Excercise 4
Implementing a centralized agent

Group №21: Dubravka Kutlešić, Nevena Drešević

November 3, 2020

1 Solution Representation

Assume that the company owns n vehicles $V = \{v_1, ...v_n\}$ and has to deliver tasks $T = \{t_1, ..t_k\}$. For each task, two task models are assigned. For a task t, the task model is a pair (t, o), where $o \in \{p, d\}$ for corresponding pickup and delivery operation. There are $2k$ task models in the problem and the set of all task models is denoted as TM. For a task t, task models (t, p) and (t, d) are called task model pair.

1.1 Variables

For each vehicle $v_i \in V$, there is an array M_i of $2k$ variables. Let $M = (M_1, ..M_n)$. M_i represents the order in which v_i picks up and delivers tasks. Each value in each M_i is either some task model or *null* i.e. the domain for each variable is $D = \{(t, o) : t \in T, o \in \{p, d\}\} \cup \{null\}$. The meaning of *null* value is that the agent has finished with the deliveries. Function $load : TM \cup \{null\} \rightarrow \mathbb{R}$ is defined as following:

$$load(x) = \begin{cases} t.weight,\ if\ x = (t, s)\ for\ some\ task\ t \\ -t.weight,\ if\ x = (t, d)\ for\ some\ task\ t \\ 0,\ if\ x = null \end{cases}$$

1.2 Constraints

- Each task model must be contained in M and it is contained only once (This guarantees that all tasks are picked up and delivered and each operation is done only once)
- Both task models in the task model pair must be contained in the same M_i (The same vehicle picks up and delivers the task)
- For task model (t, s), its index in some M_i array is less than its task model pair index in M_i (The vehicle must first pick up a task and then deliver it)
- For all M_i, there does not exist task model such that its index in M_i array is greater than any index of a *null* variable in M_i (*null* value represents that the vehicle has finished with the delivery and it is not possible to finish with the delivery and then pick up or deliver a task)
- For each $i \in 1, ..n$ and each $j \in 1, ..2k$ the sum of *load* values of first j elements in the array M_i can not exceed the capacity of the vehicle v_i (At any point, the load of the vehicle's can not exceed the vehicle's capacity)

1.3 Objective function

Let $t.p$ and $t.d$ be a pickup city and a delivery city for a task t. Let $sd : Cities \times Cities \rightarrow \mathbb{R}$ be the shortest distance between two cities. Function $dist : (TM \cup \{null\}) \times (TM \cup \{null\}) \rightarrow \mathbb{R}$ is defined as:

$$dist(x, y) = \begin{cases} sd(t_1.o_1, t_2.o_2),\ if\ x = (t_1, o_1)\ and\ y = (t_2, o_2) \\ 0,\ if\ y\ =\ null \\ an\ arbitrary\ real\ value,\ otherwise \end{cases}$$

Function $initial\_dist : V \times (TM \cup \{null\}) \to \mathbb{R}$ is defined as:

$$initial\_dist(v, x) = \begin{cases} sd(v.HomeTown, t.s),\ if\ x = (t, s)\ for\ some\ task\ t \\ 0,\ if\ x = null \\ an\ arbitrary\ real\ value,\ otherwise \end{cases}$$

Let $M_i[j]$ be j-th element in the array M_i. The cost of the company is defined as:

$$C = \sum_{i=1}^{n} v_i.costPerKm \left(initial\_dist(v_i, M_i[1]) + \sum_{j=2}^{2k} dist(M_i[j], M_i[j-1]) \right)$$

2 Stochastic optimization

2.1 Initial solution

An Initial solution can be generated using one of the following approaches:
- **biggest vehicle**: Assigning all tasks to be carried sequentially by the vehicle with the highest capacity.
- **home city**: For each city c, we determine a vehicle whose home town is closest to c. If two vehicles have the same home town, the priority is given to a bigger vehicle. First, we try to fit as many tasks to these vehicles in the way that certain vehicle will first pick up all the tasks and then deliver them. After, the rest of the tasks are randomly assigned to vehicles and appended at the end of the current plan.

