Warehouse Planning

Brandon Mossop[1], Mohammad Zakzok[2]

School of Computing, Queen's University, Kingston, Canada

Abstract
The effective management of warehouses within the context of a supply chain constitutes a fundamental challenge for organizations that engage in large-scale operations. To address this challenge, we have developed a planning model that integrates both temporal and numerical factors. This hybrid model captures the key aspects of warehouse stock management, encompassing multiple warehouses, vehicles, and item types, with the ultimate aim of maximizing profits. We evaluate our model by testing it on various settings using the enhsp-2020 planner. We conclude that warehouse management is quite difficult for enhsp-2020, as the planner fails to produce suitable plans within a reasonable time frame. However, the planner is able to produce high quality plans when presented with simpler problem settings, leading us to believe that future planners could possibly solve this problem.

1. Introduction

A supply chain is a complex network of interconnected facilities, modes of transportation, and processes designed to enable the sourcing, manufacturing, warehousing, and transportation of materials and products. An optimally designed supply chain ensures that goods and materials are efficiently and effectively delivered to the end customer.

Warehouse planning plays a pivotal role in the optimization of the supply chain, primarily due to the finite storage capacity of warehouses and the imperative to meet customer demand. The implementation of effective warehouse planning is critical to ensuring optimal utilization of storage capacity, thereby minimizing the incidence of wasted space and enhancing the overall efficiency of the supply chain.

This study aims to optimize warehouse logistics planning considering inventory and total profit using a hybrid planning formalism. The hybrid formalism enables numeric and temporal aspects of the planning domain to be modelled realistically. Our model seeks to simulate and optimize the logistics of managing the stock levels in multiple connected warehouses based on consumer demand.

2. Related Work

Our study relates to prior research conducted in warehouse planning with either a temporal or hybrid formalism.

In [1] the study investigates a multi-agent planning problems with concurrent action constraints. There are two domains considered: a warehouse domain and a vehicle testing domain. Rather than managing warehouse stock management within a supply chain, this warehouse domain problem develops a concurrent plan for a fleet of robots in a manufacturing plant, which is inspired by a real-world warehouse problem for a centralised mission planner.

Another warehouse planning study involves shuttles moving goods within a warehouse [2]. A framework is proposed to manage the shuttles in a very dense environment, allowing efficient input and output, storage, and sequencing. The framework is conflict-free, space and energy efficient and its suitability is demonstrated on a real-world problem.

In another study, warehouse stock management and transportation routing are considered jointly in order to minimize both negative warehouse stock and transport cost [3]. The multi-objective problem is modeled and solved by the proposed greedy randomized adaptive search (GRASP) algorithm. The results shows that the proposed algorithm outperforms a greedy search technique, improving negative average warehouse stock.

The management of inventory across multiple warehouses is an integral challenge faced by modern large businesses. This problem has been addressed using various methodologies, and functional solutions have already been established. Nevertheless, to the best of our knowledge, the utilization of a hybrid planning formalism that incorporates both temporal and numerical aspects has not been explored in this domain. We aim to introduce a novel solution to warehouse management by implementing a hybrid planning formalism with temporal and numerical considerations.

3. A Planning Model for Warehouse Management

Our model consists of a collection of warehouses, suppliers, vehicles, and items. The goal is to minimize the cost of our stocking operations and to maximize our profits.

CISC-813: Automated Planning – Final Project

Items are then in turn sold at each warehouse to meet an assumed rate of demand. When items are sold, revenue is added, and the warehouses will have vacant stock space to be replenished. To incentivize the planner to produce economical plans, we add a minimal associated cost with the transport of items. The user is offered a high level of flexibility in specifying the goal state as they can specify the total-profit, stock amount in a specific warehouse, location of vehicles, and maximum amount of days.

As we require the incorporation of both temporal and numerical aspects in our model, we utilize a hybrid modelling approach using PDDL+ targeted for use by the enhsp-2020 planner [4]. We created multiple similar models with various levels of success, mostly due to more complex representations overwhelming the planner [1]. As a general rule, it seemed that models that specified an exact time frame for operations caused problems for enhsp-2020 while models that only gave an upper limit performed significantly better. One of our poorly performing models is available for use in our project[2].

