Course Project: Milestone 4 - Individual Project Report

Aasish Tammana (1225545568)
Arizona State University
atammana@asu.edu

Abstract

This project focuses on the innovative implementation of a logic-based automated system designed to streamline operations in a simulated warehouse setting. The primary aim is to enhance the efficiency of robotic agents tasked with the collection and delivery of products to specific picking stations within a grid-based warehouse model. The system is intricately designed to handle various logistical elements such as layout of warehouse, including grid cells, highways, picking stations, and positioning of robots and shelves. Additionally, it manages product distribution, order fulfillment.

Inputs for the system are meticulously defined using logical facts that paint a detailed picture of the warehouse's structure and operational requirements. The output is a carefully orchestrated set of actions executed by the robots, encapsulated in time-stepped movements, product pickups, and deliveries. The key performance indicator of the system is the "makespan", which is the total duration needed to complete all given orders. The overarching goal is to minimize this timespan, thereby optimizing overall efficiency.

The system's robustness and adaptability were tested using five varied instance files, each presenting unique warehouse configurations and order specifics. These tests successfully demonstrated the system’s ability to generate effective action plans, optimizing task sequences, and ensuring resourceful management of operations. The project's outcomes underscore the potential and versatility of logic programming in enhancing operational efficiency in dynamic and complex environments such as logistics and warehouse management.

Problem Statement

This project addresses the challenge of optimizing robotic operations in an automated warehouse. In this setting, robots navigate a grid-like warehouse to deliver products from shelves to picking stations, fulfilling orders. The complexity arises from the limited space, as robots carrying shelves cannot pass under other shelves, and certain grid cells, designated as highways, restrict the placement of shelves. The objective is to develop an algorithm that orchestrates the robots' movements, including picking up, delivering, and avoiding collisions, in the most time-efficient manner. The primary goal is to minimize the total time, or 'makespan', required to complete all orders.

This task presents a unique opportunity to explore optimization strategies in a dynamic environment with multiple independent agents and spatial constraints. The project's success lies in creating a solution that efficiently manages robotic logistics in confined and complex spaces.

Project Background

The foundation of our project lies in the realm of Answer Set Programming (ASP), a form of declarative programming oriented towards difficult, primarily NP-hard, search problems. ASP is rooted in logic programming and nonmonotonic reasoning, offering a unique approach to problem-solving. It presents an alternative to traditional programming, enabling the expression of complex problems through logical rules which are then used to generate "answer sets" representing potential solutions.

Understanding ASP

- **Fundamentals**: ASP is based on idea of defining problem through a set of logical rules. These rules describe how various elements relate to each other, and constraints restrict the solutions to those that meet criteria.
- **Problem Representation**: In ASP, a problem is expressed in terms of rules, facts, and constraints in a logic program. The ASP solver then processes this program to find solutions that satisfy all the given conditions.
- **Solver Mechanics**: ASP solvers, like Clingo, generate solutions by computing stable models (or answer sets) of logic programs. This involves finding sets of literals that are self-consistent and satisfy all the rules of program.

ASP in the Context of Automated Warehousing:

- **Application:** In our project, ASP was used to simulate an automated warehouse system. This involved modeling the layout, robots' paths, shelves, and orders as a set of logical facts and rules.
- **Problem Solving:** The challenge was to optimize the robots' actions to fulfill orders in the minimum number of steps, a complex task efficiently managed by ASP's powerful combinatorial search capabilities.

In the development of our project, a significant role was played by online resources that provided a deep understanding of Answer Set Programming (ASP). Key among these was the Potassco guide, an essential starting point for learning ASP with Clingo. It offered an introductory look into Clingo's input language, modeling examples, and practical usage tips. Moreover, for a more formal introduction to Clingo and its semantics, resources available on online websites like this proved invaluable. The Potassco book was another crucial resource, delving into the foundational algorithms and modeling techniques of ASP. Additionally, teaching materials and tutorials available on teaching.potassco.org greatly facilitated interactive learning and a practical understanding of ASP in the context of project.

