Sistemi Intelligenti
Corso di Laurea in Informatica, A.A. 2017-2018
Università degli Studi di Milano

Discrete planning (an introduction)

Nicola Basilico
Dipartimento di Informatica
Via Comelico 39/41 - 20135 Milano (MI)
Ufficio S242
nicola.basilico@unimi.it
+39 02.503.16294

Sito per queste lezioni

Dijkstra

- Recall: when x is selected, it becomes dead
- Thesis: when x is selected (and closed), we know its optimal cost-to-come
- Proof sketch: induction argument over the number of dead nodes (call it **d**) obtained after selection of x from Q

Base case:

d=1 then x is the initial state for which the optimal cost-to-come is known to be 0 (d=0 cannot happen from the definition of forward search)

Inductive step:

d=N (thesis holds for the previously selected N-1 dead nodes)

1. to reach x, we must visit a node currently in Q: **impossible**, their cost-to-come is higher than that of x and x would not have been selected
2. to reach x we visit only dead nodes: **possible**, from inductive assumption we know that we got the optimal plan to each of those nodes

then C(x) is actually C*(x)

Dijkstra

- Why we need shortest paths to solve planning for feasibility?

- Because by solving for minimum cost plans we also solve for feasibility: if the algorithm computed a cost for a node x then we have a plan to reach it, otherwise the state cannot be reached

- Dijkstra runs in $O(|V|\ln|V|+|E|)$ with "clever" implementation of Q (Fibonacci heap)

- It's systematic

A*

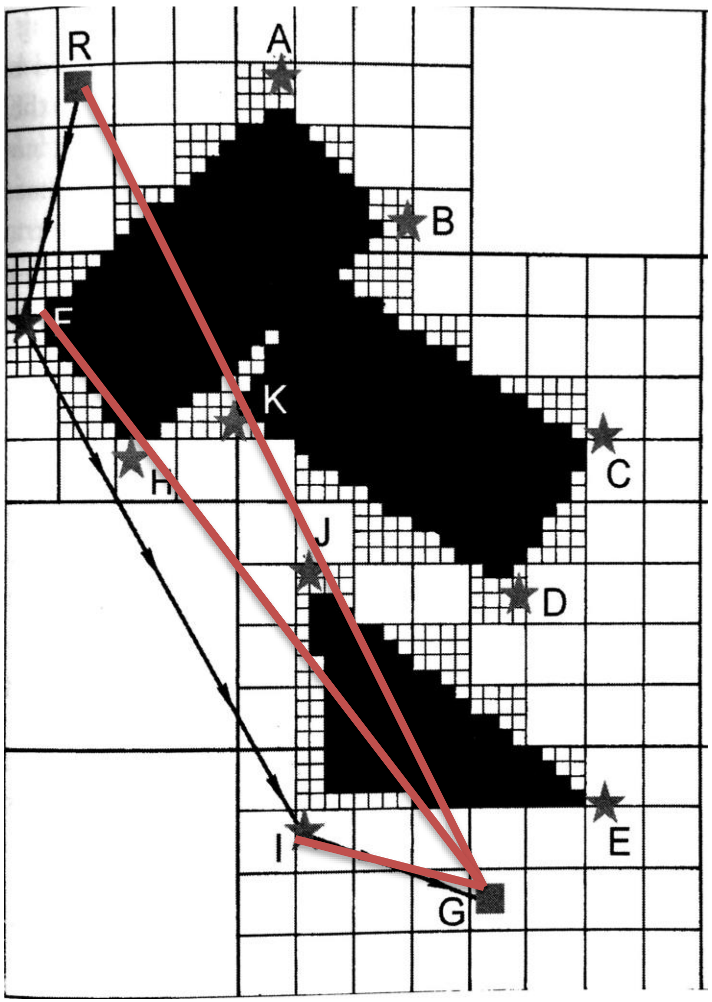
- It's a generalization of Dijkstra where the queue is sorted according to this function

$$C^*(x) + \hat{G}^*(x)$$

cost-to-come **estimate of the minimum cost-to-go**

- **cost-to-go**: the cost for going from x to the goal
- It guarantees to find the minimum cost plan (like Dijkstra) provided that we **do not overestimate** the cost-to-go (see previous proof sketch)
- If we set $\hat{G}^*(x) = 0$ we obtain Dijsktra
- The better our estimates, the fewer nodes are visited w.r.t. Dijkstra
- It's systematic