2.2 Generating neighbours

For each potential solution the total number of neighbors is determined by parameters α, β (read from the configuration file), k and n. There exist two transformations for generating the neighbor of the current solution. One is exchanging the order of any two task models that belong to the same vehicle. This operation can happen only if the order change does not violate the constraint that each task is picked up before being delivered. The number of neighbors generated by this transformation is bounded by $n\alpha$. The other transformation takes one task from a vehicle and gives it to some other vehicle (if $t.weigh < v.capacity$). Choosing of a task and vehicles is done randomly while the given task is positioned at the end of the task list for the other vehicle i.e. all previous tasks will be delivered, then the given task will be picked up and delivered. Number of neighbors created in this way is at most $k\beta$.

2.3 Stochastic optimization algorithm

With probability p the search is continued from the best neighbor of the current solution in terms of the cost function C, otherwise, we continue with the current solution. During the search, the best solution encountered so far is stored and returned at the end. The duration of one iteration depends on n, k, α and β. The maximal duration of one iteration is approximated using the values of $\alpha = 4$ and $\beta = 0.4$, 200 agents and 1000 tasks. It lasts less than $0.5s$, so $1s$ are left as a buffer. The algorithm will terminate before the defined timeout, having $0.5s$ to generate the optimal plan. This approach exploits allocated time for searching because it is not certain to estimate when the algorithm is in the state of local minimum and will not escape it in the future iterations.

3 Results

3.1 Experiment 1: Model parameters

3.1.1 Setting

In the experiment, the performance of the search is examined comparing different values for p, α and β. The England topology is used and the rngSeed is 12345. The total number of tasks is 20 and the number

or vehicles is 4. Maximum execution time of the setup is 1 second, while the planning can take up to 30 seconds. Initial solution is generated by "home city" approach.

3.1.2 Observations

	p	α	β	cost
set 1	0.4	4.0	0.2	11907
set 2	0.8	4.0	0.2	11979
set 3	0.2	4.0	0.2	12911
set 4	0.4	3.0	0.0	13175
set 5	0.4	0.0	0.8	17683

Table 1: Comparison of cost obtained for different values of p, *alpha* and *beta*

In test scenarios, all 4 vehicles are provided with tasks to carry. The optimal p, α and β values are 0.4, 4.0 and 0.2 respectively. This set of parameters leads to the best total cost, while we observe a higher cost when only one transition operation is used (sets 4 and 5). While experimenting with this setting we also observed that the probability p does not affect the optimal cost as much as tuning the number of neighbors generated by different transitions. Occasionally, it would happen that the same cost of an optimal solution is calculated with different values of p.

3.2 Experiment 2: Different configurations

3.2.1 Setting

This experiment examines the effects of initial solution approach and compares the performance for different number of tasks and vehicles. Switzerland topology is used. The probability p is 0.4, α and β are set to 4.0 and 0.2 respectively.

3.2.2 Observations

	initial solution	tasks number	vehicles number	agents	cost
set 6	biggest vehicle	20	4	3	11650
set 7	home city	20	4	3	10450
set 8	biggest vehicle	100	4	4	61450
set 9	home city	100	4	4	60350
set 10	biggest vehicle	20	8	4	13300
set 11	home city	20	8	3	10750

Table 2: Comparison of optimal cost and number of agents delivering tasks based on initial solution, amount of tasks and vehicles

The agent column represents how many vehicles carry tasks in the optimal plan. Usually, one vehicle carries most of the tasks in both initial solution approaches. For example, in set 7, vehicles carry 0, 1, 3 and 16 tasks. However, because the algorithm is stochastic and does not avoid local optima, it can happen that vehicles carry a similar number of tasks (especially in the "home task" approach). Overall, it can be concluded that optimality implies that some agents do more work than others.

Since the neighbors' processing depends on n and k, the iterations in the stochastic optimization algorithm last longer for sets 8, 9, 10 and 11, compared to sets 5 and 6.

Additionally, it can be observed that initialization with the "home city" approach performs better than "biggest vehicle".