Approach to solving the problem

I employed an approach that strategically managed the movements and actions of robots to fulfill orders efficiently. My methodology revolved around creating an optimized solution that minimizes makespan needed to complete all orders, ensuring that robots could move, pick, deliver products without collisions. Here is a breakdown:

Understanding the Problem Space:
Firstly, I analyzed the problem statement to understand dynamics of warehouse environment. This included recognizing grid-based layout, role of robots, shelves, products, and the imperative to avoid collisions and inefficient routes.

Designing the Input Format:
To represent the warehouse, I devised approach to parse the input schema facts. This schema detailed grid layout, locations of highways, picking stations, robots, shelves, products and about the orders that needed fulfillment.

Algorithm Development:
With the problem space clearly defined, I developed an algorithm tailored to navigate the robots through the grid.
Key considerations in this algorithm included:

- **Path Planning**: Ensuring robots move optimally to reduce travel time.
- **Shelf Management**: Strategically picking up and putting down shelves to clear paths and access products.
- **Order Fulfillment**: Coordinating robot actions to gather and deliver the correct quantities of products.
- **Collision Avoidance**: Implementing logic to prevent robots from occupying the same grid cell simultaneously.

Simulation and Iteration:
Using the logic-based programming tool Clingo, I simulated the warehouse environment. I iteratively tested and refined my approach, focusing on optimizing the makespan and ensuring all orders were correctly fulfilled. This involved analyzing robot actions at each time step and adjusting strategies as needed to improve efficiency.

Instance-Specific Adjustments:
I executed my code against different instances with unique layouts. This required fine-tuning the algorithm to adapt to varying configurations, product placements, and orders.

Main results and analysis

I. Key Code Logic and Analysis

Here are the different types of statements in the code with some examples each.

- **Fact Declarations**: These assert unconditional truths without requiring further proof.

Example 1: *node(NODEID) :- init(object(node, NODEID), value(at, pair(X, Y))).*
Explanation: This declares a node with a specific NODEID if there is an initialization (init) of an object with that NODEID and coordinates X, Y.
Example 2: *product(PRODUCTID) :- init(object(product, PRODUCTID), value(on, pair(SHELFID, PRODUCTQUANTITY))).*
Explanation: Declares a product with PRODUCTID if there's an initialization of a product object with PRODUCTID on a SHELFID with a certain PRODUCTQUANTITY.

- **Rules**: Contain head (conclusion) and body (condition).

Example 1:*locationofrobot(ROBOT,object(node, NEW_ND), T + 1) :- ...*
Explanation: Defines the new location of a robot at time T + 1 based on its current location and movement.
Example 2: *shelfOn(S, object(robot, ROBOTID), T + 1) :- pickUpShelf(ROBOTID, S, T), shelfOn(S, object(node, NODECOUNT), T), locationofrobot(ROBOTID, object(node, NODECOUNT), T).*
Explanation: Updates the location of a shelf to be on a robot at time T + 1 if the robot picks up the shelf at time T.

- **Choice Rules**: Allow for selection in multiple outcomes.

Example 1: *{pickUpShelf(ROBOT, SHELFID, T) : shelf(SHELFID)} 1 :- ...*
Explanation: Provides a choice for a robot to pick up any one shelf among the available shelves at time T.
Example 2: *{putDownShelf(ROBOT, SHELFID, T) : shelf(SHELFID)} 1 :- ...*

Explanation: Gives a choice for a robot to put down a shelf it is carrying at time T.

- **Hard Constraints**: These enforce strict rules that cannot be violated.

Example 1: *:- locationofrobot(ROBOTID, object(node, NODECOUNT), T), robotMove(ROBOT, move(DELTAX, DELTAY), T), locationofnode(NODECOUNT, pair(X, Y)), Y + DELTAY < 1.*
Explanation: Prevents a robot from moving outside the warehouse grid vertically (lower bound).
Example 2: *:- shelfOn(S, object(node, _), T), shelfOn(S, object(robot, _), T).*
Explanation: Ensures that a shelf cannot be simultaneously on a node and being carried by a robot at the same time.