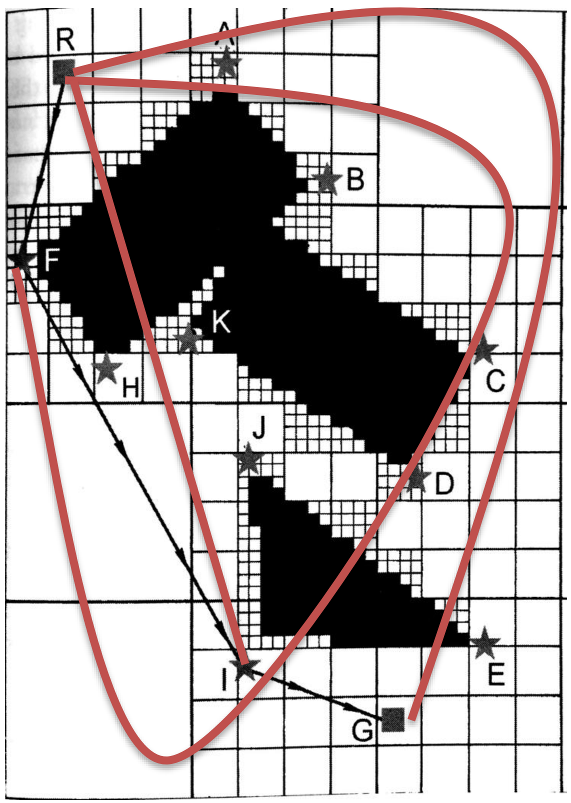
A*



Not overestimating the cost-to-go (admissible heuristic)

Best first

- Q is sorted using only the estimate of the **cost-to-go**
- Does not guarantee minimum cost plans so… it doesn't matter if costs are overestimated!



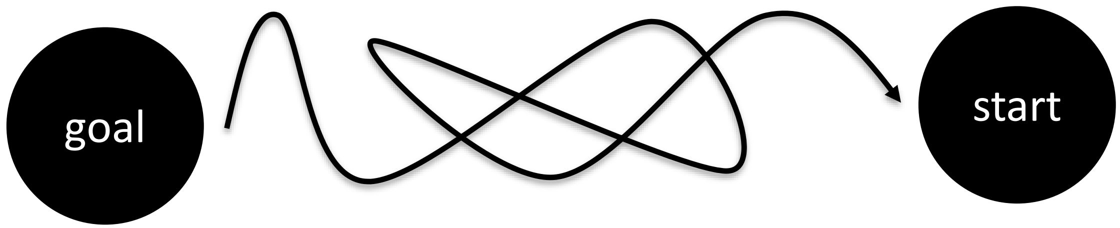
- Way faster and efficient
- If something looks good, even very early, it will take it: too greedy!
- Not systematic

Iterative deepening and IDA*

- Idea: try to make depth-first search systematic
- Straightforward approach:
  - use DFS to find all plans with length *<=i*
  - if goal was not found, *i++* and repeat
- Usually more efficient than BFS (especially with large branching factors): if the nearest goal is *i* hops away from the initial state, in the worst case BFS could try all nodes at *i+1* hops first
- IDA* naturally follows when applying this idea to A* by introducing **allowed total costs**

Backward search

- Symmetric template of forward search:



