

# Implementing centralized coordination

Intelligent Agents Course

# Pickup and Delivery Problem

- A company owns  $V$  vehicles
- Initially there are  $T$  tasks that need to be delivered
- Our goal is to build a plan for delivering all the packages with the available vehicles
- The total weight of tasks carried in each moment should not exceed the capacity of the vehicle
- Two variations
  - The vehicle can carry only one task at a time
  - The vehicle can carry more than one task at a time

# Approach

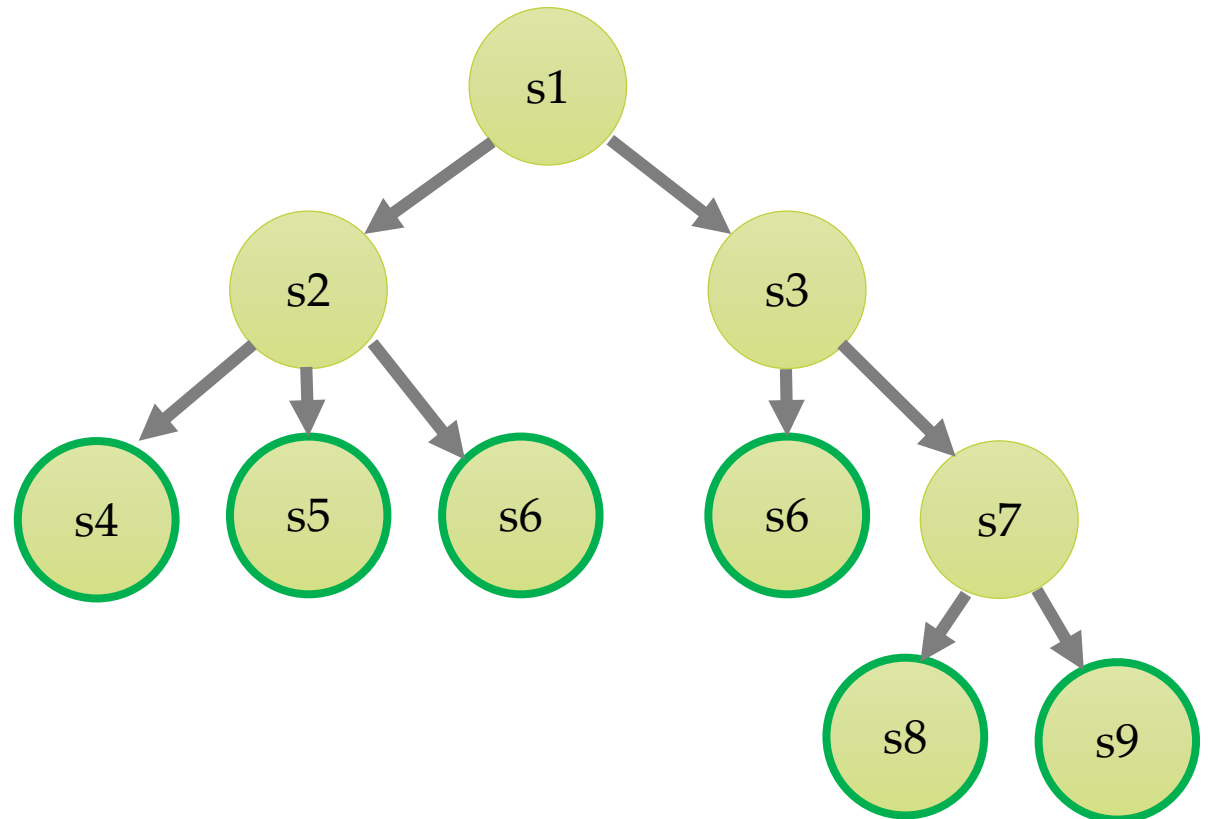
- We can assume that the company is deliberative agent, however, this is very inefficient approach
- Vehicles need to coordinate actions to achieve common goals
- The company can implement centralized coordination and build an optimal plan for delivering all tasks using all vehicles
- Vehicles execute the plan assigned to them

# How to build the plan

- Central planner needs complete information about the vehicles (positions, costs, capacity)
- State-based algorithms are not adequate:
  - Would have a too large number of states
- Instead, the problem should be solved using a Stochastic Local Planner.

