Intelligent Agent Report
A Centralized Agent for Pickup and Delivery Problem

Team 08 Cheng-Hsiang Chiu & YiLin Hu

Introduction

The assignment is to implement a centralized agent to solve the pickup and delivery problem, which is described as a Constraint Satisfaction Problem (CSP), by applying Stochastic Local Search Algorithm (SLS). A centralized agent has an mechanism to coordinate multiple agents in order to achieve a common goal. .

Implementation

In order to transform the problem to CSP, the proposed way is similar to the one expressed in the reference document. And the only difference between the two methods is the nextTask variable which is consisted of two different arrays, called nextTask of a task, and nextTask of a vehicle. Since a vehicle is able to carry more than one task, we add *delivery* into the action sets and double the size of array of nextTask of a task. Then, the *pickup* action would be followed by either a *delivery* or another *pickup*. The array is visualized in the following figure. In Fig.1, next task of p1 is p2, next task of d1 is p3, and so on.

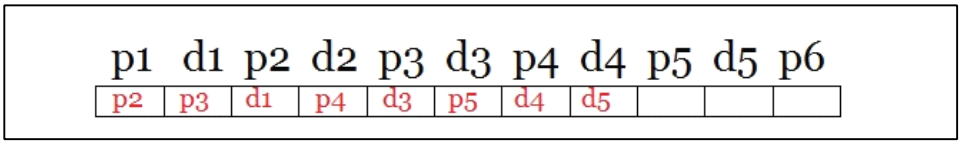


Figure 1. The demonstration of nextTask array.

As for choosing the initial assignment, vehicles are assigned the tasks which appear in the home city of the vehicle. After that, the rest of tasks are not assigned to vehicle 1 until remaining capacity of vehicle 0 is less than 0.

In the *ChooseNeighbor* procedure, we basically follow the pseudocode provided in the reference document except the two subroutines, *ChangeVehicle()* and *ChangeTaskOrder()*. In *ChangeVehicle* procedure, the first task pair which includes a *pickup* and its corresponding *delivery* of one vehicle, say v1, is reassigned to another vehicle, say v2. Additionally, the first *pickup* of v1 may not be followed by its correspond *delivery*, we need to iterate the whole array of v1 searching for the *delivery,* and update the first task to the consequent *pickup*. Now, the first task of v2 is replaced by the task pair from v1. The procedure could be visualized in Fig.2.

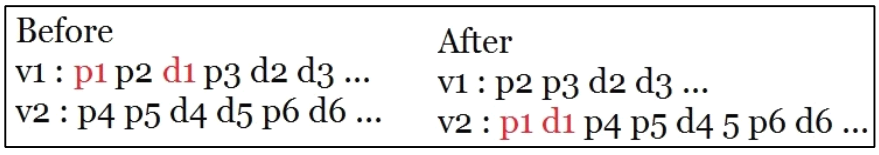


Figure 2. The array lists of v1 and v2 before and after *ChangeVehicle()*

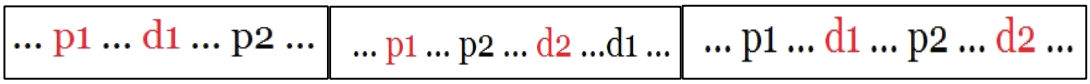
ChangeTaskOrder() is another different implementation. There are four strategies needed to be considered, *pickup-pickup*, *pickup-delivery*, *delivery-pickup*, and *delivery-delivery*. Among these strategies, *CapacityCheck()* is necessary to all and will be discussed in the next paragraph.

(1) pickup-pickup : Two pickups of one vehicle are selected to change their execution order. In most cases, this strategy is correct except one scenario which is shown in Fig. 3(a) where the corresponding *delivery*, *d1*, of a *pickup*, *p1*, can not locate between *p1* and *p2*. Otherwise, task1 will be delivered before being picked up.

(2) pickup-delivery : In Fig. 3(b), *p1* can not be switched with *d2* if *p2* lies in between.

(3) delivery-pickup : In this strategy, there is no way that a task is delivered before being picked up.

(4) delivery-delivery : In Fig. 3(c), d1 and d2 can not be switched, otherwise, task 2 will be delivered before being picked up.



(a) (b) (c)

Figure 3. The error scenarios that should be avoided.

The last critical procedure is *CapacityCheck()*. Even two tasks are changed successfully in the strategies mentioned above, the capacity might go wrong if weightings of two tasks are not identical. Hence, before *ChangeTaskOrder()* is returned, we check the capacity in the segment between the two tasks. Once, the capacity is failed to maintained, the two tasks are not switched.

Experimental Result

The experiment is conducted with several different scenarios. The termination condition is set to be 10000 iterations. P is equal to 0.5. The results are demonstrated in the following figures.

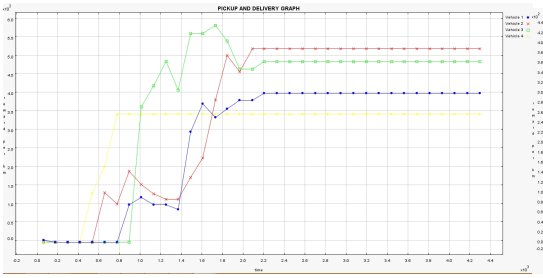


Figure 4. Four vehicles and 30 tasks.

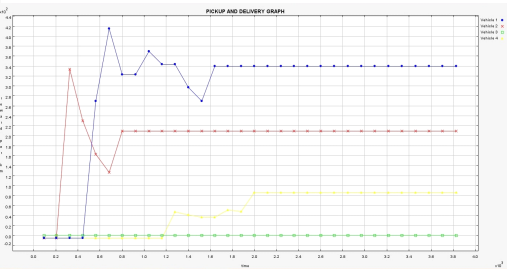


Figure 5. Four vehicles and 10 tasks.

Team 08 Cheng-Hsiang Chiu & YiLin Hu

Conculsion

Several scenarios are conducted in the experiments. In our implementation, local optimal is found with higher chance than global optimal. Some findings are worthy of mentioning.

Fairness is an issue in the optimal problem. Some vehicles are assigned more tasks than others. Sometimes, certain vehicles are not assigned any task. In Figure 5, there are 10 tasks sparsely spreading over 12 cities. In the initialization step, vehicle 4 does not get any task. And since local optimal is more likely to be found, vehicle 4 is not assigned any task at all. This is not fair for all the vehicles. Even in Figure 4, each vehicle gets some tasks to delivery, vehicle 4 still deliverys less tasks than the other vehicles.

The size of the data structure is O(Nt * Nv), where Nt is the number of tasks and Nv is the number of vehicles. Therefore, the complexity of our implementation is proportional to the O(Nt * Nv).

Centralized agents can obtain higher rewards because of coordination among vehicles. After completing the assignment, we learned how to interpret the Pickup and Delivery problem as a Constraint Satisfaction Problem and how to solve it by taking advantage of Stochastic Local Search Algorithm.