A Centralized Agent for the Pickup and Delivery Problem

Deliberative agents are very efficient in executing a plan, as long as they are not disturbed by events that have not been taken into account in the plan. As you have seen in the previous exercise, the presence of a second deliberative agent can make the whole company very inefficient. The reason for this inefficiency is the lack of coordination between deliberative agents. In the case of multi-agent systems the agents' ability to coordinate their actions in achieving a common goal is a vital condition for the overall performance of the system. In this exercise you will learn to coordinate the actions of a number of agents in solving the PDP problem.

The simplest form of coordination is *centralized coordination*, in which one entity (e.g. the logistics company in our case) instructs the agents how to act. In our problem centralized coordination means that the company builds a plan for delivering all the packages with the available vehicles, and then communicates the respective parts of the plan to each vehicle. The vehicles simply execute the plans that were given to them.

Planning as constraint satisfaction problem

Building a plan for a number of agents is by no means a trivial task. The state-based search algorithms you have used in the previous exercise (i.e. building the plan for a deliberative agent) cannot be used, due to the large number of possible states that need to be explored. Instead, the planning problem can be expressed as a Constraint Satisfaction Problem (CSP) and solved using techniques tailored to this class of problems.

The document "Finding the Optimal Delivery Plan : Model as a Constraint Satisfaction Problem" describes a possible encoding of the PDP problem as constraint optimization problem. It then shows how to solve the problem using the stochastic local search algorithm. Note that the paper makes an important simplification that vehicles can only carry one task at the time. The first part of your assignment is to generalize this model by allowing vehicles to carry multiple tasks as long as the sum of their weights is below the total capacity.

Stochastic local search

In this exercise you will implement the stochastic local search algorithm to find an efficient solution to the CSP description of the PDP. You will find a sketch of the algorithm in the document above. Note that there are many ways to encode a problem using constraints. You are encouraged to come up with you own solution rather than following the paper exactly.

Don note, however, that you are required to create an algorithm where trucks are allowed to carry **multiple** packets at the same time.

Your task

1. Implement the Stochastic Local Search algorithm for the PDP.
2. Run simulations for different configurations of the environment (i.e. different tasks and number of vehicles) in order to observe the behavior of the centralized planner using the SLS algorithm.
3. Reflect on the fairness of the optimal plans. Observe that optimality requires some vehicles to do more work than others. Illustrate this observation with an example in your report.
4. Test your program for different number of vehicles and various sizes of the task set. How does the complexity of your algorithm depend on these numbers ?

Implementation Hints

— You will need to implement the `CentralizedBehavior` interface and compute the plans for several vehicles. The plan for each vehicle has to be returned as a list in which the plans appear in the same order as their corresponding vehicles in the vehicle list.

— When you design your stopping criterion, make sure that you don't use more time than the platform timeout (see starter code).

— Again, if the *LogistPlatform* detects a problem in your plan, the simulation will exit and a detailed error message is displayed. Remember that you have to handle all tasks and that each tasks can only be picked up and delivered by one vehicle.

— There is a separate document that elaborates the relevant parts of the *LogistPlatform* for this exercise in more detail.