Initially, we had intended to give items the ability to expire, but since the planner was already struggling to find plans for non-perishable item distribution, we opted not to implement this feature. This is due to the fact it would entail that every instance of an item would be treated as a separate entity, surely overwhelming enhsp-2020.

We give a detailed description of our most successful model in terms of of objects, predicates, functions, and some of the actions and processes.

3.1. Domain

Our model exploits every aspect of the hybrid formalism. It uses object types, predicates, functions, actions, processes, and events.

3.1.1. Objects

The model has of two main object types, containers and items. We heavily nest our container type to minimize the number of functions needed.

- **Items**: Describes a type of item that is to be sold. Note that it does not describe an individual item, but rather the collection of an item (e.g. bottles, scissors, beds etc.).
- **Containers**: A container is any object that has stock space. In our model, this includes two subtypes **places** and **vehicles**. Places are further split into two more sub-types, **warehouses** and **suppliers**. The suppliers have infinite stock of items, and are where warehouses buy all their items. In turn, warehouses sell those items to the end-consumer. The vehicles are used to transport items between connected places.

[1]The model is publicly available to be used at https://github.com/Zakzok/WareHouse-Manager.

[2]The complex model can be found in *exactTiming.zip*

3.1.2. Predicates

Predicates are exclusively used to specify the movement of vehicles between places. We have three predicates, they each describe:

- **(con** p_1 p_2 **- place)**: Whether p_1 and p_2 are connected. Must be defined symmetrically.
- **(at** v **- vehicle** p **- place)**: Whether vehicle v is at place p.
- **(driving** p_1 p_2 **- place** v **- vehicle)**: Whether vehicle v is being driven from p_1 to p_2.

3.1.3. Functions

Functions do most of the heavy lifting in our model, largely replacing predicates. We split our functions into four types. The first type of functions aid in the movement of vehicles:

- **(transTime** p_1 p_2 **- place)**: Time it takes to go from p_1 to p_2, must be defined symmetrically.
- **(cur_transTime** v **- vehicle)**: Current time that vehicle v has been driven for.

The second type of functions describe inventory of containers:

- **(maxItem** c **- container)**: Total inventory space of container c.
- **(curItem** c **- container)**: Current inventory space occupied in container c.
- **(stock** c **- container** i **- item)**: Stock of item i currently available at container c.

The third type of functions describe the monetary aspects of the model:

- **(buyPrice** i **- item)**: Cost of buying item i from suppliers.
- **(sellPrice** i **- item)**: Amount gained when selling item i to customers.
- **(demand** i **- item** w **- warehouse)**: Per day demand on item i at warehouse w.
- **(sold** i **- item** w **- warehouse)**: Number of items i sold at warehouse w during the ongoing day.
- **(netProfit)**: Describes the overall net profit across all days.

The fourth and last type of functions are used purely for simulation purposes:

- **(hour)**: Describes the number of hours that passed in the day.
- **(maxHour)**: Static function, always set to 24 to describe the number of hours in a day.
- **(day)**: Describe the number of days that have passed so far.

3.1.4. Actions, Processes, and events

The model uses six actions, two processes, and two events. The actions are used to load, transport, and sell items:

- **load_Item**: Loads an item i onto a vehicle v from warehouse w.
- **unload_Item**: Unloads an item i from a vehicle v to warehouse w.
- **begin_Drive**: Begins a the drive process for vehicle v between two places p_1 and p_2.
- **end_Drive**: Ends the drive process for a vehicle v, placing it at the destination.
- **buy_Item**: Buys an item i from a supplier s and loads unto vehicle v.
- **sell_Item**: Sells an item i from warehouse w.

Processes simulate the ticking of time and the travel of vehicles between places:

- **ongoing_Drive**: Keeps a vehicle v on track while traveling from place $p1$ to place $p2$ and adds a small associated cost for every *tick* that the vehicle is being driven.
- **tick_tock**: Responsible for keeping time, continuously increasing the function $hour$ until it reaches 24.

