Projekt: Order batching and robot routing in (half-)automated warehouses

June 29, 2021

Leaderboard deadline: 08.08.2021 (source codes of python & table of results)
Project due: 22.08.2021 with documentation

1 Problem Definition

The company LüneRobotics provides robots for an AGV-assisted warehousing system that enables the cooperation of human pickers and transport robots to finish picking for the customers' orders. Each customer's order includes several SKUs (Stock Keeping Unit) with their given ordered quantities. For example, we have order 1: $o_1 = (a, 2), (b, 1), (c, 3)$ (SKU, ordered quantity). Each picking item has a weight. The orders are known in advance.

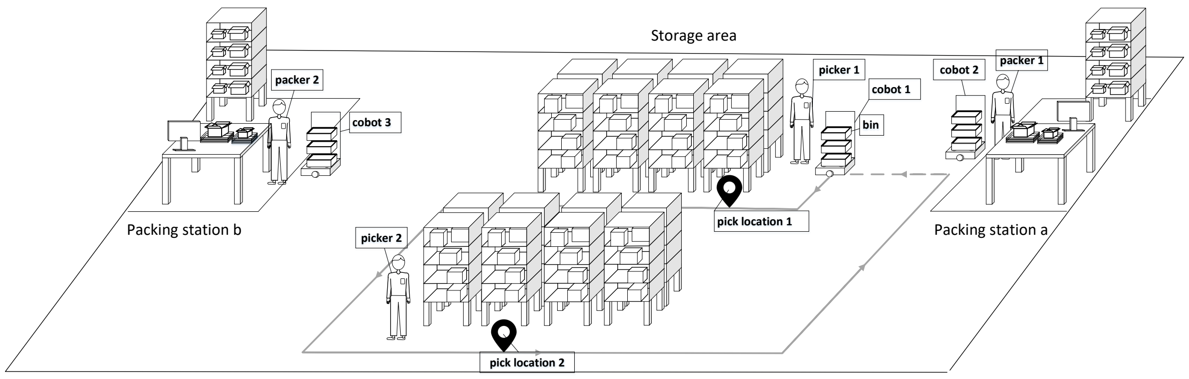

Figure 1: An example of a (half-)automated picking system (with two packing stations).

In AGV-assisted warehouses (see an example in Figure 1), mobile robots (called *cobots*) go automatically to a pick location in the storage area and wait for a human picker to arrive. Pickers identify and grasp ordered items from shelves and put them into bins carried by the cobots; after that, the cobots go to their next picking locations, while the human pickers remain in the storage area. After collecting, each cobot goes back to its packing station to drop off the loaded bins. In the example in Figure 1, cobot 1 leaves packing station a and goes firstly to pick location 1 waiting for picker 1 to load items; after that, it goes to pick locatin 2 waiting for picker 2 to load items and returns at packing station a.

Following assumptions are made in this projekt:

- There are several shelves in the storage area. The items are located in shelves. If one SKU is stored in a shelf, then all items of that SKU are stored in that shelf. We call this strage policy as *dedicated policy.* Note that there might be more than one SKU stored in a shelf.

- Each picker in the storage area works within a given area, called *zoom*. For example in Figure 1, the upper area in the storage area is a zoom, where picker 1 works. Once a cobot arrives at a picking location in a zoom, it needs to waiting for the picker. The waiting time need to be calculated depending on where the picker is. But the average walking speed of a human picker is given, i.e. 1.3 m/s. The picking time of each picker for a single item is given as well, 3 seconds.

- If more than one cobot are waiting within a same zoom for a human picker. Then we will follow the rule *first come, first served*, i.e. the human picker will follow the first incoming cobot to pick all required items in this zoom then he/she will go to the location of the next cobot, and so on. The cobots are waiting at their first picking locations in a zoom.

- If a cobot leaves its packing station, then it should go back to the same packing station. Each cobot belongs to a certain packing station. The cobot is charged in its packing station. But we ignore the charging time in this project. We assume that each cobot leaves the station has enough battery to finish a tour. The average moving speed of each cobot is given, 2.0 m/s.

