13
OPTIMUM FOUND

Models       : 1
  Optimum    : yes
Optimization : 13
Calls        : 1
Time         : 16.403s (Solving: 0.52s 1st Model: 0.25s Unsat: 0.27s)
CPU Time     : 19.821s
Threads      : 8       (Winner: 4)
```

Figure 1: inst1.asp result

```
(base) saideepthikorupolu@Saideepthi simpleInstances % clingo warehousescenario.lp inst2.asp -c t=11 -t 8
clingo version 5.6.2
Reading from warehousescenario.lp ...
Solving...
Progression : [1;inf]
Progression : [2;inf]
Answer: 1
occurs(object(robot,2),move(0,1),1) occurs(object(robot,1),move(0,-1),1) occurs(object(robot,1),move(-1,0
),2) occurs(object(robot,2),move(-1,0),3) occurs(object(robot,1),move(1,0),4) occurs(object(robot,2),move
(0,-1),5) occurs(object(robot,1),move(0,1),5) occurs(object(robot,1),move(0,1),6) occurs(object(robot,2),
move(1,0),7) occurs(object(robot,1),move(-1,0),7) occurs(object(robot,1),move(-1,0),8) occurs(object(robo
t,2),move(1,0),9) occurs(object(robot,1),move(0,-1),9) occurs(object(robot,1),move(-1,0),10) occurs(objec
t(robot,2),move(0,-1),10) occurs(object(robot,2),pickup,2) occurs(object(robot,1),pickup,3) occurs(object
(robot,2),putdown,6) occurs(object(robot,2),pickup,8) occurs(object(robot,2),deliver(1,1,1),4) occurs(obj
ect(robot,1),deliver(1,3,2),11) occurs(object(robot,2),deliver(2,2,1),11)
Optimization: 11
OPTIMUM FOUND

Models       : 1
  Optimum    : yes
Optimization : 11
Calls        : 1
Time         : 2.443s (Solving: 0.09s 1st Model: 0.09s Unsat: 0.00s)
CPU Time     : 3.071s
Threads      : 8       (Winner: 4)
```

Figure 2: inst2.asp result

```
(base) saideepthikorupolu@Saideepthi simpleInstances % clingo warehousescenario.lp inst3.asp -c t=7 -t 8
clingo version 5.6.2
Reading from warehousescenario.lp ...
Solving...
Progression : [1;inf]
Progression : [2;inf]
Answer: 1
occurs(object(robot,1),move(-1,0),1) occurs(object(robot,1),move(-1,0),2) occurs(object(robot,2),move(1,0
),2) occurs(object(robot,1),move(0,-1),3) occurs(object(robot,2),move(0,-1),4) occurs(object(robot,1),mov
e(1,0),5) occurs(object(robot,1),move(0,-1),6) occurs(object(robot,2),move(1,0),6) occurs(object(robot,2)
,pickup,3) occurs(object(robot,1),pickup,4) occurs(object(robot,2),deliver(2,4,1),5) occurs(object(robot,
1),deliver(1,2,1),7)
Optimization: 7
Progression : [3;7] (Error: 1.33333)
Progression : [4;7] (Error: 0.75)
Progression : [5;7] (Error: 0.4)
OPTIMUM FOUND