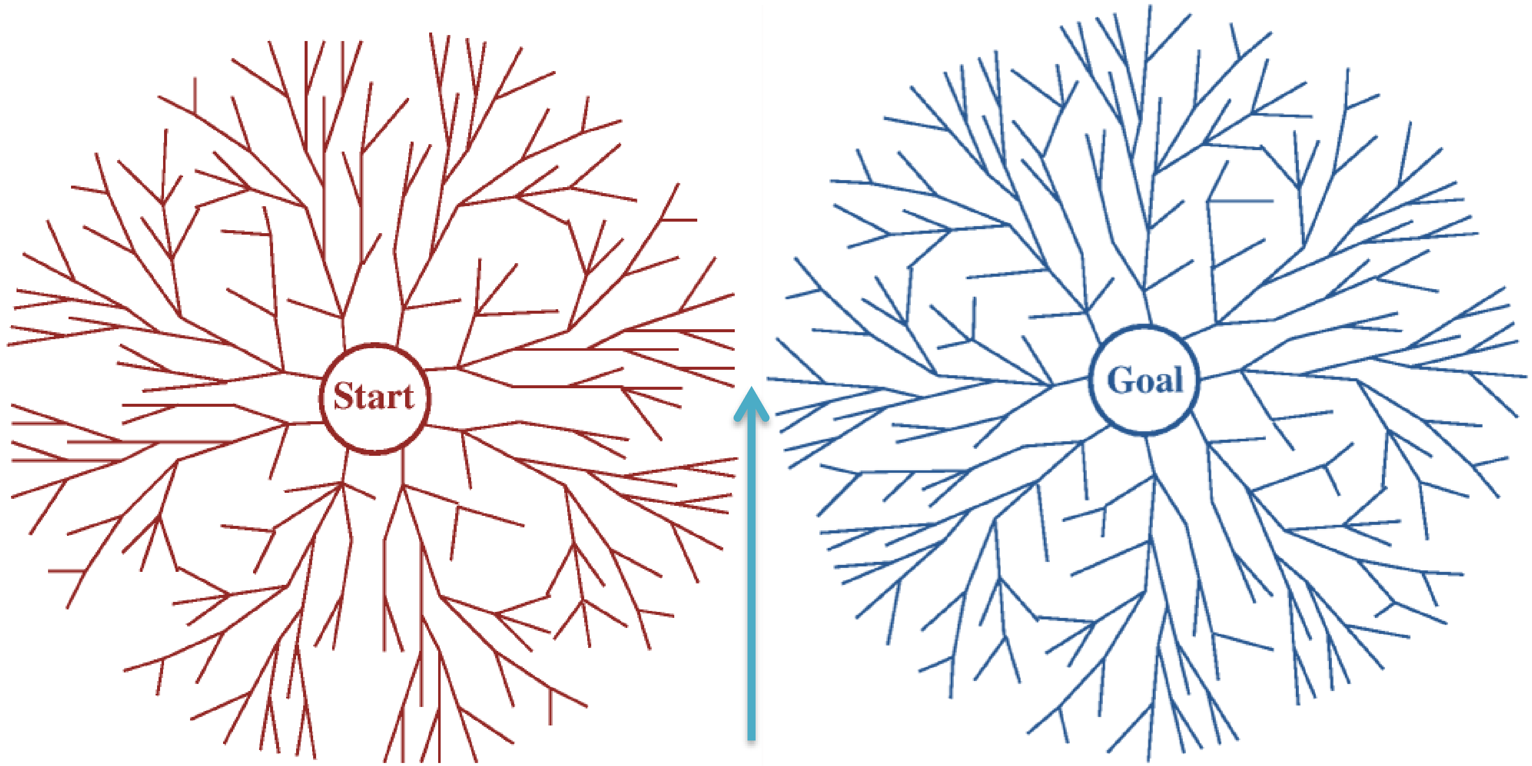
- We can just use forward search on the state transition graph where we reversed the arcs
- Useful when branching is very high when starting from x_I

Backward search (example?)

				2	8		7	
			3					8
		8			1			4
	4					7		6
	8		7	5	6		4	
5		7					1	
9			8			6		
8					9			
	2		5	4				

Bidirectional search

- Template combining forward and backward search:



what's happening here?

- We need to grow the two trees so that they tend to meet quickly. It can be difficult.