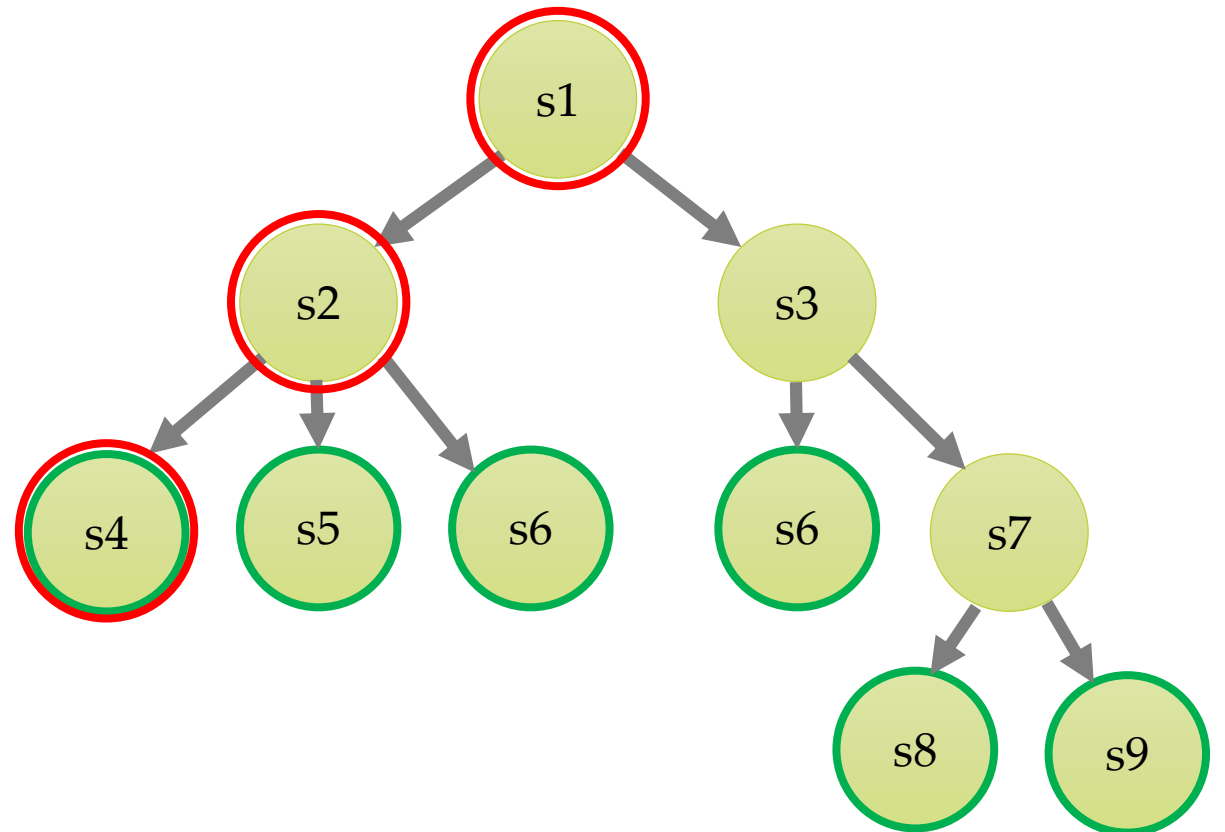
# Deliberative agent vs. centralized coordination

- Deliberative agent
  - Searches in state-space



# Deliberative agent vs. centralized coordination

- Centralized coordination
  - Searches in solution-space



# The goal of this exercise

- Implement the local search algorithm "Stochastic Local Search" to solve a COP description of the PDP, **allow a vehicle to carry multiple packages.**

# Constraint optimization problem

A discrete constraint optimization problem (COP) is a tuple  $\langle X, D, C, f \rangle$  where:

- $X = \{x_1, \dots, x_n\}$  is a set of variables.
- $D = \{d_1, \dots, d_n\}$  is a set of domains of the variables, each given as a finite set of possible values.
- $C = \{c_1, \dots, c_p\}$  is a set of constraints, where a constraint  $c_i$  is a function  $d_{i1} \times \dots \times d_{il} \rightarrow \{0, 1\}$  that returns 1 if the value combination is allowed and 0 if it is not.
- $f : d_1 \times \dots \times d_n \rightarrow \mathbb{R}$  is the objective function that we want to minimize (or maximize).