Models       : 1
  Optimum    : yes
Optimization : 7
Calls        : 1
Time         : 0.067s (Solving: 0.01s 1st Model: 0.01s Unsat: 0.00s)
CPU Time     : 0.123s
Threads      : 8       (Winner: 4)
```

Figure 3: inst3.asp result

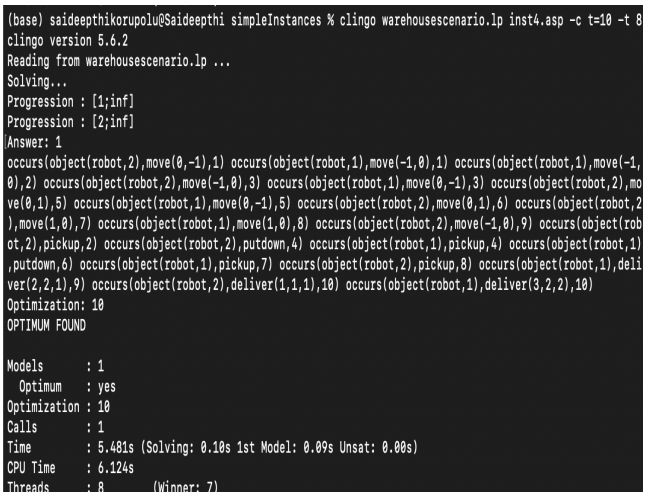

Figure 4: inst4.asp result

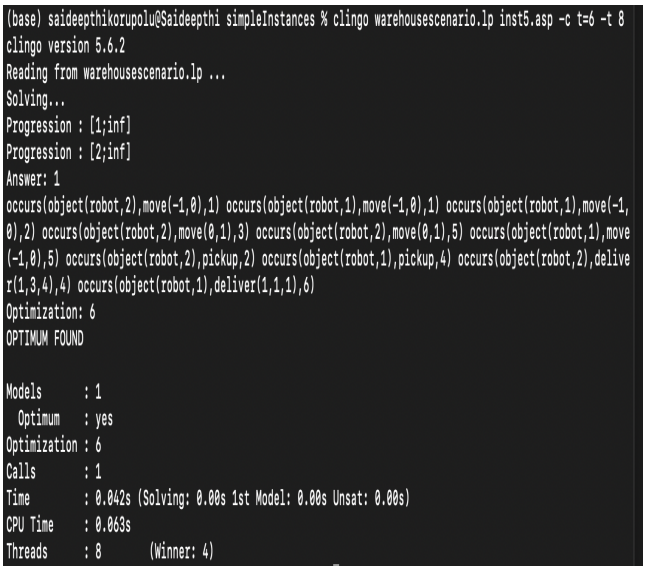

Figure 5: inst5.asp result

Finally, I ran the five instance programs on my code, and the results were successful, indicating that the code was able to efficiently solve the warehouse problem statement with the least number of steps and in the shortest time possible. My contributions to the project were essential in achieving this result.

7. LESSONS LEARNED

Some important lessons I learned from the project are

The importance of considering constraints: When tackling a complex problem like the warehouse fulfillment challenge, it is crucial to carefully consider all of the constraints involved. By implementing constraints such as limiting robot movements in time and number, and ensuring that product units do not exceed shelf capacity, the team was able to solve the problem more efficiently.

The value of identifying edge cases: One of the challenges encountered in the project was identifying edge cases that might cause issues with the program. By taking the time to observe the code for all test cases and keeping track of them over time, the team was able to identify potential edge cases and address them before they became a problem.

Programming Technologies : I gained a deeper understanding of programming technologies such as ASP (Answer Set Programming) and Clingo. I learned how to effectively use these technologies to solve complex problems and implement efficient algorithms. This experience has broadened my knowledge and skill set as a programmer and has prepared me to tackle more challenging programming projects in the future.

The value of testing and troubleshooting: Throughout the project, the team was constantly testing and troubleshooting the code to ensure that it was functioning correctly. This helped them to identify potential issues early on and make adjustments as necessary. Testing also helped to ensure that the program was efficient and optimized for the given problem.

The importance of algorithm selection: In order to achieve an optimized solution for the warehouse problem statement, the team had to carefully consider various algorithms and their complexity. By selecting the most efficient algorithm for the given problem, they were able to achieve the optimal solution in the least number of steps possible.

8. REFERENCES

[1] Joohyung Lee CSE 579 Lecture Slides and Videos

[2] Wikipedia on Answer Set Programming
https://en.wikipedia.org/wiki/Answer_set_programming

[3] Clingo installation
https://pypi.org/project/clingo/