Planning for optimality

- Let us extend the formulation we presented for the feasibility version
- Call K the length of a plan, and x_{k+1} the state reached when action u_k from the plan is applied

This plan of K actions $\pi_K = (u_1, u_2, \ldots, u_K)$

causes the following sequence of states $x_I \rightarrow x_2 \rightarrow x_3 \rightarrow \ldots \rightarrow x_{K+1}$

which we might relabel for convenience $x_1 \rightarrow x_2 \rightarrow x_3 \rightarrow \ldots \rightarrow x_F$

This plan has a cost: $$L(\pi_K) = \sum_{i=1}^{K} l(u_i, x_i) + \boxed{l_F(x_F)}$$

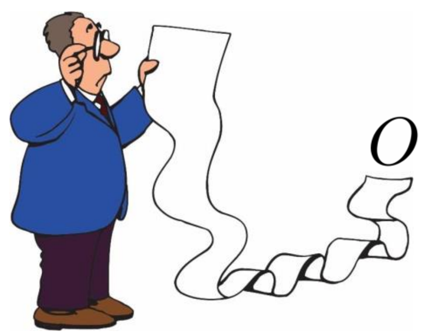
∞ X_F is not a goal state

0 X_F is a goal state

- We want to minimize the cost of a plan
- With this definition of plan cost we enforce feasibility through optimization: seeking for optimal plans entails seeking also for feasible ones

Fixed length plans

- Problem: find the optimal (minimum cost) plan which has length upper bounded by K
- Idea: let’s make a list of all the <=K length plans and evaluate their cost, then choose the best

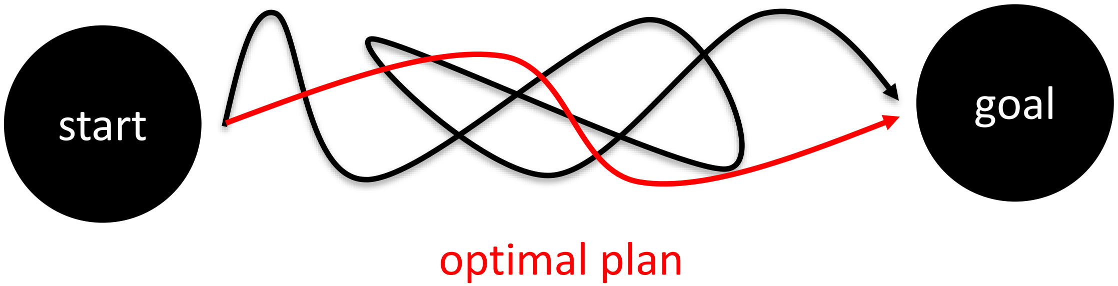


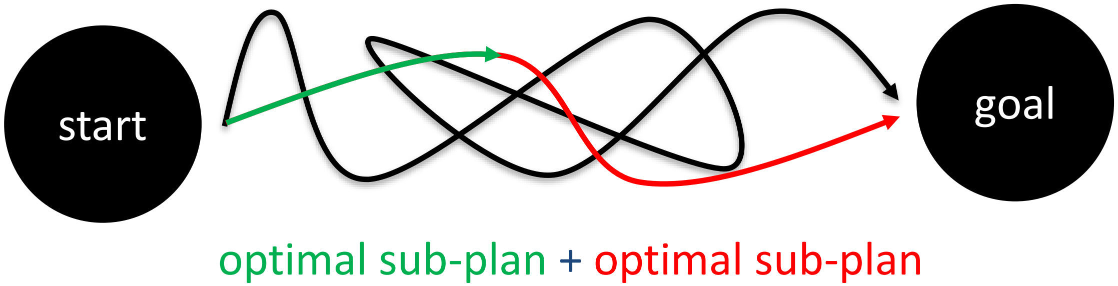
$$O(|U|^K)$$

We don’t have to do this thanks to **Bellman’s principle of optimality**

Principle of optimality

- For some problems optimality admits a recursive definition





- Optimal substructure: construct an optimal solution for P exploiting optimal solutions of sub-problems of P
- This principle leads to a resolution method called **value iteration**