Events simulate the end of a day, resetting the demand on items:

- **reset_Demand**: Triggers when the function $hour$ reaches 24, resets the demand on a specific items i in warehouse w.
- **wrap_Day**: Also triggers when the function $hour$ reaches 24, increments function day by 1 and resets $hour$ to 0.

3.2. Goal States

Complex goal states can be defined using a mixture of functions and predicates. The most basic goal states would require some net profit within a certain window of time. For example, the user can require that the planner achieves 10 net profit within 3 days:

- ($\leq$ day 3)
- ($\geq$ netProfit 10)

We could also describe far more complex goal states, allowing the user to truly micro-manage how well their warehouses are stocked. Suppose warehouses w_1, w_2 and item i are defined in the problem file, we can set the following goal state:

- ($\leq$ day 3)
- ($\geq$ netProfit 10)
- ($\geq$ (stock w_1 i) 3)
- ($\geq$ (stock w_2 i) 5)

Finally, it is also possible to completely remove any requirement of net profit and use the planner to determine the best way to transport items to our warehouses.

- ($\geq$ (stock w_1 i) 3)
- ($\geq$ (stock w_2 i) 5)

4. Evaluation

As stated earlier, our model was targeted to be used by enhsp-2020. We tested our model using two settings:

1. Default: The default settings of enhsp-2020. These settings produce better plans but could take more time to produce the plans. In some cases, the planner may take far too long to find a solution. We terminated the planner in cases where it took more than two minutes.

2. Short-sighted: We describe those settings as short-sited as they produce generally less thoughtful plans in a much faster time. Occasionally, these settings could fail to produce a plan when a plan does indeed exist. To run the short-sited version of enhsp-2020, simply add the following commands when running the planner:

```
-ha true -h hmrp -wh 4
```

4.1. Test Cases

We tested our model against three warehouse graphs, described in figures 1, 2, and 3. For each graph, we tested the model against three goal states, ranked by difficulty[3].

1. Relaxed: Presents a fairly relaxed problem and goal state. Little variety of items (two) is available and the net profit requirement is obviously feasible within the time constraint. In this case, time constraint will be very short (two days at most). This is meant to be a baseline test of the model.
2. Moderate: Similar to the previous requirement, but with more variety of items (four or five), slightly stricter profit margins, and a longer time constraint.
3. Strict: Returns to having little item variety and a shorter time constraint, but required profit margins are more slim.

[3]These problem files are available in the project files, named appropriately to be easily identified. For example, the file *Relaxed-Choke.pddl* is used to test Choke with relaxed requirements.

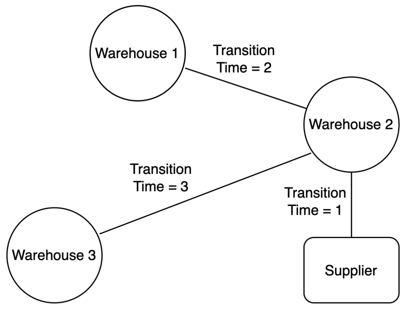

Figure 1: Choke Configuration

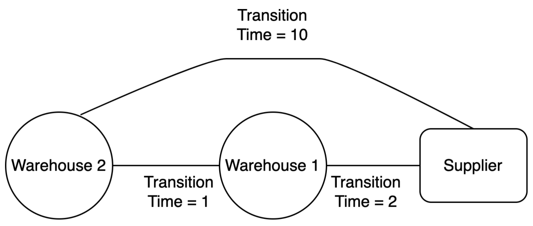

Figure 2: FlatLine Configuration

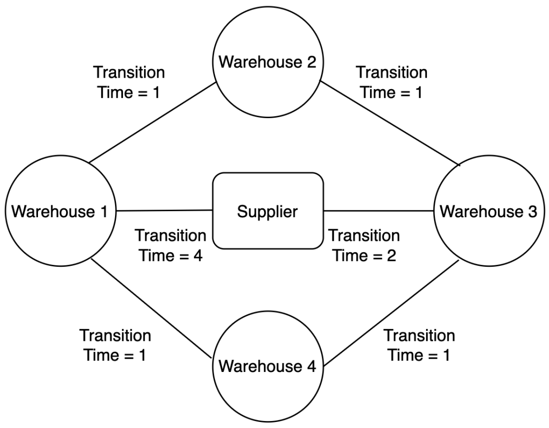

Figure 3: Diamond Configuration

4.2. Performance

We tested each of the nine problem files against both short-sighted and default enhsp-2020. For each instance, the settings were relaxed until at least one of the planner, which was always the short-sighted one, was able to find a plan. Tables 1 and 2 detail our comprehensive results. The categories are explained as follows:

