Centralized Agent

Liang Zixuan, Wang Junxiong 11/11/2015

Model

Variables

We have to assign tasks to each vehicle. This is represented in a hashmap. Map contains vehicles as keys. Value for any key is a linkedlist of states. In each state, the vehicle whether picks up a task or delivers a task. In this way, the linkedlist also represent the order of tasks for each vehicle. In order to achieve minimum total costs for all vehicles, we have to explore different ways of assigning and ordering the tasks for each vehicle.

Constraints

In the linkedlist of states of each vehicle, delivering a task must be after picking up this task. The same task cannot be delivered again. In the end all tasks have to be delivered. A vehicle can pick up multiple tasks simultaneously, but the total weight of tasks that are being picked up cannot exceed the capacity of the vehicle. The objective function as stated before is the total cost of all vehicles.

Implementation

We implement the stochastic local search algorithm following the guide from the paper.

Initialization

First we find the vehicle with the largest capacity. Then we assign all tasks to the vehicle one by one. In order words, the vehicle with the largest capacity delivers all tasks sequentially.

Choosing Neighbor Plan

We generate a neighbor plan by two ways. One is to take the first task from the tasks of one vehicle and give it to another vehicle. The other is to change to order of two tasks in the task list of a vehicle. In this way we generate a new plan set containing many neighbor plans.

Local Choice

After we have the neighbor plans set, we can choose a plan from the old plan and the neighbor plans with some probability. In our strategy, we set $p = 0.35$. Then we randomly generate

	20 tasks	25 tasks	30 tasks
3 vehicles	13717	14618	15359
4 vehicles	11713	14996	17300
5 vehicles	14118	15208	18524

Figure 1: Total cost of all vehicles for different tasks and number of vehicles

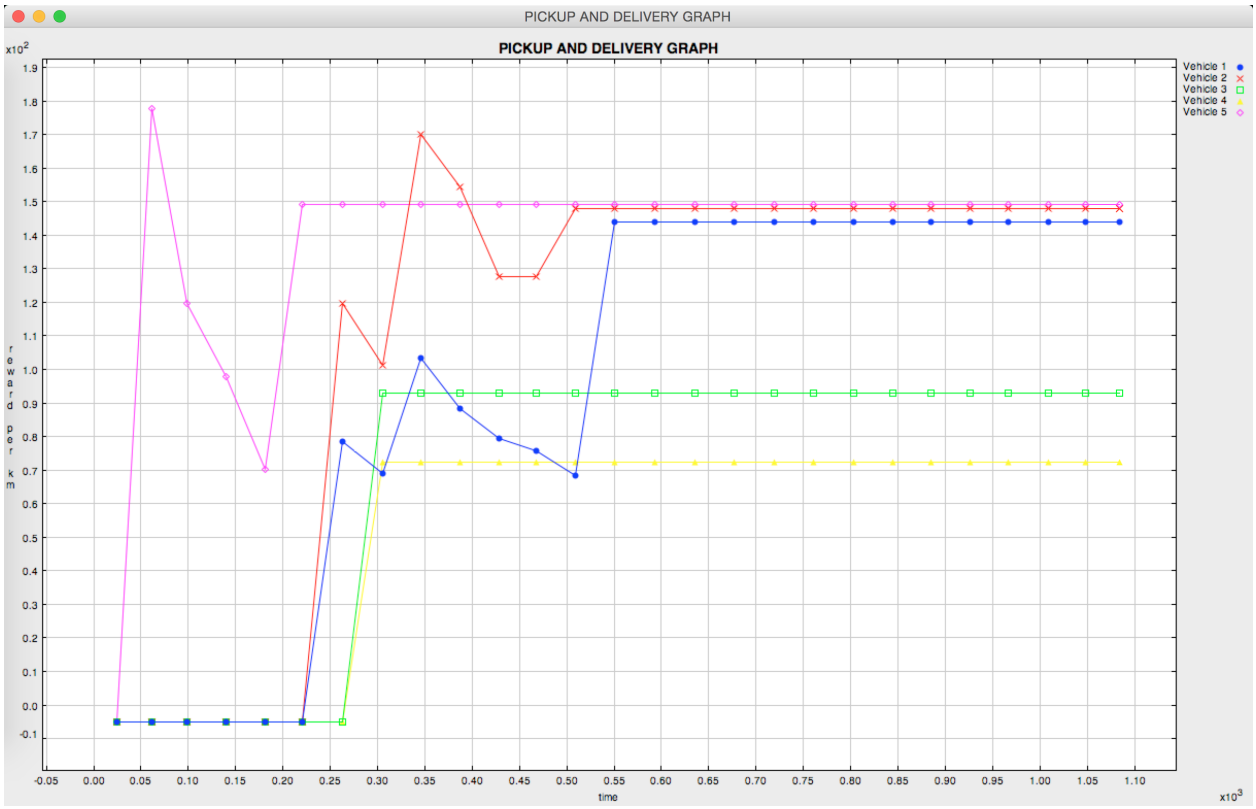
$num \in [0, 1]$. If $num < p$, we choose the plan with the smallest cost from neighbor plans set. If $p < num < 2p$, we stick with the old plan. If $p > 2p$, we randomly choose a plan from the neighbor plans set. In this way, we can sort of avoid being trapped in local minimum.
The algorithm terminates after specified number of iterations.

Evaluation of the Algorithm

We run simulations for different tasks and number of vehicles. Figure 1 shows the result of the simulations with iteration times limited to 2000.

Findings

The centralized agent implementation improves the cooperation between agents. Deliberative agents have a weak point when multiple agents are run. Collisions occur and plans have to be recalculated, and all that increases the costs for the company. Below we can see the two implementations run with 10 tasks and 5 vehicles, using England map. The centralized agent is a clear winner here. Vehicles have plans that do not interfere and that bring more profit to that company. We can clearly see that, in the example, optimality requires vehicle 1 and vehicle 2 to do all the work while vehicle 3, 4 and 5 don't do any work. Generally this phenomenon holds for other situations.



(a) Deliberative Agents

(b) Centralized Agents

Figure 2: Rewards per km for Multiple Agents of Deliberative Agents and Centralized Agents