- **Soft Constraints**: Not explicitly present in your code, but they are similar to hard constraints and guide the optimization process.

- **Optimization Statements**: Aim to find the optimal solution according to some criteria.

Example 1: *#minimize {T : occurs(O, A, T)}.*
Explanation: Minimizes the highest time step (makespan) in which any action occurs.

- **Aggregates**: Used to compute collective measures.

Example 1*: numRobots(NODECOUNT) :- NODECOUNT = #count{I : init(object(robot, I), value(at, pair(X, Y)))}.*
Explanation: Counts total number of robots.

- **Show Statements**: Control parts of solution displayed.

Example 1: *#show numActions/1.*
Explanation: Displays the total number of actions taken in the solution.
Example 2*: #show timeTaken/1.*
Explanation: Shows time taken to complete all actions.

- **Arithmetic Operations**: Perform calculations in rules.

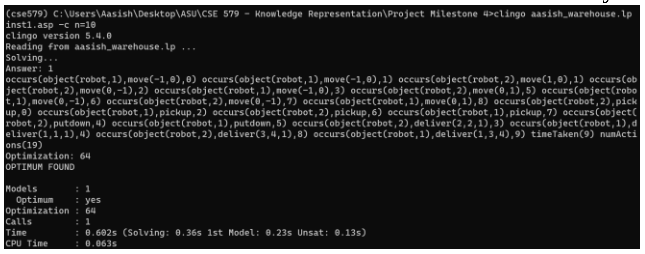
Example 1: *locationofrobot(ROBOT, object(node, NEW_ND), T + 1) :- ... locationofnode(NEW_ND, pair(X + DELTAX, Y + DELTAY)), ...*
Explanation: Calculates the new X and Y coordinates for a robot after a move.

Example 2: *orderAt(ORDERID, object(node, NODECOUNT),contains(PR,OU-DELIVERYQUANTITY),T+1) :-*
Explanation: Updates the order quantity by subtracting the delivery quantity after a delivery action.

II. DETAILED ANALYSIS OF RESULTS

a) Instance File 1 (inst1.asp)

- Initial Configuration: Included nodes, highways, picking stations, robots, shelves, and products.
- Robot Actions: Robot1 and Robot2 performed moves and actions like pickup, putdown, deliver.
- Deliveries: Deliveries included delivering products to specified picking stations (deliver(221), deliver(111), deliver(341), and deliver(134)).
- Optimization: Time - 9 time units with 19 actions.
- Analysis: This instance shows moderately complex operation with robots executing many actions. Time and number of actions indicate medium efficiency.

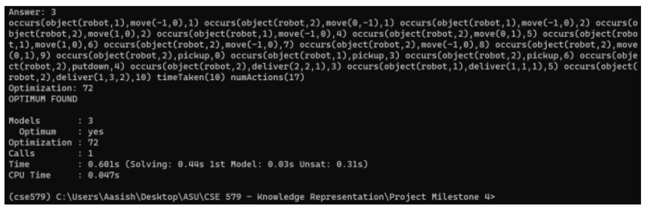


```
(cse579) C:\Users\Aasish\Desktop\ASU\CSE 579 - Knowledge Representation\Project Milestone 4>clingo aasish_warehouse.lp
inst1.asp -c n=10
clingo version 5.4.0
Reading from aasish_warehouse.lp ...
Solving...
Answer: 1
occurs(object(robot,1),move(-1,0),0) occurs(object(robot,1),move(-1,0),1) occurs(object(robot,2),move(1,0),1) occurs(ob
ject(robot,2),move(0,-1),2) occurs(object(robot,1),move(-1,0),3) occurs(object(robot,2),move(0,1),5) occurs(object(robo
t,1),move(0,-1),6) occurs(object(robot,2),move(0,-1),7) occurs(object(robot,1),move(0,1),8) occurs(object(robot,2),pick
up,0) occurs(object(robot,1),pickup,2) occurs(object(robot,2),pickup,6) occurs(object(robot,1),pickup,7) occurs(object(
robot,2),putdown,4) occurs(object(robot,1),putdown,5) occurs(object(robot,2),deliver(2,2,1),3) occurs(object(robot,1),d
eliver(1,1,1),4) occurs(object(robot,2),deliver(3,4,1),8) occurs(object(robot,1),deliver(1,3,4),9) timeTaken(9) numActi
ons(19)
Optimization: 64
OPTIMUM FOUND

Models       : 1
  Optimum    : yes
Optimization : 64
Calls        : 1
Time         : 0.602s (Solving: 0.36s 1st Model: 0.23s Unsat: 0.13s)
CPU Time     : 0.063s
```

b) Instance File 2 (inst2.asp)

