An Exploration in Answer Set Programming Applied to Autonomous Warehouses

Trey Manuszak
Arizona State University
Tempe, Arizona
tmanusza@asu.edu

Abstract

Automated warehouses provide many benefits to managing one's supply chain. Those are including, but not limited to, employee safety, minimal product mishandling and displacement, increased traceability and auditability, and increased productivity. Coordinating this automation requires a program that can quickly determine optimal, or as close to optimal as possible, scheduling of the warehouse robotics. In this work, we collect requirements and limitations of an autonomous warehouse, develop a scheduling program using answer set programming, while implementing techniques from the knowledge representation and reasoning field of artificial intelligence to generate a time-optimized course of action for the delivery of product.

Problem Statement

Finding an opimal schedule of robots delivering products in an automated warehouse is known to be a problem in the $\mathcal{NP}$-hard complexity class (Lenstra, Rinnooy Kan, and Brucker 1977). There have been many attempts to solve this problem (1999; 2012a; 2012b; 2016). Answer set programming (ASP) is designed to solve $\mathcal{NP}$-hard search problems. Of the many different ASP languages, we chose to work with `clingo` to build an automated warehouse scheduler to deliver product to picking stations (Gebser et al. 2019).

To develop an automated warehouse scheduling program we must define the constraints of the warehouse, warehouse robotics, products, shelving, and orders, then generate a time optimized route for the robots to deliver products to the picking stations. The constraints we use come from Google's Answer Set Programming Challenge 2019 and can be generally summed up as follows (Google 2019):

- The warehouse consists of tiles, some of which are highway tiles which have their own certain limitations,
- A robot can only move vetically or horizontally, they can move under shelves if they are not carrying one already, they can set shelves down on normal tiles, and they can place product in picking stations that they are next to,
- Products must be on a shelf or at a picking station,
- Shelves may only be placed on normal tiles,
- Orders consist of items to be placed at a specific picking station,
- A picking station must be on a normal tile and cannot be moved.

The program, when given a state of the warehouse and the orders to be completed, generates a schedule for completing the orders in as little time as possible. An order that is assigned to a picking station is completed, if all items in the order are delivered to that picking station. The schedule must correctly complete all orders. The program should find a schedule completing all orders in T steps, where no schedule is possible in less than T steps. It is important to note that the constraints on movement of the robots is not serializeable. That is, we are allowed to move a robot R_1, to a space just previously occupied by robot R_2, as long as robot R_2 is also moving to a diffirent space than just previously occupied by robot R_1.

Current Progress and Difficulties

With the constraints, we are also supplied with an expected input format. So far, we have programmed the initialization of the warehouse facts and fluents from the expected input format. We have also developed and definied what is necessary to allow movement of the warehouse robots according to the movement constraints and inter robot behaviors defined in the original problem.

This was originally attempted without the knowledge of reasoning about actions concepts recently acquired. It was difficult and unltimately unsuccessful to get the actions of robot movement correct without these skills. Trying to get the default behavior of the nonexistence of robot movement was challenging to implement. However, applying these new concepts made it much more effective to define the actions of robot movement and interaction, while adhering to the constraints. So, what we have currently are robots moving around a warehouse in a correct manner and this can be seen by executing the program in the Appendix on different warehouse initializations.

We tested for correct manner by testing the correctness of low complexity robot movement and gradually increas-

ing the complexity if previous tests work. To start, we tested the correct initialization of the warehouse objects on the provided test cases. Once we verified the correctness of the initializations, we created a small 2x2 tile warehouse with just one robot. Once we verified the correctness that the robot could move in all valid directions, we introduced a second robot and tested for correctness of the constraints between robots. Specifically, the completed tasks are as follows:

- Create node facts,
- Create highway facts,
- Create robot facts and fluent,
- Create shelf facts and fluent,
- Create product facts,
- Create picking station facts,
- Create order facts and fluent,
- Define move operations and constraints (only accounting for robots in the warehouse),
- Generate exogenous robot movement occurrences,
- Define the direct effect of a robot movement and the state change of the robot's location,
- Define the uniqueness and existance of a robot's location,
- Define the law of intertia for a robot's location,
- Constrain the robot to one movement per timestep.

