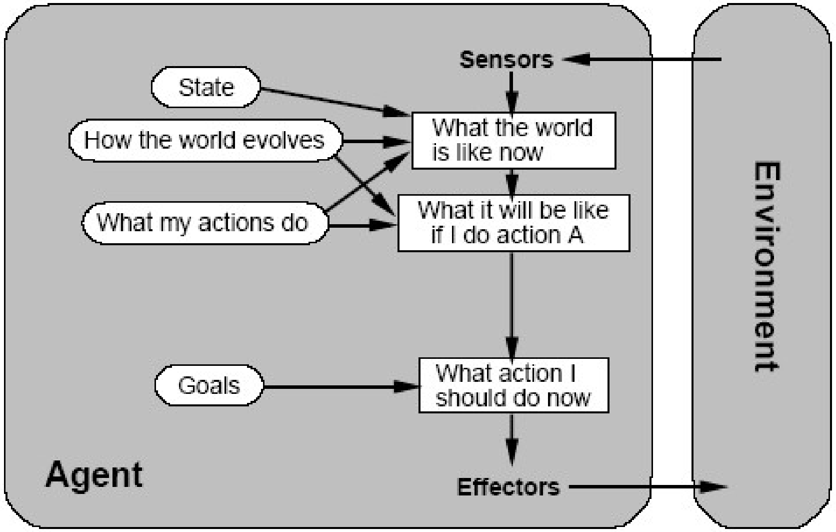
Implementing a Deliberative Agent

Intelligent Agents Course

(Credits to Radu Jurca
Michael Schumacher)

Deliberative Agents

- Also named ***mental*** or ***rational*** Agents
- The agent has an **explicit model of the world** in which it lives.
- Seeks to perform **goals**
- A **planner reasons** on the world model and decides which actions to realize by producing a plan, in order to achieve its goals.


Sensors
State
What the world is like now
How the world evolves
What my actions do
What it will be like if I do action A
Goals
What action I should do now
Agent
Effectors
Environment

Vehicles - Deliberative Agents

-> The vehicle computes an **optimal plan** using a state-based search algorithm

-> Rewards on goal states: an optimal plan to **minimize the cost**

- States of the world are known with certainty no probabilistic element to the project
- State transitions are deterministic

The Cost of a Plan

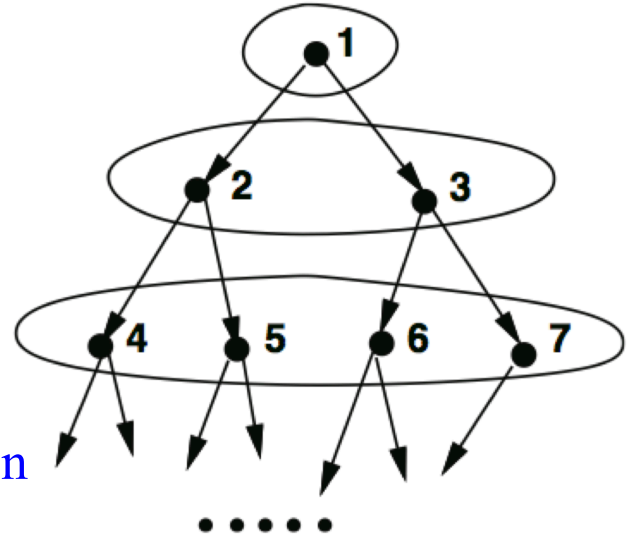
- Plan
  - Sequence of actions such that all tasks are delivered
  - Vehicle can carry multiple tasks
  - E.g. (T1,T2) ->
    - (move to pickup T1, pickup T1, move to delivery T1, deliver T1, move to pickup T2, pickup T2, move to deliver T2, deliver T2)
- Simplest cost form: distance x cost/km