- Initial Configuration: Similar to inst1.asp with adjustments in product locations and orders.
- Robot Actions: Similar pattern of movements and actions by Robot1 and Robot2.
- Deliveries: Deliveries were made for different orders (deliver(132), deliver(221), deliver(111)).
- Optimization: Time increased to 10 units,17 actions.
- Analysis: This instance, while similar to inst1.asp, shows a slight increase in time taken, possibly due to different order configurations requiring more complex routing.



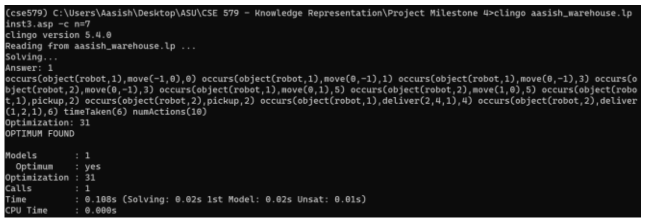
```
Answer: 3
occurs(object(robot,1),move(-1,0),1) occurs(object(robot,2),move(0,-1),1) occurs(object(robot,1),move(-1,0),2) occurs(o
bject(robot,2),move(1,0),2) occurs(object(robot,1),move(-1,0),4) occurs(object(robot,2),move(0,1),5) occurs(object(robo
t,1),move(1,0),6) occurs(object(robot,2),move(-1,0),7) occurs(object(robot,2),move(-1,0),8) occurs(object(robot,2),move
(0,1),9) occurs(object(robot,2),pickup,0) occurs(object(robot,1),pickup,3) occurs(object(robot,2),pickup,6) occurs(obje
ct(robot,2),putdown,4) occurs(object(robot,2),deliver(2,2,1),3) occurs(object(robot,1),deliver(1,1,1),5) 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         : 0.601s (Solving: 0.44s 1st Model: 0.03s Unsat: 0.31s)
CPU Time     : 0.047s

(cse579) C:\Users\Aasish\Desktop\ASU\CSE 579 - Knowledge Representation\Project Milestone 4>
```

c) Instance File 3 (inst3.asp)

- Initial Configuration: Included a single picking station and different shelf and product placements.
- Robot Actions: Both robots were involved in fewer movements and deliveries.
- Deliveries: Fewer deliveries (deliver(241), deliver(121)) than in previous instances.
- Optimization: More efficient-6 timeunits,10 actions.
- Analysis: This instance appears to be more efficient, potentially due to the reduced complexity of the setup and fewer delivery requirements.

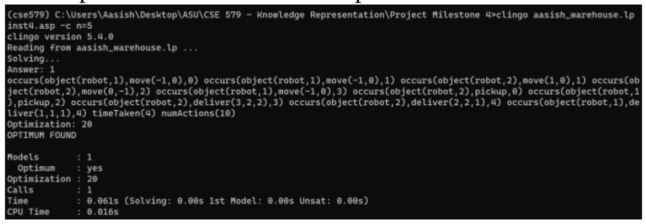


```
(cse579) C:\Users\Aasish\Desktop\ASU\CSE 579 - Knowledge Representation\Project Milestone 4>clingo aasish_warehouse.lp
inst3.asp -c n=7
clingo version 5.4.0
Reading from aasish_warehouse.lp ...
Solving...
Answer: 1
occurs(object(robot,1),move(-1,0),0) occurs(object(robot,1),move(0,-1),1) occurs(object(robot,1),move(0,-1),3) occurs(o
bject(robot,2),move(0,-1),3) occurs(object(robot,1),move(0,1),5) occurs(object(robot,2),move(1,0),5) occurs(object(robo
t,1),pickup,2) occurs(object(robot,2),pickup,2) occurs(object(robot,1),deliver(2,4,1),4) occurs(object(robot,2),deliver
(1,2,1),6) timeTaken(6) numActions(10)
Optimization: 31
OPTIMUM FOUND

Models       : 1
  Optimum    : yes
Optimization : 31
Calls        : 1
Time         : 0.108s (Solving: 0.02s 1st Model: 0.02s Unsat: 0.01s)
CPU Time     : 0.000s
```