- **Search Time**: Describes the time the planner took output a plan in seconds. Labeled as *timeout* if the planner took more than 120 seconds.
- **Plan Duration**: Describes the length of the outputted plan in hours.
- **Plan Quality**: Describes, subjectively, the quality of the outputted plan. This was done on a case-by-case bases. We often looked at how well the planner avoided long and expensive transportation trips and how well it planned for distribution by loading multiple items onto a vehicle.

In general, default enhsp-2020 found higher quality plans than its short-sited counterpart. One exception does exist in the form of the problem *relaxed-FlatLine*, where the default planner took the expensive path, resulting in a worse plan. Moreover, both versions struggled heavily when presented with any problem requiring more than one day of simulation.

Table 1
Short-Sighted Planner Performance

	Search Time	Plan Duration	Plan Quality
Relaxed-Choke	1	11	1
Relaxed-Flat	7	11	4
Relaxed-Diamond	7	24	1
Moderate-Choke	38	18	2
Moderate-Flat	13	26	3
Moderate-Diamond	Timeout	Timeout	Timeout
Strict-Choke	16	11	1
Strict-Flat	3	26	1
Strict-Diamond	27	28	2

Table 2
Default Planner Performance

	Search Time	Plan Duration	Plan Quality
Relaxed-Choke	16	3	5
Relaxed-Flat	46	15	3
Relaxed-Diamond	120	3	5
Moderate-Choke	Timeout	Timeout	Timeout
Moderate-Flat	Timeout	Timeout	Timeout
Moderate-Diamond	Timeout	Timeout	Timeout
Strict-Choke	40	3	5
Strict-Flat	45	3	5
Strict-Diamond	Timeout	Timeout	Timeout

Short-sighted enhsp-2020 was able to find plans much more quickly than the default enhsp-2020, but the plans were of low quality, as the planner almost never loaded more than one item onto a vehicle, resulting in essentially useless plans. On the other hand, default enhsp-2020 did find useful high-quality plans, but was often overwhelmed and could not find a plan within reasonable time. Although we label a *timeout* to be more than 120 seconds, in reality default enhsp-2020 failed to find plans for any moderately complicated problem setting even when given 10 minutes.

As an example, we show the output of default and short-sighted enshp-2020 on *Relaxed-Choke*. Observe that while the default version buys multiple items for its trip, the short-sited version buys only one item at a time.

default enhsp-2020 on Relaxed-Choke:

```
0.0: (buy_Item s v2 i2)
0.0: (buy_Item s v2 i2)
0.0: (buy_Item s v2 i2)
0.0: (buy_Item s v2 i2)
0.0: (begin_Drive s w2 v2)
0.0: -----waiting---- [1.0]
1.0: (end_Drive s w2 v2)
1.0: (unload_Item w2 v2 i2)
1.0: (sell_Item w2 i2)
1.0: (unload_Item w2 v2 i2)
1.0: (begin_Drive w2 w1 v2)
1.0: -----waiting---- [3.0]
3.0: (begin_Drive w1 w2 v1)
3.0: (sell_Item w2 i2)
3.0: (end_Drive w2 w1 v2)
3.0: (unload_Item w1 v2 i2)
3.0: (sell_Item w1 i2)
3.0: (unload_Item w1 v2 i2)
3.0: (sell_Item w1 i2)
```