Breadth-First Search Algorithm

there might be shared children
maybe 2,3 both have 3

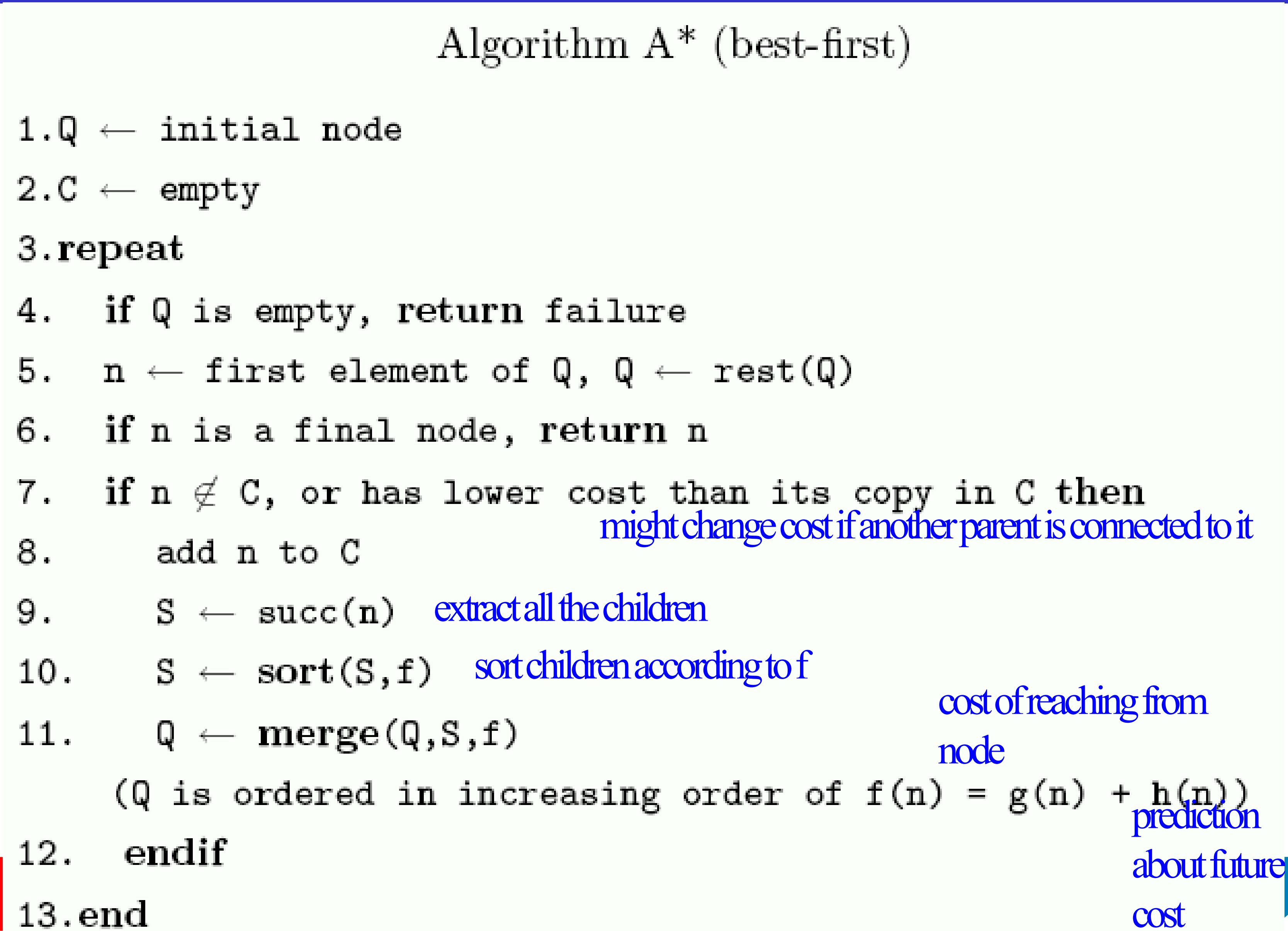
Search with cycle detection

```
Q ← initial node   Root state in the Queue
C ← empty
repeat
    if Q is empty, return failure
    n ← first element of Q , Q ← rest(Q)
  ifsi n is a leaf final node, return n
  if si n is not a member of C then
        add n to C
        S ← succ(n)   add to temp list S
      Q ← append(S, Q)   Keep visiting children
      endif
end
```



this algorithm should be modified so that it finds the optimal leaf node (lowest cost)