The optimal solution of a COP is an assignment of values to all variables that satisfies all constraints and minimizes the objective function.



# Stochastic Local Search Algorithm

---

**Algorithm 1** *SLS algorithm for COP*

---

**procedure** *SLS*( $X, D, C, f$ )

$A \leftarrow \text{SelectInitialSolution}(X, D, C, f)$

**repeat**

$A^{old} \leftarrow A$

$N \leftarrow \text{ChooseNeighbours}(A^{old}, X, D, C, f)$

$A \leftarrow \text{LocalChoice}(N, f)$

**until** termination condition met

**return**  $A$

**end procedure**

---

# Example solution

- We present solution of the pickup and delivery problem when the vehicle can carry only one task at a time
  - Vehicles carry tasks sequentially (this restriction lifted for SLS)
  - Total revenue of the company is maximized
- Solution representation

Vehicle 1: Task 1 Task 3 Task 4

Vehicle 2: Task 2 Task 5

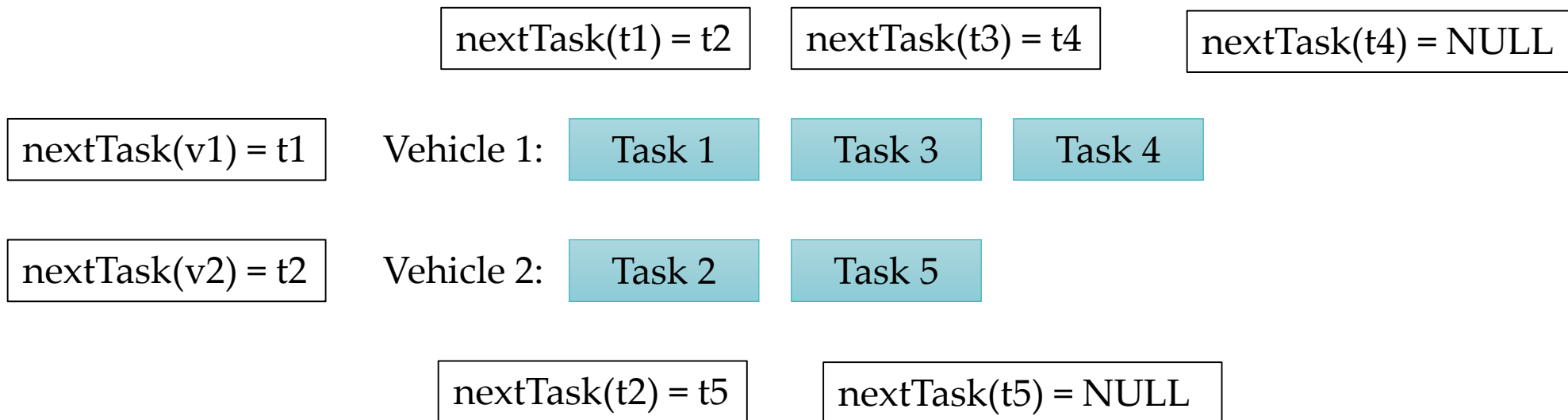
Each task is assigned to one vehicle

A vehicle cannot carry a task whose weight exceeds his capacity

The order of delivery is important

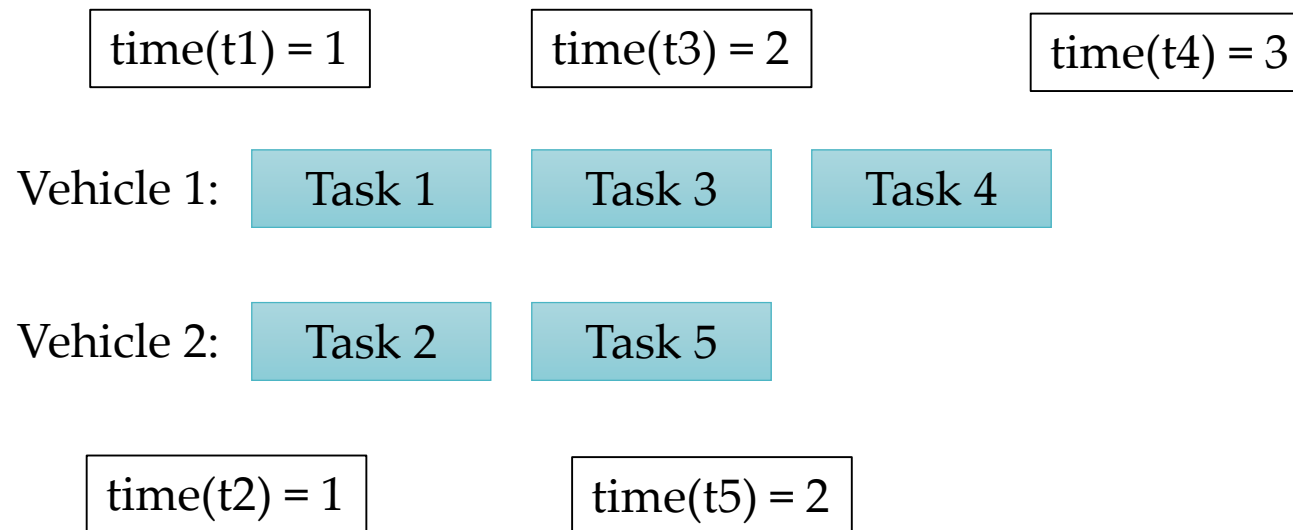
# Encoding the solution

- Three types of variables
  - **nextTask**: array of  $T + V$  variables
  - time: array of  $T$  variables
  - vehicle: array of  $T$  variables



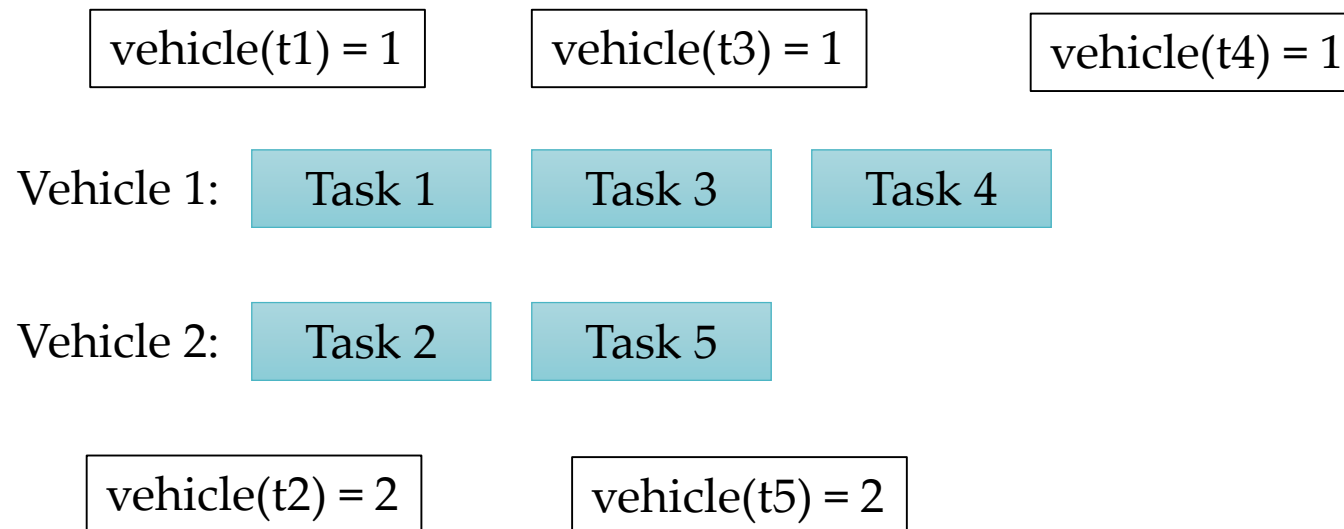
# Encoding the solution

- Three types of variables
  - nextTask: array of  $T + V$  variables
  - **time: array of  $T$  variables**
  - vehicle: array of  $T$  variables