Backward value iteration

- Idea: compute cost-to-go backwards

$$G_k^*(x_k) := \min_{u_k,\ldots,u_K} \left\{ \sum_{i=k}^{K} l(u_i, x_i) + l_F(x_F) \right\}$$

Cost from x_k to x_F along the optimal plan

$$G_F^*(x_F) = l_F(x_F)$$

Backward value iteration

$$G_F^* \rightarrow G_K^* \rightarrow G_{K-1}^* \rightarrow \ldots \rightarrow G_1^*$$

Backward value iteration

$$G_F^* \rightarrow G_K^* \rightarrow G_{K-1}^* \rightarrow \ldots \rightarrow G_1^*$$

$$G_F^*(x_F) = l_F(x_F)$$

optimal cost-to-go for any state if final or …

… optimal cost for the empty plan

Backward value iteration

$$G_F^* \rightarrow G_K^* \rightarrow G_{K-1}^* \rightarrow \ldots \rightarrow G_1^*$$

$$G_F^*(x_F) = l_F(x_F)$$

$$G_K^*(x_K) = \min_{u_K} \left\{ l(u_K, x_K) + G_F^*(f(x_K, u_K)) \right\}$$

optimal cost-to-go for any state when being right before the final one or …
.. optimal cost of the plan with length 1

We can recursively propagate this reasoning scheme obtaining …

Backward value iteration

$$G_k^*(x_k) = \min_{u_k} \left\{ l(x_k, u_k) + \boxed{\min_{u_{k+1},\ldots,u_K} \left\{ \sum_{i=k+1}^{K} l(x_i, u_i) + l_F(x_F) \right\}} \right\}$$

optimal sub-plan from x_{k+1}

$$G_k^*(x_k) = \min_{u_k} \left\{ l(x_k, u_k) + \boxed{G_{k+1}^*(x_{k+1})} \right\}$$

If we "unroll" the execution of this recursive procedure we obtain:

$$G_F^* \rightarrow G_K^* \rightarrow G_{K-1}^* \rightarrow \ldots \rightarrow G_1^*$$

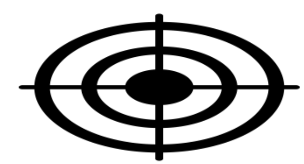
in O(K|X||U|), G^*_1 gives the cost of the optimal plan, the actual plan can be obtained by backward annotation of *argmins*

Plans of arbitrary length

- Remove the length limit and add a termination action a_t
- If applied it does not change the current state, does not cause any cost and must be applied forever from there on
- We stop our backward value iteration when it converges, i.e., G does not change anymore
- Ok only if costs are non-negative

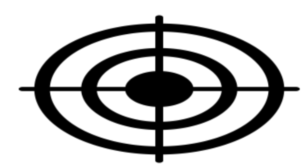
Planning under uncertainty

- Action selection is often affected by uncertainty
- Example:



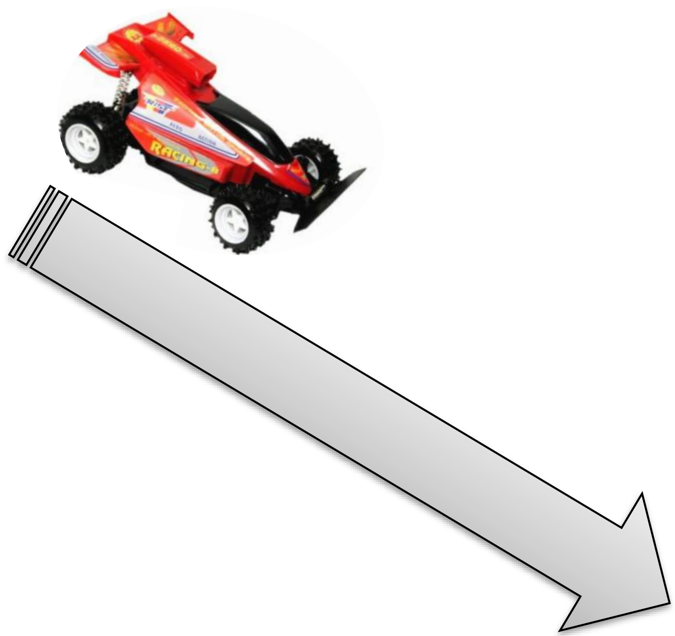
Planning under uncertainty

- Action selection is often affected by uncertainty
- Example:



Planning under uncertainty

- Action selection is often affected by uncertainty
- Example:



Fast forward for 2 seconds

Planning under uncertainty

- Action selection is often affected by uncertainty
- Example:

Fast forward for 2 seconds

Planning under uncertainty

Are the effects of
my actions
perfectly
predictable?

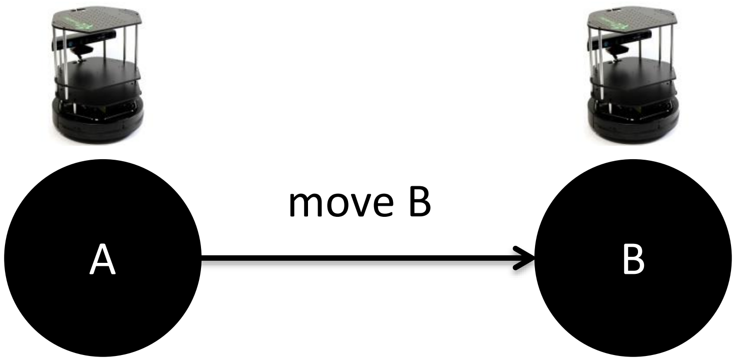
Am I always sure
about what's
going on?

Deterministic
vs
Stochastic
transitions

Fully observable
vs
Partially observable
states

Examples

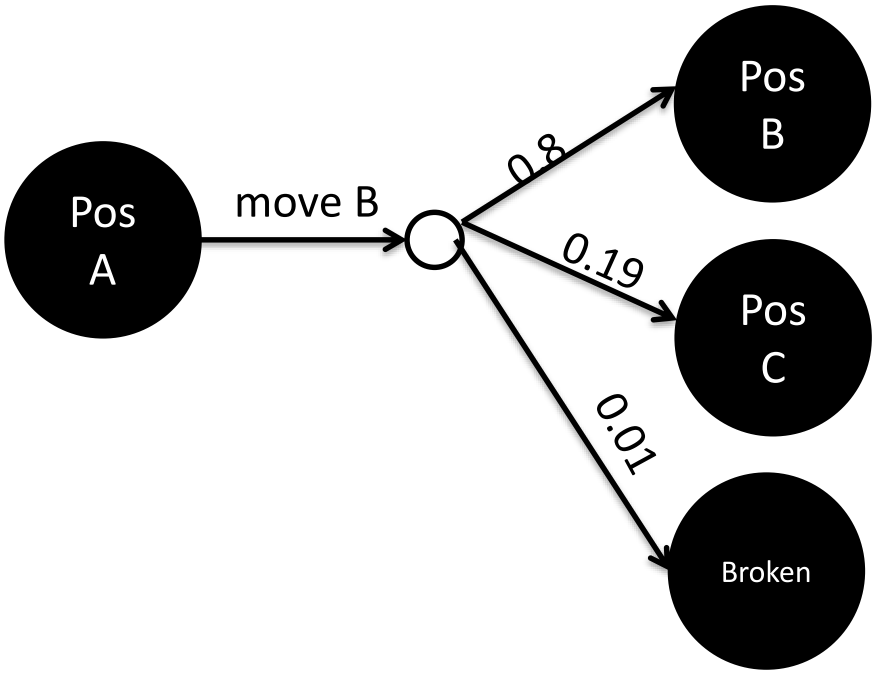
- Deterministic transitions, fully observable states
- Only actuation is needed, no sensing!