A* Search Algorithm



Algorithm A* (best-first)

```
Q ← initial node
C ← empty
repeat
    if Q is empty, return failure
    n ← first element of Q, Q ← rest(Q)
    if n is a final node, return n
    if n $\notin$ C, or has lower cost than its copy in C then
        add n to C
        S ← succ(n)
        S ← sort(S,f)
        Q ← merge(Q,S,f)
    (Q is ordered in increasing order of f(n) = g(n) + h(n))
    endif
end
```

might change cost if another parent is connected to it

extract all the children

sort children according to f

cost of reaching from node

prediction about future cost

Implementing Vehicles

- As soon as a delivery city is reached: task is delivered delivering is not an action!

multiple agents are present

- Internal planning process:

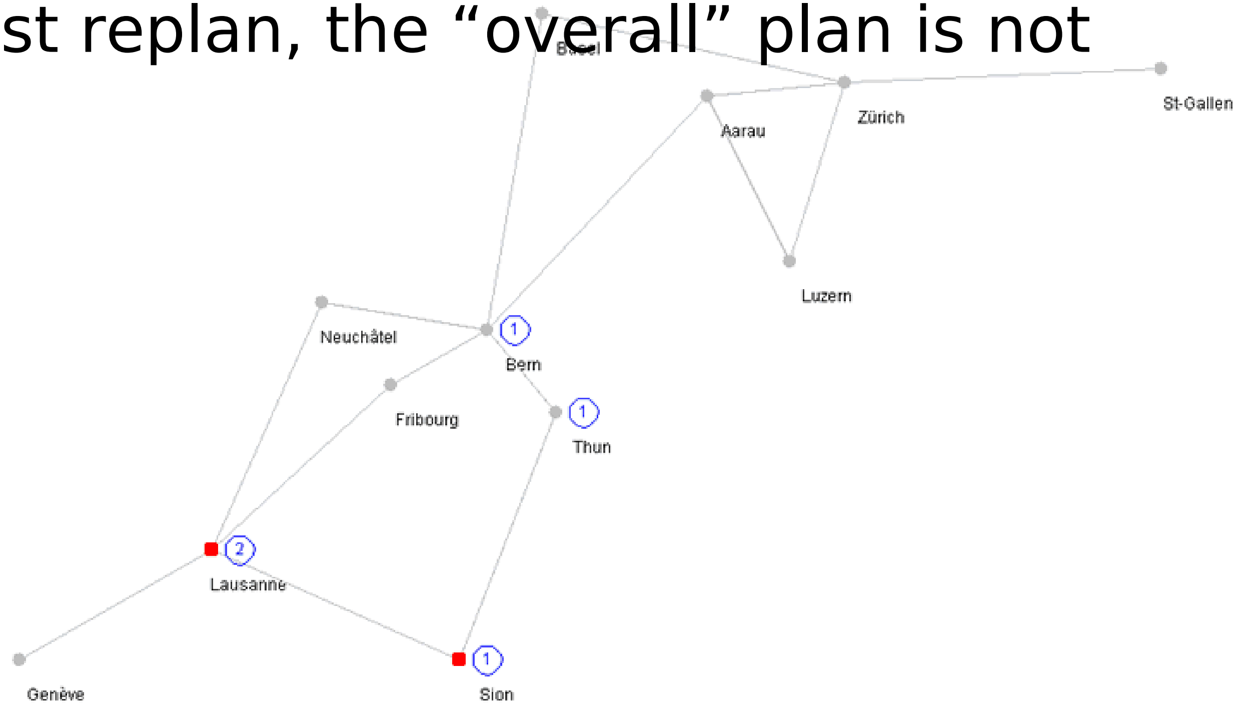
```
if (current plan is not applicable anymore) {
 then recompute plan
}
execute the plan's next action
```

if current plan is not applicable, re-compute another one with updated state of the world

-> recomputing a plan may be necessary ...

Delivery Example

- 1 vehicle is able to carry out the intended plan
- 2 vehicles: => interference
  - they must replan, the “overall” plan is not optimal



TO DO

- **Representation for the states**, transitions and goals (or final states)
- Implement breadth-first search and a **state-based A* search algorithm**
- Implement the **deliberative agent**
- Simulate **1, 2, 3** deliberative agents

Deliverable

- Check Moodle for deadlines
- 100 points