d) Instance File 4 (inst4.asp)

- Initial Configuration: Similar node and highway setup with different product / order configurations.
- Robot Actions: The pattern of robot movements and actions was consistent with earlier instances.
- Deliveries: Deliveries were completed (deliver(322), deliver(221), deliver(111)).
- Optimization: Most efficient-4 timeunits,10 actions.
- Analysis: This instance shows most efficient operation among the four, potentially due to optimal placement of shelves and products relative to orders.

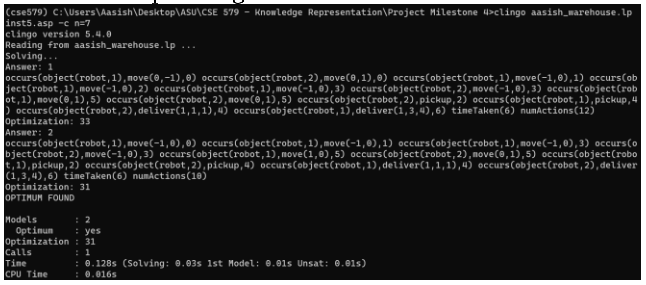


```
(cse579) C:\Users\Aasish\Desktop\ASU\CSE 579 - Knowledge Representation\Project Milestone 4>clingo aasish_warehouse.lp
inst4.asp -c n=5
clingo version 5.4.0
Reading from aasish_warehouse.lp ...
Solving...
Answer: 1
occurs(object(robot,1),move(-1,0),0) occurs(object(robot,1),move(-1,0),1) occurs(object(robot,2),move(1,0),1) occurs(ob
ject(robot,2),move(0,-1),2) occurs(object(robot,1),move(-1,0),3) occurs(object(robot,2),pickup,0) occurs(object(robot,1
),pickup,2) occurs(object(robot,2),deliver(3,2,2),3) occurs(object(robot,2),deliver(2,2,1),4) occurs(object(robot,1),de
liver(1,1,1),4) timeTaken(4) numActions(10)
Optimization: 20
OPTIMUM FOUND

Models       : 1
  Optimum    : yes
Optimization : 20
Calls        : 1
Time         : 0.061s (Solving: 0.00s 1st Model: 0.00s Unsat: 0.00s)
CPU Time     : 0.016s
```

e) Instance File 5 (inst5.asp)