short-sighted enhsp-2020 on Relaxed-Choke:

```
0.0: (buy_Item s v2 i2)
0.0: (begin_Drive s w2 v2)
0.0: -----waiting---- [1.0]
1.0: (end_Drive s w2 v2)
1.0: (unload_Item w2 v2 i2)
1.0: (sell_Item w2 i2)
1.0: (begin_Drive w2 s v2)
1.0: -----waiting---- [2.0]
2.0: (end_Drive w2 s v2)
2.0: (buy_Item s v2 i2)
2.0: (begin_Drive s w2 v2)
2.0: -----waiting---- [3.0]
3.0: (end_Drive s w2 v2)
3.0: (unload_Item w2 v2 i2)
3.0: (sell_Item w2 i2)
3.0: (begin_Drive w2 s v2)
3.0: -----waiting---- [4.0]
4.0: (end_Drive w2 s v2)
4.0: (buy_Item s v2 i2)
4.0: (begin_Drive w1 w2 v1)
4.0: (buy_Item s v2 i1)
4.0: (begin_Drive s w2 v2)
4.0: -----waiting---- [5.0]
5.0: (end_Drive s w2 v2)
5.0: -----waiting---- [6.0]
6.0: (unload_Item w2 v2 i1)
6.0: (sell_Item w2 i1)
6.0: (end_Drive w1 w2 v1)
6.0: (begin_Drive w2 w3 v2)
6.0: (begin_Drive w2 s v1)
6.0: -----waiting---- [9.0]
9.0: (end_Drive w2 w3 v2)
9.0: (unload_Item w3 v2 i2)
9.0: (sell_Item w3 i2)
9.0: (end_Drive w2 s v1)
9.0: (buy_Item s v1 i1)
9.0: (begin_Drive s w2 v1)
9.0: -----waiting---- [10.0]
10.0: (end_Drive s w2 v1)
10.0: (unload_Item w2 v1 i1)
10.0: (sell_Item w2 i1)
```

5. Summary

We created a hybrid planning model that captures the essence of multi-warehouse stock management. Our model incorporates an arbitrary number of warehouses, suppliers, vehicles, and inventory items, while considering costs associated with transportation and inventory, as well as an assumed demand. Our model attempts to maximize the profit of running our warehouses by maximizing revenue and minimizing the cost of management.

We tested our model using default and short-sighted enhsp-2020 on nine total problem settings, which consisted of three warehouse graphs that each have three goal states: relaxed, moderate, and strict. In the relaxed goal case, we only have two items and the net profit required is meager. The moderate goal state has more items, a more strict profit margin, and a longer time constraint. Finally, the strict goal state has limited items and a short time constraint, but a significant profit margin is required.

We concluded that the default version produces acceptable plans but is largely unable to produce plans for complex problems within a reasonable time. Meanwhile, short-sighted enhsp-2020 does produce plans within a reasonable time but the plans produced are of low quality.

5.1. Future Work

Based on the results, we conclude that hybrid planners, or at least hybrid planners of comparable performance to enhsp-2020, are still unable to efficiently solve moderately complex warehouse management problems. We suggest that future work on warehouse planning management take one of two routes.

The first approach would be to focus on the improvement of hybrid formalism planners. In this route, it is perhaps more realistic to focus on creating planners designed to specifically tackle warehouse management. It is also possible that through the exploration of the inner workings enhsp-2020, we could find out ways to optimize the planner so that it may be better suited for warehouse management problems.

The second approach would focus on simplifying the warehouse management problem via abstraction and segmentation of various parts. Abstraction would entail that we take away certain details from the model, making it less realistic. The other option would be to segment the model into a few separate models, each solving a small part of warehouse planning. For example, one such segmented model could be used to manage the vehicle fleets. This would result in less holistic plans but may be able to solve the problem settings far more efficiently.

References

[1] M. Crosby, R. P. Petrick, Temporal multiagent planning with concurrent action constraints, in: Proc 2nd ICAPS Workshop on Distributed and Multi-Agent Planning. ICAPS, 2014, pp. 16–24.

[2] C. Hütter, More shuttles, less cost: energy efficient planning for scalable high-density warehouse environments, in: Proceedings of the International Conference on Automated Planning and Scheduling, volume 26, 2016, pp. 403–411.

[3] C. Perez, M. A. Salido, A grasp-based search technique for scheduling travel-optimized warehouses in a logistics company. (2020).

[4] E. Scala, P. Haslum, S. Thiébaux, M. Ramirez, Subgoaling techniques for satisficing and optimal numeric planning, Journal of Artificial Intelligence Research 68 (2020) 691–752.