Future Tasks and Plan

There are still many important tasks to be completed. We still have to complete the constraints and interaction between robots and other warehouse objects. We also still have to implement the occurrences other than movement available for a robot to make, such as picking up and setting down shelves, and delivering product. Doing this will require the development of more exogenous actions, definitions of uniqueness, and application of laws of inertia. We plan to just as before, implement one action at a time and test the behavior on low complexity. Once each low level of complexity is correct, further testing on higher complexity instances and interactions will be performed until there is suffucient confidence in the implementation. Specifically, the tasks that are yet to be completed are as follows:

- Generate exogenous actions for pickup and putting down shelves, and delivering product,
- Define constrains for interactions between robots and shelves,
- Define the direct effects between carried shelves and movement, as well as, the delivery of product and orders,
- Implement the uniqueness and existence of shelves, products, and orders,
- Define the law of inertia for shelves, products, and orders,
- Implement a minimization function with respect to the time step.

References

Atieh, A. M.; Kaylani, H.; Al-Abdallat, Y.; Qaderi, A.; Ghoul, L.; Jaradat, L.; and Hdairis, I. 2016. Performance improvement of inventory management system processes by an automated warehouse management system. *Procedia Cirp* 41:568–572.

Basile, F.; Chiacchio, P.; and Coppola, J. 2012a. A hybrid model of complex automated warehouse systems—part i: Modeling and simulation. *IEEE Transactions on Automation Science and Engineering* 9(4):640–653.

Basile, F.; Chiacchio, P.; and Coppola, J. 2012b. A hybrid model of complex automated warehouse systems—part ii: Analysis and experimental results. *IEEE transactions on automation science and engineering* 9(4):654–668.

Gebser, M.; Kaminski, R.; Kaufmann, B.; and Schaub, T. 2019. Multi-shot asp solving with clingo. *Theory and Practice of Logic Programming* 19(1):27–82.

Google. 2019. Answer set programming challenge 2019. https://sites.google.com/view/aspcomp2019/.

Lenstra, J.; Rinnooy Kan, A.; and Brucker, P. 1977. Complexity of machine scheduling problems. In Hammer, P.; Johnson, E.; Korte, B.; and Nemhauser, G., eds., *Studies in Integer Programming*, volume 1 of *Annals of Discrete Mathematics*. Elsevier. 343–362.

van den Berg, J. 1999. A literature survey on planning and control of warehousing systems. *IIE Transactions* 31(8):751–762.

Appendix

```
%%%% OBJECTS AND LOCATIONS %%%%
% Here we define objects and object location (if it moves)
node(N,X,Y) :- init(object(node,N), value(at, pair(X,Y))).

% highway (fast cell in warehouse)
highway(H,X,Y) :- init(object(highway,H), value(at, pair(X,Y))).

% picking station
pickStation(P,X,Y) :- init(object(pickingStation, P),
  value(at, pair(X,Y))).

% robot and robot's location
robot(R) :- init(object(robot, R), value(at, pair(X,Y))).
robotLocation(R,X,Y,0) :- init(object(robot,R), value(at, pair(X,Y))),
  node(N,X,Y).

% shelf
shelf(S) :- init(object(shelf, S), value(at, pair(X,Y))).
shelfLocation(S,X,Y,0) :- init(object(shelf, S), value(at, pair(X,Y))),
  node(N,X,Y).

% products
productOnShelf(P,S,U,0) :- init(object(product, P), value(on, pair(S,U))).

% order
order(O,P,U,0) :- init(object(order, O), value(line, pair(P,U))).
deliverTo(O,P) :- init(object(order, O), value(pickingStation,P)).

%%%% MOVING %%%%
% move options
move(1,0; -1,0; 0,1; 0,-1).

% must be in the warehouse
:- robotLocation(R,X,Y,T), not node(_, X, Y).

% one robot per node
:- robotLocation(R1,X,Y,T), robotLocation(R2,X,Y,T), R1!=R2.

% robot movement is exogenous
{occurs(object(robot,R), move(X,Y), T)} :- robot(R), move(X,Y), T=1..m.

% direct effect
robotLocation(R,X + X1,Y + Y1,T) :- occurs(object(robot,R), move(X1,Y1), T),
  robotLocation(R,X,Y,T-1), T=1..m.

% uniqueness and existence
:- not {robotLocation(R,X,Y,T):node(N,X,Y)}=1, robot(R), T=1..m.

% robots cant switch nodes (note, serializability is not required)
:- robotLocation(R1,X1,Y1,T), robotLocation(R2,X2,Y2,T),
  robotLocation(R1,X2,Y2,T-1), robotLocation(R2,X1,Y1,T-1), R1!=R2.


%%%% LAWS OF INERTIA %%%%
{robotLocation(R,X,Y,T)} :- robotLocation(R,X,Y,T-1), T=1..m.

%%%% AXIOMS %%%%
% robot can only do one thing at a time
:- occurs(object(robot, R), A1, T), occurs(object(robot, R), A2, T), A1!=A2.
```