- All items of an order should be picked within one tour of a cobot. But it is not neccessary to finish picking orders one by one. For example, we have two orders 1 and 2 within one tour. Order 1 has items (a_1,2) and (a_2,1), while order 2 has items (b_1,1) and (b_2,3). The cobot can first pick up items a_1 and items b_1, after that b_2 and a_2, if this sequence brings better fittness value (shorter routing time).

- A *bin* contains items of an order. We assume a cobot can carry as many bins as possible as long as the load capacity (in this project: 18kg) is not violated. Each tour is limited by the load capacity. One tour is called as a *batch*.

- A human packer works in a packing station. Once a cobot has arrived in the packing station the human packer takes all bins from the cobot (*unload time*: 20 seconds) and put new empty bins to the cobot (*preparation time*: 30 seconds). Note that there is only preparation time for the first leave of each cobot. After that, the cobot is ready to leave for the next tour. Then the packer begins to pack the orders. The packing time of each order is predefinied, 60 seconds, including choosing a box, putting items from a bin to the box, packing the box. We assume that there are more than one packing stations.

- Each shelf and packing station has a location. The distance between each two locations is given.

- The number of cobots from each stations as well as the number of stations are given.

- We assume that the inventory has enough items for picking.

The goal of this problem is to minimize the makespan, while the items of all orders are picked and transported to packing stations. The *makespan* is definied as the time that the last order is finished by packing.

You have at least the following **decision problems** to achieve that goal:

- Assign orders to stations
- Assign orders to batches of cobots
- Determine the sequencing of visiting pods for each cobot within a route

2 Tasks

2.1 Heuristic design (20P)

1. Design a greedy heuristic to create a solution to the given problem.

2.2 Neighborhood design (40P)

1. You are writing a simulated annealing algorithm to solve the given problem. Design a neighborhood to generate a solution s' given a solution s.
2. Argue why your neighborhood can (or cannot) lead to an optimal solution starting from any solution in the solution space.
3. Design a perturbation strategy for an iterated local search and argue why it will (or not) work as intended.

2.3 Extension (40P)

1. Assuming that the items of an SKU can be stored in different shelves and different SKUs can stored within a shelf. We call this storage policy as *mixed policy.* Extend your implemented heuristics in the previous subsections to support this assumption.
2. Design an **adaptive large neighborhood search** to solve the extended problem.
3. Argue why your adaptive large neighborhood search can (or cannot) lead to an optimal solution starting from any solution in the solution space.
4. Describe the difficulties to apply mixed policy, e.g. which decision problem is more complex? And describe the befinits of applying this policy.

3 Instances

Note that there are more information in data as needed in the project. I will list only the useful information below.

Following instances are required for the project:

- in sku24 and sku360 files
  - **layout**: you can find two different layouts with 24 and 360 pods in `layout_sku_24_2` and `layout_sku_360_2`, respectively. You can find an .png to see each layout visually. In the .png you can see the predefined zones.

  - **pods_infos**: the 1st and 2nd columns are id and (x/y)-coordination, respectively.
  - **pods_items_...**: the items in pods (id; x/y; color/letter/quantity), you can find both files for the dedicated and mixed storage policies. Items are generated based on the combination of colors and letters.
  - **orders_...**: items are describted in `ItemDescriptions` including color, letter and weight, while orders are describted in `Orders` including items and count (quantity).

- **rafs_instance_init_.py**: the class definition of each component in the warehouse. The class `Warehouse` includes layout, instance, pods and orders.
- **instance_demo.py**: an example of processing warehouse data, including calulation of distances between two nodes, getting useful pods (which includes items of given orders).

Note that you can use `orders_10_mean_1x6_sku_24` in sku24 file to do debegging. The optimal driving distance to fulfill this order list is 59.2. Please extend your codes in `instance_demo.py`.

3.1 Output

Please use the format as `log_example.xml` to output your results.

3.2 Documentation

Please use the template in `Latex-Vorlage` to document your heuristics and results.

4 Grading

The project will be evaluated based on the following criteria:

- Solution is executable and keeps the constraints (correctness)
- The codes with comments in Python should be understandable
- Ideas of the heuristics (documentation with pseudocodes) and quality of the implementation