Assignment 4

Programming: Implementing Centralized Coordination

Intelligent Agents Course 2009/2010

Radoslaw Szymanek
Brammert Ottens

G10

Bello Ruiz, Javier
Pocard Du Cosquer De Kerviler, Tugdual

A Centralized Coordination for the Pickup and Delivery Problem

As we saw with deliberative agents, there is a lack of coordination when they are more than two, making the company very inefficient in terms of performance and in distance traveled. In this new programming exercise we have seen that, in the case of multi-agent systems, one of the most important things is the ability to coordinate them in order to achieve the goals with good performance.

We worked in the *centralized coordination*, in which the company develops a plan for delivering the tasks considering all the vehicles and then communicates the respective parts to each vehicle. Due to the large number of possible states needed to be explored we have to develop, instead of a state-based search algorithm, a *Constraint Satisfaction Problem* (CSP). For solving this type of CSP we have to work on a stochastic local search algorithm (SLS) for the simplified version of the PDP.

The solutions given by this algorithm can produce solution with multiple tasks per vehicle if they fit in it capacity. In other words, the SLS will be solving a more general version of the problem. In order to compare our non optimal cooperative planner, we have access to a central planner (`OptimalCooperativePlanner`) that finds an optimal solution to this description. This planner uses an important simplification that vehicle can carry only one task at the time if it is below its total capacity. The goal of our company is to determine a plan for each vehicle in order that all tasks assigned are delivered, vehicles carry out tasks sequentially and the total revenue is maximized. Vehicles cannot carry out tasks in parallel; a delivery plan for a vehicle is uniquely characterized by the delivery sequence of the assigned tasks.

As for the CSP we have the following variables:

· V = {v1, v2, . . . vNV} set of vehicles owned by the company.

· T = {t1, t2, . . . tNT} set of tasks to be delivered.

nextTask -array of NT+NV variables. The array contains one variable for every existing task, and one variable for every existing vehicle, they can take as a value another task, or the value NULL.

time -array of NT variables. The array contains one variable for every existing task; they can take an integer value specifying the delivery sequence number of the task in the plan of a certain vehicle.

vehicle -array of NT variables, one variable for each task, they can take as a value the code of the vehicle that delivers the corresponding task.

In the CSP problem we have several constraints that have to be considered, and we have the function of the total cost that has to be minimized:

The total cost of the company is defined as:

$$C = \sum_{i=1}^{N_T} \Big(dist\big(t_i, nextTask(t_i)\big) + length\big(nextTask(t_i)\big) \Big) \cdot cost\big(vehicle(t_i)\big) + \sum_{k=1}^{N_V} \Big(dist\big(v_k, nextTask(v_k)\big) + length\big(nextTask(v_k)\big) \Big) \cdot cost\big(v_k\big); \quad (1)$$

In order to minimize this function we considered the following stochastic algorithm to achieve the goal:

Algorithm 1 *SLS algorithm for COP*

procedure *SLS(X,D,C,f)*
  $A \leftarrow SelectInitialSolution(X, D, C, f)$
  repeat
    $A^{old} \leftarrow A$
    $N \leftarrow ChooseNeighbours(A^{old}, X, D, C, f)$
    $A \leftarrow LocalChoice(N, f)$
  until termination condition met
  return A
end procedure

The functions we wrote in the SLS algorithm used to calculate solutions of the CSP/COP are as they are written in the implementation hints document:

“SelectInitialSolution(): This function selects a complete, possibly random, assignment A of values to all variables that is consistent with the constraints.”

“ChooseNeighbours(): This function provides a set of candidate assignments that are close to the current one and could possibly improve it. In the simple case, they are generated by randomly selecting a variable xi ∈ X and generating all assignments that are equal to A but assign to xi different values in the domain of xi that are consistent with the rest of A according to the constraints.”

“LocalChoice(): It first selects the assignment A in the set of candidates that gives the best improvement of the objective function. If there are multiple equally good assignments, it chooses one randomly. Then with probability p it returns A, with probability 1 − p it returns the current assignment Aold. The probability p is a parameter of the algorithm. If p is close to 1, the algorithm converges faster but it is easily trapped into a local minima. A value of p from 0.3 to 0.5 would be a good choice.” Our p is 0.5

1. Run simulations for different configurations of the environment (i.e. different tasks and number of vehicles) in order to observe the behavior of the **optimal centralized planner**:

Number of agents	2		3	
Number of tasks	Cost	Time	Cost	Time
6	107190	681ms.	91020	847ms.
7	128985	1543ms.	113430	4019ms.
8	161325	6800ms.	145770	20678ms.
9	175275	72805ms.	159720	179156ms.

2. 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:

   It is not always true if we change the topology. For instance if we have a graph with a bridge (edge) which has a very long distance, two vehicles which begins on opposite parts from this bridge, and tasks which does not require to go through the bridge. Then it is possible to have a vehicle with higher cost per km which takes and delivers more tasks than the other. In our graph, it will be true for most of the task distribution (not for all) because it has a high connectivity. If a vehicle has a higher cost per km then an optimal solution will try to make it drive less km and obviously take fewer tasks.

3. Run simulations for different configurations of the environment (i.e. different tasks and number of vehicles) in order to observe the behavior of the non-optimal centralized planner using the SLS algorithm. Compare the performance of the system with the performance of the optimal centralized planner working in the same conditions (task sets).

Algorithm	Optimal Centralized Coordination		Non-Optimal Centralized Coordination (SLS)	
Number of tasks for two agents	Cost	Time	Cost	Time
6	107190	841ms.	104655	698ms.
7	128985	1543ms.	128985	1632ms.
8	161325	6800ms.	161325	2565ms.
9	175275	72805ms.	175275	3317ms.
10	181725	334644ms.	179190	4866ms.

Algorithm	Optimal Centralized Coordination		Non-Optimal Centralized Coordination (SLS)	
Number of tasks for three agents	Cost	Time	Cost	Time
6	91020	847ms.	1313	100ms.
7	113430	4019ms.	112815	1134ms.
8	145770	20678ms.	147390	2020ms.
9	159720	179156ms.	157185	1781ms.
10	…	…	165255	2973ms.

4. Can optimal planner with simplifying assumptions produce worse plan than the non-optimal planner which does not use the simplifying assumption that vehicle can carry only one task at the time? Can you find a task set for which this can be observed?

   Yes the optimal planner can produce worse plan than the non-optimal planner due to the simplifying assumptions as we can see in the comparing tables above.