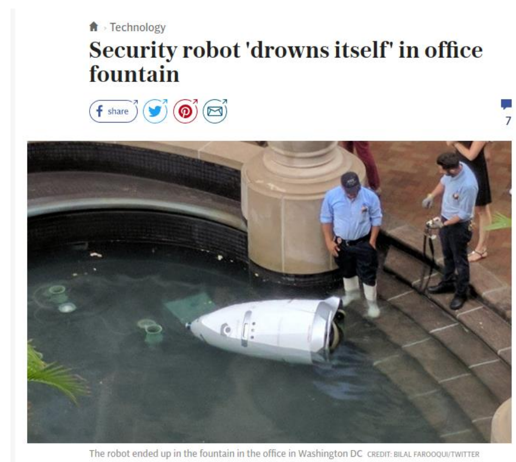


Examples

- Stochastic transitions, fully observable states

Surveillance robot





Technology

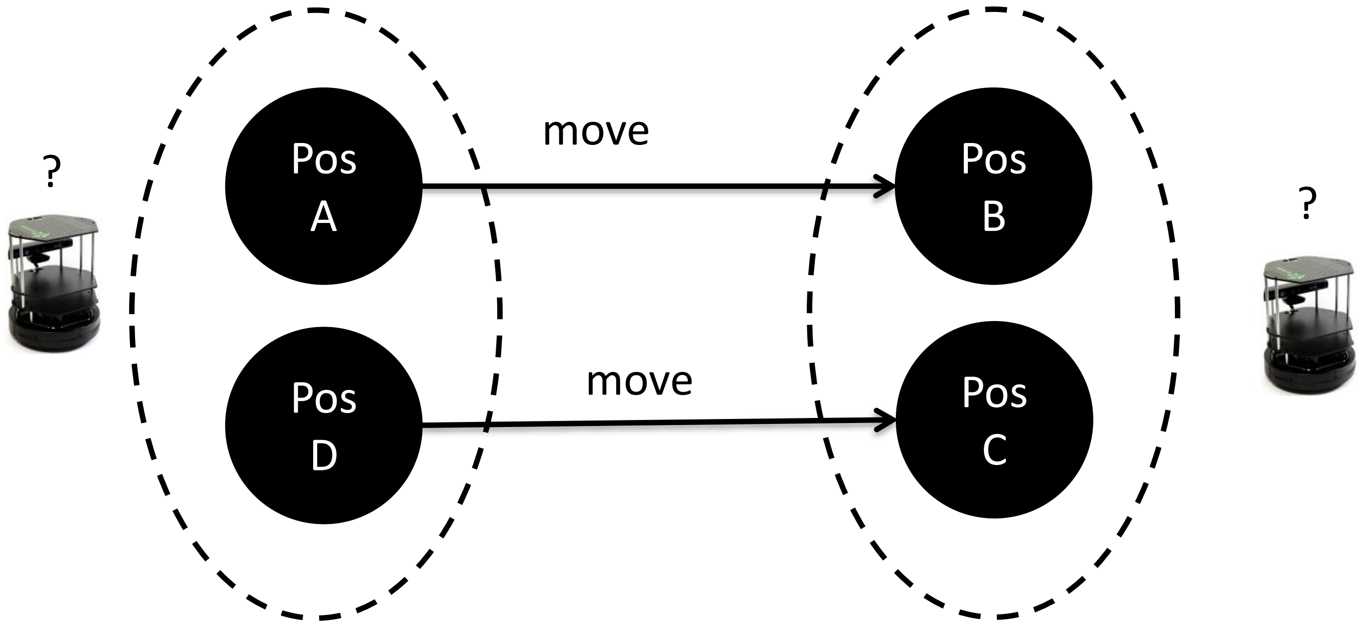
Security robot 'drowns itself' in office fountain