# Encoding the solution

- Three types of variables
  - nextTask: array of  $T + V$  variables
  - time: array of  $T$  variables
  - **vehicle: array of  $T$  variables**



# PDP as COP - Constraints

1.  $nextTask(t) \neq t$ : the task delivered after some task  $t$  cannot be the same task;
2.  $nextTask(v_k) = t_j \Rightarrow time(t_j) = 1$ : already explained;
3.  $nextTask(t_i) = t_j \Rightarrow time(t_j) = time(t_i) + 1$ : already explained;
4.  $nextTask(v_k) = t_j \Rightarrow vehicle(t_j) = v_k$ : already explained;
5.  $nextTask(t_i) = t_j \Rightarrow vehicle(t_j) = vehicle(t_i)$ : already explained;
6. all tasks must be delivered: the set of values of the variables in the  $nextTask$  array must be equal to the set of tasks  $T$  plus  $N_V$  times the value  $NULL$ .
7. the capacity of a vehicle cannot be exceeded: if  $load(t_i) > capacity(v_k) \Rightarrow vehicle(t_i) \neq v_k$

# PDP as COP – Objective function

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

# Example solution

- 2 vehicles located in Zurich and Lausanne
- 3 tasks:
  - `<Task id="T1" load="10" pickup="Geneva" delivery="Fribourg"/>`
  - `<Task id="T2" load="15" pickup="Bern" delivery="Basel"/>`
  - `<Task id="T3" load="10" pickup="Aarau" delivery="St-Gallen"/>`
- Optimal plan:
  - the vehicle v1 delivers the task T3, and
  - the vehicle v2 delivers the tasks T1 and T2 in this order



# Example solution

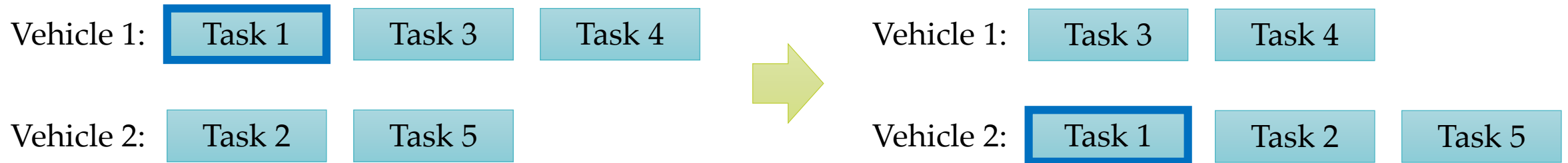
- $nextTask(v1) = T3, nextTask(v2) = T1;$
- $nextTask(T1) = T2, nextTask(T2) = NULL, nextTask(T3) = NULL;$
- $time(T1) = 1, time(T2) = 2, time(T3) = 1;$
- $vehicle(T1) = v2, vehicle(T2) = v2, vehicle(T3) = v1$

# Solving the PDP

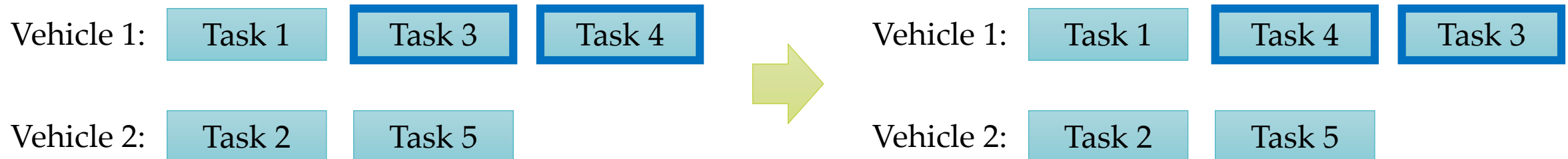
- Initial solution
  - Give all the tasks to the biggest vehicle
- Choose neighbors
  - Define a set of transformations that can be used to move from one to another close solution

# Transformations

- Move task from one to another vehicle



- Choose a vehicle and change the order of any pair of two tasks



# TO DO

- Implement the local search algorithm on PDP where each vehicle can carry more than one task at a time
- Use the stochastic planner :
  - Run simulations for different task sets