IA – Homework 4 – Report

Group 23 – Quentin Mazars-Simon & Rivo Vasta

Model description

Multitasking: focus on actions rather than on tasks

The original model of the paper was task-oriented: the optimal plan is an ordered list of tasks assigned to each vehicle. This is typically monotask: as a task is a pair of pickups and deliveries and as the vehicle executes the tasks in the order, it will pick up the task and no other until it has delivered it.
Our approach gets nearly completely rid of tasks. We consider actions only, where an action is the fact of picking up a task, or deliver it. This model allows us to put several pickups in a row, and several deliveries in a row, and more generally to have pickups and deliveries actions in any order, as long as the capacity of the vehicle is not exceeded and a delivery action of some task is not ordered before the pickup of this task.
This is implemented with the help of an actions list attached to a solution (see below). Each action keeps with it its type (pickup or delivery), the attached city, and the original task it came from (so as to not get a delivery action of some task be scheduled before the pickup of the same task).

Generating plans

To find the optimal plan, the SLS algorithm iteratively generates a new plan and makes a decision on the local best plan, for a certain number of times.

Solution

A valid solution is a plan that respects the constraints. The initial solution is to give every task of the world to the biggest vehicle.

Choosing neighbours

This is where the algorithm generates neighbouring solutions and makes a local decision. When generating a plan by changing a vehicle and the actions order, we generate a complete list of solutions by considering each vehicle and each new actions order. We then have some randomization when we have to choose between several plans with same cost.

Changing vehicle

We loop on every vehicle, generating a new plan by giving a task (its associated pickup and delivery actions) to another vehicle. For each of the vehicle, we consider every pickup actions, find the corresponding delivery action, and remove them from the vehicle actions list. We then assign them to the second vehicle at random positions, and if the corresponding plan verifies the constraints, it becomes a solution and is added to the list of solutions that will eventually be returned.

Changing task order

We loop on every pair of actions and switch them. If the corresponding plan verifies the conditions, it becomes a solution and is added to the list of solutions that will eventually be returned.

Constraints

We have two constraints:
The first one is simply that at any moment, the carried weight of the considered list of actions attached to some vehicle cannot overload its capacity.
The second one is that in the list of actions, every pickup action must be placed before the corresponding delivery action. Moreover, every pickup action must have its delivery counterpart, and same goes for every delivery action. This is simply implemented by a sort of stack recording every task corresponding to each pickup action met, and removing every task corresponding to each delivery action met.
At the same time, we verify that all the tasks of the world have indeed been delivered.

Results

Here are the comparison of the plans' cost depending on the number of task and vehicles.

Tasks\Vehicles	1	2	3	4	Naive
10	~11000	~11000	~11000	~11000	24029
20	~20000	~20000	~20000	~20000	44214
30	~36000	~25000–30000	~25000–30000	~25000–30000	68731
40	~45000	~32000–37000	~32000–37000	~32000–37000	98896

We can see that increasing the number of vehicles after 2 does not improve the results. This is because, as we will see below, our algorithm is unfair and only two vehicles are used.

For example, for 4 vehicles and 30 tasks we have the following distribution: first vehicle has 23 tasks, second has 7 and third and fourth have none. This is due to the fact that we have 30 tasks with a weight of 3, and vehicles with a capacity of 30, which means that only two vehicles can efficiently deliver the tasks. If we increase the weight of each task, or reduce the capacity of each vehicle, the distribution will be fairer as more vehicles will need to contribute, as it can be seen with a task weight of 15: V1 has 22 tasks, V2 and V3 have 4.

Conclusions

The performance of the centralised management is significantly more important than the random agent: we have a total cost usually half as expensive as the naive implementation. However, the general attribution of the tasks is quite selfish and everything but uniformly distributed.