- Initial Configuration: Similar to previous instances with slight variations in product and shelf locations.
- Robot Actions: Similar actions were taken by the robots as in previous instances.
- Deliveries: deliver(111) and deliver(134).
- Optimization: Efficiency similar to inst3.asp with time taken 6 time units and 10 actions.
- Analysis: This instance also exhibits a high level of efficiency, indicating effective robot movement and action planning in the simulated environment.



```
(cse579) C:\Users\Aasish\Desktop\ASU\CSE 579 - Knowledge Representation\Project Milestone 4>clingo aasish_warehouse.lp
inst5.asp -c n=7
clingo version 5.4.0
Reading from aasish_warehouse.lp ...
Solving...
Answer: 1
occurs(object(robot,1),move(0,-1),0) occurs(object(robot,2),move(0,1),0) occurs(object(robot,1),move(-1,0),1) occurs(ob
ject(robot,1),move(-1,0),2) occurs(object(robot,1),move(-1,0),3) occurs(object(robot,2),move(-1,0),3) occurs(object(rob
ot,1),move(0,1),5) occurs(object(robot,2),move(0,1),5) occurs(object(robot,2),pickup,2) occurs(object(robot,1),pickup,4
) occurs(object(robot,2),deliver(1,1,1),4) occurs(object(robot,1),deliver(1,3,4),6) timeTaken(6) numActions(12)
Optimization: 33
Answer: 2
occurs(object(robot,1),move(-1,0),0) occurs(object(robot,1),move(-1,0),1) occurs(object(robot,1),move(-1,0),3) occurs(o
bject(robot,2),move(-1,0),3) occurs(object(robot,1),move(1,0),5) occurs(object(robot,2),move(0,1),5) occurs(object(robo
t,1),pickup,2) occurs(object(robot,2),pickup,4) occurs(object(robot,1),deliver(1,1,1),4) occurs(object(robot,2),deliver
(1,3,4),6) timeTaken(6) numActions(10)
Optimization: 31
OPTIMUM FOUND

Models       : 2
  Optimum    : yes
Optimization : 31
Calls        : 1
Time         : 0.128s (Solving: 0.03s 1st Model: 0.01s Unsat: 0.01s)
CPU Time     : 0.016s
```

Conclusion

The results across the five instances indicate a consistent achievement of optimal solutions within constraints of the problem. Answer sets, as observed in the screenshots, show the logic program successfully calculated actions for robots that resulted in delivery of all products without any collisions and within the minimum time, as indicated by the "OPTIMUM FOUND" statement in each instance.

My self-assessment of this work would highlight the success in developing a valid solution that fulfills all orders efficiently. I managed to design an optimal plan, as indicated in the example provided in the document, which demonstrates my understanding of the problem and my ability to translate that into a practical, executable strategy. The solution I worked out adhered to all the given constraints, such as no collisions, efficient use of robots' actions per time step, and the prohibition of placing shelves on highway.

Opportunities For Future Work

For future work, there are several avenues that could be explored to build upon the foundation of this project:

- **Scalability**: Testing the solution with many robots and orders could provide insights into the scalability of the current model. It could also help identify limitations in the current logic that appear under higher complexity.
- **Real-Time Adaptability**: Incorporating dynamic changes that occur in a real warehouse environment, such as sudden changes in order priorities or obstacles, could make model more robust to real-world scenarios.
- **Machine Learning Integration**: Applying machine learning algorithms to predict order patterns could optimize the pre-staging of goods and improve the overall efficiency of the warehouse operations.
- **Robotic Collaboration**: Expanding the model to include collaborative tasks between robots, like cooperative lifting of heavy items, could be interesting area to explore.
- **User Interface Development**: Creating a user-friendly interface that allows warehouse staff to interact with the system, adjust parameters, and receive real-time updates could enhance the practicality of the solution.

References

Paper Presented at Meeting and Published Proceedings

Lifschitz, V. 2008. What is Answer Set Programming? In Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence (AAAI-08). Palo Alto, CA: AAAI Press.

Paper Presented at Meeting and Published Proceedings

Kaiser, P.; Thevapalan, A.; Roidl, M.; Kern-Isberner, G.; Wilhelm, M.: Towards Finding Optimal Solutions For Constrained Warehouse Layouts Using Answer Set Programming. In: Herberger, D.; Hübner, M.; Stich, V. (Eds.): Proceedings of the Conference on Production Systems and Logistics: CPSL 2023 - 1. Hannover : publish-Ing., 2023, S. 600-610. DOI: https://doi.org/10.15488/13480

Paper Presented at Meeting and Published Proceedings

Ten Hompel, M.; and Schmidt, T. 2007. Warehouse Management: Automation and Organisation of Warehouse and Order Picking Systems. Berlin, Heidelberg: Springer-Verlag.

Appendix

The appendix folder submitted along with this project report include the following files:

1) **Aasish_warehouse.lp** : This file contains the constrains and clingo ASP program which can be executed to generate the actions.

2) **Instance Files**: All 5 instance files provided are uploaded in this folder.

3) **Project Milestone 4 Results** : This file includes the commands executed for each instance file, the output obtained and screenshots to support the results