Algorithms for Decision Support: Lab Project 24-25
Box pickup and delivery problem for a warehouse setting

Annemie Vorstermans
Manos Thanos

---

1. Problem description

1.1. General Setting

The problem we consider involves a warehouse setting where a set of identical container boxes are stored in stacks among a single aisle, as shown in Figure 1. Within the setting there exists one buffer point serving both as an entrance and exit, as shown in Figure 2. This is where boxes arrive and from where they are retrieved to be stored in stacks, as well as where boxes must be transferred to before being delivered to customers. Stored boxes form multiple stacks at fixed locations throughout the aisle. Box transfers are executed by a limited number of small, identical vehicles V. These vehicles move forward and backwards along the aisle with a known speed u, while sequentially performing box pickups and deliveries.

Figure 1: Warehouse setting.

Transport requests R signify the need for a box to be stored or sent out. A request $r \in R$ is thus defined by a tuple $< s_r^{pu}, s_r^{pl}, b_r >$, where s_r^{pu} is r's pickup location, being either the buffer point or a storage stack, s_r^{pl} is the place location, being either a storage stack or the buffer point, respectively, and $b_r \in B$ is the associated box.

Each request r must be assigned to exactly one vehicle $v \in V$ which must travel from its current location s^v to s_r^{pu}, load b from the top of the associated stack, travel from s_r^{pu} to a delivery location s_r^{pl} and finally unload b atop s_r^{pl}. Both loading and unloading function durations are considered equal to a predefined duration L.

1.2. Constraints

Boxes must be located atop their respective stack when being picked up. Therefore, a **relocation** is generally required when a box located lower in a stack must be sent out, or when there is no space

within a target stack to store a box. The relocated box can be transferred at any stack which is available at the time of its relocation.

Vehicles have a limited capacity c^v, meaning that they are not allowed to carry more than c^v boxes at any time.

Stacks also have a limited capacity c^s, meaning that they are not allowed to carry more than c^s boxes at any time.

When multiple vehicles are employed, it is not allowed for more than one vehicle to perform (un)loading functions on the same stack. This is not the case for the buffer point. In addition, vehicles are allowed to move freely in all directions along the aisles. They are not hindered by other vehicles moving in the same or opposite direction along this aisle.

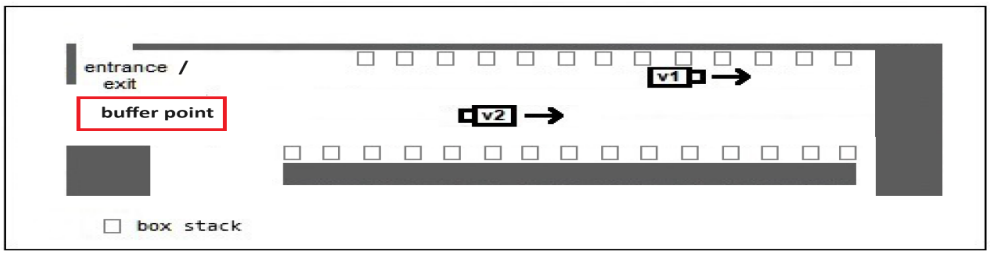

Figure 2: Network configuration.

1.3. Network characteristics

- (un)loading function duration,
- vehicle speed,
- entrance/exit position

1.4. Instance characteristics

- stack positions (x,y),
- available vehicles with their initial positions,
- initial stacks (box IDs),
- vehicle capacity,
- stack capacity,
- set of transfer requests R

Given a setting network, a set B of boxes, a set V of vehicles and a set R of requests with the aforementioned characteristics, the problem consists in finding a valid schedule for performing all requests.

2. Deliverables

2.1. Model to be gradually developed

In order to tackle the problem you must design and implement a model which can effectively address diverse instances. Each instance type may raise various challenges focusing on different problem features (i.e., number of boxes/stacks/requests, vehicle congestion). Your model must therefore be built in a manner which is both sufficiently generic and able to efficiently handle possible challenges.

A **provisional** development of the model could gradually built on the following components:

1. Network and stacks,
2. Vehicle moves: picking up and delivering a box,
3. Updating stacks,
4. I/O, (should be completed by Lab 2)
5. Vehicle moves: timestamps and positions,
6. Vehicle moves: handling multiple boxes,
7. Employ multiple vehicles (Lab 3),
8. Identify and apply relocations,
9. Multiple vehicles: Avoid conflicts in stacks (Lab 4),
10. Generate a valid complete schedule

A first indicative set of instances and output will be available before the **2nd lab** (23/10). An additional set of diverse instances will then be available after the **3rd lab** (13/11).

2.2. Challenges and statistics

Your model must be flexible at any step of the modeling process. As is often the case in industrial context, slight modifications may come up at any point and they must then be handled by the existing model. For example, such a challenge could involve the addition of an aisle or a buffer point. You should thus make sure that your model is not restricted by the very specific instance attributes, unless speedup opportunities exist for your developed algorithm(s).

In addition, you are responsible for evaluating your algorithm's performance during the implementation process. You should therefore keep track of indicative metrics related both to the generated solutions and its runtime behaviour for different instances/instance types.

2.3. Reports and Evaluation

Apart from permanent evaluation (presence, participation, work ethics) you will be evaluated based on the following submissions (see related deadlines in Table 1):

- **Intermediary Report**: A short report (up to 3 pages) with your current model. Within this report you must identify the **most significant components** of your model and present the specific data structures you utilise for each of them. Your choices must be justified based on both technical aspects and problem-specific features.
- **Current Codebase**: Your current implementation along with input/output file examples and necessary specifications.
- **Final presentation**: During the last lab session you will defend the final design of your model with a short presentation (5 minutes) in front of the whole group. This should focus on important aspects of your model, a reflection on the pros and cons of your implementation and the evaluation of your algorithm using experimental benchmarks.
- **Final report and code submission**: You will have to submit a complete technical report (up to 6 pages), ideally building on the intermediary one. Apart from updating and finalising your model and implementation status, this should additionally include a critical reflection of your final deliverable along with experimental evaluation/benchmarking. Explain whether your choices were justified by your final results and why. Finally, your final code should also be submitted, along with an executable (.jar) file and detailed running specifications (version, how to call, arguments, output, ...).

Lab session	**Date**	**Deliverable**
3	13/11	Intermediary Report
5	11/12	Current Codebase
Last	TBD	Presentation
-	TBD	Final report + codebase

Table 1: Deliverables and Deadlines

More detailed guidelines will be given before the deadline of each deliverable.

Please note that reports must be concise, informative and well structured. You will also be evaluated on the way you communicate your results and conclusions, so please do not deliver a document that you would not enjoy reading yourselves. General rule: keep it short!