share

7

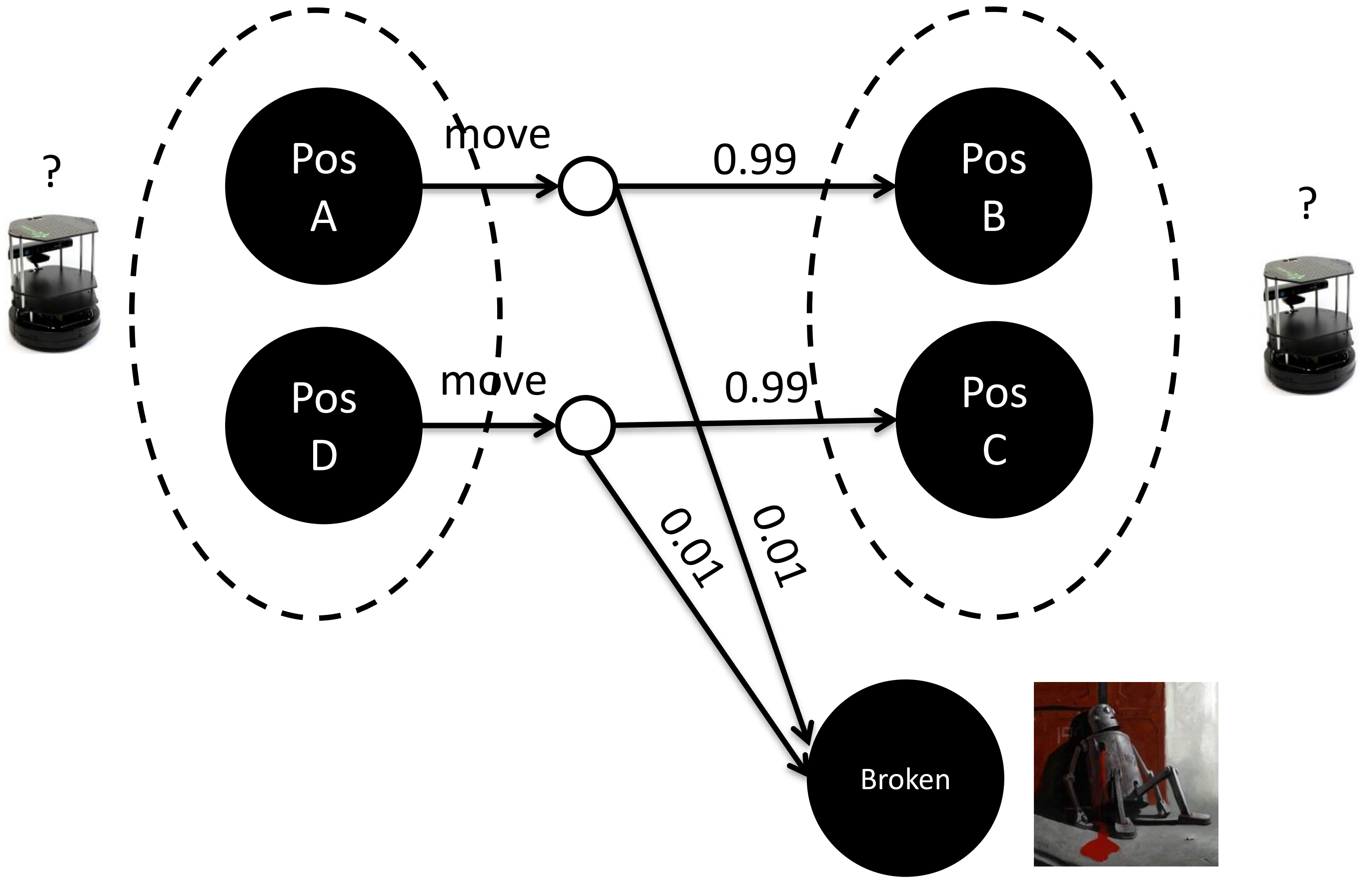
The robot ended up in the fountain in the office in Washington DC CREDIT: BILAL FAROOQUI/TWITTER

Examples



Deterministic transitions, partially observable states

Examples



Stochastic transitions, partially observable states

Sistemi Intelligenti
Corso di Laurea in Informatica, A.A. 2017-2018
Università degli Studi di Milano

Nicola Basilico
Dipartimento di Informatica
Via Comelico 39/41 - 20135 Milano (MI)
Ufficio S242
nicola.basilico@unimi.it
+39 02.503.16294